# INTERDICTION AND DISCRETE BILEVEL LINEAR PROGRAMMING

by

Scott DeNegre

Presented to the Graduate and Research Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy

in
Industrial and Systems Engineering

Lehigh University

May 2011

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
Lehigh University

Dissertation Advisor

Accepted Date

Committee:

Dr. Theodore K. Ralphs
Lehigh University, Chairman

Dr. Jeffrey T. Linderoth
Lehigh University

Dr. Aurélie C. Thiele
Lehigh University

Dr. Michael C. Ferris
University of Wisconsin-Madison

# Acknowledgments

First and foremost, I would like to thank my advisor, Ted Ralphs. Our collaboration over the last several years has been an outstanding experience, both academically and personally, and I have benefited greatly from his expertise, careful guidance, and friendship. He truly epitomizes the paradigm of lead by example, and I very much look forward to our continued work together.

I would also like to express a debt of gratitude to my committee members, Jeff Linderoth, Aurélie Thiele, and Michael Ferris, for their invaluable insight and perspective throughout the course of this research. In addition, I would like to thank my friends and colleagues at École Polytechnique Federal de Lausanne (EFPL). A significant portion of this research was completed during my time as a visiting researcher in the mathematics department at, where I had the opportunity to work closely with Thomas Liebling and Alain Prodon, from whom I learned a great deal.

During my tenure at Lehigh, I have acquired some very close friends and colleagues. In particular, I am sincerely grateful to have gotten the chance to know and work with Mike Dziecichowicz, Dave Grace, Owen Finnegan, Aleks Tomic, Alper Uygur, and my fellow COR@L members Kumar Abhishek, Menal Guzelsoy, Ashutosh Mahajan, and Jim Ostrowski.

I would like to thank my family, John, Susan, Ashley, and Ryan. As always, their unwavering love and support was a huge factor in my success during this process. Finally, I would like to thank my girlfriend, Becca, who put up with many long hours, late nights, and bad attitudes during the final stages of this work, and responded only with patience and love.

# Contents

# List of Tables

# List of Figures

# Abstract

The primary focus of this dissertation is on hierarchical decision problems, a general problem class that allows incorporation of multiple decision-makers (DMs). A variety of real-world problems involve DMs with potentially conflicting objectives, and the assumption of a single DM limits the utility of standard models for such applications. In particular, we study problems with two levels for which a subset of the variables is required to take on integer values.

In mathematical programming terminology, this problem is formalized as mixed integer bilevel program (MIBLP), and the variables are divided into groups defined by their controlling DM. A key component of these models is the dependence of the lower-level DM's feasible region on the upper-level solution. From this perspective, an MIBLP can be viewed as a mixed integer linear program (MILP) into which a second parametric MILP has been embedded. We focus our study on the theoretical properties of MIBLPs, in order to determine how its structure can be exploited for algorithm design. In addition, because of the computational challenges the general problem presents, we examine special cases that are more amenable to algorithmic development.

The first such case is that of the pure integer bilevel linear program (IBLP). In the first portion of this work, we develop a branch-and-cut framework and an accompanying open source solver, MibS, for this problem class. Our algorithm can be seen as a generalization of the well-known branch-and-cut algorithm for MILP, but invokes specialized cutting planes to separate solutions that satisfy integrality constraints, but are bilevel infeasible.

After developing our pure integer framework, we return to the general case and examine its computational complexity and place it within the so-called polynomial hierarchy. Next, we examine the extent to which methods developed for the well-studied continuous version of the problem (BLP) can be extended to MIBLP. The majority of BLP solution methods rely on the assumption that all decision variables are continuous and, thus, cannot be readily applied to the mixed integer case. However, in an effort to bridge this gap, we use intuition gained from studying the relationship between linear programs (LPs) and MILPs. In particular, we draw heavily on the recently-developed mixed integer extensions of LP duality theory to develop single-level reformulations of MIBLP. For some particular special cases, these methods yield problems to which known methods can be

applied, but the general reformulation requires the application of the subadditive dual, and cannot solved directly. In order to overcome this, we use approximations of the lower-level value function to derive an exact algorithm reminiscent of Benders' decomposition and the integer L-shaped method. The inherent difficulty of these problem means that finding exact solutions to large instances will likely be prohibitively expensive. Thus, we provide two heuristic methods, each of which attempts to balance upper- and lower-level optimality, that can be used to be find good solutions to general problems with little computational effort.

In the final section of this dissertation, we study an application in critical infrastructure protection, namely that of designing an early warning system to monitor the structural integrity of a municipal water system. The Steiner arborescence problem used to determine the optimal placement of acoustic sensors within the system is described, and a novel cutting plane algorithm is presented. Then, using this model as illustrative example, we demonstrate the utility of interdiction problems in performing a type of systematic sensitivity analysis of our optimal design to the underlying graph structure. Interdiction problems, a class of MIBLPs used to model the effect that can be exerted on an MILP through variable bound altercation are of particular interest in our work for a number of reasons, most notably their applicability for problems in homeland security and unique problem structure. We describe several methods based on this special structure and show how one might develop a problem-specific customization for MibS.

# Chapter 1

# Introduction

In this dissertation, we study the theoretical properties of hierarchical decision models, a class of decision problems with rich application potential. Many real-world decision problems involve multiple, independent decision-makers (DMs), whose interests are not necessarily aligned, and models that assume centralized control are limiting in such settings. A hierarchical model, however, is comprised of several levels of DMs, whose decisions are made sequentially and may affect the options available to those lower in the hierarchy and the payoff of those higher in the hierarchy.

A common example of such a model is that faced by the federal government. Policy decisions made at the federal level affect future decisions made by state and local governments, each of which acts in its own self-interest in reaction to federal directives. Decisions made by the state and local governments, in turn, affect the degree to which the federal government accomplishes its original objective. Thus, in order to perform an accurate analysis, the federal government must consider the reaction of the lower-level bodies, and make policy decisions accordingly. The same analysis applies in the corporate setting, where company policy is set at the highest level and interpreted and applied in smaller organizational units.

Such a modeling framework also provides a natural representation of single-round (or static) *Stackelberg games*. Stackelberg games, first introduced by von Stackelberg (1934), provide the game-theoretic foundation for modeling the behavior of economic markets and resource competition (Senn, 1996). As in a hierarchical decision model, the defining characteristic of a Stackelberg game is its sequential nature. Traditionally, a Stackelberg game is played over several rounds, where each player selects a new strategy at each iteration. The game is continued until an equilibrium is reached, where no player can improve his situation by changing his strategy, or a specified number of rounds has been played. If the game consists of a single-round, then it is known as a *static Stackelberg game*. These games are often analyzed from the perspective of the first player, whose goal is to choose the optimal strategy, in light of the expected behavior of his competitors. Viewed from this

perspective, we can see close connections between hierarchical decision models and *algorithmic game theory* (see, e.g., Nisan et al., 2007), an emerging subfield of game theory concerned with methods for computing equilibria.

Of particular interest in this research are applications in homeland security and critical infrastructure protection. Traditionally, standard mathematical programming frameworks, such as linear and mixed integer linear programming, have been the modeling frameworks of choice for these disciplines, and can be used to model a variety of problems facing the decision-makers charged with protecting private or public systems. In fact, one such class of problems to which we devote significant attention is that of early warning system design. Early warning systems are used to monitor critical infrastructure, in order maintain system stability, or recognize and react to a system disruption. Typically, these problems involve the installation of sensors that monitor a subset of the system and transmit data to a central hub for analysis. Thus, determining the optimal placement of the sensors within the system, in order to provide maximum coverage and, thus, maximum protection, becomes of immediate interest. Sensor technology varies widely from application to application, but the general idea remains relatively similar; placing sensors at strategic locations within the system allows us to monitor the health of the system, and react accordingly when (accidental or intentional) disruption occurs. A common example of this problem class is used to protect urban water systems via installation of sensors designed to detect contamination in the water network (e.g., Ostfeld and Salomons, 2004; Berry et al., 2005; Carr et al., 2006; Berry et al., 2006a,b; Krause et al., 2008).

While single-level modeling frameworks are appropriate for modeling a wide range of infrastructure protection problems, they are not suitable for all settings; one can easily imagine problems for which none of the well-known frameworks is appropriate, and modeling choice should be made carefully. For example, a natural question that arises in infrastructure protection is "how vulnerable is this system to disruption by an adversary?". Another interesting question is "how sensitive is the system design to the (potentially dynamic) system structure?". Answering these questions is difficult if one is limited to the traditional mathematical programming frameworks, due to the standard assumption that all decision variables are controlled by a single entity. However, each can be posed directly as an *interdiction problem*, a hierarchical model class that allows us to model the effect that an external entity can exert via delay or disruption of the system. Here, the adversarial nature of the hierarchy members results in problems for which the objective functions are in direct opposition (i.e., zero-sum).

The majority of research on interdiction models has focused on the *network interdiction problem* (Wollmer, 1964; McMasters and Mustin, 1970; Ghare et al., 1971; Wood, 1993; Cormican et al., 1998; Israeli and Wood, 2002; Held and Woodruff, 2005; Janjarassuk and Linderoth, 2008), in which the lower-level decision-maker represents an entity operating a network of some sort. The upper-level decision-maker (or interdictor) attempts to reduce the network performance as much

as possible via the removal (complete or otherwise) of portions (subsets of arcs or nodes) of the network. Applications of these models are limited to problems for which the an underlying network structure can be assumed, but the range of application is much broader once one drops this assumption. In fact, the underutilization of this problem class provided the initial motivation for our study of hierarchical models. We study a generalization of these network interdiction models that incorporates the "interdiction" of lower-level decision variables in depth, and demonstrate its utility in sensitivity and systems analysis (see Chapter 4).

## 1.1 Modeling with Multiple Decision-makers

From a modeling perspective, traditional mathematical programs are limited by their underlying assumptions of a single DM and a single objective. Our interest in interdiction models led us to consider alternative extensions of linear and mixed integer linear programming that provide greater flexibility with respect to competing individuals or objectives. One way to overcome the latter limitation is with the framework of *Multiobjective programming*. Multiobjective programming is a generalization of traditional mathematical programming in which multiple, conflicting objective functions can be introduced, and enables the study of tradeoffs among the multiple objectives controlled by a single DM. A particularly relevant application of this method is given by Watson et al. (2004), who utilize multiobjective programming to generalize the mixed integer linear programming formulation of sensor location optimization problem of Berry et al. (2005) and study the tradeoff between multiple performance objectives.

While multiobjective programming relaxes the latter assumption of a single objective, it remains limited by the former assumption of a single DM. This limiting assumption prevents us from accurately capturing the interactions among different DMs. Clearly, the implications of this limitation are of particular concern for adversarial problems, a class likely to be encountered in problems of homeland security.

The framework of *multilevel programming*, on the other hand, allows us to model these more general decision problems. In a multilevel program, the variables are divided into groups, each of which is controlled by a different DM. Under the assumptions of *perfect information* and *individual rationality* of the DMs, the higher-level DMs will be able to predict the reaction of lower-level DMs to decisions made above them. In this context, the assumption of individual rationality implies that each DM will choose the best solution with respect to a given objective, subject to a set of specified constraints. That is, each DM will solve a mathematical program to optimality. Here, perfect information means that each lower-level DM is aware of the actions taken by those above him. Further, this assumption implies that each DM is aware of the parameters defining the mathematical programs to be solved at lower levels of the decision hierarchy. This allows us to collapse

the entire hierarchy into a single optimization model in which the decisions made at the highest level effectively determine the outcome for the entire system. The broader focus of this dissertation is on the theoretical properties and the resulting algorithmic implications of multilevel programs, a modeling class that subsumes interdiction models, and formalizes the hierarchical decision models in a mathematical programming setting. In particular, we focus on techniques for analyzing *mixed integer bilevel linear programs* (MIBLPs), in which (1) there are two DMs, (2) the constraints are linear functions, and (3) a subset of the variables may be required to take on integer values.

## 1.2 Applications of Bilevel Programming

We have seen how one subclass of bilevel programming can be used to model problems involving adversarial DMs. More generally, bilevel models are extremely useful for modeling systems designed by one entity, but controlled by another. In this case, the parties are not necessarily in opposition, but may still have different objectives. One example of such a system is that encountered in highway toll pricing. In a toll pricing problem, the system operator seeks to maximize the revenue obtained from tolls imposed on a local road system. The revenue gained depends directly on the decisions made by the system users (drivers), over whom the operator has no control. Thus, the operator must determine the toll prices under the assumption the users will maximize their own individual utilities (Labbè et al., 1998a,b). Of course, the applicability of the bilevel programming framework is not limited to highway toll pricing models, but can be applied to the more general problem of determining how one can can influence behavior through tariff imposition (Brotcorne et al., 2000).

As in the interdiction literature, this field of study has typically been limited to road systems that can be modeled as networks, thereby allowing convenient reformulations of the resulting bilevel program. However, the bilevel framework has also been applied, for example, to the problem of determining optimal tax credits for biofuel production (Bard et al., 1998, 2000). In such an application, the government provides tax credits to the petro-chemical industry to encourage increased production of biofuels from farm crops, a process that is typically more expensive than producing fuel from hydrocarbon-based raw materials. The government (leader) seeks to minimize the total amount of tax credits paid out, while incentivizing the agricultural sector (follower) to set aside a certain level of its land for nonfood crops to be used for biofuel production. Under the assumption that the agriculture industry is neutral to the type of crops produced as long as profit is maximized, the government can effectively set the prices paid by industry via the tax credit. Assumptions of continuous production variables have again limited the utility of bilevel programming in this application area. For example, Bard et al. (2000) describe an extension of their model in which the

petro-chemical industry may choose to produce all biofuels from nonfood crops. The new formulation requires binary variables in the upper-level problem. One could also imagine model extensions requiring discrete decisions in the lower-level. For example, if there exists a fixed cost for producing a particular type of crop, production would occur only if the profit outweighed the sum of the fixed and operational costs. In order to accurately model such a scenario, lower-level binary variables would be necessary.

The bilevel modeling paradigm can also be used to perform a wide range of worst-case analyses where the highest-level DM does not represent a true decision-making entity, but allows the inclusion into the model of circumstances that cannot be controlled, such as the weather or world events at large. The question to be answered in such cases is "what is the worst that can happen?". Taking the opposite point of view, the same paradigm can be used to analyze systems in which the upper-level DM is an individual trying to influence the course of some natural process that operates according to a principle of optimality because of the laws of physics, for instance (i.e., electricity travels by a path of least resistance). In this way, such models could be used, for example, by a participant in an electricity market to model the effect of changes in their own supply of electricity on power flows through the network in order to determine their optimal production level (Bienstock and Verma, 2008). A similar application arises in the biomedical field, where we can model the effect of opening and closing pathways of blood flow to the heart for the treatment of conditions such as atrial fibrillation. This application is discussed further in Chapter 5. We also see related applications of bilevel programming in the biotechnology literature. In one such application (see, e.g., Burgard et al., 2003; Pharkya et al., 2003), bilevel programming is used to determine optimal strategies for microbial strain engineering leading to increased production of chemicals or biochemicals. That is, by knocking out specific genes and, thus, prohibiting certain cellular reactions, one can develop microbial strains with improved production capability. Here, the lower-level DM is not a true decision-making entity, but rather is used to represent metabolic behavior, controlled by internal cellular objectives.

Finally, there are deep connections between bilevel programming and the decision framework that drives the well-studied branch-and-bound algorithm. Branch and bound is a "divide and conquer" approach to solving mathematical programs. Fundamentally, branch and bound is a method that enumerates the set of feasible solutions to a mathematical programming problem. To improve efficiency, a divide and conquer approach is used to eliminate portions of the feasible region by computing bounds on the optimal objective value. The feasible region is partitioned using branching methods based on logical disjunctions, and performance depends on both the quality of the bounds used and the effectiveness with which branching disjunctions are chosen. The problem of determining the disjunction whose imposition results in the largest bound improvement within a branch-and-bound framework based on disjunctive programming is itself a bilevel program (Lodi

and Ralphs, 2009). Thus, the study of bilevel programs may lead to improved methods for solving single-level programs.

In recent years, bilevel programming has been recognized as an important field within mathematical programming, allowing the analysis of a much broader range of systems. However, in each of the fields described above, the utility of bilevel programming has been primarily limited to systems that can be modeled with continuous lower-level variables or network models. Such models typically allow for convenient single-level reformulations that can be solved by existing optimization methods. As in traditional mathematical programming, it is clear that introducing integer variables into a bilevel program yields a much richer modeling framework. Next, we formally describe our problem framework.

## 1.3 Definitions and Notation

A *linear program* (LP) is the problem of minimizing the value of a linear objective function represented by $c \in \mathbb{Q}^n$ over the polyhedral feasible region

$$\mathcal{S}_{LP} = \{x \in \mathbb{R}^n \mid Ax \geq b, x \geq 0\},$$

where $A \in \mathbb{Q}^{m \times n}$, and $b \in \mathbb{R}^m$. That is, the goal of linear programming is to determine

$$z_{LP} = \min_{x \in \mathcal{S}_{LP}} cx. \tag{LP}$$

A *mixed integer linear program* (MILP) is a natural generalization of an LP in which a specified subset of the decision variables are required to have integer values. Without loss of generality, we assume this subset is indexed 1 through $p \leq n$. Thus, the canonical MILP instance can be represented by the quadruple $(A, b, c, p)$ and has feasible region

$$\mathcal{S}_{MILP} = \left\{x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \mid Ax \geq b\right\}.$$

The goal of solving MILP is then to determine

$$z_{MILP} = \min_{x \in \mathcal{S}_{MILP}} cx. \tag{MILP}$$

A *mixed integer bilevel linear program* (MIBLP) is a generalization of a mixed integer linear program in which some of the variables are controlled by a secondary DM. Let $x \in X \subseteq \mathbb{R}^{n_1}_+$ represent the variables controlled by the *upper-level DM*, or *leader*, and let $y \in Y \subseteq \mathbb{R}^{n_2}_+$ represent the variables controlled by the *lower-level DM*, or *follower*. $X$ and $Y$ specify the integrality restrictions

on the decision variables. As before, we assume the upper- and lower-level integer variables are indexed 1 to $p_1$ and 1 to $p_2$, respectively, and define

$$X = (\mathbb{Z}^{p_1} \times \mathbb{R}^{n_1-p_1}) \quad \text{and} \quad Y = (\mathbb{Z}^{p_2} \times \mathbb{R}^{n_2-p_2}).$$

The canonical MIBLP is then the problem of determining

$$z_{MIBLP} = \min \left\{ c^1 x + d^1 y \mid x \in \mathcal{P}_U \cap X, y \in \operatorname{argmin}\{d^2 y \mid y \in \mathcal{S}_L(x) \cap Y\} \right\}, \tag{1.1}$$

where

$$\mathcal{P}_U = \left\{ x \in \mathbb{R}^{n_1} \mid A^1 x \geq b^1, x \geq 0 \right\}$$

is the *upper-level feasible region,*

$$\mathcal{S}_L(x) = \left\{ y \in \mathbb{R}^{n_2} \mid G^2 y \geq b^2 - A^2 x, y \geq 0 \right\}$$

is the *lower-level feasible region* with respect to a given $x \in \mathbb{R}^{n_1}$, $A^1 \in \mathbb{Q}^{m_1 \times n_1}$, $b^1 \in \mathbb{R}^{m_1}$, $A^2 \in \mathbb{Q}^{m_2 \times n_1}$, $G^2 \in \mathbb{Q}^{m_2 \times n_2}$, and $b^2 \in \mathbb{R}^{m_2}$. The region obtained by dropping the optimality requirement for the lower-level variables is given by

$$\Omega^I = \{(x, y) \in (X \times Y) \mid x \in \mathcal{P}_U, y \in \mathcal{S}_L(x)\}.$$

$\Omega^I$ is often referred to as the *joint feasible region.* If we also remove the conditions $x \in X$ and $y \in Y$, we obtain

$$\Omega = \{(x, y) \in (\mathbb{R}^{n_1} \times \mathbb{R}^{n_2}) \mid x \in \mathcal{P}_U, y \in \mathcal{S}_L(x)\}.$$

In later chapters, we let $A = [A^1|A^2]^T$, $G = [0|G^2]^T$, and $b = [b^1|b^2]^T$ for convenience. For each upper-level solution $x \in (\mathcal{P}_U \cap X)$, the follower's *rational reaction set* is

$$M^I(x) = \operatorname{argmin}\{d^2 y \mid y \in \mathcal{S}_L(x) \cap Y\}.$$

The *bilevel feasible set* is defined as

$$\mathcal{F}^I = \{(x, y) \mid x \in (\mathcal{P}_U \cap X), y \in M^I(x)\},$$

and often called the *inducible region.* (MIBLP) can be restated more simply as the problem of determining

$$z_{MIBLP} = \min_{(x,y) \in \mathcal{F}^I} c^1 x + d^1 y. \tag{MIBLP}$$

Because MILP is a special case of MIBLP, it is clear that MIBLP is also an NP-hard problem.

Since the literature dealing with the complexity of MIBLP is limited, we examine questions of complexity in Chapter 3. In addition, we investigate special cases of MIBLP that may be amenable to more effective solution methodologies than the general formulation. That defined by $X = \mathbb{R}^{n_1}$ and $Y = \mathbb{R}^{n_2}$ (i.e., $p_1 = p_2 = 0$) is one such case. This special form of (MIBLP) is called a bilevel linear program (BLP). We denote the feasible region of a BLP as $\mathcal{F}$, which results from dropping all integrality restrictions from $\mathcal{F}^I$. Formally, BLP is stated as

$$z_{BLP} = \min_{(x,y)\in\mathcal{F}} c^1x + d^1y. \tag{BLP}$$

BLP is a special case of a mathematical program with equilibrium constraints (MPEC), studied extensively by Luo et al. (1996).

In order to ensure the problem is well-posed and has a solution, some technical assumptions are necessary. First, we assume that $\Omega^I$ is nonempty and compact. This assumption is consistent with the literature (Moore and Bard, 1990) and allows us to apply Theorem 1.1 (see, e.g., Rudin, 1976), which guarantees that an optimal solution to the standard mathematical program

$$\min_{(x,y)\in\Omega^I} c^1x + d^1y$$

exists.

**Theorem 1.1 (Weierstrass' Theorem)** *If $f : \mathbb{R}^n \to \mathcal{R}$ is a continuous function, and $S$ is a nonempty, closed and bounded subset of $\mathbb{R}^n$, then there exists some $\hat{x} \in S$ such that $f(\hat{x}) \leq f(x)$ for all $x \in S$. Similarly, there exists some $\bar{x} \in S$ such that $f(\bar{x}) \geq f(x)$ for all $x \in S$.*

We also assume that, for every action by the upper-level DM, the problem faced by the lower-level DM is feasible and its LP counterpart is bounded. That is, $(\mathcal{S}_L(x) \cap Y) \neq \emptyset$ and

$$\min_{y\in\mathcal{S}(x)} d^2y$$

has a finite optimal solution, for all $x \in X$.[1]

As noted in Moore and Bard (1990), the assumption of lower-level feasibility is somewhat restrictive, especially in the case where the upper- and lower-level are in direct conflict (i.e. $d^1 = -d^2$). For example, if the goal of the upper-level DM is to disrupt the operation of system controlled by the lower-level DM (as in an interdiction problem), decisions that result in infeasible lower-level problems can be seen to be optimal. In this case, we may wish to relax the assumption $(\mathcal{S}_L(x) \cap Y) \neq \emptyset$. Methods for relaxing this assumption are discussed in Chapter 2. On the other hand, in some applications, it is unlikely that this assumption restricts the set of possible solutions. For example,

[1] It is clear that if the LP counterpart of the lower-level problem is bounded, so is its integer version.

in both the government and corporate systems described above, choosing upper-level solutions that result in an empty lower-level feasible region will likely be eliminated *a priori*, since it is in the best interest of both DMs to have a functioning system. Further, this assumption is consistent with the current infrastructure protection philosophy based on goal of understanding and mitigating the effects of an inevitable attack, rather than focusing on the unrealistic goal of attempting to prevent all attacks.

We also assume the lower-level DM is *semi-cooperative*, and will allow the upper-level DM to choose among alternative members of $M^I(x)$, in the case that this set is not a singleton. This is often referred to as the *optimistic formulation* of the problem. The main alternative in the literature is the *pessimistic formulation*, where one assumes the upper-level DM chooses the lower-level solution alternative corresponding to the worst outcome with respect to upper-level objective function, yielding a risk-averse formulation of the problem. The choice of the optimistic formulation is consistent with the majority of the literature and, in contrast to the pessimistic formulation, allows single-level reformulations in which the lower-level problem is replaced with appropriate equilibrium constraints (Dempe, 2003). Again, one might question this assumption in the context of a truly adversarial lower-level DM. However, we note that such a scenario would often be zero-sum, in which case the two formulations would yield identical objective values. It is, of course, possible to imagine situations which are not zero-sum, but in which the lower-level DM would prefer a solution that is worst with respect to the upper-level objective. In this case, the pessimistic formulation may be more appropriate, but solving such problems remains a significant challenge. The reader is referred to Loridan and Morgan (1996) for further insight on and discussion of the pessimistic formulation. For a broader perspective of the implications on the level DM cooperation within a competitive atmosphere, the reader is referred to the manuscript of Başar and Olsder (1999).

The *mixed integer interdiction problem* (MIPINT) is a generalization of the network interdiction model in which we broaden the class of lower-level systems to those that can be described by any MILP. In MIPINT, there exists a binary upper-level variable for each lower-level variable. These binary variables represent the upper-level decision to interdict the corresponding lower-level variables. Mathematically, the effect of interdiction is modeled using a variable upper bound constraint (VUB)

$$y \leq U(e - x)$$

in the lower-level problem, where $u \in \mathbb{R}^n$ is a vector of natural upper bounds on the vector $y$, $U = \mathrm{diag}(u)$, and $e$ is an $n$-dimensional column vector of ones. The model we consider is equivalent to the mixed integer linear system interdiction problem described in Israeli (1999). Formally, MIPINT

is the problem of determining

$$z_{MIPINT} = \max_{x \in \mathcal{P}_U^{\text{INT}} \cap \mathbb{B}^n} \min_{y \in \mathcal{S}_L^{\text{INT}}(x) \cap Y} dy \tag{MIPINT}$$

where

$$\begin{aligned}\mathcal{P}_U^{\text{INT}} &= \left\{x \in \mathbb{R}^n \mid A^1 x \leq b^1\right\} \\ \mathcal{S}_L^{\text{INT}}(x) &= \left\{y \in \mathbb{R}^n \mid G^2 y \geq b^2, -y \geq -U(e-x), y \geq 0\right\}\end{aligned}$$

and $Y = (\mathbb{Z}^p \times \mathbb{R}^{n-p}) \subseteq \mathbb{R}^n$. Both the network interdiction problem and MIPINT are examples of *zero-sum bilevel programs*.

(MIPINT) is, in fact, a special case of (MIBLP). To see this, let $\bar{c}^1 = 0$, $\bar{d}^1 = -d$, $\bar{A}^1 = -A^1$, $\bar{b}^1 = -b^1$, $\bar{d}^2 = d$,

$$\bar{G}^2 = \left[\frac{G^2}{-\mathbb{I}_{n\times n}}\right], \quad \bar{A}^2 = \left[\frac{0}{-U}\right], \quad \text{and} \quad \bar{b}^2 = \left[\frac{b^2}{-U}\right].$$

Then, we can formulate (MIPINT) as

$$z_{MIPINT} = \min\left\{\bar{c}^1 x + \bar{d}^1 y \mid x \in \bar{\mathcal{P}}_U \cap \bar{X}, y \in \operatorname{argmin}\{\bar{d}^2 y \mid y \in \bar{\mathcal{S}}_L(x) \cap \bar{Y}\}\right\},$$

where

$$\begin{aligned}\bar{\mathcal{P}}_U &= \left\{x \in \mathbb{R}^{n_1} \mid \bar{A}^1 x \geq \bar{b}^1, x \geq 0\right\}, \\ \bar{\mathcal{S}}_L(x) &= \left\{y \in \mathbb{R}^{n_2} \mid \bar{G}^2 y \geq b^2 - \bar{A}^2 x, y \geq 0\right\}, \\ \bar{X} &= \mathbb{B}^{n_1}, \\ \bar{Y} &= (\mathbb{Z}^{p_2} \times \mathbb{R}^{n_2 - p_2}),\end{aligned}$$

$n_1 = n_2 = n$, and $p_2 = p$.

As previously mentioned, the existing literature on interdiction models focuses on variations of the network interdiction problem, where applications are limited to scenarios in which the lower-level system can be described by a network model. A common example of network interdiction is the problem of maximizing the shortest path. In the *Maximum Shortest Path Problem* (MSPP), the follower attempts to move through a network along a shortest path. The leader's goal is to maximize the length of that shortest path by removing network arcs. Let $G = (N, A)$ be a graph in which the follower moves a commodity from the source node $s \in N$ and the sink node $t \in N$. We define $\delta^-(i)$ and $\delta^+(i)$ as the set of arcs directed out of and into node $i$, respectively. The length of arc $k \in A$ is given by $0 < c_k < \infty$, and the resource required to interdict this arc is $0 < r_k$. The total

*interdiction budget* available to the leader is $r_0$. One formulation of this problem is

$$\begin{aligned} \max_{x \in \mathcal{P}_U \cap \mathbb{B}^{|A|}} \quad & \sum_{k \in A} c_k y_k \qquad \text{(MSPP)} \\ \text{subject to} \quad & y \in \operatorname{argmin} \{ \sum_{k \in A} c_k y_k : \\ & \sum_{k \in \delta^-(i)} y_k - \sum_{k \in \delta^+(i)} y_k = \begin{cases} 1 & \text{for } i = s \\ 0 & \forall i \in N \setminus \{s,t\} \\ -1 & \text{for } i = t \end{cases}, \\ & 0 \le y_k \le u_k(1 - x_k) \quad \forall k \in A \}, \end{aligned}$$

where $\mathcal{P}_U = \{x \mid rx \le r_0\}$. A similar formulation of this problem by studied by Israeli and Wood (2002).

## 1.4 Previous Work

As discussed above, the initial motivation for multilevel programming in the literature arose from economic models of hierarchical competition, where bilevel programs are particularly well-suited for analyzing markets dominated by a large entity or *market-maker*. Bilevel programs subsequently proved their utility in both the private (Koopmans, 1951; Charnes et al., 1967; Cyert and March, 1955) and public (Beltramo, 1983) markets. Koopmans (1951) and Charnes et al. (1967) study hierarchical systems in the context of resource allocation. Cyert and March (1955) describe a pricing model in a oligopolistic market, based on the behavior of decision-makers at different levels within a firm's organizational structure. Bracken and McGill (1973) formalize the notion of a bilevel program by describing a mathematical program whose constraints contain optimization problems. Their model is limiting, however, since only the lower-level payoff function, rather than the set of feasible solutions, is dependent on decisions made at the higher level. Military applications of this model are given in Bracken and McGill (1974a) and a solution algorithm is suggested in Bracken and McGill (1974b). Some geometric results have been derived for the general multilevel programming problem (Benson, 1989) but, due to its inherent difficulty, finding theoretical and computational results for even the bilevel linear model has proved quite challenging. Research on this special case composes the majority of the remaining multilevel programming literature.

### 1.4.1 Bilevel Linear Programs

Falk (1973) studies the linear max-min (LMM) problem, a zero-sum bilevel problem, and suggests a branch-and-bound solution algorithm. Bard and Falk (1982) reformulate the bilevel program as a separable nonconvex program, when the lower-level problem is convex for fixed upper-level variables. A branch-and-bound algorithm that yields a global optimal solution to its piecewise linear approximation is described and several structural results are given. Namely, a single-level optimization problem is given that determines the existence of an optimal solution to the bilevel program. Candler and Townsley (1982) show that, if $\Omega$ is bounded and lower-level solutions are unique, solutions to BLPs occur at extreme points of $\Omega$. Bialas and Karwan (1982) and Bard (1984a) generalized this result, assuming only boundedness of $\Omega$. Savard (1989) provides a further generalization, showing the same result for BLPs with upper-level constraints and no assumption on $\Omega$. It is important to note that this result does not hold for quadratic or integer bilevel programs. Bard (1984b) proves that solving the linear bilevel program is equivalent to maximizing a linear function over a piecewise linear constraint set and gives necessary first order optimality for general bilevel programs. Bard (1983) gives a grid search algorithm for solving general bilevel programs (i.e. functions are not restricted to be linear). In Bard (1988), the case defined by all convex functions is considered and results similar to those found in Bard (1984b) for the linear case are given.

Fortuny-Amat and McCarl (1981) and Bard and Moore (1990) study bilevel programs with quadratic objective functions and linear constraints. In each, an algorithm designed to exploit the optimality conditions of the lower-level LP, replacing its objective with appropriate Karush-Kuhn-Tucker (KKT) conditions to yield a single-level problem, is suggested. In Fortuny-Amat and McCarl (1981), binary variables are introduced to eliminate the resulting nonlinear constraints and a mixed integer program is solved. On the other hand, Bard and Moore (1990) suggest a branch-and-bound solution methodology, where the complementarity constraints are relaxed to yield an LP, and branching is performed on the KKT multipliers. Ben-Ayed and Blair (1990) discuss the difficulties in finding exact solutions to bilevel linear programs and give a shorter proof of the problem's complexity than those given previously. Hansen et al. (1992) derive necessary optimality conditions on the tightness of the lower-level constraints and suggest new branching rules to be used in a branch-and-bound solution framework. Judice and Faustino (1992) give an algorithm based on solving a series of linear complementarity problems. For an overview of bilevel linear programming solution methods and applications, the reader is referred to the work of Anandalingam and Friesz (1992).

### 1.4.2 Mixed Integer Bilevel Linear Programs

Although bilevel linear programming has received increased attention recently, the literature on MIBLPs remains scarce. Moore and Bard (1990) introduce a general model, describe associated

computational challenges, and suggest a branch-and-bound algorithm, but the vast majority of the remaining mixed integer bilevel programming literature is restricted to various special cases. Bard and Moore (1992) develop a specialized algorithm for binary bilevel programs. Dempe (2001) considers the case in which all upper-level variables are continuous and all lower-level variables are integer and utilizes a cutting plane approach to approximate the lower-level feasible region. Wen and Yang (1990) consider the opposite case, where the lower-level problem is a linear program and the upper-level problem is an integer program. Linear programming duality is used to derive exact and heuristic solutions. One application of this special case was noted previously, for improving production of biochemicals. In their model, Burgard et al. (2003) utilize binary upper-level variables to represent yes/no decisions regarding the gene knockouts and reformulate the the problem using lower-level duality. Rather than using KKT conditions, they instead enforce lower-level primal and dual feasibility and require the primal and dual objectives to be equal. However, they neglect to include the lower-level primal variables in the dual objective and, thus, erroneously state that a MILP formulation has been derived. Recently, Köppe et al. (2009) developed a parametric integer programming approach for problems with pure integer lower-level problems.

### 1.4.3 Interdiction Problems

Interdiction problems have received a fair amount of attention in the literature, primarily due to their applicability in enemy network disruption planning. The original motivation, however, stemmed from an interest in performing sensitivity analysis on flow networks, with the goal of determining a transportation network's sensitivity to road closure (Wollmer, 1964). The interdiction model we study in Chapter 4 is reminiscent of this application, but the utility of the interdiction for sensitivity analysis has been largely overlooked in the literature. Instead, the majority of the research has its roots in military or homeland security applications. McMasters and Mustin (1970) and Ghare et al. (1971) study models for effective interdiction of a military supply network, while Wood (1993) and Washburn and Wood (1995) were motivated by the disruption of drug trafficking networks. In each, the interdictor attempts to minimize the maximum achievable flow on the underlying network; Wood (1993) gives an integer programming formulation of the problem and a proof of NP−completeness. Natural generalizations of maximum flow interdiction result by allowing partial arc interdiction Lim and Smith (2007), multiple upper-level objectives (Royset and Wood, 2007), stochastic interdiction success (Cormican et al., 1998; Held and Woodruff, 2005; Janjarassuk and Linderoth, 2008), or uncertain network structure (Morton et al., 2007). Israeli (1999) gives a comprehensive review of interdiction algorithms and studies deterministic shortest path interdiction in depth. Israeli and Wood (2002) study a closely related problem, in which interdiction by the leader causes an increase $d_k > 0$ in the length of an arc; the goal of the follower is to find the minimum length path in the resulting network. The problem is formulated as a bilevel program, and a decomposition solution

methodology is provided. More recently, deviations from lower-level network problems have been studied (Salmerón et al., 2009; Brown et al., 2009, e.g.,), but the treatment of general MILP interdiction appears to be limited to the Ph.D. thesis of Israeli (1999), in which a penalty function reformulation is introduced and solved via decomposition methods.

## 1.5 Related Problems

**MPEC.** A very closely related class of problems is that of Mathematical Programming with Equilibrium Constraints (MPEC). In fact, as noted above, BLP is a special case of MPEC, in which the equilibrium constraints arise from an optimization problem; the relationship between the two problem classes in detail by Colson et al. (2005a). MPEC generalizes BLP, by dropping the assumption of linear constraints and objective functions and allowing equilibrium constraints arising from more general conditions than those resulting from an LP.

Formally, let $C : \mathbb{R}^n \to \mathbb{R}^m$ be a set-valued map such that, for each $x \in \mathbb{R}^n$, $C(x)$ is a closed convex subset of $\mathbb{R}^n$. Then, for functions $F : \mathbb{R}^{n+m} \to \mathbb{R}$ and $f : \mathbb{R}^{n+m} \to \mathbb{R}^m$, nonempty closed set $Z \subseteq \mathbb{R}^{n+m}$, the standard MPEC formulation is

$$\begin{aligned} \min \quad & F(x,y) \\ \text{subject to} \quad & (x,y) \in Z \\ & y \in \mathcal{S}(x), \end{aligned} \tag{MPEC}$$

where, for each $x \in X_{\text{MPEC}}$, $\mathcal{S}(x)$ is the solution set of the variational inequality (VI) defined by $(f(x,\cdot), C(x))$, and $X_{\text{MPEC}} = \{x \in \mathbb{R}^n \mid (x,y) \in Z \text{ for some } y \in \mathbb{R}^m\}$.

In addition to those described in the bilevel literature, examples of MPEC can be found in robotics (Pang and Trinkle, 1996; Pang et al., 2005), facility location and production (Miller et al., 1992), engineering design (Klarbring et al., 1995; Klarbring and Rönnqvist, 1995; Kočvara and Outrata, 1990, 1995), machine learning (Mangasarian, 1996; Kunapuli, 2008), trade reform (Harrison et al., 1997), economics (Scarf, 1973), electricity market modeling (Smeers, 1997; Hobbs, 2001; Hobbs and Helman, 2004), structural mechanics (Maier and Novati, 1990; Tin-Loi and Misa, 1999; Tin-Loi and Pang, 1993), and options pricing (Huang and Pang, 1998; Benson et al., 2006).

A variety of solution methods exist for MPECs. Active set methods (Fukushima and Tseng, 2002; Izmailov and Solodov, 2008; Fukushima and Tseng, 2007; Liu and Ye, 2007; Chen and Goldfarb, 2007; Judice et al., 2007; Ralph, 2009) rely on solving a series of subproblems, whose complementarity conditions have been explicitly satisfied. Constraint regularization methods (Facchinei et al., 1999; Fukushima and Pang, 1999; Scholtes, 2001) attempt to put the constraints in a more tractable

form, often resulting in relaxations of the original problem, while penalty function methods (Mangasarian, 1976; Marcotte and Zhu, 1995; Mangasarian and Pang, 1997; Scheel and Scholtes, 2000) penalize constraint violation. Recently, sequential quadratic programming (SQP) (Liu et al., 1998; Jiang and Ralph, 1999; Jian, 2005; Fletcher et al., 2006; Liu et al., 2006) and filter methods (Fletcher and Leyffer, 2002a,b; Etoa, 2010) have been shown to be computationally effective. Other methods, with roots in traditional nonlinear methods include interior point (Byrd et al., 1999; Liu and Sun, 2004; Benson et al., 2006) and trust region (Scholtes and Stöhr, 1999; Colson et al., 2005b) algorithms. Implicit programming approaches (Outrata, 1994; Outrata and Zowe, 1995; Outrata et al., 1998) have also gotten some attention in the literature, but tend to require fairly strict assumptions on the problem (Luo et al., 1996). Unfortunately, the majority of the problems considered in the MPEC literature do not have integral variables and, thus, the solution methods are not applicable to the discrete problems we consider here. The reader is referred to the surveys by Ferris and Pang (1997), Ferris and Kanzow (2002), Colson et al. (2005a), and Hu et al. (2009) and the monograph by Luo et al. (1996) for further background on MPECs.

**Multiobjective Programs.** As discussed above, multiobjective programming is another framework used to model multiple objectives. Formally, the multiobjective program is defined as:

$$\text{vmin}_{x\in\mathcal{S}}[f_1(x), f_2(x), \ldots, f_k(x)], \tag{1.2}$$

where the operator vmin means the goal of solving (1.2) is to find *efficient* solutions in the feasible region $\mathcal{S}$. We say $\hat{x} \in S$ is efficient if there is no other $x \in \mathcal{S}$ such that $f_i(x) \leq f_i(\hat{x})$ for $i = 1, \ldots, k$ and $f_i(x) < f_i(\hat{x})$ for some $i$. That is there is no $x \in S$ that *dominates* $\hat{x}$. Further, $\hat{x} \in \mathcal{S}$ is considered *strongly efficient* if it is efficient and

$$f_i(\hat{x}) < f_i(x) \text{ for all } i.$$

Let $\mathcal{S}_E$ denote the set of efficient solutions and $Y_E$ denote the image of $\mathcal{S}_E$ in the outcome space (i.e. $Y_E = f(\mathcal{S}_E)$). $Y_E$ is the set of Pareto outcomes.

We are particularly interested in the *biobjective mixed integer linear program* (BMILP):

$$\text{vmin}_{x\in\mathcal{S}}[cx, dx], \tag{1.3}$$

where $X = \mathbb{Z}^p \times \mathbb{R}^{n-p}$, $\mathcal{S} = \{x \in X \mid Ax \geq b, x \geq 0\}$, $c, d \in \mathbb{Q}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{Q}^{m\times n}$. One approach to finding efficient solutions to (1.3) is to convert it into a single-objective problem by taking a convex combination of $c$ and $d$, to yield a so-called *weighted-sum subproblem* (Geoffrion,

1968)

$$\min_{x \in \mathcal{S}} \delta cx + (1-\delta)dx, \tag{1.4}$$

for $0 \leq \delta \leq 1$. Solutions to (1.4) for fixed $\delta$ are guaranteed to be efficient, but the converse does not hold. Thus, solving (1.4) is not sufficient if we must find all efficient solutions. However, it is a very straightforward approach and will yield, in most cases, a suitable number of efficient solutions. The reader is referred to Ralphs et al. (2006) for a discussion of alternative methods that guarantee generation of the entire solution set.

The primary difference between multiobjective and multilevel programming is the assumption that a single DM controls both objectives, in the former case. Thus, in multiobjective programming, the two objectives are optimized over a common feasible region. It has been shown, for problems in which upper-level variables appear in the lower-level objective, solutions to BLPs are not efficient in general (Bialas and Karwan, 1982; Bard, 1984b; Wen and Hsu, 1989). It is quite easy to see, by way of an example, that this result also applies to our setting.

**Example 1.** Consider the following MIBLP instance:

$$\begin{aligned} \min_{x \in \mathbb{B}^2} \quad & -3(x_1+x_2)-2(y_1+y_2) \\ \text{subject to} \quad & x_1+x_2 \leq 1 \\ & y \in \operatorname{argmin}\left\{-4y_1-5y_2 \mid -x_1-y_1 \geq -1, -x_2-y_2 \geq -1, y \in \mathbb{B}^2\right\}, \end{aligned} \tag{1.5}$$

It is easy to see, by inspection, that both $(x^1, y^1) = ((1,0),(0,1))$ and $(x^2, y^2) = ((0,1),(1,0))$ are optimal for (1.5). While both result in an upper-level objective value of $-5$, they have lower-level objective values of $-5$ and $-4$, respectively. Thus, $(x^2, y^2)$ cannot be efficient.

However, we also observe from the previous example that, although, solutions to MIBLPS are not efficient in general, there may be alternative upper-level solutions that yield efficient bilevel feasible solutions. This observation motivates a heuristic method for MIBLP that relies on determining efficient solutions to a related multiobjective program (see Chapter 3). Wen and Lin (1996) give a method for obtaining efficient pair of upper- and lower-level decisions when the DMs are willing to cooperate, but such solutions are not guaranteed to be feasible for the original bilevel problem.

**Stochastic Programs with Integer Recourse.** Stochastic programming is a framework for modeling mathematical programs in which the data is uncertain. In other words, these problems arise when a subset of the parameters in a deterministic program are replaced with random variables. One common method for modeling problem uncertainty is through *two-stage stochastic programming*. In a two-stage stochastic program, first-stage (or anticipatory) decisions must be made before the

outcomes of the random parameters are known. After the uncertainty is revealed, second-stage (or *recourse*) decisions are made in reaction to the outcome. As in bilevel programming, the decisions are sequential in theory, but the goal is to determine the decisions that yield the minimum expected objective values at the first- and second-stage "here and now."

Formally, the two-stage mixed integer stochastic programming problem is defined as:

$$z_{2SP} = \min \{cx + \mathbb{E}_\xi Q_\xi(x) \mid x \in \mathcal{P} \cap X\} \tag{1.6}$$

where $c \in \mathbb{R}^{n_1}$,

$$\mathcal{P} = \{Ax \geq b, x \geq 0\},$$

$A \in \mathbb{R}^{n_1 \times m_1}$, $b \in \mathbb{R}^{m_1}$. For any $\xi$,

$$Q_\xi(x) = \min\{dy \mid Wy \geq \omega(\xi) - Tx, y \in Y\},$$

where $W \in \mathbb{R}^{m_2 \times n_2}$ and $T(\xi) \in \mathbb{R}^{m_2 \times n_1}$. The vector $\xi$ is a random variable from the probability space $(\Xi, \mathsf{F}, \mathsf{P})$ and describes the realization of the uncertain scenarios. For each $\xi \in \Xi$, $\omega(\xi) \in \mathbb{R}^{m_2}$. Generally, $W$ and $T$ are referred to as the *recourse matrix* and *technology matrix*, respectively. As before, we assume the first- and second-stage integer variables are indexed 1 to $p_1 \leq n_1$ and 1 to $p_2 \leq n_2$, respectively, and define $X = \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1 - p_1}$ and $Y = \mathbb{Z}^{p_2} \times \mathbb{R}^{n_2 - p_2}$. As in Kong et al. (2006), the following assumptions are made on (1.6):

(A1) The random variable $\xi$ follows a discrete distribution with finite support.

(A2) The first-stage feasibility set $(\mathcal{P} \cap X)$ is nonempty and bounded.

(A3) $Q_\xi(x)$ is finite for all $x \in (\mathcal{P} \cap X)$ and $\xi \in \Xi$.

We define the functions

$$\phi(\beta) = \min\{cx \mid x \in \mathcal{P}(\beta) \cap X\},$$

where $\mathcal{P}(\beta) = \{x \mid Ax \geq b, Tx \geq \beta\}$; and

$$\varphi(\beta) = \min\{qy \mid y \in \mathcal{S}(\beta) \cap Y\},$$

where $\mathcal{S}(\beta) = \{y \mid Wy \geq \beta\}$, for all $\beta \in \mathbb{R}^{m_2}$. This allows us to reformulate (1.6) as:

$$z_{2SP} = \min \{\phi(\beta) + \mathbb{E}_\xi \varphi(\omega(\xi) - \beta) \mid \beta \in \mathbf{B}\}, \tag{2SP}$$

where $\mathbf{B} = \{\beta \mid Tx = \beta, x \in X\}$. Variables $\beta$ are the so-called *tender variables*. Assumption (A1) guarantees that $\mathbf{B}$ is finite. In the continuous version of (1.6) (i.e., $p_1 = p_2 = 0$), $\varphi$ is a piecewise

linear and convex function on $\{x \mid Ax \geq b, x \geq 0, x \in X\}$ for each realization $\omega(\xi)$. Thus, under Assumption (A1), $Q_\xi(x)$ is also piecewise linear and convex, implying that (1.6) is the problem of minimizing a convex function over a set of linear constraints. Assumption (A2) is a standard assumption, and ensures that $(\mathcal{P} \cap X)$ is a finite set. The final assumption defines a property known as *relatively complete recourse* and guarantees that $Q_\xi(x)$ is feasible for all $x \in (X \cap \mathcal{P})$ and $\xi \in \Xi$. The reader is referred to the survey paper of Schultz et al. (1996), for further insight into the implications of these assumptions, as well as an overview of two-stage stochastic programming results. A comprehensive review of the stochastic programming literature is provided by Kall and Wallace (1994) and Birge and Louveaux (1997).

Two-stage stochastic programming problems are not generally associated with bilevel programming. However, it is easy to observe the similarities by examining the mathematical formulations. In fact, algorithms developed for bilevel programs can be used to solve two-stage stochastic programs. Conversely, two-stage stochastic programming algorithms are able to solve a particular special case of the bilevel program.

**Parametric Mathematical Programs.** Sensitivity analysis is another method for dealing with uncertainty in input data. The goal of sensitivity analysis is to understand how the optimal solutions and objective values change in mathematical programs as the input data is varied. In particular, once an optimal solution is found, information gleaned from a related mathematical program allows us to predict the effects of changing the objective function, constraint matrix, and right-hand-side values. Generally, we are interested in determining conditions under which the current solution remains optimal, despite changes to the problem data. Further, sensitivity analysis describes how to obtain new optimal solutions when these conditions are violated, without resolving the problem from scratch.

Parametric programming is a method for performing systematic sensitivity analysis. In particular, parametric programming is used to obtain the set of optimal solutions over a range of input values. For example, we can formulate the *parametric program*:

$$\min \left\{ cx \mid Ax \geq (b + \theta b'), x \geq 0, x \in X \right\}, \tag{1.7}$$

parameterized by the scalar $\theta$. The goal of "solving" (1.7) is to determine an optimal solution and objective value for each value of $\theta$. Similarly, we can define models based on the parameterization of the objective function:

$$\min \left\{ (c + \varphi c')x \mid Ax \geq b, x \geq 0, x \in X \right\}, \tag{1.8}$$

and the constraint matrix:

$$\min \left\{ cx \mid (A + \lambda A')x \geq b, x \geq 0, x \in X \right\}, \tag{1.9}$$

where $\varphi$ and $\lambda$ parameterize (1.8) and (1.9), respectively. The goal of parametric programming is to determine the global dependence on the problem parameter, allowing us to construct the *parametric function* $z$, which yields the optimal value of as a function of the parameter. As we will see next, understanding how the solutions to (1.7) evolve as we change $\theta$ is essential to developing algorithms for MIBLPs.

## 1.6 Computational Challenges of MIBLP

The fact that MIBLP is NP-hard (see Section 1.3) indicates that solving MIBLPs in practice is likely to be challenging. A natural approach to developing algorithms for solving MIBLPs is to consider generalizations of the techniques that are used for MILPs. In particular, we would like to be able to generalize the paradigm of LP-based branch and bound used to solve MILPs, by replacing the LP relaxation with the BLP obtained from relaxing integrality restrictions. Unfortunately, as we will see next, this method does not yield a valid relaxation, and there is no immediately apparent way to obtain such a generalization.

In a branch-and-bound algorithm for a standard MILP, integrality constraints are removed and the resulting LP, called the LP relaxation, which is easily seen to be a relaxation of the original MILP, is solved. The solution to the LP relaxation yields useful information about the original problem. In particular, in algorithms for solving MILPs, we frequently use the following properties.

(P1) If the LP relaxation has no feasible solution, then neither does the original problem.

(P2) If the LP relaxation has a solution, then the objective value is a valid lower bound on the that of the original problem.

(P3) If the solution to the LP relaxation is integral, then it is optimal for the original problem.

Properties (P2) and (P3) result from the fact that the set of feasible solutions for the original MILP is contained in the corresponding set for the relaxation. However, for a MIBLP, this is not the case.

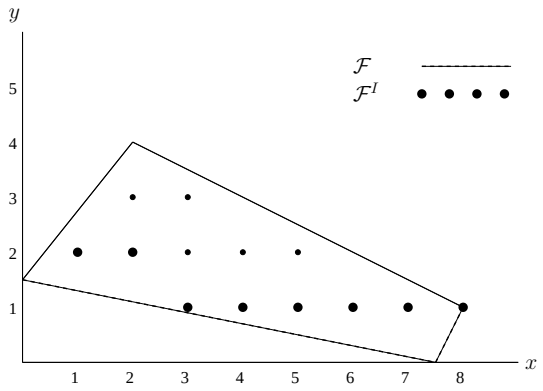



Figure 1.1: The feasible region of a MIBLP.

**Example 2.** Consider the instance of (MIBLP), from Moore and Bard (1990),

$$\begin{aligned} \min_{x \in \mathbb{Z}_+} \quad & -x - 10y \\ \text{subject to} \quad & y \in \operatorname{argmin} \{ y : 25x - 20y \geq -30 \\ & \qquad -x - 2y \geq -10 \\ & \qquad -2x + y \geq -15 \\ & \qquad 2x + 10y \geq 15 \\ & \qquad y \in \mathbb{Z}_+ \}, \end{aligned} \tag{1.10}$$

illustrated in Figure 1.1. In the figure, the polyhedron represents the set $\Omega$, while the integer points in this polyhedron comprise the discrete set $\Omega^I$. Within each of $\Omega$ and $\Omega^I$, we have indicated points that satisfy the optimality constraint on the lower-level variables (i.e. the bilevel feasible solutions). These are denoted $\mathcal{F}$ and $\mathcal{F}^I$, respectively. From the figure, it is easy to see that $\mathcal{F} \subseteq \Omega$, $\mathcal{F}^I \subseteq \Omega^I$, and $\Omega^I \subseteq \Omega$. It is not the case, however, that $\mathcal{F}^I \not\subseteq \mathcal{F}$, which implies that the set of feasible solutions to the MIBLP is not contained in that of the corresponding BLP and, hence, that this BLP does not yield a relaxation of the original problem. In this example, optimizing over the continuous region $\mathcal{F}$ yields the *integer* solution $(8, 1)$, with the upper-level objective value $-18$. However, the true solution to the MIBLP is $(2, 2)$, with upper-level objective value $-22$.

Example 2 allows us to make two important observations:

(O1) The objective value obtained by relaxing integrality is not a valid bound on the solution value of the original problem, since we may have

$$\min_{(x,y) \in \mathcal{F}} c^1 x + d^1 y > \min_{(x,y) \in \mathcal{F}^I} c^1 x + d^1.$$

(O2) Even when solutions to $\min_{(x,y)\in\mathcal{F}} c^1x + d^1y$ are in $\mathcal{F}^I$, they are not necessarily optimal.

Thus, except in certain special cases, only Property (P1) can be generalized. This implies that we cannot simply generalize MILP branch and bound for MIBLP by substituting the LP relaxation with the BLP obtained by dropping the integrality constraints in a MIBLP.

Figure 1.1 also illustrates an important difference between the continuous and integer versions of bilevel programming. It is well-known (Candler and Townsley, 1982; Bialas and Karwan, 1982; Bard, 1984a; Savard, 1989) that if a solution to (BLP) exists, it occurs at an extreme point of $\Omega$. This property has been exploited to develop algorithms based on vertex enumeration (Papavassilopoulos, 1982; Candler and Townsley, 1982; Bialas and Karwan, 1982; Dempe, 1987; Chen and Florian, 1992; Chen et al., 1992; Tuy et al., 1994). However, we can easily see from the figure, this property does not hold when we add integrality constraints on the variables. This is analogous to the situation one encounters when comparing linear and integer programming.

While Figure 1.1 demonstrates the difficulties of applying known algorithmic methods to MIBLPs, it also offers some insight into potential novel solution methods. It is easy to see, by inspection, that the set $\mathcal{F}^I$ is equivalent to the set $\{(v_1, y_1^*), \ldots, (v_k, y_k^*)\}$, where $y_i^*$ is the optimal solution to the MILP

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & -20y \geq -30 - 25v_i \\ & 2y \geq -10 + v_i \\ & y \geq -15 + 2v_i \\ & 10y \geq 15 - 2v_i \\ & y \in \mathbb{Z}_+ \end{aligned} \tag{1.11}$$

and $\{v_1, v_2, \ldots, v_8\} = \{1, 2, \ldots, 8\}$. In other words, if we knew the optimal solution of (1.11) for each $v$, we could use this information to generate $\mathcal{F}^I$ and develop an algorithm to solve (1.10). In fact, for this simple example, we could simply replace the lower-level optimality conditions with a constraint of the form $y = z(v_i)$, where $z(v_i)$ is a function that returns $y_i^*$ given $v_i$. In more general terms, $z(v_i)$ returns the optimal value of the lower-level problem for a particular right-hand-side. This provides further evidence that understanding the dependence of optimal lower-level solutions on the upper-level decision vector is crucial to designing effective methods for solving MIBLPs, and motivates our study of the MILP value function.

A bilevel program can be thought of as an optimization problem into which a second parametric optimization has been embedded. Thus, a natural method for developing algorithms for bilevel programs is to study the structure of the function that returns the optimal solution value of the

lower-level problem for a given upper-level solution. This is essentially the goal of parametric programming. Much of the work described in this dissertation has its basis in exploiting the structure of this so-called *value function* of the lower-level problems to develop algorithmic methods. In Chapter 3, we describe the theoretical properties of the MILP value function and demonstrate its utility in bilevel algorithm design.

## 1.7 Major Contributions

Over the past several decades, there have been many great accomplishments in the development of theory and methodology for solving large-scale mixed integer programs. In this dissertation, our goals are to 1) continue this development and 2) leverage known results for the development of analogous techniques for mixed integer bilevel programs. Further, we utilize knowledge gained from comparing linear and mixed integer linear programs to bridge the gap between continuous and discrete bilevel programs. The primary contributions of this dissertation are:

- Development of a branch-and-cut framework for pure integer bilevel linear programs.
- Development and distribution of an open source bilevel solver package based on our algorithmic framework which allows for easy incorporation of additional algorithmic components and problem-specific customization.
- Demonstration of solver customization using specialized methods developed for interdiction models, a class of models which encompasses the network interdiction problem and is of particular importance for applications in homeland security.
- Development of a theoretical algorithm for MIBLP based on iterative approximation of the lower-level value function.
- Provision of two novel heuristic methods for MIBLP that yield good solutions without a large computational expense.
- Derivation of a novel branch-and-cut algorithm for a class of Steiner Arborescence Problems with an application in infrastructure protection.

## 1.8 Outline

The remainder of this dissertation is as follows. In Chapter 2, we develop and branch-and-cut framework for pure integer BLPs (IBLPs). In this chapter, we also provide the implementation details of

our solver, generalizations of known enhancements for MILP algorithms, and computational results on a set of randomly-generated IBLPs. Then, in Chapter 3, we return to the general formulation of MIBLP. First, an analysis of the problem complexity is provided. Then, using the mixed-integer extensions of LP duality theory, we provide several single-level reformulations of MIBLP and an exact algorithm based on iterative approximations of the lower-level value function derived from lower-level dual solutions. Due to the inherent difficulty of these problems, finding exact solutions to large instances is a major challenge. To this end, we provide two novel heuristics, which can be used to find good solutions quickly, and demonstrate their usefulness with computational results. In Chapter 4, we focus on applications of (MIPINT), beginning with a detailed study of one particular early warning system (EWS), and an ILP used to optimize its design, and a discussion of the utility of MIPINT in a particular type of sensitivity analysis. In this chapter, we also demonstrate the foundations of problem-specific customization by way of (MIPINT) and provide several specialized methods that exploit the problem's structure. Finally, in Chapter 5 we motivate further study by describing new applications of bilevel programming, provide conclusions, and suggest directions for further research.

# Chapter 2

# Pure Integer Bilevel Linear Programming

In the previous chapter, we described several applications of discrete bilevel programs and illustrated the computational challenges these problems present. It is clear that developing solution methods for (MIBLP) that are analogous to those in the MILP literature is an important, yet ambitious, task. In this chapter, we initially focus on the development of such methods for the pure integer version of MIBLP, an important special case of the canonical problem. In particular, we develop a branch-and-cut framework for this problem class that leverages knowledge of the well-known branch and cut, an algorithm of ILP, employing modifications where necessary to deal with the more general form of the bilevel programming problem.

The pure integer version of (MIBLP), referred to henceforth as the *integer bilevel linear program* (IBLP), is the problem of determining:

$$z_{IBLP} = \min_{(x,y)\in\mathcal{F}^I} c^1x + d^1y, \tag{IBLP}$$

where

$$\mathcal{F}^I = \{(x,y) \mid x \in (\mathcal{P}_U \cap \mathbb{Z}^{n_1}), y \in M^I(x)\},$$

and

$$M^I(x) = \operatorname{argmin}\{d^2y \mid y \in \mathcal{S}_L(x) \cap \mathbb{Z}^{n_2}\},$$

which results from setting $p_1 = n_1$ and $n_2$ in (MIBLP). For the remainder of this chapter, we assume that all data necessary to define an instance of (IBLP) is integer. That is, $A^1 \in \mathbb{Z}^{m_1\times n_1}$, $b^1 \in \mathbb{Z}^{m_1}$, $A^2 \in \mathbb{Z}^{m_2\times n_1}$, $G^2 \in \mathbb{Z}^{m_2\times n_2}$, and $b^2 \in \mathbb{Z}^{m_2}$. Further, we maintain the assumptions given in Chapter 1:

(A1) $\Omega^I$ is nonempty and compact;

(A2) for each upper-level solution, the follower's rational reaction set is nonempty; and

(A3) the linear relaxation of the lower-level problem $\min_{y \in \mathcal{S}(x)} d^2 y$ has a finite optimal solution, for all $x \in X$.

As with many classes of mathematical programs, the most obvious route to achieving global optimality is the development of bounding procedures that can be used to drive a branch-and-bound algorithm. As discussed in Chapter 1, however, the bounding, fathoming, and branching procedures employed in traditional LP-based branch-and-bound algorithms cannot be applied in a straightforward way. In Section 2.1, we describe how to overcome these challenges to develop a generalized branch-and-cut algorithm for IBLPs that follows the same basic paradigm used in ILP.

## 2.1 Polyhedral Approaches to IBLP

As we have seen in Section 1.6, developing a branch-and-bound method for solving IBLP is not as straightforward as mimicking LP-based branch and bound. We cannot get a valid bound simply by dropping integrality restrictions on the variables. However, we show that the general framework can still be applied in our setting, once suitable modifications are made to obtain a valid relaxation. Further, we describe classes of inequalities that can be used in a branch-and-cut framework to separate problems that are integer feasible but not bilevel feasible. The method by which we arrive at these inequalities can be considered analogous to those used in ILP, in the sense that they are based on disjunctions arising from the integrality restrictions on the variables.

### 2.1.1 Bounding

We have already observed that the BLP

$$\min_{(x,y) \in \mathcal{F}} c^1 x + d^1 y, \tag{2.1}$$

obtained by removing the integrality restriction on all decision variables, is not a valid relaxation and does not provide a valid bound on the original problem. This is because removing the integrality restriction on the lower-level variables may actually cause solutions that were previously bilevel feasible to become infeasible. Further complicating matters is the fact that verifying the feasibility of a given solution is itself an NP-hard problem that involves solving the lower-level problem, a standard ILP.

Although (2.1) does not provide a valid bound, it is easy to see that that infeasibility of this BLP also implies infeasibility of the original instance. This result is given in the following proposition.

**Proposition 2.1** *If $\mathcal{F} = \emptyset$, then $\mathcal{F}^I = \emptyset$.*

**Proof.** Suppose, for sake of contradiction, this is not the case. This implies that $\mathcal{F} = \emptyset$, but there exists some $(\hat{x}, \hat{y}) \in \mathcal{F}^I$. By definition, $(\hat{x}, \hat{y}) \in F^I$ implies $\hat{x} \in (\mathcal{P}_U \cap X)$ and $\hat{y} \in M^I(\hat{x})$. Clearly, $\hat{x} \in (\mathcal{P}_U \cap X)$ implies $\hat{x} \in \mathcal{P}_U$, so $\mathcal{P}_U$ is not empty. Also, $\hat{y} \in M^I(\hat{x})$ implies $\hat{y} \in (\mathcal{S}_L(\hat{x}) \cap Y)$. This, in turn, implies $\hat{y} \in \mathcal{S}_L(\hat{x})$ and, therefore, $\mathcal{S}_L(\hat{x}) \neq \emptyset$. Since $\mathcal{S}_L(\hat{x}) \neq \emptyset$, we must also have

$$\operatorname{argmin}\{d^2 y \mid y \in \mathcal{S}_L(\hat{x})\} \neq \emptyset$$

Combining these implies $\mathcal{F} \neq \emptyset$ and we have a contradiction. □

Although removing the integrality restrictions on all variables does not result in a valid relaxation, removing integrality conditions and the requirement $y \in M^I(x)$ *does* yield the relaxation

$$\min_{(x,y)\in\Omega} c^1 x + d^1 y, \tag{LR}$$

similar to one suggested by Moore and Bard (1990). The resulting bound can be used in combination with a standard variable branching scheme to yield an algorithm that solves (IBLP). Not surprisingly, however, the bound is too weak to be effective on interesting problems.

In order to improve upon the bounds yielded by (LR) and to avoid the potential difficulties associated with being forced to branch when faced with an infeasible integer solution, we consider here a branch-and-cut algorithm based on the iterative generation of linear inequalities valid for $\mathcal{F}^I$ and augmentation of the linear system describing $\Omega$ until an optimal member of $\mathcal{F}^I$ is exposed or we choose to branch. The procedures we suggest are analogous to those used in the case of ILP but also address the fact that integer solutions may not be feasible in this setting.

### 2.1.2 Generating Valid Inequalities

An inequality defined by $(\pi_1, \pi_2, \pi_0)$ is called a *valid inequality* for $\mathcal{F}^I$ if $\pi_1 x + \pi_2 y \leq \pi_0$ for all $(x, y) \in \mathcal{F}^I$. Unless $\operatorname{conv}(\mathcal{F}^I) = \Omega$, there exist inequalities that are valid for $\mathcal{F}^I$, but are violated by some members of $\Omega$. Clearly, except in trivial instances, we can expect $\operatorname{conv}(\mathcal{F}^I) \neq \Omega$. In fact, in contrast to ILP, even complete generation $\operatorname{conv}(\Omega^I)$ is insufficient to solve the problem. This is illustrated in Figure 2.1. In the figure, the shaded region represents convex hull of $\mathcal{F}^I$. The closure of $\operatorname{conv}(\Omega^I)$ is shown by the dashed line outside $\operatorname{conv}(\mathcal{F}^I)$. It is clear from the picture that there exist inequalities valid for $F^I$ that are violated by members of $\Omega^I$.

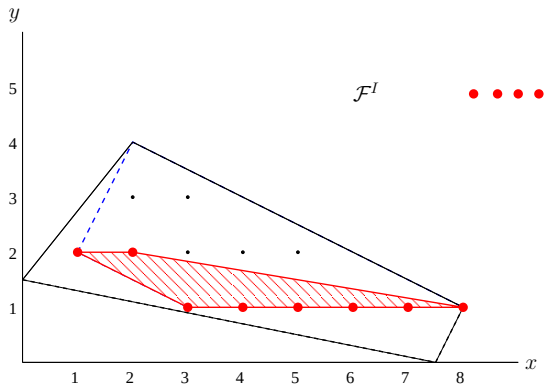



Figure 2.1: Comparing the convex hulls of $\Omega^I$ and $\mathcal{F}^I$.

In order to generate inequalities that separate members of $\mathcal{F}^I$ from $\Omega$ and $\Omega^I$, we must use information not contained in the linear description of $\Omega$. For a point $(x, y)$ to be feasible for a (IBLP), it must satisfy three conditions:

(C1) $(x, y) \in \Omega$ ,

(C2) $(x, y) \in (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$ , and

(C3) $y \in M^I(x)$.

This is in contrast to standard ILPs, where we have only the first two conditions. However, the methodology can be seen as equivalent to that used for ILPs. In order to derive inequalities for ILPs, we utilize the integrality conditions, since the linear description alone is insufficient. Here, we have an additional feasibility condition, so it is natural to assume that Conditions (C1) and (C2) alone will not suffice.

Because the first requirement is enforced by requiring membership in $\Omega$, we must derive valid inequalities from the other two conditions. We start with the following straightforward, but useful observations.

**Observation 2.1** *If the inequality $(\pi_1, \pi_2, \pi_0)$ is valid for $\Omega^I$, it is also valid for $\mathcal{F}^I$.*

**Observation 2.2** *Let $(x, y) \in \Omega$ such that $y \notin M^I(x)$. If the inequality $(\pi_1, \pi_2, \pi_0)$ is valid for $\Omega^I \setminus \{(x, y)\}$, it is also valid for $\mathcal{F}^I$.*

Observation 2.1 is derived from the relationship $\mathcal{F}^I \subseteq \Omega^I$ and allows us to separate fractional solutions to the LP resulting from removal of the lower-level optimality and integrality restrictions. Observation 2.2 states that we can separate points that are integer but not bilevel feasible. From these observations, we can derive classes of valid inequalities to be used in a cutting plane procedure.

To initialize the cutting plane procedure, we must first solve the relaxation

$$\min_{(x,y)\in\Omega} c^1x + d^1y. \tag{LR}$$

If the solution $(\hat{x}, \hat{y})$ to (LR) does not satisfy condition (C2) above, we may apply standard branching techniques used to separate points in $\Omega \setminus \Omega^I$ from $\Omega^I \supseteq \mathcal{F}^I$. In theory, we can also employ any standard cutting plane techniques used in ILP algorithms (see Cornuéjols (2008) for an overview) to separate the fractional point. However, the bilevel cut generation method we introduce here relies heavily on the integrality of the data defining the problem. Thus, we restrict these methods to those which result in new inequalities with only integer coefficients. Developing methods for employing the full set of ILP cutting plane methods in congruence with bilevel programming branch and cut is an area of future research.

If $(\hat{x}, \hat{y})$ satisfies condition (C2), then we must check whether it satisfies condition (C3). This is done by solving the lower-level problem

$$\min_{y\in\mathcal{S}_L(x)\cap Y} d^2y \tag{2.2}$$

with the fixed upper-level solution $\hat{x}$. Let the solution to this ILP be $y^*$. If $d^2\hat{y} = d^2y^*$, then $\hat{y}$ is also optimal for (2.2) and we conclude that $(\hat{x}, \hat{y})$ is bilevel feasible. Otherwise, we must again generate an inequality separating $(\hat{x}, \hat{y})$ from $\mathcal{F}^I$. In either case, however, $(\hat{x}, y^*)$ is bilevel feasible and provides a valid lower bound on the optimal solution value of the original IBLP.

Now suppose $d^2\hat{y} > d^2y^*$. In this case, $(\hat{x}, \hat{y})$ does not satisfy condition (C3) and is therefore not bilevel feasible. We may still use $(\hat{x}, y^*)$ to bound the original problem, but we would like to add an inequality to (LR) that is valid for $\mathcal{F}^I$ and violated by $(\hat{x}, \hat{y})$. We describe one such inequality next.

Based on the above discussion, the following result describes a method for generating valid inequalities for IBLPs.

**Proposition 2.2** *Let $X = \mathbb{Z}^{n_1}$ and $Y = \mathbb{Z}^{n_2}$. Let $(\hat{x}, \hat{y}) \in \Omega^I$ be a basic feasible solution to* (LR)*. Let $I$ be the set of constraints that are binding at $(\hat{x}, \hat{y})$, Then*

$$\pi_1x + \pi_2y \geq \pi_0 + 1, \tag{2.3}$$

*where $(\pi_1, \pi_2) = \sum_{i\in I}(a_i, g_i)$ and $\pi_0 = \sum_{i\in I} b_i$, is valid for $\mathcal{F}^I$.*

**Proof.** The fact that $(\hat{x}, \hat{y})$ is a basic feasible implies that there exist $n = n_1 + n_2$ linearly independent constraints in the description of $\Omega$ that are binding at $(\hat{x}, \hat{y})$. Thus, the system $a_i'x + g_i'y = b_i, i \in I$ has a unique solution, namely $(\hat{x}, \hat{y})$. This, in turn, implies that $(\hat{x}, \hat{y})$ is the unique point

of intersection between the hyperplane defined by the equation $\pi_1 x + \pi_2 y = \pi_0$ and the set $\Omega^I$. It follows that the inequality $\pi_1 x + \pi_2 y \geq \pi_0$ is valid for $\Omega^I$. Because the face of $\Omega$ induced by this inequality does not contain any other members of $\Omega^I$ and there does not exist $(x, y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2}$ such that $\pi_1 x + \pi_2 y \in (\pi_0 + 1, \pi_0)$, this implies that the inequality $\pi_1 x + \pi_2 y \geq \pi_0 + 1$ is valid for $\Omega^I \setminus \{(\hat{x}, \hat{y})\}$. Applying Observation 2.2 yields the result. □

**Example 3.** An example of the cutting plane procedure is illustrated in Figure 2.2 for the instance

$$\min \{-y \mid y \in \operatorname{argmin} \{y \mid x - y \geq -2, 2x + y \geq 2, -3x + y \geq -3, -y \geq -3, x, y \in \mathbb{Z}_+\}\}.$$

In the figure, we can see the bilevel feasible region $\mathcal{F}^I = \{(0, 2), (1, 0), (2, 3)\}$. Also shown in the

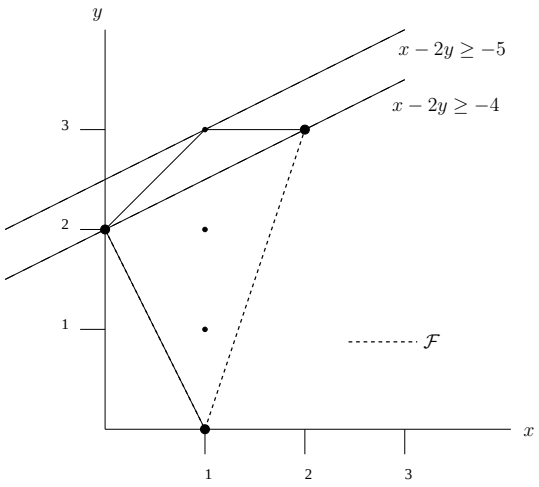



Figure 2.2: An example of the bilevel feasibility cut.

figure is the bilevel feasible region $\mathcal{F}$ of the corresponding BLP. In this example, we start with the integer point $(1, 3)$, an optimal solution to the LP

$$\min \{-y \mid x - y \geq -2, 2x + y \geq 2, -3x + y \geq -3, -y \geq -3, x, y \in \mathbb{R}_+\}.$$

It is easy to see that this point is not bilevel feasible, because the rational choice for the lower-level DM would be $y = 0$, when $x = 1$. Thus, we require a cut that separates $(1, 3)$. Combining the constraints active at $(1, 3)$ yields the half-space $\{(x, y) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \mid x - 2y \geq -5\}$ and applying the procedure described above, we obtain the new inequality

$$x - 2y \geq -4,$$

which is valid for $\mathcal{F}^I$, but not satisfied by $(1, 3)$. Note that after adding this cut, the optimal solution

is obtained in the next iteration. Without the cutting plane procedure we have just described, we would be forced to branch after producing this solution in a branch-and-bound framework. □

In order to solve problems of interesting size, additional classes of valid inequalities derived from Condition (C3) are necessary. In the following chapter, we describe one such class for MIBLPs that utilizes information from the value function of the lower-level MILP. Then, in Chapter 4, we describe two such classes for bilevel problems with binary upper-level variables. Derivation of such classes is another important area of future research.

### 2.1.3 Branching

As we have described, an important advantage of our algorithm over its predecessor from Moore and Bard (1990), is the fact that, in the case of IBLP, we are not forced to branch after producing an infeasible integer solution. Therefore, we are free to employ the well-developed branching strategies used in algorithms for traditional ILP, such as strong branching, pseudocost branching, or the recently introduced reliability branching (Achterberg et al., 2005). Of course, it is also possible to branch using disjunctions obtained from violations of Condition (C3). Examples of disjunctions on which we can branch are described in Chapter 3, for MIBLPs, and in Chapter 4, for interdiction problems.

### 2.1.4 Branch and Cut

Putting together the procedures of the preceding three sections, we obtain a branch-and-cut algorithm that consists of solving the linear relaxation (LR), iteratively generating valid inequalities to improve the bound, and branching when necessary. In addition to the obvious advantage of producing potentially improved bounds, an advantage of this approach over the one proposed by Moore and Bard (1990) is that it relies only on the solution of standard ILPs, rather than BLPs. Further, if we are able to obtain cutting planes to separate integer bilevel infeasible solutions, the algorithm preserves all the usual rules of fathoming and branching. It therefore allows us to immediately leverage our knowledge of how to solve standard ILPs. The general framework of such an algorithm is described next.

Let

$$\min_{(x,y)\in\mathcal{F}_t^I} c^1x + d^1y. \tag{IBLP$^t$}$$

be the IBLP defined at node $t$ of the branch-and-cut tree. To process node $t$, we first solve the LP

$$z^t_{\text{LP}} = \min_{(x,y)\in\Omega_t} c^1x + d^1y. \tag{LP$^t$}$$

and denote its solution by $(x^t, y^t)$ (if it exists). If either the LP is infeasible or the optimal value of (LP$^t$) is greater than the current lower bound $L$, we can fathom the current node. Otherwise, we can either generate valid inequalities to separate the current solution from $\mathcal{F}^I$ or branch. If $(x^t, y^t) \in \Omega^I$, we check for bilevel feasibility. If the solution is feasible, we can stop. Otherwise, we can either add cuts, to separate the current solution from $\Omega^I \setminus \{(x^t, y^t)\}$, or branch. In the case of an IBLP, we have the choice of adding cuts of the form (2.3), or branching on the integer variables as in Moore and Bard (1990). On the other hand, in the mixed integer case, we can use the disjunctions obtained from the lower-level value function to define a branching rule (see Chapter 3). If a fractional solution is found, we either add cuts to separate the current solution from $\Omega^t \cap (X \times Y)$ and iterate or else we branch. A general outline of the node processing subroutine is given in Algorithm 2.1. A description of our implementation of this algorithm is given in Section 2.3.

**Algorithm 2.1** Node Processing Loop

1: Solve (LP$^t$). If (LP$^t$) has an optimal solution, denote it $(x^t, y^t)$. Then:

- If (LP$^t$) is infeasible, so is (IBLP$^t$) and the current node can be pruned.
- If $z^t_{\text{LP}} \geq L$, the current node can be pruned.
- If $(x^t, y^t) \in \Omega^I$, go to Step 2, else go to Step 4.

2: Fix $x \leftarrow x^t$, and solve

$$z^t_{\text{LL}} = \min_{y \in \mathcal{S}_L(x^t) \cap Y} d^2 y.$$

If $z^t_{\text{LL}} = d^2 y^t$ set $L \leftarrow c^1 x^t + d^1 y^t$ and prune the current node; else go to Step 3.

3: Either set

$$\Omega_{t+1} = \Omega_t \cap \{(x, y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mid \pi_1 x + \pi_2 y \geq \pi_0\},$$

where $(\pi_1, \pi_2, \pi_0)$ is a valid inequality for $\mathcal{F}^I$, set $t \leftarrow t + 1$, and go to Step 1, or branch using a valid disjunction.

4: Add cuts valid for $\Omega^t \cap (X \times Y)$ to separate the current fractional solution. Resolve the new LP and let $(\hat{x}, \hat{y})$ be its optimal solution. If $(\hat{x}, \hat{y}) \in \Omega^I$, set $(x^t, y^t) \leftarrow (\hat{x}, \hat{y})$ and go to Step 2; else branch.

As described above, the combination of a cut generation procedure with the branching and bounding techniques yields a full branch-and-cut algorithm. Alternatively, in the case of IBLP, we can view our algorithm as a pure cutting plane algorithm, where a black-box solver provides solutions $(x, y) \in \Omega^I$. Since all variables in our formulation are integer, using cutting planes of the form (2.3), yields a finite algorithm.

**Theorem 2.3** *Let $X = \mathbb{Z}^{n_1}$ and $Y = \mathbb{Z}^{n_2}$. Suppose the cutting plane algorithm described above is implemented using only cuts of the form (2.3). Then, the algorithm finds an optimal solution or shows that (IBLP) is infeasible.*

**Proof.** For the cutting plane algorithm to be finite, we require two conditions. First, the polyhedral region $\Omega$ must be bounded. This is taken care of by assumption. Suppose,

$$\{(x,y) \in X \times Y \mid A^1x \geq b^1, A^2x + G^2y \geq b^2 \subseteq \{(x,y) \in X \times Y \mid \sum_{j \in N_1} x_j + \sum_{j \in N_2} y_j \leq k\}$$

where $N_1 = \{1, \ldots, n_1\}$ and $N_2 = \{1, \ldots, n_2\}$, for some suitably large integer $k$. This implies that the cardinality of $\Omega^I$ is finite (i.e., the number of feasible points $(x,y) \in \Omega \cap (\mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2})$ is finite) Second, the black-box integer programming solver used to solve the relaxation

$$\min_{(x,y) \in \Omega^I} c^1x + d^1y \tag{2.4}$$

must terminate after a finite number of iterations. But, this also follows from our assumption of boundedness and definition of $X$ and $Y$. Because the number of integer points in $\Omega$ is finite, we can find an optimal solution to, or prove infeasibility for, any ILP subproblem in finite time using complete enumeration (i.e., pure branch and bound). Further, since, by definition, each application of the cutting plane procedure either returns a bilevel feasible solution, or cuts off the current point, it must be called a finite number of times. The result follows. □

Note that our algorithm can be also used as pure branch-and-bound, if branching is performed on integer variables, and will be finite as long as $\Omega$ is bounded.

In Section 2.3, we describe the implementation of our algorithm, and the resulting solver, MibS, and provide some computational results. First, however, we introduced some algorithmic enhancements designed to improve the algorithm's performance.

## 2.2 Additional Components

The procedures described in this dissertation provide the foundations for development of a full algorithmic framework. However, it is well-known that the addition of algorithmic enhancements, such as primal heuristics and preprocessing techniques, can greatly improve the performance of a standard algorithm for ILPs. A full set of generalized methodologies for IBLPs requires the addition of heuristic and preprocessing techniques paralleling those that have been developed for solving ILPs over the last two decades. Our preliminary experimentation has shown that such methodologies have the potential to provide significant computational improvement. Further, specialized methods exploiting problem-specific structure can be used to tailor an algorithm and increase its effectiveness on a particular problem class. In this chapter, we describe some heuristic and preprocessing techniques for the general version of IBLP. In the Chapter 4, we demonstrate methods for incorporating

problem-specific customizations into our solver framework for problems with binary upper-level variables.

### 2.2.1 Primal Heuristics

For IBLPs, it is quite straightforward to develop methods for finding feasible solutions. For example, we can simply find a vector that satisfies the upper-level constraints and solve the resulting lower-level ILP to get a bilevel feasible solution. In fact, this is exactly how solutions are obtained from the bilevel feasibility check discussed in this chapter. However, our initial computational experience suggested that these solutions are not of high quality with respect to the upper-level objective. This is not surprising, since the lower-level objective is not typically included in the relaxation problems used in these algorithms. Further, it is unlikely that fixing the upper-level solution arbitrarily will improve upon the solutions generated as by-products of the bilevel feasibility check. Thus, we consider methods for improving upon the solutions obtained in this manner, as well as external techniques to generate good feasible solutions throughout the course of the algorithm.

**Improving Objective Cut.** Each time a bilevel feasibility check is performed, a feasible solution is generated. Formally, one way to determine whether a vector pair $(\hat{x}, \hat{y}) \in \Omega^I$ (i.e., an integer solution to (LR)) is bilevel feasible is by solving the lower-level problem

$$\min_{y \in \mathcal{S}_L(\hat{x}) \cap Y} d^2 y. \tag{2.5}$$

This yields the bilevel feasible solution $(\hat{x}, y^*)$, where $y^*$ is optimal for (2.5). Of course, if we discover that $d^2\hat{y} = d^2 y^*$, then $\hat{y} \in \operatorname{argmin}\{d^2 y \mid y \in (\mathcal{S}_L(\hat{x}) \cap Y)\}$ and we potentially have two feasible solutions (i.e., if $\hat{y} \neq y^*$). However, although $(\hat{x}, \hat{y})$ is optimal for

$$\min_{(x,y) \in \Omega^I} c^1 x + d^1 y, \tag{2.6}$$

we have no guarantee on the quality of $(\hat{x}, y^*)$ with respect to the upper-level objective. Because, the lower-level objective does not appear anywhere in (2.6), the solutions will, in general, be far from bilevel feasible (i.e., $d^2\hat{y} \gg d^2 y^*$).

In order to improve solutions to the relaxation problem in this sense, we can use local search methods. Using the information gained from the bilevel feasibility check, we add the cut

$$d^2 y \leq d^2 y^*$$

to the current relaxation and reoptimize over

$$\Omega^I \cap \{(x, y) \in (X \times Y) \mid d^2y \leq d^2y^*\}.$$

This method attempts to drive the solution towards bilevel feasibility, while maintaining its quality with respect to the upper-level objective. It is important to note solutions to this new problem are not guaranteed to be bilevel feasible, and a second bilevel feasibility check is necessary. Due to the nature of the bilevel feasibility check, we are always guaranteed to generate a feasible solution using this method. In practice, computational experimentation should be performed to ensure that the increase in time necessary to solve the additional ILPs warrants the use of this heuristic. However, if the lower-level problems are relatively small, the additional time should be minimal.

**Lower-level Priority.** Another method for driving solutions towards feasibility is to give temporary priority to the lower-level DM. In other words, we can attempt to find solutions $(x, y) \in \Omega^I$ such that $y \in M^I(x)$ by replacing the upper-level objective with that of the lower-level DM and optimizing over $\Omega^I$. Formally, this method is based on solving problems of the form:

$$\min_{(x,y)\in\Omega^I} d^2y. \tag{2.7}$$

Here, we are essentially allowing the lower-level DM to choose the upper-level decision $x$ that is best with respect to the lower-level objective. Solutions to (2.7) are guaranteed to be feasible. However, it is unlikely that these solutions will be good with respect to the upper-level objective. Thus, we must again consider methods for balancing feasibility and upper-level optimality. In order to improve these solutions, we can add cuts of the form

$$c^1x + d^1y \leq L$$

to (2.7), where $L$ is the value of the best known feasible solution. Note that once cuts are added to the original set of linear constraints, we are not guaranteed feasibility because we have added an unnatural restriction to the lower-level problem. Thus, we must test for bilevel feasibility after resolving. Again, performing this check guarantees that we will eventually generate a feasible solution using this heuristic.

### 2.2.2 Preprocessing Techniques

Preprocessing methods have proved quite useful for decreasing the computational effort required for solving difficult ILPs (see, e.g., Savelsbergh (1994)). The methods employed in the ILP literature can be directly applied to the relaxation $\max_{(x,y)\in\Omega^I} c^1x + d^1y$ to speed up the generation of

integer solutions. However, in order to be truly effective, preprocessing techniques should be tailored to account for the special structure of IBLP. The generalization of known MILP preprocessing techniques, as well as the creation of novel techniques developed specifically for IBLP is an important area of future research. We describe one method that allows us to fix variables before entering the branch-and-bound phase of our algorithm here. This technique utilizes information from the optimal basis of the LP relaxation

$$\min_{(x,y)\in\Omega} c^1 x + d^1 y, \tag{LR}$$

In a manner similar to that used for MILPs, we fix variables based on this basis information and a known bound for IBLP.

Reduced cost fixing is a well-known method used in LP-based branch and bound for MILPs. Let $z_{LP}$ be the current value of the LP relaxation (i.e., the optimal value of (LR) at the current tree node), $L$ the best known solution, and $\bar{c}_j$ the reduced cost of some nonbasic variable $j$. Then, if

$$| \bar{c}_j | \geq L - z_{LP},$$

variable $j$ can be fixed to its current value. The same method can be applied for IBLP branch and bound, since we use an LP relaxation method. However, as noted in Atamturk et al. (1995), this method is very dependent on the quality of the relaxation and incumbent solution. As we have discussed, the bound obtained by (LR) is fairly weak. The addition of cutting planes will improve the quality of this bound, but we still do not expect reduced cost fixing to yield results as powerful for IBLPs as those seen in MILP solvers.

## 2.3 Solver Implementation

A primary goal of this dissertation research was the development of an open source package called the Mixed Integer Bilevel Solver (MibS) to be distributed through the Computational Infrastructure for Operations Research (COIN-OR) repository (Lougee-Heimer, 2003). The branch-and-cut algorithm described in Section 2.1.4 was implemented in C++, utilizing standard software components available from COIN-OR. In particular, the implementation uses the COIN-OR High Performance Parallel Search (CHiPPS) described in Xu et al. (2009) to perform the branch and bound, the MILP solver framework BLIS (part of CHiPPS), the COIN-OR LP Solver (CLP) for solving the LPs that arise in branch and cut, the COIN-OR Branch and Cut for solving the lower-level ILPs, the Cut Generation Library (CGL) for generating cutting planes, and the Open Solver Interface (OSI) for interfacing with CHiPPS and CBC.

### 2.3.1 Class Description

The primary classes the comprise MibS are as follows:

- `MibSModel`: The model class is derived from the virtual BLIS class `BlisModel` and stores information about the original problem.
- `MibSCutGenerator`: The cut generation class is derived from the virtual BLIS class `BlisConGenerator`, and is used to generate cuts of the form described in Section 2.1.2, when CHiPPS finds integer, bilevel infeasible solutions.
- `MibSSolution`: The solution class is derived from the virtual BLIS class `BlisSolution` and is used to store and print integer bilevel feasible solutions.
- `MibSBilevel`: The bilevel class is specific to MibS and is used to transform solutions returned by CHiPPS into a format convenient for our setting. The main function of `MibSBilevel` is to test bilevel feasibility of given solution, by solving the lower-level problem at a fixed upper-level solution.
- `MibSHeuristic`: This class is also specific to MibS and is generate heuristic solutions to improve the lower bound and increase the algorithm's speed.

An effort has been made to keep this framework as general as possible, allowing for easy introduction of enhancements generated from future research. In particular, it is quite easy for users to add their own heuristics, preprocessing methods, and cutting planes. In addition, the manner through which one defines solver parameters is intuitive and designed to make problem-specific tuning straightforward. As stated above, MibS will be made available to the community via the COIN-OR repository. The first release of MibS includes the branch-and-cut algorithm of Section 2.1.4, as well as the algorithmic enhancements described thereafter. The specialized methods provided in Chapter 4 are also part of MibS, and demonstrate how one might customize the solver a particular problem structure. We also intend to include the test sets described below, in order to make replication of our results and comparison with future solvers relatively easy.

### 2.3.2 Practical Assumptions

In Chapter 1, we made two basic assumptions to guarantee the problem was well posed and has a solution. These assumptions were made to ease the exposition, but may be prohibitive in practice. We discuss methods for relaxing them here.

The first of these assumptions is that the feasible region $\mathcal{F}^I$ is nonempty and compact. This guarantees that a solution to (IBLP) exists. However, checking such the checking the validity of this

assumption is not straightforward. In particular, no methods for determining if $\mathcal{F}^I$ is closed are readily available. Thus, we must consider the possibility that a problem fails to satisfy this assumption and escapes detection. A common method for overcoming this type difficulty is to replace $\mathcal{F}^I$ with its closure. This is the method we employ in our implementation. That is, we effectively, restate (IBLP) as:

$$\min_{(x,y)\in \mathrm{cl}(\mathcal{F}^I)} c^1 x + d^1 y.$$

The second assumption made ensures that the lower-level DM will have some room to react for each $x \in X$. However, as discussed in Chapter 1, this may be restrictive in certain applications. In fact, for applications in which the DMs are in direct opposition, creating an infeasible lower-level problem may be the primary goal of the upper-level DM. One way to relax this is to use the standard convention

$$\min\{d^2 y \mid y \in \mathcal{S}_L(x) \cap Y\} = \infty \quad \text{if} \quad \mathcal{S}(x) \cap Y = \emptyset.$$

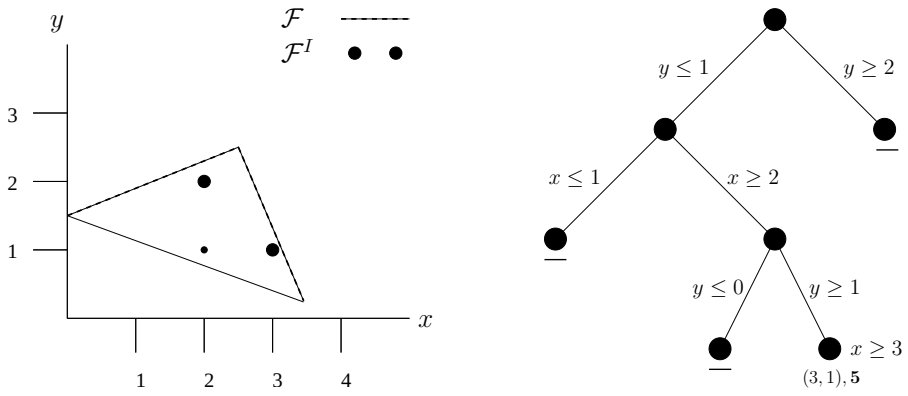
This, in turn, results in $z_{IBLP} = -\infty$ (if $d^1 = -d^2$), as desired. We can to implement this convention in practice is by introducing an artificial variable to capture the infeasibility of the lower-level problem. By assigning this variable a sufficiently large (small) objective function value, we can entice the upper-level to choose $x$ such that the resulting lower-level problem is feasible (infeasible). This is similar to the "big-M" method used in finding initial bases for LPs (Bertsimas and Tsitsiklis, 1997). This is not currently a built-in feature of MibS, since it may not be suitable for all applications. However, it is quite easy to modify a bilevel model to include this artificial variable before reading it into the solver. If this is done, MibS will yield the appropriate solution. In the current version, the solver implicitly assumes this assumption is satisfied since candidate solutions are obtained from the LP relaxation.

## 2.4 Computational Results

### 2.4.1 Illustrative Instances

To our knowledge, the only other general IBLP algorithm proposed in the literature has been that of Moore and Bard (1990) [1]. We do not have the test set of Moore and Bard (1990) or an implementation of their algorithm available, so a comprehensive comparison to their algorithm is not feasible. In order to provide *some* basis for comparison, we did examine the branch-and-cut tree constructed by our algorithm on the examples given in their original paper. The results below reflect the *vanilla* version of MibS, absent of the algorithmic enhancements described above.

[1] The algorithm of Moore and Bard (1990) is capable of solving mixed integer problems, as well, but provides a nice comparison nonetheless.




(a) The feasible region of Example 4.

(b) The resulting branch-and-cut tree.

Figure 2.3: Illustrating the implementation on Example 4

**Example 4.**

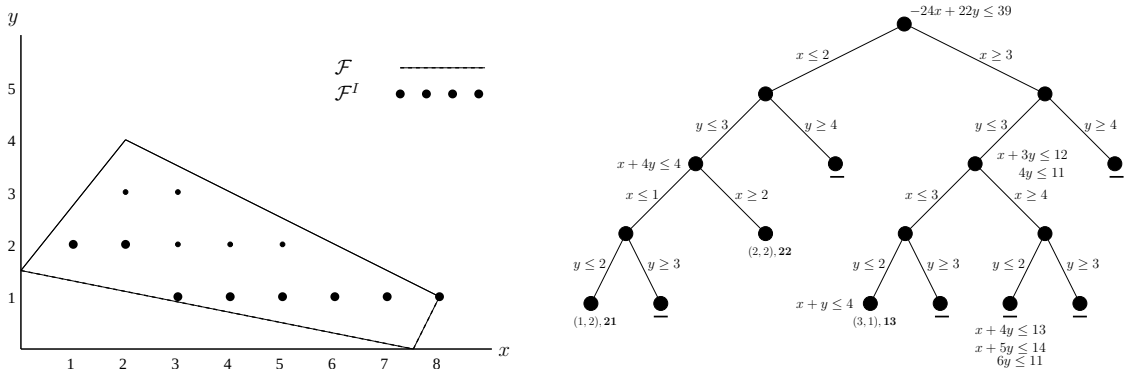
$$\begin{aligned} \min_{x \in \mathbb{Z}_+} \quad & x + 2y \\ \text{subject to} \quad & y \in \operatorname{argmin} \{ y : -x + 2.5y \leq 3.75 \\ & \qquad x + 2.5y \geq 3.75 \\ & \qquad 2.5x + y \leq 8.75 \\ & \qquad y \in \mathbb{Z}_+ \} \end{aligned} \tag{2.8}$$

The feasible region of the IBLP (2.8) and our branch-and-cut tree are shown in Figure 2.3. In this simple case, our algorithm generated a total of seven nodes, and processed five, while the same example in the paper of Moore and Bard (1990) required twelve nodes. Of course, this comparison is only a single instance, but examination of the two search trees does provide some evidence for our intuition that certain aspects of Moore and Bard's algorithm, such as the requirement to branch on integer variables, result in a less efficient search.

**Example 5.** We also tested our algorithm on the the IBLP examined in Chapter 1 from Moore and Bard (1990):

$$\begin{aligned} \min_{x \in \mathbb{Z}_+} \quad & -x - 10y \\ \text{subject to} \quad & y \in \operatorname{argmin} \{ y : 25x - 20y \geq -30 \\ & \qquad -x - 2y \geq -10 \\ & \qquad -2x + y \geq -15 \\ & \qquad 2x + 10y \geq 15 \\ & \qquad y \in \mathbb{Z}_+ \} . \end{aligned}$$

The branch-and-cut tree resulting from their algorithm was not provided, so we are unable to perform a comparison as above. However, for illustration purposes, the feasible region and the resulting branch-and-cut tree is shown in Figure 2.4.




(a) The feasible region of Example 5.

(b) The resulting branch-and-cut tree.

Figure 2.4: Illustrating the implementation on Example 5

### 2.4.2 Problem Generation and Results

In this section, we describe the computational experiments performed and the results obtained using our solver implementation. These results demonstrate both the difficulty of the problems discussed in this dissertation, as well as the benefits of our algorithmic framework. To our knowledge, a standard test bed for IBLPs does not exist. Thus, in order to test our algorithm it was necessary to derive such a test bed. As previously mentioned, all instances used for our tests will be included as part of the first MibS release through COIN-OR. Below, we present the results of our algorithm on a set of randomly-generated IBLP problem instances. Then, in Chapter 4, we examine the computational benefit of applying the specialized interdiction methods via three variants of (MIPINT).

To create the IBLP instances, we first created ILP instances with two objectives and then randomly chose a set of lower-level columns. All coefficients were chosen in the range $[-50, 50]$, and it was assumed that all ILP rows were controlled by the lower-level DM (as in the examples from Moore and Bard (1990)). Five tests sets were created, categorized by the number of *total columns*, *total rows*, and *lower-level columns*. The test sets are summarized in Table 2.1.

| Problem Class | Num Rows | Num Cols | Num Lower |
|---|---|---|---|
| 1 | 20 | 20 | 5 |
| 2 | 20 | 20 | 10 |
| 3 | 20 | 20 | 15 |
| 4 | 30 | 20 | 10 |
| 5 | 40 | 20 | 10 |

Table 2.1: IBLP Instance Class Description.

The results obtained using the *vanilla* version of MibS (i.e., without any algorithmic enhancements), as well as those obtained after employing the primal heuristics of Section 2.2.1, are summarized in Table 2.2. In the table, we denote the number of instances solved within our limit of 30000 CPU seconds and the average optimality gap of those instances for which optimality was not proven by *No. Optimal* and *Avg. Gap (%)*, respectively. Also shown are the average number of nodes in our search tree and the number of bilevel feasibility cuts of the form (2.3) generated during the search, denoted *Avg. No. Nodes* and *No. Cuts*, respectively. Note that only cuts of the form (2.3) were used in these experiments—no generic MILP cuts were used. Finally, in the column titled *Avg. CPU (s)*, we provide the average CPU time required for those instances solved to optimality. All computational tests were performed on an AMD Opteron Processor 6128 with 32GB of memory.

| | No. Optimal | | Avg. Gap (%) | | Avg. No. Nodes | | Avg. No. Cuts | | Avg. CPU (s) | |
|---|---|---|---|---|---|---|---|---|---|---|
| Class | Vanilla | Full | Vanilla | Full | Vanilla | Full | Vanilla | Full | Vanilla | Full |
| 1 | 9 | 9 | 25.10 | 25.04 | 116755.40 | 104156.40 | 50168.20 | 43879.70 | 128.31 | 198.23 |
| 2 | 5 | 5 | 31.88 | 34.49 | 462590.60 | 294798.30 | 229046.50 | 135170.30 | 1596.71 | 3787.24 |
| 3 | 2 | 2 | 46.61 | 49.45 | 479245.90 | 231665.80 | 285278.40 | 132728.10 | 166.73 | 600.82 |
| 4 | 5 | 5 | 61.32 | 61.55 | 439927.80 | 286957.90 | 235215.60 | 153211.00 | 771.89 | 3101.58 |
| 5 | 6 | 6 | 24.34 | 25.28 | 347703.70 | 190189.10 | 139005.20 | 79750.50 | 108.47 | 414.09 |

Table 2.2: Comparison of results with and without heuristic methods.

From this table, we observe that the ability of our solver to find an optimal solution appears to be dependent on the percentage of lower-level columns in an instance. In each of the tables, we can see that the solver was able to prove optimality within the time limit for only 2 out of 10 instances in Test Set 3, the test set for which this percentage is highest. On the other hand, in Test Set 1, the percentage of lower-level columns is lowest, and both versions of the solver were able to prove optimality 9 out of 10 times. When one compares the results of Test Sets 1, 2, and 3, this relationship becomes even more evident. Each of these sets has 20 total rows and columns, but the number of

lower-level columns increases by five from Test Set 1 to Test Set 2, and again from Test Set 2 to Test Set 3. From the results, we can see that the number of problems solved to optimality decreases as this number increases. Further evidence of this relationship is found in the results of Test Sets 2, 4, and 5, each of which has 10 out of 20 lower-level columns. One might expect to see the problem difficulty increase with the number of total rows, but these results appear to indicate otherwise, since roughly the same number of problems were solved in each of these sets.

In addition, we observe that the addition of our heuristic methods significantly reduces the average number of tree nodes and cuts required to solve the IBLPs. On the other hand, the full version of MibS is not able to solve any more instances to optimality than the vanilla version and, on average, requires more computing time. The additional computing time is likely a direct result of the time required to find the heuristic solutions. Assuming this is the case, one might question whether the reduction in nodes and cuts required results simply because the additional time required for the heuristics prevents the solver from proceeding as quickly and, thus, generating as many tree nodes and feasibility cuts within the time allotted. However, in Table 2.3, we provide the relevant results for those instances for which optimality was proven, and we can see the observation still holds. The fact that the full version of MibS is not able to provide more optimal solutions or a significant change in optimality gap is evidence that improved cutting plane methods are required for solving larger instances. In Chapter 4 we derive specialized methods for interdiction problems, but the development of methods for general IBLPs and MIBLPs is an essential area of future work.

| | No. Optimal | | Avg. No. Nodes | | Avg. No. Cuts | |
|---|---|---|---|---|---|---|
| Class | Vanilla | Full | Vanilla | Full | Vanilla | Full |
| 1 | 9 | 9 | 143309.25 | 127569.25 | 3877.22 | 3909.33 |
| 2 | 5 | 5 | 484782.60 | 3200059.40 | 62199.00 | 62093.40 |
| 3 | 2 | 2 | 422551.00 | 311383.00 | 9290.50 | 9299.00 |
| 4 | 5 | 5 | 515762.60 | 336843.20 | 21359.00 | 21317.00 |
| 5 | 6 | 6 | 281901.17 | 132110.50 | 3543.67 | 3527.83 |

Table 2.3: Comparison on instances solved to optimality.

The complete set of results for the vanilla and full versions of MibS are provided in Tables 2.4 and 2.5, respectively. From these tables, we can see that the time required to solve the instances in our test set is quite volatile. Many of the instances failed to reach optimality within the allotted time, while others were solved in less than a minute. It is likely that this wide range is a result of the way in which we generated our instances—because the coefficients of the upper and lower objective functions were chosen randomly, no control was exerted over the degree to which they coincided. This choice was deliberate, since we sought to test the performance of our solver on a generic IBLP. However, determining how the relationship between the two objective functions affects the difficulty of an MIBLP is an interesting area of future work. We consider this issue for certain special classes in the following chapter, but do not address the general case.

| Instance | Obj. Value | No. Nodes | Depth | Gap (%) | No. Cuts | CPU (s) |
|---|---|---|---|---|---|---|
| miblp-20-20-50-0110-5-1 | -548 | 20621 | 30 | — | 3846 | 60.62 |
| miblp-20-20-50-0110-5-2 | -558 | 913215 | 45 | 25.10 | 466498 | LIM |
| miblp-20-20-50-0110-5-3 | -477 | 101 | 10 | — | 3 | 0.14 |
| miblp-20-20-50-0110-5-4 | -753 | 187 | 14 | — | 3 | 0.22 |
| miblp-20-20-50-0110-5-5 | -392 | 97 | 17 | — | 3 | 0.11 |
| miblp-20-20-50-0110-5-6 | -1061 | 232185 | 38 | — | 31235 | 1091.91 |
| miblp-20-20-50-0110-5-7 | -547 | 213 | 17 | — | 35 | 0.35 |
| miblp-20-20-50-0110-5-8 | -936 | 271 | 18 | — | 6 | 0.32 |
| miblp-20-20-50-0110-5-9 | -877 | 205 | 16 | — | 5 | 0.24 |
| miblp-20-20-50-0110-5-10 | -340 | 459 | 24 | — | 48 | 0.85 |
| miblp-20-20-50-0110-10-1 | -353 | 741172 | 46 | 47.01 | 475682 | LIM |
| miblp-20-20-50-0110-10-2 | -659 | 5019 | 32 | — | 937 | 15.82 |
| miblp-20-20-50-0110-10-3 | -618 | 45449 | 39 | — | 6156 | 120.31 |
| miblp-20-20-50-0110-10-4 | -597 | 775159 | 44 | 25.66 | 291809 | LIM |
| miblp-20-20-50-0110-10-5 | -1003 | 31 | 8 | — | 4 | 0.06 |
| miblp-20-20-50-0110-10-6 | -672 | 586626 | 48 | 26.22 | 407028 | LIM |
| miblp-20-20-50-0110-10-7 | -618 | 827234 | 50 | 36.85 | 457158 | LIM |
| miblp-20-20-50-0110-10-8 | -667 | 75329 | 38 | — | 17236 | 997.46 |
| miblp-20-20-50-0110-10-9 | -256 | 703953 | 39 | — | 286662 | 6849.91 |
| miblp-20-20-50-0110-10-10 | -429 | 865934 | 46 | 23.64 | 347793 | LIM |
| miblp-20-20-50-0110-15-1 | -289 | 631835 | 45 | 60.64 | 422190 | LIM |
| miblp-20-20-50-0110-15-2 | -645 | 686790 | 59 | 23.22 | 421883 | LIM |
| miblp-20-20-50-0110-15-3 | -593 | 482219 | 44 | 20.21 | 159567 | LIM |
| miblp-20-20-50-0110-15-4 | -396 | 378201 | 49 | 36.42 | 287079 | LIM |
| miblp-20-20-50-0110-15-5 | -75 | 233108 | 54 | 90.11 | 167495 | LIM |
| miblp-20-20-50-0110-15-6 | -596 | 677582 | 54 | 40.38 | 468935 | LIM |
| miblp-20-20-50-0110-15-7 | -471 | 855197 | 41 | 27.99 | 360242 | LIM |
| miblp-20-20-50-0110-15-8 | -242 | 798795 | 62 | 73.87 | 546812 | LIM |
| miblp-20-20-50-0110-15-9 | -584 | 46307 | 33 | — | 18137 | 324.33 |
| miblp-20-20-50-0110-15-10 | -251 | 2425 | 26 | — | 444 | 9.12 |
| miblp-30-20-50-0110-10-1 | -471 | 9533 | 30 | — | 984 | 22.05 |
| miblp-30-20-50-0110-10-2 | -478 | 84885 | 36 | — | 19902 | 770.26 |
| miblp-30-20-50-0110-10-3 | -678 | 801376 | 48 | 23.28 | 485021 | LIM |
| miblp-30-20-50-0110-10-4 | 207 | 792137 | 44 | 178.03 | 460991 | LIM |
| miblp-30-20-50-0110-10-5 | -135 | 3 | 1 | — | 0 | 0.01 |
| miblp-30-20-50-0110-10-6 | -171 | 973272 | 42 | 60.41 | 695335 | LIM |
| miblp-30-20-50-0110-10-7 | -375 | 355485 | 38 | — | 85303 | 3055.75 |
| miblp-30-20-50-0110-10-8 | -461 | 578398 | 43 | 16.33 | 189672 | LIM |
| miblp-30-20-50-0110-10-9 | -672 | 801446 | 48 | 28.56 | 414342 | LIM |
| miblp-30-20-50-0110-10-10 | -168 | 2743 | 27 | — | 606 | 11.38 |
| miblp-40-20-50-0110-10-1 | -198 | 265 | 19 | — | 47 | 0.73 |
| miblp-40-20-50-0110-10-2 | -120 | 738229 | 49 | 75.44 | 436226 | LIM |
| miblp-40-20-50-0110-10-3 | -675 | 56779 | 37 | — | 10051 | 409.27 |
| miblp-40-20-50-0110-10-4 | -270 | 13153 | 29 | — | 2952 | 52.75 |
| miblp-40-20-50-0110-10-5 | -537 | 697 | 19 | — | 29 | 1.52 |
| miblp-40-20-50-0110-10-6 | -425 | 4997 | 29 | — | 1207 | 25.00 |
| miblp-40-20-50-0110-10-7 | -975 | 811699 | 47 | 14.55 | 394306 | LIM |
| miblp-40-20-50-0110-10-8 | -849 | 945827 | 42 | 4.15 | 339321 | LIM |
| miblp-40-20-50-0110-10-9 | -800 | 879806 | 40 | 3.22 | 198937 | LIM |
| miblp-40-20-50-0110-10-10 | -398 | 25585 | 29 | — | 6976 | 161.56 |

Table 2.4: Results from IBLPs without heuristic methods.

| Instance | Obj. Value | No. Nodes | Depth | Gap (%) | No. Cuts | CPU (s) |
|---|---|---|---|---|---|---|
| miblp-20-20-50-0110-5-1 | -548 | 20573 | 31 | — | 3845 | 142.24 |
| miblp-20-20-50-0110-5-2 | -561 | 788371 | 45 | 25.04 | 403902 | LIM |
| miblp-20-20-50-0110-5-3 | -477 | 101 | 10 | — | 3 | 0.19 |
| miblp-20-20-50-0110-5-4 | -753 | 187 | 14 | — | 3 | 0.27 |
| miblp-20-20-50-0110-5-5 | -392 | 97 | 17 | — | 3 | 0.18 |
| miblp-20-20-50-0110-5-6 | -1061 | 231109 | 38 | — | 30949 | 1636.73 |
| miblp-20-20-50-0110-5-7 | -547 | 213 | 17 | — | 35 | 1.35 |
| miblp-20-20-50-0110-5-8 | -936 | 271 | 18 | — | 6 | 0.37 |
| miblp-20-20-50-0110-5-9 | -877 | 205 | 16 | — | 5 | 0.32 |
| miblp-20-20-50-0110-5-10 | -340 | 437 | 24 | — | 46 | 2.42 |
| miblp-20-20-50-0110-10-1 | -321 | 330155 | 46 | 53.16 | 208459 | LIM |
| miblp-20-20-50-0110-10-2 | -659 | 5001 | 32 | — | 937 | 44.87 |
| miblp-20-20-50-0110-10-3 | -618 | 45445 | 39 | — | 6157 | 309.46 |
| miblp-20-20-50-0110-10-4 | -597 | 506129 | 44 | 26.45 | 188933 | LIM |
| miblp-20-20-50-0110-10-5 | -1003 | 35 | 10 | — | 6 | 0.85 |
| miblp-20-20-50-0110-10-6 | -657 | 151122 | 45 | 28.97 | 103710 | LIM |
| miblp-20-20-50-0110-10-7 | -657 | 692239 | 48 | 33.16 | 353899 | LIM |
| miblp-20-20-50-0110-10-8 | -667 | 66773 | 38 | — | 15872 | 3637.47 |
| miblp-20-20-50-0110-10-9 | -256 | 710493 | 39 | — | 287495 | 14943.56 |
| miblp-20-20-50-0110-10-10 | -405 | 440591 | 45 | 30.72 | 186235 | LIM |
| miblp-20-20-50-0110-15-1 | -234 | 201617 | 43 | 69.21 | 123930 | LIM |
| miblp-20-20-50-0110-15-2 | -645 | 423894 | 57 | 23.63 | 256045 | LIM |
| miblp-20-20-50-0110-15-3 | -593 | 99818 | 43 | 23.71 | 39409 | LIM |
| miblp-20-20-50-0110-15-4 | -323 | 85857 | 47 | 49.83 | 57090 | LIM |
| miblp-20-20-50-0110-15-5 | -75 | 32545 | 46 | 90.46 | 23278 | LIM |
| miblp-20-20-50-0110-15-6 | -596 | 287340 | 50 | 41.73 | 197862 | LIM |
| miblp-20-20-50-0110-15-7 | -471 | 560396 | 40 | 29.34 | 232520 | LIM |
| miblp-20-20-50-0110-15-8 | -301 | 576523 | 61 | 67.69 | 378549 | LIM |
| miblp-20-20-50-0110-15-9 | -584 | 46243 | 33 | — | 18154 | 1164.63 |
| miblp-20-20-50-0110-15-10 | -251 | 2425 | 26 | — | 444 | 37.01 |
| miblp-30-20-50-0110-10-1 | -471 | 8283 | 31 | — | 776 | 38. 68 |
| miblp-30-20-50-0110-10-2 | -478 | 85005 | 36 | — | 19902 | 2950.18 |
| miblp-30-20-50-0110-10-3 | -678 | 609115 | 46 | 23.59 | 331295 | LIM |
| miblp-30-20-50-0110-10-4 | 207 | 365112 | 43 | 173.61 | 213973 | LIM |
| miblp-30-20-50-0110-10-5 | -135 | 3 | 1 | — | 0 | 0.00 |
| miblp-30-20-50-0110-10-6 | -171 | 766072 | 42 | 61.24 | 554528 | LIM |
| miblp-30-20-50-0110-10-7 | -375 | 349685 | 38 | — | 85301 | 9422.16 |
| miblp-30-20-50-0110-10-8 | -461 | 141237 | 42 | 20.21 | 48691 | LIM |
| miblp-30-20-50-0110-10-9 | -672 | 542324 | 47 | 29.10 | 277038 | LIM |
| miblp-30-20-50-0110-10-10 | -168 | 2743 | 27 | — | 606 | 33.96 |
| miblp-40-20-50-0110-10-1 | -198 | 265 | 19 | — | 47 | 2.45 |
| miblp-40-20-50-0110-10-2 | -117 | 408981 | 47 | 76.52 | 241787 | LIM |
| miblp-40-20-50-0110-10-3 | -675 | 56311 | 37 | — | 9956 | 1649.93 |
| miblp-40-20-50-0110-10-4 | -270 | 12883 | 29 | — | 2952 | 132.49 |
| miblp-40-20-50-0110-10-5 | -537 | 695 | 19 | — | 29 | 3.96 |
| miblp-40-20-50-0110-10-6 | -425 | 4997 | 29 | — | 1207 | 104.67 |
| miblp-40-20-50-0110-10-7 | -1028 | 508318 | 45 | 10.81 | 223631 | LIM |
| miblp-40-20-50-0110-10-8 | -830 | 572806 | 42 | 7.44 | 229471 | LIM |
| miblp-40-20-50-0110-10-9 | -797 | 311050 | 41 | 6.33 | 81449 | LIM |
| miblp-40-20-50-0110-10-10 | -398 | 25585 | 29 | — | 6976 | 591.02 |

Table 2.5: Results from IBLPs with heuristic methods.

# Chapter 3

# Mixed Integer Bilevel Linear Programming

In the previous chapter, we described an algorithmic framework for IBLP. In theory, a branch-and-cut framework could be used to solve the more general problem of MIBLP, if one were able to derive suitable cutting plane methods. However, the results given in Chapter 2 rely heavily on the assumption that all decision variables are integral and, thus, are not applicable to MIBLP. Deriving cutting plane methods for the general case appears to be a much less straightforward endeavour. It is clear, however, that the general form allows us to capture a much wider range of applications and, thus, understanding this problem is an area of important research. Towards this end, we consider the general MIBLP problem in this chapter. We first review known results on complexity, provide some new results on the general problem and interesting special cases, and place MIBLP in the overall complexity landscape. Then, we utilize duality theory and value function methods to derive several single-level reformulations that can be solved via direct methods or provide insight into the problem structure. Using this insight, we derive an algorithm for the general MIBLP based on iterative approximations of the lower-level value function. Finally, we suggest some heuristic methods that are useful in finding reasonably good solutions to MIBLP with little computational effort.

Recall, from Chapter 1, the canonical instance:

$$z_{MIBLP} = \min_{(x,y)\in\mathcal{F}^I} c^1 x + d^1 y, \tag{MIBLP}$$

where the feasible region $\mathcal{F}^I$ is contained in $X \times Y$, for $X = \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1-p_1}$ and $Y = \mathbb{Z}^{p_2} \times \mathbb{R}^{n_2-p_2}$. In what follows, we often further define our problem by specifying $p_1$ and $p_2$ (i.e., setting $p_1 = n_1$ and $p_2 = n_2$ yields a BLP) and introducing additional variable bounds (i.e., when combined with

integrality constraints, upper bounds of one yield binary restrictions). Rather than providing a complete taxonomy of all cases, however, we choose to note particular special cases that have convenient properties for reformulation or algorithm design. The reader is referred to the work of Vicente et al. (1996), Dempe (2001), and H. Gümüş and Floudas (2005) for a comparison among the subclasses defined by different forms for $X$ and $Y$.

## 3.1 Problem Complexity

As noted in Chapter 1, we gain insight into the complexity of MIBLP by considering its well-known special cases. For example, removing integrality restrictions on the variables (i.e., setting $X = \mathbb{R}^{n_1}$ and $Y = \mathbb{R}^{n_2}$) yields a BLP, a known NP−hard problem. Similarly, removing the lower-level variables (i.e., setting $n_2 = 0$) yields an MILP, another known NP−hard problem. Thus, its clear that MIBLP is also NP−hard. However, the computational issues discussed in Chapter 1 taken together with the challenges found in algorithm design suggest that MIBLP is characterized by complexity challenges not shared by its well-known special cases. For this reason, we explore the complexity of the general problem in some depth here, but also focus considerable attention on those lesser-known special cases that have computationally attractive properties. Before beginning this discussion, however, we define the decision problems of several relevant problems and provide some basic results on BLP. For the remainder of this discussion, we assume all data necessary to specify instances of our problems is rational. This implies that we can, in theory, scale all data appropriately and form equivalent problems using only integer data, simplifying the exposition.

The decision versions of several problems relevant to our discussion are defined below. We adopt the notation $\Pi_1 \propto \Pi_2$, from Garey and Johnson (1979), to denote that there exists a polynomial transformation from $\Pi_1$ to $\Pi_2$. Additionally, we let $Y_\Pi$ denote the set of instances for which the answer to the decision problem $\Pi$ is yes.

**BILEVEL LINEAR PROGRAMMING (DBLP)**

INSTANCE: Rational vectors $c^1 \in \mathbb{Q}^{n_1}$, $d^1, d^2 \in \mathbb{Q}^{n_2}$, $b^1 \in \mathbb{Q}^{m_1}$, $b^2 \in \mathbb{Q}^{m_2}$, rational matrices $A^1 \in \mathbb{Q}^{m_1 \times n_1}$, $A^2 \in \mathbb{Q}^{m_2 \times n_1}$, $G^2 \in \mathbb{Q}^{m_2 \times n_2}$, and integer $L \in Z$.

QUESTION: Do there exist vectors $x \in \mathbb{R}^{n_1}_+$ and $y \in \mathbb{R}^{n_2}_+$ such that $A^1 x \geq b^1$, $y \in \operatorname{argmin}\{d^2 y \mid G^2 y \geq b^2 - A^2 x, y \in \mathbb{R}^{n_2}_+\}$, and $c^1 x + d^1 y \leq L$?

**MIXED INTEGER BILEVEL LINEAR PROGRAMMING (DMIBLP)**

INSTANCE: Rational vectors $c^1 \in \mathbb{Q}^{n_1}$, $d^1, d^2 \in \mathbb{Q}^{n_2}$, $b^1 \in \mathbb{Q}^{m_1}$, $b^2 \in \mathbb{Q}^{m_2}$, rational matrices $A^1 \in \mathbb{Q}^{m_1 \times n_1}$, $A^2 \in \mathbb{Q}^{m_2 \times n_1}$, $G^2 \in \mathbb{Q}^{m_2 \times n_2}$, and integer $L \in \mathbb{Z}$.

QUESTION: Do there exist vectors $x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_1-p_1}$ and $y \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})$ such that $A^1x \geq b^1$, $y \in \operatorname{argmin}\left\{d^2y \mid G^2y \geq b^2 - A^2x, y \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})\right\}$, and $c^1x + d^1y \leq L$?

**MIXED INTEGER LINEAR PROGRAMMING (DMILP)**

INSTANCE: Rational vectors $c \in \mathbb{Q}^n$, $b \in \mathbb{Q}^m$, rational matrix $A \in \mathbb{Q}^{m\times n}$, and integer $B \in \mathbb{Z}$.

QUESTION: Does there exist a vector $z \in (\mathbb{Z}_+^p \times \mathbb{R}_+^{n-p})$ such that $Az \geq b$ and $cz \leq B$?

**MIXED INTEGER INTERDICTION (DMIPINT)**

INSTANCE: Rational vectors $d \in \mathbb{Q}^n$, $u \in \mathbb{Q}^n$, $b^1 \in \mathbb{Q}^{m_1}$, $b^2 \in \mathbb{Q}^{m_2}$, rational matrices $A^1 \in \mathbb{Q}^{m_1\times n}$, $G^2 \in \mathbb{Q}^{m_2\times n}$, and integer $L \in \mathbb{Z}$.

QUESTION: Do there exist vectors $x \in \mathbb{B}^n$ and $y \in (\mathbb{Z}_+^p \times \mathbb{R}_+^{n-p})$ such that $A^1x \leq b^1$ and $y \in \operatorname{argmin}\left\{dy \mid G^2y \geq b^2, -y \geq -U(e-x), y \in (\mathbb{Z}_+^p \times \mathbb{R}_+^{n-p})\right\}$, and $dy \geq L$?

**BINARY KNAPSACK PROBLEM (DKNAP)**

INSTANCE: A finite set of items $J$ and, for each item, a size defined by the vector $r \in \mathbb{Z}_+^{|J|}$ and a value defined by $v \in \mathbb{Z}_+^{|J|}$, and positive integers $R \in \mathbb{Z}$ and $B \in \mathbb{Z}$.

QUESTION: Does there exist a subset of $J' \subseteq J$ such that $\sum_{j\in J'} r_j \leq R$ and $\sum_{j\in J'} v_j \geq B$?

A crucial element of our analysis is the fact that LPs can be solved in polynomial time via the *ellipsoid method*. This is formalized in the following result.

**Theorem 3.1 (Khachian (1979))** (LP) *with all integer data can be solved in polynomial time using the ellipsoid method.*

The following are well-known properties of BLP. However, for completeness, we restate them here.

**Theorem 3.2 (Jeroslow (1985))** *BLP is in the complexity class* NP*−hard.*

**Theorem 3.3** *DBLP is in the complexity class* NP.

**Proof.** To show DBLP $\in$ NP, we must show that it can be solved by a nondeterministic polynomial-time algorithm. Let $x \in \mathbb{R}_+^{n_1}$ and $y \in \mathbb{R}_+^{n_1}$ be given. Then, given an instance of DBLP, we can use the following algorithm to check bilevel feasibility. We first check the requirements $c^1x' + d^1y' \leq L$ and $A^1x' \geq b^1$, which can be done in polynomial time. Assuming these are satisfied, we solve the lower-level problem (in polynomial-time via ellipsoid algorithm) with $x = x'$ to yield lower-level solution $y^*$. If $d^2y' = d^2y^*$, then $I \in Y_{DBLP}$. □

Using the result above, and those given in Audet et al. (1997), allows us to establish the following result.

**Theorem 3.4** *DBLP is in the complexity class* NP*-complete.*

This result follows from the fact that DBLP is in the class NP, by Theorem 3.3, and the reformulation from mixed integer binary programming (MIB) given in Audet et al. (1997), which yields a polynomial reduction from the decision version of MIB to DBLP by replacing the optimization problems with their decision counterpart.

It is also well-known, and clear from our discussion above, that MIBLP is NP−hard, but we again include a separate proof for completeness.

**Theorem 3.5** *MIBLP is in the complexity class* NP*-hard.*

**Proof.** Clearly, if we can show that DMIBLP is NP-hard, we will also have shown MIBLP is NP-hard. In order to show DMIBLP is NP-hard, we show that there exists a polynomial reduction from the decision version of MILP (DMILP), a known NP-complete problem, to DMIBLP. Thus, we must show that there exists a function $f$, computable in polynomial time, that maps an instance of DMILP to DMIBLP and such that an instance $I \in Y_{\text{DMILP}}$ if and only if $f(I) \in Y_{\text{DMIBLP}}$. We can define $f$ as follows. Suppose an instance of DMILP is defined by $c \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^m$ and integer $B$. Then, we can define an instance of DMIBLP by setting $L = B$, $A^1 = A$, $b^1 = b$, $c^1 = c$, and all remaining problem parameter matrices and vectors to zero. This yields an instance of DMIBLP with $n_1 = n$ and $n_2 = 0$ and a vacuous lower-level problem. Clearly, this transformation can be completed in polynomial time. Thus, it remains to show that there exists a vector $z \in (\mathbb{Z}^p \times \mathbb{R}^{n-p})$ such that $Az \leq b$ and $cz \leq B$ if and only if there exist vectors $x \in (\mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_1-p_1})$ and $y \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})$ such that $c^1x + d^1y \leq L$. Suppose that $\hat{z} \in (\mathbb{Z}_+^p \times \mathbb{R}_+^{n-p})$ is a vector such that $A\hat{z} \geq b$ and $c\hat{z} \leq B$. Then, setting $x = \hat{z}$ and $y = 0$ yields $A^1x = A\hat{z} \geq b = b^1$. Since the lower-level problem optimality condition is trivially satisfied for all $(x, y)$, the vector $(\hat{z}, 0)$ is feasible for DMIBLP and $c^1x + d^1y = cx = c\hat{z} \leq B = L$. Conversely, suppose $(\hat{x}, \hat{y})$ is a solution to DMIBLP. This implies $A\hat{x} = A^1\hat{x} \geq b^1 = b$ and $c\hat{x} = c^1\hat{x} + d^1\hat{y} \leq L = B$. Thus, DMILP $\propto$ DMIBLP. □

So far, we have established that both BLP and MIBLP are NP−hard. This is neither surprising, nor terribly elucidating. The following result provides more insight into the difference between the continuous and mixed integer problems.

**Theorem 3.6** *Unless* P = NP*, DMIBLP is not in the complexity class* NP.

**Proof.** Suppose $\mathsf{P} \neq \mathsf{NP}$ and, by way of contradiction, that DMIBLP is in $\mathsf{NP}$. This implies that there exists a nondeterministic polynomial time algorithm that solves DMIBLP. This, in turn, implies that there exists a polynomial time algorithm to test the condition

$$y \in \operatorname{argmin}\left\{d^2 y \mid G^2 y \geq b^2 - A^2 x, y \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})\right\}, \tag{3.1}$$

for fixed $x \in (\mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_2-p_2})$. Let $OPT(x)$ denote the optimal lower-level objective value when the upper-level decision is $x$ (i.e., $OPT(x) = d^2 y$ for $y$ satisfying condition (3.1)). Then, this condition can be stated as the following decision problem, which we denote FEASCHECK:

*Does there exist a vector $y \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})$ such that $G^2 y \geq b^2 - A^2 x$ and $d^2 y \leq OPT(x)$?*

We proceed by demonstrating a polynomial reduction from FEASCHECK to the following, known $\mathsf{NP}-$complete decision problem, which is the complement of DMILP:

*Does there exist a vector $z \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})$ such that $Gz \geq b$ and $dz > L$?*

Suppose an instance of this problem is specified by $d \in \mathbb{Q}^{n_2}$, $G \in \mathbb{Q}^{m_2 \times n_2}$, and $b \in \mathbb{Q}^{m_2}$ and $L = -OPT(x) - \epsilon$, for some small $\epsilon > 0$. Then, we can specify FEASCHECK by setting $d^2 = -d$, $G^2 = G$, $b^2 - A^2 x = b$, which can clearly be done in polynomial time. Thus, it remains to show that there exists a vector $z \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})$ such that $Gz \geq b$ and $OPT(x) < dz$ if and only there exists $y \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})$ such that $G^2 y \geq b^2 - A^2 x$ and $d^2 y \leq OPT(x)$. Let $\hat{z} \in (\mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2})$ such that $G\hat{z} \geq b$ and $d\hat{z} > L$ be given. Then, setting $y = \hat{x}$ yields $G^2 y = G\hat{z} \geq b = b^2 - A^2 x$ and $d^2 y = -d\hat{z} < -L = OPT(x) + \epsilon \leq OPT(x)$, which clearly implies the desired condition. Conversely, suppose $\hat{y}$ satisfies the conditions of FEASCHECK. Then, setting $z = \hat{y}$ yields $Gz = G^2\hat{y} \geq b^2 - A^2 x = b$ and $dz = -d^2\hat{y} \geq -OPT(x) = L + \epsilon > L$, as desired. Thus, FEASCHECK is at least as hard as this mixed integer decision problem and, unless $\mathsf{P} = \mathsf{NP}$, we have a contradiction. □

The relationship between the complexity of MIBLP and BLP is illustrated in Figure 3.1. The previous result provides some explanation for the additional difficulty of MIBLP. In order to fully understand the differences in complexity, however, we must employ the notion of *polynomial-time hierarchy* ($\mathsf{P}-$hierarchy), defined in terms of a *nondeterministic oracle Turing machines*, described by Meyer and Stockmeyer (1972) and Stockmeyer (1977). A nondeterministic oracle Turing machine is defined as a nondeterministic Turing machine augmented with an oracle tape. As described in Garey and Johnson (1979), a nondeterministic oracle Turing machine with an oracle for problem

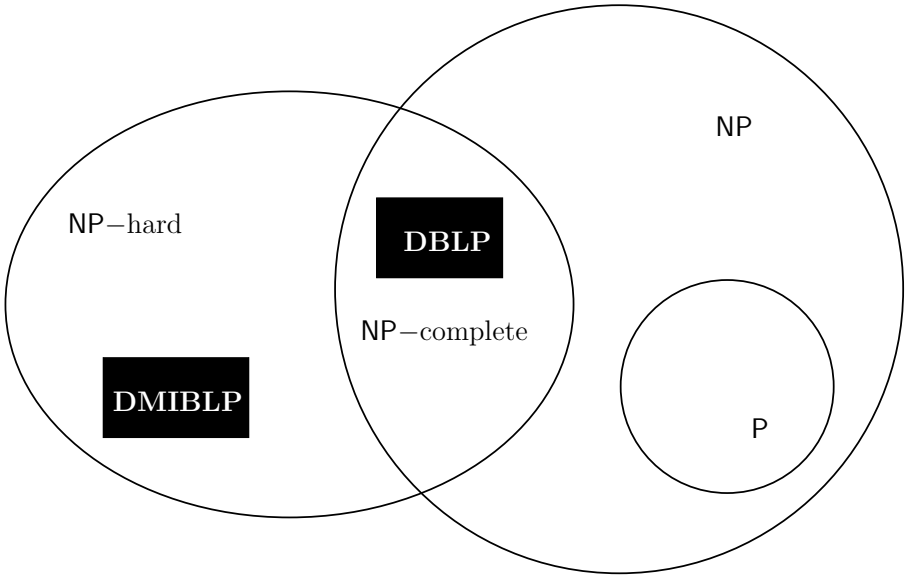



Figure 3.1: The relationship between MIBLP and BLP complexity, assuming $\mathsf{P} \neq \mathsf{NP}$.

$\prod$ can be thought of as a nondeterministic algorithm containing a subroutine for $\prod$ that can be run in constant time. Following the notation in Stockmeyer (1977), let $M(B)$ denote the language accepted by the nondeterministic oracle Turing machine $M$ with oracle $B$. Then, the polynomial hierarchy can be defined as follows.

**Definition 3.1** *The polynomial-time hierarchy (*$\mathsf{P}-$*hierarchy) is* $\left\{\sum_k^p, \prod_k^p, \Delta_k^p : k \geq 0\right\}$*, where*

$$\sum_0^p = \prod_0^p = \Delta_0^p = \mathsf{P};$$

*and for* $k \geq 0$*,*

- $\sum_{k+1}^p = \mathsf{NP}(\sum_k^p)$,
- $\prod_{k+1}^p = co\mathsf{NP}(\sum_k^p)$,
- $\Delta_{k+1}^p = \mathsf{P}(\sum_k^p)$.

*Also, define* $\mathsf{PH} = \bigcup_{k=0}^{\infty} \sum_k^p$.

The $\mathsf{P}-$hierarchy is illustrated in Figure 3.2. In the figure, the darker classes are contained in the lighter classes, and incomparable classes are given the same color. Note that none of these inclusions is known to be strict. We have the following result for the *binary integer bilevel linear program* (BIBLP), which results from (MIBLP) if we set $X = \mathbb{B}^{n_1}$ and $Y = \mathbb{B}^{n_2}$.

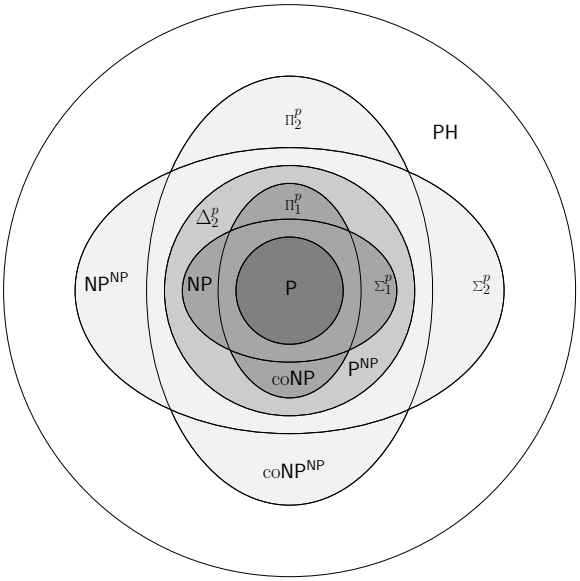



Figure 3.2: The polynomial-hierarchy, assuming $\mathsf{P} \neq \mathsf{NP}$. (Rothe, 2005)

**Theorem 3.7** *DBIBLP is in the complexity class* $\sum_2^p$.

Theorem 3.7 follows from the more general result of Jeroslow (1985), which states that the problem of checking optimality for a $k-$level binary LP is in $\prod_k^p$. Given an upper-level objective value $\bar{z}$ and a candidate solution $\bar{x}$, this problem can be posed as the question "is $OPT \geq \bar{z}$", where $OPT$ denotes the optimal solution of the program, for which the answer is yes when $\bar{x}$ is optimal. We can ask the complementary question "is $OPT < \bar{z}$", for which the answer is no when $\bar{x}$ is optimal. When, $k = 2$, under the assumption of rational data, this is equivalent to DBIBLP. Since $BIBLP$ is a special case of MIBLP, we are able to state the following.

**Theorem 3.8** *DMIBLP is* $\sum_2^p$.

Combining this with the results above on BLP, provides some insight into the complexity of the bilevel programs with integer variables.

We can show that MIPINT is $\mathsf{NP}-$hard using the same method as in the proof of Theorem 3.5. However, in this case, we use a transformation to the decision version of the *binary knapsack problem* instead.

**Theorem 3.9** *MIPINT is in the complexity class* $\mathsf{NP}$*-hard.*

**Proof.** As in the previous proof, it suffices to show that there exists a polynomial reduction from the decision version of the knapsack problem (DKNAP) to DMIPINT. Suppose an instance of DKNAP

is defined as above. Then, we can define an instance of DMIPINT by setting $A^1 = [r \mid 0]^T$, $b^1 = [R \mid 0]$, $d = -v$, $U = \mathbb{I}_{n\times n}$, $G^2 = 0$, and $b^2 = 0$. Thus, we have an instance of MIPINT with $n = |J|$, $m_1 = 1$, and $m_2 = 0$. Also define $L = B - \sum_{j\in J} v_j$. Suppose there exists a subset $J' \subseteq J$ such that $\sum_{j\in J'} r_j \leq R$ and $\sum_{j\in J'} v_j \geq B$. Set $x_j = 1$ for all $j \in J'$ and $x_j = 0$ for all $j \in (J \setminus J')$. Clearly, this is a feasible upper-level solution. Then, the resulting lower-level problem is:

$$\min\{-py \mid y_{(J\setminus J')} \leq 1, y_{J'} = 0, y \in Z^{|J|} \times \mathbb{R}^{|J|-p}\}.$$

Since $v \in \mathbb{Z}^{|J|}$, the optimal lower-level solution is to set $y_j = 1$ for all $j \in (J \setminus J')$, yielding $dy = -\sum_{j\in(J\setminus J')} v_j = B - \sum_{j\in J} v_j = L$. Conversely, suppose there exists a vector $x \in \mathbb{B}^n$ such that $x_j = 1$ for all $j \in K'$ and $x_j = 0$ for all $j \in (K \setminus K')$, for some $K' \subseteq K = \{1, \ldots, n\}$, and $rx \leq R$. Suppose also that there exists a vector

$$y \in \operatorname{argmin}\left\{-vy \mid y_{(K\setminus K')} \leq 1, y_{J'} = 0, y \in Z^{|K|} \times \mathbb{R}^{|(K\setminus K')|}\right\}$$

and $dy \geq L$. If we set $J = K$ and $J' = K'$, clearly the condition $\sum_{j\in J'} r_j \leq R$ is satisfied. Further $\sum_{j\in J'} v_j = \sum_{j\in J} v_j - \sum_{j\in(J\setminus J')} v_j = \sum_{j\in J} v_j + dy \geq \sum_{j\in J} v_j + L = B$. Thus, DKNAP $\propto$ DMIPINT

□

Of course, the fact that the Maximum Shortest Path Problem (MSPP), a known NP−complete problem (Wood, 1993), is a special case of MIPINT implies that MIPINT is NP−hard, but the independent proof may provide additional insight for the reader. From the proof of Theorem 3.6 we gained the intuition that the nature of the lower-level problem is a key component of a bilevel program's complexity. Namely, when checking feasibility requires solution of a MILP, the bilevel program is not in NP, unless P = NP. This intuition holds for MIPINT, as well; interdiction of a MILP (or ILP) is not in NP, but interdiction of an LP is. Below, we see the role the lower-level problem plays in our ability to reformulate the problem and solve it via direct methods.

## 3.2 Reformulations and Exact Solution Methods

It is clear that each of the problems discussed above poses significant algorithm design challenges. Thus, obtaining exact solutions for large instances of such problems will likely require further research or significant solver customization. In the following chapter, we demonstrate methods for solver customization via MIPINT and our solver MibS. Later in this chapter, we consider an alternative approach to finding exact solutions, namely arriving at good solutions quickly via heuristic methods.

In this section, we describe single-level reformulations possible through the application of optimality conditions. Several of the reformulations given can be solved using existing methods, for which solvers are readily available. However, for the general case, the reformulation we derive via optimality conditions yields a problem for which known methods are not applicable. In the second half of this section, we describe how approximations of the value function can lead to practical methods for this case. For each methodology, we examine special cases of the general problem for which a further simplification is possible. Before considering dual reformulations, however, we first describe one fairly trivial special case for which well-known methods can immediately be applied.

### 3.2.1 Separable Problems

Intuitively, it is clear that the presence of the lower-level variables in the upper-level objective function is the essential element that makes the analysis and design of algorithms for MIBLP difficult. This is formalized in the following discussion, where we see that, if these variables are not present, we have a much closer relationship to traditional MILP. We note that under the assumptions made in Chapter 1, we need only consider upper-level feasibility, but the following results hold without this assumption.

Let $d^1 = 0$ in (MIBLP), a special case hereafter called MIBLP($d^1 = 0$), and denote the optimal value of (MIBLP) by $z_{MIBLP}$. We have the following result.

**Theorem 3.10** *Let*

$$z_{MILP} = \min_{(x,y)\in\Omega^I} c^1 x. \tag{3.2}$$

*If $d^1 = 0$, then we have $z_{MIBLP} = z_{MILP}$.*

**Proof.** Let $(x^*, y^*)$ be an optimal solution to (3.2) with value $c^1 x^*$. $(x^*, y^*)$ optimal implies $(x^*, y^*) \in \Omega^I$. Thus, there exists some $x \in (\mathcal{P}_U \cap X)$, namely $x^*$, and some $y \in M^I(x^*)$, namely $y^*$. So, $\mathcal{F}^I \neq \emptyset$. Since, $c^1 x$ is the same for any choice of $y$, we just need to show that $x^*$ is optimal for the upper-level DM in (MIBLP). Suppose it's not. That means there exists some $\hat{x}$ such that $\hat{x} \in (\mathcal{P}_U \cap X)$ with $c^1\hat{x} < c^1 x^*$. But, this contradicts the optimality of $x^*$ for (3.2). On the other hand, consider some solution $(x', y')$ that is optimal for (MIBLP). $(x', y')$ optimal implies $(x', y') \in \mathcal{F}^I$, which implies $(x', y') \in \Omega^I$. Thus, $(x', y')$ is feasible for (3.2). Since, again, the objective value $c^1 x$ is the same for any $y$, the existence of $\bar{x}$ such that $c^1\bar{x} < c^1 x'$ contradicts the optimality of $x'$. □

In fact, in this special case, we can solve the MIBLP by simply solving two MILPs. For this reason, we refer to problems of this type as *separable problems*. Let $(\hat{x}, \hat{y})$ be a solution to (3.2), and

consider the lower-level problem of (MIBLP):

$$\min\{d^2 y \mid G^2 y \geq b^2 - A^2 \hat{x}, y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}\}. \tag{3.3}$$

Let $y^*$ be an optimal solution to (3.3). Clearly, $(\hat{x}, y^*)$ is feasible for (MIBLP).[1] And, since the upper-level objective does not depend on $y$, this solution must also be optimal for (MIBLP), since (3.2) is a valid upper bound on (MIBLP).

This sequential solution method for this class of problems is similar to the lexicographic method for solving multiobjective problems (see, e.g., Waltz (1967); Stadler (1988); Rentmeesters et al. (1996); Sun et al. (1998); Korhonen and Siitari (2007)). In the lexicographic method, objective functions are ranked by importance, and optimization is performed according to this ordering. Formally, for the biobjective problem

$$\text{vmin}_{x \in S}[f_1(x), f_2(x)], \tag{3.4}$$

using the lexicographic method means first solving the problem

$$\min_{x \in S} f_1(x), \tag{3.5}$$

to obtain the optimal value $f_1^*$. Then, a new feasible region is defined as:

$$S^1 = \{x \in S \mid f_1(x) = f_1^*\}$$

and the second problem

$$\min_{x \in S^1} f_2(x), \tag{3.6}$$

is solved. This is essentially the method described above for this special class of bilevel programming. However, one important difference exists. In lexicographic optimization, the underlying feasible region $S$ is the same for each objective function, regardless of rank. However, in a bilevel program, the lower-level feasible region does not include the upper-level constraints.

### 3.2.2 MILP Duality and the Value Function

Duality theory can be thought of as the study of methods for generating lower-bounding approximations for value function of mathematical programs. Not surprisingly, evaluating the value function $z$ at even a single point is an NP$-$hard problem in general. Given the difficulty of constructing the value function, we often focus our attention on developing methods to find the best approximation for $z$. In what follows, we use results from duality theory extensively. LP duality is a well-studied

[1] Of course, this relies on the assumption that no lower-level variables appear in the upper-level constraints.

field and has proven invaluable in the development of effective algorithms and sensitivity analysis methods for LP. In theory, the majority of the results from LP duality generalize to the MILP case, but obtaining practical implementations of such methods is quite difficult. Historically, the impact of duality theory has been primarily limited to problems with continuous variables. Recently, however, new advances have been made that demonstrate that similar practical benefits can be obtained from dual information in discrete problems. The interested reader is referred to Guzelsoy and Ralphs (2007) and Guzelsoy (2009) for a full review of duality theory and the more recent advances that lead to tractable MILP duality results.

In the bilevel programming literature, there is evidence of a similar roadblock encountered when discrete variables appear in the lower-level problems. As described above, several solution methods for BLP rely on replacing the requirement $y \in \operatorname{argmin}\{d^2 y \mid y \in \mathcal{S}_L(x)\}$ with the appropriate optimality conditions. Typically, these optimality conditions are derived from LP duality theory and, thus, are not readily applied to MIBLP. However, in this dissertation, we utilize the relationship between LP and MILP duality to bridge the gap between BLP and MIBLP. Following the work in the MILP literature, we demonstrate that many of the same methods can be applied when the lower-level variables are discrete, by applying the appropriate theoretical generalizations. Below, we describe the primary relevant results connecting LP and MILP duality theory to provide a foundation for the analogous connections we draw in subsequent chapters.

**LP Duality.** Recall the linear programming problem of determining:

$$z_{LP} = \min_{x \in \mathcal{S}_{LP}} cx, \tag{LP}$$

where

$$\mathcal{S}_{LP} = \{x \in \mathbb{R}^n \mid Ax \geq b, x \geq 0\}.$$

Changing any member of the triple $(A, b, c)$ yields a perturbation of the LP. In many applications, it is natural to consider changes to the right-hand-side (RHS) vector $b$, because $b$ can be thought of as the resources available to the system being modeled. In a bilevel program, the upper-level DM can indirectly alter the resources available to the lower-level DM through the vector $A^2 x$. That is, a change in the upper-level decision vector results in a perturbation of the lower-level RHS. As we saw in Section 1.6, understanding the effect this has on the solution to the bilevel program can lead to a method for solving bilevel programs.

Consider the parameterized version of (LP):

$$z_{LP}(v) = \min_{x \in \mathcal{S}_{LP}(v)} cx, \tag{LP($v$)}$$

where

$$\mathcal{S}_{LP}(v) = \left\{x \in \mathbb{R}^n_+ \mid Ax \geq v\right\},$$

for all $v \in \mathbb{R}^n$. The LP value function $z_{LP} : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\infty\}$ returns the optimal value of (LP) for each $v \in \mathbb{R}^m$. By convention, we say $z_{MILP} = +\infty$ if $d \notin \Omega_{MILP} = \{v \in \mathbb{R}^m \mid \mathcal{S}_{IP}(v) \neq \emptyset\}$ and $z_{MILP} = -\infty$ if the objective value is unbounded.

By definition, we call a function $F$, such that

$$F(v) \leq z_{LP}(v), \forall v \in \mathbb{R}^m,$$

a *weak dual* function. If (LP$(v)$) has a finite optimal solution and for fixed $b \in \mathbb{R}$, $F(b) = z_{LP}(b)$, we say $F$ is a *strong dual* function. This means that, for the RHS $b$, $F$ yields an exact approximation of $z_{LP}$, and can be used as a substitute for the value function itself. The associated *dual problem* is:

$$\max\{F(b) \mid F(v) \leq z_{LP}(v), v \in \mathbb{R}^m, F : \mathbb{R}^m \to \mathbb{R}\}, \tag{3.7}$$

which returns the "best" dual function with respect to the lower bound at $b$.

It is well-known (Bertsimas and Tsitsiklis, 1997, see, e.g.,) that restricting $F$ to the class of linear functions allows us to rewrite (3.7) as a second LP:

$$\max\{ub \mid uA \leq c, u \in \mathbb{R}^m_+\}. \tag{3.8}$$

Further, if the primal problem is bounded, the optimal solution of (3.8) yields a strong dual function. The reader is referred to Guzelsoy (2009) for a full review on the implications of this result. Utilizing this relationship allows us to write the LP value function as:

$$z_{LP}(v) = \max\{uv \mid uA \leq c, u \in \mathbb{R}^{m_2}_+\}. \tag{3.9}$$

The function $z_{LP}$ is piecewise-linear and convex over $\Omega_{LP} = \{v \in \mathbb{R}^m \mid \mathcal{S}_{LP}(v) \neq \emptyset\}$, where $\mathcal{S}_{LP}(v) = \{y \in \mathbb{R}^{n_2}_+ \mid Ax \geq v\}$. For a fixed right-hand-side $b$, an optimal solution $u^*$ to (3.8) is a *subgradient* of $z_{LP}$ at $b$. In other words,

$$z_{LP}(b) + u^*(v - b) \leq z_{LP}(v), \quad \forall v \in \Omega_{LP}.$$

Further, for a sufficiently small neighborhood of $b$, $u^*$ remains optimal and $z_{LP}(v) = u^*v$. This relationship is illustrated in Figure 3.3. One important by-product of this relationship is our ability to approximate the value of a linear program using a set of dual solutions. Suppose the dual feasible

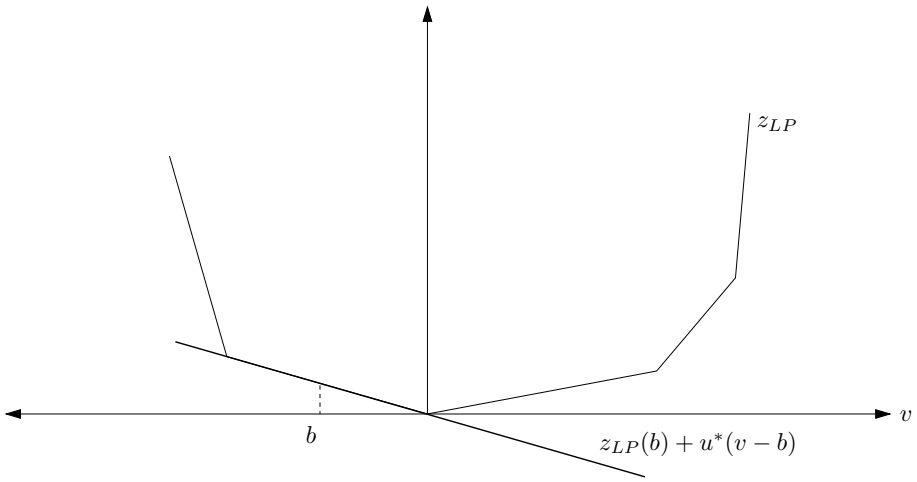



Figure 3.3: A slice of the value function $z_{LP}$ and the subgradient at $b$.

set $\{u \in \mathbb{R}^m \mid uA \leq c\}$ is a polytope and let $\mathcal{R}$ be its set of extreme points. Then, we can write

$$z_{LP}(v) = \max_{\rho \in \mathcal{R}} \{\rho v\}, \quad \forall v \in \mathbb{R}^m, \tag{3.10}$$

We refer to (3.10) as the *extreme point form* of the LP value function. Maximizing over a subset of $\mathcal{R}$ yields an approximation for the value function. This approximation is illustrated in Figure 3.4, where we see a slice of the LP value function and the portions of the approximation derived from different dual solutions.

One application of this technique is the well-known *Benders decomposition algorithm*. In this context, Benders algorithm can be seen as the iterative generation of gradients $\rho \in \mathcal{R}$. We apply a similar idea to MIBLP later in this chapter. The consequences of linear programming duality results have been used extensively for the development of efficient algorithms. Unfortunately, as we will see next, mixed integer programming does not enjoy as many convenient qualities.

**MILP Duality.** The parameterized version of (MILP) is defined as:

$$z_{MILP}(v) = \min_{x \in \mathcal{S}_{MILP}(v)} cx, \tag{MILP$(v)$}$$

where,

$$S_{MILP}(v) = \{x \in \mathbb{Z}^p_+ \times \mathbb{R}^{n-p}_+ \mid Ax \geq v\}$$

for all $v \in \mathbb{R}^m$. As in the LP case, the MILP value function $z_{MILP} : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\infty\}$ returns the optimal value of the program as a function of the RHS vector. As before, we let $z_{MILP} = +\infty$ if $d \notin \Omega_{MILP} = \{v \in \mathbb{R}^m \mid \mathcal{S}_{IP}(v) \neq \emptyset\}$ and $z_{MILP} = -\infty$ if the objective value is unbounded. The

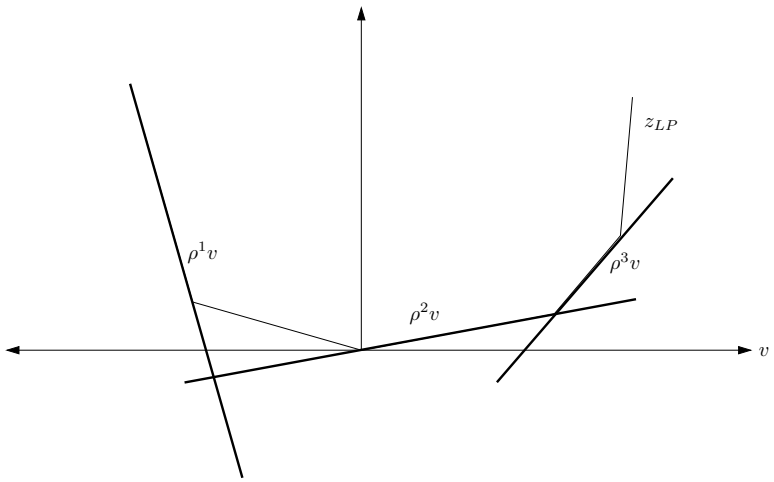



Figure 3.4: An approximation of value function $z_{LP}$.

function $z_{MILP}$ is known to be piecewise polyhedral, but *nonconvex*. Further, Blair and Jeroslow (1977) and Blair (1995) show that $z_{MILP}$ is defined by the value function of a related pure integer program and a linear correction term obtained from the coefficients of the continuous variables. Their construction utilizes a special class of functions, namely *Gomory functions*, a subset of the class of *Chvátal functions*.

**Definition 3.2** *Chvátal functions are the smallest set of functions $\mathcal{C}^m$ such that*

*(i) If $h \in \mathcal{L}^m$, where $\mathcal{L}^m$ is the set of linear functions $f : \mathbb{R}^m \to \mathbb{R}$, then $h \in \mathcal{C}^m$.*

*(ii) If $h_1, h_2 \in \mathcal{C}^m$ and $\alpha, \beta \in \mathbb{Q}_+$, then $\alpha h_1 + \beta h_2 \in \mathcal{C}^m$.*

*(iii) If $h \in \mathcal{C}^m$, then $\lceil h \rceil \in \mathcal{C}^m$.*

*Gomory functions are the smallest set of functions $\mathcal{G}^m \subseteq \mathcal{C}^m$ that satisfy (i)-(iii), and*

*(iv) If $h_1, h_2 \in \mathcal{G}^m$, then $\max\{h_1, h_2\} \in \mathcal{G}^m$.*

Let $\mathcal{E}$ consist of the index sets of dual feasible bases of

$$\min \left\{ \sum_{i=p+1}^{n} c_i x_i \mid \sum_{i=p+1}^{n} a_i x_i \leq v, x_i \geq 0, \forall i \in [p+1, n] \right\}, \tag{3.11}$$

the linear program obtained by dropping the integral variables from (MILP), for a fixed $v \in \Omega_{MILP}$. Since $A$ is rational, we can choose $M \in \mathbb{Z}_+$ such that for any $E \in \mathcal{E}$, $MA_E^{-1}a^j \in \mathbb{Z}^m$ for all

$j = 1, \ldots, p$, where $a^j$ is the $j^{th}$ column of $A$. For $E \in \mathcal{E}$, let $u_E$ be the corresponding basic feasible solution to the dual of

$$\min \left\{ \frac{1}{M} \sum_{i=p+1}^{n} c_i x_i \mid \frac{1}{M} \sum_{i=p+1}^{n} a_i x_i \leq v, x_i \geq 0, \forall i \in [p+1, n] \right\}. \tag{3.12}$$

For $v$ and $E$, let $\lfloor v \rfloor_E = A_E \lfloor A_E^{-1} v \rfloor$. Then, we have the following result.

**Theorem 3.11 (Blair (1995))** *For the MILP* (MILP)*, there is a* $h \in \mathcal{G}^m$ *such that*

$$z_{MILP}(d) = \min_{E \in \mathcal{E}} h(\lfloor v \rfloor_E) + u_E(v - \lfloor v \rfloor_E) \tag{3.13}$$

*for any* $v \in \Omega_{MILP}$*, where* $\mathcal{G}^m$ *is the set of Gomory functions.*

Equation (3.13) is the so-called *Jeroslow Formula*. This result utilizes the value function of a pure integer program (i.e., $p = n$), which can be described by a particular Gomory function (Blair and Jeroslow, 1982), but is still difficult to construct in general.

Using the notion of duality as a bounding method, we can write the MILP dual problem:

$$\max\{F(b) \mid F(v) \leq z_{MILP}(v), v \in \mathbb{R}^m, F : \mathbb{R}^m \to \mathbb{R}\}. \tag{3.14}$$

The value function of the LP relaxation of (MILP) is given by

$$F_{LP}(v) = \max\{uv \mid uA \leq c, v \in \mathbb{R}^m_+\}. \tag{3.15}$$

If we define

$$F(v) = \begin{cases} F_{LP}(v) & \text{for } v \in \Omega_{MILP} \\ 0 & \text{otherwise} \end{cases},$$

where $\Omega_{MILP} = \{v \in \mathbb{R}^m \mid S(v) \neq \emptyset\}$, $F : \mathbb{R}^m \to \mathbb{R}$, and the LP relaxation is bounded, $F$ is feasible for (3.14) (i.e., a weak dual). Such a function provides the best piecewise-linear, convex bounding function for $z_{MILP}$ and is strong for *some* RHS, but is not necessarily strong for a given RHS (see Figure 3.5).

The dual problem (3.14) as stated above is too general to be useful. Motivated by the subadditivity of the MILP value function, Johnson (1973, 1974, 1979), suggested limiting the set of dual functions to one which is more structured. Let $\Gamma^m$ be the set of functions $F : \mathbb{R}^m \to \mathbb{R}$ that are subadditive [2],

[2] A function $F$ is *subadditive* over domain $\Theta$ if $F(\lambda_1) + F(\lambda_2) \geq F(\lambda_1 + \lambda_2)$ for all $\lambda_1, \lambda_2, \lambda_1 + \lambda_2 \in \Theta$.

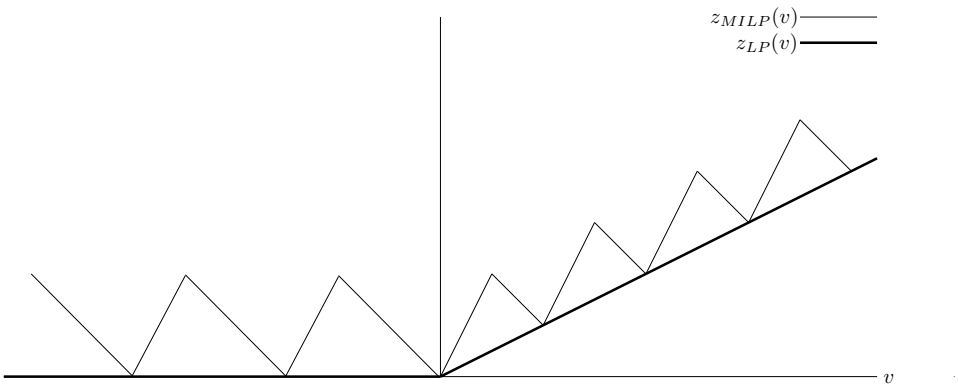



Figure 3.5: The value function of the LP relaxation of a MILP and $z_{MILP}$.

nonincreasing, and for which $F(0) = 0$. The *subadditive dual* of (MILP) is:

$$\begin{aligned} \max \quad & F(b) \\ & F(a^j) \leq c_j, \quad \forall j = 1, \ldots, p \\ & \bar{F}(a^j) \leq c_j, \quad \forall j = p+1, \ldots, n \\ & F \in \Gamma^m \end{aligned} \tag{3.16}$$

where

$$\bar{F}(v) = \limsup_{\delta \to 0^+} \frac{F(\delta v)}{\delta}, \quad \forall v \in \mathbb{R}^m.$$

$\bar{F}$ is the *upper v-directional derivative* of $F$ at zero. As noted in Guzelsoy and Ralphs (2007), $\bar{F}$ is only required in (3.16) if $p < n$ and ensures that solutions to the subadditive dual have gradients that do not exceed those of the value function near zero. The subadditive dual enjoys many of the nice properties of the LP dual problem. We briefly review these properties next.

As with linear programming, a feasible solution to (3.16) can be used to bound the objective value of (MILP).

**Theorem 3.12 (Weak Duality by Jeroslow (1978, 1979))** *If $F$ is feasible to (3.16) and $x$ is feasible to (MILP), then $cx \geq F(b)$.*

The following result shows that (3.16) is a strong dual for (MILP).

**Theorem 3.13 (Jeroslow (1978, 1979); Wolsey (1981))** *If either (MILP) or (3.16) has a finite optimal value, then there exists an optimal primal feasible solution $x^*$ and an optimal dual feasible solution $F^*$ such that $cx^* = F^*(b)$. Further,*

*(i) If (MILP) is infeasible, either (3.16) is infeasible or unbounded from above.*

*(ii) If* (3.16) *is infeasible, either* (MILP) *is infeasible or unbounded from below.*

The subadditive dual (3.16) can also be used to generalize complementary slackness conditions.

**Theorem 3.14 (Complementary Slackness by Jeroslow (1978); Johnson (1979); Wolsey (1981))** *If $x^*$ is feasible to* (MILP) *and $F^*$ is feasible to* (3.16)*, then $x^*$ and $F^*$ are optimal if and only if*

$$\begin{aligned}(F^*(a^j) - c_j)x_j^* = 0, \quad &\textit{for} \quad j = 1, \dots, p \\ (\bar{F}^*(a^j) - c_j)x_j^* = 0, \quad &\textit{for} \quad j = p+1, \dots, n\end{aligned} \tag{3.17}$$

*and*

$$\begin{aligned}\sum_{j=1}^{p} F^*(a^j)x_j^* + \sum_{j=p+1}^{n} \bar{F}^*(a^j)x_j^* = F^*(b) \\ F^*(b - \sum_{j=1}^{p} a^j x_j^*) + \bar{F}^*(b - \sum_{j=p+1}^{n} a^j x_j^*) = 0.\end{aligned} \tag{3.18}$$

The first condition is analogous to the well-known LP complementary slackness conditions. The second condition, sometimes referred to as *complementary linearity*, holds trivially if $F$ and $\bar{F}$ are linear (Llewellyn and Ryan, 1993).

In theory, appropriate optimality conditions for the lower-level problem can be applied directly to the bilevel program, immediately yielding a single-level reformulation. If the resulting formulation can be solved with an existing method, we can solve the bilevel program with a black-box method. However, as we will see next, reformulating the problem in this manner often leads to a problem for which no solution method is known. In this case, we can enforce lower-level optimality indirectly through iterative approximation of the MILP value function. In what follows, we adopt some additional notation, to simplify the exposition. Let $\text{MIBLP}_{\mho_L}^{\mho_U}$ a special case of the canonical problem defined by conditions $\mho_U$ and $\mho_L$ on the upper- and lower-level variables, respectively. For example, the MIBLP in which all upper-level variables are binary and all lower-level variables continuous would be written $\text{MIBLP}_{\mathbb{R}^{n_2}}^{\mathbb{B}^{n_1}}$, and our canonical problem would be $\text{MIBLP}_{Y}^{X}$. If either, or both, of the conditions are left blank, it should be assumed that the restrictions on the variables are as stated in (MIBLP).

### 3.2.3 Reformulations

One well-known approach to single-level reformulation found in the BLP literature relies on replacing the optimality constraint on the lower-level variables with appropriate KKT conditions (see e.g.

Fortuny-Amat and McCarl, 1981; Bard and Moore, 1990). If the lower-level problem is a LP (as in BLP), this means replacing the optimality constraint with primal feasibility, dual feasibility, and complementary slackness conditions. In theory, we can apply the same method to (MIBLP) using MILP duality theory. Below, we demonstrate how to apply an analogous technique for MIBLP, using the subadditive dual described in the previous section. After introducing the general reformulation, we provide several special cases for which the the method yields more useful reformulations.

**General MIBLP.** As shown above, the subadditive dual has many of the same properties of the LP dual. For fixed $\hat{x}$, the lower-level MILP is:

$$\min\{d^2 y \mid G^2 y \geq b^2 - A^2\hat{x}, y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2}\}. \tag{3.19}$$

The associated subadditive dual is then:

$$\begin{aligned} \max \quad & F(b^2 - A^2\hat{x}) \\ & F((g^2)^j) \leq d_j^2, \quad \forall j = 1, \ldots, p_2 \\ & \bar{F}((g^2)^j) \leq d_j^2, \quad \forall j = p_2 + 1, \ldots, n_2 \\ & F \in \Gamma^m \end{aligned} \tag{3.20}$$

where $g_j^2$ and $\bar{F}$ are defined as in Section 3.2.2. Applying the duality results given in Section 3.2.2 yields the following result.

**Proposition 3.15** *If $y^*$ is an optimal feasible solution for (3.19) and $F^*$ is an optimal dual feasible solution for (3.20), then $y^*$ and $F^*$ must satisfy*

$$G^2 y^* \geq b^2 - A^2\hat{x} \tag{3.21a}$$

$$F^*(g_j^2) \leq d_j^2, \quad \forall j = 1, \ldots, p_2 \tag{3.21b}$$

$$\bar{F}^*(g_j^2) \leq d_j^2, \quad \forall j = p_2 + 1, \ldots, n_2 \tag{3.21c}$$

$$(F^*(g_j^2) - d_j^2) y_j^* = 0, \quad \forall j = 1, \ldots, p2 \tag{3.21d}$$

$$(\bar{F}^*(g_j^2) - d_j^2) y_j^* = 0, \quad \forall j = p_2 + 1, \ldots, n_2 \tag{3.21e}$$

$$\sum_{j=1}^{p_2} F^*(g_j^2) y_j^* + \sum_{j=p_2+1}^{n_2} \bar{F}^*(g_j^2) y_j^* = F^*(b^2 - A^2\hat{x}) \tag{3.21f}$$

$$F^*(b^2 - \sum_{j=1}^{p_2} (g^2)^j) y_j^*) + \bar{F}^*(b^2 - \sum_{j=p_2+1}^{n_2} (g^2)^j) y_j^*) = 0 \tag{3.21g}$$

*for any $x = \hat{x}$.*

**Proof.** If $y^*$ is feasible for (3.19) and $F^*$ is feasible for (3.20), then (3.21a)-(3.21c) must be satisfied. Since $y^*$ and $F^*$ are optimal solutions for (3.19) and (3.20), respectively, we can apply Theorem 3.14, which yields (3.21d)-(3.21g). □

Proposition 3.15 implies that we can replace the lower-level problem with optimality conditions (3.21), give control of all variables to the upper-level DM, and introduce the following equivalent single-level reformulation of (MIBLP):

$$
\begin{aligned}
\max \quad & c^1x + d^1y \\
\text{subject to} \quad & A^1x \leq b^1 \\
& -A^2x - G^2y \leq -b^2 \\
& F(g_j^2) \leq d_j^2, \quad \forall j = 1, \dots, p_2 \\
& \bar{F}(g_j^2) \leq d_j^2, \quad \forall j = p_2+1, \dots, n_2 \\
& (F(g_j^2) - d_j^2)y_j = 0, \quad \forall j = 1, \dots, p_2 \\
& (\bar{F}(g_j^2) - d_j^2)y_j = 0, \quad \forall j = p_2+1, \dots, n_2 \\
& \sum_{j=1}^{p_2} F(g_j^2)y_j + \sum_{j=p_2+1}^{n_2} \bar{F}(g_j^2)y_j = F(b^2 - A^2\hat{x}) \\
& F(b - \sum_{j=1}^{p_2}(g^2)^j y_j) + \bar{F}(b - \sum_{j=p_2+1}^{n_2}(g^2)^j y_j) = 0 \\
& x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_2-p_2}, y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2}, F \in \Gamma^{m_2}.
\end{aligned}
\tag{MIBLP-1}
$$

Removing the complementarity terms

$$
\begin{aligned}
&(F(g_j^2) - d_j^2)y_j = 0, \forall j = 1, \dots, p_2 \\
&(\bar{F}(g_j^2) - d_j^2)y_j = 0, \forall j = p_2+1, \dots, n_2
\end{aligned}
$$

and

$$
\begin{aligned}
&\sum_{j=1}^{p_2} F(g_j^2)y_j^* + \sum_{j=p_2+1}^{n_2} \bar{F}(g_j^2)y_j^* = F(b^2 - A^2\hat{x}) \\
&F(b - \sum_{j=1}^{p_2}(g^2)^j y_j) + \bar{F}(b - \sum_{j=p_2+1}^{n_2}(g^2)^j y_j) = 0
\end{aligned}
$$

from (MIBLP-1) yields the *subadditive relaxation problem*

$$\begin{aligned} \min \quad & c^1x + d^1y \\ \text{subject to} \quad & A^1x \geq b^1 \\ & A^2x + G^2y^* \geq b^2 \\ & F(g_j^2) \leq d_j^2, \quad \forall j = 1, \dots, p_2 \\ & \bar{F}(g_j^2) \leq d_j^2, \quad \forall j = p_2 + 1, \dots, n_2 \\ & x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_2 - p_2}, y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}, F \in \Gamma^{m_2}. \end{aligned} \tag{SRP}$$

**Proposition 3.16** (SRP) *provides a valid lower bound on* (MIBLP-1).

**Proof.** Let $(x^*, y^*, F^*)$ be an optimal solution to (MIBLP-1). Suppose, for sake of contradiction,

$$c^1x^* + d^1y^* < z^*_{SRP}$$

where $z^*_{SRP}$ is the optimal objective value of (SRP). However, this immediately leads to a contradiction since the fact that $(x^*, y^*, F^*)$ is feasible for (MIBLP-1) implies that $(x^*, y^*, F^*)$ is feasible for (SRP). □

The difficulty inherent in the employment of this formulation is that both (MIBLP-1) and (SRP) involve solving an optimization problem for which one of the "variables" is a subadditive function. There are no direct methods for solving such optimization problems. If, however, we were able to solve a problem of the form (SRP) we could immediately generalize the complementarity branch-and-bound algorithm given in Bard and Moore (1990). If all variables in the lower-level problem are required to be integer, we can use the linear representation of the subadditive dual to transform (MIBLP-1) into something more amenable to traditional optimization solvers.

**Pure Integer Lower-level Problems.** Suppose, for all $x \in (\mathcal{P}_U \cap X)$, the lower-level problem is a bounded pure integer program (i.e. MIBLP$_{\mathbb{Z}^{n_1}}$) and $b^2 - A^2x \in \mathbb{Q}_+^{m_2}$. Then, for fixed $x = \hat{x}$, (3.20) reduces to

$$\begin{aligned} \max \quad & F(b^2 - A^2\hat{x}) \\ & F((g^2)^j) \leq d_j^2, \quad \forall j = 1, \dots, n_2 \\ & F \in \Gamma^m. \end{aligned} \tag{3.22}$$

Further, suppose we scale $G^2$ and $b^2 - A^2\hat{x}$ to be integer. Then, we can reformulate (3.22) as:

$$\begin{aligned} \max \quad & \eta(\hat{b}) \\ & \eta(\lambda) + \eta(\mu) \geq \eta(\lambda + \mu), \forall 0 \leq \lambda \leq \hat{b}, 0 \leq \mu \leq \hat{b}, 0 \leq \lambda + \mu \leq \hat{b} \\ & \eta((g^2)^j) \leq d_j^2, \quad \forall j = 1, \ldots, n_2 \\ & \eta(0) = 0. \end{aligned} \tag{3.23}$$

where $\hat{b} = b^2 - A^2\hat{x}$ and $\eta : \{\alpha \mid \alpha \leq \hat{b}\} \to \mathbb{R}$. This follows from the fact that, if the primal problem is a bounded pure integer program, we can substitute the subadditive function with the values it takes over the finite domain $\{\lambda \in \mathbb{Z}_+^{m_2} \mid \lambda \leq \hat{b}\}$ and a set of constraints which ensure that $\eta$ is subadditive (Gomory, 1969; Johnson, 1979).

This immediately leads to a mixed integer nonlinear programming (MINLP) reformulation of MIBLP$_{\mathbb{Z}^{n_2}}$:

$$\begin{aligned} \min \quad & c^1 x + d^1 y \\ \text{subject to} \quad & A^1 x \geq b^1 \\ & A^2 x + G^2 y \geq b^2 \\ & \eta(\lambda) + \eta(\mu) \geq \eta(\lambda + \mu), \\ & \qquad \forall 0 \leq \lambda \leq b^2 - A^2 x, 0 \leq \mu \leq b^2 - A^2 x, 0 \leq \lambda + \mu \leq b^2 - A^2 x \\ & \eta(g_j^2) \leq d_j^2, \quad \forall j = 1, \ldots, n_2 \\ & (\eta(g_j^2) - d_j^2) y_j = 0, \quad \forall j = 1, \ldots, n_2 \\ & \eta(0) = 0 \\ & x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}_+^{n_2 - p_2}, y \in \mathbb{Z}_+^{n_2}. \end{aligned} \tag{MIBLP$_{\mathbb{Z}^{n_2}}$-2}$$

In the MINLP (MIBLP$_{\mathbb{Z}^{n_2}}$-2), the variables $\eta$ represent the actual values of the subadditive function over its domain. The constraints

$$\begin{aligned} \eta(\lambda) + \eta(\mu) \geq \eta(\lambda + \mu), \forall 0 \leq \lambda \leq b^2 - A^2 x, 0 \leq \mu \leq b^2 - A^2 x, 0 \leq \lambda + \mu \leq b^2 - A^2 x \\ \eta(g_j^2) \leq d_j^2, \quad \forall j = 1, \ldots, n_2 \\ \eta(0) = 0 \end{aligned}$$

enforce the subadditive requirement on $\eta : \{\alpha \mid \alpha \leq \hat{b}\} \to \mathbb{R}$. It can be shown that the row dimension of this MINLP can be reduced using a discrete analog of Farkas' Lemma (Lasserre, 2004a,b, 2009). Applying this method may yield an MINLP reformulation of MIBLP$_{\mathbb{Z}^{n_2}}$ that can be solved via direct methods. Exploring the computational properties of this problem is an area of interesting future research.

Not surprisingly, another case for which this reformulation method is simplified is that of a continuous lower-level problem. In fact, in this case, the method is greatly simplified since we can return to familiar LP dual for the reformulation.

**Continuous Lower-level Problems.** The underlying approach used in the BLP depends only on the structure of $Y$. Thus, we can easily apply the same approach to MIBLP, if the lower-level problem is continuous (i.e., $Y = \mathbb{R}^{n_2}$). This yields the single-level MINLP reformulation of MIBLP$_{\mathbb{R}^{n_2}}$:

$$\begin{aligned} \min \quad & c^1 x + d^1 y \\ \text{subject to} \quad & A^1 x \geq b^1 \\ & A^2 x + G^2 y \geq b^2 \\ & uG^2 \leq d^2 \\ & u(b^2 - G^2 y - A^2 x) = 0 \\ & (d^2 - uG^2) y = 0 \\ & x \in X, y \in \mathbb{R}^{n_2}_+, u \in \mathbb{R}^{m_2}_+. \end{aligned} \qquad \text{(MIBLP}_{\mathbb{R}^{n_2}}\text{-1)}$$

Of course, if the upper-level variables are also continuous, MIBLP$_{\mathbb{R}^{n_2}}$ is equivalent to (BLP), and the reformulation (MIBLP$_{\mathbb{R}^{n_2}}$-1) reduces to the *linear program with equilibrium constraints* (LPEC) reformulation of (BLP) (see e.g. Judice and Faustino, 1992). A variety of solution methods have been suggested for LPECs, including branch and bound (Fortuny-Amat and McCarl, 1981; Bard and Falk, 1982; Bard and Moore, 1990), and interior point methods (Luo et al., 1996). The reader is referred to Vicente and Calamai (1994) and Luo et al. (1996) for a comprehensive review of such solution methods. In fact, this reformulation method is not limited to MIBLP$_{\mathbb{R}^{n_2}}$, but only requires $P_L(x)$ to be a convex polyhedral set, a property shared by a variety of MPECs (Luo et al., 1996).

Audet et al. (1997) show that a BLP can be reformulated as standard MILP, in which all integer variables are binary. Their reformulation utilizes the LPEC reformulation of BLP as an intermediate step, and a common modeling trick to replace the nonlinear complementarity conditions. As before, this reformulation technique does not depend on the structure of $X$, but only requires continuous variables in the lower-level. Thus, we use apply the same general method for the mixed integer case, substituting (MIBLP$_{\mathbb{R}^{n_2}}$-1) in the intermediate step.

Let $e_n$ be an $n-$dimensional column vector of ones and suppose the optimal value of (MIBLP$_{\mathbb{R}^{n_2}}$-1)

is finite. Applying the methodology of Audet et al. (1997) to (MIBLP$_{\mathbb{R}^{n_2}}$-1) yields the MILP:

$$\begin{aligned}
\min \quad & c^1x + d^1y \\
\text{subject to} \quad & A^1x \geq b^1 \\
& A^2x + G^2y \geq b^2 \\
& -uG^2 \geq -d^2 \\
& -A^2x - G^2y - L\lambda \geq -Le_{n_2} - b^2 \\
& -u + L\lambda \geq 0 \\
& uG^2 - L\mu \geq d^2 \\
& -y - L\mu \geq -Le_{m_2} \\
& x \in X, y \in \mathbb{R}^{n_2}_+, u \in \mathbb{R}^{m_2}_+, \lambda \in \mathbb{B}^{m_2}, \mu \in \mathbb{B}^{n_1}.
\end{aligned} \tag{3.24}$$

for some large finite constant $L > 0$. It is easy to see that when $\lambda_i = 1$, we have:

$$\begin{aligned}
a_i^2x + g_i^2y &= b_i^2 \\
u_i &\leq L
\end{aligned}$$

for $i = 1 \ldots, m_2$. Alternatively, when $\lambda_i = 0$:

$$\begin{aligned}
a_i^2x + g_i^2y &\leq b_i^2 + L \\
u_i &= 0.
\end{aligned}$$

Since $\lambda \in \mathbb{B}^{m_2}$, the combination of these constraints enforces the complementarity condition

$$u(b^2 - G^2y - A^2x) = 0.$$

A similar argument shows how the condition $(d^2 - uG^2)y = 0$ is enforced. This result is stated formally in the following proposition.

**Proposition 3.17** *Suppose the optimal value of* (MIBLP$_{\mathbb{R}^{n_2}}$-1) *is bounded, and let* $(x^*, y^*)$ *be a finite optimal solution. There exists a large finite constant* $L > 0$ *and* $u \in \mathbb{R}^{m_2}_+, \lambda \in \mathbb{B}^{m_2}, \mu \in \mathbb{B}^{n_1}$, *such that* $(x^*, y^*, u^*, \lambda^*, \mu^*)$ *is an optimal solution of* (3.24). *On the other hand, for such an* $L$, *if* $(x^*, y^*, u^*, \lambda^*, \mu^*)$ *is an optimal solution of* (3.24), *then* $(x^*, y^*)$ *is an optimal solution of* (MIBLP$_{\mathbb{R}^{n_2}}$-1).

If $X = \mathbb{R}^{n_1}$, (3.24) reduces to the MILP formulation in Audet et al. (1997) and provides a single-level reformulation (BLP).

Audet et al. (1997) also describe how to reformulate MIBLPs as BLPs, when all upper-level integer variables are binary and all lower-lever variables are continuous. In the following example, we demonstrate how to extend their method to yield a MILP reformulation of MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$.

**Binary upper-level and Continuous Lower-level.** Let $X = (\mathbb{B}^{p_1} \times \mathbb{R}^{n_1-p_1})$ and $Y = \mathbb{R}^{n_2}$. Applying the methods of Audet et al. (1997) yields a BLP reformulation of MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$:

$$\begin{aligned}
\min \quad & c^1x + d^1y \\
\text{subject to} \quad & A^1x \geq b^2 \\
& 0 \leq x_i, && \forall i \in [p_1+1, n_1] \\
& 0 \leq x_i \leq 1, && \forall i \in [1, p_1] \\
& \gamma = 0 \\
& y \in \operatorname{argmin}\big\{ d^2y + e^T_{n_1-p_1}\gamma : A^2x + G^2y \geq b^2 \\
& && -\gamma_i \geq -x_i, \quad \forall i \in [1, p_1] \\
& && -\gamma_i \geq -(1-x_i), \quad \forall i \in [1, p_1] \\
& && y \geq 0 \,\big\}
\end{aligned} \tag{3.25}$$

We can use (3.25) to reformulate MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$ as a MILP.

Let $x_I = \{x_i \mid i \in [1, p_1]\}$. Consider the lower-level problem of (3.25), for fixed $\hat{x}$:

$$\begin{aligned}
\min \quad & d^2y + e^T_{p_1}\gamma \\
\text{subject to} \quad & G^2y \geq b^2 - A^2\hat{x} \\
& -\gamma \geq -\hat{x}_I \\
& -\gamma \geq -(1-\hat{x}_I) \\
& y \geq 0.
\end{aligned} \tag{3.26}$$

The dual of (3.26) is given by:

$$\begin{aligned}
\max \quad & u(b^2 - A^2\hat{x}) - v^1\hat{x}_I - v^2(1-\hat{x}_I) \\
\text{subject to} \quad & uG^2 \leq d^2 \\
& -v^1 = e \\
& -v^2 = e \\
& u, v^1, v^2 \geq 0.
\end{aligned} \tag{3.27}$$

Applying the reformulation method of (MIBLP$_{\mathbb{R}^{n_2}}$-1), where we utilize the lower-level optimality

conditions, to (3.25) yields:

$$
\begin{aligned}
\min \quad & c^1 x + d^1 y \\
\text{subject to} \quad & A^1 x \geq b^2 \\
& 0 \leq x_i, \quad \forall i \in [p_1 + 1, n_1] \\
& 0 \leq x_i \leq 1, \quad \forall i \in [1, p_1] \\
& \gamma = 0 \\
& A^2 x + G^2 y \geq b^2 \\
& \gamma_i \leq x_i, \quad \forall i \in [1, p_1] \\
& \gamma_i \leq (1 - x_i), \quad \forall i \in [1, p_1] \\
& u G^2 \leq d^2 \\
& -v^1 = e \\
& -v^2 = e \\
& u(b^2 - G^2 y - A^2 x) = 0 \\
& v_i^1(\gamma_i - x_i) = 0, \quad \forall i \in [1, p_1] \\
& v_i^2(\gamma_i - 1 + x_i) = 0, \quad \forall i \in [1, p_1] \\
& (d^2 - u G^2) y = 0 \\
& y, u, v^1, v^2 \geq 0.
\end{aligned}
\tag{3.28}
$$

Then, applying the reformulation method of (3.24) to the complementarity problem (3.28) yields

the MILP reformulation of MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$:

$$
\begin{aligned}
\min \quad & c^1x + d^1y \\
\text{subject to} \quad & A^1x \geq b^2 \\
& x_i \leq 1, \quad \forall i \in [1, p_1] \\
& \gamma = 0 \\
& A^2x + G^2y \geq b^2 \\
& \gamma_i \leq x_i, \quad \forall i \in [1, p_1] \\
& \gamma_i \leq (1 - x_i), \quad \forall i \in [1, p_1] \\
& uG^2 \leq d^2 \\
& -v^1 = e \\
& -v^2 = e \qquad \text{(MIBLP}^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}\text{-1)} \\
& A^2x + G^2y + L\lambda^1 \leq Le_{m_2} + b_2 \\
& u - L\lambda^1 \leq 0 \\
& -\gamma_i + L\lambda^2_i \leq Le_{p_1} - x_i, \quad \forall i \in [1, p_1] \\
& v^1_i - L\lambda^2_i \leq 0 \quad \forall i \in [1, p_1] \\
& -\gamma_i + L\lambda^3_i \leq Le_{p_1} - (1 - x_i), \quad \forall i \in [1, p_1] \\
& v^2_i - L\lambda^3_i \leq 0 \quad \forall i \in [1, p_1] \\
& -uG^2 + L\mu \leq -d^2 \\
& y + L\mu \leq Le_{n_2} \\
& x \in \mathbb{R}^{n_1}_+, y \in \mathbb{R}^{n_2}_+, u \in \mathbb{R}^{m_2}, v^1, v^2 \in \mathbb{R}^{p_1}_+ \\
& \lambda^1 \in \mathbb{B}^{m_2}, \lambda^2, \lambda^3 \in \mathbb{B}^{p_1}, \mu \in \mathbb{B}^{n_2},
\end{aligned}
$$

for some large finite constant $L > 0$.

**Proposition 3.18 (Audet et al. (1997))** *Suppose the optimal value of* (3.25) *is bounded, and let* $(x^*, y^*, \gamma^*)$ *be a finite optimal solution. There exists a large finite constant* $L > 0$ *and* $u \in \mathbb{R}^{m_2}, v^1, v^2 \in \mathbb{R}^{p_1}_+, \lambda^1 \in \mathbb{B}^{m_2}, \lambda^2, \lambda^3 \in \mathbb{B}^{p_1}, \mu \in \mathbb{B}^{n_2}$*, such that*

$$(x^*, y^*, \gamma^*, u^*, v^{1^*}, v^{2^*}, \lambda^{1^*}, \lambda^{2^*}, \lambda^{3^*}, \mu^*)$$

*is an optimal solution of* (MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$-1)*. On the other hand, for such an* $L$*, if*

$$(x^*, y^*, \gamma^*, u^*, v^{1^*}, v^{2^*}, \lambda^{1^*}, \lambda^{2^*}, \lambda^{3^*}, \mu^*)$$

*is an optimal solution of* (MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$-1)*, then* $(x^*, y^*, \gamma^*)$ *is an optimal solution of* (3.25).

Combining the previous arguments yields the following result.

**Theorem 3.19** *Let* $X = \mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}$ *and* $Y = \mathbb{R}^{n_2}$*. If* $(x^*, y^*)$ *is an optimal solution of* (MIBLP)*, then there exists a large finite constant* $L > 0$ *and* $u \in \mathbb{R}^{m_2}, v^1, v^2 \in \mathbb{R}^{p_1}_+, \lambda^1 \in \mathbb{B}^{m_2}, \lambda^2, \lambda^3 \in \mathbb{B}^{p_1}, \mu \in \mathbb{B}^{n_2}$*, such that*

$$(x^*, y^*, \gamma^*, u^*, v^{1^*}, v^{2^*}, \lambda^{1^*}, \lambda^{2^*}, \lambda^{3^*}, \mu^*)$$

*is an optimal solution of* (MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$-1)*. On the other hand, for such an* $L$*, if*

$$(x^*, y^*, \gamma^*, u^*, v^{1^*}, v^{2^*}, \lambda^{1^*}, \lambda^{2^*}, \lambda^{3^*}, \mu^*)$$

*is an optimal solution of* (MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$-1)*, then* $(\hat{x}^*, \hat{y}^*)$ *is an optimal solution of* (MIBLP)*.*

Audet et al. (2007) provide an alternative reformulation of the LPEC reformulation of BLP that is convenient for disjunctive cut generation. This method is also applicable to MIBLP$_{\mathbb{R}^{n_2}}$.

**Disjunctive Reformulation for Continuous Lower-level Problems.** Let

$$CC_k(x, y, u) := \begin{cases} u_k(b^2 - G^2y - A^2x)_k = 0 & 1 \le k \le m_2, \\ y_{k-m_2}(d^2 - uG^2) = 0 & 1 \le k - m_2 \le n_2. \end{cases}$$

Then, substitution yields the reformulation of (MIBLP$_{\mathbb{R}^{n_2}}$-1):

$$\begin{aligned} \min_x \quad & c^1x + d^1y \\ \text{subject to} \quad & A^1x \ge b^1 \\ & A^2x + G^2y \ge b^2 \\ & uG^2 \le d^2 \\ & CC_k(x, y, u) = 0, \quad k = 1, 2, \ldots, m_2 + n_2 \\ & x \in X, u \in \mathbb{R}^{m_2}_+, y \in \mathbb{R}^{n_2}_+. \end{aligned} \qquad \text{(MIBLP}_{\mathbb{R}^{n_2}}\text{-2)}$$

The reformulations (MIBLP$^{\mathbb{B}^{p_1}\times\mathbb{R}^{n_1-p_1}}_{\mathbb{R}^{n_2}}$-1) and (MIBLP$_{\mathbb{R}^{n_2}}$-2) are straightforward applications of methods borrowed from the BLP literature. Each is useful in its ability to solve bilevel programs via direct methods, but both are limited by their reliance on a continuous lower-level problem. On the other hand, the reformulation (MIBLP-1) can be applied to the general case, but may be limited by computational difficulties in all but simple cases. Previously, we have alluded to the potential

utility of the lower-level value function for MIBLP algorithm design. In the next section, we describe methods for approximating the value function, allowing us to enforce optimality conditions indirectly, and yielding the foundations of a a solution framework.

### 3.2.4 Exact Solution Methods

In the previous section, we used the optimality conditions on the lower-level problems to yield single-level reformulations. Alternatively, if we know the full lower-level value function, we can use it to reformulate the problem as

$$\begin{aligned} \min \quad & c^1 x + d^1 y \\ \text{subject to} \quad & A^1 x \geq b^1 \\ & A^2 x + G^2 y \geq b^2 \\ & d^2 y = z_{MILP}(b^2 - A^2 x) \\ & x \in \mathbb{Z}_+^{p_1} \times \mathbb{R}^{n_1 - p_1}, y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}, \end{aligned} \tag{3.29}$$

where

$$z_{MILP}(b^2 - A^2 x) = \min\{d^2 y \mid G^2 \geq b^2 - A^2 x, y \geq 0, y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}\}. \tag{3.30}$$

Unfortunately, as described previously, determining the complete structure of the value function is very difficult in general. However, one may be able to discover enough of the structure to compute a function that approximates the value function. In this section, we discuss methods based on approximations of the lower-level value function. The underlying idea of these methods is that, if we are able to find strong approximations, we can effectively represent the value function using a series of bounding functions. These methods ensure lower-level optimality for a subset of upper-level solutions and lead to algorithms that enforce a strong bound only when necessary. Algorithmically, we begin with simple approximations, then iteratively improve them by generating new functions for additional values of the right-hand side $b^2 - A^2 x$, for $x \in \mathcal{P}_U \cap X$. These methods can be seen as a way of enforcing optimality conditions indirectly.

In what follows, we first we describe an upper-bounding method that can be applied to the general MIBLP. Then, we demonstrate how the upper-bounding method reduces when the lower-level problem has a particular special structure. Finally, we examine recourse problems, another special case of MIBLP, and derive a lower-bounding method that can be used to solve problems of this type. Each of the algorithms presented are theoretical in nature and would require the development of additional methodology to be transformed into practical methods.

**General MIBLP.** One way to obtain upper bounds on the lower-level objective value is to consider *restrictions* of the problem. An obvious restriction of the lower-level problem results from fixing the integer variables, yielding an LP with the value function:

$$z_C(\beta) = \min\{d_C^2 y_C \mid G_C^2 y_C \geq \beta, y_C \geq 0\}. \tag{3.31}$$

where $I = \{1, \dots p_2\}$, $C = \{p_2 + 1, \dots, n_2\}$. We assume throughout that the function (3.31) is finite. This assumption can easily be relaxed, but requires a different method of obtaining a restriction.

**Theorem 3.20 (Guzelsoy (2009))** *Let $(\hat{x}, y^{\hat{x}}) \in \mathcal{F}^I$ be a bilevel feasible solution to* (MIBLP)*. More precisely, let $y^{\hat{x}}$ be an optimal solution to the lower-level problem when the upper-level solution is fixed to $\hat{x}$. Define the function*

$$f^{\hat{x}}(v) = d_I^2 y_I^{\hat{x}} + z_C(v - G_I^2 y_I^{\hat{x}}). \tag{3.32}$$

*Then, $f^{\hat{x}}$ satisfies $f^{\hat{x}}(v) \geq z_{IP}(v)$ for all $v \in \mathbb{R}^{m_2}$ with $f^{\hat{x}}(b^2 - A^2\hat{x}) = z_{IP}(b^2 - A^2\hat{x})$. Hence, $f^{\hat{x}}$ is a strong upper-bounding function.*

One way we can employ this bounding function is to obtain disjunctions to be used in a branch-and-cut framework. Consider the upper-bounding function for some $\hat{x} \in (\mathcal{P}_U \cap X)$, and the corresponding optimal lower-level solution $y^{\hat{x}}$:

$$f^{\hat{x}}(v) = d_I^2 y_I^{\hat{x}} + z_C(v - G_I^2 y_I^{\hat{x}}), \tag{3.33}$$

where $z_C$ is defined as above. Recall that $z_C$ can be written

$$z_C(\beta) = \max_{\rho \in \mathcal{R}} \{\rho\beta\}, \tag{3.34}$$

where $\mathcal{R}$ is the set of extreme points of the dual polyhedron

$$\{u \in \mathbb{R}^{m_2} \mid uG_C^2 \leq d_C^2, u \geq 0\}. \tag{3.35}$$

For fixed $\hat{x} \in (\mathcal{P}_U \cap X)$, the upper-bounding function obtained by Theorem 3.20 is

$$f^{\hat{x}}(v) = d_I^2 y_I^{\hat{x}} + \max_{\rho \in \mathcal{R}} \left\{ \left( \rho(v - G_I^2 y_I^{\hat{x}} \right) \right\},$$

where $y^{\hat{x}} \in \operatorname{argmin}\{d^2 y \mid y \in \mathcal{S}_L(\hat{x}) \cap Y\}$ is the lower-level solution obtained during the bilevel feasibility check. For each $\rho \in \mathcal{R}$, we define

$$\Lambda_\rho = \left\{ v \in \mathbb{R}^{m_2} \mid f(v) = d_I^2 y_I^{\hat{x}} + \rho \left( v - G_I^2 y_I^{\hat{x}} \right) \right\}.$$

In other words, $\Lambda_\rho$ is the set of right-hand-sides for which $\rho$ is the optimal dual solution of the continuous relaxation of the lower-level problem with $y_I = y_I^{\hat{x}}$. Then, we have a disjunction of the form:

$$\left\{ \begin{array}{rcl} v & \in & \Lambda_1 \\ d^2 y & \leq & d_I^2 y_I^{\hat{x}} + \rho \left( v - G_I^2 y_I^{\hat{x}} \right) \end{array} \right\} \bigvee \cdots \bigvee \left\{ \begin{array}{rcl} v & \in & \Lambda_R \\ d^2 y & \leq & d_I^2 y_I^{\hat{x}} + \rho \left( v - G_I^2 y_I^{\hat{x}} \right) \end{array} \right\},$$

where $\mathcal{R} = \{\rho^1, \ldots \rho^R\}$. This disjunction can be used in either a branching scheme or a cut generation routine to separate the integer bilevel infeasible point $(\hat{x}, \hat{y}) \in \Omega^I$. In theory, this disjunction can be applied to yield a branch-and-cut algorithm for solving (MIBLP). However, developing practical methods for branching or cut generation over a disjunction of this form remains an open question.

An alternative way to use the upper-bounding functions is similar to the well-known constraint generation method. We demonstrate such a method using the upper-bounding function obtained by applying Theorem 3.20, but note that any appropriate strong upper-approximation will suffice. In particular, alternative methods for restricting the lower-level problem will yield different approximations which can replace or augment the approximation used here.

Suppose we knew the upper-bounding function $f^x$ for some finite subset $J \subseteq (\mathcal{P}_U \cap X)$ of the upper-level decisions. Then, we have a relaxation of (MIBLP):

$$\begin{array}{rl} \min & c^1 x + d^1 y \\ \text{subject to} & A^1 x \geq b^1 \\ & A^2 x + G^2 y \geq b^2 \\ & d^2 y \leq f^{\hat{x}}(b^2 - A^2 x), \quad \forall \hat{x} \in J \\ & x \in X, y \in Y. \end{array} \tag{3.36}$$

This follows from inequality

$$z_{MILP}(b^2 - A^2 x) = \min_{\hat{x} \in (\mathcal{P}_U \cap X)} f^{\hat{x}}(b^2 - A^2 x) \leq \min_{\hat{x} \in J} f^{\hat{x}}(b^2 - A^2 x).$$

It is clear that if the set $J$ is large, the approximation quickly becomes unmanageable. However, we expect only a small subset of the constraints to be binding at optimality. This is similar to

the rationale that supports Benders' Reformulation for MILP, and this intuition leads directly to an algorithm in which we solve relaxations of (3.36), derived by dropping the bounding constraint

$$d^2y \leq f^{\hat{x}}(b^2 - A^2x)$$

for some (or all) $\hat{x} \in J$, and iteratively apply these constraints as they are found to violated.

In order to derive this method, we utilize reformulations of (3.36) that result from the extreme point form of the LP value function.

Before describing our algorithm, however, we note that if $Y = \mathbb{R}^{n_2}$, the bound (3.32) reduces to

$$f^{\hat{x}}(v) = z_C(v), \tag{3.37}$$

which is the value function of the lower-level LP. Thus, we can rewrite the inequality

$$d^2y \leq \min_{\hat{x} \in J} f^{\hat{x}}(b^2 - A^2x).$$

as

$$d^2y \leq \max_{\rho \in \mathcal{R}} \{\rho(b^2 - A^2x)\},$$

since the set $\mathcal{R}$ is the same for all $x \in J$. In what follows, one can make the appropriate simplifications that result from this substitution to yield an algorithm for the special case MIBLP$_{\mathbb{R}^{n_2}}$. We now return to the general case and present a formal description of our algorithm.

Let $J \subseteq (\mathcal{P}_U \cap X)$ be some finite set of feasible upper-level decisions. As described above, for each $J \subseteq (\mathcal{P}_U \cap X)$, we have a relaxation of (MIBLP):

$$\begin{aligned} \min \quad & c^1x + d^1y \\ \text{subject to} \quad & A^1x \geq b^1 \\ & A^2x + G^2y \geq b^2 \\ & d^2y \leq f^{\hat{x}}(b^2 - A^2x) \quad \forall \hat{x} \in J \\ & x \in X, y \in Y. \end{aligned} \tag{3.38}$$

We refer to (3.38) as the *master problem*. Since the approximation is strong, we are guaranteed that the constraint for $\hat{x} \in J$

$$d^2y \leq f^{\hat{x}}(b^2 - A^2x) \tag{3.39}$$

will be tight for some $x \in J$. In particular, this constraint will be binding at the lower-level RHS $(b^2 - A^2\hat{x})$ for which it was obtained. Thus, for any upper-level solution $\hat{x} \in J$, we are guaranteed

to satisfy the original constraint

$$d^2y = z_{MILP}(b^2 - A^2\hat{x}).$$

This, in turn, ensures satisfaction of the condition $y \in M^I(\hat{x})$ and, thus, that bilevel feasibility conditions are met. Because (3.38) is a relaxation of (3.36) and, thus, of (MIBLP), solutions to (3.38) are optimal for (MIBLP), if they are feasible. An iterative approximation algorithm using this relaxation is summarized in Algorithm 3.1.

**Algorithm 3.1** Iterative Upper Approximation

1: *Initialization.* Set $J^1 = \emptyset$ and $t \leftarrow 1$.
2: *Iteration* $t$. Solve (3.38) with $J = J^t$ to obtain $(x^t, y^t)$. Set $x = x^t$ and solve the lower-level problem, for fixed $x$.

(i) If $d^2y^t = z_{MILP}(b^2 - A^2x^t)$, stop. $(x^t, y^t)$ is an optimal solution.

(ii) If $d^2y^t > z_{MILP}(b^2 - A^2x^t)$, apply Theorem 3.20 with $x = x^t$ to obtain upper-bounding function $f^t$. Set $J^{t+1} = J^t \cup \{t\}$ and $t \leftarrow t + 1$.

Algorithm 3.1 outlines a procedure for iteratively improving the value function approximation. However, (3.38) contains a piecewise linear constraint and, thus, cannot be solved by traditional methods. However, it is possible to reformulate this problem, as we see next.

Let $y^{\hat{x}}$ denote the optimal lower-level solution obtained for the RHS $b^2 - A^2\hat{x}$ and

$$\delta_{\hat{\rho}}^{\hat{x}} = \begin{cases} 1 & \hat{\rho} \in \operatorname{argmax}_{\rho \in \mathcal{R}}\{\rho(b^2 - A^2\hat{x})\} \\ 0 & \text{otherwise}, \end{cases}$$

for fixed $\hat{x} \in (\mathcal{P}_U \cap X)$. Note we can model constraint (3.39) with the system:

$$d^2y \leq d_I^2 y_I^{\hat{x}} + z^{\hat{x}} \tag{3.40}$$

$$z^{\hat{x}} \geq \rho(b^2 - A^2x - G_I^2 y_I^{\hat{x}}), \quad \forall \rho \in \mathcal{R} \tag{3.41}$$

$$z^{\hat{x}} \leq M_\rho^{\hat{x}}(1 - \delta_\rho^{\hat{x}}) + \rho(b^2 - A^2x - G_I^2 y_I^{\hat{x}}), \quad \forall \rho \in \mathcal{R} \tag{3.42}$$

$$\sum_{\rho \in \mathcal{R}} \delta_\rho^{\hat{x}} = 1 \tag{3.43}$$

$$\delta_\rho^{\hat{x}} \in \mathbb{B}, \quad \forall \rho \in \mathcal{R}, \tag{3.44}$$

where

$$M_\rho^{\hat{x}} \geq \max_{x \in (\mathcal{P}_U \cap X)} \{\rho(b^2 - A^2x - G_I^2 y_I^{\hat{x}})\}, \quad \rho \in \mathcal{R},$$

or some other suitable upper bound. This follows from the fact that (3.42) enforces

$$z^{\hat{x}} \leq \rho(b^2 - A^2 x - G_I^2 y_I^{\hat{x}}),$$

if $\delta_\rho^{\hat{x}} = 1$, (3.43) ensures that will be the case for some $\rho$, for all $\hat{x} \in J$, and (3.41) forces $z^{\hat{x}}$ to be at least as large as $\max_{\rho \in \mathcal{R}}\{\rho(b^2 - A^2\hat{x} - G_I^2 y_I^{\hat{x}})\}$. Thus, equality will hold for at least one $\rho^{\hat{x}} \in \mathcal{R}$, and must hold for that which achieves the maximum.

Thus, we can rewrite (3.38) as:

$$\begin{aligned}
\min \quad & c^1 x + d^1 y \\
\text{subject to} \quad & A^1 x \geq b^1 \\
& A^2 x + G^2 y \geq b^2 \\
& d^2 y \leq d_I^2 y_I^{\hat{x}} + z^{\hat{x}}, \quad \forall \hat{x} \in J \\
& z^{\hat{x}} \geq \rho(b^2 - A^2 x - G_I^2 y_I^{\hat{x}}), \quad \forall \hat{x} \in J, \rho \in \mathcal{R} \\
& z^{\hat{x}} \leq M_\rho^{\hat{x}}(1 - \delta_\rho^{\hat{x}}) + \rho(b^2 - A^2 x - G_I^2 y_I^{\hat{x}}), \quad \forall \hat{x} \in J, \rho \in \mathcal{R} \\
& \sum_{\rho \in \mathcal{R}} \delta_\rho^{\hat{x}} = 1, \quad \forall \hat{x} \in J \\
& x \in X, y \in Y, \delta_\rho^{\hat{x}} \in \mathbb{B}, \forall \hat{x} \in J, \rho \in \mathcal{R}.
\end{aligned} \tag{3.45}$$

Note that to solve this subproblem, as written, one would need to generate all extreme points of the dual polyhedron, a problem known as the *vertex enumeration problem*. A survey of existing methods and complexity can be found in Avis et al. (1997). One promising algorithm is that of Avis and Fukuda (1992), which has several advantages. Namely, the algorithm requires very little storage space above that required to represent the dual polyhedron, does not produce duplicate vertices, and has a running time that is polynomial in the size of the dual problem. Of course, the vertex enumeration problem is difficult, in general, and its complexity is largely dependent on the nature of the polyhedron (see, e.g., Fukuda et al., 1997; Bussieck and Lübbecke, 1998; Goodman and O'Rourke, 2004). However, substituting any $\mathcal{R}' \subseteq \mathcal{R}$ in (3.45) yields a relaxation of the original problem. Thus, algorithmically, we can initialize with some $\mathcal{R}' \subseteq \mathcal{R}$, and apply a constraint generation algorithm to solve the subproblem.

**Single Constraint in the Lower Level.** In this section, we consider the special case of (MIBLP) in which the lower-level problem contains only a single equality constraint. That is, for fixed $\hat{x}$, the lower-level problem is that of determining

$$\min\{d^2 y \mid g^2 y = b^2 - a^2 x, y \geq 0, y \in Y\}. \tag{3.46}$$

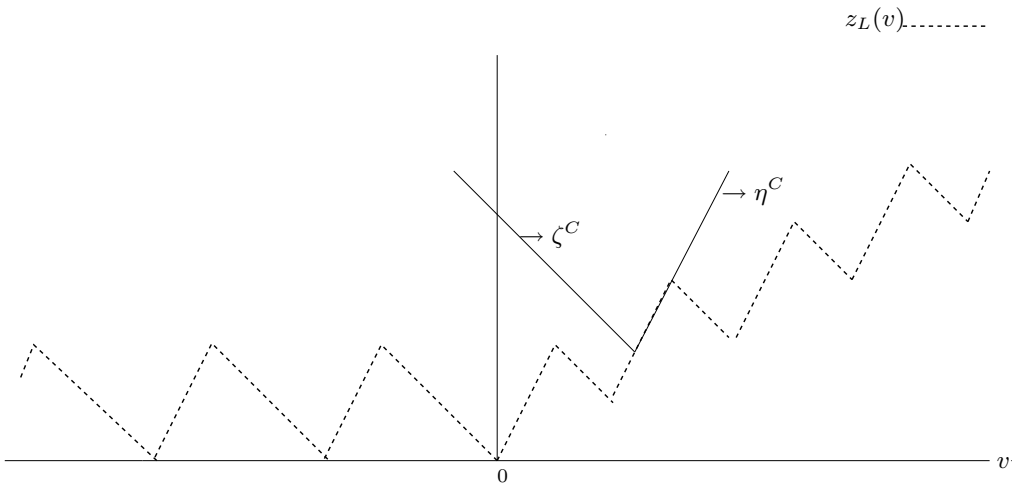



Figure 3.6: An upper-bounding function for $z_{MILP}$.

Let

$$\eta^C = \min\left\{\frac{d_i^2}{g_i^2} \mid g_i^2 > 0, i \in [p_2+1, n_2]\right\} \quad \text{and} \quad \zeta^C = \max\left\{\frac{d_i^2}{g_i^2} \mid g_i^2 < 0, i \in [p_2+1, n_2]\right\}.$$

Then, in this case, we have the following closed form of the upper-bounding function:

$$\begin{aligned} f(b^2 - a^2x) &= \min\left\{d_C^2 y_C \mid g_C^2 y_C = b^2 - a^2x, y_C \geq 0\right\} \\ &= \begin{cases} \eta^C(b^2 - a^2x), & \text{if } b^2 - a^2x \geq 0 \\ \zeta^C(b^2 - a^2x), & \text{if } b^2 - a^2x < 0. \end{cases} \end{aligned}$$

This bound is a special case of Theorem 3.20, and effectively just the maximal subadditive extension of the value function *carried* to the right-hand-side $G_I^2 y_I^*$ (Guzelsoy, 2009). The bounding function is illustrated in Figure 3.6.

In this case, the lower-level value function is defined as:

$$z_{MILP}(b^2 - a^2x) = \min_{y \in \mathcal{S}_L(v)} d^2 y$$

where $\mathcal{S}_L(v) = \{y \in \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2} \mid g^2 y = v\}$. We can apply the results of Guzelsoy and Ralphs (2006) to find the structure of our value function and derive disjunctions valid for MIBLPs.

In general, $z_{MILP}$ is piecewise-linear and can be written as the value function of a pure integer program and an appropriate linear correction term (see Theorem 3.11). Guzelsoy and Ralphs (2006)

show how to apply this property to MILPs with a single equality constraint, to more fully characterize $z_{MILP}$. Let $\eta^C$ and $\zeta^C$ be defined as above and $t^+, t^- \in C$ be such

$$\eta^C = \frac{d_{t^+}^2}{g_{t^+}} \quad \text{if } \eta^C < \infty,$$

and

$$\zeta^C = \frac{d_{t^-}^2}{g_{t^-}} \quad \text{if } \zeta^C > -\infty,$$

respectively. Also, let $T = \{t^+ \mid \eta^C < \infty\} \cup \{t^- \mid \zeta^C > -\infty\}$.

**Proposition 3.21 (Guzelsoy and Ralphs (2006))** *Let*

$$f(v) = \min\{d_I^2 y_I + d_T^2 y_T \mid g_I^2 + g_T^2 y_T = v, y_I \in \mathbb{Z}_+^I, y_T \in \mathbb{R}_+^T\},$$

*where $I = N \setminus C$. Then $f(v) = z_{MILP}(v)$ for all $v \in \mathbb{R}$.*

This result implies sufficiency of the two continuous variables to describe $z_{MILP}$, and is used to simplify the *Jeroslow Formula*.

Let $M \in \mathbb{Z}_+$ be such that for any $t \in T$, $\frac{Mg_j^2}{g_t^2} \in \mathbb{Z}$, for all $j \in I$ (which exists by rationality of $g^2$). Also, let

$$\begin{aligned} h(q) = \min \quad & d_I^2 y_I + \frac{1}{M} d_T^2 y_T + z(\phi) w \\ \text{s.t} \quad & g_I^2 y_I + \frac{1}{M} g_T^2 y_T + \phi w = q \\ & y_I \in \mathbb{Z}_+^I, y_T \in \mathbb{Z}_+^T, w \in \mathbb{Z}_+ \end{aligned}$$

for all $q \in \mathbb{R}$, where $\phi = -\frac{1}{M} \sum_{t \in T} g_t^2$. Finally, for $t \in T$, define

$$\omega(v) = h(\lfloor v \rfloor_t) + \frac{d_t^2}{g_t^2}(v - \lfloor v \rfloor_t)$$

for all $v \in \mathbb{R}$, where $\lfloor v \rfloor_t = \frac{g_t^2}{M} \lfloor \frac{Mv}{g_t^2} \rfloor$. Guzelsoy and Ralphs (2006) apply Theorem 3.11, to obtain

$$z_{MILP}(v) = \min_{t \in T} \omega_t(v), \quad \forall v \in \mathbb{R}, \tag{3.47}$$

which yields the result that $z_{MILP}$ can be described by a finite number of linear segments which coincides with either $\omega_{t^+}$ or $\omega_{t^-}$, and whose slope is either $\eta^C$ or $\zeta^C$.

Figure 3.7 illustrates the structure of the value function for the single-constraint case. With knowledge of this special structure, we can derive bounds on the value of the lower-level objective function as the upper-level solution varies. As stated earlier, for each solution $(\hat{x}, \hat{y})$ to (LR), we may

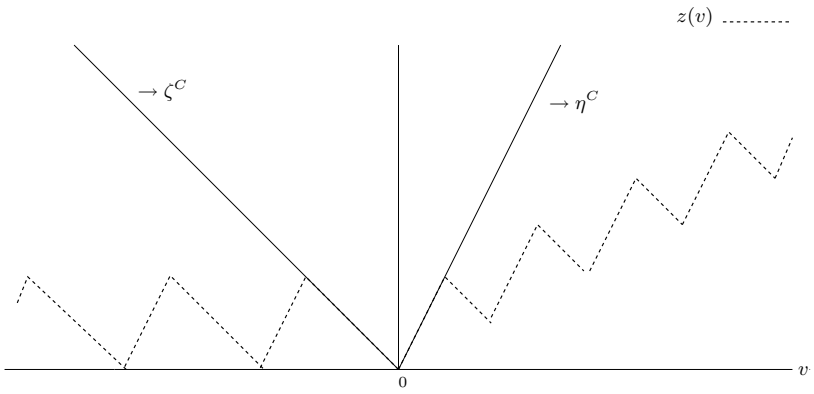



Figure 3.7: The value function of a MILP.

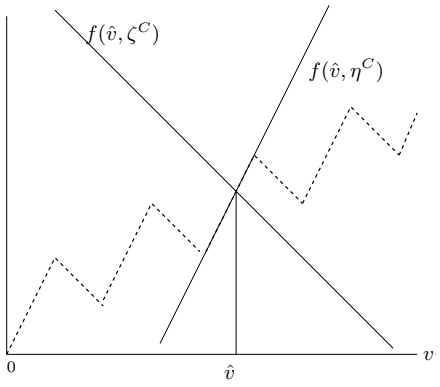



Figure 3.8: Linear bounding functions for the value function.

check for bilevel feasibility by solving the lower-level problem with a fixed upper level solution. Each bilevel feasibility check yields a bilevel feasible pair $(\hat{x}, y^*)$, where $y^* \in \operatorname{argmin}\{d^2 y \mid y \in P_L(\hat{x}) \cap Y\}$. In other words, each bilevel feasibility check yields the value of $z_{MILP}(b^2 - A^2\hat{x}) = d^2 y^*$, where $z_{MILP}$ is the value function of the lower-level problem. Because the value function is piecewise linear with segments whose gradients alternate between two values, we can extend this information to determine the equation of the line on which the bilevel feasible point lies.

Let $\hat{v} = b^2 - a^2\hat{x}$ and consider the affine functions $f(\hat{v}, \eta^C)\}$ and $f(\hat{v}, \zeta^C)$ illustrated in Figure 3.8 with slopes $\eta^C$ and $\zeta^C$, respectively, and each passing through the point $(\hat{v}, z_{MILP}(\hat{v}))$. From the figure, it is easy to see that, for any $v \leq \hat{v}$,

$$z_{MILP}(v) \leq \max\{f(\hat{v}, \eta^C), f(\hat{v}, \zeta^C)\} = f(\hat{v}, \zeta^C).$$

Similarly, for any $v \geq \hat{v}$,

$$z_{MILP}(v) \leq \max\{f(\hat{v}, \eta^C), f(\hat{v}, \zeta^C)\} = f(\hat{v}, \eta^C),$$

where $f(\hat{v}, \zeta^C) = \zeta^C \hat{v} + z_{MILP}(\hat{v})$ and $f(\hat{v}, \eta^C) = \eta^C \hat{v} + z_{MILP}(\hat{v})$. Thus, if $(\hat{x}, \hat{y}) \in (X \times Y)$ is a solution to (LR) such that $\hat{y} \notin M^I(\hat{x})$ (i.e., $(\hat{x}, \hat{y})$ is *not* bilevel feasible), then after substitution, we obtain the valid disjunction

$$\begin{array}{rcl} a^2x & \geq & a^2\hat{x} \\ \zeta^C a^2 x + d^2 y & \leq & \zeta^C a^2 \hat{x} + d^2 y^* \end{array} \quad \text{OR} \quad \begin{array}{rcl} a^2x & \leq & a^2\hat{x} \\ \eta^C a^2 x + d^2 y & \leq & \eta^C a^2 \hat{x} + d^2 y^*, \end{array}$$

which is violated by $(\hat{x}, \hat{y})$, but satisfied by all members of $\mathcal{F}^I$. This disjunction can be used directly as a branching rule to to be applied whenever solutions $(\hat{x}, \hat{y}) \in (X \times Y)$ to (LR) that are not bilevel feasible.

Alternatively, we can use this disjunction to generate a disjunctive cut by considering the two polyhedra, denoted $\mathcal{P}^1$ and $\mathcal{P}^2$, that result if we combine this disjunction with the original set of constraints in $\Omega$:

$$\mathcal{P}^1 = \left\{ \begin{array}{rcl} A^1 x & \geq & b^1 \\ a^2 x + g^2 y & = & b^2 \\ a^2 x & \geq & a^2 \hat{x} \\ -\zeta^C a^2 x - d^2 y & \geq & -\zeta^C a^2 \hat{x} - d^2 y^* \\ x, y & \geq & 0 \end{array} \right\}$$

and

$$\mathcal{P}^2 = \left\{ \begin{array}{rcl} A^1 x & \geq & b^1 \\ a^2 x + g^2 y & = & b^2 \\ -a^2 x & \geq & -a^2 \hat{x} \\ -\eta^C a^2 x - d^2 y & \geq & -\eta^C a^2 \hat{x} - d^2 y^* \\ x, y & \geq & 0. \end{array} \right\}$$

It is well-known that if $(u^i, v^i, w^i, z^i)$ are multipliers for the constraints describing polyhedron $\mathcal{P}^i$, then the following inequalities are valid for $\mathcal{P}^1$ and $\mathcal{P}^2$, respectively:

$$\begin{aligned} u^1 A^1 x + v^1 a^2 x + w^1 a^2 x - z^1 \zeta^C a^2 x + v^1 g^2 y - z^1 d^2 y \geq \\ u^1 b^1 + v^1 b^2 + w^1 a^2 \hat{x} - z^1 (\zeta^C a^2 \hat{x} + d^2 y^*) \\ u^2 A^1 x + v^2 a^2 x - w^2 a^2 x - z^2 \eta^C a^2 x + v^2 g^2 y - z^2 d^2 y \geq \\ u^2 b^1 + v^2 b^2 - w^2 a^2 \hat{x} - z^2 (\eta^C a^2 \hat{x} + d^2 y^*). \end{aligned}$$

Given inequalities $\pi_1^1 x + \pi_2^1 y \geq \pi_0^1$ and $\pi_1^2 x + \pi_2^2 y \geq \pi_0^2$ valid for $\mathcal{P}^1$ and $\mathcal{P}^2$, the disjunctive procedure constructs an inequality $\alpha x + \beta y \geq \gamma$ that is valid for $\text{conv}(\mathcal{P}^1 \cup \mathcal{P}^2)$ by selecting $\alpha$, $\beta$, and $\gamma$ such that

$$\alpha \geq \max\{\pi_1^1, \pi_1^2\}, \quad \beta \geq \max\{\pi_2^1, \pi_2^2\}, \quad \text{and} \quad \gamma \leq \min\{\pi_0^1, \pi_0^2\}.$$

It is then possible to formulate a linear program that will generate the most-violated valid inequality that can be derived from a given disjunction, as in the well-known lift-and-project algorithm studied by Balas et al. (1993), Balas et al. (1996), Balas and Perregaard (2003), and Cornuéjols (2008), and based on the earlier work of Balas (1979).

This linear program, the so-called *cut generation LP*, is given by:

$$\begin{aligned}
\min \quad & \alpha\hat{x} + \beta\hat{y} - \gamma \\
\text{s.t.} \quad & \alpha - u^1 A^1 - v^1 a^2 - w^1 a^2 + z^1 \zeta^C a^2 \geq 0 \\
& \alpha - u^2 A^1 - v^2 a^2 + w^2 a^2 + z^2 \eta^C a^2 \geq 0 \\
& \beta - v^1 g^2 + z^1 d^2 \geq 0 \\
& \beta - v^2 g^2 + z^2 d^2 \geq 0 \\
& \gamma - u^1 b^1 - v^1 b^2 - w^1 a^2 \hat{x} + z^1 (\zeta^C a^2 \hat{x} - d^2 y^*) \leq 0 \\
& \gamma - u^2 b^1 - v^2 b^2 + w^2 a^2 \hat{x} + z^2 (\eta^C a^2 \hat{x} - d^2 y^*) \leq 0 \\
& \sum_{i=1}^{m_1} u_i^1 + v^1 + w^1 + z^1 + \sum_{i=1}^{m_1} u_i^2 + v^2 + w^2 + z^2 = 1 \\
& u^1, u^2, v^1, v^2, w^1, w^2, z^1, z^2 \geq 0.
\end{aligned} \tag{3.48}$$

**Recourse Problems.** An interesting special case of (MIBLP) arises when the upper-level objective depends only on the value of the lower-level problem. These problems are referred to as *objective value problems*, or *recourse problems*. Continuous recourse problems are studied in Shimizu et al. (1997) and Patriksson and Wynter (1997), but the treatment of the integer version appears to be limited to the related work in the stochastic programming literature (see, e.g., Caroe and Tind, 1998; Kong et al., 2006).

Formally, we define the recourse version of MIBLP as:

$$\min_{(x,y)\in\mathcal{F}^I} c^1 x + az(x), \tag{3.49}$$

where $z(x)$ is the optimal value of the lower-level problem for fixed $x$ and $a$ is a nonnegative scalar. Intuitively, one might expect this version of the problem to be easier to solve than the general case, because the objectives of the upper- and lower-level DMs are in agreement. Further, problems of the form (3.49) do not require a lower-level solution $y \in Y$, but rather only its *value*, to evaluate the upper-level objective. Because of the special structure of these problems, we are able to develop more compact single-level reformulations and effective algorithms. In this section, we consider the case of (3.49) where $a = 1$, which is precisely MIBLP with $d^1 = d^2$. Before addressing this problem in detail, however, we first describe a general method of bounding the MILP value function

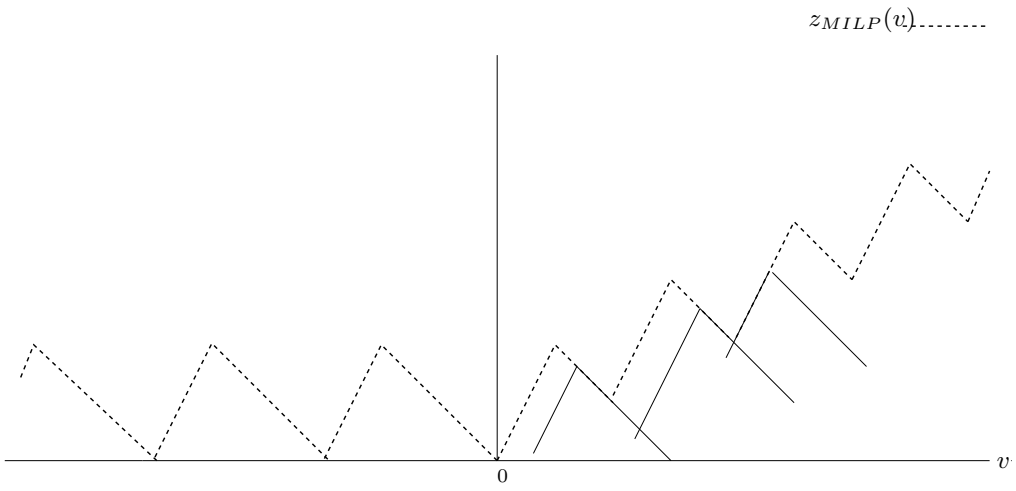



Figure 3.9: A lower-approximation of the value function $z_{MILP}$.

from below.

The bound described next is particularly convenient because it is generated as a natural by-product of the bilevel feasibility check. In particular, if we use branch and bound to solve the lower-level problem, for fixed upper-level solution $x \in X$, we obtain the bound directly from the resulting search tree. A similar bound results if the lower-level problem is solved with branch and cut, rather than branch and bound, but the analogous results require the assumption that a subadditive representation is known for each cut generated. In practice, this is generally not the case. Further, it makes the exposition quite a bit more complicated. The reader is referred to Guzelsoy and Ralphs (2007) for more details on the branch-and-cut case.

Suppose the lower-level MILP (3.19) has a finite optimum and has been solved to optimality by branch and bound for some $\hat{x} \in (\mathcal{P}_U \cap X)$. Let $T$ be the set of feasibly pruned leaf nodes of the resulting tree and let $\hat{w}_t^{\hat{x}} = (w_t^{\hat{x}}, \underline{w}_t^{\hat{x}}, \overline{w}_t^{\hat{x}})$ be the solution of the dual of the LP relaxation at leaf $t$ (i.e. that which allowed us to prune the node). Then, we have the following.

**Theorem 3.22 (Guzelsoy and Ralphs (2007))** *If we define the function*

$$F^{\hat{x}}(v) = \min_{t \in T} w_t^{\hat{x}} v + \underline{w}_t^{\hat{x}} \ell_t^{\hat{x}} - \overline{w}_t^{\hat{x}} u_t^{\hat{x}}, \tag{3.50}$$

*then $F^{\hat{x}}(b^2 - A^2\hat{x}) = z_{MILP}(b^2 - A^2\hat{x})$, where $u_t^{\hat{x}}, \ell_t^{\hat{x}} \in \mathbb{Z}^{n_2}$ are the branching bounds applied to the integer variables in the LP relaxation at $t$.*

$F^{\hat{x}}$ is, in fact, an optimal dual solution to a particular dual of (3.19) (Guzelsoy and Ralphs, 2007). The bounding function is illustrated in Figure 3.9. It is clear that changing the right-hand-side of the primal problem does not affect the constraints of the dual problem. Thus, any function that is

optimal for the dual problem associated with a particular right-hand-side remains feasible for all other right-hand-sides. Further, by weak duality, the objective value of the dual problem evaluated at any feasible solution yields a lower bound on the primal problem. In other words, $F^{\hat{x}}$ satisfies $F^{\hat{x}}(v) \leq z_{MILP}(v)$ for all $v = b^2 - A^2x$ such that $\mathcal{S}(x) \neq \emptyset$. We can derive a global approximation by taking the maximum over a set of such lower-bounding functions. Suppose we knew the lower-bounding function $F^{\hat{x}}$ for all $\hat{x} \in (\mathcal{P}_U \cap X)$. Then, in theory, we could rewrite (MIBLP) as:

$$\begin{aligned} \min \quad & c^1x + d^1y \\ \text{subject to} \quad & A^1x \geq b^1 \\ & A^2x + G^2y \geq b^2 \\ & d^2y = \max_{\hat{x} \in (\mathcal{P}_U \cap X)} F^{\hat{x}}(b^2 - A^2x) \\ & x \in X, y \in Y. \end{aligned} \tag{3.51}$$

While such a reformulation may be of theoretical interest, it does not appear to offer any immediate assistance in the way of computation. Obtaining all such functions $F^{\hat{x}}$, requires solving the lower-level problem for all $x \in (\mathcal{P}_U \cap X)$, which already provides the solution to the original problem. Further, there is no obvious way to form a useful relaxation of (3.51), since we require equality in the optimality constraint, thus constraint generation methods are not immediately applicable. However, this reformulation method may be useful for special cases of the general MIBLP. For example, if the lower-level problem is an LP, (3.50) reduces to

$$F^{\hat{x}}(v) = w^{\hat{x}}v, \tag{3.52}$$

since the problem will be solved at the root node. Note that, since $w^{\hat{x}}$ is the optimal dual solution for the RHS $v$, (3.52) is simply the value function of the lower-level LP. After demonstrating how the reformulation (3.51) can be simplified for general recourse problems, we use this knowledge to reduce the problem even further for problems with continuous lower-level variables.

When we require only the value of the lower-level objective, rather than the actual lower-level solution, (3.51) reduces to:

$$\begin{aligned} \min \quad & c^1x + \theta \\ \text{subject to} \quad & A^1x \geq b^1 \\ & \theta \geq F^{\hat{x}}(b^2 - A^2x), \quad \forall \hat{x} \in \mathcal{P}_U \cap X \\ & x \in X. \end{aligned} \tag{3.53}$$

Note that, in this formulation, we have dropped the lower-level constraints and the requirement

of equality for the bounding constraint. We are able to drop the lower-level constraints because we do not require the lower-level solution $y$ explicitly, and have assumed that $\mathcal{S}(x) \neq \emptyset$, for all $x \in X$. This assumption could easily be relaxed, since lower-level feasibility will necessarily be checked during each bilevel feasibility check, but would require the addition of a feasibility cut to the algorithm below.

On the other hand, we are able to relax the bounding condition because it will be satisfied at optimality. Suppose $(x^*, \theta^*)$ is optimal for (3.53). To satisfy the original constraint, we require the constraint

$$\theta^* \geq \max_{\hat{x} \in \mathcal{P}_U \cap X} F^{\hat{x}}(b^2 - A^2 x^*)$$

to be tight. Suppose this was not the case, and

$$\theta^* > F^{\hat{x}}(b^2 - A^2 x^*), \quad \forall \hat{x} \in \mathcal{P}_U \cap X.$$

This contradicts the optimality of $(x^*, \theta^*)$ since, certainly, setting

$$\theta = \max_{\hat{x} \in \mathcal{P}_U \cap X} F^{\hat{x}}(b^2 - A^2 x^*)$$

would yield a better upper-level objective value. By Theorem 3.22, we have

$$z_{MILP}(b^2 - A^2 x^*) = F^{x^*}(b^2 - A^2 x^*),$$

because the lower bound is guaranteed to be tight for the RHS for which it was obtained. Thus, at optimality,

$$z_{MILP}(b^2 - A^2 x^*) = \max_{\hat{x} \in (\mathcal{P}_U \cap X)} F^{\hat{x}}(b^2 - A^2 x^*),$$

as originally required.

As written, (3.53) still requires a bound for each upper-level solution $\bar{x} \in (\mathcal{P}_U \cap X)$. However, this formulation naturally lends itself to a constraint generation algorithm, starting with the obvious relaxation that arises by substituting a subset $J \subseteq (\mathcal{P}_U \cap X)$:

$$\begin{aligned} \min \quad & c^1 x + \theta \\ \text{subject to} \quad & A^1 x \geq b^1 \\ & \theta \geq F^{\hat{x}}(b^2 - A^2 x), \quad \forall \hat{x} \in J \\ & x \in X. \end{aligned} \tag{3.54}$$

An algorithm then proceeds as follows. For each solution $(\bar{x}, \bar{\theta})$ to the *master problem* (3.54), we

perform the standard bilevel feasibility check by solving the lower-level problem

$$z_{MILP}(b^2 - A^2\bar{x}) = \min_{y \in \mathcal{S}(\bar{x}) \cap Y} d^2 y.$$

If $\bar{\theta} \geq z_{MILP}(b^2 - A^2\bar{x})$, we have found an optimal solution, else we add a cut of the form described in Theorem 3.22 and iterate. This constraint generation algorithm is similar to the well-known Benders' decomposition algorithm for LP and the recent decomposition algorithms for stochastic programs with integer recourse (Caroe and Tind, 1998; Kong et al., 2006). The method is summarized in Algorithm 3.2.

**Algorithm 3.2** Iterative Lower Approximation

1: *Initialization.* Set $J^1 = \emptyset$ and $t \leftarrow 1$.
2: *Iteration* $t$. Solve (3.54) with $J = J^t$ to obtain $(x^t, \theta^t)$. Set $x = x^t$ and solve the lower-level problem, for fixed $x$.

(i) If $\theta^t \geq z_{MILP}(b^2 - A^2 x^t)$, stop. $(x^t, \theta^t)$ is an optimal solution.

(ii) If $\theta^t < z_{MILP}(b^2 - A^2 x^t)$, apply Theorem 3.22 with $x = x^t$ to obtain lower-bounding function $F^t$. Set $J^{t+1} = J^t \cup \{t\}$ and $t \leftarrow t + 1$.

Note that, dropping the assumption of lower-level feasibility would require a third condition in Step 2 of Algorithm 3.2 to cover the possibility of infeasibility. However, the algorithm would proceed in a similar manner and, if infeasibility was detected, the required feasibility cut would be immediately available from the lower-level dual information.

As we alluded to when describing the lower-bounding method, an even further simplification is possible when $Y = \mathbb{R}^{n_2}$. Suppose this is the case and that the lower-level dual feasible set $\{u \in \mathbb{R}^{m_2} \mid uG^2 \leq d^2\}$ is a polytope. Recall that the LP value function can be written as:

$$z_{LP}(v) = \max_{\rho \in \mathcal{R}} \{\rho v\}, \quad \forall v \in \mathbb{R}^{m_2}_+, \tag{3.55}$$

where $\mathcal{R}$ is the set of extreme points of the lower-level dual feasible set. Using this form of the value function allows us to define the LP analog of (3.53):

$$\begin{aligned} \min \quad & c^1 x + \theta \\ \text{subject to} \quad & A^1 x \geq b^1 \\ & \theta \geq \rho(b^2 - A^2 x), \forall \rho \in \mathcal{R} \\ & x \in X. \end{aligned} \tag{3.56}$$

Note that substituting any $\mathcal{R}' \subseteq \mathcal{R}$ in (3.56) yields a relaxation. Thus, algorithmically, we can

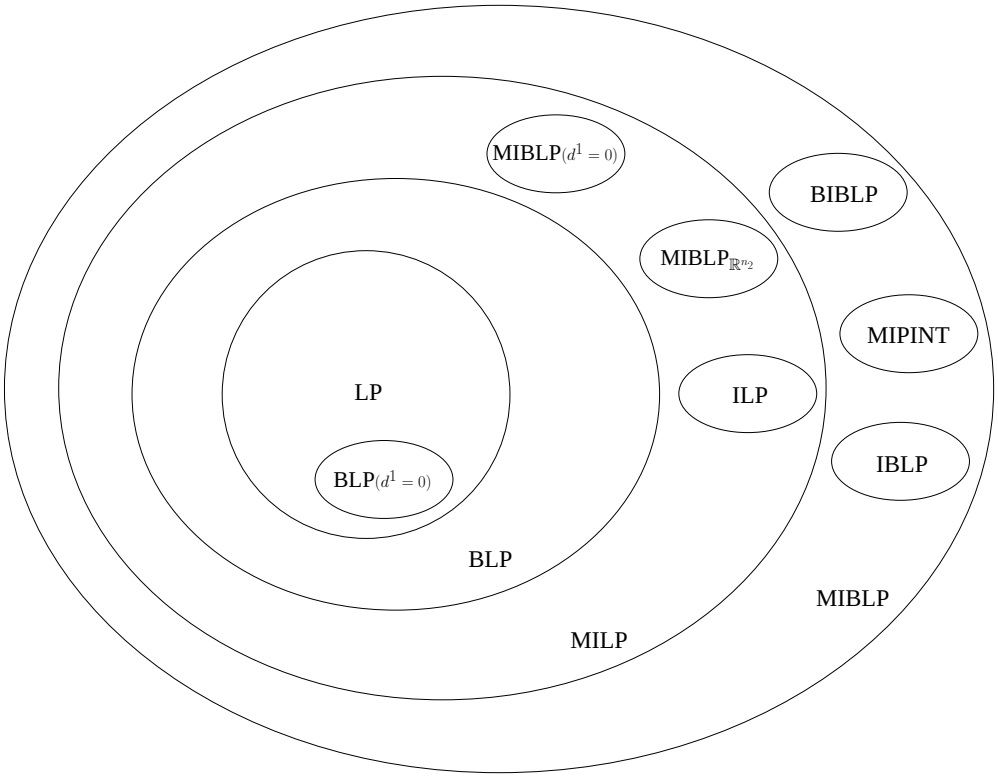



Figure 3.10: Relationships between integer problems.

initialize with some $\mathcal{R}' \subseteq \mathcal{R}$, and iteratively improve our approximation, in a manner similar to the *L-shaped Method* of Slyke and Wets (1969), using the same general method as that described in Algorithm 3.2.

We have examined several different methods of reformulating MIBLPs, with the intent of discovering relationships among the problem subclasses and determining which variants may be approachable via direct methods. The relationships we have discovered are illustrated in Figure 3.10. Note that the relationships shown in the figure are not meant to delineate among complexity classes, but rather show equivalence between variants of MIBLP and known problem classes. While some of the cases discussed above may be suitable for exact solution methods, it is likely that such difficult problems are more effectively tackled by heuristic methods, especially as the problem dimension grows. In the following section, we introduced two such methods.

## 3.3 Heuristic Methods

It should be clear from our discussions of the computational difficulties of solving MIBLPs and MIBLP complexity, that solving the general problem to optimality will be a challenge for problems

of interesting size. As an alternative to the development of exact solution methods, we introduce two heuristic methods that can be used to arrive a good solutions in reasonable computing time. Both heuristic methods described in this section are based in an attempt to balance the upper- and lower-objectives, which can be seen as a balance between optimality and feasibility.

### 3.3.1 Efficient Solutions

In Chapter 2, we presented several heuristics and, during this presentation, discussed the need to balance feasibility and optimality. In essence, we must achieve a balance between the quality of the solutions, with respect to the upper-level objective, and satisfaction of the constraint $y \in M^I(x)$. In contrast to the heuristic methods of Chapter 2, this method attempts to obtain this balance directly, rather than adjusting solutions that favor one condition or the other. To accomplish this, we borrow technology from the multicriteria programming literature to generate feasible solutions derived from efficient solutions to a related multicriteria program:

$$\text{vmin}_{(x,y)\in\Omega^I}[c^1x + d^1y, d^2y]. \tag{3.57}$$

As described in Chapter 1, the goal of (3.57) is to generate solutions $(\hat{x}, \hat{y})$ that are nondominated, or efficient, with the following properties:

- There is no other $(x, y) \in \Omega^I$ such that

$$c^1x + d^1y \leq c^1\hat{x} + d^1\hat{y} \quad \text{and} \quad d^2y \leq d^2\hat{y}.$$

- At least one of

$$c^1x + d^1y < c^1\hat{x} + d^1\hat{y} \quad \text{or} \quad d^2y < d^2\hat{y}$$

  holds.

Because solutions to (3.57) are efficient, they are good candidates for providing a balance between the conditions discussed above.

In our implementation, we find efficient solutions to (3.57) using the weighted-sum subproblem (Geoffrion, 1968):

$$\min_{(x,y)\in\Omega^I} \delta(c^1x + d^1y) + (1-\delta)d^2y, \tag{3.58}$$

for $0 \leq \delta \leq 1$. Recall that solutions to (3.58) for fixed $\delta$ are guaranteed to be efficient, but the converse does not hold. However, for the purposes of a heuristic method, generating a portion of the efficient set is sufficient.

Let $P(\delta)$ denote the subproblem defined by (3.57) with the objective replaced by

$$f = \delta f_1 + (1 - \delta) f_2,$$

where $f_1 = c^1 + d^1 y$ and $f_2 = d^2 y$, as in (3.58). Let $\pi = (x, y)^\delta$ denote a solution to $P(\delta)$. The outcome of $P(\delta)$ is defined as:

$$z^\delta = f(\pi^\delta) = (f_1(\pi^\delta), f_2(\pi^\delta)).$$

Let

$$\delta_{pq} = \frac{z_2^* - z_2^q}{y_1^* - z_1^p + z_2^* - z_2^q}, \tag{3.59}$$

and $N$ be the cardinality of the set of efficient solutions. We call $z^* = (z^1, z^N)$ the *ideal point*. We can then use Algorithm 3.3 to generate a set of efficient solutions to (3.57). Each member of $\mathcal{L}$ is

**Algorithm 3.3** Weighted Sums

1: Solve $P(1)$ and $P(0)$ to identify optimal outcomes $z^1$ and $z^N$, respectively, and $z^* = (z^1, z^N)$. Set $\mathcal{I} = \{(z^1, z^N)\}$ and $\mathcal{L} = \{(\pi^1, z^1, ), (\pi^N, z^N)\}$.
2: While $\mathcal{I} \neq \emptyset$ do:

- Remove any $(z^p, z^q)$ from $I$.
- Compute $\delta_{pq}$ as in (3.59) and solve $P(\delta_{pq})$. If the outcome is $z^p$ or $z^q$, then $z^p$ and $z^q$ are adjacent in the list $(z^1, z^2, \ldots, z^N)$.
- Otherwise, a new outcome $z^r$ is generated. Add $(\pi^r, z^r)$ to $\mathcal{L}$. Add $(z^p, z^r)$ and $(z^r, z^q)$ to $\mathcal{I}$.

a potential candidate for a good bilevel feasible solution. However, we must take one more step to ensure feasibility for (MIBLP). As in our standard bilevel feasibility check, for each $\pi^r \in \mathcal{L}$, we fix the upper-level portion of the solution $x^r$ and solve the resulting lower-level problem to obtain $y^{r^*}$. Combining $x^r$ and $y^{r^*}$ yields a bilevel feasible solution to the original problem. In practice, we select from among these feasible solutions that which has the lowest upper-level solution value.

The heuristic methods discussed in Chapter 2 are primarily meant to be embedded in another algorithmic framework, such as branch and cut, in order to improve its speed. This method can also be used in the same manner. However, it can also be implemented as a stand-alone heuristic algorithm. One major advantage to this algorithm is its applicability to nonlinear problems. Solutions to (3.58) are still guaranteed to be efficient if the objective functions and constraints are nonlinear (Eswaran et al., 1986; Geoffrion, 1968). Thus, this algorithm can be used to find feasible solutions to mixed integer bilevel *nonlinear* programs (MIBNPs). One such problem is discussed in Chapter 5. We demonstrate the effectiveness of the heuristic in Section 3.4.

### 3.3.2 Stationary Point Heuristic

Rather than attempting to balance optimality and feasibility by a combined objective function, we can instead construct solutions composed of upper- and lower-level solutions of high quality with respect to their individual objective functions. We have seen that solutions to the underlying MILP are typically infeasible, due to their suboptimality with respect to the resulting lower-level problem. On the other hand, solutions found by optimizing with respect to the lower-level objective $d^2y$ over $\Omega^I$ are unlikely optimal for the original (upper-level) objective. Thus, we introduce a heuristic aimed at finding an equilibrium between the dual objectives, by combining upper-and lower-level solution components and iterating until we find a solution which cannot be improved with respect to either objective.

Recall the lower-level problem, for fixed $x \in X$:

$$z_{LL}(x) = \min_{y \in \mathcal{S}(x)} d^2 y. \tag{3.60}$$

In a similar manner, we can define the *constrained upper-level problem*, for fixed $y \in Y$:

$$\begin{aligned} \min \quad & c^1 x + d^1 y \\ \text{subject to} \quad & A^1 x \geq b^1 \\ & A^2 x \geq b^2 - G^2 y \\ & x \in X. \end{aligned} \tag{3.61}$$

The main idea of the heuristic is to alternate between solutions to (3.60) and (3.61) until we arrive at a solution $(\hat{x}, \hat{y})$ to (3.61) that is optimal for (3.60), with $x = \hat{x}$. The heuristic is summarized in Algorithm 3.4.

**Algorithm 3.4** Stationary Point Heuristic

1: *Initialization.* Solve

$$\min_{(x,y) \in \Omega^I} c^1 x + d^2 y$$

to obtain an initial solution $(x^0, y^0)$. Set $x = x^0$ and solve (3.60), for fixed $x$, to obtain $y^{*^0}$. If $z_{LL}(x^0) = d^2 y^0$, terminate with optimal solution $(x^0, y^0)$, else fix $y^1 = y^{*^1}$ and set $t \leftarrow 1$.

2: *Iteration* $t$. Solve (3.61) with $y = y^t$ to obtain $(x^t, y^t)$. Set $x = x^t$ and solve (3.60), for fixed $x$, to obtain $y^{*^t}$.

(i) If $d^2 y^t = z_{LL}(x^t)$, stop. $(x^t, y^t)$ is an optimal solution.

(ii) If $d^2 y^t > z_{LL}(x^t)$, fix $y^{t+1} = y^{*^t}$. Set $t \leftarrow t + 1$.

## 3.4 Computational Results

In this section, we examine the performance of our heuristic methods on the problem classes described in Chapter 2. In order to provide some insight into the quality of the solutions generated, we compare the objective values found by the heuristics to the lower bound provided by solving the underlying MILP:

$$\min_{(x,y)\in\Omega^I} c^1x + d^1y,$$

as well as upper bounds on the optimal solution value derived from simple heuristic methods. The first of these upper bounds is obtained by simply fixing the upper-level portion of a solution to (3.4) and solving the lower-level problem. We get our second feasible solution by optimizing over $\Omega^I$ with respect to the lower-level objective function, just as in the Lower-level Priority Heuristic described in Section 2.2.1. In the tables below, these bounds are denoted *MILP Bound*, *Easy Bound*, and *Lower Obj. Bound*, respectively. The results from the Weighed Sums Heuristic are shown Table 3.1 and those from the Stationary Point Heuristic are in Table 3.2. All computational tests were performed on an AMD Opteron Processor 6128 with 32GB of memory.

Of the instances tested, the average gap between the Weighted Sums objective value and that of the underlying MILP is approximately 45%, while the improvement over the best objective obtained by the simple heuristics is approximately 41%. For the Stationary Point Heuristic, the average gap over the MILP bound and the improvement over the simple heuristics was found to be approximately 54% and 10%, respectively. These results seem to imply that the Weighted Sums Heuristic performs better, especially when one considers the negligible difference in computation time. However, for larger instances, the required computational effort may be a larger consideration. In this case, one may prefer to use the Stationary Point Heuristic, as it required roughly half the computation time, on average.

We also compared the objective values found by the Weighted Sums and Stationary Point methods to the best known value obtained by our solver, MibS. A full comparison is shown in Table 3.3, where the minimum value obtained by the heuristic methods is in bold. From the table, we can see that the best objective value obtained by MibS is always less than that obtained by the heuristic methods. However, in 11 of the 50 instances tested, at least one of the heuristics performs just as well as MibS. The average increase in objective value over the best known MibS solution, hereafter *MibS gap*, is approximately 34%, but in 24 out of 50 instances the MibS gap is less than 10%, suggesting that a large amount of computational effort can be avoided, with fairly minimal solution quality detriment. These results demonstrate that the heuristic methods find reasonably good solutions with very little computational effort.

From Table 3.3, we can also see that that Weighted Sums Heuristic obtained a lower objective value

than that of the Stationary Point Heuristic in 25 instances, while the reverse was true only 6 times. Further, for those instances in which the Weighted Sums Heuristic performed better, it tended to beat the Stationary Point Heuristic by approximately 74%, on average. On the other hand, when the Stationary Point Heuristic yielded the better solution, it was only about 2% better than the Weighted Sums solution, on average.

The performance profiles (Dolan and Moré, 2002) for the MibS gap are shown in Figure 3.11. The results were altered slightly, to improve the effectiveness of the presentation. First, each MibS gap was increased by a small $\epsilon$, to ensure that the gaps of zero would be handled accurately. Second, all instances for which the heuristics obtained equal objectives were removed from the performance profile, to provide a better comparison on those instances for which their performance differs. From the figure, we can see that the Weighted Sums Heuristic resulted in a smaller MibS gap in 80% of the instances, and clearly dominates the Stationary Point Heuristic, with respect to solution quality, on our test set.

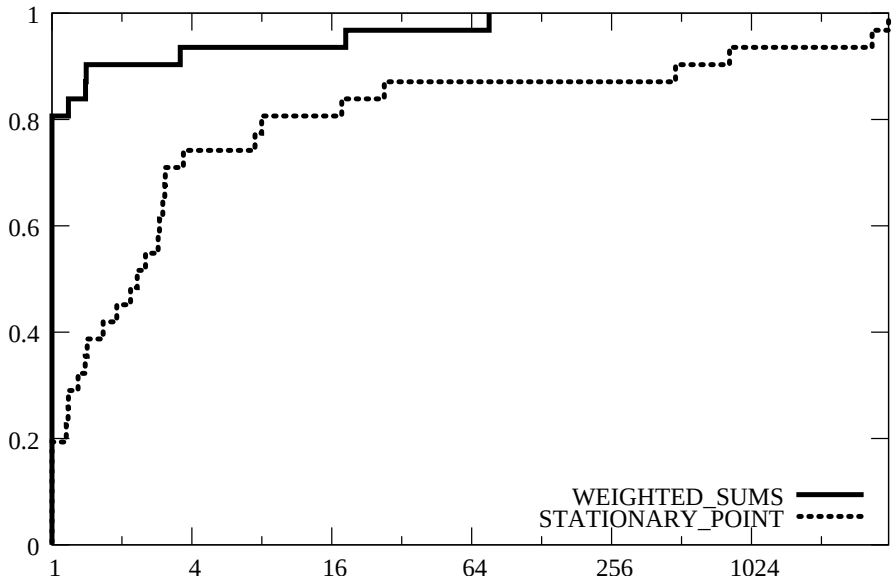



Figure 3.11: Performance Profiles for the two heuristic methods.

| Instance | CPU (s) | No. Subproblems | Weighted Sums Obj. | MILP Bound | Easy Bound | Lower Obj. Bound |
|---|---|---|---|---|---|---|
| miblp-20-20-50-0110-5-1 | 2.92 | 23 | -502.0 | -642.0 | -502.0 | 358.0 |
| miblp-20-20-50-0110-5-2 | 1.33 | 15 | -460.0 | -863.0 | -405.0 | -67.0 |
| miblp-20-20-50-0110-5-3 | 1.06 | 13 | -477.0 | -477.0 | -477.0 | 60.0 |
| miblp-20-20-50-0110-5-4 | 0.94 | 5 | -753.0 | -753.0 | -753.0 | -560.0 |
| miblp-20-20-50-0110-5-5 | 1.17 | 15 | -392.0 | -392.0 | -392.0 | -75.0 |
| miblp-20-20-50-0110-5-6 | 5.18 | 23 | -1061.0 | -1166.0 | -938.0 | -40.0 |
| miblp-20-20-50-0110-5-7 | 0.93 | 17 | -547.0 | -551.0 | -502.0 | 635.0 |
| miblp-20-20-50-0110-5-8 | 4.23 | 19 | -936.0 | -936.0 | -936.0 | 156.0 |
| miblp-20-20-50-0110-5-9 | 0.64 | 13 | -689.0 | -889.0 | -689.0 | -339.0 |
| miblp-20-20-50-0110-5-10 | 1.5 | 15 | -290.0 | -374.0 | -290.0 | 417.0 |
| miblp-20-20-50-0110-10-1 | 2.67 | 21 | -232.0 | -779.0 | -119.0 | -16.0 |
| miblp-20-20-50-0110-10-2 | 0.66 | 13 | -634.0 | -709.0 | -360.0 | -269.0 |
| miblp-20-20-50-0110-10-3 | 3.39 | 27 | -451.0 | -659.0 | -254.0 | 338.0 |
| miblp-20-20-50-0110-10-4 | 3.65 | 25 | -579.0 | -892.0 | -579.0 | -55.0 |
| miblp-20-20-50-0110-10-5 | 0.91 | 21 | -1003.0 | -1003.0 | -1003.0 | 414.0 |
| miblp-20-20-50-0110-10-6 | 5.43 | 25 | -589.0 | -964.0 | -589.0 | 17.0 |
| miblp-20-20-50-0110-10-7 | 3.62 | 25 | -591.0 | -1078.0 | -440.0 | -199.0 |
| miblp-20-20-50-0110-10-8 | 5.12 | 44 | -231.0 | -760.0 | -231.0 | 293.0 |
| miblp-20-20-50-0110-10-9 | 0.28 | 17 | -121.0 | -428.0 | 176.0 | 318.0 |
| miblp-20-20-50-0110-10-10 | 2.46 | 23 | 162.0 | -721.0 | 162.0 | 623.0 |
| miblp-20-20-50-0110-15-1 | 2.57 | 21 | -14.0 | -841.0 | 67.0 | 140.0 |
| miblp-20-20-50-0110-15-2 | 7.13 | 19 | -629.0 | -874.0 | -525.0 | -241.0 |
| miblp-20-20-50-0110-15-3 | 4.64 | 27 | -593.0 | -836.0 | -321.0 | -94.0 |
| miblp-20-20-50-0110-15-4 | 3.66 | 19 | 13.0 | -688.0 | 13.0 | 13.0 |
| miblp-20-20-50-0110-15-5 | 3.57 | 31 | 373.0 | -840.0 | 548.0 | 614.0 |
| miblp-20-20-50-0110-15-6 | 0.64 | 17 | -569.0 | -1151.0 | -569.0 | -569.0 |
| miblp-20-20-50-0110-15-7 | 2.99 | 17 | -443.0 | -782.0 | -387.0 | -131.0 |
| miblp-20-20-50-0110-15-8 | 0.87 | 17 | -158.0 | -1000.0 | 182.0 | 138.0 |
| miblp-20-20-50-0110-15-9 | 0.48 | 11 | -563.0 | -803.0 | -544.0 | -317.0 |
| miblp-20-20-50-0110-15-10 | 0.37 | 11 | -118.0 | -345.0 | 85.0 | 185.0 |
| miblp-30-20-50-0110-10-1 | 0.96 | 7 | -401.0 | -528.0 | -296.0 | -223.0 |
| miblp-30-20-50-0110-10-2 | 1.57 | 17 | -169.0 | -581.0 | -122.0 | 193.0 |
| miblp-30-20-50-0110-10-3 | 3.49 | 25 | -638.0 | -961.0 | 74.0 | 237.0 |
| miblp-30-20-50-0110-10-4 | 1.16 | 19 | 437.0 | -374.0 | 437.0 | 437.0 |
| miblp-30-20-50-0110-10-5 | 0.02 | 2 | -135.0 | -135.0 | -135.0 | -123.0 |
| miblp-30-20-50-0110-10-6 | 0.61 | 13 | -168.0 | -660.0 | -90.0 | 426.0 |
| miblp-30-20-50-0110-10-7 | 1.7 | 15 | -361.0 | -536.0 | -278.0 | -116.0 |
| miblp-30-20-50-0110-10-8 | 6.2 | 19 | -450.0 | -646.0 | -417.0 | -74.0 |
| miblp-30-20-50-0110-10-9 | 6.24 | 35 | -323.0 | -1028.0 | -177.0 | 319.0 |
| miblp-30-20-50-0110-10-10 | 1.33 | 13 | -160.0 | -275.0 | -104.0 | 87.0 |
| miblp-40-20-50-0110-10-1 | 0.68 | 11 | -198.0 | -237.0 | -121.0 | 25.0 |
| miblp-40-20-50-0110-10-2 | 1.88 | 19 | -85.0 | -578.0 | -58.0 | -78.0 |
| miblp-40-20-50-0110-10-3 | 2.35 | 17 | -513.0 | -766.0 | -299.0 | 19.0 |
| miblp-40-20-50-0110-10-4 | 1.19 | 13 | -236.0 | -371.0 | -199.0 | 90.0 |
| miblp-40-20-50-0110-10-5 | 1.48 | 15 | -316.0 | -550.0 | -316.0 | -50.0 |
| miblp-40-20-50-0110-10-6 | 2.06 | 21 | -372.0 | -485.0 | -261.0 | 637.0 |
| miblp-40-20-50-0110-10-7 | 3.2 | 31 | -911.0 | -1275.0 | -433.0 | 315.0 |
| miblp-40-20-50-0110-10-8 | 4.14 | 19 | -682.0 | -961.0 | -542.0 | -411.0 |
| miblp-40-20-50-0110-10-9 | 3.15 | 15 | -603.0 | -916.0 | -568.0 | -506.0 |
| miblp-40-20-50-0110-10-10 | 1.49 | 21 | -395.0 | -515.0 | -395.0 | 403.0 |

Table 3.1: Results from the Weighted Sums Heuristic.

| Instance | CPU (s) | No. Subproblems | Stationary Point Obj. | MILP Bound | Easy Bound | Lower Obj. Bound |
|---|---|---|---|---|---|---|
| miblp-20-20-50-0110-5-1 | 1.43 | 6 | -502.0 | -642.0 | -502.0 | 358.0 |
| miblp-20-20-50-0110-5-2 | 1.27 | 3 | -489.0 | -863.0 | -405.0 | -67.0 |
| miblp-20-20-50-0110-5-3 | 0.17 | 0 | -477.0 | -477.0 | -477.0 | 60.0 |
| miblp-20-20-50-0110-5-4 | 0.42 | 0 | -753.0 | -753.0 | -753.0 | -560.0 |
| miblp-20-20-50-0110-5-5 | 0.18 | 0 | -392.0 | -392.0 | -392.0 | -75.0 |
| miblp-20-20-50-0110-5-6 | 0.32 | 1 | -1010.0 | -1166.0 | -938.0 | -40.0 |
| miblp-20-20-50-0110-5-7 | 0.24 | 2 | -502.0 | -551.0 | -502.0 | 635.0 |
| miblp-20-20-50-0110-5-8 | 0.49 | 0 | -936.0 | -936.0 | -936.0 | 156.0 |
| miblp-20-20-50-0110-5-9 | 0.59 | 7 | -689.0 | -889.0 | -689.0 | -339.0 |
| miblp-20-20-50-0110-5-10 | 0.1 | 1 | -290.0 | -374.0 | -290.0 | 417.0 |
| miblp-20-20-50-0110-10-1 | 0.39 | 2 | -152.0 | -779.0 | -119.0 | -16.0 |
| miblp-20-20-50-0110-10-2 | 0.36 | 2 | -567.0 | -709.0 | -360.0 | -269.0 |
| miblp-20-20-50-0110-10-3 | 0.95 | 3 | -254.0 | -659.0 | -254.0 | 338.0 |
| miblp-20-20-50-0110-10-4 | 0.78 | 1 | -592.0 | -892.0 | -579.0 | -55.0 |
| miblp-20-20-50-0110-10-5 | 0.05 | 0 | -1003.0 | -1003.0 | -1003.0 | 414.0 |
| miblp-20-20-50-0110-10-6 | 9.67 | 7 | -589.0 | -964.0 | -589.0 | 17.0 |
| miblp-20-20-50-0110-10-7 | 1.48 | 6 | -454.0 | -1078.0 | -440.0 | -199.0 |
| miblp-20-20-50-0110-10-8 | 1.49 | 3 | -231.0 | -760.0 | -231.0 | 293.0 |
| miblp-20-20-50-0110-10-9 | 0.14 | 1 | 157.0 | -428.0 | 176.0 | 318.0 |
| miblp-20-20-50-0110-10-10 | 0.57 | 3 | 162.0 | -721.0 | 162.0 | 623.0 |
| miblp-20-20-50-0110-15-1 | 0.49 | 2 | 67.0 | -841.0 | 67.0 | 140.0 |
| miblp-20-20-50-0110-15-2 | 1.49 | 3 | -525.0 | -874.0 | -525.0 | -241.0 |
| miblp-20-20-50-0110-15-3 | 0.79 | 3 | -357.0 | -836.0 | -321.0 | -94.0 |
| miblp-20-20-50-0110-15-4 | 0.85 | 1 | 13.0 | -688.0 | 13.0 | 13.0 |
| miblp-20-20-50-0110-15-5 | 0.97 | 2 | 548.0 | -840.0 | 548.0 | 614.0 |
| miblp-20-20-50-0110-15-6 | 0.12 | 1 | -569.0 | -1151.0 | -569.0 | -569.0 |
| miblp-20-20-50-0110-15-7 | 5.02 | 3 | -387.0 | -782.0 | -387.0 | -131.0 |
| miblp-20-20-50-0110-15-8 | 0.44 | 1 | 114.0 | -1000.0 | 182.0 | 138.0 |
| miblp-20-20-50-0110-15-9 | 0.2 | 1 | -544.0 | -803.0 | -544.0 | -317.0 |
| miblp-20-20-50-0110-15-10 | 0.16 | 1 | 85.0 | -345.0 | 85.0 | 185.0 |
| miblp-30-20-50-0110-10-1 | 0.88 | 2 | -308.0 | -528.0 | -296.0 | -223.0 |
| miblp-30-20-50-0110-10-2 | 0.41 | 2 | -122.0 | -581.0 | -122.0 | 193.0 |
| miblp-30-20-50-0110-10-3 | 0.72 | 1 | 30.0 | -961.0 | 74.0 | 237.0 |
| miblp-30-20-50-0110-10-4 | 0.15 | 1 | 437.0 | -374.0 | 437.0 | 437.0 |
| miblp-30-20-50-0110-10-5 | 0.03 | 0 | -135.0 | -135.0 | -135.0 | -123.0 |
| miblp-30-20-50-0110-10-6 | 0.06 | 1 | -90.0 | -660.0 | -90.0 | 426.0 |
| miblp-30-20-50-0110-10-7 | 0.65 | 2 | -365.0 | -536.0 | -278.0 | -116.0 |
| miblp-30-20-50-0110-10-8 | 6.32 | 6 | -450.0 | -646.0 | -417.0 | -74.0 |
| miblp-30-20-50-0110-10-9 | 0.94 | 3 | -177.0 | -1028.0 | -177.0 | 319.0 |
| miblp-30-20-50-0110-10-10 | 0.47 | 1 | -104.0 | -275.0 | -104.0 | 87.0 |
| miblp-40-20-50-0110-10-1 | 0.18 | 1 | -131.0 | -237.0 | -121.0 | 25.0 |
| miblp-40-20-50-0110-10-2 | 0.54 | 2 | -85.0 | -578.0 | -58.0 | -78.0 |
| miblp-40-20-50-0110-10-3 | 0.61 | 1 | -485.0 | -766.0 | -299.0 | 19.0 |
| miblp-40-20-50-0110-10-4 | 0.6 | 2 | -236.0 | -371.0 | -199.0 | 90.0 |
| miblp-40-20-50-0110-10-5 | 0.69 | 2 | -525.0 | -550.0 | -316.0 | -50.0 |
| miblp-40-20-50-0110-10-6 | 0.6 | 2 | -380.0 | -485.0 | -261.0 | 637.0 |
| miblp-40-20-50-0110-10-7 | 0.57 | 2 | -693.0 | -1275.0 | -433.0 | 315.0 |
| miblp-40-20-50-0110-10-8 | 1.99 | 2 | -682.0 | -961.0 | -542.0 | -411.0 |
| miblp-40-20-50-0110-10-9 | 1.71 | 2 | -568.0 | -916.0 | -568.0 | -506.0 |
| miblp-40-20-50-0110-10-10 | 0.18 | 1 | -398.0 | -515.0 | -395.0 | 403.0 |

Table 3.2: Results from the Stationary Point Heuristic.

| Instance | MibS Obj. | Weighted Sums Obj. | Stationary Point Obj | Best Heuristic | Increase (%) |
|---|---|---|---|---|---|
| miblp-20-20-50-0110-5-1 | -548 | -502 | -502 | -502 | 8.39 |
| miblp-20-20-50-0110-5-2 | -561 | -460 | **-489** | -489 | 12.83 |
| miblp-20-20-50-0110-5-3 | -477 | -477 | -477 | -477 | — |
| miblp-20-20-50-0110-5-4 | -753 | -753 | -753 | -753 | — |
| miblp-20-20-50-0110-5-5 | -392 | -392 | -392 | -392 | — |
| miblp-20-20-50-0110-5-6 | -1061 | **-1061** | -1010 | -1061 | — |
| miblp-20-20-50-0110-5-7 | -547 | **-547** | -502 | -547 | — |
| miblp-20-20-50-0110-5-8 | -936 | -936 | -936 | -936 | — |
| miblp-20-20-50-0110-5-9 | -877 | -689 | -689 | -689 | 21.44 |
| miblp-20-20-50-0110-5-10 | -340 | -290 | -290 | -290 | 14.71 |
| miblp-20-20-50-0110-10-1 | -353 | **-232** | -152 | -232 | 34.28 |
| miblp-20-20-50-0110-10-2 | -659 | **-634** | -567 | -634 | 3.79 |
| miblp-20-20-50-0110-10-3 | -618 | **-451** | -254 | -451 | 27.02 |
| miblp-20-20-50-0110-10-4 | -597 | -579 | **-592** | -592 | 0.84 |
| miblp-20-20-50-0110-10-5 | -1003 | -1003 | -1003 | -1003 | — |
| miblp-20-20-50-0110-10-6 | -672 | -589 | -589 | -589 | 12.35 |
| miblp-20-20-50-0110-10-7 | -657 | **-591** | -454 | -591 | 10.05 |
| miblp-20-20-50-0110-10-8 | -667 | -231 | -231 | -231 | 65.37 |
| miblp-20-20-50-0110-10-9 | -256 | **-121** | 157 | -121 | 52.73 |
| miblp-20-20-50-0110-10-10 | -429 | 162 | 162 | 162 | 137.76 |
| miblp-20-20-50-0110-15-1 | -289 | **-14** | 67 | -14 | 95.16 |
| miblp-20-20-50-0110-15-2 | -645 | **-629** | -525 | -629 | 2.48 |
| miblp-20-20-50-0110-15-3 | -593 | **-593** | -357 | -593 | — |
| miblp-20-20-50-0110-15-4 | -396 | 13 | 13 | 13 | 103.28 |
| miblp-20-20-50-0110-15-5 | -75 | **373** | 548 | 373 | 597.33 |
| miblp-20-20-50-0110-15-6 | -596 | -569 | -569 | -569 | 4.53 |
| miblp-20-20-50-0110-15-7 | -471 | **-443** | -387 | -443 | 5.94 |
| miblp-20-20-50-0110-15-8 | -301 | **-158** | 114 | -158 | 47.51 |
| miblp-20-20-50-0110-15-9 | -584 | **-563** | -544 | -563 | 3.60 |
| miblp-20-20-50-0110-15-10 | -251 | **-118** | 85 | -118 | 52.99 |
| miblp-30-20-50-0110-10-1 | -471 | **-401** | -308 | -401 | 14.86 |
| miblp-30-20-50-0110-10-2 | -478 | **-169** | -122 | -169 | 64.64 |
| miblp-30-20-50-0110-10-3 | -678 | **-638** | 30 | -638 | 5.90 |
| miblp-30-20-50-0110-10-4 | 207 | 437 | 437 | 437 | 111.11 |
| miblp-30-20-50-0110-10-5 | -135 | -135 | -135 | -135 | — |
| miblp-30-20-50-0110-10-6 | -171 | **-168** | -90 | -168 | 1.75 |
| miblp-30-20-50-0110-10-7 | -375 | -361 | **-365** | -365 | 2.67 |
| miblp-30-20-50-0110-10-8 | -461 | -450 | -450 | -450 | 2.39 |
| miblp-30-20-50-0110-10-9 | -672 | **-323** | -177 | -323 | 51.93 |
| miblp-30-20-50-0110-10-10 | -168 | **-160** | -104 | -160 | 4.76 |
| miblp-40-20-50-0110-10-1 | -198 | **-198** | -131 | -198 | — |
| miblp-40-20-50-0110-10-2 | -120 | -85 | -85 | -85 | 29.17 |
| miblp-40-20-50-0110-10-3 | -675 | **-513** | -485 | -513 | 24.00 |
| miblp-40-20-50-0110-10-4 | -270 | -236 | -236 | -236 | 12.59 |
| miblp-40-20-50-0110-10-5 | -537 | -316 | **-525** | -525 | 2.23 |
| miblp-40-20-50-0110-10-6 | -425 | -372 | **-380** | -380 | 10.59 |
| miblp-40-20-50-0110-10-7 | -1028 | **-911** | -693 | -911 | 11.38 |
| miblp-40-20-50-0110-10-8 | -849 | -682 | -682 | -682 | 19.67 |
| miblp-40-20-50-0110-10-9 | -800 | **-603** | -568 | -603 | 24.63 |
| miblp-40-20-50-0110-10-10 | -398 | -395 | **-398** | -398 | — |

Table 3.3: Comparison against optimal solutions.

# Chapter 4

# Applications in Interdiction

In this chapter, we discuss applications of bilevel programming. In particular, we describe the problem class whose utility in homeland security and infrastructure protection protection planning originally lead us to study discrete bilevel programs. This class of problems can be used to model methods aimed at attack prevention and mitigation, enemy operations disruption, or early warning system (EWS) design, for example, and encompasses network interdiction models, a problem class of crucial importance for homeland security applications. Recall from Chapter 1, the formulation of MIPINT:

$$z_{MIPINT} = \max_{x \in \mathcal{P}_U^{\text{INT}} \cap \mathbb{B}^n} \min_{y \in \mathcal{S}_L^{\text{INT}}(x) \cap Y} dy \qquad \text{(MIPINT)}$$

where

$$\mathcal{P}_U^{\text{INT}} = \left\{ x \in \mathbb{R}^n \mid A^1 x \leq b^1 \right\}$$
$$\mathcal{S}_L^{\text{INT}}(x) = \left\{ y \in \mathbb{R}^n \mid G^2 y \geq b^2, -y \geq -U(e - x), y \geq 0 \right\}$$

and $Y = (\mathbb{Z}^p \times \mathbb{R}^{n-p}) \subseteq \mathbb{R}^n$. Aside from their wide applicability, the interdiction models described in this chapter are of interest because of their special structure, which can be exploited for more effective algorithm design. Before describing the algorithmic methods that result, however, we first introduce a particular EWS and discuss the ILP used to optimize its design. After studying the underlying problem in depth, we motivate a particular bilevel extension of the model that can be used to conduct a form of systematic sensitivity analysis, thereby further illustrating the utility of interdiction problems.

The EWS design problem we consider is that of optimizing a novel acoustic leakage detection system for urban water distribution networks. The system is composed of detectors and transponders placed in water hydrants, with the goal of providing a desired coverage under given budget restrictions. We model the problem as a particular Prize-Collecting Steiner Arborescence (PCSA)

problem, and present a branch-and-cut-and-bound approach which exploits its special structure. After presenting the exact algorithm, we demonstrate how to obtain approximations of provably high quality, by employing a suitable stopping criteria, and test both the exact and approximate algorithm on series of problems composed of both real water distribution networks and randomly-generated instances. Implicit in our model is the assumption that any number of detectors may be installed, as long as it is beneficial to do so. We test the sensitivity of our algorithm to this assumption by introducing a detector limit and systematically altering its value. We then present the bilevel extension of the model used for an alternative sensitivity analysis and describe specialized methods to solve the resulting bilevel program. Two novel classes of valid inequalities for bilevel problems with binary upper-level variables are derived, and a greedy interdiction heuristic method is suggested. The combination of the methods described here yields a solver customization for interdiction problems. This customization has been implemented in MibS. After describing the methods, we provide results from the customized solver for both the full customization, as well as the heuristic as a standalone method.

## 4.1 Leakage Detection Sensor Location

In this section, we describe a model whose goal is to find the most effective strategy for monitoring the structural integrity of a water distribution network. In particular, we seek to determine the optimal placement, with respect to installation cost and resulting benefit, of leakage detection sensors within the water network. This system is one example of an EWS–an alarm is triggered when a possible vulnerability in the system is discovered, allowing us the opportunity to investigate before a major disruption occurs.

It is clear that leakages can be a major concern in urban water distribution networks - the damage caused by a leaky pipe in the network can range from sizable water loss to catastrophic damage to people and buildings, depending on the size and location of the leak. Therefore, development of an effective monitoring system for early detection of water losses is of significant importance to network managers.

Here, we consider the optimization of a network of acoustic water leakage sensors and accompanying radio relays being tested by the city of Lausanne, Switzerland. Various such systems have been proposed in the past. The particular technology underlying the following is called *LORNO*, and is composed of acoustic sensors placed at various hydrants and transponders that store and transmit the monitored and received information from other transponders to a central station. Each acoustic sensor “hears” problematic signals within a neighborhood defined by its placement and dependent on local network topology and geometry; such a neighborhood must be estimated for each potential placement. For each each hydrant within the system, we have the option of installing

- a sensor and a transponder,
- a transponder only, or
- nothing at all.

Clearly, installing a sensor with no means of communication (i.e. a transponder) is not sensible, but supplementary transponders may be needed to carry all information to the central station, and can be installed at a lower cost than the full system.

For a given city, this gives rise to a family of combinatorial optimization problems. One example is:

*Given the set of hydrants and the neighborhood covered by each sensor, find a minimum cost placement of sensors covering the entire network, as well as of transponders enabling the corresponding information to be transmitted to a central station.*

Another version is:

*For a given budget, find a maximum utility placement of sensors and transponders, where utility is measured by the information transmitted and the location from where it is collected.*

Such optimal placement problems in a water distribution network have been formulated as directly as ILPs (see Carr et al., 2006). However, the problems we consider here are more specific and contain a hard constraint, namely that which ensures that the solution induces a connected subgraph of a given network, in order to transmit data to the central station. These connectivity constraints lead us to model the problems posed above as variants of Steiner's Problem and the Prize Collecting Steiner's Problem, respectively. These problems known to be $\mathsf{NP}-$hard , except for special graphs (e.g. Margot et al., 1994), but polyhedral approaches like those described by Johnson et al. (2000), Fischetti (1991), Goemans and Williamson (1995), and Goemans and Williamson (1997) may help to find optimal or good approximate solutions.

In this section, we present a novel branch-and-bound-and-cut approach for solving these problems and compare it with others from the literature. Our approach is initially tested on real data from Lausanne's water supply network. Then, in order to provide better-founded empirical validation of our approach, we also test it on specially-constructed water supply systems, tailored to be realistic. For a more concise description of this problem and the resulting methodological approach, the reader is referred to the paper of Prodon et al. (2010).

The remainder of the section is structured as follows. First, we give a description of the *LORNO* system and mathematical formulations of the optimization problems. We next describe our approach for solving the Prize Collecting Steiner Arborescence Problem. Then, we present and discuss our computational results, both on real-world and realistically-simulated models. Finally, we describe a bilevel extension of the model that allows a specific type of sensitivity analysis to be performed.

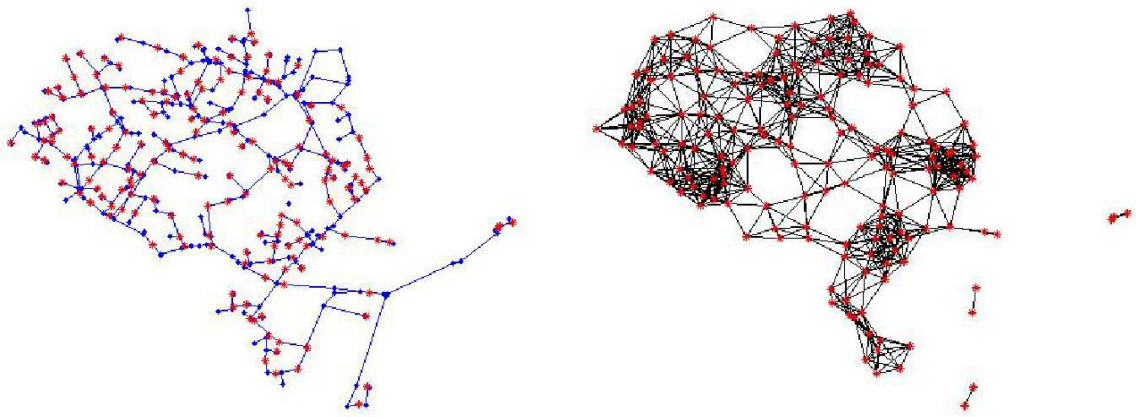
### 4.1.1 The Leakage Problem

The motivation for this work arises from problems experienced by the the city of Lausanne, Switzerland. In the Lausanne water distribution network, water losses of approximately 20% have been experienced. In other cities, this loss proportion is known to be as high as 50%. Such losses may have many different causes, such as leaky or broken pipes or unregistered utilization, resulting from exercises performed by local fire departments. In a region where water supply is not a problem, losses due to unregistered utilization are not crucial, but broken pipes have the potential to cause serious damage and related cost (e.g., traffic perturbation, floods, water distribution break-down, contamination). In a distribution system, leaks are often the best available sign that a pipe is not structurally stable, making leak detection crucial for the network manager.

The *LORNO* system (Hinni, 2010), hereafter *LORNO*, has been developed to detect leaks in a water distribution network. *LORNO* relies on recognizing the unusual noises that arise in the pipes due to the leaks. It consists of units placed in the hydrants and a central server for data collection. Each full unit is composed of an auditory component, installed within the hydrant, and a radio transmitter installed on external portion of the hydrant. The auditory function is comprised of an acoustic sensor coupled with electronic chips for signal analysis. The acoustic sensor is capable of receiving signals from all pipes within a certain surrounding area, whose breadth depends on the network topology, and measuring the amount of water drawn from the hydrant in which it is placed. This data is transmitted via radio signals to the central server, and a leak report is generated if the values do not match the stored reference data. In order to limit electricity consumption, low power transmitters, capable of communicating with neighboring hydrants located within a certain distance - 200 to 500 meters, depending on the physical obstacles - are used. Thus, the signals must be transmitted from hydrant to hydrant, through what we call the communication network, until the server is reached.

Based on historical data and professional experience, the engineer's rule-of-thumb suggests that equipping half of the hydrants in a water network with full *LORNO* units is sufficient for leak detection, and even equipping one third of the hydrants already provides good coverage. However, if a hydrant is unable to transmit its signal to the central server, because it lies outside of the feasible communication range, its information is lost. Thus, having a connected communication network is essential. In order to achieve a connected network and ensure that all data collected may be transmitted, it is also possible to equip a hydrant with only a radio transmitter, rather than the full installation. Thus, for each hydrant, the system operator must decide if he will equip it with a full *LORNO* unit, a radio transmitter only, or nothing at all.

Figure 4.1 shows an example of a mid-sized water distribution network in Lausanne, Switzerland. The network contains 173 hydrants, represented in the figure with stars. In Figure 4.1(a), the physical pipe network is shown. Here, the edges correspond to the actual pipes that connect the hydrants



(a) The physical pipe network. (b) The communication network.

Figure 4.1: A water distribution network in Lausanne, Switzerland.

in the network. On the other hand, in Figure 4.1(b), the edges indicate that two hydrants are able to communicate, and the graph represents a mathematical description of the communication ability among the hydrants. The optimization problem facing the system operator is then the following: choose a subset of hydrants to equip with a full *LORNO* unit and possibly a subset of hydrants to equip with a radio transmitter only, in order to get a connected communication network, and minimize (maximize) the expected cost (profit).

The cost of installing a *LORNO* system consists of two components. There is a fixed price for the necessary software and the overhead for the central server, as well as a cost proportional to the number of *LORNO* units that must be acquired. For our modeling purposes, we decompose the cost of installing a *LORNO* unit into the cost of the radio transmitter plus the cost of the rest of the installation. It is important to note that the cost of installing *each* unit is identical. To some extent, this is due to the pricing strategy of the *LORNO* producer. However, while one could imagine that certain below-ground installations may be more challenging, and thus more costly, it is reasonable to assume that the cost of installing the transmitter would be independent of location. We will see the implications of this cost structure in the following sections.

The profit of installing a *LORNO* unit is more difficult to quantify. The benefit of installing a *LORNO* unit at a particular site depends on the probability of a pipe breaking at that site, and the potential damage caused by such an event. These quantities may vary depending on material and age of the pipes and, of course, on the surrounding environment (presence of residences, industry, hospitals, electricity, or telecom cables, and so on). In the real-world data available from Lausanne, these factors had not been evaluated, and we use only the natural assumption that the benefit of one unit is proportional to the length of the pipes within its neighborhood. However, determining more accurate benefit for *LORNO*, and other types of sensors measuring network stability, is an

interesting area of future work.

We also assume that the pipes are discretized into small enough pieces such that each is "scanned" entirely, or not at all, from a given hydrant. Note that such a piece may be heard by more than one hydrant, but the corresponding benefit should be counted only once. Here, it is tacitly assumed that redundant detection is not necessary to protect against equipment failure, since it tests itself continually by sending around appropriate signals. It should be noted, however, that this feature is specific to *LORNO*, and if one wishes to model an alternate system, this functionality should be verified before proceeding.

### 4.1.2 The LORNO Sensor Location Model

We model the optimization problem as a rooted PCSA problem in a directed graph $G = (V, E)$, constructed as follows. Let $H$ be the set of hydrants and $R$ the set of pipe pieces. For each hydrant $h_i \in H$ we introduce two nodes $\overline{h_i}$ and $\underline{h_i}$ and for each pipe piece $r_i \in R$, a node denoted also $r_i$. Denote these node sets by $\overline{H}, \underline{H}$ and $R$, respectively. Then, $V = \{r_0\} \cup \overline{H} \cup \underline{H} \cup R$, where $r_0$ is a special node, the root. It is easier to draw these nodes at different levels: the root $r_0$ at level 0, $\overline{H}$ at level 1, $\underline{H}$ at level 2 and $R$ at level 3. The arcs of graph $G$ are decomposed into four types:

(i) Arcs $(r_0, \overline{h_i})$, for $\overline{h_i} \in \overline{H}$, with zero cost

(ii) Opposite arcs $\{(\overline{h_i}, \overline{h_j}), (\overline{h_j}, \overline{h_i})\}$ for each pair of hydrants that can communicate by radio station (i.e., for each edge in the communication network), with the cost $c_{(\overline{h_i}, \overline{h_j})}$ and $c_{(\overline{h_j}, \overline{h_i})}$ of radio installation on hydrant $\overline{h_j}$ and $\overline{h_i}$, respectively.

(iii) An arc $(\overline{h_i}, \underline{h_i})$ for each hydrant, with the cost $c_{(\overline{h_i}, \underline{h_j})}$ of auditory installation (i.e. the cost of *LORNO* minus the cost of radio installation) on hydrant $i$

(iv) An arc $(\underline{h_i}, r_j)$, for each region $r_j \in R$ that can be heard by hydrant $\underline{h_i}$, with cost $c_{(\underline{h_i}, r_j)} < 0$ representing the benefit of hearing region $r_j$.

An example of the graph can be seen in Figure 4.2. The elements of the network given by the graph and its weights have the following interpretation. Node $r_0$ represents the central station, which will actually be located in the vicinity of some hydrant. The choice of this hydrant may be free or restricted. This is modeled by an appropriate choice of arcs leaving $r_0$ . Level 1 represents the communication network, while Level 2 represents the auditory components of the *LORNO* system. Note that the only way to reach a node $\underline{h_i}$ from $r_0$ is by using $\overline{h_i}$. Finally, Level 3 represents the regions we are interested in monitoring. Note that this level may be further simplified by aggregating all nodes with the same set of predecessors in a single node with the sum of the benefits, so that the

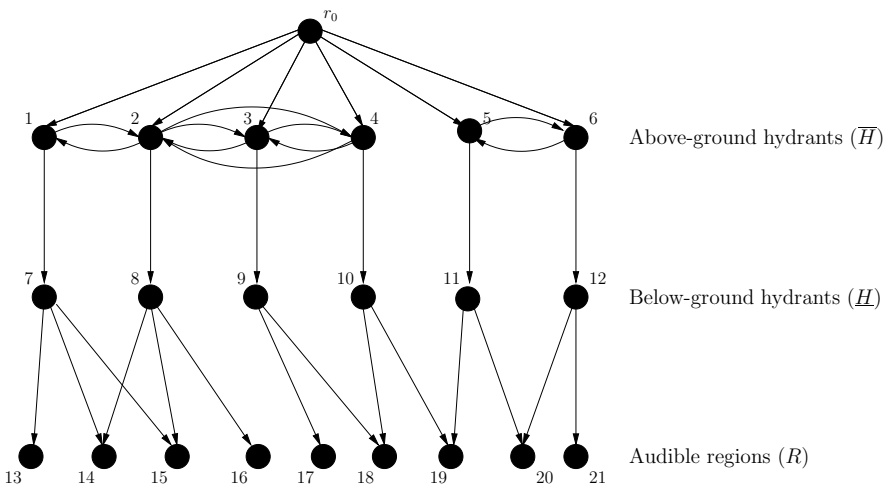



Figure 4.2: A *LORNO* network

complexity does not depend on the discretization used, but only on the network's topology. Now, one can see that each solution to the optimization problem corresponds to an arborescence in $G$, by choosing some spanning tree in the subgraph of the connection network induced by the nodes used and choosing arbitrarily which *LORNO* unit hears a region in case of multiple possibilities. Conversely, each rooted arborescence in $G$ having at most one arc leaving $r_0$ defines such a solution.

Thus the problem can be formulated as that of finding an optimal $r_0$-rooted arborescence $T = (V_T, E_T)$ with the property of having exactly one arc leaving $r_0$ [1] and, as an arborescence has exactly one arc entering each of its nodes, we report the cost of each node (positive or negative) on each of its entering arcs, thereby getting a standard PCSA problem.

### 4.1.3 Solving Prize-Collecting Steiner Arborescence Problems

**Definitions and formulation.** The prize-collecting Steiner problem was originally defined on an undirected graph, with non-negative benefits associated with its nodes and non-negative costs with its edges, as the problem of finding an optimal connected subgraph; there will then be a tree among the optimal solutions. The rooted version ensures a given node $r_0$ will be part of the solution. This definition extends in a straightforward manner to a rooted directed graph, in which we are looking for an optimal Steiner arborescence, with the property that all costs and benefits can then be transferred with appropriate signs on the corresponding incoming arcs.

[1] For simplicity, we discard the theoretically possible solution of installing nothing.

The well-known *cut formulation* of the PCSA problem is introduced in Fischetti (1991):

$$\min \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{4.1a}$$

$$\text{subject to} \quad \sum_{(j,i)\in E} x_{ji} = y_i, \quad \forall i \in V - \{r_0\} \tag{4.1b}$$

$$x\left(\delta^-(S)\right) \geq y_k, \quad \forall k \in S, \forall S \subset V - \{r_0\} \tag{4.1c}$$

$$\sum_{(r_0,i)\in E} x_{r_0 i} = 1 \tag{4.1d}$$

$$x_{ij}, y_i \in \{0,1\}, \quad \forall i \in V - \{r_0\}, \forall (i,j) \in E \tag{4.1e}$$

where $c_{ij}$ is the cost of including edge $(i,j)$ in the solution, $x(A) = \sum_{e\in A} x_e$,

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E_T \\ 0 & \text{otherwise} \end{cases}, \qquad y_i = \begin{cases} 1 & \text{if } i \in V_T \\ 0 & \text{otherwise} \end{cases},$$

and

$$\delta^-(S) = \{(i,j) \in E | i \in \overline{S}, j \in S\}.$$

Here, constraints (4.1b) enforce that each node $i$ in the solution must have exactly one incoming arc, while constraint (4.1d) implies that exactly one arc leaves $r_0$. The constraints (4.1c), which we call *connectivity constraints*, ensure that, if the solution contains node $k$, it also contains a path from the root $r_0$ to $k$ and, thus, at least one arc in each cut induced by a node set $S$ containing $k$ and not $r_0$. Ljubic et al. (2005) use this formulation as a starting point to solve PCSA problems, and we use their ideas extensively here. Not surprisingly, the difficulty inherent in this formulation is managing the connectivity constraints (4.1c). There are an exponential number of these constraints, so only those truly necessary for a given instance should be used. This structure naturally lends itself to a cut generation algorithm.

Relaxing all but some $(\overline{S}, \overline{R})$ connectivity constraints (4.1c) (i.e., selecting a subset $\mathcal{L}$ of valid pairs

$(S,k)$), yields the following formulation, denoted *current program* (CP):

$$\min \quad \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{4.2a}$$

$$\text{subject to} \quad \sum_{(j,i)\in E} x_{ji} = y_i, \quad \forall i \in V - \{r_0\} \tag{4.2b}$$

$$\sum_{(r_0,i)\in E} x_{r_0 i} = 1 \tag{4.2c}$$

$$x\left(\delta^-(S)\right) \geq y_k, \quad \forall (S,k) \in \mathcal{L} \tag{4.2d}$$

$$\sum_{(j,i)\in E} x_{ji} \leq \sum_{(i,j)\in E} x_{ij}, \quad \forall i \notin R \cup \{r_0\} \tag{4.2e}$$

$$y_i \leq 1 - x_{r_0 j}, \quad \forall i < j, \{i,j\} \subset \overline{H} \tag{4.2f}$$

$$x_{ij} + x_{ji} \leq y_i, (x_{ij} \leq y_i), \quad \forall (i,j) \in E, i \in V - \{r_0\} \tag{4.2g}$$

$$x_{ij}, y_i \in \{0,1\}, \quad \forall i \in V - \{r_0\}, \forall (i,j) \in E, \tag{4.2h}$$

Due to the symmetric structure of the communication network and to the symmetries in the cost function, many equivalent solutions exist. In order to combat this symmetry, we have added some symmetry breaking constraints (4.2f), which force a connection between the root and the node of the communication network in the solution with smallest index.

We have also added constraints (4.2e) and (4.2g) for strengthening the relaxed LP formulation. The constraints (4.2e) ensure that there are at least as many arcs leaving as there are arcs entering an internal node, which is valid for any arborescence. We also experimented with both forms of constraints 4.2g, the stronger form, $x_{ij} + x_{ji} \leq y_i$, avoiding cycles of length 2, and the weaker form $x_{ij} \leq y_i$, forcing use of both end nodes with each choice of edge.

The connectivity cuts, i.e. the pairs $(S,k)$, introduced at the root node are obtained in the following way:

(i) For $r_i \in R$, $i = 1,\ldots,|R|$ we consider the adjacent vertices $\underline{h_i} \in \underline{H}$ and form the cuts associated with the subset $S_i = \left\{r_i, \underline{h_{i1}}, \underline{h_{i2}}, \ldots, \underline{h_{ik}}\right\}$.

(ii) For $r_i \in R$, $i = 1,\ldots,|R|$ we consider the adjacent vertices $\underline{h_i} \in \underline{H}$ and their predecessors in $\overline{H}$ and form the cuts associated with the subset $S_i = \left\{r_i, \underline{h_{i1}}, \ldots, \underline{h_{ik}}, \overline{h_{i1}}, \ldots, \overline{h_{ik}}\right\}$.

(iii) If the connection network is not connected, we build for each connected component $H_i$ the set $S_i = \left\{\overline{h_j} | \overline{h_j} \in H_i \cap \overline{H}\right\}$ of nodes in that component and add a cut $(S_i,k)$ for each $k \in S_i$.

For example, in Figure 4.2, for $r = 15$, we would have $S_1 = \{15,7,8\}$ for type 1, $S_2 = \{15,7,8,1,2\}$ for type 2, and $S_3 = \{1,2,3,4\}$ for type 3. The motivation for this choice is that these are constraints whose associated dual variables may have a positive value.

### 4.1.4 Branch-and-bound algorithm

The approach used in Ljubic et al. (2005) to solve PCSA is to solve the linear relaxation of the current program, find violated connectivity constraints, introduce them in the current program, and iterate until all connectivity constraints are satisfied by the current solution. Finally, branch and bound is used if the optimal solution found so far is not integer. Finding the most violated connectivity constraint for a given terminal node $r_i$ can be done efficiently by solving a max flow problem. If the maximum $r - r_i$ flow value is less than $y_i$, the corresponding minimum cut produces such a violated constraint.

Unfortunately, due to two special properties of our instances, this approach is ineffective in solving problems of interesting size. First, our instances tend to have a very large number of terminal nodes, roughly $|V|/2$, meaning a lot of violated constraints are found, typically with the same amount of violation. We have no good criteria for choosing among them and adding all violated constraints to the current problem results in huge memory requirements. Second, the special cost structure of the communication network (i.e. all arcs have identical cost) yields very poor convergence when applying the method of Ljubic et al. (2005).

In order to overcome these difficulties we use an approach based on finding integer solutions to the current program, using standard branch-and-bound methods. If the integer solution found is an arborescence we are done. Otherwise, the special structure of our instance allows either to find an arborescence with the same value, and thus we are done, or to find effective connectivity cuts, which we add to the current problem and iterate in the same way. Though it may seem counterintuitive to solve a series of ILPs, rather than LPs, this method was shown to be very effective for our problem instances, specifically due to the following property.

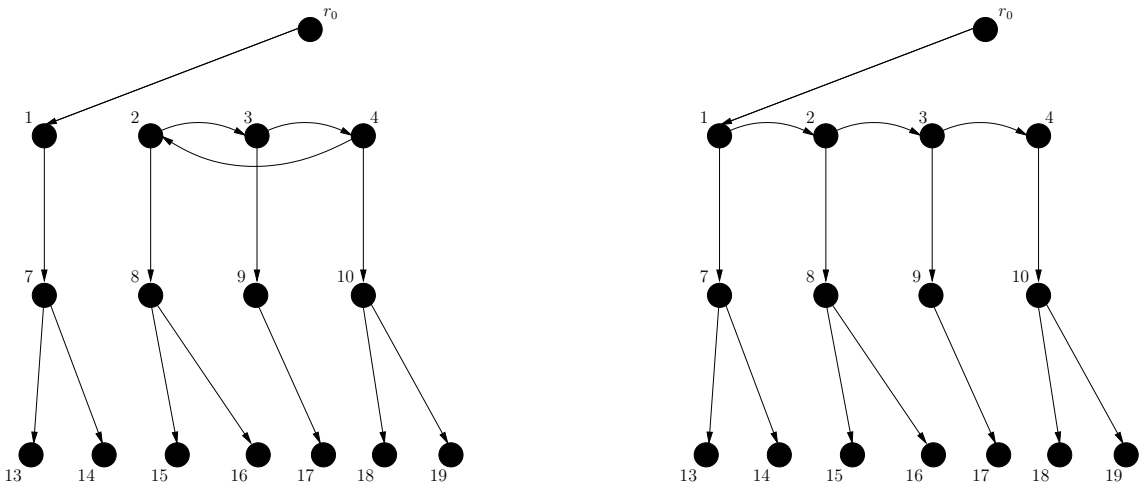
Let $G_{sol} = (V_{sol}, A_{sol})$ denote the graph associated with the solution to the current problem (4.2), $G$ the *LORNO* network, and $G(S)$ the subgraph of $G$ induced by the nodes in $S$. Recall the the special form of the networks (see Figure 4.2) we consider:

(P1) All arcs are either directed from level $i$ to level $i+1$, $i = 0, \ldots, 2$, or have both end nodes in $\overline{H}$ (at level 1), thus all circuits are entirely contained in $G(\overline{H})$.

(P2) All arcs having both end nodes in $\overline{H}$ have the same cost.

We have the following result.

**Proposition 4.1 (Prodon et al. (2010))** *If $G_{sol}$ is not an arborescence but $G(V_{sol} \cap \overline{H})$ is a connected graph, then there exists an arborescence with the same value as the current solution to the current problem* (4.2).




(a) A disconnected solution to CP.

(b) A connected solution with the same cost.

Figure 4.3: Illustrating the connection process.

**Proof.** From (4.2b) and (4.2c), it follows

$$|V_{sol}| = \sum_{i \in V} y_i = 1 + \sum_{i \in V - r} y_i = 1 + \sum_{i \in V - r} \sum_{(j,i) \in E} x_{ji} = 1 + |A_{sol}|.$$

Thus, if $G_{sol}$ is not an arborescence, then it is disconnected. From (4.2b) and (4.2g), we have that if the solution contains a node $i \neq r_0$ , it also contains exactly one arc $(j, i)$ and also node $j$. Thus it contains a path going (backward) from $i$ either to $r_0$ or to a node $k$ contained in a circuit which is, from the precedent property, entirely contained in $G(\overline{H})$. We also have

$$\left|V_{sol} \cap \overline{H}\right| = \sum_{i \in \overline{H}} \sum_{(j,i) \in E} x_{ji} = 1 + \sum_{i \in \overline{H}} \sum_{j \in \overline{H}} x_{ji}.$$

That is, the solution has in $G(\overline{H})$ a number of arcs equal to its number of nodes minus one. If $G(V_{sol} \cap \overline{H})$ is connected, it contains a spanning tree which has the same number of arcs as the solution in $G(\overline{H})$. Replacing the arcs of the solution in $G(\overline{H})$ by such a spanning tree (properly oriented) gives an arborescence with the same value as $G_{sol}$. □

This process is illustrated in Figure 4.3. On the left, a solution of the current problem associated with the network in Figure 4.2 is shown. This solution is not connected, but can be transformed in a connected solution with the same cost, as shown on the right. This result leads to Algorithm 4.1.

It is important to note that one of the reasons our instances are difficult to solve is the fact that all arcs in the communication network have the same weight. This leads to solutions to the LP

**Algorithm 4.1** LORNO Branch and Cut

1: Solve the current problem (4.2) (using branch and bound).
2: Find the connected components $C_1, \ldots, C_l$ of $G_{sol}$. If $G_{sol}$ is connected, STOP, else go to Step 3.
3: Find the connected components of $G(V_{sol} \cap \overline{H})$. If $G(V_{sol} \cap \overline{H})$ is connected, go to Step 4, else go to Step 5.
4: Find a spanning tree $T$ of $G(V_{sol} \cap \overline{H})$, replace the arcs of $A_{sol}$ contained in $G(\overline{H})$ by those of $T$ properly oriented, prune if necessary the leaves which are not in $R$ and terminate with this solution.
5: Insert the cut

$$x(\delta^-(C_i)) \geq y_k, \quad k \in C_i$$

into (4.2), for all $i = 1, \ldots, l$, and return to Step 1.

relaxation that are not connected graphs. The approach described here aims at overcoming this difficulty by searching for trees with the same weight as the current solution. An additional benefit of this approach is that it allows us to easily find approximation solutions. That is, feasible solutions whose value is guaranteed to be at most $\alpha$ times the optimal one. In fact, it suffices to stop the branch-and-bound procedure as soon as the gap first hits $\alpha$. This becomes particularly important when finding an optimal solution is no longer possible within reasonable computing time. We investigate the potential loss of solution quality resulting from applying a stopping criteria as part of our presentation of results in the following section.

### 4.1.5 Computational experiments

With our algorithms, we were able to successfully process three real-world instances, stemming from the water distribution network of the city of Lausanne and surrounding region, the largest instance comprising 606 hydrants. While determining a solution for the problems facing the city of Lausanne was the primary goal of the work, we also tested our approach on a set of random instances. Below we give results from fifty such instances.

**Problem generation.** As described above, the PCSA instances dealt with in the present study have a special structure. We were therefore led to develop a procedure enabling us to generate random problem instances having the required characteristics. We proceed as follows. First, we generate a planar representation of a planar graph representing the pipe network. Then, we choose a number of hydrants and their locations, and compute the pipe portions that could be monitored by each hydrants, if *LORNO* equipped. Then, we generate a communications network and, build the corresponding *LORNO* network (i.e. where our Steiner arborescence lives). In order to do this, we first generate $n$ uniformly-distributed points in a square of side length proportional to $\sqrt{n}$. Then, the Delaunay triangulation of this set of points (a sparse planar connected graph, containing the

optimal spanning tree and easy to compute, see e.g. de Berg et al., 2000) is computed, yielding a graph $G_D$ on $n$ nodes, in which a minimum euclidean length spanning tree $S_D$ is determined. For each edge $e$ in $E(G_D)$, a probability $p_e$ proportional to its length is assigned. Then, edges $e \in E(G_D) - E(S_D)$ are randomly eliminated with probability $p_e$, until a desired average node degree $d_a$ is achieved. This process yields a planar graph $G'_D$ with $n$ nodes and average degree $d_a$. We then add $|H|$ hydrants to randomly-chosen edges of this graph. The position of each hydrant on the edge is also determined randomly, at a location close to the center of the edge. For each hydrant, a node is added to the graph and the associated edge is split, creating two new edges and one new node. Then, the communication network is generated by adding an edge between hydrants $h_1, h_2 \in H$ with probability

$$p_{h_1,h_2} = \begin{cases} 0 & d(h_1,h_2) > r_2 \\ p_2 & r_1 \leq d(h_1,h_2) \leq r_2 \\ p_1 & r_1 > d(h_1,h_2). \end{cases}$$

Finally, we determine for each edge, the set of hydrants from which it can be heard. An edge $e = (e_1, e_2)$ is audible by hydrant $h$ if $d(h, e_1) < r_L$ and $d(h, e_2) < r_L$. Results are reported below for instances generated with $|H| = n/2$, and

$$d_a = 2.3, \quad r_1 = 250, \quad r_2 = 400, \quad p_1 = 0.8, \quad \text{and} \quad p_2 = 0.5.$$

The name of each instance in the following tables follows the naming convention established by the main data: `r1-10050` means that it is a random instance, the seed of the random generator was 1, 100 points have been generated in the plane and 50 hydrants have been placed in the resulting graph.

**Algorithms.** The branch-and-cut algorithm, denoted CONN, was implemented in C/C++, using the libraries available from the Computational Infrastructure for Operations Research (COIN-OR) repository (Lougee-Heimer, 2003). The Open Solver Interface (OSI) was used to interface with the integer and linear programming solvers. All results reported here reflect the use of OSI CPLEX interface, where CPLEX 9.1 was used to solve the integer and linear programming instances generated throughout the course of the algorithm. The algorithm was tested on an Intel Xeon 2.4GHz processor with 4GB of memory.

After carefully examining our preliminary results, we realized that a significant amount of running time was being spent on proving the optimality of solutions to CP that would eventually be discarded because of constraint violation.

As mentioned previously, the design of our algorithm allows the ability to change the optimality

requirements of CP solutions without altering the overall structure of the algorithm. Aside from the ability to get good solutions quickly, this also allows us to change our optimality gap dynamically within the algorithm. The cut generation routine requires only a feasible solution to produce valid cuts. Thus, it is possible to widen the optimality gap in the early iterations of the algorithm and generate several rounds of cuts in a much smaller amount of CPU time. In order to test the benefit of this procedure, we experimented with a slight variant of the branch-and-cut algorithm denoted GAPCONN, where we systematically modify the optimality requirement during the course of the algorithm. We refer to algorithms of this type as *dynamic*, while the standard algorithms can be described as *static*. As already described, the objective function comprises an easily quantifiable component, namely the investment costs, and one which is less so, corresponding to the drawn benefits. These are quantized by the cost of a full *LORNO* unit and that of a transponder unit, respectively. Rather than using a relative gap measure for stopping criteria, we selected to use an absolute gap. We chose to bound the gap successively by the cost of a full *LORNO* unit, that of a transponder unit and by $\epsilon$, a sufficiently small parameter to prove optimality. Note that for our numerical examples these values make good sense, since they add up to at most a fraction of a percent of the objective function value.

To evaluate the effectiveness of our separation routine, we implemented an alternate algorithm similar to that described in Ljubic et al. (2005). In this algorithm, denoted FLOW, a modification of Goldberg's maximum flow algorithm (Cherkassky, 1997) is used find violated constraints with respect to solutions of the LP-relaxation of CP. In the case that such a solution does not violate any constraints, an integer programming solver is called to find an integer solution to CP. If the resulting integer solution is also feasible to the original problem, it must be optimal. Else, new violated inequalities are added to CP, and the algorithm continues. This algorithm was also implemented in C/C++, using COIN-OR's libraries to interface with CPLEX 9.1.

As mentioned previously, in addition to the comparison of separation routines, we also experimented with the form of the constraints (4.2g). The algorithms included in our experiments are summarized in Table 4.1.

| | CONN | FLOW | GAPCONN |
|---|---|---|---|
| $x_{ij} + x_{ji} \leq y_i$ | CONN2 | FLOW2 | GAPCONN2 |
| $x_{ij} \leq y_i$ | CONN4 | — | GAPCONN4 |

Table 4.1: The algorithm variants used in the computational study.

### 4.1.6 Results

We present here results from two classes of experiments. The first consists of a series of tests where we compare the algorithmic variants on data instances of increasing size. In each table, the set of columns labeled *Iterations* gives the number of number of algorithmic loops necessary to find the optimal solution. The group of columns labeled *CPU sec* gives the running time of the algorithms on the platform described above. Unless otherwise noted, all results presented represent a CPLEX optimality gap of $1 \times 10^{-4}$. For the algorithms in which the gap is changed dynamically, this corresponds to a choice of $\epsilon = 1 \times 10^{-4}$.

For this study, a maximum running time of 5000 CPU seconds was allotted. In the tables, instances that were not solved within this time limit are indicated with a dash in the corresponding row of the table. Note that the set of unsolved instances includes both those instances that exceeded the time limit, as well as those whose memory requirements were too large for the resources available. We do not differentiate between these two types of unsolved instances in the presentation of our results. However, we do note that the FLOW algorithm frequently fails due to memory requirements. This suggests that if this algorithm is used, unnecessary cuts should be removed dynamically throughout the course of the algorithm.

The complete output for the experimental study is shown in Table 4.3. From the table, we can see that the dynamic variants of the branch-and-cut algorithm (GAPCONN2 and GAPCONN4) clearly dominate their static counterparts. Further, there is no problem that GAPCONN2 is able to solve that GAPCONN4 cannot. Thus, we can say that GAPCONN4 is the most robust, with respect to number of problems solved, of all the branch-and-cut variants. Additionally, in Section 4.1.6, we compare the performance of GAPCONN2 and GAPCONN4 across different optimality requirements, and see that when the optimality gap is equal to the cost of a full *LORNO* installation, GAPCONN4 is able to solve all but two problems in our test set.

It is not immediately obvious, however, how the branch-and-cut variants compare to FLOW2, since there are problems that FLOW2 is able to solve where all branch-and-cut algorithms fail. However, a comparison between FLOW2 and GAPCONN4 shows that this occurs only twice in the entire test set. Additionally, the total number of problems solved by each of the branch-and-cut algorithms is significantly higher than that by FLOW2.

The results are summarized in Table 4.2. In this table, we report the average number of instances solved, the average required iterations and average CPU time required for each algorithm. Table 4.2 gives further evidence that GAPCONN4 is the most robust of all algorithm variants, solving almost 90% of the problems in the test set, but also shows that it requires the second highest average CPU time. FLOW2 is the fastest algorithm, on average, but solves only 54% of the problems. CONN4 seems to yield the best balance between speed and robustness, solving 80% of the test problems,

| | Success Ratio | Avg No. Iter. | Avg CPU sec |
|---|---|---|---|
| CONN2 | 0.78 | 7.85 | 249.15 |
| CONN4 | 0.80 | 7.38 | 118.49 |
| FLOW2 | 0.54 | 66.30 | 154.52 |
| GAPCONN2 | 0.82 | 15.54 | 331.41 |
| GAPCONN4 | 0.88 | 20.43 | 318.15 |

Table 4.2: Summary results for all algorithms.

with the second lowest average CPU time. This table also suggests that, for the problems we study here, the weaker form of constraint (4.2g) is preferable to the stronger form since, for both the static and dynamic variants, using this form yielded a higher success rate and a lower average speed. It should be noted, however, that the results in Table 4.2 may be somewhat misleading, since the averages do not account for those instances that remain unsolved by each algorithm. The performance profiles shown in Figure 4.4 provide a more equitable comparison.

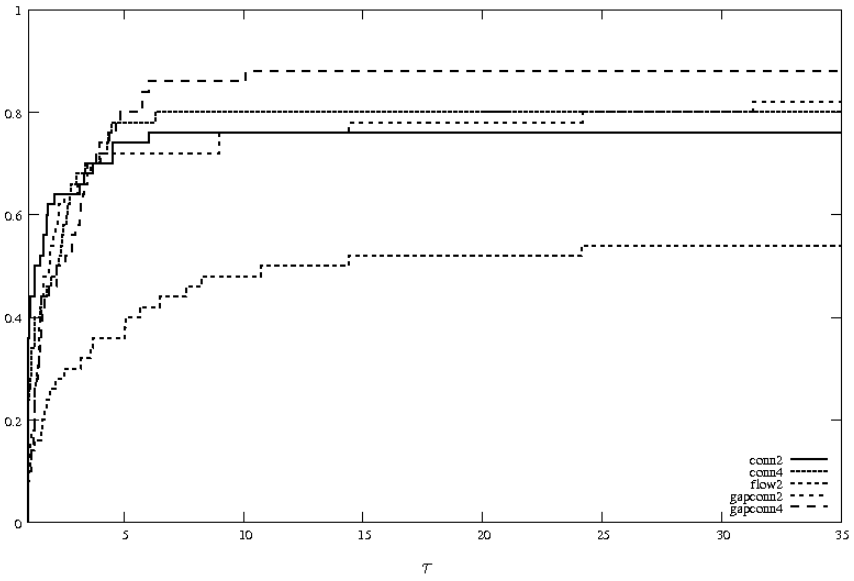



Figure 4.4: Performance profiles of each of the solvers described.

From the figure, we can see that CONN2 achieved the minimum solution time on the largest number of problems (roughly 40%). Thus, in a loose sense, we can say this algorithm is the fastest. Finding points of intersection within the plot allows us to determine those values of $\tau$ for which a subset of the algorithms is equivalent with respect to running time. From the plot, we can see that CONN2, CONN4, GAPCONN2, GAPCONN4 will all solve a given problem within a factor of 4 of the fastest algorithm roughly 70% of the time.

In fact, for a range of values of $\tau$ between 3.5 and 4, the algorithms CONN2 and CONN4 are

| | Iterations | | | | | CPU sec | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Instance | CONN2 | CONN4 | FLOW2 | GCONN2 | GCONN4 | CONN2 | CONN4 | FLOW2 | GCONN2 | GCONN4 |
| r1-10050 | 1 | 3 | 1 | 2 | 3 | 0.780 | 0.432 | 0.540 | 1.320 | 0.492 |
| r2-10050 | 1 | 2 | 4 | 1 | 2 | 0.428 | 1.044 | 0.712 | 0.468 | 1.352 |
| r3-10050 | 6 | 8 | 47 | 6 | 7 | 8.005 | 16.313 | 3.804 | 8.165 | 16.585 |
| r4-10050 | 4 | 5 | — | 41 | 6 | 2.356 | 2.192 | — | 31.550 | 3.216 |
| r5-10050 | 2 | 1 | 28 | 2 | 1 | 1.160 | 0.720 | 7.708 | 1.360 | 0.988 |
| r6-10050 | 1 | 1 | 19 | 1 | 1 | 0.340 | 0.324 | 1.832 | 0.428 | 0.408 |
| r7-10050 | 20 | 13 | 168 | 11 | 15 | 38.994 | 15.877 | 19.881 | 10.553 | 16.281 |
| r8-10050 | 2 | 1 | 3 | 2 | 1 | 1.000 | 0.300 | 0.524 | 1.080 | 0.360 |
| r9-10050 | — | — | 146 | — | — | — | — | 82.513 | — | — |
| r10-10050 | 4 | 4 | 4 | 5 | 3 | 3.748 | 2.692 | 0.620 | 5.568 | 3.560 |
| r1-200100 | — | — | — | 51 | 22 | — | — | — | 991.814 | 110.323 |
| r2-200100 | 1 | 5 | 53 | 1 | 5 | 8.028 | 10.229 | 66.240 | 11.601 | 9.845 |
| r3-200100 | — | — | — | — | — | — | — | — | — | — |
| r4-200100 | — | — | — | — | 110 | — | — | — | — | 1128.250 |
| r5-200100 | 1 | 1 | 29 | 1 | 1 | 3.852 | 4.300 | 14.301 | 4.748 | 4.752 |
| r6-200100 | 2 | 3 | 7 | 2 | 3 | 5.616 | 10.097 | 4.412 | 6.712 | 11.145 |
| r7-200100 | 3 | 1 | 73 | 3 | 1 | 9.253 | 5.228 | 75.293 | 11.697 | 5.728 |
| r8-200100 | 3 | 2 | 103 | 3 | 2 | 6.756 | 5.368 | 129.632 | 8.129 | 7.176 |
| r9-200100 | 3 | 3 | 9 | 4 | 4 | 5.400 | 11.109 | 4.240 | 7.828 | 16.285 |
| r10-200100 | 1 | 4 | — | 1 | 4 | 5.784 | 19.433 | — | 8.065 | 19.973 |
| r1-300150 | 3 | 9 | — | 3 | 9 | 13.197 | 52.631 | — | 19.201 | 61.420 |
| r2-300150 | 1 | 6 | 48 | 1 | 6 | 17.081 | 51.199 | 54.579 | 27.918 | 57.000 |
| r3-300150 | 7 | 7 | 33 | 18 | 31 | 43.075 | 40.911 | 26.370 | 106.323 | 152.254 |
| r4-300150 | 2 | 4 | — | 2 | 6 | 38.286 | 106.547 | — | 47.807 | 107.479 |
| r5-300150 | 54 | 81 | — | 76 | 362 | 381.604 | 903.728 | — | 560.043 | 3850.540 |
| r6-300150 | 2 | 5 | 73 | 4 | 7 | 25.490 | 44.879 | 128.984 | 41.183 | 55.572 |
| r7-300150 | — | — | — | 21 | 23 | — | — | — | 565.779 | 846.493 |
| r8-300150 | 3 | 10 | — | 3 | 7 | 25.774 | 162.306 | — | 34.490 | 76.277 |
| r9-300150 | — | — | 70 | — | — | — | — | 56.452 | — | — |
| r10-300150 | 2 | 9 | — | 5 | 8 | 22.445 | 100.690 | — | 45.923 | 73.241 |
| r1-400200 | 3 | 1 | 31 | 6 | 1 | 47.439 | 27.070 | 58.040 | 67.664 | 33.910 |
| r2-400200 | 1 | 6 | — | 2 | 6 | 46.519 | 59.192 | — | 51.271 | 87.137 |
| r3-400200 | 1 | 2 | 35 | 1 | 4 | 32.934 | 63.120 | 80.733 | 32.546 | 91.018 |
| r4-400200 | — | — | — | — | — | — | — | — | — | — |
| r5-400200 | 79 | 2 | — | — | 2 | 2586.150 | 29.058 | — | — | 45.847 |
| r6-400200 | 8 | 22 | 186 | 20 | 37 | 106.175 | 229.622 | 535.125 | 238.467 | 516.076 |
| r7-400200 | 5 | 7 | — | 6 | 9 | 79.577 | 215.217 | — | 99.922 | 265.701 |
| r8-400200 | 9 | 6 | 89 | 11 | 15 | 98.942 | 82.037 | 238.915 | 66.496 | 263.248 |
| r9-400200 | — | 10 | — | 116 | 9 | — | 94.446 | — | 2284.760 | 134.308 |
| r10-400200 | — | — | — | — | — | — | — | — | — | — |
| r1-500250 | 25 | 12 | — | 29 | 17 | 2727.330 | 605.638 | — | 2084.730 | 610.334 |
| r2-500250 | 2 | 6 | — | 3 | 10 | 97.542 | 251.568 | — | 139.333 | 310.511 |
| r3-500250 | 15 | 10 | 186 | 44 | 23 | 2181.300 | 772.440 | 1099.330 | 1304.160 | 692.931 |
| r4-500250 | 15 | 5 | 64 | 113 | 5 | 605.138 | 133.924 | 202.733 | 4192.560 | 222.622 |
| r5-500250 | 3 | 2 | 128 | 3 | 9 | 73.285 | 80.161 | 474.438 | 90.942 | 160.966 |
| r6-500250 | — | — | — | — | — | — | — | — | — | — |
| r7-500250 | 1 | 2 | — | 2 | 3 | 69.864 | 77.853 | — | 75.961 | 86.357 |
| r8-500250 | — | — | — | — | 70 | — | — | — | — | 2909.050 |
| r9-500250 | 6 | 5 | — | 7 | 7 | 190.108 | 193.848 | — | 186.352 | 293.374 |
| r10-500250 | 4 | 9 | 153 | 4 | 22 | 106.075 | 255.872 | 803.998 | 113.095 | 638.072 |

Table 4.3: Results from all variants on the full test set.

almost indistinguishable with respect to running time. This is not so surprising, since these two solvers differ only in the form of the constraints (4.2g). From the larger values of $\tau$, we can see the probability that the solvers solve a problem within our test set. GAPCONN4 is the most likely to solve a random instance, solving approximately 90% of the problems tested. GAPCONN2 and CONN4 both solve roughly 80% of the instances, and CONN2 is successful 75% of the time. From the plots, we can also see that the probability of success does not significantly increase for $\tau > 5$ for any of our solvers, except GAPCONN4. Thus, if GAPCONN2, CONN2 or CONN4 is able to solve an instance, it is likely that it will solve the instance within five times the speed of the fastest solver. Figure 4.4 confirms our earlier assertion that the flow algorithm is dominated by all variants of the branch-and-cut algorithm. FLOW2 is the fastest solver only 15% of the time and solves only half of the instances in the test set.

In the formulation described previously, we have assumed that we are free to install as many full *LORNO* installations as desired. However, this may not be a realistic assumption, since this number may be limited by physical or financial constraints. In order to test the sensitivity of our algorithm to this assumption, a second computational test was performed. We add the constraint

$$\sum_{i \in \underline{H}} y_i \leq B \tag{4.3}$$

to the formulation and apply the solution algorithms for varying bounds $B$ on the number of installed auditory components. Due the increased difficulty of these restricted problems, we chose to relax our optimality requirements. As mentioned previously, the design of our algorithm allows the user to change the desired optimality gap without alteration of the algorithm. For this experiment, we used an optimality gap equal to the cost of one *LORNO* installation. These results are also presented in tabular format, as before. Here, we have two additional columns, labeled $B$ and *Obj*, which indicate the limit placed on the number of full *LORNO* installations and the resulting optimal objective value, respectively. The data used for the experiment consisted of both a real water network from Lausanne, as well as a randomly-generated instance with similar characteristics. The full results are shown in Table 4.4. Aside from testing the sensitivity of our algorithm, this study allows us to examine the inherent tradeoff that exists between the installation limit and the resulting benefit. Figure 4.5 illustrates this relationship for both data instances. The portions of the tradeoff curves in Figure 4.5 in which we are most interested are those with a steep slope. These areas represent critical points, where small increases in $B$ yield substantial increases in the optimal benefit. Assuming the budget constraint (4.3) is somewhat flexible, these critical points indicate where it is worthwhile to increase the installation limit, assuming an increase is physically possible. In Section 4.2.3, we perform a similar analysis on a set of interdiction problems, via multicriteria programming. Analyses of this type are particularly relevant to applications facing government DMs. Due to

| dail | | | | | | | r1-300150 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GAPCONN2 | | | GAPCONN4 | | | | GAPCONN2 | | | GAPCONN4 | | |
| $B$ | Iter | CPU sec | Obj | Iter | CPU sec | Obj | $B$ | Iter | CPU sec | Obj | Iter | CPU sec | Obj |
| 45 | — | — | — | — | — | — | 55 | — | — | — | — | — | — |
| 47 | 15 | 5166.830 | -2083350 | 23 | 2862.860 | -2083350 | 57 | 47 | 999.510 | -3064700 | 64 | 3058.000 | -3064700 |
| 49 | 12 | 628.147 | -2117540 | 8 | 213.957 | -2117540 | 59 | 18 | 342.893 | -3099980 | 9 | 383.148 | -3100240 |
| 51 | 4 | 174.463 | -2149400 | 2 | 81.645 | -2150150 | 61 | 13 | 176.339 | -3131050 | 21 | 978.061 | -3131320 |
| 53 | 1 | 65.984 | -2180740 | 1 | 66.388 | -2180740 | 63 | 19 | 166.990 | -3157350 | 34 | 715.033 | -3157610 |
| 55 | 1 | 46.515 | -2206060 | 1 | 70.976 | -2206060 | 65 | 24 | 202.717 | -3181040 | 7 | 117.647 | -3181040 |
| 57 | 1 | 47.531 | -2224240 | 1 | 73.185 | -2225680 | 67 | 8 | 33.002 | -3199500 | 7 | 108.987 | -3200220 |
| 59 | 1 | 42.639 | -2242660 | 1 | 63.808 | -2243540 | 69 | 7 | 24.373 | -3216620 | 5 | 62.208 | -3216620 |
| 61 | 1 | 42.079 | -2258840 | 1 | 69.864 | -2258840 | 71 | 3 | 20.285 | -3228980 | 6 | 69.136 | -3228980 |
| 63 | 1 | 57.576 | -2271700 | 1 | 67.428 | -2271700 | 73 | 3 | 9.397 | -3240300 | 3 | 39.331 | -3240300 |
| 65 | 1 | 51.567 | -2281670 | 1 | 68.800 | -2281670 | 75 | 3 | 12.941 | -3249670 | 2 | 35.974 | -3249670 |
| 67 | 1 | 59.312 | -2286980 | 1 | 60.852 | -2286980 | 77 | 5 | 17.605 | -3256390 | 2 | 44.183 | -3256390 |
| 69 | 1 | 52.735 | -2287640 | 1 | 55.947 | -2287640 | 79 | 5 | 15.209 | -3256390 | 3 | 52.935 | -3256390 |

Table 4.4: Results from budget study.

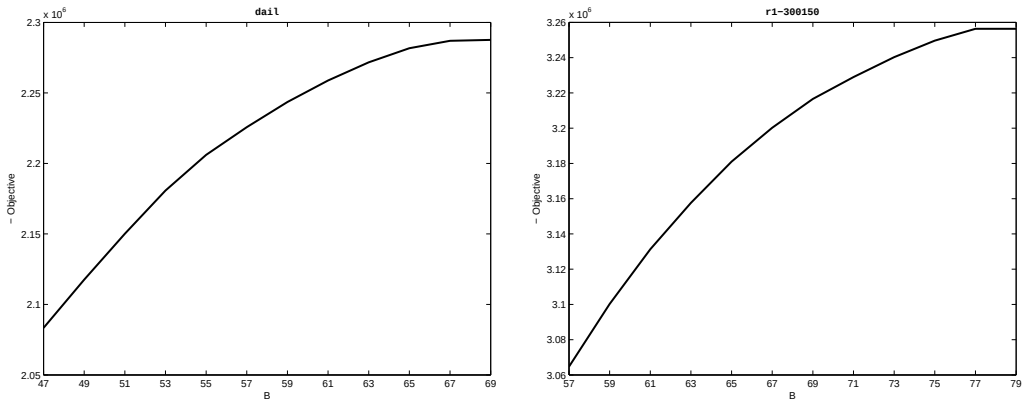



Figure 4.5: Tradeoff between the installation budget and the objective value.

limited availability of funds, it is crucial to ensure that the allocated funds are being used effectively. On the other hand, if there is significant societal benefit realized with a slight increase in the budget, the government may be able to allocate additional funding to achieve this benefit.

**The Cost of Optimality.** In this section, we compare the cost of proving optimality for GAPCONN2 and GAPCONN4. Tables 4.5 and 4.6 show the full sets of results for the two algorithms. In each table, we see the required iterations and CPU time, as well as the objective value for each value of optimality gap discussed in the previous section. We denote these values by *LORNO*, the cost of a full *LORNO* installation, *TRANS*, the cost of installing a radio transponder only, and *EPS*, a sufficiently small parameter that yields "true" optimality. For GAPCONN2, the results indicate that, on average, moving from *LORNO*-optimality to TRANS-optimality requires approximately 107% more CPU time, while improving the objective by only 0.007%. Further, to achieve EPS-optimality, GAPCONN2 requires approximately 162% more CPU time than for *LORNO*-optimality, and yields only a 0.012% objective improvement. The results for GAPCONN4 are less dramatic, but show a similar tendency. For GAPCONN4, the average difference in runtime between *LORNO*- and TRANS-optimality is approximately 37% with a 0.005% objective improvement, and obtaining EPS-optimality requires a 59% increase in CPU time and yields a 0.007% objective improvement. Thus, we can conclude that, for these instances, employing an optimality gap does not significantly decrease solution quality, but results in much faster computing times.

### 4.1.7 Sensitivity to Graph Structure

In the previous section, we conduct one type of sensitivity analysis; namely, we analyze the sensitivity of our solution quality to the number of hydrants we are able to install. In this section, we describe another method for conducting sensitivity analysis.

Suppose we seek to understand the benefit of individual hydrants in our monitoring system. Put another way, suppose we wish to determine the effect of removing a node from the communication graph on our ability to monitor the water network. If we find that certain hydrants within the network are crucial to our ability to monitor the system, protecting those crucial hydrants may be worthwhile. This idea is illustrated in examples that follow. In Example 6, we consider removal an above-ground hydrant from the network and, in Example 7 removal of a below-ground hydrant is considered.

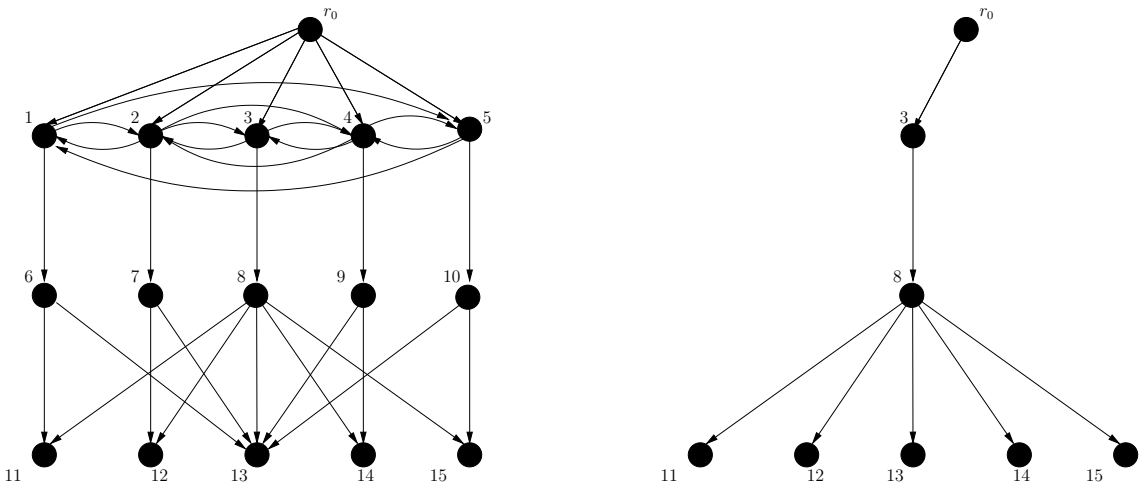
**Example 6. Removal of an above-ground hydrant.** Consider the *LORNO* network shown in Figure 4.6(a). We can find connected solution with one full *LORNO* installation, as shown in Figure 4.6(b). Let $c = c_{\overline{h}} + c_{\underline{h}}$ be the cost of a full *LORNO* installation. Recall that cost $c_{(\underline{h_i}, r_j)} < 0$

| | LORNO | | | TRANS | | | EPS | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Iterations | CPU sec | Objective | Iterations | CPU sec | Objective | Iterations | CPU sec | Objective |
| r1-10050 | 1 | 0.848 | -960341 | 1 | 0.876 | -960341 | 2 | 1.320 | -961061 |
| r2-10050 | 1 | 0.464 | -890994 | 1 | 0.468 | -890994 | 1 | 0.468 | -890994 |
| r3-10050 | 3 | 0.764 | -1062120 | 3 | 0.792 | -1062120 | 6 | 8.165 | -1062120 |
| r4-10050 | 37 | 30.074 | -1139590 | 40 | 31.478 | -1140310 | 41 | 31.550 | -1141030 |
| r5-10050 | 2 | 1.344 | -957468 | 2 | 1.356 | -957468 | 2 | 1.360 | -957468 |
| r6-10050 | 1 | 0.420 | -1113020 | 1 | 0.428 | -1113020 | 1 | 0.428 | -1113020 |
| r7-10050 | 10 | 7.152 | -813227 | 10 | 7.176 | -813227 | 11 | 10.553 | -813227 |
| r8-10050 | 2 | 1.072 | -1023130 | 2 | 1.080 | -1023130 | 2 | 1.080 | -1023130 |
| r9-10050 | 8 | 6.296 | -969108 | 8 | 6.316 | -969108 | — | — | — |
| r10-10050 | 5 | 4.772 | -987484 | 5 | 4.820 | -987484 | 5 | 5.568 | -987484 |
| r1-200100 | 7 | 27.214 | -2209600 | 51 | 967.125 | -2210320 | 51 | 991.814 | -2210320 |
| r2-200100 | 1 | 9.421 | -2080110 | 1 | 9.505 | -2080110 | 1 | 11.601 | -2080110 |
| r3-200100 | 14 | 78.121 | -1910670 | — | — | — | — | — | — |
| r4-200100 | 57 | 274.581 | -2027370 | — | — | — | — | — | — |
| r5-200100 | 1 | 4.708 | -2166220 | 1 | 4.732 | -2166220 | 1 | 4.748 | -2166220 |
| r6-200100 | 2 | 6.676 | -2017630 | 2 | 6.700 | -2017630 | 2 | 6.712 | -2017630 |
| r7-200100 | 3 | 11.665 | -2008300 | 3 | 11.689 | -2008300 | 3 | 11.697 | -2008300 |
| r8-200100 | 3 | 5.836 | -2010260 | 3 | 6.416 | -2010260 | 3 | 8.129 | -2010260 |
| r9-200100 | 3 | 3.548 | -2176660 | 4 | 3.892 | -2177380 | 4 | 7.828 | -2177380 |
| r10-200100 | 1 | 6.552 | -2086780 | 1 | 8.053 | -2086780 | 1 | 8.065 | -2086780 |
| r1-300150 | 3 | 13.077 | -3256390 | 3 | 13.233 | -3256390 | 3 | 19.201 | -3256390 |
| r2-300150 | 1 | 19.161 | -3131920 | 1 | 27.606 | -3131920 | 1 | 27.918 | -3131920 |
| r3-300150 | 17 | 105.683 | -3133680 | 17 | 105.807 | -3133680 | 18 | 106.323 | -3134400 |
| r4-300150 | 2 | 47.331 | -2725630 | 2 | 47.487 | -2725630 | 2 | 47.807 | -2725630 |
| r5-300150 | 76 | 551.646 | -3048320 | 76 | 551.802 | -3048320 | 76 | 560.043 | -3048320 |
| r6-300150 | 3 | 36.258 | -3098540 | 4 | 37.302 | -3099260 | 4 | 41.183 | -3099260 |
| r7-300150 | 21 | 458.725 | -3134060 | 21 | 499.796 | -3134060 | 21 | 565.780 | -3134060 |
| r8-300150 | 3 | 16.913 | -3114700 | 3 | 17.121 | -3114700 | 3 | 34.490 | -3114700 |
| r9-300150 | 12 | 97.430 | -3029640 | 12 | 97.586 | -3029640 | — | — | — |
| r10-300150 | 4 | 33.390 | -3161470 | 5 | 40.239 | -3162190 | 5 | 45.923 | -3162190 |
| r1-400200 | 5 | 66.032 | -4290830 | 5 | 66.236 | -4290830 | 6 | 67.664 | -4291550 |
| r2-400200 | 1 | 45.059 | -4273880 | 2 | 51.003 | -4274600 | 2 | 51.271 | -4274600 |
| r3-400200 | 1 | 32.438 | -4105300 | 1 | 32.514 | -4105300 | 1 | 32.546 | -4105300 |
| r4-400200 | — | — | — | — | — | — | — | — | — |
| r5-400200 | 68 | 658.465 | -4178710 | 187 | 2132.825 | -4179430 | — | — | — |
| r6-400200 | 15 | 184.216 | -3880780 | 15 | 184.440 | -3880780 | 20 | 238.467 | -3881500 |
| r7-400200 | 5 | 79.665 | -4269060 | 6 | 89.642 | -4269780 | 6 | 99.922 | -4269780 |
| r8-400200 | 11 | 66.416 | -4324840 | 11 | 66.468 | -4324840 | 11 | 66.496 | -4324840 |
| r9-400200 | 29 | 303.971 | -4091240 | 116 | 2278.291 | -4091960 | 116 | 2284.763 | -4091960 |
| r10-400200 | 5 | 70.220 | -3987600 | 7 | 108.671 | -3987600 | — | — | — |
| r1-500250 | 15 | 350.346 | -5196930 | 16 | 468.133 | -5196930 | 29 | 2084.723 | -5197650 |
| r2-500250 | 2 | 90.938 | -5257910 | 2 | 91.294 | -5257910 | 3 | 139.333 | -5257910 |
| r3-500250 | 31 | 685.811 | -5040440 | 44 | 1278.000 | -5041160 | 44 | 1304.158 | -5041160 |
| r4-500250 | 100 | 3346.910 | -5420330 | 113 | 4192.203 | -5421050 | 113 | 4192.555 | -5421050 |
| r5-500250 | 3 | 66.344 | -5236380 | 3 | 72.665 | -5236380 | 3 | 90.942 | -5236380 |
| r6-500250 | — | — | — | — | — | — | — | — | — |
| r7-500250 | 1 | 64.088 | -5088330 | 2 | 75.521 | -5089050 | 2 | 75.961 | -5089050 |
| r8-500250 | 18 | 399.773 | -5030270 | — | — | — | — | — | — |
| r9-500250 | 3 | 101.246 | -5154660 | 7 | 185.963 | -5155070 | 7 | 186.351 | -5155070 |
| r10-500250 | 4 | 110.307 | -5401620 | 4 | 112.843 | -5401620 | 4 | 113.095 | -5401620 |

Table 4.5: Comparing the cost of optimality for GAPCONN2.

| | LORNO | | | TRANS | | | EPS | | |
|---|---|---|---|---|---|---|---|---|---|
| Instance | Iterations | CPU sec | Objective | Iterations | CPU sec | Objective | Iterations | CPU sec | Objective |
| r1-10050 | 3 | 0.488 | -961061 | 3 | 0.492 | -961061 | 3 | 0.492 | -961061 |
| r2-10050 | 2 | 1.348 | -890994 | 2 | 1.352 | -890994 | 2 | 1.352 | -890994 |
| r3-10050 | 4 | 2.056 | -1062120 | 4 | 2.080 | -1062120 | 7 | 16.585 | -1062120 |
| r4-10050 | 2 | 0.668 | -1140310 | 2 | 0.696 | -1140310 | 6 | 3.216 | -1141030 |
| r5-10050 | 1 | 0.972 | -957468 | 1 | 0.980 | -957468 | 1 | 0.988 | -957468 |
| r6-10050 | 1 | 0.400 | -1113020 | 1 | 0.404 | -1113020 | 1 | 0.408 | -1113020 |
| r7-10050 | 14 | 11.037 | -813227 | 14 | 11.061 | -813227 | 15 | 16.281 | -813227 |
| r8-10050 | 1 | 0.356 | -1023130 | 1 | 0.356 | -1023130 | 1 | 0.360 | -1023130 |
| r9-10050 | 15 | 10.957 | -968388 | 25 | 16.057 | -969108 | — | — | — |
| r10-10050 | 3 | 2.348 | -987484 | 3 | 2.372 | -987484 | 3 | 3.560 | -987484 |
| r1-200100 | 22 | 86.185 | -2210320 | 22 | 89.534 | -2210320 | 22 | 110.323 | -2210320 |
| r2-200100 | 5 | 7.020 | -2080110 | 5 | 7.140 | -2080110 | 5 | 9.845 | -2080110 |
| r3-200100 | 34 | 253.704 | -1910670 | — | — | — | — | — | — |
| r4-200100 | 110 | 1121.570 | -2028090 | 110 | 1121.666 | -2028090 | 110 | 1128.246 | -2028090 |
| r5-200100 | 1 | 4.732 | -2166220 | 1 | 4.744 | -2166220 | 1 | 4.752 | -2166220 |
| r6-200100 | 3 | 11.125 | -2017630 | 3 | 11.137 | -2017630 | 3 | 11.145 | -2017630 |
| r7-200100 | 1 | 5.700 | -2008300 | 1 | 5.720 | -2008300 | 1 | 5.728 | -2008300 |
| r8-200100 | 2 | 4.764 | -2010260 | 2 | 5.216 | -2010260 | 2 | 7.176 | -2010260 |
| r9-200100 | 4 | 12.429 | -2177380 | 4 | 12.541 | -2177380 | 4 | 16.285 | -2177380 |
| r10-200100 | 4 | 18.497 | -2086790 | 4 | 19.965 | -2086790 | 4 | 19.973 | -2086790 |
| r1-300150 | 9 | 52.667 | -3256390 | 9 | 52.867 | -3256390 | 9 | 61.420 | -3256390 |
| r2-300150 | 6 | 50.579 | -3131920 | 6 | 56.616 | -3131920 | 6 | 57.000 | -3131920 |
| r3-300150 | 19 | 83.625 | -3133680 | 19 | 83.833 | -3133680 | 31 | 152.254 | -3134400 |
| r4-300150 | 3 | 44.267 | -2724910 | 6 | 107.263 | -2725630 | 6 | 107.479 | -2725630 |
| r5-300150 | 357 | 3757.180 | -3047600 | 361 | 3839.161 | -3047600 | 362 | 3850.542 | -3048320 |
| r6-300150 | 5 | 43.099 | -3098540 | 7 | 51.791 | -3099260 | 7 | 55.571 | -3099260 |
| r7-300150 | 23 | 357.806 | -3134060 | 23 | 667.617 | -3134060 | 23 | 846.492 | -3134060 |
| r8-300150 | 7 | 55.731 | -3114700 | 7 | 55.944 | -3114700 | 7 | 76.277 | -3114700 |
| r9-300150 | 6 | 35.250 | -3028920 | 40 | 347.438 | -3029640 | — | — | — |
| r10-300150 | 8 | 64.168 | -3162190 | 8 | 64.380 | -3162190 | 8 | 73.241 | -3162190 |
| r1-400200 | 1 | 33.786 | -4291550 | 1 | 33.870 | -4291550 | 1 | 33.910 | -4291550 |
| r2-400200 | 6 | 86.541 | -4274600 | 6 | 86.857 | -4274600 | 6 | 87.137 | -4274600 |
| r3-400200 | 2 | 67.876 | -4104580 | 3 | 90.242 | -4104580 | 4 | 91.018 | -4105300 |
| r4-400200 | — | — | — | — | — | — | — | — | — |
| r5-400200 | 2 | 34.474 | -4180150 | 2 | 34.730 | -4180150 | 2 | 45.847 | -4180150 |
| r6-400200 | 37 | 515.456 | -3881500 | 37 | 515.764 | -3881500 | 37 | 516.076 | -3881500 |
| r7-400200 | 7 | 212.777 | -4269060 | 9 | 244.911 | -4269780 | 9 | 265.700 | -4269780 |
| r8-400200 | 4 | 84.905 | -4323400 | 11 | 153.734 | -4324120 | 15 | 263.249 | -4324840 |
| r9-400200 | 9 | 121.748 | -4091960 | 9 | 122.028 | -4091960 | 9 | 134.309 | -4091960 |
| r10-400200 | 3 | 56.243 | -3987600 | 5 | 101.106 | -3987600 | — | — | — |
| r1-500250 | 17 | 580.528 | -5197650 | 17 | 609.634 | -5197650 | 17 | 610.334 | -5197650 |
| r2-500250 | 5 | 173.587 | -5257190 | 10 | 279.890 | -5257910 | 10 | 310.512 | -5257910 |
| r3-500250 | 22 | 482.546 | -5041160 | 22 | 482.962 | -5041160 | 23 | 692.931 | -5041160 |
| r4-500250 | 5 | 197.368 | -5421050 | 5 | 222.194 | -5421050 | 5 | 222.622 | -5421050 |
| r5-500250 | 9 | 138.053 | -5236380 | 9 | 138.509 | -5236380 | 9 | 160.966 | -5236380 |
| r6-500250 | — | — | — | — | — | — | — | — | — |
| r7-500250 | 2 | 63.268 | -5088330 | 3 | 83.061 | -5089050 | 3 | 86.357 | -5089050 |
| r8-500250 | 68 | 2442.070 | -5030270 | 70 | 2505.382 | -5030990 | 70 | 2909.051 | -5030990 |
| r9-500250 | 2 | 146.837 | -5154660 | 7 | 277.413 | -5155070 | 7 | 293.374 | -5155070 |
| r10-500250 | 19 | 527.873 | -5400900 | 20 | 573.680 | -5400900 | 22 | 638.072 | -5401620 |

Table 4.6: Comparing the cost of optimality for GAPCONN4.



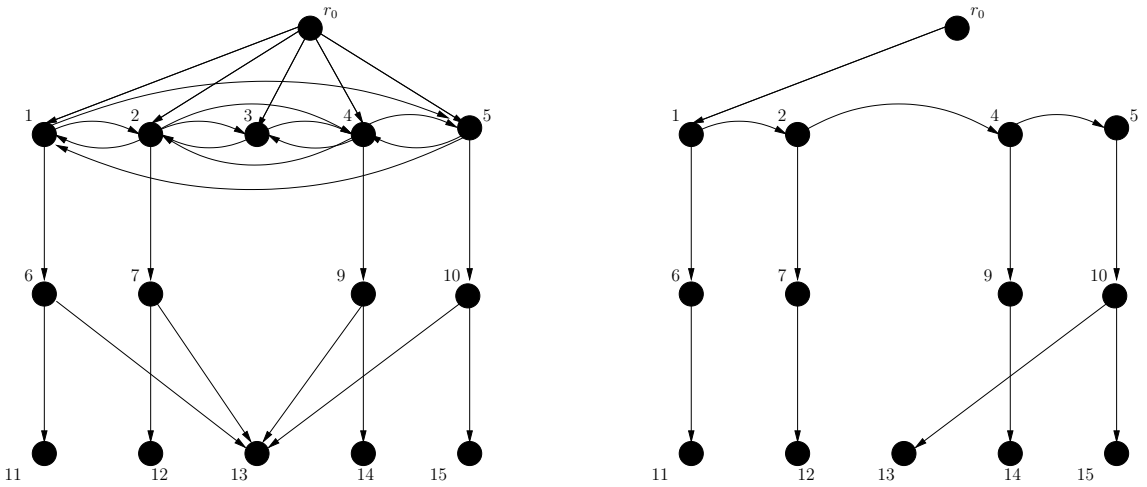

(a) The *LORNO* network. (b) A connected solution with one full installation.

Figure 4.6: The *LORNO* network and corresponding solution from Example 6.

represents the benefit of hearing region $j$ and, is thus, the same for all $\underline{h_i}$ such that $(\underline{h_i}, r_j) \in E$. The solution shown in Figure 4.6(b) has a total cost of

$$c - (c_{8,11} + c_{8,12} + c_{8,13} + c_{8,14} + c_{8,15})\,.$$

However, if we remove the underground hydrant represented by vertex 8, we have the resulting network shown in Figure 4.7(a). In order to maintain the same level of coverage, we require four




(a) The network after a hydrant is removed. (b) A connected solution on the new graph.

Figure 4.7: The resulting network and solution after removal of a *below-ground* hydrant.

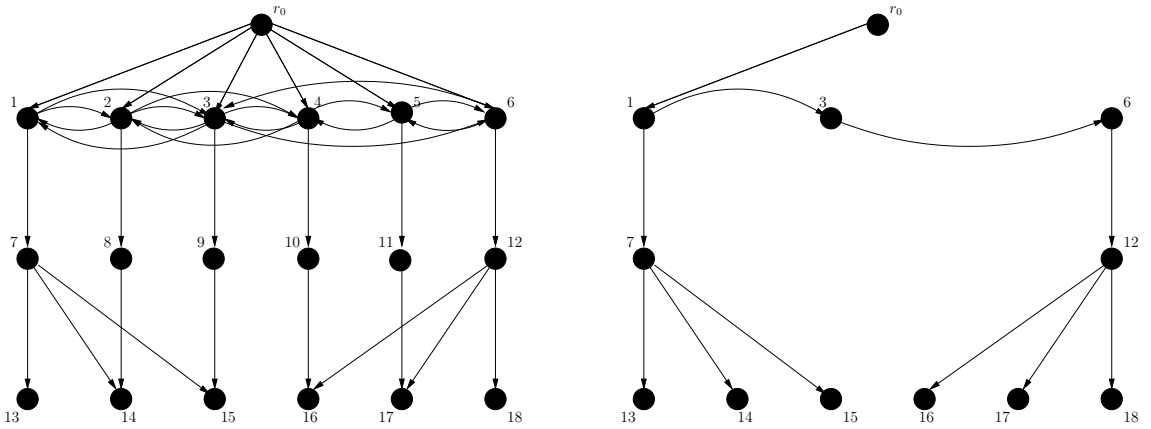
full *LORNO* installations. Such a solution is shown in Figure 4.7(b). This solution has a total cost

of

$$4c - (c_{6,11} + c_{7,12} + c_{10,13} + c_{9,14} + c_{10,15}),$$

an increase of $3c$ over the previous solution.

**Example 7. Removal of an below-ground hydrant.** Now, consider the *LORNO* network shown in Figure 4.8(a). We can find connected solution with two full *LORNO* installations and one radio




(a) The *LORNO* network. (b) A connected solution with 3 above-ground hydrants.

Figure 4.8: The *LORNO* network and corresponding solution from Example 7.

installation, as shown in Figure 4.7(b). The solution shown in Figure 4.9(b) has a total cost of

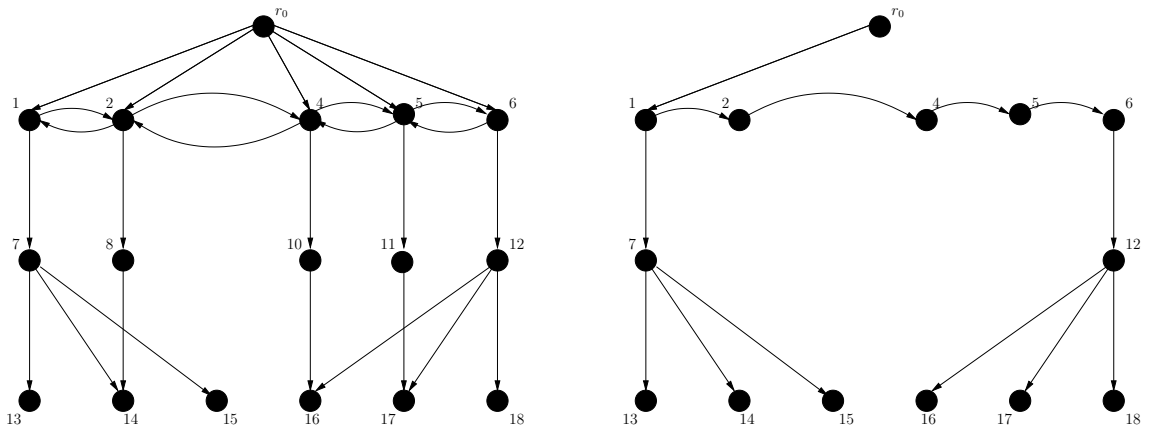
$$2c + c_{\overline{h}} - (c_{8,11} + c_{8,12} + c_{8,13} + c_{8,14} + c_{8,15}).$$

However, if we remove the above-ground hydrant represented by vertex 3, we have the resulting network shown in Figure 4.9(a). In order to maintain the same level of coverage, we require two additional radio installations *LORNO* installations. A solution is shown in Figure 4.7(b). This solution has a total cost of

$$2c + 3c_{\overline{h}} - (c_{6,11} + c_{7,12} + c_{10,13} + c_{9,14} + c_{10,15}),$$

an increase of $2c_{\overline{h}}$ over the previous solution.

One way to quantify the benefit of a hydrant is to determine the loss in total leak detection benefit if it is removed from the system. Of particular interest are those hydrants whose removal results in the largest loss in leak detection benefit. One question we may seek to answer is: What are the $p$ worst hydrants to remove, with respect to leak detection? We can employ the bilevel programming




(a) The resulting network after the hydrant is removed.

(b) Example of a connected solution on the new graph.

Figure 4.9: The resulting network and solution after removal of an *above-ground* hydrant.

framework to answer such a question. In particular, we will model the problem as a ILP interdiction problem.

Bilevel programs have previously been used to plan interdiction efforts against terrorist groups operating on a physical network (Wood, 1993; Israeli, 1999; Israeli and Wood, 2002). In these applications, the analyst generally adopts the point of view of the law enforcement agency attempting to reduce the effectiveness of the terrorist group's operations. However, we can also use these models from the opposite perspective, where we adopt the operator's point of view, in order to determine the sensitivity of our leak detection abilities to the structure of our network.

**Interdicting the LORNO Network.** Suppose we have a total interdiction budget $p$, and the cost of interdicting each hydrant is one (i.e., we are simply interested in the number of hydrants interdicted). In other words, we are interested in the effect of removing any $p$ hydrants. Let

$$w_i = \begin{cases} 1 & \text{if hydrant } i \text{ is interdicted} \\ 0 & \text{otherwise} \end{cases}.$$

Then, we can define an interdiction model which yields the worst-case outcome of resulting from removal of $p$ hydrants from the water network. For the sensitivity analysis described above, it may be desirable to simply set $p = 1$, to determine the single hydrant whose removal would have the greatest impact on cost (benefit). After adding the necessary interdiction constraints, we have the

following network design interdiction formulation, an special case of (MIPINT) (and IBLP):

$$
\begin{aligned}
\max\min \quad & \sum_{(i,j)\in E} c_{ij}x_{ij} \\
\text{subject to} \quad & \sum_{i\in\overline{H}} w_i \le p \\
& \sum_{(j,i)\in E} x_{ji} = y_i, \quad \forall i \in V - \{r_0\} \\
& \sum_{(r_0,i)\in E} x_{r_0 i} = 1 \\
& x\left(\delta^-(S)\right) \ge y_k, \quad \forall (S,k) \in \mathcal{L} \\
& \sum_{(j,i)\in E} x_{ji} \le \sum_{(i,j)\in E} x_{ij}, \quad \forall i \notin R \cup \{r_0\} \\
& y_i \le 1 - x_{r_0 j}, \quad \forall i < j, \{i,j\} \subset \overline{H} \\
& x_{ij} + x_{ji} \le y_i, (x_{ij} \le y_i), \quad \forall (i,j) \in E, i \in V - \{r_0\} \\
& y_i \le 1 - w_i, \qquad \forall i \in V - \{r_0\} \\
& x_{ij}, y_i \in \{0,1\}, \quad \forall i \in V - \{r_0\}, \forall (i,j) \in E, \\
& w_i \in \{0,1\}, \quad \forall i \in V - \{r_0\}.
\end{aligned}
$$

Here, we assume that the central server cannot be removed, since this would effectively render the system useless. Other than this caveat, we allow removal of any hydrant (above- or below-ground) in the *LORNO* network, but note that several of the algorithms in this dissertation rely on the additional assumption of feasibility on the lower-level problem, meaning we only consider hydrants whose removal does not prevent a connected communication network. In this application, those hydrants whose removal does result in such an infeasible lower-level problem is likely of interest, but we can easily modify the algorithmic assumptions to include this case (see Chapter 2).

It is important to note that this formulation does not yield the resulting level of protection upon removal of the hydrants from the current, *LORNO*-monitored network. Rather, it provides the lowest-cost connected communication network that can be designed without these hydrants available. If a hydrant is removed from the network, and the *LORNO* sensor optimization problem is not resolved, it is possible that we are left with a disconnected network, rendering some (or all) of the remaining sensors useless. In Chapter 2, we provided an algorithm to solve the general IBLP. In the following section, we describe specialized methods for interdiction problems aimed at exploiting problem structure meant to improve the performance of our algorithm on interdiction problems.

## 4.2 General Interdiction Problems

In this section, we demonstrate methods for incorporating problem-specific customizations into our solver framework. In order to illustrate such methods, we derive several methods that exploit the structure of (MIPINT), thereby leading to more effective algorithm design. While these methods have important computational implications for interdiction problems, they are also meant as an example of how users of MibS can supplement the built-in methods to yield better results for their own applications.

### 4.2.1 Cutting Plane Methods

In this section, we describe two methods for generating cutting planes. These methods are, in fact, applicable to a slightly wider range of problems than those covered by (MIPINT), since their derivation relies only on the requirement $X = \mathbb{B}^{n_1}$.

**No-good Cuts.** During preliminary computational experiments with MibS on interdiction problems, we discovered that our algorithm frequently generates sequences of integer bilevel infeasible solutions of the form

$$(\hat{x}, y^1), (\hat{x}, y^2), \ldots, (\hat{x}, y^k)$$

such that $y^i \notin M^I(\hat{x})$ for $i < k$. In particular, the bilevel feasibility cut (2.3) of Chapter 2 (by design) separates only the current integer point, allowing for this type of sequence generation.

If $x \in \mathbb{B}^{n_1}$, information obtained from the lower-level problem can be used to avoid this problem. While checking bilevel feasibility, we obtain an optimal solution $y^*$ and associated optimal value $z_L(\hat{x})$ for the lower-level problem

$$\min_{y \in \mathcal{S}_L(\hat{x}) \cap Y} d^2 y$$

and, thus, a feasible solution to (IBLP). This leads to the implication

$$x = \hat{x} \Rightarrow d^2 y = z_L(\hat{x}).$$

Therefore, if we store the solution $(\hat{x}, y^*)$, we can add a cut that separates $(\hat{x}, y)$ for $y \in Y$ from $\Omega^I$.

Let $I_0 := \{i \mid \hat{x}_i = 0\}$ and $I_1 := \{i \mid \hat{x}_i = 1\}$. Note that for $x \in \mathbb{B}^{n_1}$, we have that

$$\sum_{i \in I_0} x_i + \sum_{i \in I_1} (1 - x_i) = 0$$

if and only if $x = \hat{x}$. Otherwise,

$$\sum_{i \in I_0} x_i + \sum_{i \in I_1} (1 - x_i) \geq 1.$$

Thus, adding the cut

$$\sum_{i \in I_0} x_i + \sum_{i \in I_1} x_i \geq 1 - |I_1|,$$

imposes $x \neq \hat{x}$.

**Increasing Objective Cuts.** Let $Y = \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}$, and

$$\mathcal{S}_L(\hat{x}) = \left\{ G^2 y \geq b^2 - A^2 \hat{x}, y \in Y \right\}. \tag{4.4}$$

Suppose $\mathcal{S}_L(\hat{x}) \subseteq \mathcal{S}_L(x)$ for some $x \in (\mathcal{P}_U \cap X)$. Then, it is clear that

$$\min_{y \in \mathcal{S}_L(x)} d^2 y = z_{MILP}(x) \leq z_{MILP}(\hat{x}) = \min_{y \in \mathcal{S}_L(\hat{x})} d^2 y$$

since $y \in \mathcal{S}_L(x)$ implies $y \in \mathcal{S}_L(\hat{x})$. Thus, $z_{MILP}(\hat{x})$ yields an upper bound on the lower-level objective function for such an $x$.

Note that, for $\mathcal{S}_L(\hat{x}) \subseteq \mathcal{S}_L(x)$, we only require $b^2 - A^2 x \leq b^2 - A^2 \hat{x}$. Thus, if $A^2 \in \mathbb{R}_+^{m_2 \times n_1}$, $\mathcal{S}_L(\hat{x}) \subseteq \mathcal{S}_L(x)$ for any $x \geq \hat{x}$. Alternatively, if $A^2 \in \mathbb{R}_-^{m_2 \times n_1}$, $\mathcal{S}_L(\hat{x}) \subseteq \mathcal{S}_L(x)$ for any $x \leq \hat{x}$. This is formalized in the following proposition.

**Proposition 4.2** *Let $Y = \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2 - p_2}$, and $\mathcal{S}_L(\hat{x})$ be defined as in (4.4). Then, if:*

*(i)* $A^2 \in \mathbb{R}_+^{m_2 \times n_1}$, $\mathcal{S}_L(\hat{x}) \subseteq \mathcal{S}_{LL}(x)$ *for all* $x \geq \hat{x}$

*(ii)* $A^2 \in \mathbb{R}_-^{m_2 \times n_1}$, $\mathcal{S}_L(\hat{x}) \subseteq \mathcal{S}_{LL}(x)$ *for all* $x \leq \hat{x}$

*In either case, we have*

$$\min_{y \in \mathcal{S}_L(x)} d^2 y = z_{MILP}(x) \leq z_{MILP}(\hat{x}) = \min_{y \in \mathcal{S}_L(\hat{x})} d^2 y.$$

One case for which we know these conditions to hold is (MIPINT). Recall from Chapter 1:

$$\mathcal{S}_L^{\text{INT}}(x) = \left\{ y \in \mathbb{R}^n \mid G^2 y \leq b^2, -y \geq -U(e - x), y \geq 0 \right\}.$$

**Corollary 4.3** *Let*

$$z^{INT}_{MILP}(x) = \min\{d^2y \mid y \in (\mathcal{S}^{INT}_L(x) \cap Y)\}.$$

*Then,*

$$z^{INT}_{MILP}(x) \leq z^{INT}_{MILP}(\hat{x}),$$

*for all* $x \leq \hat{x}$.

**Proof.** (MIPINT) satisfies Condition (ii) of Proposition 4.2. □

Note that these do not depend on $X = \mathbb{B}^{n_1}$. However, when all upper-level variables are binary, generating valid inequalities from these results becomes more simple. We will see this next.

Using the results above, we would like to derive a cut to separate solutions that are integer, but not bilevel feasible. Let $(\hat{x}, \hat{y}) \in \Omega^I$ be a solution to (LR) and $y^* \in \operatorname{argmin}\{d^2y \mid y \in (\mathcal{S}_L(\hat{x}) \cap Y)\}$ be an optimal lower-level solution determined during the bilevel feasibility check. Also, suppose that $A^2 \in \mathbb{R}^{m_2 \times n_2}_-$. [2] Then, we have the following implication:

$$x \leq \hat{x} \Rightarrow d^2y \leq d^2y^*.$$

It is possible to model this implication by introducing indicator variables $\delta_i \in \mathbb{B}$ for $i = 1, \ldots, n_1+1$ and the set of constraints

$$x_i - (m_i - \epsilon)\delta_i \geq \hat{x}_i + \epsilon, \quad \forall i = 1, \ldots n_1 \tag{4.5}$$

$$\sum_{i=1}^{n_1} \delta_i - \epsilon\delta_{n_1+1} \leq n_1 - \epsilon \tag{4.6}$$

$$d^2y + M\delta_{n_1+1} \leq M + d^2y^*, \tag{4.7}$$

where $m$ is a lower bound on $x_i - \hat{x}_i$, $M$ is an upper bound on $d^2y - d^2y^*$, and $\epsilon$ is a small tolerance. Here, $m_i = l_i - u_i$, where $l_i$ and $u_i$ are natural upper and lower bounds on $x_i$, and $M = \max\{d^2y \mid (x, y) \in \Omega^I\} - d^2y^*$ will suffice. While, (4.5)-(4.7) can be used for the general case $X = \mathbb{Z}^{p_1}_+ \times \mathbb{R}^{n_1-p_1}_+$, it is likely that $m_i \ll 0$ for some $i$ and $M \gg d^2y^*$. This may cause computational difficulties when implemented.

However, if $X = \mathbb{B}^{n_1}$, we can use the special structure to derive a more "well-behaved" implementation. As before, let $I_0 := \{i \mid \hat{x}_i = 0\}$ and $I_1 := \{i \mid \hat{x}_i = 1\}$. For $x \in \mathbb{B}^{n_1}$, we have that

$$\sum_{i \in I_0} x_i = 0$$

[2] Proposition 4.2 also covers the case $A^2 \in \mathbb{R}^{m_2 \times n_2}_+$, but we omit the results here since they are analogous.

if and only if $x \leq \hat{x}$. Otherwise, we have

$$\sum_{i \in I_0} x_i \geq 1.$$

Thus, applying the results above yields the disjunction

$$\begin{array}{rcl} \sum_{i \in I_0} x_i & \leq & 0 \\ d^2 y & \leq & d^2 y^* \end{array} \quad \text{OR} \quad \begin{array}{rcl} \sum_{i \in I_0} x_i & \geq & 1, \end{array}$$

which is violated by $(\hat{x}, \hat{y})$, but satisfied by all members of $\mathcal{F}^I$. Disjunctions of this type can be applied directly as a branching rule used whenever solutions $(\hat{x}, \hat{y}) \in X \times Y$ to (LR) that are not bilevel feasible.

Alternatively, we can use this disjunction to generate a disjunctive cut using the same methodology as discussed previously in Section 3.2.4. The two polyhedra, denoted $\mathcal{P}^1$ and $\mathcal{P}^2$, that result if we combine this disjunction with the original set of constraints in $\Omega$:

$$\mathcal{P}^1 = \left\{ \begin{array}{rcl} -A^1 x & \leq & -b^1 \\ -A^2 x - G^2 y & \leq & -b^2 \\ \sum_{i \in I_0} x_i & \leq & 0 \\ d^2 y & \leq & d^2 y^* \\ x, y & \geq & 0 \end{array} \right\}$$

and

$$\mathcal{P}^2 = \left\{ \begin{array}{rcl} -A^1 x & \leq & -b^1 \\ -A^2 x - G^2 y & \leq & -b^2 \\ -\sum_{i \in I_0} x_i & \leq & -1 \\ x, y & \geq & 0 \end{array} \right\}$$

Let $(u^i, v^i, w^i, z^i)$ be multipliers for the constraints defining $\mathcal{P}^i$. The following inequalities are valid for $\mathcal{P}^1$ and $\mathcal{P}^2$, respectively:

$$\begin{array}{lll} -u^1 A^1 x - v^1 A^2 x + w^1 \sum_{i \in I_0} x_i - v^1 G^2 y + z^1 d^2 y \leq u^1 b^1 - v^1 b^2 & & + z^1 d^2 y^* \\ -u^2 A^1 x - v^2 A^2 x - w^2 \sum_{i \in I_0} x_i - v^2 G^2 y & \leq u^2 b^1 - v^2 b^2 - w^2 & . \end{array}$$

As in Chapter 3, we derive the *cut generation LP*:

$$\begin{aligned}
\min \quad & \alpha\hat{x} + \beta\hat{y} - \gamma \\
\text{s.t.} \quad & \alpha + u^1A^1 + v^1A^2 - w^1e^{I_0} \leq 0 \\
& \alpha u^2A^1 + v^2A^2 + w^2e^{I_0} \leq 0 \\
& \beta + v^1G^2 - z^1d^2 \leq 0 \\
& \beta + v^2G^2 \leq 0 \\
& \gamma + u^1b^1 + v^1b^2 - z^1d^2y^* \geq 0 \\
& \gamma + u^2b^1 + v^2b^2 + w^2 \geq 0 \\
& \sum_{i=1}^{m_1} u_i^1 + \sum_{i=1}^{m_2} v_i^1 + w^1 + z^1 + \sum_{i=1}^{m_1} u_i^2 + \sum_{i=1}^{m_2} v_i^2 + w^2 + z^2 = 1 \\
& u^1, u^2, v^1, v^2, w^1, w^2, z^1, z^2 \geq 0,
\end{aligned} \tag{4.8}$$

where $e^{I_0}$ is a row vector such that $e_i^{I_0} = 1$ if $i \in I_0$ and $e_i = 0$ otherwise. This is formalized in the following result.

**Theorem 4.4** *Let $X = \mathbb{B}^{n_1}$, $Y = \mathbb{Z}_+^{p_2} \times \mathbb{R}_+^{n_2-p_2}$, $I_0 := \{i \mid \hat{x}_i = 0\}$ and $I_1 := \{i \mid \hat{x}_i = 1\}$. Let $e^{I_0}$ and $e^{I_1}$ be row vectors such that*

$$e_i^{I_0} = \begin{cases} 1 & \text{if } i \in I_0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad e_i^{I_1} = \begin{cases} 1 & \text{if } i \in I_1 \\ 0 & \text{otherwise} \end{cases}.$$

*Finally, let $(\hat{x}, \hat{y}) \in \Omega^I$ be a solution to (LR) and $y^* \in \operatorname{argmin}\{d^2y \mid y \in \mathcal{P}_L(\hat{x}) \cap Y\}$. Then, if:*

*(i) $A^2 \in \mathbb{R}_+^{m_2 \times n_1}$ and $(\alpha^*, \beta^*, \gamma^*, u^*, v^*, w^*, z^*)$ is a solution to*

$$\begin{aligned}
\alpha + u^1A^1 + v^1A^2 + w^1e^{I_1} &\leq 0 \\
\alpha + u^2A^1 + v^2A^2 - w^2e^{I_1} &\leq 0 \\
\beta + v^1G^2 - z^1d^2 &\leq 0 \\
\beta + v^2G^2 &\leq 0 \\
\gamma u^1b^1 - v^1b^2 + |I_1|w^1 - z^1d^2y^* &\geq 0 \\
\gamma u^2b^1 - v^2b^2 - (|I_1| - 1)w^2 &\geq 0 \\
\sum_{i=1}^{m_1} u_i^1 + \sum_{i=1}^{m_2} v_i^1 + w^1 + z^1 + \sum_{i=1}^{m_1} u_i^2 + \sum_{i=1}^{m_2} v_i^2 + w^2 + z^2 &= 1 \\
u^1, u^2, v^1, v^2, w^1, w^2, z^1, z^2 &\geq 0,
\end{aligned} \tag{4.9}$$

*such that* $\alpha^*\hat{x} + \beta^*\hat{y} > \gamma^*$*, then* $\alpha^*\hat{x} + \beta^*\hat{y} \leq \gamma^*$ *is a valid cutting plane separating* $(\hat{x}, \hat{y})$ *from* $\text{conv}(\mathcal{F}^I)$.

*(ii)* $A^2 \in \mathbb{R}^{m_2 \times n_1}_-$ *and* $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{u}, \bar{v}, \bar{w}, \bar{z})$ *is a solution to*

$$\begin{aligned}
\alpha + u^1A^1 + v^1A^2 - w^1e^{I_0} &\leq 0 \\
\alpha + u^2A^1 + v^2A^2 + w^2e^{I_0} &\leq 0 \\
\beta + v^1G^2 - z^1d^2 &\leq 0 \\
\beta + v^2G^2 &\leq 0 \\
\gamma u^1b^1 + v^1b^2 - z^1d^2y^* &\geq 0 \\
\gamma u^2b^1 + v^2b^2 + w^2 &\geq 0 \\
\sum_{i=1}^{m_1} u_i^1 + \sum_{i=1}^{m_2} v_i^1 + w^1 + z^1 + \sum_{i=1}^{m_1} u_i^2 + \sum_{i=1}^{m_2} v_i^2 + w^2 + z^2 &= 1 \\
u^1, u^2, v^1, v^2, w^1, w^2, z^1, z^2 &\geq 0,
\end{aligned} \tag{4.10}$$

*such that* $\bar{\alpha}\hat{x} + \bar{\beta}\hat{y} > \bar{\gamma}$*, then* $\bar{\alpha}\hat{x} + \bar{\beta}\hat{y} \leq \bar{\gamma}$ *is a valid cutting plane separating* $(\hat{x}, \hat{y})$ *from* $\text{conv}(\mathcal{F}^I)$.

One way we can utilize these cuts is by considering maximal and minimal upper-level solutions. For example, if $A^2 \in \mathbb{R}^{m_2 \times n_1}_-$, let $\mathcal{S}_L(\hat{x})$ such that $\hat{x}$ is maximal with respect to $\sum_{i=1}^{n_1} x_i$. In other words,

$$\hat{x} \in \text{argmax}\left\{e_{n_1}^T x \mid \mathcal{S}_{LL(x)} \neq \emptyset\right\}.$$

### 4.2.2 Greedy Interdiction

Now, we describe a heuristic method for generating feasible solutions which exploits the special structure of (MIPINT) and utilizes sensitivity information obtained from solving the lower-level LP relaxation. Note that finding feasible solutions for interdiction problems is straightforward under our assumptions, since we must only specify a feasible interdiction plan and solve the resulting lower-level problem. That is, choosing a set of indices $\mathcal{I} \in \{1, \ldots, n_1\}$ such that $A^1 x_I \leq b^1$ and solving

$$\min_{y \in Y}\{d^2y \mid G^2y \geq b^2, y \leq U, y_i = 0, i \in \mathcal{I}\}$$

yields a feasible MIPINT solution. Of course, the choice of $\mathcal{I}$ will dictate the quality of such a solution. Thus, a variety of heuristics could be derived by following this basic framework and specifying methods for choosing $\mathcal{I}$. One obvious method is to choose interdiction variables which give the greatest immediate decrease in the lower-level objective. That is, at iteration $t$ of the

algorithm, we add variable $j$ to the current interdiction set $\mathcal{I}^t$, where

$$j \in \text{argmin}_{j \in N \setminus \mathcal{I}^{t-1}} d_j^2.$$

We must, of course, maintain upper-level feasibility throughout the course of the algorithm, so must check

$$\sum_{i \in \mathcal{I}^t} a_{ij}^1 x_i \leq b_j^1, \quad j = 1, \ldots, m_1$$

at each iteration. If this condition is not satisfied, we discard $j$ and move to the next-best interdiction choice. Once an interdiction variable is chosen, it is kept for the remainder of the algorithm. This heuristic is summarized in Algorithm 4.2. One potential variation of this algorithm would involve replacing previously-chosen interdiction variables based on a specified criteria. This method can

**Algorithm 4.2** Greedy Interdiction

1: Set $\mathcal{I}^0 = \emptyset$, $N^0 = \{1, \ldots, n\}$, and $t = 1$.
2: While $N \neq \emptyset$ do:

- Let $j^t = \text{argmin}_{j \in N \setminus \mathcal{I}^{t-1}} d_j^2$ with ties broken arbitrarily.
- If
$$\sum_{i \in \mathcal{I}^t} a_{ij}^1 x_i > b_j^1,$$
for any $j = 1, \ldots, m_1$, stop. $\mathcal{I}^{t-1}$ is a greedy solution.
- If
$$\sum_{i \in \mathcal{I}^t} a_{ij}^1 x_i \leq b_j^1, \quad j = 1, \ldots, m_1,$$
set $\mathcal{I}^t = \mathcal{I}^{t-1} \cup \{j^t\}$.

also be used as a stand-alone algorithm. When embedded in an exact algorithmic framework, it is necessary to modify the algorithm to ensure that interdiction plans are not repeated. One way in which to implement such a modification is to add randomness to Step 2.

### 4.2.3 Computational Results

As mentioned previously, determining effective methods for solving (MIPINT) is of interest for several reasons. First, MIPINTs have important applications in infrastructure protection and other in homeland security problems; especially for analyzing systems where network interdiction models are limited by their assumption on system structure. Second, MIPINT can be used to perform a type of sensitivity analysis to determine the effect of removing variables from the model. Through such an analysis, we may discover that the optimal solution, or the model itself, is heavily reliant

on a small number of decision variables. This would suggest that perhaps a solution with less vulnerability may be worth the potential degradation in objective value. Further, it allows us to examine the inherent vulnerability in the system and, potentially, discover ways in which it can be made more robust. Finally, using instances of MIPINT allows us to demonstrate the effectiveness of our specialized methods for interdiction problems.

We tested our branch-and-cut algorithm on two sets of interdiction problems with special structure. In the first set, the lower-level problems were binary knapsack problems with a single constraint. In the second, the lower-level were composed of assignment constraints. In addition, we also tested our algorithm on more general instances of MIPINT, where the lower-level problems are randomly-generated MILPs. Each of these tests is discussed below. All computational tests were performed on an Intel Xeon 2.4GHz processor with 4GB of memory.

**Knapsack Interdiction.** For the knapsack interdiction, the goal of the upper-level DM was to minimize the maximum profit achievable by the lower-level DM by fixing a subset of the variables in the lower-level problem to zero. A cost was associated with the fixing of each lower-level variable to zero and the upper-level problems contained a single constraint, representing the available interdiction budget.

To create these instances, data files describing bicriteria knapsack problems were taken from the *Multiple Criteria Decision Making* library (Figueira, 2000). The first objective in each file was used to define a lower-level objective function, while the second objective provided a budget constraint. We chose instances with no correlation between the two objectives. The available budget was chosen to be $\lceil \sum_{i=1}^{n} a_i/2 \rceil$, where $a_i$ is the cost of interdicting lower-level variable $i$. For a knapsack problem with $n$ items, this construction yielded a problem with $2n$ variables and $n + 2$ constraints.

Summarized results of two sets of runs on the knapsack set – one in which we used maximum infeasibility branching to select branching candidates and one in which we used strong branching – are shown in Table 4.7, where the results for each problem size reflect the average of 20 instances. In each case, all results shown reflect the use of the specialized methods described in this chapter, as well as the general heuristic methods provided in Chapter 2.

Implicit in the formulation of the knapsack interdiction problems described above is the assumption that the interdiction budget is fixed. However, as suggested in our analysis of the *LORNO* system, this may not be the case in a real application. Rather, we may wish to understand the tradeoff between the interdiction budget and the resulting effect on the follower's objective function value. One way to gain this understanding in the interdiction setting is via the multiobjective framework described in Chapter 1 and used to motivate a heuristic in Chapter 3.

Table 4.7: Summary results from the knapsack interdiction.

| | Maximum Infeasibility | | | Strong Branching | | |
|---|---|---|---|---|---|---|
| $2n$ | Avg Nodes | Avg Depth | Avg CPU (s) | Avg Nodes | Avg Depth | Avg CPU (s) |
| 20 | 359.30 | 8.65 | 9.32 | 358.30 | 8.65 | 11.07 |
| 22 | 658.40 | 9.85 | 18.50 | 658.20 | 9.85 | 18.92 |
| 24 | 1414.80 | 10.85 | 46.03 | 1410.80 | 10.75 | 46.46 |
| 26 | 2725.00 | 12.05 | 97.55 | 2723.50 | 12.05 | 100.17 |
| 28 | 5326.40 | 12.90 | 214.97 | 5328.60 | 12.95 | 220.26 |
| 30 | 10625.00 | 14.05 | 482.70 | 10638.00 | 14.10 | 538.32 |

**Biobjective Interdiction Problems.** Suppose row $i$ of upper-level constraint system represents the interdiction budget constraint. Let $A^1_i$ represent the $i$th row of $A^1$ and let $A^1_{-i}$ and $b^1_{-i}$ represent the upper-level constraint matrix and right hand side after removing the $i$th row. Then, for each $i = 1, \ldots, m_1$, we can define a biobjective version of MIPINT (BMIPINT):

$$\text{vmax}_{x \in \mathcal{P}^{\text{INT}}_{U_{-i}} \cap \mathbb{B}^n, y \in \mathcal{S}^{\text{INT}}_L} [dy, -A^1_i x] \qquad \text{(BMIPINT}_i\text{)}$$

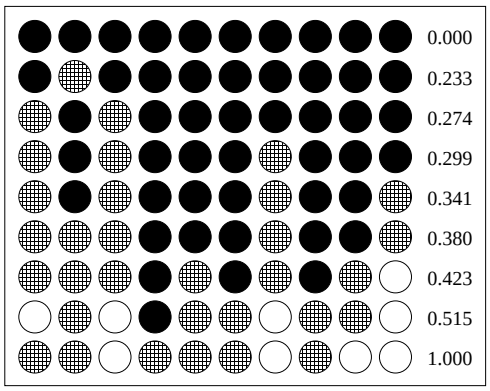
where

$$\mathcal{P}^{\text{INT}}_{U_{-i}} = \left\{ x \in \mathbb{R}^n \mid A^1_{-i} x \leq b^1_{-i} \right\}.$$

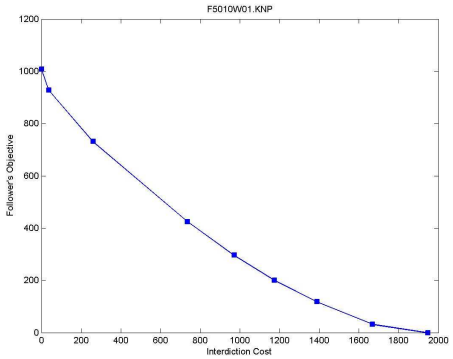
(BMIPINT$_i$) is an example of *biobjective mixed integer bilevel linear program* (BMIBLP).

In order to illustrate how one might use a BMIBLP, or BMIPINT in this case, to perform tradeoff analysis, we return to our knapsack test set. Moving our budget constraint to the to the objective yields an instance of BMIPINT, which can then be converted to a standard MIPINT instance using the single-level reformulation (1.4). Recall that solutions to the resulting subproblem are guaranteed to be efficient, and systematic variation of the weighting $\delta$ will yield a portion of the efficient set. In our setting, this means that the solution that results from each weighting $\delta$ is an efficient interdiction plan.

Figure 4.10(a) illustrates how the optimal interdiction plans change as we vary the weighting $\delta$. In the figure, each column represents a potential activity to be undertaken by the system operator (lower-level DM). The rows correspond to different weightings of $\delta$ (indicated by the values at the right of the figure). In each row, the black dots represent activities that are interdicted by the attacker and the hashed dots represent activities undertaken by the follower. The white dots represent actions neither taken by the follower nor interdicted by the attacker. In this example, the objectives were to minimize the amount of resources consumed by the upper-level and maximize the effect of interdiction (on the lower-level problem). These objectives were given weights $\delta$ and $1 - \delta$, respectively. Examining the figure, from top to bottom, we can see how as $\delta$ increases, ($1 - \delta$ decreases), decisions become more contingent on resource consumption, and less aggressive interdiction plans




(a) Interdiction plans evolution.




(b) Tradeoff curve for MIPINT.

Figure 4.10: Illustrating the tradeoff between the interdiction expense and effectiveness.

are implemented. Visually, this is seen by the appearance of less black dots, which indicates a more passive interdiction strategy, and more hashed dots, which indicates greater flexibility for the lower-level DM. In the extreme cases, where $\delta$ lies on the boundary of $[0, 1]$, we have placed all emphasis on one objective or another. For example, when $\delta = 0$, effectively, we are saying that money is no object, and we are only concerned with the effect of the interdiction efforts. In this case, it is clear that we should interdict all of the defender's possible decisions, to ensure the maximum effect. On the other hand, when $\delta = 1$, our only concern is the consumption of the resources, with no weight being placed on the effect of the interdiction strategy. In this case, there is no reason to take any action, since doing nothing will provide the lowest interdiction cost. An example of the tradeoff curve generated by the algorithm for one particular instance is shown in Figure 4.10(b). From this curve, we see the effect of the interdiction on the lower-level DM's ability to achieve her objective. On the far left side of the plot, no resources have been spent on interdiction, the follower is allowed to act freely, and achieves the best possible scenario for herself. This point corresponds to the last row in Figure 4.10(a). On the other hand, the far right hand side shows the case where the upper-level DM is not limited by interdiction resources and can, therefore, completely prevent the defender from operating her system. This point corresponds to the top row of Figure 4.10(a). The portions of the curve in which we are most interested are those areas with a steep slope. These areas represent critical points, where small increases in the planned interdiction budget yield very substantial increases in effectiveness. Assuming the budget is somewhat flexible, these critical points determine where it is worthwhile to increase the planned resources for the interdiction effort. Alternatively, we may discover that, although the full budget allocated is being used, a significant portion of the resources is being used for a very small marginal increase in effectiveness. Then, we may wish to forgo this increase and allocate the resources to alternative efforts, where they can be used more effectively.

**Assignment Interdiction.** As in the knapsack interdiction described above, the goal of the upper-level DM in assignment interdiction is to maximize the minimum cost achievable by the lower-level DM by fixing a subset of the variables in the lower-level problem to zero. As before, a cost was associated with the interdicting each lower-level variable. The upper-level problems contained a single knapsack constraint, representing the available interdiction budget.

Data files obtained from *Multiple Criteria Decision Making* library (Figueira, 2000) were also used to create these instances. In this case, the original problems represented bicriteria assignment problems. Again, the objectives were used to define a lower-level objective function and budget constraint. The budget for the assignment problems was chosen to be a fixed percentage of $\sum_{i=1}^{n} a_i$. Each problem contains 50 (i.e. $2n, n = 25$) variables and 45 (i.e $m + n + 1, m = 20$) inequality constraints. Note that, in a loose sense, the difficulty of these problems is determined by the number of possible upper-level solutions; thus, budgets were chosen to yield interesting problems that could be handled on a single processor. The results for the assignment set are shown in Table 4.8, where the results again reflect the same specialized interdiction methods and primal heuristics as in the knapsack test set. All tests were performed on an AMD Opteron Processor 6128 with 32GB of memory.

| Instance | Obj. Value | No. Nodes | Depth | Gap (%) | No. Cuts | CPU (s) |
|---|---|---|---|---|---|---|
| 2AP05-1 | -36 | 7045 | 25 | — | 3460 | 39.95 |
| 2AP05-2 | -46 | 19607 | 24 | — | 3992 | 80.17 |
| 2AP05-3 | -46 | 3431 | 25 | — | 1370 | 16.08 |
| 2AP05-4 | -25 | 4313 | 25 | — | 1382 | 17.61 |
| 2AP05-5 | -38 | 3743 | 25 | — | 1294 | 16.85 |
| 2AP05-6 | -32 | 2355 | 25 | — | 1008 | 12.85 |
| 2AP05-7 | -49 | 3391 | 25 | — | 1598 | 18.22 |
| 2AP05-8 | -41 | 3543 | 25 | — | 2154 | 22.23 |
| 2AP05-9 | -54 | 1917 | 25 | — | 1142 | 11.18 |
| 2AP05-10 | -46 | 5085 | 25 | — | 2348 | 26.99 |
| 2AP05-11 | -36 | 759 | 25 | — | 290 | 3.18 |
| 2AP05-12 | -49 | 5445 | 27 | — | 3008 | 33.85 |
| 2AP05-13 | -74 | 1985 | 25 | — | 884 | 9.77 |
| 2AP05-14 | -68 | 4621 | 25 | — | 2854 | 28.43 |
| 2AP05-15 | -48 | 2845 | 25 | — | 1264 | 15.34 |
| 2AP05-16 | -34 | 2317 | 25 | — | 706 | 10.54 |
| 2AP05-17 | -66 | 8909 | 25 | — | 4628 | 56.78 |
| 2AP05-18 | -35 | 7615 | 25 | — | 3230 | 39.82 |
| 2AP05-19 | -39 | 2317 | 25 | — | 1114 | 13.04 |
| 2AP05-20 | -42 | 2117 | 25 | — | 830 | 10.46 |
| 2AP05-21 | -47 | 1897 | 25 | — | 1230 | 12.06 |
| 2AP05-22 | -62 | 1741 | 25 | — | 990 | 9.77 |
| 2AP05-23 | -68 | 2543 | 26 | — | 962 | 12.42 |
| 2AP05-24 | -76 | 51 | 16 | — | 10 | 0.12 |
| 2AP05-25 | -45 | 9457 | 26 | — | 3882 | 47.96 |

Table 4.8: Results from assignment test set.

**ILP Interdiction.** We also tested our algorithms on a set of randomly-generated IPINT instances. In these problems, the goal of the upper-level DM is to minimize the maximum cost solution achievable by the lower-level DM, who is solving a random ILP. As in the previous two examples, a cost was associated with the interdiction of each lower-level variable and the upper-level problems contained a single knapsack constraint, representing the available interdiction budget. In order to create these instances, we first generated a set ILPs of the desired size, with randomly-chosen coefficients in the range $[-50, 50]$. From these ILPs, we created IPINT instances by setting interdiction costs for each lower-level variable.

We assigned costs in two ways for these test problems. Initially, we assigned a unit cost to the interdiction of each lower-level variable. Such a cost structure is appropriate if we are concerned with questions such as "what are the $k$ most crucial variables in the lower-level problem?", similar to that asked in Section 4.1.7. For these problems, we allow up three lower-level variables to be interdicted. We also generated a second set of IPINT instances with randomly-selected costs and interdiction budget. In order to allow comparison between the two instance classes, we chose the interdiction budgets in the second class such that the number of variable interdictions allowed was approximately the same in the first class. The ILP classes we interdicted for these experiments are summarized in Table 4.9. The column and row dimensions of the full IPINT are equal to $2n$ and $m + n + 1$, respectively, where $n$ and $m$ are the corresponding dimensions of the ILP instance.

| Problem Class | Num Rows | Num Cols |
|---|---|---|
| 1 | 10 | 10 |
| 2 | 15 | 10 |
| 3 | 20 | 20 |

Table 4.9: ILP Classes Interdicted.

We used these instances to test the performance of our exact algorithm, with the additional methods described in this chapter, as well as the greedy algorithm as a standalone heuristic. All tests were performed on an AMD Opteron Processor 6128 with 32GB of memory. A summary presentation of the results is given in Table 4.10. The summary results are useful for comparing the average difficulty of the two test sets. We can conclude that, on average, the random instances require less computational effort than their symmetric counterparts.

| | Avg. No. Nodes | | Avg. Depth | | Avg. Gap (%) | | Avg. No. Cuts | | Avg. CPU (s) | |
|---|---|---|---|---|---|---|---|---|---|---|
| Class | Symmetric | Random | Symmetric | Random | Symmetric | Random | Symmetric | Random | Symmetric | Random |
| 1 | 13255.20 | 13199.40 | 23.00 | 23.60 | — | — | 13327.40 | 10828.80 | 187.12 | 162.42 |
| 2 | 58639.40 | 54055.20 | 24.30 | 24.60 | — | — | 49633.20 | 35861.00 | 583.62 | 452.66 |
| 3 | 364098.70 | 559028.00 | 46.20 | 47.40 | 161.23 | 160.78 | 294524.40 | 249707.80 | — | — |

Table 4.10: Summary results from the IPINT instances.

The complete results from the exact solver on the instances with symmetric and randomly-chosen

costs are shown Table 4.11 and Table 4.12, respectively. In Table 4.13, a subset of the results from both test sets are shown together. From this table, we can compare the effect of the interdiction and the required computational effort for the two test sets. Not surprisingly, when the interdiction costs are not symmetric, we see significant differences in the resulting lower-level objective functions for several of the instances, since the set of feasible interdiction plans is different, in general.

| Instance | Obj. Value | No. Nodes | Depth | Gap (%) | No. Cuts | CPU (s) |
|---|---|---|---|---|---|---|
| miblp-10-10-50-0110-0-1 | 0 | 875 | 15 | — | 1090 | 10.24 |
| miblp-10-10-50-0110-0-2 | 542 | 1595 | 25 | — | 2172 | 281.78 |
| miblp-10-10-50-0110-0-3 | 168 | 5153 | 23 | — | 6524 | 76.25 |
| miblp-10-10-50-0110-0-4 | 212 | 3809 | 20 | — | 3846 | 45.85 |
| miblp-10-10-50-0110-0-5 | 54 | 14931 | 29 | — | 14568 | 247.22 |
| miblp-10-10-50-0110-0-6 | 89 | 4033 | 24 | — | 4702 | 60.87 |
| miblp-10-10-50-0110-0-7 | 0 | 1009 | 17 | — | 1312 | 10.04 |
| miblp-10-10-50-0110-0-8 | 99 | 2161 | 23 | — | 2878 | 36.33 |
| miblp-10-10-50-0110-0-9 | 35 | 2453 | 17 | — | 2536 | 26.30 |
| miblp-10-10-50-0110-0-10 | 0 | 96533 | 37 | — | 93646 | 1076.33 |
| miblp-15-10-50-0110-0-1 | 19 | 1399 | 17 | — | 1764 | 25.27 |
| miblp-15-10-50-0110-0-2 | 0 | 251 | 10 | — | 334 | 2.62 |
| miblp-15-10-50-0110-0-3 | 47 | 727 | 15 | — | 872 | 7.20 |
| miblp-15-10-50-0110-0-4 | 10 | 11355 | 24 | — | 11476 | 115.37 |
| miblp-15-10-50-0110-0-5 | 267 | 6965 | 25 | — | 7552 | 426.43 |
| miblp-15-10-50-0110-0-6 | 6 | 20813 | 30 | — | 18268 | 180.07 |
| miblp-15-10-50-0110-0-7 | 96 | 263 | 11 | — | 352 | 4.08 |
| miblp-15-10-50-0110-0-8 | 75 | 25307 | 32 | — | 23898 | 409.66 |
| miblp-15-10-50-0110-0-9 | 0 | 411127 | 41 | — | 353148 | 3815.48 |
| miblp-15-10-50-0110-0-10 | 8 | 108187 | 38 | — | 78668 | 850.02 |
| miblp-20-20-50-0110-0-1 | 526 | 244598 | 45 | 162.57 | 261042 | LIM |
| miblp-20-20-50-0110-0-2 | 162 | 419998 | 42 | 143.40 | 353910 | LIM |
| miblp-20-20-50-0110-0-3 | 200 | 323470 | 38 | 159.05 | 321920 | LIM |
| miblp-20-20-50-0110-0-4 | 315 | 430951 | 48 | 158.36 | 362260 | LIM |
| miblp-20-20-50-0110-0-5 | 218 | 407914 | 41 | 170.00 | 389212 | LIM |
| miblp-20-20-50-0110-0-6 | 133 | 1009676 | 68 | 116.28 | 494430 | LIM |
| miblp-20-20-50-0110-0-7 | 310 | 336973 | 43 | 147.54 | 305098 | LIM |
| miblp-20-20-50-0110-0-8 | 376 | 325434 | 39 | 232.51 | 318276 | LIM |
| miblp-20-20-50-0110-0-9 | 325 | 65567 | 35 | 152.09 | 68352 | LIM |
| miblp-20-20-50-0110-0-10 | 873 | 76406 | 63 | 170.49 | 70744 | LIM |

Table 4.11: Exact results from IPINTs with symmetric costs.

| Instance | Obj. Value | No. Nodes | Depth | Gap (%) | No. Cuts | CPU (s) |
|---|---|---|---|---|---|---|
| miblp-10-10-50-0110-0-1 | 11 | 975 | 15 | — | 926 | 9.19 |
| miblp-10-10-50-0110-0-2 | 607 | 1929 | 26 | — | 1866 | 234.46 |
| miblp-10-10-50-0110-0-3 | 210 | 3425 | 23 | — | 3962 | 47.05 |
| miblp-10-10-50-0110-0-4 | 210 | 2767 | 21 | — | 2616 | 30.86 |
| miblp-10-10-50-0110-0-5 | 122 | 12389 | 30 | — | 10906 | 194.77 |
| miblp-10-10-50-0110-0-6 | 89 | 3965 | 25 | — | 3930 | 52.86 |
| miblp-10-10-50-0110-0-7 | 0 | 1045 | 18 | — | 1114 | 8.52 |
| miblp-10-10-50-0110-0-8 | 99 | 1713 | 23 | — | 2028 | 26.47 |
| miblp-10-10-50-0110-0-9 | 35 | 2699 | 17 | — | 2250 | 24.85 |
| miblp-10-10-50-0110-0-10 | 0 | 101087 | 38 | — | 78690 | 995.19 |
| miblp-15-10-50-0110-0-1 | 28 | 1347 | 18 | — | 1174 | 19.70 |
| miblp-15-10-50-0110-0-2 | 0 | 219 | 10 | — | 242 | 2.00 |
| miblp-15-10-50-0110-0-3 | 51 | 721 | 16 | — | 674 | 5.39 |
| miblp-15-10-50-0110-0-4 | 10 | 10815 | 24 | — | 8802 | 91.11 |
| miblp-15-10-50-0110-0-5 | 415 | 5433 | 24 | — | 5852 | 321.75 |
| miblp-15-10-50-0110-0-6 | 6 | 20149 | 31 | — | 15076 | 153.56 |
| miblp-15-10-50-0110-0-7 | 99 | 245 | 12 | — | 252 | 3.41 |
| miblp-15-10-50-0110-0-8 | 75 | 27595 | 32 | — | 19252 | 338.08 |
| miblp-15-10-50-0110-0-9 | 0 | 377649 | 41 | — | 250354 | 2949.34 |
| miblp-15-10-50-0110-0-10 | 8 | 96379 | 38 | — | 56932 | 642.23 |
| miblp-20-20-50-0110-0-1 | 538 | 493866 | 46 | 164.39 | 262674 | LIM |
| miblp-20-20-50-0110-0-2 | 160 | 718853 | 43 | 142.18 | 305730 | LIM |
| miblp-20-20-50-0110-0-3 | 200 | 517380 | 39 | 157.51 | 249392 | LIM |
| miblp-20-20-50-0110-0-4 | 331 | 757069 | 48 | 161.57 | 338806 | LIM |
| miblp-20-20-50-0110-0-5 | 218 | 685632 | 42 | 172.33 | 320710 | LIM |
| miblp-20-20-50-0110-0-6 | 43 | 973159 | 65 | 105.13 | 297378 | LIM |
| miblp-20-20-50-0110-0-7 | 264 | 565006 | 46 | 140.46 | 315354 | LIM |
| miblp-20-20-50-0110-0-8 | 409 | 628377 | 46 | 245.71 | 267310 | LIM |
| miblp-20-20-50-0110-0-9 | 300 | 116540 | 36 | 147.92 | 72416 | LIM |
| miblp-20-20-50-0110-0-10 | 878 | 134398 | 63 | 170.56 | 67308 | LIM |

Table 4.12: Exact results from IPINTs with random costs.

| | Obj. Value | | Avg. No. Nodes | | Avg. No. Cuts | |
|---|---|---|---|---|---|---|
| Instance | Symmetric | Random | Symmetric | Random | Symmetric | Random |
| miblp-10-10-50-0110-0-1 | 0 | 11 | 875 | 975 | 1090 | 926 |
| miblp-10-10-50-0110-0-2 | 542 | 607 | 1595 | 1929 | 2172 | 1866 |
| miblp-10-10-50-0110-0-3 | 168 | 210 | 5153 | 3425 | 6524 | 3962 |
| miblp-10-10-50-0110-0-4 | 212 | 210 | 3809 | 2767 | 3846 | 2616 |
| miblp-10-10-50-0110-0-5 | 54 | 122 | 14931 | 12389 | 14568 | 10906 |
| miblp-10-10-50-0110-0-6 | 89 | 89 | 4033 | 3965 | 4702 | 3930 |
| miblp-10-10-50-0110-0-7 | 0 | 0 | 1009 | 1045 | 1312 | 1114 |
| miblp-10-10-50-0110-0-8 | 99 | 99 | 2161 | 1713 | 2878 | 2028 |
| miblp-10-10-50-0110-0-9 | 35 | 35 | 2453 | 2699 | 2536 | 2250 |
| miblp-10-10-50-0110-0-10 | 0 | 0 | 96533 | 101087 | 93646 | 78690 |
| miblp-15-10-50-0110-0-1 | 19 | 28 | 1399 | 1347 | 1764 | 1174 |
| miblp-15-10-50-0110-0-2 | 0 | 0 | 251 | 219 | 334 | 242 |
| miblp-15-10-50-0110-0-3 | 47 | 51 | 727 | 721 | 872 | 674 |
| miblp-15-10-50-0110-0-4 | 10 | 10 | 11355 | 10815 | 11476 | 8802 |
| miblp-15-10-50-0110-0-5 | 267 | 415 | 6965 | 5433 | 7552 | 5852 |
| miblp-15-10-50-0110-0-6 | 6 | 6 | 20813 | 20149 | 18268 | 15076 |
| miblp-15-10-50-0110-0-7 | 96 | 99 | 263 | 245 | 352 | 252 |
| miblp-15-10-50-0110-0-8 | 75 | 75 | 25307 | 27595 | 23898 | 19252 |
| miblp-15-10-50-0110-0-9 | 0 | 0 | 411127 | 377649 | 353148 | 250354 |
| miblp-15-10-50-0110-0-10 | 8 | 8 | 108187 | 96379 | 78668 | 56932 |
| miblp-20-20-50-0110-0-1 | 526 | 538 | 244598 | 493866 | 261042 | 262674 |
| miblp-20-20-50-0110-0-2 | 162 | 160 | 419998 | 718853 | 353910 | 305730 |
| miblp-20-20-50-0110-0-3 | 200 | 200 | 323470 | 517380 | 321920 | 249392 |
| miblp-20-20-50-0110-0-4 | 315 | 331 | 430951 | 757069 | 362260 | 338806 |
| miblp-20-20-50-0110-0-5 | 218 | 218 | 407914 | 685632 | 389212 | 320710 |
| miblp-20-20-50-0110-0-6 | 133 | 43 | 1009676 | 973159 | 494430 | 297378 |
| miblp-20-20-50-0110-0-7 | 310 | 264 | 336973 | 565006 | 305098 | 315354 |
| miblp-20-20-50-0110-0-8 | 376 | 409 | 325434 | 628377 | 318276 | 267310 |
| miblp-20-20-50-0110-0-9 | 325 | 300 | 65567 | 116540 | 68352 | 72416 |
| miblp-20-20-50-0110-0-10 | 873 | 878 | 76406 | 134398 | 70744 | 67308 |

Table 4.13: Comparison of results from IPINTs with different cost structures.

The results from the greedy heuristic are shown in Tables 4.14 and 4.15. As in Chapter 3, we compare the objective values found by the heuristic to the lower bound provided by solving the underlying MILP:

$$\min_{(x,y)\in\Omega^I} c^1x + d^1y,$$

as well as the upper bounds obtained from the simple heuristic methods. Note that, for interdiction problems, the bounds obtained from the simple heuristic methods are likely to be equivalent, assuming that the solution to the underlying MILP does not contain any nonnegative upper-level variables. For all instances tested, these bounds were, in fact, identical and only one is shown. In the tables below, the bounds are denoted *MILP Bound*, *Easy Bound* and have the same interpretation as in Chapter 3.

| Instance | CPU (s) | Greedy Obj. | MILP Bound | Easy Bound |
|---|---|---|---|---|
| miblp-10-10-50-0110-0-1 | 0.02 | 44.0 | -99.0 | 208.0 |
| miblp-10-10-50-0110-0-2 | 0.28 | 764.0 | -202.0 | 873.0 |
| miblp-10-10-50-0110-0-3 | 0.05 | 257.0 | -420.0 | 548.0 |
| miblp-10-10-50-0110-0-4 | 0.04 | 225.0 | -365.0 | 444.0 |
| miblp-10-10-50-0110-0-5 | 0.04 | 54.0 | -300.0 | 227.0 |
| miblp-10-10-50-0110-0-6 | 0.05 | 105.0 | -239.0 | 469.0 |
| miblp-10-10-50-0110-0-7 | 0.02 | 0.0 | -104.0 | 147.0 |
| miblp-10-10-50-0110-0-8 | 0.05 | 99.0 | -360.0 | 808.0 |
| miblp-10-10-50-0110-0-9 | 0.04 | 35.0 | -358.0 | 269.0 |
| miblp-10-10-50-0110-0-10 | 0.05 | 0.0 | -277.0 | 290.0 |
| miblp-15-10-50-0110-0-1 | 0.04 | 26.0 | -140.0 | 142.0 |
| miblp-15-10-50-0110-0-2 | 0.01 | 66.0 | -314.0 | 172.0 |
| miblp-15-10-50-0110-0-3 | 0.02 | 228.0 | -123.0 | 261.0 |
| miblp-15-10-50-0110-0-4 | 0.04 | 10.0 | -167.0 | 210.0 |
| miblp-15-10-50-0110-0-5 | 0.13 | 301.0 | -241.0 | 624.0 |
| miblp-15-10-50-0110-0-6 | 0.03 | 6.0 | -336.0 | 305.0 |
| miblp-15-10-50-0110-0-7 | 0.04 | 127.0 | -241.0 | 211.0 |
| miblp-15-10-50-0110-0-8 | 0.11 | 75.0 | -425.0 | 238.0 |
| miblp-15-10-50-0110-0-9 | 0.06 | 0.0 | -419.0 | 149.0 |
| miblp-15-10-50-0110-0-10 | 0.04 | 8.0 | -524.0 | 235.0 |
| miblp-20-20-50-0110-0-1 | 0.31 | 526.0 | -887.0 | 885.0 |
| miblp-20-20-50-0110-0-2 | 0.14 | 276.0 | -478.0 | 302.0 |
| miblp-20-20-50-0110-0-3 | 0.31 | 252.0 | -409.0 | 446.0 |
| miblp-20-20-50-0110-0-4 | 0.47 | 560.0 | -566.0 | 600.0 |
| miblp-20-20-50-0110-0-5 | 0.28 | 288.0 | -376.0 | 585.0 |
| miblp-20-20-50-0110-0-6 | 0.23 | 180.0 | -859.0 | 526.0 |
| miblp-20-20-50-0110-0-7 | 0.59 | 310.0 | -697.0 | 619.0 |
| miblp-20-20-50-0110-0-8 | 0.15 | 398.0 | -349.0 | 571.0 |
| miblp-20-20-50-0110-0-9 | 1.09 | 325.0 | -707.0 | 592.0 |
| miblp-20-20-50-0110-0-10 | 0.79 | 992.0 | -1271.0 | 1359.0 |

Table 4.14: Heuristic results from IPINTs with symmetric costs.

| Instance | CPU (s) | Greedy Obj. | MILP Bound | Easy Bound |
|---|---|---|---|---|
| miblp-10-10-50-0110-0-1 | 0.03 | 208.0 | -99.0 | 208.0 |
| miblp-10-10-50-0110-0-2 | 0.28 | 764.0 | -202.0 | 873.0 |
| miblp-10-10-50-0110-0-3 | 0.03 | 462.0 | -420.0 | 548.0 |
| miblp-10-10-50-0110-0-4 | 0.04 | 276.0 | -365.0 | 444.0 |
| miblp-10-10-50-0110-0-5 | 0.05 | 132.0 | -300.0 | 227.0 |
| miblp-10-10-50-0110-0-6 | 0.05 | 105.0 | -239.0 | 469.0 |
| miblp-10-10-50-0110-0-7 | 0.02 | 0.0 | -104.0 | 147.0 |
| miblp-10-10-50-0110-0-8 | 0.05 | 99.0 | -360.0 | 808.0 |
| miblp-10-10-50-0110-0-9 | 0.04 | 35.0 | -358.0 | 269.0 |
| miblp-10-10-50-0110-0-10 | 0.06 | 0.0 | -277.0 | 290.0 |
| miblp-15-10-50-0110-0-1 | 0.05 | 32.0 | -140.0 | 142.0 |
| miblp-15-10-50-0110-0-2 | 0.01 | 66.0 | -314.0 | 172.0 |
| miblp-15-10-50-0110-0-3 | 0.01 | 228.0 | -123.0 | 261.0 |
| miblp-15-10-50-0110-0-4 | 0.03 | 10.0 | -167.0 | 210.0 |
| miblp-15-10-50-0110-0-5 | 0.13 | 473.0 | -241.0 | 624.0 |
| miblp-15-10-50-0110-0-6 | 0.04 | 6.0 | -336.0 | 305.0 |
| miblp-15-10-50-0110-0-7 | 0.04 | 167.0 | -241.0 | 211.0 |
| miblp-15-10-50-0110-0-8 | 0.11 | 75.0 | -425.0 | 238.0 |
| miblp-15-10-50-0110-0-9 | 0.05 | 0.0 | -419.0 | 149.0 |
| miblp-15-10-50-0110-0-10 | 0.02 | 8.0 | -524.0 | 235.0 |
| miblp-20-20-50-0110-0-1 | 0.26 | 686.0 | -887.0 | 885.0 |
| miblp-20-20-50-0110-0-2 | 0.13 | 276.0 | -478.0 | 302.0 |
| miblp-20-20-50-0110-0-3 | 0.3 | 252.0 | -409.0 | 446.0 |
| miblp-20-20-50-0110-0-4 | 0.6 | 560.0 | -566.0 | 600.0 |
| miblp-20-20-50-0110-0-5 | 0.28 | 323.0 | -376.0 | 585.0 |
| miblp-20-20-50-0110-0-6 | 0.24 | 180.0 | -859.0 | 526.0 |
| miblp-20-20-50-0110-0-7 | 0.64 | 264.0 | -697.0 | 619.0 |
| miblp-20-20-50-0110-0-8 | 0.16 | 512.0 | -349.0 | 571.0 |
| miblp-20-20-50-0110-0-9 | 0.69 | 323.0 | -707.0 | 592.0 |
| miblp-20-20-50-0110-0-10 | 0.8 | 1340.0 | -1271.0 | 1359.0 |

Table 4.15: Heuristic results from IPINTs with random costs.

The effects of interdiction by the exact and heuristic strategies are summarized in Table 4.16. In this table, the columns titled *Avg. Best Effect (%)* and *Avg. Heuristic Effect (%)*, show the average effect on the lower-level problem value that results from the optimal (or best known) and heuristic interdiction strategies. We define the effect of interdiction as the decrease in the optimal lower-level solution that results from variable interdiction, and report it as a percentage of the optimal lower-level objective value.

| | Symmetric | | Random | |
|---|---|---|---|---|
| Class | Avg. Best Effect (%) | Avg. Heuristic Effect (%) | Avg. Best Effect (%) | Avg. Heuristic Effect (%) |
| 1 | 79.15 | 72.23 | 74.16 | 56.02 |
| 2 | 83.87 | 70.59 | 80.57 | 65.51 |
| 3 | 49.20 | 36.82 | 51.12 | 30.63 |

Table 4.16: Summary of heuristic and exact interdiction success.

From the summary presentation, it is clear that the exact solver is able to produce more effective interdiction strategies, as we would expect, and the difference in effectiveness tends to increase with the size of the lower-level ILP. However, we can also see from the table, that the heuristic strategies do still have a significant effect on the lower-level DM's objective value, and provide a reasonable alternative if a solution is needed quickly. The difference in effectiveness is even less dramatic for the case of symmetric interdiction costs. A full comparison between the exact and heuristic methods, for each of the test set variants, is provided in Tables 4.17 and 4.18.

| Instance | Best Known | Greedy Obj. | MILP Bound | Easy Bound | Best Effect (%) | Heuristic Effect (%) |
|---|---|---|---|---|---|---|
| miblp-10-10-50-0110-0-1 | 0 | 44 | -99 | 208 | 100.00 | 78.85 |
| miblp-10-10-50-0110-0-2 | 542 | 764 | -202 | 873 | 37.92 | 12.49 |
| miblp-10-10-50-0110-0-3 | 168 | 257 | -420 | 548 | 69.34 | 53.10 |
| miblp-10-10-50-0110-0-4 | 212 | 225 | -365 | 444 | 52.25 | 49.32 |
| miblp-10-10-50-0110-0-5 | 54 | 54 | -300 | 227 | 76.21 | 76.21 |
| miblp-10-10-50-0110-0-6 | 89 | 105 | -239 | 469 | 81.02 | 77.61 |
| miblp-10-10-50-0110-0-7 | 0 | 0 | -104 | 147 | 100.00 | 100.00 |
| miblp-10-10-50-0110-0-8 | 99 | 99 | -360 | 808 | 87.75 | 87.75 |
| miblp-10-10-50-0110-0-9 | 35 | 35 | -358 | 269 | 86.99 | 86.99 |
| miblp-10-10-50-0110-0-10 | 0 | 0 | -277 | 290 | 100.00 | 100.00 |
| miblp-15-10-50-0110-0-1 | 19 | 26 | -140 | 142 | 86.62 | 81.69 |
| miblp-15-10-50-0110-0-2 | 0 | 66 | -314 | 172 | 100.00 | 61.63 |
| miblp-15-10-50-0110-0-3 | 47 | 228 | -123 | 261 | 81.99 | 12.64 |
| miblp-15-10-50-0110-0-4 | 10 | 10 | -167 | 210 | 95.24 | 95.24 |
| miblp-15-10-50-0110-0-5 | 267 | 301 | -241 | 624 | 57.21 | 51.76 |
| miblp-15-10-50-0110-0-6 | 6 | 6 | -336 | 305 | 98.03 | 98.03 |
| miblp-15-10-50-0110-0-7 | 96 | 127 | -241 | 211 | 54.50 | 39.81 |
| miblp-15-10-50-0110-0-8 | 75 | 75 | -425 | 238 | 68.49 | 68.49 |
| miblp-15-10-50-0110-0-9 | 0 | 0 | -419 | 149 | 100.00 | 100.00 |
| miblp-15-10-50-0110-0-10 | 8 | 8 | -524 | 235 | 96.60 | 96.60 |
| miblp-20-20-50-0110-0-1 | 526 | 526 | -887 | 885 | 40.56 | 40.56 |
| miblp-20-20-50-0110-0-2 | 162 | 276 | -478 | 302 | 46.36 | 8.61 |
| miblp-20-20-50-0110-0-3 | 200 | 252 | -409 | 446 | 55.16 | 43.50 |
| miblp-20-20-50-0110-0-4 | 315 | 560 | -566 | 600 | 47.50 | 6.67 |
| miblp-20-20-50-0110-0-5 | 218 | 288 | -376 | 585 | 62.74 | 50.77 |
| miblp-20-20-50-0110-0-6 | 133 | 180 | -859 | 526 | 74.71 | 65.78 |
| miblp-20-20-50-0110-0-7 | 310 | 310 | -697 | 619 | 49.92 | 49.92 |
| miblp-20-20-50-0110-0-8 | 376 | 398 | -349 | 571 | 34.15 | 30.30 |
| miblp-20-20-50-0110-0-9 | 325 | 325 | -707 | 592 | 45.10 | 45.10 |
| miblp-20-20-50-0110-0-10 | 873 | 992 | -1271 | 1359 | 35.76 | 27.01 |

Table 4.17: Heuristic versus exact results from IPINTs with symmetric costs.

| Instance | Best Known | Greedy Obj. | MILP Bound | Easy Bound | Best Effect (%) | Heuristic Effect (%) |
|---|---|---|---|---|---|---|
| miblp-10-10-50-0110-0-1 | 11 | 208 | -99 | 208 | 94.71 | 0.00 |
| miblp-10-10-50-0110-0-2 | 607 | 764 | -202 | 873 | 30.47 | 12.49 |
| miblp-10-10-50-0110-0-3 | 210 | 462 | -420 | 548 | 61.68 | 15.69 |
| miblp-10-10-50-0110-0-4 | 210 | 276 | -365 | 444 | 52.70 | 37.84 |
| miblp-10-10-50-0110-0-5 | 122 | 132 | -300 | 227 | 46.26 | 41.85 |
| miblp-10-10-50-0110-0-6 | 89 | 105 | -239 | 469 | 81.02 | 77.61 |
| miblp-10-10-50-0110-0-7 | 0 | 0 | -104 | 147 | 100.00 | 100.00 |
| miblp-10-10-50-0110-0-8 | 99 | 99 | -360 | 808 | 87.75 | 87.75 |
| miblp-10-10-50-0110-0-9 | 35 | 35 | -358 | 269 | 86.99 | 86.99 |
| miblp-10-10-50-0110-0-10 | 0 | 0 | -277 | 290 | 100.00 | 100.00 |
| miblp-15-10-50-0110-0-1 | 28 | 32 | -140 | 142 | 80.28 | 77.46 |
| miblp-15-10-50-0110-0-2 | 0 | 66 | -314 | 172 | 100.00 | 61.63 |
| miblp-15-10-50-0110-0-3 | 51 | 228 | -123 | 261 | 80.46 | 12.64 |
| miblp-15-10-50-0110-0-4 | 10 | 10 | -167 | 210 | 95.24 | 95.24 |
| miblp-15-10-50-0110-0-5 | 415 | 473 | -241 | 624 | 33.49 | 24.20 |
| miblp-15-10-50-0110-0-6 | 6 | 6 | -336 | 305 | 98.03 | 98.03 |
| miblp-15-10-50-0110-0-7 | 99 | 167 | -241 | 211 | 53.08 | 20.85 |
| miblp-15-10-50-0110-0-8 | 75 | 75 | -425 | 238 | 68.49 | 68.49 |
| miblp-15-10-50-0110-0-9 | 0 | 0 | -419 | 149 | 100.00 | 100.00 |
| miblp-15-10-50-0110-0-10 | 8 | 8 | -524 | 235 | 96.60 | 96.60 |
| miblp-20-20-50-0110-0-1 | 538 | 686 | -887 | 885 | 39.21 | 22.49 |
| miblp-20-20-50-0110-0-2 | 160 | 276 | -478 | 302 | 47.02 | 8.61 |
| miblp-20-20-50-0110-0-3 | 200 | 252 | -409 | 446 | 55.16 | 43.50 |
| miblp-20-20-50-0110-0-4 | 331 | 560 | -566 | 600 | 44.83 | 6.67 |
| miblp-20-20-50-0110-0-5 | 218 | 323 | -376 | 585 | 62.74 | 44.79 |
| miblp-20-20-50-0110-0-6 | 43 | 180 | -859 | 526 | 91.83 | 65.78 |
| miblp-20-20-50-0110-0-7 | 264 | 264 | -697 | 619 | 57.35 | 57.35 |
| miblp-20-20-50-0110-0-8 | 409 | 512 | -349 | 571 | 28.37 | 10.33 |
| miblp-20-20-50-0110-0-9 | 300 | 323 | -707 | 592 | 49.32 | 45.44 |
| miblp-20-20-50-0110-0-10 | 878 | 1340 | -1271 | 1359 | 35.39 | 1.40 |

Table 4.18: Heuristic versus exact results from IPINTs with random costs.

# Chapter 5

# Conclusions and Research Extensions

In this chapter, we summarize the results and contributions of the dissertation. In addition, we suggest future research directions and suggest how to extend the results here to further field of study. First, however, we describe some interesting application areas, to further motivate the utility of MIBLP in practice.

## 5.1 Applications of Interest

### 5.1.1 Atrial Fibrillation Ablation

Atrial fibrillation (AF) is a form of arrhythmia caused by electrophysiological abnormalities in the heart's electrical conduction system (Finta and Haines, 2004). It is the most prevalent form of arrhythmia, affecting approximately 1% of the population (Waktare, 2002) and is well-known to be a leading cause of stroke.

In a healthy heart, the heartbeat is controlled primarily by the sinoatrial (SA) and atrioventricular (AV) nodes, located in the upper portion of the right atrium and at the intersection of the atria and the ventricles, respectively. Electrical impulses are sent from the SA node, which acts as a natural pacemaker, across the atria via electrical conduction, eventually reaching the AV node. In the AV node, these impulses are delayed for a fraction of a second, then sent across the ventricles, causing contraction and dictating heart rhythm. In AF, impulses originating from sources other than the SA node reach the AV node, causing a more rapid activation pattern of ventricle contraction. Clinical evidence suggests that AF may be the result of impulse cycling within macroreentrant circuits, electrical or physical pathways in the atria, triggered by a source other than the SA node (Finta and Haines, 2004). It has recently been observed that the most likely origin of these auxiliary impulses

is a focal point within the atria. The arrhythmia can be treated by disconnecting this focal point from the rest of the atria (Veenhuyzen et al., 2004).

AF ablation procedures are intended to block these unwanted impulses from reaching the AV node. The most prominent surgical ablation technique is the Cox-Maze procedure (Cox et al., 1991). This procedure, although known to be extremely effective (Gillinov and McCarthy, 2004) in the treatment of AF, requires complex intrusive surgery and cardiac arrest to complete. An alternative procedure is catheter ablation, which does not require opening the heart or surgically incising the patient. Instead, disconnection from the AV is accomplished by transmitting energy (frequently radio frequency (RF)) to appropriate locations via catheter insertion. In either ablation procedure, the treatment of AF requires the disconnection of auxiliary pathways from the AV node. If we assume that the electrical impulses are traveling via the path of lowest resistance (or energy), we can model their flow using a mathematical program. Then, the optimal strategy for disconnection is determined by the solution of an interdiction problem whose lower-level is defined by this program. Further research is necessary to determine if this is a realistic model with which to guide ablation surgery. However, even a simple model may yield valuable information in AF treatment.

### 5.1.2 Corporate Strategy

A straightforward application of bilevel programming is the analysis of decentralized decision making within a large company. Although it is likely that the each level of hierarchy within the company recognizes the benefit of maximizing the overall health of the company, it is certainly plausible that different levels have different notions of the measurement of health. Additionally, it is easy to imagine situations where individual components of the company are myopic, in the sense that their primary goals may reflect the betterment of their division, without due consideration of the effects on the company as a whole. In this case, it is in the company's best interest to realize these possibilities and make decisions accordingly. Thus, at the highest level, strategies that consider the behavior of lower-level decision makers should be considered. Bilevel programming is well suited for this type of analysis. Of course, as is the case when one compares applications of linear and integer programming (see Nemhauser and Wolsey (1999) for examples), using the more general model of mixed integer bilevel linear programming yields more applicability.

The inherent hierarchical structure is readily apparent in the intra-company model described above. However, in some applications, the underlying hierarchical structure is not as obvious. One example of such an application arises when two firms compete for economic market share. In particular, a decision hierarchy results in analysis of markets dominated by a large entity, or "market-maker." In this case, the larger of the two companies has the power to exhibit influence over the other because of its dominance and ability to make decisions which affect the market itself. Thus, a

hierarchy exists due to the relative influence of the companies, rather than the corporate structure of a particular company, as in the application described above. Economic analysis was the original motivation for the study of multilevel programming (Koopmans, 1951; Charnes et al., 1967; Cyert and March, 1955). However, we describe a different economic application than those considered in the traditional economic literature.

Suppose the larger company (Company A) wishes to gain a controlling interest in the smaller company (Company B). Presumably, the lower the value of the Company B (as measured by profit or stock price, or any number of valuing techniques), the easier this goal will be to obtain. Of course, Company B would like this value to be as high as possible, to ensure the future health of the company. This leads to a bilevel optimization problem where Company A seeks to minimize the value of Company B, while Company B seeks to maximize its own value. If we assume that both companies value Company B in the same manner, we have zero-sum problem. Alternatively, we may consider a more general case where the companies have conflicting, but not necessarily opposite, objectives, yielding a non-zero-sum model.

We have described a general application of bilevel optimization above. We now suggest a particular setting in which to apply the general ideas. Suppose Company B wishes to determine its marketing strategy for the upcoming fiscal year. Specifically, suppose Company B is deciding which demographic or geographic regions to target, subject to a specified marketing budget $C$. We assume that there exist a finite number $N$ of potential regions available to Company B. We also assume that there exists a cost $c_i$ to establish a marketing campaign in region $i$ and that there is a benefit $p_i$ for marketing the company's products in region $i$. Let

$$y_i = \begin{cases} 1 & \text{if region } i \text{ is chosen for the campaign} \\ 0 & \text{otherwise} \end{cases}.$$

Then, Company B solves an integer program where it seeks to maximize the marketing benefit $\sum_{i=1}^{N} p_i y_i$ subject to the budget constraint $\sum_{i=1}^{N} c_i y_i \leq C$. Now, suppose that, due its market dominance, if Company A targets the same region as Company B, Company B is unable to establish a worthwhile marketing campaign. Then, Company A can interdict the marketing problem to be solved by Company B. Assuming that Company A also has some budget $D$ available for the

disruption of Company B's strategy and that interdicting region $i$ has cost $d_i$, we have the MIPINT

$$\begin{aligned} \min_{x\in\mathbb{B}^N} \max_{y\in\mathbb{B}^N} \quad & \sum_{i=1}^{N} p_i y_i \\ \text{subject to} \quad & \sum_{i=1}^{N} d_i x_i \leq D \\ & c_i y_i \leq C \\ & y_i \leq 1 - x_i, \quad i = 1, \dots N, \end{aligned}$$

where

$$x_i = \begin{cases} 1 & \text{if region } i \text{ is interdicted} \\ 0 & \text{otherwise} \end{cases}$$

In this simple model, each company is constrained by a single knapsack budget constraint. Of course, we can add additional constraints to make the model more realistic. Also, we can easily drop the assumption that both companies value the marketing benefits of each region identically and introduce separate cost vectors for A and B. This yields a non-zero-sum MIBLP that resembles a MIPINT in its system of constraints.

### 5.1.3 Wireless MANET

Another interesting application of multilevel programming arises in cross-layer network design optimization problems. These problems are encountered in mobile ad hoc networks (MANET) consisting of moving nodes, each equipped with cognitive radios that dynamically adjust their transmission power and constellation size in response to channel and interference states. One example of such a network exists in the military, where the mobile nodes represent foot soldiers. In this type of network, the objective is to utilize the minimum amount of transmission power in the network's physical layer, while maximizing the capacity of the links among the nodes, thereby throughput, in the network layer. If this can be achieved in all radios, then the maximum amount of throughput can be attained at the network layer, yielding the greatest amount of communication among the radios.

In this section, we discuss previous attempts at modeling and analyzing this system and motivate the introduction of several new models which allow for further analysis. The models introduced here incorporate the relevant aspects of the previous models, while generalizing the modeling framework in order to provide a more flexible framework and an alternative solution approaches.

**Previous Models.** In the past, the problems of finding the optimal cross-layer network design with respect to network throughput for a mobile ad-hoc wireless network and determining the minimum necessary transmit power in the network's physical layer have been considered as separate optimization problems. In this section, we consider the traditional models, separately, then reformulate the problem using the frameworks of multicriteria and multilevel programming.

**Design Problem Description.** A wireless MANET is composed of five layers: physical , medium access control (MAC), network, transport, and application. We describe joint optimization models across the first three layers. Our intention is to demonstrate methods for combining the cross layer design model of Fridman et al. (2008) and an ILP that determines minimum transmission power for a fixed capacity graph and set of constellation sizes. In the following sections, we discuss each of these and their roles separately. Then, in Section 5.1.3, we describe an optimization model that has been previously used to determine the optimal network design with respect to network throughput.

**Physical Layer.** The primary functions of the physical layer of a MANET are to control transmission power and constellation size. In this layer, there exist a set of mobile nodes, each equipped with a cognitive radio permitting dynamic selection of both transmit power and constellation size. Let $N = \{1, \ldots, n\}$ denote the set of mobile nodes in the network. Let $p^t \in \mathcal{P}$ denote the power vector, where $p_i^t$ is the power of node $i \in N$ at time $t$, and $\mathcal{P}$ is a finite discrete set. We also denote the constellation vector by $m^t \in \mathcal{M}$, where $m_i^t$ is the constellation size of node $i$ at time $t$, and $\mathcal{M}$ is also a finite discrete set.[1] At any time $t$, a subset of the mobile nodes are transmitting. We denote this subset by $\tau^t$. At time $t$, the *Signal to Interference plus Noise Ratio* (SINR) for node $j \in N$, when listening to node $i \in N \setminus \{i\}$, is given by

$$SINR_j^t = \frac{p_i^t d_{ij}^{-\alpha}}{\sum_{k \in \tau^t \setminus \{i\}} p_j^t d_{kj}^{-\alpha} + \sigma^2}, \tag{5.1}$$

where $d_{ij}$ is the distance between node $i$ and node $j$, $\alpha > 2$ is the path-loss constant, and the constant $\sigma^2$ is the additive Gaussian noise to which the channel is subject. We also define the *Bit Error Rate* (BER) for each receiver:

$$BER_j^t = 2Q\left(\sqrt{2SINR_j^t}\sin\frac{\pi}{m_i^t}\right). \tag{5.2}$$

[1] For example, in the case of the well-known modulation scheme, Quadrature Amplitude Modulation (QAM), $\mathcal{M} = \{1, 2, 4, \ldots, m_{\max}\}$, where $m_{\max}$ is the maximum constellation size (Fridman et al., 2008).

In equation (5.2), $Q$ is the *Q Function*, $Q(Z) = P\{Z > z\}$ for $Z \sim N(0,1)$, which is estimated as:

$$Q(z^t) \approx \frac{1}{\frac{1}{2}z^t + \frac{1}{2}\sqrt{z^{t^2} + \frac{8}{\pi}}} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^{t^2}}{2}}, \tag{5.3}$$

where

$$z^t = \sqrt{2SINR_j^t} \sin \frac{\pi}{m_i^t}.$$

The maximum allowable BER on any link is given by $\beta$.

The cognitive radios used in MANETs can change both transmission power and constellation size. Increasing the constellation size $m_i$:

- increases the capacity of links emanating from node $i$,
- increases the chance of symbol decoding error (BER), and
- has no effect on the neighboring nodes (other than link capacity).

On the other hand, increasing the transmission power $p_i$:

- increases the ratio of ratio of signal to noise (SINR) for transmitter $i$, and
- decreases the SINR for all other receivers.

Thus, these two functions can be used in a complementary manner to improve network performance. This relationship motivates our study of multiobjective and multilevel models for this application.

**MAC Layer.** The primary purpose of the MAC layer is to determine the optimal scheduling for data transmission. Here, we assume the network uses a slotted protocol. In each slot, a mobile node can be transmitting, receiving, or idle. Further, for each time slot, there exists an associated power vector that identifies the available resources for the node set. Let $S$ denote the set of time slots for each round of scheduling. The duration of each time slot is given by the constant $\eta$, and each transmitter $i \in N$ is allowed to transmit in at most $s_i$ of the time slots. In this context, a schedule is defined by $|S|$ different $N \times N$ matrices, $B^1 \ldots, B^s$, where

$$B_{ij}^t = \begin{cases} 1 & \text{if node } i \in N \text{ transmits to node } j \in N \text{ in slot } t \in S \\ 0 & \text{otherwise,} \end{cases}$$

and a feasible schedule is one in which each node $i \in N$ transmits in at least one time slot $t \in S$. Note that, if $B_{ij}^t = 1$, then $BER_j^t < \beta$. That is, the schedule, by construction, will satisfy the BER

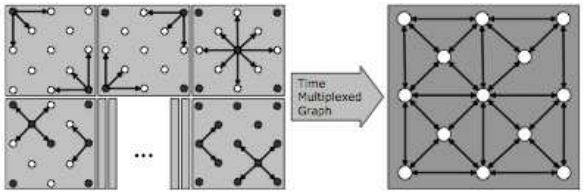



Figure 5.1: Illustrating the capacity graph construction. (Fridman et al., 2008)

constraint for each attempted reception (Fridman et al., 2008). The nominal capacity of transmitter $i$ is:

$$c_i = \sum_{t \in S} \log_2 m_i^t.$$

But, since node $i$ can transmit in at most $s_i$ time slots, the *effective capacity* of link $(i, j)$ is given by:

$$\tilde{c}_{ij} = \frac{s_i}{|S|} \sum_{t \in S} \log_2 m_i^t. \tag{5.4}$$

The capacity for a network link, by its transmitting, or source, node $i$.

It is important to note that, in our formulation, $B$ is not a free variable and is determined before the optimization process begins. This is consistent with the work of Fridman et al. (2008), where the random packing heuristic of Wu et al. (2005) is employed. As noted in Fridman et al. (2008), each time slot yields a disconnected graph on $N$ nodes, where links exist between designated transmission nodes and their potential receivers. The network utilizes successive relaying to transmit packets from the desired source to sink.

In order to combine the disconnected graphs determine by the schedule $B$, we create a *capacity graph* $G = (V, E)$ as follows. There exists a vertex $i \in V$ for each mobile node with a positive effective capacity. Then, for each vertex pair $i, j \in V \times V$, there exists an edge $(i, j)$ if and only if $i$ transmits to $j$ in at least one time slot. Formally,

$$V = \{i \in N \mid \tilde{c}_{ij} > 0, \text{ for some } j \in N \setminus \{i\}\}$$
$$E = \{(i, j) \in V \times V \mid B_{ij}^t = 1, \text{for some } t \in S\}.$$

Thus, an edge exists in $G$ if it carries transmission in one or more times slots. For each edge $e \in E$, we assign the capacity $\tilde{c}_{ij}$. This process is illustrated in Figure 5.1. $G$ defines the available resources for links in the networks physical layer, and we can send a packet from a node $i$ to a node $j$ if there exists an $(i, j)-$path in $G$. It is important to note that, although $G$ is directed, in most cases, we have both edges $(i, j)$ and $(j, i)$, each with an associated capacity.

**Network Layer.** The network layer is responsible for managing the transmission of data packets between a specified source and destination. We consider the flow of several commodities across the network layer, each with a different source and sink. Let $K = \{1, \ldots, k\}$ be the set of commodities to be sent across the network. For each commodity, $k \in K$, let $\sigma_k \in N$ and $\delta_k \in N$ be the source and sink node, respectively. Let $x_e^k \geq 0$ be the flow of commodity $k \in K$ over edge $e \in E$. The sum of the flow over all edges which terminate at the sink node for commodity $k$,

$$\sum_{i \in N} x_{i,\delta_k}^k = f_k,$$

yields the total *throughput* $f_k$ of the commodity.

**Multicommodity Maximum Throughput Design Problem (MMTP).** The overall objective of the design problem is maximize the total amount of commodity sent over the network. In other words we wish to maximize the sum of all commodity flows,

$$F(f) = \sum_{k \in K} \sum_{i \in N} x_{i,\delta_k}^k. \tag{5.5}$$

For each node pair $(i, j)$, we have the capacity constraint

$$\sum_{k \in K} x_{ij}^k \leq \tilde{c}_{ij} = \frac{s_i}{|S|} \sum_{t \in S} \log_2 m_i^t, \tag{5.6}$$

which states that we cannot send more flow over any edge than the edge's capacity will allow, where $\tilde{c}_{ij}$ is given by equation (5.4). Additionally, as mentioned previously, whenever a node is receiving data, it is required to satisfy the bound on maximum allowable BER. Thus, we introduce the constraint

$$B_{ij}^t \cdot BER_j^t \leq \beta, \quad \forall i, j \in N, t \in S, \tag{5.7}$$

which states that if any node $i$ is transmitting to node $j$ in some slot $t$ (i.e. $B_{ij}^t = 1$), then the BER for node $j$ must not exceed $\beta$. In order to make sure the model is well-defined, we must add the standard *flow conservation* constraints,

$$\sum_{j \in N} x_{ji}^k = \sum_{j \in N} x_{ij}^k, \quad \forall i \in N \setminus \{\sigma_k, \delta_k\}, k \in K, \tag{5.8}$$

which require the flow out of each node to be equal to the flow into that node, for all commodities on the network. This yields the MMTP:

$$\begin{aligned}
\max \quad & \sum_{k=1}^{K} \sum_{i \in N} x_{i,\delta_k}^k \\
\text{subject to} \quad & \sum_{k \in K} x_{ij}^k \leq \frac{s_i}{|S|} \sum_{t \in S} \log_2 m_i^t, \quad \forall i, j \in N \\
& \sum_{j \in N} x_{ji}^k = \sum_{j \in N} x_{ij}^k, \quad \forall i \in N \setminus \{\sigma_k, \delta_k\}, k \in K \\
& B_{ij}^t \cdot \text{BER}_j^t \leq \beta, \quad \forall i, j \in N, t \in S \\
& p \in \mathcal{P}, m \in \mathcal{M}, x \geq 0,
\end{aligned} \tag{MMTP}$$

As modeled, (MMTP) is a nonlinear mixed integer program. To highlight the relationship among the layers, we note that, in the preceding model, the link capacity constraints depend on the underlying temporal schedule $B$. In turn, the feasibility of the underlying temporal schedule depends on the physical layer power vector $p$ and the constellation size vector $m$. In the following section, we discuss another model, which determines the minimum necessary transmit power on the physical level and describe its relationship to (MMTP).

**Transmit Power Problem Description.** As mentioned above, when designing a wireless network, one must also determine the amount of transmission power to allocate to the links in the physical layer. Traditionally, this problem has been treated as a separate, unrelated problem from that described in Section 5.1.3. However, it is clear that the problems are, in fact, quite related, since they are modeling different aspects of the same network and both need to determine a power strategy $p$. In this section, we describe the integer linear program (ILP) that has been used previously to determine power allocation. In Section 5.1.3, we describe alternative models which combine the other two models.

In this section, we use the same definition of the power vector $p^t$ as in Section 5.1.3. The goal of the power allocation problem is to minimize the total transmit power,

$$\sum_{i \in N} p_i^t,$$

for each time slot $t \in S$. Let $\gamma$ denote a power threshold below which a link cannot communicate effectively. That is, $\gamma$ is the minimum QoS requirement for a channel. Thus, in order for the network to operate effectively, we must have

$$SINR_j^t \geq \gamma, j \in N. \tag{5.9}$$

So, we can model the transmit power problem (TPP) for time $t \in S$ as the (ILP):

$$\begin{aligned} \min \quad & \sum_{i \in N} p_i^t \\ \text{subject to} \quad & \frac{p_i^t d_{ij}^{-\alpha}}{\sum_{k \in \tau^t \setminus \{i\}} p_k^t d_{kj}^{-\alpha} + \sigma^2} \geq \gamma, i, j \in N, \\ & p \in \mathcal{P}. \end{aligned} \tag{TPP$_t$}$$

**Combined Models.** In this cross layer design problem there are, in fact, several performance measures of importance in addition to total throughput:

- The total network capacity
$$\sum_{i,j \in N} \tilde{c}_{ij} \tag{5.10}$$
- The total transmission power
$$\sum_{t \in S} \sum_{i \in N} p_i^t \tag{5.11}$$
- Sum of the node constellation sizes
$$\sum_{t \in S} \sum_{i \in N} m_i^t \tag{5.12}$$

In what follows, we demonstrate how to incorporate objective (5.11) into the optimization model. We consider two alternative methods of incorporating this objective in the network design problem. The new models differ in the way in which we define decision-making authority. In Section 5.1.3, we present a biobjective integer nonlinear framework for the network design. Then, in Section 5.1.3, we introduce a mixed integer bilevel nonlinear programming model that provides an alternative view of the problem.

**A Biobjective Integer Programming Model.** One way in which we can combine the models given in Sections 5.1.3 and 5.1.3 is to employ the biobjective integer programming framework discussed in previous chapters. This class of models enables us to study the tradeoffs between two conflicting objectives by a single DM.

Applying the biobjective framework to our design problem means combining the constraints of the separate problems, and forming an objective function that incorporates the goal of minimizing

transmission power, while providing the maximum network performance. This yields the *biobjective design problem* (BODP):

$$\begin{aligned}
\text{vmax} \quad & \left[\sum_{k=1}^{K}\sum_{i\in N} x^k_{i,\delta_k}, \sum_{t\in S}\sum_{i\in N} p^t_i\right] \\
\text{subject to} \quad & \sum_{k\in K} x^k_{ij} \le \frac{s_i}{|S|}\sum_{t\in S}\log_2 m^t_i, \quad \forall i,j \in N \\
& \sum_{j\in N} x^k_{ji} = \sum_{j\in N} x^k_{ij}, \quad \forall i \in N\setminus\{\sigma_k,\delta_k\}, k\in K, \\
& B^t_{ij}\cdot \text{BER}^t_j \le \beta, \quad \forall i,j\in N, t\in S \\
& p\in\mathcal{P}, m\in\mathcal{M}, x\ge 0,
\end{aligned} \tag{BODP}$$

for fixed schedule $B$. Note that, in (BODP), the second objective minimizes total transmission power across all time slots, in contrast to the objective in ($\text{TPP}_t$), which considers each slot individually. In addition, we have removed the constraint

$$SINR^t_j \ge \gamma, j\in N,$$

since we assume the network will operate effectively if the BER constraint, which is dependent on SINR, is satisfied. This model can be solved using an algorithm similar to Algorithm 3.3.

**A Mixed Integer Bilevel Programming Model** Another way to combine the previous models is to introduce a second DM. As with many mathematical programs, the models described above are still limited by their assumption of a centralized decision-making structure. However, in this application, since decisions are made at different times, and potentially in different geographic locations, it is likely that multiple DMs will be involved. For example, suppose that we wish to control the network flow and constellation size at a central command unit. This may be the case if one DM controls the flows for several subunits, each using a MANET. It is reasonable to assume that this DM would also control the constellation sizes, since they have a direct effect on network capacity. In this scenario, we assume that the internal algorithms installed in the mobile nodes determine optimal transmission power for each node and each time slot, given a constellation size. Although the central DM's primary objective is total throughput, it is reasonable that total power consumed weighs into his decisions, was well. In fact, there is most likely some cost, known to the central DM but not the radios, associated with each unit of power consumed. If we let $\theta^t_i$ denote the unit cost of power at node $i$ at time $t$, then we can construct the central objective function:

$$\sum_{k=1}^{K}\sum_{i\in N} x^k_{i,\delta_k} - \sum_{t\in S}\sum_{i\in N}\theta^t_i p^t_i,$$

which maximizes the sum of total throughput and negative transmission cost. This leads us to a natural bilevel program:

$$
\begin{aligned}
\max_{x\geq 0, m\in\mathcal{M}} \quad & \sum_{k=1}^{K}\sum_{i\in N} x^k_{i,\delta_k} - \sum_{t\in S}\sum_{i\in N}\theta^t_i p^t_i \\
\text{subject to} \quad & \sum_{k\in K} x^k_{ij} \leq \frac{s_i}{|S|}\sum_{t\in S}\log_2 m^t_i, \quad \forall i,j\in N \\
& \sum_{j\in N} x^k_{ji} = \sum_{j\in N} x^k_{ij}, \quad \forall i\in N, k\in K, \\
& p^t \in \operatorname{argmin}_{p^t\in\mathcal{P}} \left\{ \sum_{i\in N} p^t_i : B^t_{ij}\cdot \mathrm{BER}^t_j \leq \beta, \quad \forall i,j\in N \right\}, \forall t\in S,
\end{aligned}
\tag{BLDPMF}
$$

for fixed schedule $B$. (BLDP) is a *mixed integer nonlinear bilevel program* with multiple followers. Aside from its nonlinearity, this multilevel model differs from those described previously because multiple DMs (i.e. radios) exist at the second level. However, it is show in Calvete and Galé (2007) that (BLDPMF) is equivalent to the *bilevel design problem*

$$
\begin{aligned}
\max_{x\geq 0, m\in\mathcal{M}} \quad & \sum_{k=1}^{K}\sum_{i\in N} x^k_{i,\delta_k} - \sum_{t\in S}\sum_{i\in N}\theta^t_i p^t_i \\
\text{subject to} \quad & \sum_{k\in K} x^k_{ij} \leq \frac{s_i}{|S|}\sum_{t\in S}\log_2 m^t_i, \quad \forall i,j\in N \\
& \sum_{j\in N} x^k_{ji} = \sum_{j\in N} x^k_{ij}, \quad \forall i\in N, k\in K, \\
& (p^1,\ldots,p^{|S|}) \in \operatorname{argmin}_{p^t\in\mathcal{P}} \left\{ \sum_{t\in S}\sum_{i\in N} p^t_i : B^t_{ij}\cdot \mathrm{BER}^t_j \leq \beta, \quad \forall i,j\in N \right\},
\end{aligned}
\tag{BLDP}
$$

since each follower's problem contains only upper-level variables and $p^t$, and the objective functions are defined by linear functions. In this situation, we refer to the followers as *independent*. While, in practice, all decisions may be made during deployment, this model gives us a way to predict the overall performance of the network ahead of time.

**Solution Methodology** With the exception of the ILP described in Section 5.1.3, each of the models described in the previous sections contains nonlinear functions in its constraints set. Nonlinear programming models already present a difficult class of problems, due to the possibility of multiple local minima (see e.g. Bazaraa et al. (1979). When combined with integrality restrictions on the decision variables and modeled as bilevel programs, these models present a significant computational challenge. We have described exact algorithms for mixed integer bilevel linear programs and pure integer bilevel linear programs. However, to our knowledge, no exact algorithms exist

for the nonlinear bilevel models of the form (BLDP). Thus, in order to solve these models, we must either remove the nonlinear functions, using approximations or relaxations, or generalize the algorithms developed for the linear case to be applied to the current framework. Alternatively, we can use the *Efficient Solution* heuristic method described in Chapter 3, since it does not depend on linearity of the constraints.

## 5.2 Conclusions and Suggested Future Work

In this dissertation, we have discussed the wide applicability of multilevel programming, motivating the study of these models through applications in homeland security, production planning, economic market analysis, and algorithm design. We have demonstrated areas in which bilevel linear programs have made significant contribution by allowing assumptions of a single decision-maker to be relaxed. However, it has also been argued that these models continue to limit the true utility of bilevel programming, by constricting its application to those systems for which continuous lower-level models are appropriate. Thus, the further development of methods for solving models with discrete variables is essential if we wish to fully realize the benefits of bilevel models.

We have also demonstrated the inherent challenges associated with solving mixed integer bilevel linear programming problems. It is clear that this is a very difficult class of problems, for which algorithmic development is not straightforward. However, leveraging the recent advancements in large-scale integer programming and integer programming duality, we have made some progress towards the development of an algorithmic framework which can handle these types of problems. In particular, we have described a theoretical and methodological groundwork of algorithms for solving MIBLPs directly. We discussed a generalization of the well-known branch-and-cut algorithm used for solving integer programs. By expanding our notion of feasibility, we demonstrated that the methods are analogous to those used for integer programs, but require cutting planes which encapsulate the lower-level optimality conditions. In the case of pure integer bilevel programs, a simple argument provides one such class of cuts.

By leveraging the newly-developed extensions to LP duality theory, we have shown how to derive single-level integer programming reformulations of the problem, several of which are analogous to those used to derive reformulations for the continuous problem, For these cases, we have used the relationship between linear and integer program to illustrate these similarities. For some special cases of MIBLP, the reformulation methods yield problems that can be solved by known methods, but reformulations for the general case lead to problems for which no direct methods are known. However, using information obtained during our standard bilevel feasibility check, we have shown how one can derive iterative approximation methods for the lower-level value function and derived

theoretical algorithms based on this approach. In order for these method to have true practical utility, however, development of effective methods for the resulting subproblems is essential.

The primary advantage of our approaches is the ability to exploit the vast array of existing technology for solving integer programs. There are several important research directions stemming from each of these approaches. Certainly, the branch-and-cut algorithm we have developed would benefit greatly from additional classes of cutting planes, especially if they utilize information contained in the lower-level value function or optimality conditions. Further, the use of value function approximations appears to be a promising area of future work, and different methods for obtaining approximations will likely lead to significant computational improvements.

From an application perspective, our primary focus has been on problems in infrastructure protection. In particular, we have derived methods for solving the Steiner arborescence problem that arises in the design of a particular early warning system used to monitor a Swiss urban water network. Then, using this application as an example, we have described one way in which interdiction problems can be used for sensitivity analysis, and provided several problem-specific methods for the mixed integer interdiction problem.

To our knowledge, no integer bilevel programming solvers are available to the mathematical programming community. Thus, one of the main contributions of this research has been the development a bilevel programming solver package to be made available through the COIN-OR repository. The design of the solver is such that future researchers can easily add additional cutting planes, branching methods, heuristics, and preprocessing methods with minimal effort. We hope this framework will benefit the research community and spur computational experimentation on and methodological development for integer bilevel programs. The current version of the solver package contains the branch-and-bound method for IBLP, as well as the customized features derived for interdiction problems. The purpose of this customization is meant to demonstrate the way in which users can employ enhancements based on problem structure to improve the algorithm's effectiveness. There is large amount of work to be done towards the development of a complete bilevel programming solver. The implementation of other known methods, for both continuous and discrete problems, represents a significant effort in itself. In addition, further customized implementations should be explored for those problems with a wide array of applications.

# Appendix A

# List of Acronyms

| | | |
|---|---|---|
| AF | – | Atrial Fibrillation |
| BER | – | Bit Error Rate |
| BIBLP | – | Binary integer bilevel linear program(ming) |
| BLP | – | Bilevel linear program(ming) |
| BMIBLP | – | Biobjective mixed integer bilevel linear program |
| BMILP | – | Biobjective mixed integer linear program |
| BMIPINT | – | Biobjective mixed integer interdiction |
| CP | – | Current problem, from a specialized branch–and–cut algorithm |
| DBLP | – | Decision version of bilevel linear programming |
| DKNAP | – | Decision version of the knapsack problem |
| DM | – | Decision-maker |
| DMIBLP | – | Decision version of mixed integer bilevel linear programming |
| DMILP | – | Decision version of mixed integer linear programming |
| DMIPINT | – | Decision version of mixed integer programming interdiction |
| EWS | – | Early warning system |
| GFCPA | – | Gomory Fractional Cutting Plane Algorithm |
| IBLP | – | Integer bilevel linear program(ming) |
| ILP | – | Integer linear program(ming) |
| IP | – | Integer program(ming) |
| KKT | – | Karush-Kuhn-Tucker |
| LMM | – | Linear max-min problem |
| LP | – | Linear program(ming) |
| LPEC | – | Linear program with equilibrium constraints |
| MAC | – | Medium access control |

| | | |
|---|---|---|
| MANET | – | Mobile ad hoc network |
| MIBLP | – | Mixed integer bilevel linear program(ming) |
| MIBNP | – | Mixed integer bilevel nonlinear program(ming) |
| MILP | – | Mixed integer linear program(ming) |
| MINLP | – | Mixed integer nonlinear program(ming) |
| MIPINT | – | Mixed integer programming interdiction |
| MP | – | Mathematical program(ming) |
| MPEC | – | Mathematical program with equilibrium constraints |
| MSPP | – | Maximum Shortest Path Problem |
| PCSA | – | Prize–collecting Steiner arborescence |
| RHS | – | Right-hand-side |
| SINR | – | Signal to Interference plus Noise Ratio |

Table A.1: List of acronyms used in this dissertation.

# Bibliography


Achterberg, T., T. Koch, and A. Martin 2005. Branching rules revisited. *Operations Research Letters* **33**(1), 42–54.

Anandalingam, G. and T. Friesz 1992. Hierarchical optimization: An introduction. *Annals of Operations Research* **34**, 1–11.

Atamturk, A., G. Nemhauser, and M. W. P. Savelsbergh 1995. A combined lagrangian, linear programming and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics* **1**, 247–259.

Audet, C., J. Haddard, and G. Savard 2007. Disjunctive cuts for continuous linear bilevel programming. *Optimization Letters* **1**(3), 259–267.

Audet, C., P. Hansen, B. Jaumard, and G. Savard 1997. Links between linear bilevel and mixed 0-1 programming problems. *Journal of Optimization Theory and Applications* **93**(2), 273–300.

Avis, D., D. Bremner, and R. Seidel 1997. How good are convex hull algorithms. *Computational Geometry: Theory and Applications* **7**, 265–301.

Avis, D. and K. Fukuda 1992. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Computational Geometry* **8**, 295–313.

Başar, T. and G. J. Olsder 1999. *Dynamic Noncooperative Game Theory*. SIAM.

Balas, E. 1979. Disjunctive programming. *Annals of Discrete Mathematics* **5**, 3–51.

Balas, E., S. Ceria, and G. Cornuéjols 1993. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming* **58**, 295324.

— 1996. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science* **42**, 12291246.

Balas, E. and M. Perregaard 2003. A precise correspondence between lift-and-project cuts, simple disjunctive cuts and mixed integer gomory cuts for 0-1 programming. *Mathematical Programming B* **94**, 221–245.

Bard, J. 1983. An algorithm for solving the general bilevel programming problem. *Mathematics of Operations Research* **8**(2), 260–272.

— 1984a. An investigation of the linear three level programming problem. *IEEE Transactions on Systems, Man, and Cybernetics* **14**, 711–717.

— 1984b. Optimality conditions for the bilevel programming problem. *Naval Research Logistics Quarterly* **31**, 13–26.

— 1988. Convex two-level optimization. *Mathematical Programming* **40**, 15–27.

Bard, J. and J. Falk 1982. An explicit solution to the multi-level programming problem. *Computers and Operations Research* **9**(1), 77–100.

Bard, J. and J. Moore 1990. A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing* **11**(2), 281–292.

— 1992. An algorithm for the discrete bilevel programming problem. *Naval Research Logistics* **39**, 419–435.

Bard, J., J. Plummer, and J. Sourie 1998. Determining tax credits for converting nonfood crops to biofuels: An application of bilevel programming. In A. Migdalas, P. Pardalos, and P. Värbrand, editors, *Multilevel Optimization: Algorithms and Applications*, pages 23–50. Kluwer Academic Publishers.

— 2000. A bilevel programming approach to determining tax credits for biofuel production. *European Journal of Operational Research* **120**, 30–46.

Bazaraa, M., H. Sherali, and C. Shetty 1979. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, Inc.

Beltramo, M. 1983. *Dual Production Sources in the Procurement of Weapon Systems: A Policy Analysis*. Ph.D. thesis, RAND Graduate Institute.

Ben-Ayed, O. and C. Blair 1990. Computational difficulties of bilevel linear programming. *Operations Research* **38**, 556–560.

Benson, H. 1989. On the structure and properties of a linear multilevel programming problem. *Journal of Optimization Theory and Applications* **60**(3), 353–373.

Benson, H., A. Sen, D. Shanno, and R. Vanderbei 2006. Interior-point algorithms, penalty methods and equilibrium problems. *Computational Optimization and Applications* **34**(2), 155–182.

Berry, J., L. Fleischer, W. Hart, C. Phillips, and J. Watson 2005. Sensor placement in municipal water networks. *Journal of Water Resources Planning and Management* **131**(3), 237–243.

Berry, J., W. Hart, C. Phillips, J. Uber, and J. Watson 2006a. Sensor placement in municipal water networks with temporal integer programming models. *Journal of Water Resources Planning and Management* **132**(4), 218–224.

Berry, J., W. Hart, C. Phillips, and J. Watson 2006b. A facility location approach to sensor placement optimization. In *8th Annual Symposium on Water Distribution Systems Analysis*. Cincinnati, OH.

Bertsimas, D. and J. Tsitsiklis 1997. *Introduction to Linear Optimization*. Athena Scientific.

Bialas, W. and M. Karwan 1982. On two-level optimization. *IEEE Transactions of Automatic Control* **AC-27**(1), 211–214.

Bienstock, D. and A. Verma 2008. The n-k problem in power grids: New models, formulations and computation. Available at `http://www.columbia.edu/~dano/papers/nmk.pdf`.

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

Blair, C. 1995. A closed-form representation of mixed-integer program value functions. *Mathematical Programming* **71**, 127–136.

Blair, C. and R. Jeroslow 1977. The value function of a mixed integer program: I. *Discrete Mathematics* **19**, 121–138.

— 1982. The value function of an integer program. *Mathematical Programming* **23**, 237–273.

Bracken, J. and J. McGill 1973. Mathematical programs with optimization problems in the constraints. *Operations Research* **21**, 37–44.

— 1974a. Defense applications of mathematical programs with optimization problems in the constraints. *Operations Research* **22**, 1086–1096.

— 1974b. A method for solving mathematical programs with nonlinear programs in the constraints. *Operations Research* **22**, 1097–1101.

Brotcorne, L., M. Labbè, P. Marcotte, and G. Savard 2000. A bilevel model and solution algorithm for a freight tariff setting problem. *Transportation Science* **34**, 289–302.

Brown, C., W. Harney, R. Skroch, and R. Wood 2009. Interdicting a nuclear weapons project. *Operations Research* **57**, 866–877.

Burgard, A., P. Pharkya, and C. Maranas 2003. Optknock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and Bioengineering* **84**, 647–657.

Bussieck, M. and M. Lübbecke 1998. The vertex set of a 0/1 polytope is strongly p-enumerable. In *Computational Geometry*, pages 103–109.

Byrd, R., M. Hribar, and J. Nocedal 1999. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization* **9**(4), 877–900.

Calvete, H. and C. Galé 2007. Linear bilevel multi-follower programming with independent followers. *Journal of Global Optimization* pages 409–417.

Candler, W. and R. Townsley 1982. A linear two-level programming problem. *Computers and Operations Research* **9**, 59–76.

Caroe, C. and J. Tind 1998. L-shaped decomposition of two-stage stochastic integer programs with integer recourse. *Mathematical Programming* **83**, 451–464.

Carr, R. D., H. J. Greenberg, W. E. Hart, G. Konjevod, E. Lauer, H. Lin, T. Morrison, and C. A. Phillips 2006. Robust optimization of contaminant sensor placement for community water systems. *Mathematical Programming* **107**(1-2), 337–356.

Charnes, A., R. Clower, and K. Kortanek 1967. Effective control through coherent decentralization with preemptive goals. *Econometrica* **35**(2), 294–319.

Chen, L. and D. Goldfarb 2007. An active-set method for mathematical programs with linear complementarity constraints. Technical report, Columbia University.

Chen, Y. and M. Florian 1992. On the geometry structure of linear bilevel programs: A dual approach. Technical Report CRT-867, Centre de Recherche sur les Transports.

Chen, Y., M. Florian, and S. Wu 1992. A descent dual approach for linear bilevel programs. Technical Report CRT-866, Centre de Recherche sur les Transports.

Cherkassky, B. V. 1997. On implementing the push-relabel method for the maximum flow problem. *Algorithmica* **19**(4), 390–410.

Colson, B., P. Marcotte, and G. Savard 2005a. Bilevel programming: A survey. *4OR: A Quarterly Journal of Operations Research* **3**(2), 87–107.

— 2005b. A trust-region method for nonlinear programming: Algorithm and computational experience. *Computational Optimization and Applications* **30**(3), 211–227.

Cormican, K., D. Morton, and R. Wood 1998. Stochastic network interdiction. *Operations Research* **46**(2), 184–197.

Cornuéjols, G. 2008. Valid inequalities for mixed integer linear programs. *Mathematical Programming B* **112**, 3–44.

Cox, J., R. Schuessler, H. D. Jr., C. Stone, B. Chang, M. Cain, P. Corr, and J. Boineau 1991. The surgical treatment of atrial fibrillation. iii. development of a definitive surgical procedure. *The Journal of Thoracic and Cardiovascular Surgery* **101**, 569–583.

Cyert, R. and J. March 1955. Organizational structure and pricing behavior in an oligopolistic market. *The American Economic Review* **45**(1), 129–139.

de Berg, M., M. van Kreveld, M. Overmars, and O. Schwarzkopf 2000. *Computational Geometry. Algorithms and Applications*. Springer-Verlag, Berlin.

Dempe, S. 1987. A simple algorithm for the linear bilevel programming problem. *Optimization* **18**, 373–385.

— 2001. Discrete bilevel optimization problems. Technical Report D-04109, Institut fur Wirtschaftsinformatik, Universitat Leipzig, Leipzig, Germany.

— 2003. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization* **52**, 333–359.

Dolan, E. D. and J. J. Moré 2002. Benchmarking optimization software with performance profiles. *Mathematical Programming* **91**(2), 201–213.

Eswaran, P., A. Ravindran, and H. Moskowitz 1986. Algorithms for nonlinear integer bicriterion problems. *Journal of Optimization Theory and Applications* **63**(2), 261–279.

Etoa, J. 2010. A filter method to solve nonlinear bilevel programmign problems. In *Information Computing and Applications*, pages 395–406. Springer.

Facchinei, F., H. Jiang, and L. Qi 1999. A smoothing method for mathematical programs with equilibrium constraints. *Mathematical Programming* **85**(1), 107–134.

Falk, J. E. 1973. A linear max-min problem. *Mathematical Programming* **5**, 169–188.

Ferris, M. and C. Kanzow 2002. Complementarity and related problems. In P. Pardalos and M. Resende, editors, *Handbook of Applied Optimization*, pages 514–530. Oxford University Press.

Ferris, M. and J. Pang 1997. Engineering and economic applications of complementarity problems. *SIAM Review* **39**(4), 669–713.

Figueira, J. 2000. MCDM Numerical Instances Library.

Finta, B. and D. Haines 2004. Catheter ablation therapy for atrial fibrillation. *Cardiology Clinics* **22**(1), 127–145.

Fischetti, M. 1991. Facets of two steiner arborescence polyhedra. *Mathematical Programming* **51**(3), 401–419.

Fletcher, R. and S. Leyffer 2002a. Nonlinear programming without a penalty function. *Mathematical Programming* **91**(2), 239–269.

— 2002b. Numerical experience with solving mpecs by nonlinear programmign methods. Technical Report Numerical Analysis Report NA/210, University of Dundee.

Fletcher, R., S. Leyffer, D. Ralph, and S. Scholtes 2006. Local convergence of sqp methods for mathematical programs with equilibrium constraints. *SIAM Journal on Optimization* **17**(1), 259–286.

Fortuny-Amat, J. and B. McCarl 1981. A representation and economic interpretation of a two-level programming problem. *Journal of the Operations Research Society* **32**, 783–792.

Fridman, A., S. Weber, K. Dandekar, and M. Kam 2008. Cross-layer multicommodity capacity expansion on ad hoc wireless networks of cognitive radios. In *The Proceedings of the 42nd Annual Conference on Information Sciences and Systems (CISS 2008)*, pages 676–680.

Fukuda, K., T. Liebling, and F. Margot 1997. Analysis of backtrack algorithms for listing all vertices and faces of a convex polyhedron. *Computational Geometry Theory and Applications* **8**, 1–12.

Fukushima, M. and J. Pang 1999. Complementarity constraint qualifications and simplified b-stationary conditions for mathematical programs with equilibrium constraints. *Computational Optimization and Applications* **13**, 111–136.

Fukushima, M. and P. Tseng 2002. An implementable active-set algorithm for computing a b-stationary point of a mathematical program with linear complementarity constraints. *SIAM Journal on Optimization* **12**, 724–739.

— 2007. An implementable active-set algorithm for computing a b-stationary point of a mathematical program with linear complementarity constraints: Erratum. *SIAM Journal on Optimization* **17**, 1253–1257.

Garey, M. and D. Johnson 1979. *Computers and Intractability: A Guide to the Thoery of NP-Completeness*. W.H. Freeman and Company.

Geoffrion, A. 1968. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications* **22**, 618–630.

Ghare, P., D. Montgomery, and W. Turner 1971. Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly* **18**, 27–45.

Gillinov, A. M. and P. McCarthy 2004. Advances in the surgical treatment of atrial fibrillation. *Cardiology Clinics* **22**, 147–157.

Goemans, M. and D. Williamson 1995. A general approximation technique for constrained forest problems. *SIAM Journal on Computing* **24**(2), 296–317.

— 1997. The primal-dual method for approximation algorithms and its application to network design problems. In D. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, pages 144–191. PWS Publishing Company, Boston.

Gomory, R. 1969. Some polyhedra related to combinatorial problems. *Linear Algebra and Its Applications* **2**, 451–558.

Goodman, J. and J. O'Rourke 2004. *Handbook of Discrete and Computational Geometry*. Chapman & Hall/CRC.

Guzelsoy, M. 2009. *Dual Methods in Mixed Integer Programming*. Ph.D. thesis, Lehigh University.

Guzelsoy, M. and T. Ralphs 2006. The value function of a mixed-integer linear program with a single constraint. To be submitted.

— 2007. Duality for mixed-integer linear programs. *The International Journal of Operations Research* **4**, 118–137.

H. Gümüş, H. Z. and C. Floudas 2005. Global optimization of mixed-integer bilevel programming problems. *Computational Management Science* **2**(3), 181–212.

Hansen, P., B. Jaumard, and G. Savard 1992. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing* **13**(5), 1194–1217.

Harrison, G., T. Rutherford, and D. Tarr 1997. Quantifying the uruguay round. *Economic Journal* page 107.

Held, H. and D. Woodruff 2005. Heuristics for multi-stage interdiction of stochastic networks. *Journal of Heuristics* **11**(5-6), 483–500.

Hinni 2010. Monitoring system lorno.

Hobbs, B. 2001. Linear complementarity models of nash-cournot competition in bilateral and poolco power markets. *IEEE Transactions on Power Systems* **16**, 194 – 202.

Hobbs, B. and U. Helman 2004. Complementarity-based equilibrium modeling for electric power markets. In D. Bunn, editor, *Modeling Prices in Competitive Electricity Markets*. J. Wiley.

Hu, J., J. Mitchell, J.-S. Pang, and J. Yu 2009. On linear programs with linear complementarity constraints. To appear in *Journal of Global Optimization*.

Huang, J. and J.-S. Pang 1998. Option pricing and linear complementarity. *Journal of Computational Finance* **2**(3), 31–60.

Israeli, E. 1999. *System Interdiction and Defense*. Ph.D. thesis, Naval Postgraduate School.

Israeli, E. and R. Wood 2002. Shortest path network interdiction. *Networks* **40**(2), 97–111.

Izmailov, A. and M. Solodov 2008. An active-set newtown method for mathematical programs with complementarity constraints. *SIAM Journal on Optimization* **19**(3), 1003–1027.

Janjarassuk, U. and J. Linderoth 2008. Reformulation and sampling to solve a stochastic network interdiction problem. *Networks* **52**, 120–132.

Jeroslow, R. 1978. Cutting plane theory: Algebraic methods. *Discrete Mathematics* **23**, 121–150.

— 1979. Minimal inequalities. *Mathematical Programming* **17**, 1–15.

— 1985. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming* **32**, 146–164.

Jian, J.-B. 2005. A superlinearly convergent implicit smooth sqp algorithm for mathematical programs with nonlinear complementarity constraints. *Computational Optimization and Applications* **31**, 335–361.

Jiang, H. and D. Ralph 1999. Smooth sqp methods for mathematical programs with nonlinear complementarity constraints. *SIAM Journal on Optimization* **10**(3), 779–808.

Johnson, D. S., M. Minkoff, and S. Phillips 2000. The prize collecting steiner tree problem: Theory and practice. *Proceedings of the 11th Symposium on Discrete Algorithms* pages 760–769.

Johnson, E. 1973. Cyclic groups, cutting planes, and shortest paths. In T. Hu and S. Robinson, editors, *Mathematical Programming*, pages 185–211. Academic Press, New York, NY.

— 1974. On the group problem for mixed integer programming. *Mathematical Programming Study* **2**, 137–179.

— 1979. On the group problem and a subadditive approach to integer programming. *Annals of Discrete Mathematics* **5**, 97–112.

Judice, J. and A. Faustino 1992. A sequential lcp method for bilevel linear programming. *Annals of Operations Research* **34**, 89–106.

Judice, J., H. Sherali, and I. R. A. Faustino 2007. Complementarity active-set algorithm for mathematical programming problems with equilibrium constraints. *Journal of Optimization Theory and Applications* **134**(3), 467–481.

Kall, P. and S. Wallace 1994. *Stochastic Programming*. Wiley.

Khachian, L. 1979. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady* **20**, 191–194.

Klarbring, A., J. Petersson, and M. Rönnqvist 1995. Truss topology optimization involving unilateral contact. *Journal of Optimization Theory and Applications* **87**, 1–31.

Klarbring, A. and M. Rönnqvist 1995. Nested approach to structural optimization in non-smooth mechanics. *Structural Optimization* **10**, 79–86.

Kočvara, M. and J. Outrata 1990. On optimization systems governed by implicit complementarity problems. *Numerical Functional Analysis and Optimization* **15**, 869–887.

— 1995. On the solution of optimum design problems with variational inequalities. In D. Du, L. Qi, and R. Womersly, editors, *Recent Advances in Nonsmooth Optimization*, pages 172–192. World Scientific, Singapore.

Kong, N., A. Schaefer, and B. Hunsaker 2006. Two-stage integer programs with stochastic right-hand sides: A superadditive dual approach. *Mathematical Programming B* **108**, 275–296.

Koopmans, T. C. 1951. Efficient allocation of resources. *Econometrica* **19**(4), 455–465.

Köppe, M., M. Queyranne, and C. Ryan 2009. A parametric integer programming algorithm for bilevel mixed integer programs. Available at `http://arxiv.org/abs/0907.1298v2`.

Korhonen, P. and P.-A. Siitari 2007. Using lexicographic parametric programming for identifying efficient units in dea. *Computers and Operations Research* **34**(2), 2177–2190.

Krause, A., J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos 2008. Efficent sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management* **134**(6), 516–526.

Kunapuli, G. 2008. *A Bilevel Optimization Approach to Machine Learning*. Ph.D. thesis, Rensselaer Polytechnic Institute.

Labbè, M., P. Marcotte, and G. Savard 1998a. A bilevel model of taxation and its application to optimal highway pricing. *Management Science* **44**, 1608–1622.

— 1998b. On a class of bilevel programs. In D. di Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 1–24. Kluwer Academic Publishers.

Lasserre, J. 2004a. Generating functions and duality for integer programs. *Discrete Optimization* **1**, 167 – 187.

— 2004b. The integer hull of a convex rational polytope. *Discrete & Computational Geometry* **32**(1), 129 – 139.

— 2009. Duality and a Farkas lemma for integer programs. In E. Hunt and C. Pearce, editors, *Optimization: Structure and Applications*. Kluwer Academci Publishers.

Lim, C. and J. Smith 2007. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions* **39**(1), 15–26.

Liu, G., J. Han, and S. Wang 1998. A trust region algorithm for bilevel programming problems. *Chinese Science Bulletin* **43**(10), 820–824.

Liu, G. and J. Ye 2007. A merit function piecewise sqp algorithm for solving mathematical programs with equilibrium constraints. *Journal of Optimization Theory and Applications* **135**, 623–641.

Liu, X., G. Perakis, and J. Sun 2006. A robust sqp method for mathematical programs with linear complementarity constraints. *Computational Optimization and Applications* **34**(1), 5–33.

Liu, X. and J. Sun 2004. Generalized stationary points and an interior point method for mpec. *Mathematial Programming* **101**(1), 231–261.

Ljubic, I., R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti 2005. An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. *Mathematical Programming, Series B*. To appear.

Llewellyn, D. and J. Ryan 1993. A primal-dual integer programming algorithm. *Discrete applied mathematics* **45**, 261–276.

Lodi, A. and T. Ralphs 2009. Bilevel programming and maximally violated valid inequalities. In *Cologne Twente Workshop on Graphs and Combinatorial Optimization*.

Loridan, P. and J. Morgan 1996. Weak via strong stackelberg problem: New results. *Journal of Global Optimization* **8**(3), 263–287.

Lougee-Heimer, R. 2003. The Common OPtimization INterface for Operations Research. *IBM Journal of Research and Development* **47**(1), 57–66.

Luo, Z.-Q., J.-S. Pang, and D. Ralph 1996. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press.

Maier, G. and G. Novati 1990. A shakedown and bounding theory allowing for nonlinear hardening and second order geometric effects with reference to discrete structural models. In *Inelastics Solids and Structures, A. Sawczuk Memorial Volume*, pages 451–471. Pineridge Press.

Mangasarian, O. 1976. Equivalence of the complementarity problem to a system of nonlinear equations. *SIAM Journal on Applied Mathematics* **31**, 89–92.

— 1996. Mathematical programming in machine learning. In G. Di. Pillo and F. Giannessi, editors, *Nonlinear Optimization and Applications*, pages 283–295. Plenum, New York.

Mangasarian, O. and J. Pang 1997. Exact penalty functions for mathematical programs with linear complementarity constraints. *Optimization* **42**, 1–8.

Marcotte, P. and D. Zhu 1995. Exact and inexact penalty methods for the generalized bilevel programming problem. *Mathematical Programming A* **74**, 141–157.

Margot, F., A. Prodon, and T. M. Liebling 1994. Tree polytope on 2-trees. *Mathematical Programming* **63**(2), 183–191.

McMasters, A. and T. Mustin 1970. Optimal interdiction of a supply network. *Naval Research Logistics Quarterly* **17**, 261–268.

Meyer, A. and L. Stockmeyer 1972. The equivalence problem for regular expressions with squaring requires exponential space. In *The Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory*, pages 125–129.

Miller, T., T. Friesz, and R. Tobin 1992. Heuristic algorithms for delivered price spatially competitive network facility location problems. *Annals of Operations Research* **34**, 177–202.

Moore, J. and J. Bard 1990. The mixed integer linear bilevel programming problem. *Operations Research* **38**(5), 911–921.

Morton, D., F. Pan, and K. Saeger 2007. Models for nuclear smuggling interdiction. *IIE Transactions* **39**(1), 3–14.

Nemhauser, G. and L. Wolsey 1999. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc.

Nisan, N., T. Roughgarden, E. Tardos, and V. Vazirani., editors 2007. *Algorithmic Game Theory*. Cambridge University Press.

Ostfeld, A. and E. Salomons 2004. Optimal layout of early warning detection stations for water distribution systems security. *Journal of Water Resources Planning and Management* **130**(5), 377–385.

Outrata, J. 1994. On optimization problems with variational inequality constraints. *SIAM Journal on Optimization* **4**(2), 340357.

Outrata, J., M. K. M, and J. Zowe 1998. *Nonsmooth Approach to Optimization Problems with Equilibrium Constraints: Theory, Applications and Numerical Results*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Outrata, J. and J. Zowe 1995. A numerical approach to optimization problems with variational inequality constraints. *Mathematical Programming* **68**, 105–130.

Pang, J., V. Kumar, and P. Song 2005. Convergence of time-stepping method for initial and boundary-value frictional compliant contact problems. *SIAM Journal on Numerical Analysis* **43**(5), 2200–2226.

Pang, J. and J. Trinkle 1996. Complementarity formulations and existence of solutions of multi-rigid-body contact problems with coulomb friction. *Mathematical Programming* **73**, 199–226.

Papavassilopoulos, G. 1982. Algorithms for static stackelberg games with linear costs and polyhedral constraints. In *The Proceedings of the 21st IEEE Conference on Decisions and Control*, pages 647–652.

Patriksson, M. and L. Wynter 1997. Stochastic mathematical programs with equilibrium constraints. *Operations Research Letters* **25**, 159–167.

Pharkya, P., A. Burgard, and C. Maranas 2003. Exploring the overproduction of amino acids using the bilevel optimization framework optknock. *Biotechnology and Bioengineering* **84**, 887–899.

Prodon, A., S. DeNegre, and T. Liebling 2010. Locating leak detecting sensors in a water distribution network by solving prize-collecting steiner arborescence problems. *Mathematical Programming B* **124**, 119–141.

Ralph, D. 2009. Optimization with equilibrium constraints: A piecewise sqp approach. In C. Floudas and P. Pardalos, editors, *Encyclopedia of Optimization*, pages 2807–2813. Springer.

Ralphs, T., M. Saltzman, and M. Wiecek 2006. An improved algorithm for biobjective integer programming. *Annals of Operations Research* **147**, 43–70.

Rentmeesters, M., W. Tsai, and K.-J. Lin 1996. A theory of lexicographic multi-criteria optimization. In *Second IEEE International Conference on Engineering of Complex computer Sytems*, pages 76–79. IEEE Computer Society Press.

Rothe, J. 2005. *Complexity Theory and Cryptology: An Introduction to Cryptocomplexity*. Springer.

Royset, J. and R. Wood 2007. Solving the bi-objective maximum-flow network-interdiction problem. *INFORMS Journal on Computing* **19**(2), 175–184.

Rudin, W. 1976. *Real Analysis*. McGraw-Hill.

Salmerón, J., K. Wood, and R. Baldick 2009. Worst-case interdiction analysis of large-scale electric power grids. *IEEE Transactions on Power Systems* **24**, 96–104.

Savard, G. 1989. *Contributions à la programmation mathématique à deux niveaux*. Ph.D. thesis, Université de Montréal, École Polytechnique.

Savelsbergh, M. 1994. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing* **6**(4), 445–454.

Scarf, H. 1973. *The Computation of Economic Equilibria*. Yale University Press.

Scheel, H. and S. Scholtes 2000. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research* **25**(1), 1–22.

Scholtes, S. 2001. Convergence properties of a regularisation scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization* **11**, 918–936.

Scholtes, S. and M. Stöhr 1999. Exact penalization of mathematical programs with equilibrium constraints. *SIAM Journal on Control and Optimization* **37**(2), 617652.

Schultz, R., L. Stougie, and M. H. V. D. Vlerk 1996. Two-stage stochastic integer programming: a survey. *Statistica Neerlandica* **50**, 404–416.

Senn, P. 1996. Heinrich von stackelberg in the history of economic ideas. *Journal of Economic Studies* **23**(5/6), 15–39.

Shimizu, K., Y. Ishizuka, and J. Bard 1997. *Nondifferentiable and Two-Level Mathematical Programming*. Kluwer Academic Publishers.

Slyke, R. V. and R. Wets 1969. L-shaped linear programs with applications to optimal control stochastic linear programs. *SIAM Journal of Applied Mathematics* **17**, 638–663.

Smeers, Y. 1997. Computable equilibrium models and the restructuring of the european electricity and gas markets. *The Energy Journal* **18**(4), 1–31.

Stadler, W. 1988. Fundamentals of multicriteria optimization. In W. Stadler, editor, *Multicriteria Optimization in Engineering and in the Sciences*, pages 1–25. Plenum Press, New York.

Stockmeyer, L. 1977. The polynomial-time hierarchy. *Theoretical Computer Science* **3**, 1–22.

Sun, C., S. Ritchie, K. Tsai, and R. Jayakrishnan 1998. Use of vehicle signature analysis and lexicographic optimization for vehicle reidentification on freeways. *Transportation Research Part C: Emerging Technologies* **7**(4), 167–185.

Tin-Loi, F. and J. Misa 1999. Large displacement of elastoplastic analysis of semirigid steel frames. *International Journal for Numerical Methods in Engineering* **39**, 741–762.

Tin-Loi, F. and J. Pang 1993. Elastoplastic analysis of structures with nonlinear hardening. *Computer Methods in Applied Mechanics and Engineering* **107**, 299–312.

Tuy, H., A. Migdalas, and P. Värbrand 1994. A global optimization approach for the linear two-level program. *Journal of Global Optimization* **4**, 243–263.

Veenhuyzen, G., C. Simpson, and H. Abdollah 2004. Atrial fibrillation. *Canadian Medical Association Journal* **171**(7), 755–760.

Vicente, L. and P. Calamai 1994. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization* **5**, 291–306.

Vicente, L., G. Savard, and J. Judice 1996. Discrete linear bilevel programming problem. *Journal of Optimization Theory and Applications* **89**(3), 597–614.

von Stackelberg, H. 1934. *Marktform und Gleichgewicht*. Julius Springer.

Waktare, J. 2002. Atrial fibrillation. *Circulation* **106**, 14–16.

Waltz, F. 1967. An engineering approach: Hierarchical optimization criteria. *IEEE Transactions on Automatic Control* **AC-12**, 179–180.

Washburn, A. and K. Wood 1995. Two-person zero sum games for network interdiction. *Operations Research* **43**, 243–251.

Watson, J.-P., H. Greenberg, and W. Hart 2004. A multiple-objective analysis of sensor placement optimization in water networks. In *Proceedings of the World Water and Environmental Resources Conference*. Reston, VA.

Wen, U. and S. Hsu 1989. A note on a linear bi-level programming algorithm based on bi-criterion programming. *Computer & Operations Research* **16**, 79–83.

Wen, U. and S. Lin 1996. Finding an efficient solution to linear bilevel programming problem: An effective approach. *Journal of Global Optimization* **8**, 295–306.

Wen, U. and Y. Yang 1990. Algorithms for solving the mixed integer two-level linear programming problem. *Computers & Operations Research* **17**(2), 133–142.

Wollmer, R. 1964. Removing arcs from a network. *Operations Research* **12**(6), 934–940.

Wolsey, L. 1981. Integer programming duality: Price functions and sensitivity analysis. *Mathematical Programming* **20**, 173–195.

Wood, R. 1993. Deterministic network interdiction. *Mathematical and Computer Modelling* **17**(2), 1–18.

Wu, Y., P. Chou, Q. Zhang, K. Jain, and S. Kung 2005. Network planning in wireless ad hoc networks: A cross-layer approach. *IEEE Journal on Selected Areas in Communications* **23**(1), 136–150.

Xu, Y., T. Ralphs, and M. Saltzman 2009. Computational experience with a software framework for parallel integer programming. *INFORMS Journal on Computing* **21**, 383–397.

# Biography

Scott DeNegre is a doctoral candidate in the Industrial & Systems Engineering Department at Lehigh University. His primary research interests include bilevel programming, integer programming, interdiction, and algorithmic game theory. Prior to entering into candidacy for his doctorate, Scott earned a B.S in Mathematical Sciences from Johns Hopkins University and an M.S. in Management Science from Lehigh. Scott is a member of the Institute for Operations Research and Management Science (INFORMS), the Society for Industrial and Applied Mathematics (SIAM), the Military Operations Research Society (MORS), the Society of Cost Estimating and Analysis (SCEA), the Computational Infrastructure for Operations Research (COIN-OR), and the Computational Optimization Research at Lehigh (COR@L) Lab. Scott holds the position of Operations Research Analyst at Technomics, Inc., where his current methodological focus is on satellite system analysis. Previously, Scott has held positions as a visiting researcher in the Interventional Guidance Technology Group at Philips Research North America, an actuarial analyst at Watson Wyatt, and a visiting researcher in the mathematics department at École Polytechnique Federal de Lausanne in Switzerland.