MIXED-INTEGER PROGRAMMING APPROACHES FOR SOME NON-CONVEX AND COMBINATORIAL OPTIMIZATION PROBLEMS

by

Srikrishna Sridhar

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2014

Date of final oral examination: 01/23/2014

The dissertation is approved by the following members of the Final Oral Committee:
Stephen J. Wright, Professor, Computer Sciences (Advisor)
Jeffrey Linderoth, Professor, Industrial and Systems Engineering (Advisor)
James Luedtke, Assistant Professor, Industrial and Systems Engineering (Advisor)
Thomas Rutherford, Professor, Agricultural and Applied Economics
Christopher Ré, Assistant Professor, Computer Science (Stanford University)

ACKNOWLEDGMENTS

First and foremost, I express the deepest thanks to my advisors Prof. Stephen Wright, Prof. Jeffrey Linderoth and Prof. James Luedtke. I consider myself to be one of the luckier students at the University of Wisconsin-Madison because I had the chance to work freely with three of the world's leading experts in the field of numerical optimization. In addition to being warehouses of knowledge, they were ideal mentors that gave me the freedom to explore research directions that I found fascinating. My countless interactions with them will have a lasting impact on my professional career as well as personal life. They are model academics and I wish them the best in nurturing a future generation of more model academics.

I would also like to give a personal thanks to Prof. Christopher Ré for his endless supply of energy, enthusiasm, and ideas. The clarity in his thoughts, his passion for work and high standards of quality and productivity will forever be an inspiration to me. My work with him and his students have rekindled my passion for building high-performance software.

I would like to thank Victor Bittorf, a fellow graduate student at the University of Wisconsin-Madison. The year I spent hacking code with him has been my most productive and fun year in graduate school. I hope that, in the future, I get a chance to work with peers that are as motivated, smart, and fun as him.

The University of Wisconsin-Madison has provided me with opportunities to learn from so many excellent teachers and mentors. I would like to thank Prof. Patricia Brennan and Prof. Tom Rutherford for their valuable advice and Prof. Recht, Ferris, Coen, Newton, Chawla for offering courses that I enjoyed.

I have had the chance to interact with many amazing graduate students. They have helped me resolve challenges, critiqued my presentations and manuscripts, and listened to my cricket rambles. In no particular order, I would like to thank Ji Liu, Siddharth (Sid) Barman, Badri Bhaskar, Yongjia Song, Taedong Kim, Nilay Vaish, Parikshit Shah, Nikhil Rao, Gautam Dasarathy, Kevin Jamieson, Mark Wellons, Madhu Vadali, Jesse Holzer, Cong Han Lim, Ce Zhang, Zack Harmany, Matt Malloy, Arun Kumar, Merve Bodur, Mahdi Hameezi, Sang-Kyun Lee and Tyson Williams.

A special thanks to Ji and Sid for being my theory pillars.

Finally, I would like to thank my wife, brother, and parents whose endless support and patience has given me the confidence to succeed in anything that I put my mind to.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASCD	Asynchronous Stochastic Co-ordinate Descent
CC	Convex Combination Model
DCC	Disaggregated Convex Combination Model
IP	Integer Program
IRR	Internal Rate of Return
LP	Linear Program
MCM	Multiple Choice Model
MINLP	Mixed-Integer Non Linear Program
MIP	Mixed-Integer (linear) Program
MIQP	Mixed-Integer Quadratic Program
NLP	Non Linear Program
NPV	Net Present Value
NUMA	Non Uniform Memory Access
PLF	Piecewise Linear Function
PSC	Production Sharing ContractModel
QP	Quadratic Program
ROI	Return On Investment
SCD	Stochastic Coordinate Descent
SGD	Stochastic Gradient Descent
SOS2	Specially Ordered Set of type 2

ABSTRACT

In this dissertation we study several nonconvex and combinatorial optimization problems with applications in production planning, machine learning, advertising, statistics, and computer vision. The common theme is the use of algorithmic and modelling techniques from mixed-integer programming (MIP) which include formulation strengthening, decomposition, and linear programming (LP) rounding.

We first consider MIP formulations for piecewise linear functions (PLFs) that are evaluated when an indicator variable is turned on. We describe modifications to standard MIP formulations for PLFs with desirable theoretical properties and superior computational performance in this context.

Next, we consider a production planning problem where the production process creates a mixture of desirable products and undesirable byproducts. In this production process, at any point in time the fraction of the mixture that is an undesirable byproduct increases monotonically as a function of the cumulative mixture production up to that time. The mathematical formulation of this continuous-time problem is nonconvex. We present a discrete time mixed-integer nonlinear programming (MINLP) formulation that exploits the increasing nature of the byproduct ratio function. We demonstrate that this new formulation is more accurate than a previously proposed MINLP formulation. We describe three different mixed-integer linear programming (MIP) approximation and relaxation models of this nonconvex MINLP, and derive modifications that strengthen the LP-relaxations of these models. We provide computational experiments that demonstrate that the proposed formulation is more accurate than the previous formulation, and that the strengthened MIP approximation and relaxation models can be used to obtain near-optimal solutions for large instances of this nonconvex MINLP.

We then study production planning problems in the presence of realistic business rules like taxes, tariffs, and royalties. We propose two different solution techniques. The first is a MIP formulation while the second is a search algorithm based on a novel continuous domain formulation. We then discuss decomposition methods to compute bounds on the optimal solution. Our computational

experiments demonstrate the impact of our formulations, solution techniques, and algorithms on a sample application problem.

Finally, we study three classes of combinatorial optimization problems: set packing, set covering, and multiway-cut. Near-optimal solutions of these combinatorial problems can be computed by rounding the solution of an LP. We show that one can recover solutions of comparable quality by rounding an approximate LP solution instead of an exact one. These approximate LP solutions can be computed efficiently by solving a quadratic-penalty formulation of the LP using a parallel stochastic coordinate descent method. We derive worst-case runtime and solution quality guarantees of this scheme using novel perturbation and convergence analyses. Our experiments demonstrate that on these combinatorial problems our rounding scheme is up to an order of magnitude faster than Cplex (a commercial LP solver) while producing solutions of similar quality.

1 INTRODUCTION

Optimization is a powerful mathematical language that can be used to aid decision making in complex systems occurring in various fields such as health care, data analysis, and manufacturing systems. An optimization model, which we refer to simply as a *model* throughout this dissertation, involves a set of mathematical relationships that capture the interactions of a system under study, and that state the our objectives for the operation of this system. The nature of these relationships (linear, nonlinear) and the nature of the decisions (discrete, continuous) govern the expressivity and computational difficulty in solving these models. The process of modeling requires a deep understanding of the application under consideration, knowledge of the expressive power of each modeling paradigm, and a familiarity with the computational principles involved in solving a model.

In this work, we study several problems that can be modelled using a specific modeling paradigm, known as mixed-integer linear programming (MIP). In a MIP model, the decision options open to agents are called *decision variables* that can be continuous variables $x \in \mathbb{R}^p$ or discrete variables $z \in \{0,1\}^q$. The requirements of the system are expressed using *constraints* while the *objective function* is a score that measures the quality of the outcome of the decisions. In a MIP model, the objective function ($a^T x + b^T z$) and constraints $Cx + Dz \leqslant e$ are expressed using linear functions of the decision variables x and z. The data required by a MIP model is encoded in the parameters $a \in \mathbb{R}^p$, $b \in \mathbb{R}^q$, $C \in \mathbb{R}^{n \times p}$, $D \in \mathbb{R}^{n \times q}$, and $e \in \mathbb{R}^n$ whose values are fixed in each instantiation of the model, but may change if the model is adapted to different data sets and different situations. Commercial MIP

solvers can now solve MIP models with thousands of decision variables. (There are still MIP models with only hundreds of variables that are solvable in a reasonable amount of time.) A MIP can be formally written as:

$$\begin{aligned} & \min a^T x + b^T z \\ \text{subject to} \quad & Cx + Dz \leqslant e, \\ & x \in \mathbb{R}^p, z \in \{0,1\}^q \end{aligned} \tag{1.1}$$

Any solution (x, z) to (1.1) that satisfies both domain restrictions $x \in \mathbb{R}^p, z \in \{0,1\}^q$ and constraints $Cx + Dz \leqslant e$ is known as a *feasible solution*. The set of all feasible solutions is called the *feasible region*. The feasible solution(s) that achieves the lowest objective value is known as the optimal solution(s). Problems without any feasible solutions are called *infeasible* while those whose optimal objective can be infinitely negative, while still being feasible, are called *unbounded*. Note that the model (1.1) does not contain any nonlinear functions like products of decision variables or absolute value of a variables. While this might seem restrictive at first glance, one can often express such conditions using simple linear transformations.

In this dissertation, we study MIP models with applications in production planning, machine learning, advertising, statistics, and computer vision. The common theme is the use of algorithmic and modelling techniques from MIP. In this chapter, we review some basic concepts that are extensively used throughout this dissertation.

1.1 MIP Formulations

In this section, we review how one can provide an algebraic description of a practical problem using MIP. Such a description is known as a *MIP formulation*. We review ideas that help us better understand why there may be multiple (possibly infinite) MIP formulations for the same problem and that some of them are inherently *better* than others. An understanding of the theoretical properties of MIP formulations helps in formulating MIP models with desirable computational properties. Consider the following definition of a MIP formulation for a set $X \subseteq \mathbb{R}^p \times \{0,1\}^q$.

Definition 1.1. *A subset of $\mathbb{R}^n$ described by a finite set of linear constraints $P := \{x \in \mathbb{R}^n : Ax \leqslant b\}$ is called a polyhedron. (Definition 1.1 of Wolsey [109])*

Definition 1.2. *Given a set $X \subseteq \mathbb{R}^p \times \{0,1\}^q$, a polyhedron $P \subseteq \mathbb{R}^{p+q}$ is a formulation for the set X iff $X = P \cap (\mathbb{R}^p \times \{0,1\}^q)$. (Definition 1.2 of Wolsey [109])*

From the discussion above, it is easy to see that there may be many valid MIP formulations for X. In Figure 1.1, we illustrate two different valid formulations for a problem with three feasible points in the set X (shown using red dots). In fact, one can construct infinitely many valid MIP formulations with the same feasible region. Based on Definitions 1.1 and 1.2, some natural questions are: Are some MIP formulations are *better* than others? Is there a notion of a *best* or *ideal* MIP formulation? To answer these questions, we focus on the *linear programming (LP) relaxation* of a MIP formulation (denoted as $LP(P)$). For the MIP formulation defined in (1.1), the LP-relaxation is obtained by replacing the restrictions on the variables

$z \in \{0,1\}^q$ with $z \in [0,1]^q$:

$$\begin{aligned} &\min a^T x + b^T z \\ \text{subject to } &Cx + Dz \leqslant e, \\ &x \in \mathbb{R}^p, z \in [0,1]^q \end{aligned} \tag{1.2}$$

The LP defined in (1.2) can be solved in polynomial time [52] while the formulation (1.1) is NP-hard. Moreover, (1.2) is a relaxation of (1.1) because its feasible region $\{x \in \mathbb{R}^p, z \in [0,1]^q : Cx + Dz \leqslant e\}$ is a superset of the feasible region of (1.1). Hence the cost of the optimal solution of (1.2) is always lower (or equal) to that of (1.1).

Comparing Formulations. Consider the two MIP formulations (for the same set) illustrated in Figure 1.1. For us to chose between them, we must consider the LP-relaxation (illustrated using the pink shaded region). The smaller (or tighter) the LP-relaxation, the better the MIP model. The following definition formalizes this notion.

Definition 1.3. *Given a feasible region* X *and two valid formulations* P_1 *and* P_2 *for the set* X, P_1 *is better than* P_2 *if* $P_1 \subset P_2$. *(Definition 1.4 of Wolsey [109])*

The definition above assumes that the decision variables of P_1 and P_2 are are in the same domain. We can easily extend this definition to compare two formulations $P \subseteq \mathbb{R}^{p+q}$ and $Q \subseteq \mathbb{R}^{p+q} \times \mathbb{R}^r$ of $X \subseteq \mathbb{R}^p \times \{0,1\}^q$ by comparing P with the projection

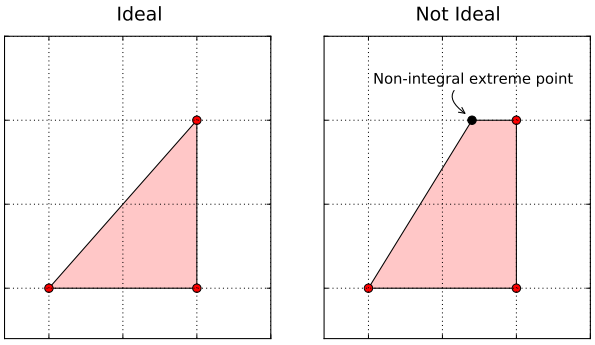


Figure 1.1: Two different formulations of an MIP.

of Q onto the subspace $\mathbb{R}^{p+q}$, defined as

$$\text{Proj}_x(Q) := \{x \in \mathbb{R}^{p+q} : (x, w) \in Q \text{ for some } w \in \mathbb{R}^r\}.$$

$\text{Proj}_x(Q) \subseteq \mathbb{R}^{p+q}$ is a formulation of the set X in the space of the original variables $x \in \mathbb{R}^{p+q}$. Using this definition, we assert that Q is a better formulation of X than P if $\text{Proj}_x(Q) \subset P$. The formulation Q is called an *extended formulation* of the set X because it introduces additional variables $w \in \mathbb{R}^r$.

Ideal Formulations. We now extend Definition 1.3 to define an *ideal* or a *best* formulation of X as one with the smallest possible LP-relaxation. In such a formulation, the optimal solution of the LP-relaxation always satisfies integrality constraints on the binary variables. Hence, in an *ideal* formulation, solving the LP-relaxation is equivalent to solving the MIP formulation. We can formalize this idea as:

Definition 1.4. *Given the feasible region* $X \subseteq \mathbb{R}^{p+q}$ *of* (1.2)*, the convex hull of* X *is defined*

as

$$conv(X) := \Big\{ y \in \mathbb{R}^{p+q} : y = \sum_{i=1}^{t} \lambda_i y^i, \sum_{i=1}^{t} \lambda_i = 1, \lambda_i \geqslant 0 \; \forall i = 1, \dots t \\ \textit{over all finite subsets } \{y^1, y^2, \dots y^t\} \textit{ of } X \Big\}.$$

(Definition 1.3 of Wolsey [109])

Hence, the best one can do is to formulate $\text{conv}(X)$. If not, we must try to formulate problems using MIPs that are as close as possible to $\text{conv}(X)$.

Locally Ideal & Sharp Formulations.

We can study the polyhedral properties of a subset of a MIP formulation and still learn something about the entire formulation. Padberg and Rijal [74] address this issue by defining a *locally ideal* MIP formulation as one, which in the absence of other constraints, satisfies the property that all extreme points of its corresponding LP-relaxation satisfy integrality conditions. The term *'ideal'* refers to the property that this is the best that any MIP formulation can do from any perspective while the term *'local'* highlights that it only applies to a selected portion and not the entire MIP formulation. A locally ideal formulation is a minimal and complete linear description of the polytope corresponding to every extreme point of the convex hull of the MIP formulation [74].

Jeroslow and Lowe [50] define another desirable property of MIP formulations called *sharpness* which is slightly weaker than local idealalness. Consider an extended MIP formulation $Q \subseteq \mathbb{R}^{p+q+r}$ for a set $X \subseteq \mathbb{R}^p \times \{0,1\}^q$ whose optimal

integral solution is

$$z^* = \{\min c^T x + d^T w \quad \text{subject to } (x, w) \in Q \cap (\mathbb{R}^p \times \{0,1\}^q \times \mathbb{R}^r)\}.$$

The optimal solution of the LP-relaxation of Q is given by

$$z^{LP} = \{\min c^T x + d^T w \quad \text{subject to } (x, w) \in Q\}.$$

The value z^{LP} is known as the *LP-relaxation bound*. Jeroslow and Lowe [50] define a formulation as *sharp* if it achieves the best possible value of the LP-relaxation bound amongst all formulations that model the set X. An extended MIP formulation $Q \subseteq \mathbb{R}^{p+q} \times \mathbb{R}^r$ of the set X is sharp when the set of extreme points of $\text{Proj}_x(Q)$ and $\text{conv}(X)$ are exactly the same. Based on this description, it is easy to see that every locally ideal formulation is sharp because a locally ideal formulation is exactly $\text{conv}(X)$. The key difference between sharp formulations and locally ideal formulations is that sharp formulations are strongest possible with respect to the LP-relaxation bounds but offer no guarantees on integrality of the optimal solution of the LP-relaxation. Locally ideal formulations are *ideal* with respect to the bounds from the LP-relaxation bound as well as integer feasibility. It is important to study the polyhedral properties of MIP formulations because desirable formulation attributes (like locally ideal) can significantly impact performance when solved using state of the art solvers. Even though state of the art solvers use complex algorithms and heuristics that make it hard to predict exactly which formulation performs better, it is almost always the case that formulations with stronger (or

smaller) LP-relaxations will perform better.

Example: Piecewise-Linear Functions.

We present an example of a problem with multiple MIP formulations. We review some known results about the strength of these MIP formulations and discuss their consequences in practical applications. We consider MIP formulations for piecewise-linear functions (PLFs); a structure that occurs frequently in this dissertation.

Consider a continuous univariate PLF $f : [B_0, B_n] \to \mathbb{R}$ with its domain $[B_0, B_n]$ divided into an increasing sequence of breakpoints $\{B_0, B_1, \ldots, B_n\}$. The function $f(\cdot)$ can be written as

$$f(x) := \begin{cases} m_1 x + c_1, & x \in [B_0, B_1] \\ m_2 x + c_2, & x \in [B_1, B_2] \\ \vdots & \\ m_n x + c_n, & x \in [B_{n-1}, B_n] \end{cases} \tag{1.3}$$

where $m_i \in \mathbb{R}, c_i \in \mathbb{R}$. There are several known MIP formulations [10, 64, 50, 27, 67, 105] for the set

$$PL = \{(x, y) \in \mathbb{R}^2 : y = f(x)\}.$$

Of these, we consider two formulations, one of which is provably stronger than the other.

Convex Combination Model. The first formulation for PL is called the *convex combination* model [27] (also known as the *lambda method*). The convex combination model introduces continuous variables $\boldsymbol{\lambda} \in \mathbb{R}_+^{n+1}$ and binary variables $\mathbf{b} \in \{0,1\}^n$. The function argument x and function value y are expressed as a convex combination of the variables $\boldsymbol{\lambda} \in \mathbb{R}_+^{n+1}$ using:

$$y = \sum_{i=0}^{n} \lambda_i F_i, \quad x = \sum_{i=0}^{n} \lambda_i B_i, \quad 1 = \sum_{i=0}^{n} \lambda_i \tag{1.4}$$

where $F_i := f(B_i) = m_i B_i + c_i$, $\forall i \in \{1 \dots n\}$. The binary variables $\mathbf{b}$ are used to enforce the following *adjacency condition*:

$$\begin{aligned} & b_1 = 0 \Rightarrow \lambda_0 = 0, \\ & \Big(b_i = 1 \Rightarrow \lambda_j = 0, \quad \forall j \notin \{i, i-1\} \Big) \quad \forall i \in \{1 \dots n-1\} \\ & b_n = 0 \Rightarrow \lambda_n = 0. \end{aligned} \tag{1.5}$$

Finally, we enforce $\sum_{i=0}^{n} b_i = 1$ to make sure that exactly one binary variable from $\mathbf{b}$ is set to 1. Constraints (1.4) and (1.5) ensure that we select exactly one piece of $f(\cdot)$. Once that piece is chosen, we express x and y as a linear combination of two end points of the chosen piece. For example, let us assume that kth piece of $f(\cdot)$ is selected. Setting $b_k = 1$ ensures that $\lambda_i = 0 \; \forall i \notin \{k, k-1\}$ which implies that the function value y (and argument x) is a linear combination of F_k and F_{k-1} (B_k and

B_{k-1} resp.). The convex combination model for the set PL is given by:

$$\begin{aligned} CC := \Big\{ (x, y, \boldsymbol{\lambda}, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}_+^{n+1} \times \{0,1\}^n : \\ x = \sum_{i=0}^{n} \lambda_i B_i,\ y = \sum_{i=0}^{n} \lambda_i F_i,\ \sum_{i=0}^{n} \lambda_i = 1,\ \sum_{i=1}^{n} b_i = 1, \\ \lambda_0 \leqslant b_1,\ \lambda_n \leqslant b_n,\ \lambda_i \leqslant b_i + b_{i+1}\ \forall i \in \{1 \dots n-1\} \Big\}. \end{aligned}$$

Multiple Choice Model. The multiple choice model was first introduced by Jeroslow and Lowe [50] and analyzed by Balakrishnan and Graves [3]. In this model, we introduce continuous variables $\boldsymbol{w} := \{w_1, \dots w_n\}$ and binary variables $\mathbf{b} := \{b_1, \dots b_n\}$ to enforce the logical implication that $w_i = x$ if x is in the ith interval and $w_i = 0$ otherwise. This condition can be expressed algebraically using:

$$\begin{aligned} &\sum_{i=1}^{n} w_i = x, \\ &B_{i-1} b_i \leqslant w_i \leqslant B_i b_i \quad \forall i \in \{1 \dots n\} \end{aligned}$$

Again, we require $\sum_{i=0}^{n} b_i = 1$ to ensure that exactly one piece is chosen. The function value y can be expressed in terms of the variables $\boldsymbol{w}$ and $\mathbf{b}$ using:

$$y = \sum_{i=1}^{n} (m_i w_i + c_i b_i).$$

It is easy to see that when $b_i = 1$ the multiple choice model enforces $y = m_i x + c_i$. The multiple choice model for PL is given by:

$$\text{MCM} := \Big\{(x, y, \boldsymbol{w}, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \times \{0,1\}^n : \sum_{i=1}^{n} w_i = x,\ y = \sum_{i=1}^{n} (m_i w_i + c_i b_i),\ B_{i-1} b_i \leqslant w_i \leqslant B_i b_i\ \forall i \in \{1 \dots n\}\Big\}.$$

Comparing CC and MCM. Balakrishnan and Graves [3] show that MCM is locally ideal while Padberg [73] and Lee and Wilson [56] show that CC is not locally ideal. Vielma et al. [102] show that CC is sharp. They also conducted a computational study comparing several MIP formulations of univariate PLFs and conclude that Cplex, a commercial IP solver, can solve MCM up to 10x faster than CC.

1.2 Solving MIPs Exactly: Branch and Bound

In this section, we describe the *branch and bound* algorithm, a popular algorithm for finding optimal solutions of discrete optimization problems. In branch-and-bound, we systematically enumerate all candidate solutions and discard large subsets of feasible solutions using upper and lower bound estimates. This algorithm has been successfully implemented in many commercial MIP solvers. Refer to Wolsey [109] for details on the theoretical as well as computational aspects of this algorithm.

Consider the following optimization problem:

$$z^* = \min\{c^T x : x \in X\} \tag{1.6}$$

In branch-and-bound, the set X is decomposed into subsets $X_1, X_2 \dots X_K$ such that $X = \bigcup_{i=1}^{K} X_i$. These subsets are then used to solve smaller subproblems whose solutions

$$z_i = \min\{f(x) : x \in X_i\} \quad i = 1 \dots K$$

can be aggregated to find the optimal solution of (1.6) using $z^* = \min_i z_i$. As the name suggests, a generic branch-and-bound algorithm consists of *branch* and *bound* steps. In the *branch* step, we recursively decompose X into a collection of subsets (called nodes) that define a tree structure (called the search tree). For the MIP formulation (1.1), the nodes are of the form $S := \{x \in \mathbb{R}^p, z \in \{0,1\}^q : Cx + Dz \leqslant e, Fz \leqslant f\}$ where $Fz \leqslant f$ are a set of additional linear constraints specified on the binary variables that are commonly derived from the optimal solution of the LP-relaxation (1.2). For example, let (x^*, z^*) denote an optimal solution of the LP-relaxation of a node S (denoted as $LP(S)$). Let $z_i^* = \delta$ denote an optimal value of the decision variable z_i. A valid decomposition, in the branch step, would be to divide S into the subsets $S_1 := \{(x, z) \in S : z_i \leqslant \lfloor \delta \rfloor\}$ and $S_2 := \{(x, z) \in S : z_i \geqslant \lfloor \delta \rfloor + 1\}$.

In the *bound* step, we first select a node $S \subseteq X$ created in the branch step. Depth-first search and best bound search are examples of commonly use node selection strategies. In the depth-first search strategy, we select one of the most recently created nodes. In the breath first strategy, we select an unexplored node with the least lower bound. Commercial solvers implement a combination of both strategies, depending on the problem in consideration. Once a node S is selected, we compute upper and lower bounds for the problem $P(S) := \min\{a^T x + b^T z : (x, z) \in S\}$. For MIP formulations, an optimal solution of $\hat{P}(S) := \min\{a^T x + b^T z : (x, z) \in LP(S)\}$ is

a valid lower bound for $P(S)$. Upper bounds for $P(S)$ are determined from feasible solutions using heuristics, or from $\hat{P}(S)$ if the extreme points satisfy the integrality conditions. If the lower bound obtained from $\hat{P}(S)$ is higher than the best found feasible solution so far, then the node S need not be explored again in subsequent branch phases. From this discussion, the importance of the LP-relaxation of (1.1) is apparent. The stronger the LP-relaxation, the better are the lower bounds computed from $\hat{P}(S)$ which then allows more nodes to be pruned from the search tree and consequently fewer subproblems need to be explored.

1.3 Solving MIPs Approximately

Given a sufficient amount of time, the branch-and-bound algorithm can find an optimal solution of a MIP. Unfortunately, we may need to explore several million nodes before finding even one feasible solution of the MIP. In fields like machine learning, computer vision, advertising, and statistics, it is quite common to encounter MIP formulations with millions of binary decision variables. In these situations, we can trade solution quality for computational complexity by replacing an exact but expensive algorithm (like branch-and-bound) with an approximate but computationally inexpensive algorithm. In some situations, the exact solution may not be any more useful from a practical standpoint than an inexact solution because input data may have noise.

There has been a lot of successful research on finding approximate solutions for MIP problems using heuristic algorithms. These algorithms can yield good results

in practice but do not offer any theoretical bounds on runtime and solution quality. The field of *approximation algorithms* provides a theoretical framework to design and analyze polynomial time algorithms that can find provably approximate solutions of NP-hard problems.

There are many known approximation algorithms for *combinatorial-optimization* problems; a special class of MIP problems defined over sets, graphs and matroids. These algorithms are either combinatorial or LP-based. The combinatorial algorithms include greedy algorithms, randomized algorithms, and reduction to problems with known polytime algorithms. LP-based algorithms broadly fall into two categories; LP-rounding and primal dual algorithms. In LP rounding, we convert an optimal solution of the LP-relaxation to a feasible integral solution of the MIP formulation and in the process, ensure that the total cost of the objective function does not increase too much. Primal-dual algorithms exploit LP-duality by constructing integral solutions to the primal problem and feasible solutions to the dual problem iteratively. The dual feasible solutions are lower bounds on the objective function of the integer program and primal integral solution are upper bounds. Approximation guarantees are established by comparing costs of the dual feasible solution and the primal integral solution. Both LP-based approaches are extensively used to design approximation algorithms for many practical problems. Vazirani [101] provides a comprehensive survey of LP-rounding and primal-dual schemas.

1.4 Lagrangian Relaxation

Lagrangian (sometimes spelled as Lagrangean) relaxation is a popular relaxation technique to compute bounds on the optimal solution of a MIP. We review some basic concepts of the Lagrangian relaxation technique for MIPs. The discussion here is based on the survey paper by Fisher [32] and the textbook by Wolsey [109].

Consider the following MIP formulation

$$\begin{aligned} z^* = \min \quad & c^T x \quad \text{subject to} \\ & Ax \leqslant b, \\ & Cx \leqslant d, \\ & x \in \{0,1\}^n. \end{aligned} \tag{1.7}$$

where $A \in \mathbb{R}^{p \times n}$ and $C \in \mathbb{R}^{m \times n}$. The constraints $Ax \leqslant b$ are *simple* in the sense that

$$z^R = \min c^T x \quad \text{subject to } \; Ax \leqslant b, \; x \in \{0,1\}^n \tag{1.8}$$

can be solved relatively easily. However, the presence of the constraints $Cx \leqslant d$ makes the problem computationally challenging. Since the feasible region of (1.7) is a subset of the feasible region of (1.8), the optimal objective of (1.8) is a bound on the optimal objective of (1.7). However, the relaxation in (1.8) can be very weak. Fortunately, we can do much better than (1.8). Given a set of nonnegative

parameters $\lambda \in \mathbb{R}^m_+$, consider the following optimization problem

$$z(\lambda) = \min c^T x - \lambda^T (d - Cx) \quad \text{subject to } Ax \leqslant b,\ x \in \{0,1\}^n, \tag{1.9}$$

whose optimal solution $z(\lambda)$ is a function of the parameter λ (known as the *Lagrangian multiplier* associated with the constraints $Cx \leqslant d$). For any $\lambda \geqslant 0$, it is easy to see that $z(\lambda) \leqslant z^*$. In (1.9), the *complicating constraints* $Cx \leqslant d$ are handled in the objective as a simple linear penalty term. In order to find the best (largest) possible lower bound $z(\lambda)$ for z^*, we solve the following optimization problem (known as the *Lagrangian dual* problem)

$$w^* = \max_{\lambda \geqslant 0} z(\lambda). \tag{1.10}$$

Let us now try and understand how z^* compares with the best bound w^* attainable by solving the Lagrangian dual. For simplicity, consider a set $X = \{x \in \{0,1\}^n : Ax \leqslant b\}$ which consists of a finite number of feasible points $\{x_1 \dots x_t\}$. In this setting, we can rewrite (1.10) as

$$\begin{aligned} w^* = \max_{\lambda,\eta}\ & \eta \quad \text{subject to} \\ & \eta \leqslant c^T x_i - \lambda(d - Cx_i) \quad i \in \{1 \dots t\} \\ & \eta \in \mathbb{R}, \lambda \in \mathbb{R}^m_+. \end{aligned} \tag{1.11}$$

The optimization problem defined in (1.11) is an LP whose dual can be written as

$$w^* = \min_{\mu} \sum_{i=1}^{t} \mu_i (c^T x_i) \quad \text{subject to}$$
$$\sum_{i=1}^{t} \mu_i (Cx_i - d) \leqslant 0,\ \sum_{i=1}^{t} \mu_i = 1,\ \mu \in \mathbb{R}_+^t, \tag{1.12}$$

which can be rewritten as

$$w^* = \min\ c^T x \quad \text{subject to} \quad Cx \leqslant d,\ x \in \text{conv}(X). \tag{1.13}$$

It can be shown that (1.13) holds for the feasible region $X = \{x \in [0,1]^n : Ax \leqslant b\}$ of any integer program.

Theorem 1.5. $w^* = \min\{c^T x : Cx \leqslant d,\ x \in \textit{conv}(X)\}$ *(Theorem 10.3 of Wolsey [109]).*

The expression (1.13) suggests that in the worst case, if $\text{conv}(X) = \{x \in [0,1]^n : Ax \leqslant b\}$ then the Lagrangian dual is no better than solving the LP-relaxation of (1.7). In addition to the strength of the Lagrangian dual, (1.7) reveals that the structure of the function $z(\lambda)$ is piecewise linear concave. The classical subgradient algorithm due to Polyak [78] for minimizing non-smooth convex functions is one of the most widely used methods to solve the Lagrangain dual problem in (1.10). In Chapter 4, we use Lagrangian relaxation to decompose a single MIP formulation into many smaller MIP formulations that are easier to solve.

1.5 Overview

We provide an overview of the MIP formulations considered in this work. In Chapter 2, we consider MIP formulations for PLFs. There are several well known MIP formulations for modeling PLFs including the specially ordered sets of type II (SOS2) model [10], the incremental model, or delta method (Delta) [64], the multiple choice model (MCM) [50], the convex combination model (CC) [27], and the disaggregated convex combination model (DCC) [67, 105]. The structure of interest, in Chapter 2, is the modelling of PLFs where setting a binary indicator variable to zero forces the argument of the function to zero which in turn forces the function value to be zero. Models with this structure are used in a wide range of applications like gas network optimization [65], transmissions expansion planning [1], sales resources allocation [60], and thermal unit commitment [21]. We describe modifications to standard MIP formulations for PLFs with desirable theoretical properties and superior computational performance in this context.

In Chapter 3, we study a production planning problem, where the production process creates a mixture of useful products and undesirable byproducts. As more of the mixture is produced, the fraction of the mixture that is a useful product decreases monotonically. Conversely, the fraction of each byproduct increases monotonically as a function of cumulative mixture production. Production planning problems with these characteristics arise in engineering applications like the extraction of natural resources such as oil and gas [49, 93, 96, 97] from fields, hydro turbine performance modelling [15, 72], chemical process design [66], and compressor scheduling in petroleum reservoirs [19]. Optimizing production for

problems that contain this structure is complicated because the amount of each product produced is a nonconvex function of the cumulative mixture production up to that time. The problem can be formulated using a nonconvex mixed-integer nonlinear programming (MINLP) model that is hard to solve directly even with state-of-the-art software packages such as Baron [94] or Couenne [12]. Our approach is to develop accurate and computationally useful MIP formulations for this model. A time-discretization of this production planning problem is required to obtain a model that is suitable for implementation and numerical evaluation. Tarhan et al. [93] introduce one such discrete-time MINLP formulation. The main contribution of Chapter 3 is an alternative discrete-time MINLP formulation that is both more accurate and more computationally tractable than the one by Tarhan et al. [93]. We derive three different MIP models that approximate or relax the convex/concave functions. We also demonstrate two techniques for improving the LP-relaxations of the proposed MILP formulations. The first is based on the ideas discussed in Chapter 2 while the second exploits the fact that the cumulative total production, which is the argument to the nonlinear functions being approximated, is increasing over time. An important substructure found in this production model is the use of multiple nonlinear functions that share the same domain. We propose nonlinear programming (NLP) formulations for determining the best possible piecewise-linear approximations and relaxations of multiple univariate convex/-concave functions, with the requirement that the piecewise-linear functions share the same set of break points. We conclude with computational experiments that demonstrate that the proposed formulation is more accurate than the previous for-

mulation [93], and that the strengthened MIP approximation and relaxation models can be used to obtain provably near-optimal solutions for large instances of this nonconvex MINLP. Experiments also illustrate the quality of the piecewise-linear approximations produced by our nonlinear programming formulations.

In strategic planning problems, like the ones discussed in Chapter 3, the modelling of realistic business rules like taxes, tariffs, contracts and royalties can greatly impact decision making. In Chapter 4, we develop optimization models, solution techniques, and algorithms for production planning problems in the presence of a type of fiscal contract called production sharing contracts (PSC) in which the tax incurred by a contractor is a piecewise-constant function of the internal rate-of-return (IRR). An important aspect of PSCs that we address is the presence of *administrative blocks*. These administrative blocks, called *markets* or *ring fences*, are entities in the project whose fiscal calculations are grouped together so that they are completely independent of each other. Problems with this structure arise in applications such as hydrocarbon field infrastructure planning [49, 98, 44, 58, 22], portfolio optimization [77] and production planning [2, 86, 91]. The challenge while modelling IRR-based PSCs is that the objective function is a nonconvex, discontinuous function of the decision variables. We propose two different solution techniques. The first is a MIP formulation that eliminates the inherent nonlinearity present in IRR-based PSCs. However, this MIP formulation has a weak LP-relaxation because it requires variable bounds that are difficult to estimate. The second solution technique is a search algorithm based on a novel continuous domain formulation. Motivated by the observation that our MIP formulation can be effectively used to solve problems

involving a single market, we propose a market-based decomposition scheme to compute bounds on the optimal solution. We conclude this chapter with computational experiments that demonstrate the impact of our formulations, solution techniques and decomposition algorithms on a sample application problem.

In Chapter 5, we study the use of LP-rounding in approximating solutions for NP-hard combinatorial optimization problems such as set cover, set packing, and multiway-cut. LP-rounding based approximation schemes have been successfully used for a wide range of NP-hard problems in applications like machine learning [85, 112, 57, 84], computer vision [53, 16, 24], natural language processing [14, 54] and statistics [7, 99]. In this Chapter, we first show that one can recover solutions of comparable quality by rounding an *approximate* LP solution instead of the exact one. Our intuition is that in LP-rounding, since we ultimately round the LP to obtain an approximate solution of the combinatorial problem, a crude solution of the LP may suffice. Hence, an approach that can find approximate solutions of large LPs quickly may be suitable, even if it is inefficient for obtaining highly accurate solutions. We then build a solver, called Thetis, that can compute approximate LP solutions efficiently by applying an asynchronous parallel stochastic-coordinate descent (SCD) method to a quadratic-penalty formulation of the LP. The SCD algorithm [69, 95, 11, 111] is suitable for solving large-scale optimization problems because each iteration of the algorithm modifies a single coordinate and requires only the direction of the gradient along that coordinate. Recent work on asynchronous parallel versions of SCD [70, 59] make the entire LP-rounding scheme suitable for execution on multi core, shared-memory architectures. We derive worst-case

runtime and solution quality guarantees for our solver using novel perturbation and convergence analysis. Our experiments demonstrate that on such combinatorial problems as vertex cover, independent set and multiway-cut, our approximate rounding scheme is up to an order of magnitude faster than Cplex (a commercial LP solver) while producing solutions of similar quality.

In this dissertation, we focus on the *two different ways to model the combinatorial world*. One in which, we require extensive computational effort in providing exact solutions to accurate models of practical problems and another in which inaccurate or approximate solutions are sufficient because the models are fundamentally inexact. Chapters 2-4 focus on the former modelling paradigm while Chapter 5 focusses on the latter. on

2 LOCALLY IDEAL FORMULATIONS FOR PIECEWISE LINEAR FUNCTIONS WITH INDICATOR VARIABLES

In this chapter, we consider MIP formulations for piecewise linear functions (PLFs) that are evaluated when an indicator variable is turned on. We describe modifications to standard MIP formulations for PLFs with desirable theoretical properties and superior computational performance in this context. The results in this chapter appear in Sridhar et al. [90].

2.1 Introduction

Optimization problems involving PLFs appear in a wide range of applications. PLFs are frequently used to approximate nonlinear functions and to model cost functions involving economies of scale and fixed charges. Problems involving nonconvex PLFs are commonly formulated as MIP problems [10, 64, 27, 3, 102]. Consider a univariate PLF $f : [B_0, B_n] \to \mathbb{R}$ with its domain $[B_0, B_n]$ divided into an increasing sequence of breakpoints $\{B_0, B_1, \ldots, B_n\}$. For simplicity, we assume that $f(\cdot)$ is continuous, $B_0 = 0$ and $f(0) = 0$. Our results can be extended to the case when $f(\cdot)$ is lower semi-continuous, $B_0 \neq 0$, and $f(B_0) \neq 0$. The function $f(\cdot)$ can be written using

$$f(x) := m_i x + c_i, \quad x \in [B_{i-1}, B_i] \quad \forall i \in \{1 \ldots n\} \tag{2.1}$$

where $m_i \in \mathbb{R}, c_i \in \mathbb{R}$ and $B_0 < B_1 < \cdots < B_n$. Methods for modeling PLFs include specially ordered sets of type II (SOS2) [10], the incremental model, or delta

method (Delta) [64], the multiple choice model (MCM) [50], the convex combination (CC) model [27], the disaggregated convex combination model (DCC) [67], and approaches that require only logarithmically many binary variables [105].

In this chapter, we present MIP formulations for PLFs where setting a binary indicator variable to zero forces the argument of the function of $f(\cdot)$ to zero which in turn forces the function to take a zero value. In other words,

$$z = 0 \Rightarrow x = 0, f(x) = 0. \tag{2.2}$$

The goal of this work is to present a theoretical and computational comparison of MIP formulations that enforce the logical conditions in (2.2). Specifically, we examine properties of different formulations of the three variable set

$$X := \bigcup_{i=1}^{n} \{(x, y, z) : x \in [B_{i-1}, B_i], y = m_i x + c_i, z = 1\} \bigcup \{(0, 0, 0)\}. \tag{2.3}$$

In some applications, notably those where the PLF appears in a minimization objective, the relevant set to study has the variable y constrained to lie in the epigraph of a convex function. We denote

$$X^{\geqslant} := \bigcup_{i=1}^{n} \{(x, y, z) : x \in [B_{i-1}, B_i],\ y \geqslant m_i x + c_i, z = 1\} \bigcup \{(0, 0, 0)\}.$$

as the set where the equality relationship in (2.3) is replaced with $y \geqslant m_i x + c_i$. The results we derive will apply equally well to the case where the y is forced to lie exactly on $f(\cdot)$ i.e $y = m_i x + c_i$ when $x \in [B_{i-1}, B_i]$.

Background

MIP formulations for piecewise linear functions with indicator variables arise in various applications including gas network optimization [65], production planning [93, 91], transmissions expansion planning [1], oil field infrastructure development [49, 42], sales and advertising budget allocation [88, 60, 115], unit commitment problems [15, 33, 21] as well as in substructures used by general purpose MINLP algorithms [26]. We explain using three specific models where the structure in (2.2) has been modeled.

Gas Network Optimization. In the field of gas network optimization, Martin et al. [65] proposed MIP formulations to help design cost effective gas networks that satisfy consumers with demands for gas at a certain pressure. The gas network is modeled as a directed graph $G = (V, E)$ where edges E are valves/compressors that regulate pressure and the nodes V are either sources/consumers of gas. The compressors help maintain gas pressure but consume a fraction of the gas flowing through the network. PLFs are used to approximate the *fuel-gas consumption* of a compressor $e \in E$ between nodes (u, v) given by

$$f(p_u, p_v, q_e) = \gamma\left(\left(\frac{p_u}{p_v}\right)^{\frac{\kappa-1}{\kappa}} - 1\right) q_e$$

where $q_e \in \mathbb{R}_+$ is the gas volume flow in each edge $e \in E$, $p_v \in \mathbb{R}_+$ is the pressure of gas at each node $v \in V$ and κ,γ are constants. Binary variables s_e determine if a compressor in each edge $e \in E$ is switched on. If the compressor in a segment is turned off ($s_e = 0$), then the gas volume flow q_e, and the fuel-gas consumption

$f(p_u, p_v, q_e)$ are both zero.

Sales Resource Allocation. In another application related to sales/advertising, several authors [88, 60, 115] have formulated sales resource allocation problems using MIPs. The goal is to maximize a nonconvex return on investment (ROI) function while satisfying budget constraints. In these models, a company is required to allocate a budget B among several sales entities each with several choices of implementation strategies. The company incurs a fixed cost for using each sales entity and an operational cost depending on the implementation strategy chosen. Figure 2.1 illustrates a commonly used nonconvex PLF (called S-curves) used to model the ROI from different sales entities. Again, we observe the structure that a binary variable (choosing a sales entity) determines whether or not a PLF (contribution to the ROI) must be evaluated.

Global Optimization of MINLPs. In a third application, D'Ambrosio et al. [26] proposed a global optimization algorithm for mixed integer nonlinear programs with separable nonconvexity. As a substructure in their algorithm, the authors explicitly model each univariate nonconvex function $g : \mathbb{R} \to \mathbb{R}$ as a piecewise convex/concave function which are defined using ordered set of breakpoints $P_0 < P_1 \ldots < P_{\bar{p}}$ representing the points where the convexity/concavity of $g(\cdot)$ changes. For each interval $p \in \{1, \ldots \bar{p}\}$ the convex/concave piece $g_p : \mathbb{R} \to \mathbb{R}$ is modeled using PLF with a binary variable z_p to determine if the piece contributes to the original function $g(\cdot)$. This substructure is another example of the use of the set X while modeling applications involving PLFs who's evaluation depends on the state

of a binary variable.

Table 2.1: Applications Using PLFs with Indicator Variables.

PLF Model Used	Application	Reference
SOS2	Gas network optimization	[65]
Delta	Transmissions expansion planning	[1]
Delta	Thermal unit commitment	[21]
CC	Oil field development	[42]
CC	Hydro Scheduling	[15]
MCM	Sales resource allocation	[60]

PLFs with indicator variables arise in many other settings. Table 2.1 lists several applications in the literature that have modeled PLFs using standard MIP formulations for PLFs in conjunction with variable upper bound constraints of the form

$$x \leqslant B_n z \tag{2.4}$$

to enforce the logical on-off condition (2.2).

Main Results

In this work, we propose a simple modeling artifice for PLFs that also enforces the logical condition (2.2), and we demonstrate its desirable theoretical and computational properties. We start by describing the idea using SOS2 to model a PLF

as

$$x = \sum_{i=0}^{n} \lambda_i B_i, \tag{2.5}$$

$$y = \sum_{i=0}^{n} \lambda_i F_i, \tag{2.6}$$

$$1 = \sum_{i=0}^{n} \lambda_i, \tag{2.7}$$

$$\boldsymbol{\lambda} := \left\{ \lambda_i \in \mathbb{R}_+ : \ \forall i \in \{0 \dots n\} \right\} \text{ is SOS2.}$$

where $F_i := f(B_i) = m_i B_i + c_i \ \forall i \in \{0 \dots n\}$. In this formulation, the function $f(\cdot)$ and its argument x are expressed as convex combinations of breakpoints $\mathcal{B} := \{B_0 \dots B_n\}$ and their corresponding function evaluations $\{F_0 \dots F_n\}$ respectively. The formulation introduces a non-negative set of variables $\boldsymbol{\lambda} \in \mathbb{R}^{n+1}$ that satisfy the SOS2 property—at most two of the variables can be positive, and if two variables are positive then they must be consecutive in the ordered set. Most modern general purpose MIP solvers enforce the SOS2 condition algorithmically by branching [10].

Using variable upper bound constraints (2.4) to enforce the logical condition (2.2) has two problems. First, the use of "bigM" constraints may considerably weaken the LP relaxation of the MIP formulation. Second, the model introduces an additional constraint $x \leqslant B_n z$. We propose the following simple strengthening that replaces $x \leqslant B_n z$ and $\sum_{i=0}^{n} \lambda_i = 1$ with

$$\sum_{i=0}^{n} \lambda_i = z. \tag{2.8}$$

Setting the binary variable $z = 0$ in (2.8) forces $\lambda_i = 0 \ \forall i \in \{0 \dots n\}$, which in turn

forces forces the function to take a zero value. If the binary variable $z = 1$, then $\sum_{i=0}^{n} \lambda_i = 1$, which reduces to (2.7). We show in Section 2.2 that a formulation using (2.8) has the desirable property of being *locally ideal*, while one that uses $x \leqslant B_n z$ does not. Borghetti et al. [15] created a formulation of X that employed the strengthening techniques we describe. They used the convex combination method to model the PLFs which does not have the locally ideal property [102].

Contributions

In this work, we focus on MIP formulations for PLFs with indicator variables; a general class of problems that occur in various applications. We highlight four important contributions of this work.

First, we present tight formulations for X by extending existing MIP formulations for PLFs using the SOS2 model. Our formulations have theoretical grounding in addition to superior computational performance in comparison with standard textbook models.

As a second contribution, we study many theoretical properties relating to the quality and tightness of our MIP formulations and compare them with previous work. We show SOS2 models for PLFs that use (2.8) are *locally ideal*. Desirable local properties imply that our formulations, in the absence of other constraints, model the set X perfectly. We also show that formulations that use the SOS2 model in conjunction with constraints (2.4) are *not locally ideal*. Finally, we show that for the special case of convex PLFs, our formulations are equivalent to the perspective reformulation of convex sets [23]. Hence our model achieves the same strength as

the perspective reformulation for convex PLFs. Such a theoretical understanding of the quality and tightness of MIP formulations plays an important role when the resulting problems are solved using general purpose MIP solvers.

As a third contribution, we apply the same formulation strengthening to several MIP formulations of PLFs including the incremental model, the multiple choice model [3], the disaggregated convex combination model [102, 105], and several logarithmic models [102]. Therefore, this formulation strengthening technique could be directly applied to all of the applications listed in Table 2.1. In all cases, we show that our model retains the desirable theoretical property of the underlying PLF modeling method, either idealness or sharpness, but using a variable upper bound constraint in (2.4) destroys the property.

Finally, we conduct a computational study to illustrate the benefits of the new formulations. In our experiments, we observed that our formulation compute optimal solutions on average 40x faster, explore 15x fewer nodes and produce LP relaxations that are 20% closer to the optimal solution.

The rest of this chapter is organized as follows. In Section 2.2, we study the theoretical properties of SOS2 models for X. In Section 2.2, we illustrate how our formulation strengthening can be applied to various other models of PLFs. We conclude, in Section 2.3 with numerical experiments that illustrate the impact of our formulation strengthening.

2.2 Properties of MIP formulations

Padberg and Rijal [74] define a *locally ideal* MIP formulation as one where the vertices of its corresponding LP relaxation satisfy all required integrality conditions. Extending this definition, Croxton et al. [25] and Keha et al. [51] define a locally ideal SOS2 formulation as one whose LP relaxation has extreme points that all satisfying the SOS2 property. As shown by Vielma et al. [102], all commonly used MIP formulations of PLFs, except for the original convex combination (CC) model, are known to be locally ideal. In this section, we demonstrate the same theoretical strength of our proposed formulations for X that include the logical condition (2.2).

SOS2 Model

We consider the following two SOS2-based formulations for X:

$$S_1 := \Big\{(x, y, \boldsymbol{\lambda}, z) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}_+^{n+1} \times \{0,1\} : x = \sum_{i=0}^{n} B_i \lambda_i,\ y = \sum_{i=0}^{n} F_i \lambda_i,\ 1 = \sum_{i=0}^{n} \lambda_i,\ x \leqslant B_n z,\ \boldsymbol{\lambda} \text{ is SOS2} \Big\}$$

$$S_2 := \Big\{(x, y, \boldsymbol{\lambda}, z) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}_+^{n+1} \times \{0,1\} : x = \sum_{i=0}^{n} B_i \lambda_i,\ y = \sum_{i=0}^{n} F_i \lambda_i,\ z = \sum_{i=0}^{n} \lambda_i,\ \boldsymbol{\lambda} \text{ is SOS2} \Big\}$$

where S_1 is a standard SOS2 model for PLFs that uses the constraint (2.4), while formulation S_2 uses the constraint (2.8) to model the logical condition (2.2). One can easily show that both S_1 and S_2 are valid formulations of X. In other words, for

either $T = S_1$ or $T = S_2$,

$$X = \Big\{(x, y, z) : \exists \boldsymbol{\lambda} \in [0,1]^{n+1} \text{ s.t } (x, y, z, \boldsymbol{\lambda}) \in T\Big\}.$$

We use the standard definition of the LP-relaxation of a model as the relaxation obtained by replacing integrality restrictions on variables with simple bound restrictions and by removing adjacency requirements for SOS2 variables. We now prove that the formulation S_2 is locally ideal while S_1 is not.

Theorem 2.1. *Formulation S_2 is locally ideal.*

Proof. The LP relaxation of S_2 has $n + 4$ variables, three equality constraints

$$x = \sum_{i=0}^{n} B_i \lambda_i,\ y = \sum_{i=0}^{n} F_i \lambda_i,\ z = \sum_{i=0}^{n} \lambda_i,$$

and $n + 2$ inequality constraints, $z \leqslant 1$ and $\lambda_i \geqslant 0\ \forall i \in \{0 \dots n\}$. Extreme points of the LP relaxation of S_2 have $n + 4$ binding constraints, which forces at least n variables from $\boldsymbol{\lambda} \in \mathbb{R}_+^{n+1}$ to be exactly equal to zero. Thus, the extreme points of the LP relaxation of S_2 are

$$\{(x = B_i,\ y = F_i,\ \boldsymbol{\lambda} = B_i \vec{e}_i,\ z = 1)\ \ \forall i \in \{1 \dots n\}\} \bigcup (x = 0, y = 0, \boldsymbol{\lambda} = \vec{0}, z = 0) \tag{2.9}$$

where $\vec{e}_i$ are the n dimensional unit vectors. All points in (2.9) have $z \in \{0, 1\}$ and satisfy the SOS2 properties for the $\boldsymbol{\lambda}$ variables. Hence, S_2 is locally ideal. □

A point $(x, y, \boldsymbol{\lambda}, z)$ can only be an extreme point of the set

$$P_2^{\geqslant} := \left\{(x, y, \boldsymbol{\lambda}, z) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}_+^{n+1} \times [0,1] : \; x = \sum_{i=0}^{n} B_i \lambda_i, \; y \geqslant \sum_{i=0}^{n} F_i \lambda_i, \; z = \sum_{i=0}^{n} \lambda_i \right\}$$

if $y = \sum_{i=0}^{n} F_i \lambda_i$. Therefore, the proof of Theorem 2.1 also establishes that expressing logical condition (2.2) using (2.8) also results in a locally ideal formulation of $X^{\geqslant}$. Similar logic applies in our subsequent proofs of the local idealness of other formulations of X (Theorems 2.4 and 2.6). In each case, our proposed modeling of the logical condition (2.2) also yields a locally ideal formulation of $X^{\geqslant}$.

Theorem 2.2. *Formulation S_1 is not locally ideal.*

Proof. Consider an instance with $n = 3$, $\mathbf{B} = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$, and $\mathbf{F} = \{0, 4, 2, 3\}$. The point $\{x = \frac{1}{3}, y = 4, \boldsymbol{\lambda} = (0, 1, 0, 0), z = \frac{1}{3}\}$ is feasible to the LP relaxation of S_1 but not feasible for the LP relaxation of S_2. The point $\{x = \frac{1}{3}, y = 4, \boldsymbol{\lambda} = (0, 1, 0, 0), z = \frac{1}{3}\}$ is an extreme point of the LP relaxation of S_1 since it satisfies the $n + 4 = 7$ equations:

$$\lambda_0 = 0, \lambda_2 = 0, \lambda_3 = 0, x = B_3 z, x = \sum_{i=0}^{3} \lambda_i B_i, \; y = \sum_{i=0}^{3} F_i \lambda_i, \text{ and } \sum_{i=0}^{3} \lambda_i = 1.$$

The value of the binary variable $z = 1/3$ is not integral in this extreme point. □

An interesting consequence of Theorem 2.1 is that when the PLF is convex, the application of the reformulation technique we suggest to the set $X^{\geqslant}$ is equivalent to the *perspective reformulation* [41], a preprocessing technique for (convex) mixed integer nonlinear programs that have the logical indicator structure (2.2). If $f(\cdot)$ is

convex, then $m_1 > m_2 > \ldots > m_n$, and the perspective reformulation of $X^{\geqslant}$ is

$$P = \{(x, y, z) \in \mathbb{R}^2 \times [0, 1] : y \geqslant m_i x + c_i z \ \ \forall i \in \{1 \ldots n\},\ 0 \leqslant x \leqslant B_n z\},$$

where $m_i := (F_i - F_{i-1})/(B_i - B_{i-1})$ and $c_i := (F_{i-1} - B_{i-1}(F_i - F_{i-1})/(B_i - B_{i-1}))$. Günlük and Linderoth [41] show that if $f(\cdot)$ is convex, then $P = \text{conv}(X^{\geqslant})$. The formulation S_2 is locally ideal, so $P_2^{\geqslant}$ must also be a formulation that is similarly strong.

Corollary 2.3. *$Proj_{xyz}(P_2^{\geqslant}) = P = conv(X^{\geqslant})$.*

Incremental Model

The incremental method (sometimes referred to as the Delta method) was first introduced by Markowitz and Manne [64]. Several articles [27, 73, 51] have studied the polyhedral properties of the incremental model. The incremental model introduces a set of non-negative variables $\delta := \{\delta_1, \ldots, \delta_n\}$ to model the portion of each interval "filled" by the variable x. The interval $i + 1$ can be filled ($\delta_{i+1} > 0$) only if the interval i is already filled ($\delta_i = 1$). Unlike the SOS2 model, the incremental model specifically requires the introduction of binary variables $b \in \{0, 1\}^{n-1}$ to enforce the necessary ordering conditions. To model the on-off logical condition (2.2), the incremental model can be augmented with a variable upper bound constraint

$x \leqslant B_n z$, resulting in a formulation

$$\Delta_1 := \Big\{(x, y, \boldsymbol{\delta}, z, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \times [0,1] \times [0,1]^{n-1} \ : \\ x = \sum_{i=1}^{n} [B_i - B_{i-1}]\delta_i,\ y = \sum_{i=1}^{n} [F_i - F_{i-1}]\delta_i,\ x \leqslant B_n z, \\ \delta_1 \leqslant 1,\ 0 \leqslant \delta_n,\ \delta_{i+1} \leqslant b_i \leqslant \delta_i\ \forall i \in \{1 \dots n-1\}\Big\}.$$

Alternatively, the on-off condition can be enforced by replacing the constraint $\delta_1 \leqslant 1$ with $\delta_1 \leqslant z$, yielding the formulation

$$\Delta_2 := \Big\{(x, y, \boldsymbol{\delta}, z, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \times [0,1] \times [0,1]^{n-1} \ : \\ x = \sum_{i=1}^{n} [B_i - B_{i-1}]\delta_i,\ y = \sum_{i=1}^{n} [F_i - F_{i-1}]\delta_i, \\ \delta_1 \leqslant z,\ 0 \leqslant \delta_n,\ \delta_{i+1} \leqslant b_i \leqslant \delta_i\ \forall i \in \{1 \dots n-1\}\Big\}.$$

Incremental models that use $\delta_1 \leqslant z$ are locally ideal, while those that use $x \leqslant B_n z$ are not.

Theorem 2.4. *Formulation Δ_2 is locally ideal.*

Proof. We prove that D_2 is locally ideal i.e the extreme points of the LP relaxation of $D_2 := LP(\Delta_2)$ satisfies the integrality for variables z and $\mathbf{b}$. The set D_2 is defined

using $2n+2$ variables with $2n+2$ inequality constraints given by

$$\begin{aligned}
a_1 &:= 1 - z && \geqslant 0, \\
a_2 &:= z - \delta_1 && \geqslant 0, \\
a_{2+i} &:= \delta_i - b_i && \geqslant 0 \quad \forall i \in \{1 \dots n\}, \\
a_{n+1+i} &:= b_i - \delta_{i+1} && \geqslant 0 \quad \forall i \in \{1 \dots n\}, \\
a_{2n+2} &:= \delta_n && \geqslant 0
\end{aligned} \tag{2.10}$$

along with the two equality constraints relating variables x, y and $\boldsymbol{\delta}$. Hence every extreme point of D_2 is defined by the intersection of $2n+2$ constraints. Since there are two equality constraints, at most one inequality from the system (2.10) can be strict. Since $\sum_{i=1}^{2n+2} a_i = 1$ and each $a_i \geqslant 0$, the additional condition that there can be at most one strict inequality implies that inequalities of the set D_2 satisfy $a_k = 1, a_i = 0 \; \forall i \neq k$. From this, we conclude that each extreme point of D_2 satisfies $(z, \mathbf{b}) \in [0,1]^{n+1}$ which makes the formulation Δ_2 locally ideal. □

Theorem 2.5. *Formulation Δ_1 is not locally ideal.*

Proof. Consider an instance with $n = 3$, $\mathbf{B} = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$ and $f(\mathbf{B}) = \{0, 4, 2, 3\}$. The fractional point $\{x = \frac{1}{3}, y = 4, \delta = (1,0,0), z = \frac{1}{3}, b = (0,0)\}$ is feasible to the LP relaxation of Δ_1 but not feasible for the LP relaxation of Δ_2. The fractional point is an extreme point for the LP-relaxation of Δ_1, since it satisfies the following $2n + 2 = 8$

equalities

$$x = z,\ \delta_1 = 1,\ \delta_3 = 0,\ b_2 = \delta_2,\ \delta_2 = b_1,\ \delta_3 = b_2,$$
$$x = \frac{1}{3}\delta_1 + \frac{1}{3}\delta_2 + \frac{1}{3}\delta_3, \text{ and } y = 4\delta_1 - 2\delta_2 + \delta_3.$$

□

Multiple choice model

The multiple choice model for PLFs was introduced and analyzed in Jeroslow and Lowe [50] and Balakrishnan and Graves [3]. As discussed in Section 1.1, in this model, a non-negative set of variables $\boldsymbol{w} := \{w_1, \dots w_n\}$ and an additional set of binary variable $\mathbf{b} := \{b_1, \dots b_n\}$ are introduced, with the logical implication that $w_i = x$ if x is in the ith interval, and $w_i = 0$ otherwise. Using a variable upper bound constraint to enforce the logical condition (2.2) with the multiple choice model gives the following formulation of X:

$$M_1 := \Big\{(x, y, \boldsymbol{w}, z, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \times [0,1] \times [0,1]^n : \sum_{i=1}^{n} w_i = x,\ y = \sum_{i=1}^{n} (m_i w_i + c_i b_i),\ x \leqslant B_n z,$$
$$\sum_{i=1}^{n} b_i = 1,\ \ B_{i-1} b_i \leqslant w_i \leqslant B_i b_i\ \forall i \in \{1 \dots n\}\Big\}.$$

Instead, the on-off condition can be formulated by replacing the constraints $\sum_{i=1}^{n} b_i = 1$ with $\sum_{i=1}^{n} b_i = z$, yielding a formulation

$$M_2 := \Big\{ (x, y, \boldsymbol{w}, z, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \times [0,1] \times [0,1]^n : \sum_{i=1}^{n} w_i = x,\ y = \sum_{i=1}^{n} (m_i w_i + c_i b_i), \sum_{i=1}^{n} b_i = z,\ \ B_{i-1} b_i \leqslant w_i \leqslant B_i b_i \ \forall i \in \{1 \dots n\} \Big\}.$$

Theorem 2.6. *Formulation M_2 is locally ideal.*

Proof. Following Balas [4], we write an extended formulation for the convex hull of the union of the $n+1$ polytopes

$$X_0 = \{(0,0,0)\},$$
$$X_i = \{(x,y,z) : B_{i-1} \leqslant x \leqslant B_i, y = m_i x + c_i, z = 1\} \ \forall i \in \{1 \dots n\}$$

as those (x, y, z) for which there exist vectors $\boldsymbol{w} = [w_0, \dots w_n], \boldsymbol{v} = [v_0, \dots v_n], \mathbf{u} =$

$[u_0, \dots u_n], \mathbf{b} = [b_0, \dots b_n]$ such that the following inequality system is satisfied:

$$x = \sum_{i=0}^{n} w_i,\ y = \sum_{i=0}^{n} v_i,\ z = \sum_{i=0}^{n} u_i,\ 1 = \sum_{i=0}^{n} b_i,$$

$$w_0 = 0,\ v_0 = 0,\ u_0 = 0,$$

$$b_i \geqslant 0 \qquad \forall i \in \{0 \dots n\},$$

$$B_{i-1} b_i \leqslant w_i \leqslant B_i b_i \qquad \forall i \in \{1 \dots n\},$$

$$v_i = m_i w_i + c_i b_i \qquad \forall i \in \{1 \dots n\},$$

$$u_i = b_i \qquad \forall i \in \{1 \dots n\}.$$

We can eliminate $b_0, \mathbf{u}$, and $\boldsymbol{v}$ from this system to obtain

$$x = \sum_{i=1}^{n} w_i,\ y = \sum_{i=1}^{n} (m_i w_i + c_i b_i),\ z = \sum_{i=1}^{n} b_i,\ z \leqslant 1,$$

$$b_i \geqslant 0 \qquad \forall i \in \{1 \dots n\},$$

$$B_{i-1} b_i \leqslant w_i \leqslant B_i b_i \qquad \forall i \in \{1 \dots n\},$$

which is equivalent to the LP relaxation of M_2. □

Theorem 2.7. *Formulation* M_1 *is not locally ideal.*

Proof. Consider an instance with $n = 3$, $\mathbf{B} = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$, and $f(\mathbf{B}) = \{0, 4, 2, 3\}$. The point $\{x = \frac{1}{3}, y = 4, w = (0, \frac{1}{3}, 0), z = \frac{1}{3}, b = (0, 1, 0)\}$ is feasible to the linear programming relaxation of M_1, but not feasible for M_2. A fractional extreme point of the linear programming relaxation of M_1 is $\{z = \frac{1}{3}, y = 4, x = \frac{1}{3}, w = (0, \frac{1}{3}, 0), b = (0, 1, 0)\}$. The point satisfies the $2n + 3 = 9$ equations $x = w_1 + w_2 + w_3$,

$y = 12w_1 - 6w_2 + 6b_2 + 3w_3$, $x = z$, $b_1 + b_2 + b_3 = 1$, $0 = w_1$, $w_1 = \frac{1}{3}b_1$, $\frac{1}{3}b_2 = w_2$, $\frac{2}{3}b_3 = w_3$, and $w_3 = b_3$. □

Convex Combination Model

Another popular formulation for PLFs is the convex combination model, also known as the lambda method. As discussion in Section 1.1, the convex combination model uses continuous variables $\boldsymbol{\lambda} \in \mathbb{R}^{n+1}$ and binary variables $\mathbf{b} \in [0,1]^n$. The continuous variables are used to express x and y in terms of the breakpoints $\mathbf{B}$ and function values $\mathbf{F}$. The binary variables are used to enforce the adjacency condition that $b_i = 1 \Rightarrow \lambda_j = 0, \forall j \notin \{i-1, i\}$. Using a variable upper bound to model the logical on-off condition (2.2) in combination with the most commonly used convex combination model gives the following formulation of X:

$$C_1 := \Big\{(x, y, \boldsymbol{\lambda}, z, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}_+^{n+1} \times [0,1] \times [0,1]^n : x = \sum_{i=0}^{n} \lambda_i B_i,\ y = \sum_{i=0}^{n} \lambda_i F_i,\ x \leqslant B_n z,\ \sum_{i=0}^{n} \lambda_i = 1,\ \sum_{i=1}^{n} b_i = 1,\ \lambda_0 \leqslant b_1,\ \lambda_n \leqslant b_n,\ \lambda_i \leqslant b_i + b_{i+1}\ \forall i \in \{1 \dots n-1\}\Big\}.$$

Instead, the on-off condition can be directly imposed by replacing $\sum_{i=1}^{n} b_i = 1$ and $\sum_{i=0}^{n} \lambda_i = 1$ with the constraints $\sum_{i=1}^{n} b_i = \sum_{i=0}^{n} \lambda_i = z$. This gives the following

formulation of X:

$$C_2 := \Big\{(x, y, \boldsymbol{\lambda}, z, \mathbf{b}) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}_+^{n+1} \times [0,1] \times [0,1]^n : \\ x = \sum_{i=0}^{n} \lambda_i B_i,\ y = \sum_{i=0}^{n} \lambda_i F_i,\ \sum_{i=0}^{n} \lambda_i = z,\ \sum_{i=1}^{n} b_i = z, \\ \lambda_0 \leqslant b_1,\ \lambda_n \leqslant b_n,\ \lambda_i \leqslant b_i + b_{i+1}\ \forall i \in \{1 \dots n-1\}\Big\}.$$

It has been shown by Padberg [73] and Lee and Wilson [56] that the convex combination model that uses the constraints

$$\lambda_0 \leqslant b_1, \lambda_n \leqslant b_n, \lambda_i \leqslant b_i + b_{i+1}\ \forall i \in \{1 \dots n-1\} \tag{2.11}$$

to model adjacency is *not* locally ideal. Padberg [73] gives the following improved formulation of the adjacency conditions:

$$\sum_{i=j}^{n} \lambda_i \leqslant \sum_{i=j}^{n} b_i,\ \sum_{i=0}^{j-1} \lambda_i \leqslant \sum_{i=1}^{j} b_i\ \forall j = 1, \dots, n,$$

which does result in a locally ideal formulation of PLFs. However, in most presentations of the convex combination model in the literature [27, 15, 42, 36] the non-ideal formulation (2.11) is used. The convex combination model with constraints (2.11) does not result in a formulation that is locally ideal, however it does satisfy *sharpness*, a slightly weaker desirable property. An extended MIP formulation of a convex set is *sharp* if the extreme points of the projection of the LP relaxation of the formulation to the original space of variables (x, y, z in this case) satisfy integrality [50].

Vielma et al. [102] showed that the convex combination model that uses adjacency constraint (2.11) is sharp. We now show that the formulation C_2 is sharp while C_1 is not sharp.

Theorem 2.8. *Formulation* C_2 *is sharp.*

Proof. Suppose that $t = (x, y, \boldsymbol{\lambda}, z, \mathbf{b})$ is an extreme point of the linear programming relaxation of C_2 with $0 < z < 1$. For $\epsilon > 0$ define the points $t^+ = (x^+, y^+, \boldsymbol{\lambda}^+, z^+, \mathbf{b}^+)$ and $t^- = (x^-, y^-, \boldsymbol{\lambda}^-, z^-, \mathbf{b}^-)$ as

$$b_i^+ = (1+\epsilon)b_i, \lambda_i^+ = (1+\epsilon)\lambda_i,\ \forall i \in \{1, \dots, n\}$$
$$z^+ = (1+\epsilon)z,\ x^+ = \sum_{i=0}^{n} \lambda_i^+ B_i,\ y^+ = \sum_{i=0}^{n} \lambda_i^+ F_i$$

$$b_i^- = (1-\epsilon)b_i, \lambda_i^- = (1-\epsilon)\lambda_i,\ \forall i \in \{1, \dots, n\}$$
$$z^- = (1-\epsilon)z,\ x^- = \sum_{i=0}^{n} \lambda_i^- B_i,\ y^- = \sum_{i=0}^{n} \lambda_i^- F_i.$$

For some $\epsilon > 0$, the points t^+, t^- are both feasible for the linear programming relaxation of C_2, and $t = 0.5(t^+ + t^-)$, so t must not have been an extreme point. □

Theorem 2.9. *Formulation* C_1 *is not sharp.*

Proof. Consider an instance with $n = 3$, $\mathbf{B} = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$, and $f(\mathbf{B}) = \{0, 4, 2, 3\}$. One can verify that one of extreme points of the projection of the linear programming relaxation of C_1 is $\{x = \frac{1}{3}, y = 4, z = \frac{1}{3}\}$, which does not satisfy the required

integrality constraint on z. One can easily verify (using porta) that the presented point is an extreme point of the corresponding LP-relaxation. □

Other formulations

The disaggregated convex combination model for PLFs uses two sets of non-negative variables $\boldsymbol{\lambda} := \{\lambda_i \; \forall i \in \{1 \dots n\}\}$ and $\boldsymbol{\mu} := \{\mu_i \; \forall i \in \{1 \dots n\}\}$ and a set of binary variables $\mathbf{b} := \{b_i \; \forall i \in \{1 \dots n\}\}$. The disaggregated convex combination model for a PLF is

$$y = \sum_{i=1}^{n} (\lambda_i F_i + \mu_i F_{i-1}), \tag{2.12}$$

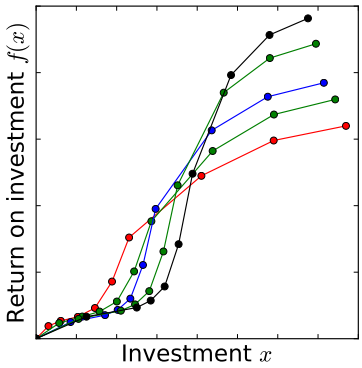
$$x = \sum_{i=1}^{n} (\lambda_i B_i + \mu_i B_{i-1}),$$

$$1 = \sum_{i=1}^{n} b_i, \tag{2.13}$$

$$b_i = \lambda_i + \mu_i \;\; \forall i \in \{1 \dots n\}. \tag{2.14}$$

This formulation can be extended to model X by replacing the constraints $\sum_{i=1}^{n} b_i = 1$ with $\sum_{i=1}^{n} b_i = z$. Disaggregated convex combination models that use these constraints are a locally ideal formulation of X. Vielma and Nemhauser [105] modify the disaggregated convex combination model to use a logarithmic number of binary variables. Using notation defined in Vielma and Nemhauser [105], replacing $\sum_{i=1}^{n} \lambda_i = 1$ with $\sum_{i=1}^{n} \lambda_i = z$ is a valid locally ideal reformulation of model X. For the sake of brevity, we have omitted detailed discussions and proofs concerning disaggregated convex combination models.

Figure 2.1: Sample curves modeling ROI for five different product/strategy pairs.



2.3 Computational Results

In this section, we illustrate with numerical experiments the impact of using locally ideal formulations (S_2 and Δ_2) instead of a weaker model (S_1 and Δ_1) that is not locally ideal.

Sample Application

To make the numerical comparison, we consider an advertising budget allocation problem introduced by Zoltners and Sinha [115]. In this problem, a company is required to allocate an advertising budget D among a set $\mathcal{K}$ of advertising strategies for a set of $\mathcal{P}$ products. Let x_{jk} denote the amount of the advertising resource allocated to strategy $k \in \mathcal{K}$ for product $j \in \mathcal{J}$. The company incurs a fixed cost G_j for entering the market with product $j \in \mathcal{J}$ as well as a variable cost c_{jk} for each unit of the resource allocated to strategy $k \in \mathcal{K}$ of product $j \in \mathcal{J}$. The ROI is evaluated by piecewise-linear functions $y_{jk} = f_{jk}(x_{jk})$ which have the typical form shown in

Table 2.2: Summary of the 120 datasets of the nonlinear knapsack problem $AP(X)$.

Products ($\|\mathcal{J}\|$)	Strategies ($\|\mathcal{K}\|$)	Breakpoints (n)	# Instances
50	50	10	20
50	100	10	20
100	100	10	20
50	50	20	20
50	100	20	20
100	100	20	20

Figure 2.1. A MIP formulation for this problem is:

$$\begin{aligned} &\max \sum_{i\in\mathcal{J}}\sum_{j\in\mathcal{K}} y_{jk} \\ \text{subject to } &\sum_{i\in\mathcal{J}}\sum_{j\in\mathcal{K}} c_{jk}x_{jk} + \sum_{i\in\mathcal{J}} G_j z_j i \leqslant D \qquad (AP) \\ &(x_{jk}, y_{jk}, z_j) \in X_{jk} \;\; \forall j \in \mathcal{J}, k \in \mathcal{K} \end{aligned}$$

where X_{jk} is meant to denote that each of the triplets of variables (x_{jk}, y_{jk}, z_j) must lie in a set X (defined in (2.3)) specific to the product/strategy pair. We denote by $AP(S_1)$ and $AP(\Delta_1)$ the MIP formulations of AP that uses S_1 and Δ_1 respectively to model (2.3) and $AP(S_2)$ and $AP(\Delta_2)$ as the MIP formulation of AP that uses the stronger formulation S_2 and Δ_2 respectively.

Dataset Generation

We now discuss the procedure used to generate the datasets for the sample applicaiton problem (AP). The three components of the datasets are (a) the return on

investment functions $f_{jk}(\cdot)\ \forall j \in \mathcal{J}\ k \in \mathcal{K}$, (b) fixed costs $G_j\ \forall j \in \mathcal{J}$ for entering each product market and (c) variable costs $c_{jk}\ \forall j \in \mathcal{J},\ k \in \mathcal{K}$ per unit of budget allocated for marketing strategy $k \in \mathcal{K}$ of product $j \in \mathcal{J}$. We now describe how each of these components were generated.

Return on investment functions In our sample application, the return on investment is evaluated by piecewise-linear functions $f_{jk}(\cdot)$ which have the typical form shown in Figure 2.1.

Let $R(a, b, i, n)$ denote a random variable that lies between $a + \frac{i(b-a)}{n}$ and $a + \frac{(i+1)(b-a)}{n}$ with a distribution $a + \frac{b-a}{n}(i + \beta(2,2))$ where $\beta(2,2)$ is the beta distribution with both parameters set to 2. For each product $j \in \mathcal{J}$, the domain of $f_{jk}(\cdot)\ \forall k \in \mathcal{K}$ was generated using

$$d_j \sim R(4, 8, j, |\mathcal{J}|)$$

and the range was generated as

$$r_j \sim R(0.5, 1, j, |\mathcal{J}|)$$

where the notation j overloads both the product $j \in \mathcal{J}$ and an unique index for the product between 1 and $\mathcal{J}$. The desired s-shaped functions were generated by dividing the domain $[0, d_j]$ of $f_{jk}(\cdot)\ \forall k \in \mathcal{K}$ into three parts such that $f_{jk}(\cdot)$ is concave increasing in $[0, a^1_{jk}]$, convex increasing in $[a^1_{jk}, a^2_{jk}]$ and concave increasing

again in $[a^2_{jk}, d_j]$. The random variables a^2_{jk} and $a^2_{j,k}$ were generated using

$$a^1_{jk} \sim d_j R(0.1, 0.5, j, |\mathcal{J}|)$$
$$a^2_{jk} \sim d_j R(0.3, 0.7, j, |\mathcal{J}|).$$

The set of breakpoints B_{jki} $\forall i \in \{1 \dots n\}$ were calculating by dividing each of the three domains into approximately $\frac{n}{3}$ equal parts which can be written as

$$\begin{aligned}
B_{jki} &= 3i\frac{a^1_{jk}}{n} && i = 1 \dots \left\lfloor \frac{n}{3} \right\rfloor \\
B_{jki} &= a^1_{jk} + 3i\frac{a^2_{jk} - a^1_{jk}}{2n} && i = \left\lfloor \frac{n}{3} \right\rfloor + 1 \dots \left\lfloor \frac{2n}{3} \right\rfloor \\
B_{jki} &= a^2_{jk} + 3i\frac{d_j - a^3_{jk}}{n} && i = \left\lfloor \frac{2n}{3} \right\rfloor + 1 \dots n.
\end{aligned}$$

The corresponding function evaluations $F_{jki} := f_{jk}(B_{jki})$ were generated as

$$\begin{aligned}
F_{jki} &= b^1_{jk}\sqrt{\frac{B_{jki}}{B_{jk\lfloor \frac{n}{3} \rfloor}}} && i = 1 \dots \left\lfloor \frac{n}{3} \right\rfloor \\
F_{jki} &= F_{jk\lfloor \frac{n}{3} \rfloor} + b^2_{jk}\left(\frac{B_{jki} - B_{jk\lfloor \frac{n}{3} \rfloor}}{B_{jk\lfloor \frac{2n}{3} \rfloor} - B_{jk\lfloor \frac{n}{3} \rfloor}}\right)^2 && i = \left\lfloor \frac{n}{3} \right\rfloor + 1 \dots \left\lfloor \frac{2n}{3} \right\rfloor \\
F_{jki} &= F_{jk\lfloor \frac{2n}{3} \rfloor} + b^3_{jk}\sqrt{\frac{B_{jki} - B_{jk\lfloor \frac{2n}{3} \rfloor}}{B_{jk\lfloor \frac{2n}{3} \rfloor} - B_{jk\lfloor \frac{n}{3} \rfloor}}} && i = \left\lfloor \frac{2n}{3} \right\rfloor + 1 \dots n
\end{aligned}$$

where b^1_{jk}, b^2_{jk} and b^3_{jk} are random variables distributed by

$$b^1_{jk} \sim r_j R(0.05, 0.1, j, |\mathcal{J}|)$$
$$b^2_{jk} \sim r_j R(0.4, 0.7, j, |\mathcal{J}|)$$
$$b^3_{jk} \sim r_j R(0.7, 1, j, |\mathcal{J}|).$$

Costs and Budget For each strategy $k \in \mathcal{K}$ and product $j \in \mathcal{J}$, the per-unit operating costs were generated as

$$c_{jk} \sim \beta(2,2)\ R(0.8, 1.2, j, |\mathcal{J}|)\ R(0.8, 1.2, k, |\mathcal{K}|) \qquad \forall j \in \mathcal{J}, k \in \mathcal{K}$$

and the fixed costs were generated as

$$G_j \sim E_G\ R(0.5, 1, j, |\mathcal{J}|)\ U(0.8, 1.2) \qquad \forall j \in \mathcal{J}.$$

where $E_G = 0.105|\mathcal{J}|\ |\mathcal{K}|$. This procedure ensured that the total fixed costs are of the same order as the total variable costs. The overall budget D was set to $6E_G$.

Numerical Results

We report tests conducted on 120 simulated instances of $AP(X)$. We created 20 random instances for each of the six problem sizes in Table 2.2. All instances were solved to 0.1% optimality using Gurobi 4.5.1 with default options on 2.66GHz Intel Core2 Quad CPU Q9400 processor with 8GB RAM. For all instances, we compare the quality of the LP relaxation as the percentage gap between the root LP relaxation

Table 2.3: Summary of performance of formulations $AP(S_2)$ and $AP(S_1)$ on 120 simulated instances. Arithmetic mean, standard deviation, and geometric mean are shown.

Metric	Model	A.M	St. Dev	G.M
LP gap (%)	$AP(S_2)$	0.05	0.05	0.03
	$AP(S_1)$	19.7	1.5	19.6
Time (s)	$AP(S_2)$	16.8	12.0	12.3
	$AP(S_1)$	703.0	853.1	255.6
Nodes	$AP(S_2)$	26.3	33.1	9.2
	$AP(S_1)$	402.9	312.4	314.5

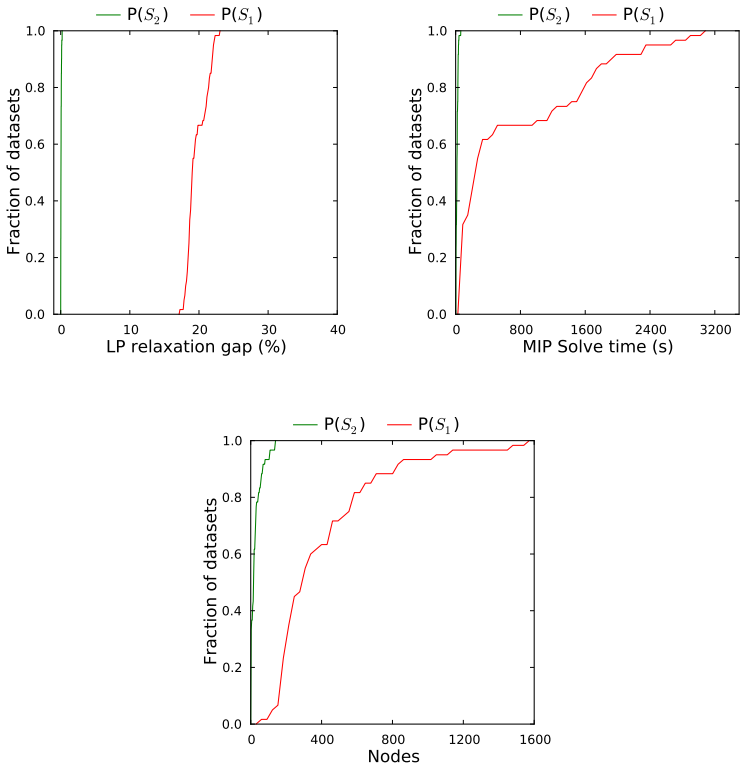
value of the MIP formulations $AP(S_2)$ and $AP(\Delta_2)$ with the alternate formulations $AP(S_1)$ and $AP(\Delta_1)$ relative to the optimal solution for each instance. We also measure the CPU time taken (using a single thread) and number of nodes in the search tree.

Table 2.4: Summary of performance of incremental models $AP(\Delta_2)$ and $AP(\Delta_1)$ on 120 datasets of the nonlinear knapsack problem.

Metric	Formulation	A.M	St. Dev	G.M
LP relaxation gap(%)	$AP(\Delta_2)$	0.06	0.05	0.03
	$AP(\Delta_1)$	19.66	1.47	19.60
Time (s)	$AP(\Delta_2)$	531.49	1240.89	171.25
	$AP(\Delta_1)$	1131.36	1401.04	481.13
Nodes	$AP(\Delta_2)$	38.10	63.75	7.85
	$AP(\Delta_1)$	179.22	129.51	98.70

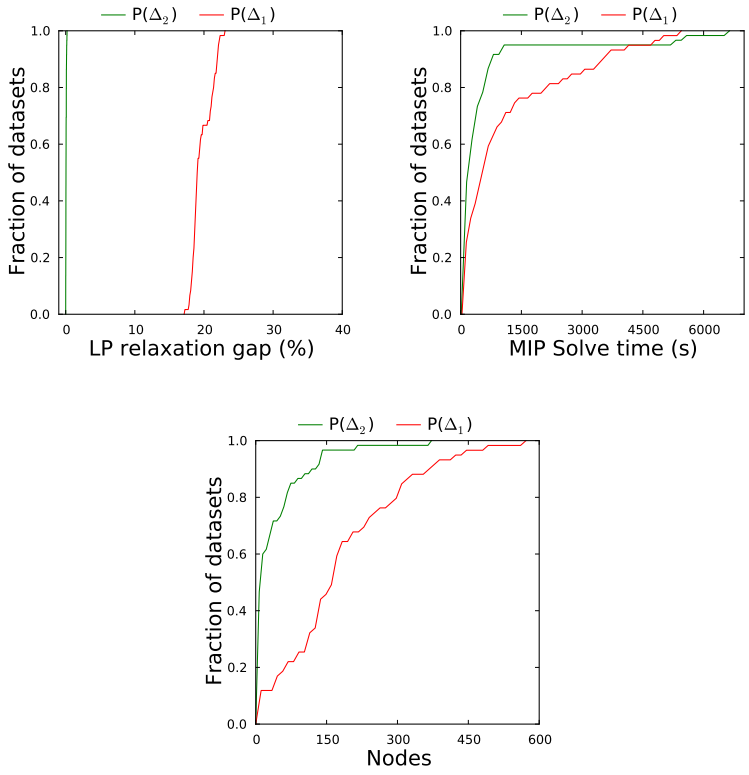
SOS2 Model. We compare the two SOS2 models $AP(S_2)$ and $AP(S_1)$. Table 2.3 shows the summary statistics of our experiment and figure 2.2 plots the empirical

Figure 2.2: Comparing performance of SOS2 models $AP(S_2)$ and $AP(S_1)$ on 120 datasets of the nonlinear knapsack problem.



cumulative distribution functions of each of our three performance metrics. The results convincingly demonstrate the advantage of using the locally ideal formulation $AP(S_2)$. The average root gap for $AP(S_2)$ was 0.05%, while for $AP(S_1)$ the average root gap was 19.6%. In fact, the best root gap for any instance of $AP(S_1)$ was 17.1%. In terms of MIP solve times, $AP(S_1)$ was solved on average in 703 seconds, while $AP(S_2)$ was solved 41.8 times faster on average. In the worst case, Gurobi explored 1117 times more nodes on an instance modeled with $AP(S_1)$ than with

Figure 2.3: Comparing performance of incremental models $AP(\Delta_2)$ and $AP(\Delta_1)$ on 120 datasets of the nonlinear knapsack problem.



$AP(S_2)$. Clearly, one should use the locally ideal model $AP(S_2)$.

Incremental Model. We compare the two incremental models $AP(\Delta_2)$ and $AP(\Delta_1)$. Table 2.4 shows the summary statistics of our experiment and figure 2.3 plots the empirical cumulative distribution functions of each of our three performance metrics. Again, our results demonstrate the advantage of using the locally ideal formulation $AP(\Delta_2)$ over $AP(\Delta_1)$. Formulation $AP(\Delta_2)$ produced stronger root LP relaxations which resulted the MIP being solved, on average, 2.1x faster than $AP(\Delta_1)$ with some

instances being as high as 25x faster.

2.4 Concluding remarks

In this chapter, we present a theoretical and computational comparison of MIP models for PLFs where a binary indicator variable determines if the function is required to be evaluated. We propose strong formulations for this general class of MIP models by extending standard textbook PLF models including the incremental method, SOS2-based models, the multiple choice model, the convex combination model, and others. We showed in all cases that our formulations are either locally ideal or sharp, while a standard formulation that uses a variable upper bound constraint is not. Our numerical experiments demonstrate that our proposed formulations have significant computational advantages.

3 MODELS AND SOLUTION TECHNIQUES FOR PRODUCTION PLANNING PROBLEMS WITH INCREASING BYPRODUCTS

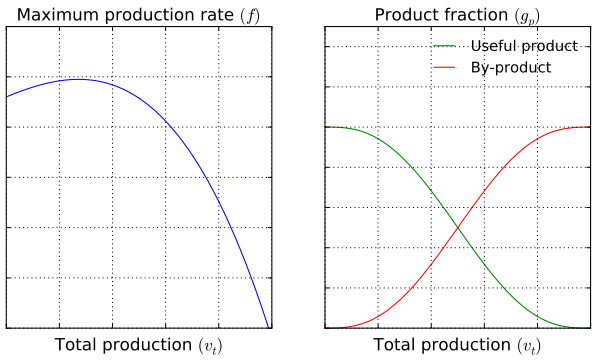
In this chapter, we consider a production planning problem where the production process creates a mixture of desirable products and undesirable byproducts. In this production process, at any point in time the fraction of the mixture that is an undesirable byproduct increases monotonically as a function of the cumulative mixture production up to that time. The mathematical formulation of this continuous-time problem is nonconvex. This chapter is based on results in Sridhar et al. [91].

3.1 Introduction

In this chapter, we study a production planning problem in which the production process creates a mixture of products $\mathcal{P} = \mathcal{P}^+ \cap \mathcal{P}^-$, where the products $p \in \mathcal{P}^+$ are useful and the products $p \in \mathcal{P}^-$ are undesirable byproducts. As more of the mixture is produced, the fraction of the mixture that is a useful product decreases monotonically. Conversely, the fraction of each byproduct increases monotonically as a function of cumulative mixture production. Production planning problems with these characteristics arise in engineering applications like the extraction of natural resources such as oil and gas [43, 49, 93, 96, 97] from fields where a pressure differential determines the rate at which the resources are extracted. This pressure difference drops as more of the resources are extracted which reduces the maximum rate at which more resources can be extracted. In these applications, the purity

of the resources obtained reduces as more of the resources are extracted. The problem structure studied in this work also naturally appears while modeling the performance of hydro turbines [15, 72], designing chemical processes [66] and scheduling compressors in petroleum reservoirs [19].

Figure 3.1: Example production functions.



We describe and analyze the model for a single production process exhibiting these characteristics. We are motivated by applications that contain many processes having these characteristics, which are linked in some way, such as having common processing facilities or distribution networks. In Section 3.5 we describe an example application in which the facilities are part of a network in which demands are to be met. For such applications, the model we develop would be used for each production facility in the problem. Optimizing production for problems that contain multiple processes of this structure is complicated because the amount of each product produced is a *nonconvex* function of the cumulative mixture production up to that time. In addition, there are fixed costs associated with starting production, requiring the use of {0,1}-decision variables. We find that this nonconvex mixed-

integer nonlinear programming problem is hard to solve directly using state-of-the-art software packages such as BARON [94] or Couenne [12], because the general purpose techniques for relaxing the nonconvex constraints used in these solvers appear to produce weak lower bounds. Our approach is to develop accurate and computationally useful formulations for an individual process model, which can then be used in a formulation that plans multiple such processes.

We now describe the production model for a single production process. Consider a production horizon $[0, L]$ with $x(t)$ representing the production rate of a mixture of products $\mathcal{P}$ at time $t \in [0, L]$. The cumulative production of the mixture up to time t is denoted by $v(t)$ and can be expressed as

$$v(t) = \int_0^t x(s)\,ds, \quad t \in [0, L]. \tag{3.1}$$

As input to the problem, we are given a production function $f(\cdot) : \mathbb{R}_+ \to \mathbb{R}_+$ that defines the maximum production rate of the mixture as a function of cumulative production up to time t. Hence, $x(t)$ must satisfy:

$$x(t) \leqslant f(v(t)), \quad t \in [0, L]. \tag{3.2}$$

For each product $p \in \mathcal{P}$, we are given a function $g_p : \mathbb{R}_+ \to [0,1]$, that specifies the fraction of the mixture that is product $p \in \mathcal{P}$ as a function of the cumulative mixture production. The rate of production of product p at time t, denoted by $y_p(t)$, is given by:

$$y_p(t) = x(t)\, g_p(v(t)), \quad p \in \mathcal{P}, t \in [0, L]. \tag{3.3}$$

We make the following assumptions on the nonlinear functions in this model.

Assumption 3.1. *The production function $f : \mathbb{R}_+ \to \mathbb{R}_+$ is concave, the ratio functions $g_p : \mathbb{R}_+ \to [0,1]$ are non-increasing for $p \in \mathcal{P}^+$ (useful products) and non-decreasing for $p \in \mathcal{P}^-$ (byproducts), and the functions $g_p, p \in \mathcal{P}$ satisfy $\sum_{p\in\mathcal{P}} g_p(v) = 1$ for all $v \geqslant 0$.*

To facilitate modeling of a fixed cost for beginning production at time $t \in [0, L]$, the model contains a binary step function $z : [0, L] \to \{0, 1\}$ to indicate if production can occur at time $t \in [0, L]$. Let M^v be an upper bound on the maximum cumulative production. Then the following constraint ensures that production occurs only after the production facility has been opened:

$$v(t) \leqslant M^v z(t), \quad t \in [0, L]. \tag{3.4}$$

Constraints (3.1) and (3.4) ensure that once production commences, it may continue until the end of the time horizon. In other words, for any $t_1, t_2 \in [0, L]$, if $t_1 > t_2$ then $z(t_1) \geqslant z(t_2)$.

Figure 3.1 gives an example of how the maximum production rate $f(\cdot)$, useful

product fraction $g_p(\cdot), p \in \mathcal{P}^+$, and byproduct fraction $g_p(\cdot), p \in \mathcal{P}^-$ evolve as a function of the cumulative mixture production. Our model can be extended to the case where $f(\cdot)$ is not necessarily concave and the functions $g_p(\cdot)\ \forall p \in \mathcal{P}$ are not necessarily monotone. The primary modification that would be required is in the method for constructing the piecewise-linear approximations that are used in our formulation.

A time-discretization of this production planning problem is required to obtain a model that is suitable for implementation and numerical evaluation. Tarhan et al. [93] introduce a natural discrete-time mixed-integer nonlinear (MINLP) formulation (reviewed in Section 3.2) which discretizes the production horizon $[0, L]$ into T time periods and transforms constraints (3.1)-(3.4) into their discrete-time counterparts.

Contributions

We highlight three main contributions of this work. The first contribution of our work is to introduce an alternative discrete-time MINLP formulation that is more accurate than the one which appeared in [93]. This alternative formulation has also been independently derived by Gupta and Grossmann [43]. The key difference between the alternative formulation and that proposed in [93] lies in the discretization of constraints (3.3). In [93], the authors discretize the product production rate ($y_{p,t}$) during a time period t by multiplying the discretized production rate at the start of a time period (x_t) with the product production ratio at the end of the previous time period ($g_p(v_{t-1})$). We show in Section 3.2 that this discretization yields an MINLP formulation that may have significant errors in the approximation

of the actual product production rate $y_p(t)$.

In this work, we propose an alternative discrete-time MINLP formulation that *exactly* calculates the amount of product produced during each time period. This formulation is based on the cumulative product production $\int_0^t y_p(s)\,ds$. In Section 3.2, we demonstrate how this reformulation results in a model that is a more accurate representation of the underlying continuous-time formulation. This improvement in accuracy is realized irrespective of the nature of the production functions $f(\cdot)$ and $g_p(\cdot)$. However, if the production functions satisfy the concavity and monotonicity assumptions, our proposed reformulation exploits the property that the integral of a nondecreasing (nonincreasing) function is convex (resp. concave) thereby transforming the terms involving products of non-convex functions ($x_t g_p(v_{t-1})$) into *univariate convex/concave* functions representing the cumulative product production. As a result, as demonstrated by numerical experiments in Section 3.5, this reformulation yields a model that is not only more accurate but is also easier to approximately solve.

As the second contribution of this work, we derive three different mixed-integer *linear* programming (MILP) models that approximate or relax the convex/concave functions. The first model, piecewise-linear approximation (PLA), approximates each of the nonlinear functions involved using piecewise-linear forms modeled with variables that are constrained to be special ordered sets of type 2 (SOS2) [10]. The PLA model is an approximation of the production planning problem but does not provide a bound on the optimal value of the MINLP model. In order to provide guarantees on solution quality, we also propose two MILP *relaxations* of the

MINLP model based on piecewise-linear under- and over- approximations of the nonlinear functions. The 1-secant relaxation (1-SEC) uses multiple tangents and a single secant to envelope the convex/concave cumulative production functions. An advantage of this formulation is that it does not require SOS2 variables. We also extend the 1-SEC relaxation to the k-secant relaxation (k-SEC), which uses multiple tangents and multiple secants to relax the convex/concave cumulative production function. . The addition of multiple secants requires the use of SOS2 variables, similar to the PLA model. In Section 3.5, we compare the MILP formulations in terms of computational difficulty to solve, quality of solution produced, and in the case of the relaxations, the quality of the bound produced.

The third contribution of this work is to demonstrate two techniques for improving the LP relaxations of the proposed MILP formulations. These techniques exploit two special properties of this problem. First, a model of each of the nonlinear functions is required only when a corresponding binary indicator variable is set to one. This allows a formulation with a better LP relaxation bound to be obtained by slightly modifying the constraints of the SOS2 formulation. Second, we exploit the fact that the cumulative total production, which is the argument to the nonlinear functions being approximated, is increasing over time. This enables the derivation of valid inequalities that relate the variables of the SOS2 approximations of these functions in consecutive time periods to each other.

One special property of this production planning problem is that all the nonlinear production functions $f(\cdot)$ and $g_p(\cdot), p \in \mathcal{P}$ share the same domain $[0, M^v]$ because they are functions of the same argument $v(t)$. One way to model piecewise-

linear approximations and relaxations of these nonlinear functions is to introduce a set of break points for each function approximation/relaxation. The advantage of this method is that the choice of break points approximating each function $f(\cdot)$ and $g_p(\cdot), p \in \mathcal{P}$ is flexible, thereby allowing each function to use break points that provide the best approximation for that function. However, this approach requires introducing a separate set of branching entities (SOS2 variables) for each function, which can significantly increase the difficulty in solving the resulting model. An alternate model introduces a single set of SOS2 variables that are shared between all the production functions $f(\cdot)$ and $g_p(\cdot), p \in \mathcal{P}$. Such a formulation is possible only if the functions share the same domain and use the same set of break points in that domain.

While sharing the SOS2 variables is computationally desirable, the requirement that the same set of break points be used for all functions may reduce the accuracy of each of the individual function approximations/relaxations. In addition, most existing algorithms [17, 83, 94] for finding piecewise-linear approximations/relaxations are designed only for a single univariate convex/concave function. In Section 3.4, we propose nonlinear programming (NLP) formulations for determining the best possible piecewise-linear approximations and relaxations of multiple univariate convex/concave functions, with the requirement that the piecewise-linear functions share the same set of break points. We demonstrate that our method can reap the computational benefits of sharing break points while still maintaining the accuracy of each function approximation. The recent work of [39] has also considered the problem of finding piecewise-linear approximations of multiple functions sharing

the same domain and using the same set of break points. The most important difference in our work is that in addition to finding a piecewise linear approximation, we also consider the problem of finding a good piecewise-linear *relaxation*.

Gupta and Grossmann [43] independently discovered the alternative formulation that we present in Section 3.2, and similarly found that this formulation yields computational benefits and improved formulation accuracy. Gupta and Grossmann also derived a MILP formulation of a piecewise-linear approximation of the alternative formulation. While the key idea of this reformulation is the same, there are a number of differences which we now highlight. First, our production model is slightly different in that we consider a production process in which we are given functions for the *fraction* of the total mixture that corresponds to each product, as opposed to the ratio of one component of the mixture to another. This formulation may be advantageous numerically because the fraction is bounded between zero and one. More significantly, there are several differences between our approaches for obtaining piecewise-linear approximations. In addition to obtaining a piecewise-linear approximation as in [43], we also obtain piecewise-linear relaxations, which can be used to obtain bounds on the best possible MINLP solution value. We also show how the MILP formulations of the piecewise-linear models can be strengthened, and when necessary we use special ordered sets of type II (SOS2) to model the piecewise-linear functions, as opposed to explicitly introducing binary decision variables as in [43]. The SOS2 formulation is locally ideal, which is a desirable property for a mixed-integer programming formulation, whereas the formulation used in [43] does not have this property [103]. Finally, we study the

problem of finding piecewise-linear approximations and relaxations when multiple functions sharing the same argument are to be approximated, with the restriction that they be approximated using the same set of break points, which is an important component of a solution process that uses this formulation.

The remainder of this chapter is organized as follows. In Section 3.2, we present the two discrete-time MINLP formulations. In Section 3.3, we provide the three MILP approximations/relaxations of the proposed MINLP formulation, and demonstrate how these formulations can be strengthened. In Section 3.4, we introduce NLP formulations that identify piecewise-linear functions that most accurately approximate and relax multiple nonlinear functions that share the same domain. In Section 3.5 we present a computational study that compares the accuracy and performance of all formulations introduced in Section 3.2 and 3.3. We close with concluding remarks in Section 3.6.

3.2 Discrete-Time Formulations

This section describes two discrete-time MINLP formulations of the production planning problem that considers a decision horizon of finite decision-making periods $\mathcal{T} = \{1, 2, \ldots, T\}$ each of length Δ_t $(t \in \mathcal{T})$, such that $L = \sum_{t=1}^{T} \Delta_t$. For each $t \in \mathcal{T}$, let x_t represent the total mixture production quantity during period t, v_t represent the cumulative mixture production up to and including time period t, and $y_{p,t}$ represent the amount of product $p \in \mathcal{P}$ produced in period t. Also, let the binary variable $z_t \in \{0, 1\}$ indicate if production can occur during period $t \in \mathcal{T}$.

The discrete-time MINLP formulations we present in this section all include the constraints

$$v_t = v_{t-1} + x_t, \qquad t \in \mathcal{T} \tag{3.5a}$$

$$z_t \geqslant z_{t-1}, \qquad t \in \mathcal{T}\backslash\{1\} \tag{3.5b}$$

$$v_t \leqslant M^v\, z_t, \qquad t \in \mathcal{T} \tag{3.5c}$$

$$x_t \leqslant \Delta_t\, f(v_{t-1}), \qquad t \in \mathcal{T}. \tag{3.5d}$$

where $v_0 := 0$ and $P := |\mathcal{P}|$. For ease of notation, we define

$$Y := \left\{ z \in \{0,1\}^T, x \in \mathbb{R}_+^T, v \in \mathbb{R}_+^{T+1}, y \in \mathbb{R}_+^{P\times T} : \ (3.5) \right\}.$$

Constraint (3.5a) is a direct discrete-time analog to (3.1) and (3.5b) simply states that once a facility is opened it must stay open. The constraint (3.5c) is a discrete-time analog of (3.4). Constraint (3.5d) is an approximation of the continuous-time constraints (3.2), where we assume that the maximum flow rate during period $t \in \mathcal{T}$ is determined by the cumulative production at the beginning of the period v_{t-1} and the interval length Δ_t. Alternative discretizations of (3.2) are possible, such as one that limits the production rate based on the cumulative production at the end of the time period, or one that uses an average of $f(v_{t-1})$ and $f(v_t)$. However, these alternative discretizations do not affect this work, so we consistently use equation (3.5d) in all of our formulations.

Formulation MINLP1

The first discrete-time formulation is MINLP1, defined as the set of $(z, x, v, y) \in Y$ that satisfy:

$$y_{p,t} = x_t g_p(v_{t-1}), \quad p \in \mathcal{P}, t \in \mathcal{T}. \tag{3.6}$$

Constraint (3.6) is a natural discretization of (3.3), which approximates the amount of each product $p \in \mathcal{P}$ produced in period t by multiplying the total mixture production in the period, x_t, with the corresponding product p fraction, $g_p(v_{t-1})$ determined based on the cumulative production up to the beginning of period t. Such a discrete-time approximation has been used, for example, in [93].

There are two drawbacks of formulation MINLP1. To describe them, we first reformulate (3.6) as a combination of two constraints involving intermediate decision variables $u \in \mathbb{R}_+^{P \times T}$:

$$y_{p,t} = x_t u_{p,t}, \qquad p \in \mathcal{P}, t \in \mathcal{T} \tag{3.7a}$$

$$u_{p,t} = g_p(v_{t-1}), \quad p \in \mathcal{P}, t \in \mathcal{T}. \tag{3.7b}$$

The first drawback is that MINLP1 contains two types of nonconvexities: the bilinear terms, written explicitly in (3.7a), and the nonconvex functions $g_p(\cdot)$ for each product $p \in \mathcal{P}$ in (3.7b). This combination of nonconvexities results in a formulation that is computationally difficult to solve, even approximately, as we illustrate in Section 3.5. The second drawback of MINLP1 is the inaccuracy of the discrete-time

approximation used in (3.6). We next discuss an alternative MINLP formulation that eliminates this inaccuracy.

Formulation MINLP2

The second formulation is obtained by using an alternative approach to calculate $y_{p,t}$, the amount of each product $p \in \mathcal{P}$ that is produced in period $t \in \mathcal{T}$. First, we *exactly* calculate the cumulative amount of product $p \in \mathcal{P}$ produced up to and including time period $t \in \mathcal{T}$ using the continuous-time formulation. We define $w_p(t)$ as the cumulative production for each product $p \in \mathcal{P}$ up to time $t \in [0, L]$. Specifically, for any $p \in \mathcal{P}$ and $t \in [0, L]$,

$$w_p(t) := \int_0^t y_p(s)\,ds = \int_0^t x(s) g_p(v(s))\,ds \tag{3.8}$$

$$= \int_0^{v(t)} g_p(\theta)\,d\theta \qquad \left(\text{since } \frac{dv(t)}{dt} = x(t)\right) \tag{3.9}$$

$$:= h_p(v(t)). \tag{3.10}$$

The change in the variables of the integral from equation (3.8) to (3.9) yields $h_p(\cdot)$, the cumulative production function which *exactly* evaluates the total amount of product $p \in \mathcal{P}$ produced up to time $t \in [0, L]$. Now, we evaluate $y_p(t)$, the product production rate at a time $t \in [0, L]$ using the relationship

$$y_p(t) = \frac{dw_p(t)}{dt} \qquad p \in \mathcal{P}, t \in [0, L]. \tag{3.11}$$

A time-discretization of the constraints (3.10) and (3.11) yields an alternate

formulation that uses variables $w_{p,t}$: MINLP2 is the set of $(z, x, v, y, w) \in Y \times \mathbb{R}_+^{P\times T}$ that satisfy

$$w_{p,t} = h_p(v_t), \qquad p \in \mathcal{P}, t \in \mathcal{T} \tag{3.12a}$$

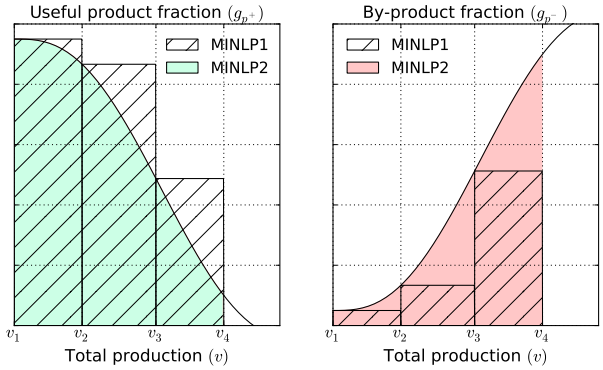
$$y_{p,t} = w_{p,t} - w_{p,t-1}, \qquad p \in \mathcal{P}, t \in \mathcal{T} \tag{3.12b}$$

where $w_{p,0} := 0$ for $p \in \mathcal{P}$. Because the formulations MINLP2 and MINLP1 share the constraints (3.5) included in Y, they model the bounds on the mixture production variables, $x_t \leqslant \Delta_t f(v_{t-1})$, in the same way. However, in contrast to equation (3.12), formulation MINLP1 evaluates the cumulative product production $w_{p,t}$ as

$$w_{p,t} \sum_{s\leqslant t} y_{p,s} = \sum_{s\leqslant t} x_s g_p(v_{s-1}), \quad p \in \mathcal{P}, t \in \mathcal{T}. \tag{3.13}$$

Equation (3.13) indicates that MINLP1 assumes that the product production fraction remains fixed at $g_p(v_{t-1})$ through the whole time period t. . Figure 3.2 illustrates the difference in the calculation of the product production $y_{p,t}$ using formulations MINLP1 and MINLP2 for a problem with four time periods. The total production up to and including time period t is denoted by v_t for each t. The hatched region represents the cumulative production $w_{p,t}$ calculated using equation (3.13) of the formulation MINLP1. In (3.13), the product fraction is assumed fixed throughout a time period, and so the estimate of the amount produced in a period is given by the area of the rectangle having base length equal to the total mixture amount produced in the period ($x_t = v_t - v_{t-1}$) and the height corresponding to that fixed product fraction, $g_p(v_{t-1})$. In contrast, the shaded region illustrates

Figure 3.2: Differences in the estimation of product production $y_{p,t}$ using MINLP1 and MINLP2.



the correct calculation of the cumulative production production $w_{p,t}$ using $h_p(\cdot)$ defined in equation (3.12a) of formulation MINLP2. Clearly, MINLP1 overestimates the amount of useful product and understimates the amount of byproduct.

Formulation MINLP2 also has two desirable computational properties. First, the total mixture production x_t and individual product productions $y_{p,t}$ are evaluated using *univariate* functions $f(\cdot)$ and $h_p(\cdot), p \in \mathcal{P}$, thereby eliminating the need to relax the product terms in equation (3.6). This enables us to derive univariate piecewise-linear approximations and relaxations, described in section 3.3, which, as we demonstrate in our numerical experiments in section 3.5, can yield high quality solutions and relaxation bounds. Second, since $g_p(\cdot)$ is non-increasing for useful products $p \in \mathcal{P}^+$, and $g_p(\cdot)$ is non-decreasing for byproducts $p \in \mathcal{P}^-$, it follows that $h_p(\cdot)$ is concave for $p \in P^+$ and convex for $p \in P^-$. Hence, product production profiles $y_{p,t}$ can be evaluated as a difference of univariate convex functions for byproducts and as a difference of univariate concave functions for useful products.

3.3 Piecewise-Linear Approximations and Relaxations

MINLPs are challenging to solve since they may include both nonlinear (and possibly nonconvex) functions and integer variables. A common approach is to approximate the nonlinear functions with piecewise-linear functions, and then solve the corresponding approximation as a mixed-integer *linear* program (MILP) (e.g., [3, 10, 27, 37, 56, 64, 83, 104, 103]). In this section, we describe one MILP approximation and two MILP relaxations of the set MINLP2.

The formulations in this section are based on piecewise-linear functions that either approximate or provide bounds on $f(\cdot)$ and $h_p(\cdot)$, $p \in \mathcal{P}$, over the domain. A piecewise-linear function $\bar{f}$ having m line segments and domain $[0, M^v]$ is defined using break points $B \in \mathbb{R}^{m+1}$ with $0 = B_0 < B_1 < B_2 < \cdots < B_m = M^v$, and function values $F = (F_0, F_1, \ldots, F_m)$ at these points:

$$\bar{f}(v; B, F) := F_{k-1} + \left(\frac{F_k - F_{k-1}}{B_k - B_{k-1}}\right)(v - B_{k-1}), \quad B_{k-1} \leqslant v \leqslant B_k,\ k = 1, \ldots, m. \tag{3.14}$$

Similarly, the piecewise-linear functions $\bar{h}_p$ for $p \in \mathcal{P}$ are defined as follows:

$$\bar{h}_p(v; B, H_p) := H_{p,k-1} + \left(\frac{H_{p,k} - H_{p,k-1}}{B_k - B_{k-1}}\right)(v - B_{k-1}), \quad B_{k-1} \leqslant v \leqslant B_k,\ k = 1, \ldots, m \tag{3.15}$$

where $H_p \in \mathbb{R}^{m+1}$, $p \in \mathcal{P}$. The choice of the break points B and piecewise-linear function values, F and $H_p, p \in \mathcal{P}$, depends on whether we are seeking an approximation as in Section 3.3, or a relaxation, as in Section 3.3.

Piecewise-linear approximation (PLA)

The first piecewise-linear approximation is obtained by choosing break-points $\hat{B}$, and function values $\hat{F}$ and $\hat{H}_p, p \in \mathcal{P}$, such that $\bar{f}(v; \hat{B}, \hat{F}) \approx f(v)$ and $\bar{h}_p(v; \hat{B}, \hat{H}_p) \approx h_p(v), p \in \mathcal{P}$ over $v \in [0, M^v]$. Figure 3.3 demonstrates an approximation scheme with $\hat{F}_k = f(\hat{B}_k)$ and $\hat{H}_{p,k} = h_p(B_k)$, i.e., each function approximation is constituted from linear approximations taken at points that lie on the curve. We assume that $\hat{H}_{p,0} = h_p(0)\ (= 0)$, so that $\bar{h}_p(0; \hat{B}, \hat{H}_p) = h_p(0)$ for each $p \in \mathcal{P}$. In Section 3.4 we discuss a method for choosing the break points $\hat{B}$ and function approximation values $\hat{F}$ and $\hat{H}_p$.

The piecewise-linear approximation model is then obtained by replacing the functions $f(\cdot)$ and $h_p(\cdot)$ for $p \in \mathcal{P}$ with their piecewise-linear approximations $\bar{f}(\cdot\,; \hat{B}, \hat{F})$ and $\bar{h}_p(\cdot\,; \hat{B}, \hat{H}_p)$ for $p \in \mathcal{P}$ in (3.5d) and (3.12a), respectively, yielding:

$$x_t \leqslant \Delta_t\ \bar{f}(v_{t-1}; \hat{B}, \hat{F}), \qquad t \in \mathcal{T} \tag{3.16a}$$

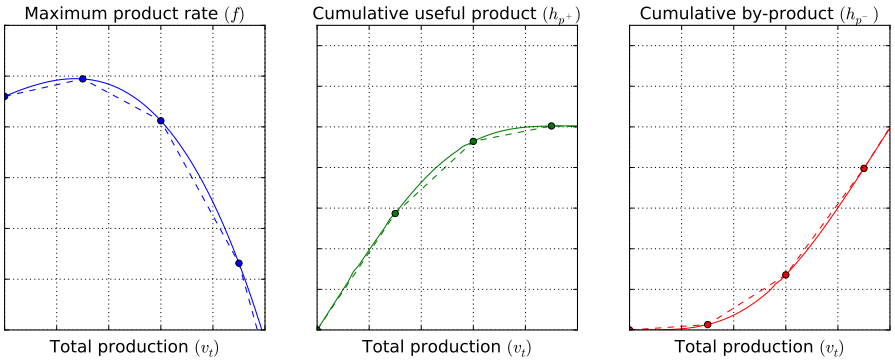
$$w_{p,t} = \bar{h}_p(v_t; \hat{B}, \hat{H}_p), \qquad p \in \mathcal{P}, t \in \mathcal{T}. \tag{3.16b}$$

Note that, if the function approximations $\bar{f}$ and $\bar{h}$ are obtained by using the function values at the break points, then (3.16a) is a restriction of (3.5d), whereas (3.16b) is an approximation.

We next discuss how to model (3.16) using linear constraints and variables that are constrained to be a special ordered set of type 2 (SOS2) [10]. Define $\mathcal{M} = \{0, 1, \ldots, m\}$. For each period $t \in \mathcal{T}$, we introduce a set of non-negative decision variables $\lambda_{t,k},\ k \in \mathcal{O}$ which are constrained to be SOS2: at most two elements in

the set can be non-zero, and these two non-zero elements must be in adjacent positions (with respect to the ordering). Then, a piecewise-linear approximation of the formulation MINLP2 is obtained using additional variables $\lambda \in \mathbb{R}_{+}^{T \times (m+1)}$ and replacing (3.16) with the following constraints:

Figure 3.3: Piecewise-linear approximation model (PLA).



$$1 = \sum_{k \in \mathcal{O}} \lambda_{t,k}, \qquad t \in \mathcal{T} \tag{3.17a}$$

$$v_t = \sum_{k \in \mathcal{O}} \hat{B}_k \, \lambda_{t,k}, \qquad t \in \mathcal{T} \tag{3.17b}$$

$$x_t \leqslant \Delta_t \sum_{k \in \mathcal{O}} \hat{F}_k \, \lambda_{t-1,k}, \qquad t \in \mathcal{T} \tag{3.17c}$$

$$w_{p,t} = \sum_{k \in \mathcal{O}} \hat{H}_{p,k} \, \lambda_{t,k}, \qquad p \in \mathcal{P}, t \in \mathcal{T} \tag{3.17d}$$

$$\{\lambda_{t,k} \mid k \in \mathcal{O}\} \in \text{SOS2}, \qquad t \in \mathcal{T}. \tag{3.17e}$$

For each $t \in \mathcal{T}$, equations (3.17a) and (3.17b) ensure v_t is written as a convex combination of the break points $\{\hat{B}_k \mid k \in \mathcal{M}\}$, and (3.17c) and (3.17d) model $\bar{f}(v_t; \hat{B}, \hat{F})$ and $\bar{h}_p(v_t; \hat{B}, \hat{H}_p)$ for $p \in \mathcal{P}$ as a corresponding convex combination of the function values. The constraints (3.17e) then ensure that the convex combination is obtained only by using adjacent break points [9]. We adopt the convention that $\lambda_{0,0} = 1$ and $\lambda_{0,k} = 0$ for $k \neq 0$, so that (3.17c) for $t = 1$ enforces $x_1 \leqslant \Delta_1 f(0)$ which is consistent with the approximation (3.16a) because $v_0 = 0$.

To define a formulation that uses the model (3.17), first define the set W, which will be used in all formulations in this section, as

$$
\begin{aligned}
W := \big\{ z \in \{0,1\}^T, x \in \mathbb{R}_+^T, v \in \mathbb{R}_+^{T+1}, y \in \mathbb{R}_+^{P\times T}, w \in \mathbb{R}_+^{P\times T}, \lambda \in \mathbb{R}_+^{P\times(T+1)} : \\
v_t = v_{t-1} + x_t, \qquad & t \in \mathcal{T} \\
z_t \geqslant z_{t-1}, \qquad & t \in \mathcal{T}\backslash\{1\} \\
y_{p,t} = w_{p,t} - w_{p,t-1}, \qquad & p \in \mathcal{P}, t \in \mathcal{T} \big\}.
\end{aligned}
$$

The constraints in the definition of W are just restatements of (3.5a), (3.5b), and (3.12b). Then, we define the formulation PLA by:

$$\text{PLA} := \{(z, x, v, y, w, \lambda) \in W : \text{ (3.17)}, \ v_t \leqslant M^v z_t, \ t \in \mathcal{T}\}.$$

The constraints $v_t \leqslant M^v z_t, \ t \in \mathcal{T}$ are a restatement of (3.5c), which enforce that $v_t = 0$ when $z_t = 0$.

PLA approximates the formulation MINLP2 but is not guaranteed to be either a restriction nor a relaxation of MINLP2. PLA includes integer restrictions and

SOS2 constraints, both of which can be directly handled by commercial integer programming software.

We next discuss two techniques for improving the linear programming relaxation of the approximation PLA. The first is an application of a technique introduced in [90]. The idea is to exploit the fact that, when $z_t = 0$, the models of the functions $\bar{f}(v_t; \hat{B}, \hat{F})$ and $\bar{h}_p(v_t; \hat{B}, \hat{H}_p)$ for $p \in \mathcal{P}$ are not necessary, as in that case we immediately know that in this case $x_t = v_t = y_{p,t} = w_{p,t} = 0$, $p \in \mathcal{P}$, and these conditions can all be obtained by replacing the constraint (3.17a) with

$$z_t = \sum_{k \in \mathcal{O}} \lambda_{t,k}, \qquad t \in \mathcal{T}. \tag{3.18}$$

In addition, with this substitution, the inequalities $v_t \leqslant M^v z_t, t \in \mathcal{T}$ also become redundant because $z_t = 0$ already implies $v_t = 0$ via (3.18) and (3.17b). We define PLA-S as the formulation that includes (3.18) but excludes $v_t \leqslant M^v z_t, t \in \mathcal{T}$ and (3.17a), specifically,

$$\text{PLA-S} = \{(z, x, v, y, w, \lambda) \in W : \text{ (3.17b)} - \text{(3.17e)}, \text{ (3.18)}\}.$$

The following proposition proves the validity of the formulation PLA-S.

Proposition 3.1. *Proj*$_{(z,x,v,y,w)}$(PLA) = *Proj*$_{(z,x,v,y,w)}$(PLA-S).

Proof. Let $(z, x, v, y, w) \in \text{Proj}_{(z,x,v,y,w)}(\text{PLA})$ so there exists λ' such that $(z, x, v, y, w, \lambda') \in W$ and satisfies $v_t \leqslant M^v z_t$, $t \in \mathcal{T}$ and (3.17). Let $\lambda_{t,k} = \lambda'_{t,k} z_t$ for all $t \in \mathcal{T}, k \in \mathcal{M}$, so that (3.18) and (3.17e) hold by construction. Let $t \in \mathcal{T}$. If $z_t = 1$, then $\lambda_{t,k} = \lambda'_{t,k}$

for $k \in \mathcal{M}$ and hence (3.17b) - (3.17d) trivially hold. Now suppose $z_t = 0$ so that $\lambda_{t,k} = 0$ for $k \in \mathcal{M}$. The inequalities (3.5c) imply that $v_t = 0$, and hence $x_t = 0$ follows from $v_t = v_{t-1} + x_t$. Also, because $v_t = 0$ and $\hat{B}_k > 0$ for $k > 0$, (3.17b) implies that $\lambda'_{t,0} = 1$. Therefore, (3.17d) implies $w_{p,t} = 0$ for each $p \in \mathcal{P}$ since $\hat{H}_{p,0} = 0$. Thus, (3.17b) - (3.17d) all hold with zeros on both sides of the equations, and thus $(x, z, v, y, w) \in \text{Proj}_{(z,x,v,y,w)}(\text{PLA-S})$.

Now let $(z, x, v, y, w) \in \text{Proj}_{(z,x,v,y,w)}(\text{PLA-S})$ so there exists λ such that $(z, x, v, y, w, \lambda) \in W$ satisfy (3.17b) - (3.17e) and (3.18). Let $t \in \mathcal{T}$. If $z_t = 1$, set $\lambda'_{t,k} = \lambda_{t,k}$ for all $k \in \mathcal{M}$. If $z_t = 0$, set $\lambda'_{t,0} = 1$ and $\lambda'_{t,k} = 0$ for $k > 0$. Then it is easy to check that $(z, x, v, y, w, \lambda') \in \text{PLA}$, which yields the result. □

The following proposition shows that PLA-S has a stronger LP relaxation than PLA. For a set S defined by linear inequalities and integrality or SOS2 constraints, we define $\mathcal{R}(S)$ as the polyhedral relaxation of S, obtained by keeping the linear inequalities defining S but dropping the integrality and SOS2 constraints in S.

Proposition 3.2. *$\text{Proj}_{(z,x,v,y,w)}\mathcal{R}(\text{PLA-S}) \subseteq \text{Proj}_{(z,x,v,y,w)}\mathcal{R}(\text{PLA})$ and the inclusion can be strict.*

Proof. Let $(z, x, v, y, w) \in \text{Proj}_{(z,x,v,y,w)}\mathcal{R}(\text{PLA-S})$ and so let λ be such that $(z, x, v, y, w, \lambda) \in \mathcal{R}(\text{PLA-S})$. By (3.17b) and (3.18), for each $t \in \mathcal{T}$, $v_t = \sum_{k\in\mathcal{O}} \hat{B}_k \lambda_{t,k} \leqslant \sum_{k\in\mathcal{O}} \lambda_{t,k} M^v \leqslant M^v z_t$. Next, for each $t \in \mathcal{T}$, let $\lambda'_{t,0} = \lambda_{t,0} + 1 - z_t$ and $\lambda'_{t,k} = \lambda_{t,k}$ for $k \in \mathcal{M} \setminus \{0\}$. Then $\sum_{k\in\mathcal{M}} \lambda'_{t,k} = \sum_{k\in\mathcal{M}} \lambda_{t,k} + 1 - z_t = 1$ by (3.18) and so (3.17a) holds. Also, $v_t = \sum_{k\in\mathcal{M}} \hat{B}_k \lambda_{t,k} = \sum_{k\in\mathcal{M}} \hat{B}_k \lambda'_{t,k} + (1 - z_t)\hat{B}_0 = \sum_{k\in\mathcal{M}} \hat{B}_k \lambda'_{t,k}$ since $\hat{B}_0 = 0$, and so (3.17b) holds. (3.17c) and (3.17d) hold by an analogous

argument. Thus, $(z, x, v, y, w, \lambda') \in \mathcal{R}(\text{PLA})$ and the first claim follows. Next, consider an instance with $T = 1$, $\mathcal{P} = \emptyset$ (so there are no y or w variables and the index t can be dropped) and with break points $(\hat{B}_0, \hat{B}_1, \hat{B}_2) = (0, 1/2, 1)$ and function values $(\hat{F}_0, \hat{F}_1, \hat{F}_2) = (0, 1/2, 0)$. The point $(z, x, v) = (1/2, 1/2, 1/2)$ is in $\text{Proj}_{(z,x,v)}\mathcal{R}(\text{PLA})$ since it is easy to check that $(z, x, v, \lambda) \in \mathcal{R}(\text{PLA})$ with $\lambda = (0, 1, 0)$. However, $(1/2, /1/2, /1/2) \notin \text{Proj}_{(z,x,v)}\mathcal{R}(\text{PLA-S})$ as there is no $\lambda' \in \mathbb{R}^3_+$ that satifies (3.18), (3.17b) and (3.17c). Indeed, (3.18) and (3.17b) take the form $\lambda'_0 + \lambda'_1 + \lambda'_2 = 1/2$ and $(1/2)\lambda'_1 + \lambda'_2 = 1/2$, which because $\lambda' \geqslant 0$ implies $\lambda'_0 = \lambda'_1 = 0$ and $\lambda'_2 = 1/2$. But then (3.17c) is violated because $x = 1/2 > 0 = \sum_{k=0}^{2} \lambda'_k \hat{F}_k$. □

Results in [90] also demonstrate that MILP formulations of similar structure which use (3.18) are *locally ideal* [74] in the sense that the extreme points of the LP relaxation of such formulations, in the absence of other constraints, satisfy the integrality/SOS2 property.

The next technique for improving the formulation PLA exploits the fact that we are building piecewise-linear models of multiple functions, and that the arguments of these functions, v_t for $t \in \mathcal{T}$, are related by the inequalities $v_t \geqslant v_{t-1}$ for $t \in \mathcal{T}$.

Proposition 3.3. *Let* $(z, x, v, y, w, \lambda) \in$ PLA-S. *Then,* λ *satisfies the inequalities:*

$$\sum_{k=k'}^{m} \lambda_{t,k} \geqslant \sum_{k=k'}^{m} \lambda_{t-1,k}, \quad k' \in \mathcal{M}, t \in \mathcal{T} \setminus \{1\}. \tag{3.19}$$

Proof. Let $t \in \mathcal{T} \setminus \{1\}$ and $k' \in \mathcal{M}$. If $z_{t-1} = 0$, then $\sum_{k=0}^{m} \lambda_{t-1,k} = 0$ and so (3.19) holds trivially. Therefore, assume $z_{t-1} > 0$, and so also $z_t > 0$ because $z_t \geqslant z_{t-1}$. Then, let $k_1^* = \max\{k \in \mathcal{M} \mid \lambda_{t-1,k} > 0\}$ and $k_2^* = \max\{k \in \mathcal{M} \mid \lambda_{t,k} > 0\}$. Then,

because $v_t \geqslant v_{t-1}$, (3.17e) implies $k_1^* \leqslant k_2^*$. Observe that, if $k' > k_1^*$, then (3.19) is trivial because the right-hand side is zero, so assume $k' \leqslant k_1^*$. If $k_1^* = 0$, then $k' = 0$ and (3.19) holds because $\sum_{k=0}^{m} \lambda_{t,k} = z_t \geqslant z_{t-1} = \sum_{k=0}^{m} \lambda_{t-1,k}$. Thus, assume $k_1^* > 0$. Then (3.17e) implies $\lambda_{t-1,k} = 0$ for all $k \notin \{k_1^*, k_1^* - 1\}$ and $\lambda_{t,k} = 0$ for all $k \notin \{k_2^*, k_2^* - 1\}$. Next, if $k' < k_2^*$ then

$$\sum_{k=k'}^{m} \lambda_{t,k} = z_t \geqslant z_{t-1} \geqslant \sum_{k=k'}^{m} \lambda_{t-1,k}$$

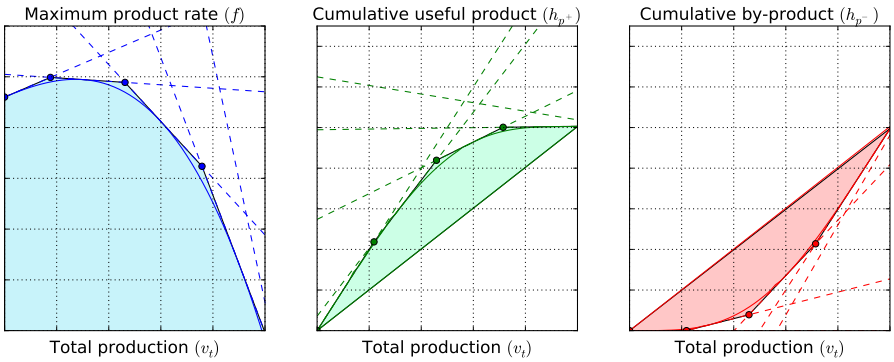
so that (3.19) holds. The remaining case is $k' \geqslant k_2^*$ and $k' \leqslant k_1^*$, which reduces to $k' = k_1^* = k_2^*$ because $k_1^* \leqslant k_2^*$. Then (3.17b) implies that $\hat{B}_{k'-1}\lambda_{t-1,k'-1} + \hat{B}_{k'}\lambda_{t-1,k'} = v_{t-1}$ and $\hat{B}_{k'-1}\lambda_{t,k'-1} + \hat{B}_{k'}\lambda_{t,k'} = v_t$. Then, using $v_t \geqslant v_{t-1}$, $\lambda_{t-1,k'-1} + \lambda_{t-1,k'} = 1$, and $\lambda_{t,k'-1} + \lambda_{t,k'} = 1$, it follows that $\lambda_{t,k'} \geqslant \lambda_{t-1,k'}$ which implies (3.19). □

We define PLA-SS as the formulation obtained by adding the valid inequalities (3.19) to PLA-S, specifically:

$$\text{PLA-SS} = \{(z, x, v, y, w, \lambda) \in W : \ (3.17b) - (3.17e),\ (3.18),\ (3.19)\}.$$

A special property of this problem is that for each $t \in \mathcal{T}$, the nonlinear production functions $f(v_t)$ and $h_p(v_t), p \in \mathcal{P}$ are functions of the same argument v_t. We have exploited this property by introducing a *single* set of SOS2-constrained variables $\{\lambda_{t,k} \mid k \in \mathcal{M}\}$ to simultaneously approximate all these functions of v_t. In order to do this, we require that the set of break points in the interval $[0, M^v]$ used to build the piecewise-linear approximations of $f(v_t)$ and $h_p(v_t), p \in \mathcal{P}$ must be the same for each of these functions. The potential drawback of this is that this

Figure 3.4: Piecewise-linear relaxation model with a single secant (SEC).



added restriction may lead to a less accurate approximation of these functions for the same number of break points, compared to an approach that allows the break points to be selected separately for each function. However, the latter approach would require introducing a different set of SOS2-constrained variables for each function and each t, yielding a formulation with many more variables, and more significantly, many more SOS2 constraints.

Relaxations

In this section we present methods for using piecewise-linear functions to obtain MILP formulations that are a relaxation of MINLP2. Thus, the optimal value of a problem in which one of these MILP formulations is used in place of MINLP2 yields a bound on the best possible solution to MINLP2.

Secant Relaxation (SEC**)**

To obtain the first relaxation, we assume we choose break points $\bar{B}$ and functions values $\bar{F}$ and $\bar{H}_p, p \in \mathcal{P}$ such that

$$f(v) \leqslant \bar{f}(v; \bar{B}, \bar{F}), \qquad v \in [0, M^v], \tag{3.20a}$$

$$h_p(v) \leqslant \bar{h}_p(v; \bar{B}, \bar{H}_p), \quad v \in [0, M^v],\ p \in \mathcal{P}^+ \tag{3.20b}$$

$$h_p(v) \geqslant \bar{h}_p(v; \bar{B}, \bar{H}_p), \quad v \in [0, M^v],\ p \in \mathcal{P}^- \tag{3.20c}$$

and such that the piecewise-linear functions $\bar{f}(v; \bar{B}, \bar{F})$ and $\bar{h}_p(v; \bar{B}, \bar{H}_p), p \in \mathcal{P}^+$ are concave, the piecewise-linear functions $\bar{h}_p(v; \bar{B}, \bar{H}_p), p \in \mathcal{P}^-$ are convex, and $\bar{h}_p(0; \bar{B}, \bar{H}_p) = 0$ for $p \in \mathcal{P}$. Figure 3.4 illustrates one such choice of $\bar{B}, \bar{F}$, and $\bar{H}_p, p \in \mathcal{P}$.

We now obtain a relaxation of MINLP2 by replacing equations (3.5d) and (3.12a) with the following:

$$x_t \leqslant \Delta_t \bar{f}(v_{t-1}; \bar{B}, \bar{F}), \qquad t \in \mathcal{T} \tag{3.21a}$$

$$(\bar{H}_{p,m}/M^v) v_t \leqslant w_{p,t} \leqslant \bar{h}_p(v_t; \bar{B}, \bar{H}_p), \qquad p \in \mathcal{P}^+, t \in \mathcal{T} \tag{3.21b}$$

$$\bar{h}_p(v_t; \bar{B}, \bar{H}_p) \leqslant w_{p,t} \leqslant \left(\bar{H}_{p,m}/M^v\right) v_t, \qquad p \in \mathcal{P}^-, t \in \mathcal{T}. \tag{3.21c}$$

It is immediate from (3.20a) that (3.21a) is a relaxation of (3.5d). For $p \in \mathcal{P}$, the constraint (3.12a), $w_{p,t} = h_p(v_t)$, is relaxed by using $\bar{h}_p(v_t; \bar{B}, \bar{H}_p)$ as an upper bound on $h_p(v_t)$. Because $h_p(v)$ is concave for $p \in \mathcal{P}^+$, the *secant* inequality $h_p(v) \geqslant h_p(0) + ((h_p(M^v) - h_p(0))/(M^v - 0))v = (\bar{H}_{p,m}/M^v)v$ is valid for $v \in [0, M^v]$.

Similarly, for byproducts $p \in \mathcal{P}^-$, the function $h_p(v_t)$ is convex so the secant provides an upper bound over $[0, M^v]$, and the lower bound is obtained using the piecewise-linear function $\bar{h}_p(v; \bar{B}, \bar{H}_p)$.

The constraints (3.21) can be compactly formulated using the variables $\lambda \in \mathbb{R}_+^{T \times (m+1)}$ with the constraints:

$$1 = \sum_{k \in \mathcal{O}} \lambda_{t,k}, \qquad t \in \mathcal{T} \tag{3.22a}$$

$$v_t = \sum_{k \in \mathcal{O}} \bar{B}_k \, \lambda_{t,k}, \qquad t \in \mathcal{T} \tag{3.22b}$$

$$x_t \leqslant \Delta_t \sum_{k \in \mathcal{O}} \bar{F}_k \, \lambda_{t-1,k}, \qquad t \in \mathcal{T} \tag{3.22c}$$

$$w_{p,t} = \sum_{k \in \mathcal{O}} \bar{H}_{p,k} \, \lambda_{t,k}, \qquad p \in \mathcal{P}, t \in \mathcal{T}. \tag{3.22d}$$

The equations (3.22b) and (3.22d) enforce that for each $p \in \mathcal{P}, t \in \mathcal{T}$, $(v_t, w_{p,t})$ are written as a convex combination of the points $(\bar{B}_k, \bar{H}_{p,k})$, $k \in \mathcal{O}$, which is equivalent to (3.21b) for $p \in \mathcal{P}^+$ and to (3.21c) for $p \in \mathcal{P}^-$. Observe that, although we use λ variables and the convex combination constraints (3.22a) as in the formulation PLA, in SEC we *do not* need to enforce the SOS2 constraints as in (3.17e). This is because in this case, for a fixed binary vector z, the feasible region we are modeling is a polyhedral set as illustrated in Figure 3.4. Note that this polyhedral set could alternatively be formulated using linear inequalities to define the lower and upper limits of the regions. We choose to use this extreme point based formulation of these polyhedra because this requires fewer constraints and has a close connection to the next formulation we will present.

We let SEC be the MILP formulation defined as follows:

$$\text{SEC} = \{(z, x, v, y, w, \lambda) \in W : \ (3.22),\ v_t \leqslant M^v z_t,\ t \in \mathcal{T}\}.$$

Similar to formulation PLA, we share the same sets of break points $\{\bar{B}_k \mid k \in \mathcal{O}\}$ across the $P + 1$ piecewise-linear functions $\bar{f}(\cdot\,; \bar{B}, \bar{F})$ and $\bar{h}(\cdot\,; \bar{B}, \bar{H}_p), p \in \mathcal{P}$.

As with formulation PLA, we can improve the LP relaxation of the formulation SEC by replacing the constraint (3.22a) with (3.18), and also removing the constraints $v_t \leqslant M^v z_t,\ t \in \mathcal{T}$ which are then redundant, obtaining formulation SEC-S defined as

$$\text{SEC-S} = \{(z, x, v, y, w, \lambda) \in W : \ (3.18),\ (3.22b) - \ (3.22d)\}.$$

Because SEC does not include SOS2 constraints, the inequalities (3.19) will not improve the LP relaxation, and so are not considered for SEC-S.

k-Secant Relaxation (k-SEC)

As Figure 3.4 illustrates, the formulation SEC relaxes the constraints $w_{p,t} = h_p(v_t)$ to allow points that lie in the shaded feasible region. The use of a single secant inequality to obtain a lower bound on $h_p(v)$ for $p \in \mathcal{P}^+$, or an upper bound on $h_p(v)$ for $p \in \mathcal{P}^-$ may allow solutions in the relaxation that are far from being feasible. We can obtain a tighter relaxation by using piecewise-linear functions for this purpose, as illustrated in Figure 3.5.

As in Section 3.3, we assume we choose break points $\bar{B}$ and functions values $\bar{F}$

and $\bar{H}_p, p \in \mathcal{P}$ that satisfy (3.20). In addition, we choose function values $\widetilde{H}_p, p \in \mathcal{P}$ that satisfy

$$h_p(v) \geqslant \bar{h}_p(v; \bar{B}, \widetilde{H}_p), \quad v \in [0, M^v],\ p \in \mathcal{P}^+ \tag{3.23a}$$

$$h_p(v) \leqslant \bar{h}_p(v; \bar{B}, \widetilde{H}_p), \quad v \in [0, M^v],\ p \in \mathcal{P}^-. \tag{3.23b}$$

We can then obtain a relaxation of the constraints (3.5d), $x_t \leqslant \Delta_t f(v_{t-1}),\ t \in \mathcal{T}$ and (3.12a), $w_{p,t} = h_p(v_t),\ t \in \mathcal{T}$, using variables $\lambda \in \mathbb{R}_+^{T \times (m+1)}$, the constraints (3.22a) - (3.22c), and

$$\sum_{k \in \mathcal{O}} \widetilde{H}_{p,k}\, \lambda_{t,k} \leqslant w_{p,t} \leqslant \sum_{k \in \mathcal{O}} \lambda_{t,k} \bar{H}_{p,k}, \qquad p \in \mathcal{P}^+, t \in \mathcal{T} \tag{3.24a}$$

$$\sum_{k \in \mathcal{O}} \bar{H}_{p,k}\, \lambda_{t,k} \leqslant w_{p,t} \leqslant \sum_{k \in \mathcal{O}} \lambda_{t,k} \widetilde{H}_{p,k}, \qquad p \in \mathcal{P}^-, t \in \mathcal{T} \tag{3.24b}$$

$$\{\lambda_{t,k} \mid k \in \mathcal{O}\} \in \text{SOS2}, \qquad t \in \mathcal{T}. \tag{3.24c}$$

We let k-SEC be the resulting MILP formulation:

$$k\text{-SEC} = \big\{(z, x, v, y, w, \lambda) \in W :\ (3.22a) - (3.22c),\ (3.24),\ v_t \leqslant M^v z_t,\ t \in \mathcal{T}\big\}.$$

Figure 3.5 illustrates the difference between formulations SEC and k-SEC.

Again, as with formulation PLA, we can improve the LP relaxation of the formulation k-SEC by replacing the constraint (3.17a) with (3.18), and also removing the constraints $v_t \leqslant M^v z_t,\ t \in \mathcal{T}$ which are then redundant. In addition, because k-SEC includes SOS2 constraints, the inequalities (3.19) have the potential to further

improve the LP relaxation. We define the formulation k-SEC-SS as:

$$k\text{-SEC-SS} = \big\{(z, x, v, y, w, \lambda) \in W : \ (3.18),\ (3.19),\ (3.22b) - \ (3.22c),\ (3.24)\big\}.$$

Table 3.1: Summary of MILP formulations. All formulations contain variables $(z, x, v, y, w, \lambda) \in W$. The column SOS2 indicates whether or not the λ variables are SOS2-constrained.

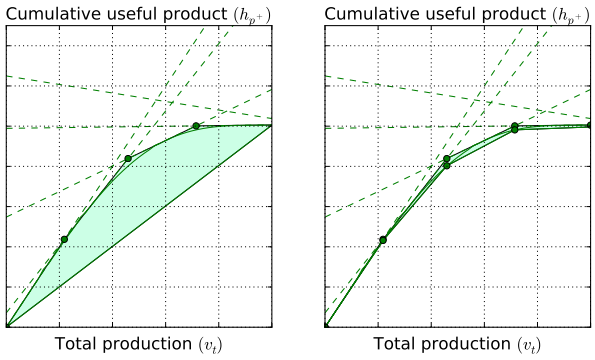
Model	Constraints	SOS2?	Description
PLA	(3.5c),(3.17)	Yes	Approximation of MINLP2
PLA-S	(3.17b)-(3.17e),(3.18)	Yes	Stronger LP relaxation than PLA
PLA-SS	(3.17b)-(3.17e),(3.18),(3.19)	Yes	Stronger LP relaxation than PLA-S
SEC	(3.5c),(3.22)	No	Relaxation of MINLP2
SEC-S	(3.18),(3.22b)-(3.22c)	No	Stronger LP relaxation than SEC
k-SEC	(3.5c),(3.22a)-(3.22c),(3.24)	Yes	Relaxation of MINLP2
k-SEC-SS	(3.18),(3.19),(3.22b)-(3.22c),(3.24)	Yes	Stronger LP relaxation than k-SEC

Table 3.1 summarizes the MILP formulations that have been introduced in this section.

3.4 Construction of piecewise-linear models

A key feature of our MILP formulations is that, for each period $t \in \mathcal{T}$, the functions $f(v_t)$ and $h_p(v_t)$ are approximated using piecewise-linear functions that share a common set of break points. In this section, we describe NLP formulations that

Figure 3.5: Comparing 1-SEC and k-SEC formulation (k = 2).



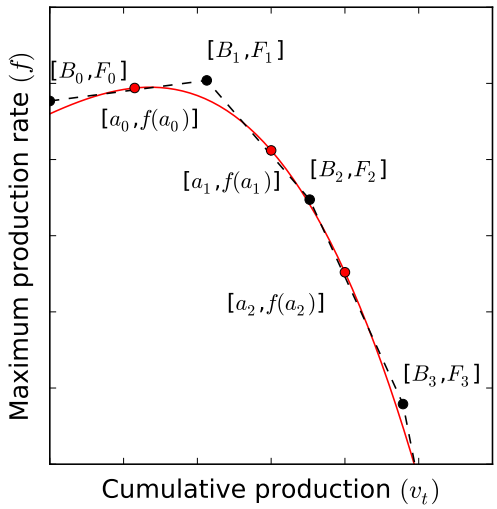
identify the tightest possible approximation/relaxation of multiple univariate convex/concave functions with piecewise-linear forms that share the same set of break points. These formulations are nonconvex, and hence we can only expect to achieve local optimal solutions. However, our computational experience reported in Section 3.5 indicates that these formulations are easily solved by a commercial NLP solver, and the solutions are high quality. Our development of these NLP formulations is motivated by the need to simultaneously approximate a set of nonlinear functions. However, another benefit of these formulations is that they are flexible in the sense that it is easy to impose additional requirements on the piecewise-linear functions produced, such as requiring them to be convex or concave, monotone increasing or decreasing, etc.

Piecewise-linear approximation

We now describe a method for obtaining piecewise-linear approximations of nonlinear functions, with the goal to minimize the maximum error of the approximation.

In this section, we extend previous work [17, 38, 48, 83] to propose an NLP formulation that identifies piecewise-linear approximations of multiple univariate convex/concave functions that share the same domain and break points.

Figure 3.6: Piecewise ($m = 4$) linear approximation of a concave function $f(\cdot)$.



Given a differentiable concave function $f(\cdot)$, differentiable concave functions $h_p(\cdot), p \in \mathcal{P}^+$, and differentiable convex functions $h_p(\cdot), p \in \mathcal{P}^-$, all defined on the domain $[0, M^v]$, we seek a common set of break points $B \in \mathbb{R}_+^{m+1}$ with $0 = B_0 < B_1 < \cdots < B_m = M^v$ and approximation function values $F \in \mathbb{R}^{m+1}$ and $H_p \in \mathbb{R}^{m+1}, p \in \mathcal{P}$, in order to minimize the weighted sum of maximum errors

between the given functions and the constructed piecewise-linear approximations:

$$\min_{B,F,H_p,p\in\mathcal{P}} w^f \epsilon^f + \sum_{p\in\mathcal{P}} w_p^h \epsilon_p^h, \tag{3.25a}$$

$$\text{subject to } \epsilon^f \geqslant |f(v) - \bar{f}(v;B,F)|, \qquad v \in [0, M^v] \tag{3.25b}$$

$$\epsilon_p^h \geqslant |h_p(v) - \bar{h}(v;B,H_p)|, \qquad v \in [0, M^v],\ p \in \mathcal{P} \tag{3.25c}$$

$$0 = B_0 \leqslant B_1 \leqslant \cdots \leqslant B_m = M^v \tag{3.25d}$$

where $w^f > 0$ and $w_p^h > 0, p \in \mathcal{P}$ are fixed weights. We use weights $w^f = 1/\max\{|f(v)| : v \in [0, M_v]\}$ and $w_p^h = 1/\max\{|h_p(v)| : v \in [0, M_v]\}\ \forall p \in \mathcal{P}$ to prevent any single function from dominating the objective function.

To formulate the semi-infinite program (3.25) as an NLP, we reformulate (3.25b) and (3.25c) using finitely many constraints using ideas from Geoffrion [38]. Consider first the concave function $f(\cdot)$, and consider an arbitrary interval $[B_{k-1}, B_k]$. Note that, on the this interval, the function $\bar{f}(\cdot;B,F)$ is affine:

$$\bar{f}(v;B,F) = F_{k-1} + \left(\frac{F_k - F_{k-1}}{B_k - B_{k-1}}\right)(v - B_{k-1}).$$

Because $f(\cdot)$ is concave, the function $err_f^+(v) := f(v) - \bar{f}(v;B,F)$ is also concave on the interval $[B_{k-1}, B_k]$ for fixed B, F, and so the maximum of $err_f^+(v)$ over this interval is given by $err_f^+(a_k)$, where a_k is the point at which the derivative of $err_f^+(v)$ equals zero. The point a_k satisfies:

$$f'(a_k) = \frac{F_k - F_{k-1}}{B_k - B_{k-1}} \Leftrightarrow f'(a_k)(B_k - B_{k-1}) = F_k - F_{k-1} \tag{3.26}$$

since we can assume $B_k > B_{k-1}$. See Figure 3.6 for an example of the point a_k. Observe that, using (3.26),

$$\begin{aligned} \mathrm{err}_f^+(a_k) &= f(a_k) - \left(F_{k-1} + \left(\frac{F_k - F_{k-1}}{B_k - B_{k-1}} \right) (a_k - B_{k-1}) \right) \\ &= f(a_k) - F_{k-1} - f'(a_k)(a_k - B_{k-1}). \end{aligned}$$

Thus, $\epsilon^f \geqslant \mathrm{err}_f^+(x)$ for all $x \in [B_{k-1}, B_k]$ if and only if there exists $a_k \in [B_{k-1}, B_k]$ that satisfies (3.26) and

$$\epsilon^f + F_{k-1} \geqslant f(a_k) - f'(a_k)(a_k - B_{k-1}).$$

Consider now $\mathrm{err}_f^-(v) := \bar{f}(v; B, F) - f(v)$. Because $f(\cdot)$ is concave, it holds that

$$\max\{\mathrm{err}_f^-(v) : v \in [B_{k-1}, B_k]\} = \max\{\mathrm{err}_f^-(B_{k-1}), \mathrm{err}_f^-(B_k)\}.$$

Thus, $\epsilon^f \geqslant \mathrm{err}_f^-(x)$ for all $x \in [B_{k-1}, B_k]$ if and only if

$$\epsilon^f \geqslant \mathrm{err}_f^-(B_i) = F_i - f(B_i), \quad i = k-1, k.$$

Putting this all together, we obtain that (3.25b) can be formulated using additional

variables $a_k, k = 1, \ldots, m$ and the constraints

$$\epsilon^f + F_{k-1} \geqslant f(a_k) - f'(a_k)(a_k - B_{k-1}), \quad k = 1, \ldots, m, \tag{3.27a}$$

$$\epsilon^f \geqslant F_k - f(B_k), \quad k = 0, 1, \ldots, m, \tag{3.27b}$$

$$f'(a_k)(B_k - B_{k-1}) = F_k - F_{k-1}, \quad k = 1, \ldots, m, \tag{3.27c}$$

$$B_{k-1} \leqslant a_k \leqslant B_k, \quad k = 1, \ldots, m. \tag{3.27d}$$

Identical arguments can be used to formulate constraints (3.25c), using additional variables $b_{k,p}, k = 1, \ldots, m, p \in \mathcal{P}$ and the constraints:

$$\epsilon^h_p + H_{k-1,p} \geqslant h_p(b_{k,p}) - h'_p(b_{k,p})(b_{k,p} - H_{k-1,p}), \quad k = 1, \ldots, m, \quad p \in \mathcal{P}^+, \tag{3.28a}$$

$$\epsilon^h_p - H_{k-1,p} \geqslant -h_p(b_{k,p}) + h'_p(b_{k,p})(b_{k,p} - H_{k-1,p}), \quad k = 1, \ldots, m, \quad p \in \mathcal{P}^-, \tag{3.28b}$$

$$\epsilon^h_p \geqslant H_{k,p} - h_p(B_k), \quad k = 0, 1, \ldots, m, p \in \mathcal{P}^+, \tag{3.28c}$$

$$\epsilon^h_p \geqslant h_p(B_k) - H_{k,p}, \quad k = 0, 1, \ldots, m, p \in \mathcal{P}^-, \tag{3.28d}$$

$$h'_p(b_{k,p})(B_k - B_{k-1}) = H_{k,p} - H_{k-1,p}, \quad k = 1, \ldots, m, \quad p \in \mathcal{P}, \tag{3.28e}$$

$$B_{k-1} \leqslant b_{k,p} \leqslant B_k, \quad k = 1, \ldots, m, \quad p \in \mathcal{P}, \tag{3.28f}$$

where the constraints (3.28a) and (3.28c) are analogous to (3.27a) and (3.27b) since the functions h_p, $p \in \mathcal{P}^+$ are concave, whereas (3.28b) and (3.28d) follow because the functions h_p, $p \in \mathcal{P}^-$ are convex.

We also require that the piecewise-linear functions $\bar{h}_p(v; B, H_p)$ be exact at $v = 0$, and hence we enforce

$$H_{0,p} = 0, \quad p \in \mathcal{P}. \tag{3.29}$$

We therefore obtain the NLP formulation:

$$\min_{B,F,H,a,b} w^f \epsilon_f + \sum_{p \in \mathcal{P}} w^h_p, \tag{3.30a}$$

$$\text{subject to } (3.27),\ (3.28),\ (3.29),\ B_0 = 0,\ B_m = M^v. \tag{3.30b}$$

where the inequalities $B_0 \leqslant B_1 \leqslant \cdots \leqslant B_m$ are omitted because they are implied by (3.27d). We use a locally optimal solution obtained by solving this NLP to define the break points $\hat{B}$ and function values $\hat{F}$ and $\hat{H}_p$, $p \in \mathcal{P}$, used in Section 3.3.

Piecewise-linear relaxation

We now describe an NLP formulation that can be used to generate piecewise-linear functions that provide lower and upper bounds for a set of convex and concave functions, respectively. The key new feature of our approach is that it simultaneously approximates a set of functions sharing the same domain using the same set of break points for each function. Our work builds on ideas of algorithms [17, 83, 94] that find tight piecewise-linear upper and lower approximations of a single univariate

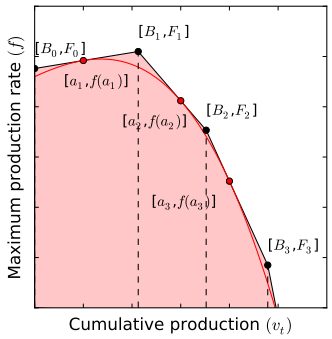
convex function.

We seek piecewise-linear upper bounds on the concave differentiable production functions $f(\cdot)$, $h_p(\cdot), p \in \mathcal{P}^+$ and lower bounds on the convex differentiable functions $h_p(\cdot), p \in \mathcal{P}^-$. All these functions share the same domain $[0, M^v]$. Observe that $f(\cdot)$ and $h_p(\cdot)$ are non-negative over $[0, M^v]$. The goal of the NLP formulation we propose is to minimize a weighted sum of the areas between the curves of the functions being approximated and their corresponding piecewise-linear approximations.

The primary variables in the formulation are the break points $B \in \mathbb{R}^{m+1}$, and function values $F \in \mathbb{R}^{m+1}$ and $H_p \in \mathbb{R}^{m+1}, p \in \mathcal{P}$. The domain $[0, M^v]$ is divided into m intervals, $[B_{k-1}, B_k]$, $k = 1, \ldots, m$, and the constructed piecewise-linear functions are linear on each interval, defined according to (3.14) and (3.15). On each interval, for each function being relaxed we require there be a point in the interval such that the piecewise-linear function is tangent to the function being approximated at that point. For the function $f(\cdot)$, we introduce variables a_k, $k = 1, \ldots, m$, to identify these points, and for the functions $h_p(\cdot), p \in \mathcal{P}$, we introduce variables $b_{k,p}$, $k = 1, \ldots, m$ to identify these points. Figure 3.7 highlights the notation used in this section and depicts a possible four-piece relaxation of a single function $f(\cdot)$.

For each interval $k = 1, \ldots, m$, the piecewise-linear approximation of $f(\cdot)$ in that interval is defined by the affine function $\ell_k(v) = f(a_k) + f'(a_k)(v - a_k)$ obtained as the first-order approximation of $f(\cdot)$ at the point a_k. As $f(\cdot)$ is concave on $[0, M^v]$, it follows that $\ell_k(v) \geqslant f(v)$ on $[0, M^v]$. The break points, B_k, and function values F_k, $k \in \mathcal{M}$, are then determined uniquely by finding the intersection of consecutive

Figure 3.7: Piecewise-linear ($m = 4$) relaxation of a concave function $f(\cdot)$.



affine functions $\ell_k(v)$ and $\ell_{k+1}(v)$, as illustrated in Figure 3.7. Specifically, for $1 \leqslant k \leqslant m$, since F_k represents the value of the piecewise linear function at the point B_k, we must have $F_k = \ell_k(B_k)$, and for $0 \leqslant k < m$, we must also have $F_k = \ell_{k+1}(B_k)$, i.e.,

$$F_k = f(a_k) + f'(a_k)(B_k - a_k), \qquad k = 1, \ldots, m, \tag{3.31a}$$

$$F_k = f(a_{k+1}) + f'(a_{k+1})(B_k - a_{k+1}), \qquad k = 0, \ldots, m-1. \tag{3.31b}$$

We write similar constraints for the functions $h_p(\cdot)$:

$$H_{p,k} = h_p(b_{k,p}) + h'_p(B_k)(B_k - b_{k,p}), \qquad p \in \mathcal{P}, k = 1, \ldots, m, \tag{3.31c}$$

$$H_{p,k} = h_p(b_{k+1,p}) + h'_p(B_{k+1})(B_k - b_{k+1,p}), \qquad p \in \mathcal{P}, k = 0, \ldots, m-1. \tag{3.31d}$$

We measure the quality of the relaxation using the area between the two curves defined by the piecewise-linear approximation function and the corresponding function being relaxed. We combine the measures for the different functions

into a single objective function using weights w^f for $f(\cdot)$ and w^h_p for each function $h_p(\cdot), p \in \mathcal{P}$. We use weights of $w^f = 1/\int_0^{M_v} f(s)\,ds$ and $w^h_p = 1/\int_0^{M_v} h^p(s)\,ds$ for our experiments. We therefore obtain the following NLP formulation:

$$\min_{B,F,H,a,b} w^f\Big[\frac{1}{2}\sum_{k\in\mathcal{M}}(F_k + F_{k-1})(B_k - B_{k-1}) - \int_0^{M^v} f(s)\,ds\Big] \tag{3.32a}$$
$$+ \sum_{p\in\mathcal{P}^+} w^h_p\Big[\frac{1}{2}\sum_{k\in\mathcal{M}}(H_{p,k} + H_{p,k-1})(B_k - B_{k-1}) - \int_0^{M^v} h_p(s)\,ds\Big]$$
$$+ \sum_{p\in\mathcal{P}^-} w^h_p\Big[\int_0^{M^v} h_p(s)\,ds - \frac{1}{2}\sum_{k\in\mathcal{M}}(H_{p,k} + H_{p,k-1})(B_k - B_{k-1})\Big]$$

subject to (3.31), (3.29)

$$B_{k-1} \leqslant a_k \leqslant B_k, \quad k = 1,\dots,m \tag{3.32b}$$

$$B_{k-1} \leqslant b_{k,p} \leqslant B_k, \quad k = 1,\dots,m, p \in \mathcal{P} \tag{3.32c}$$

$$B_0 = 0,\ B_m = M^v, \tag{3.32d}$$

where we again include the constraints (3.29) which enforce that $\bar{h}_p(0; B, H_p) = 0$, $p \in \mathcal{P}$. The first term in the objective calculates the difference between the area under the piecewise-linear approximation of $f(\cdot)$, calculated by summing the area under each piece, and the area under the curve defined by $f(\cdot)$, calculated with the integral $\int_0^{M^v} f(s)\,ds$. Similar calculations are done for the by-product functions $h_p(\cdot)$, $p \in \mathcal{P}$. Since the definite integrals $\int_0^{M^v} f(s)\,ds$ and $\int_0^{M^v} h_p(s)\,ds$ are constants, they can be eliminated from the objective in (3.32). We use a locally optimal solution obtained from solving this NLP to define the break points $\bar{B}$ and functions values $\bar{F}$ and $\bar{H}_p, p \in \mathcal{P}$ used in the SEC and k-SEC models in Section 3.3.

For the k-SEC model, we also require functions values $\widetilde{H}_p, p \in \mathcal{P}$ that satisfy (3.23). Having obtained the break points $\bar{B}$ from the NLP (3.32), we simply set

$$\widetilde{H}_{p,k} = h(B_{p,k}), \quad k = 0, 1, \ldots, m, p \in \mathcal{P}.$$

Then, for $p \in \mathcal{P}^+$, $\bar{h}(v; \bar{B}, \widetilde{H}_p) \leqslant h_p(v)$ for $v \in [0, M^v]$ by concavity of $h_p(\cdot)$, and similarly for $p \in \mathcal{P}^-$, $\bar{h}(v; \bar{B}, \widetilde{H}_p) \geqslant h_p(v)$ for $v \in [0, M^v]$ by convexity of $h_p(\cdot)$, and so (3.23) is satisfied.

3.5 Computational Results

We conducted numerical experiments to test the effectiveness of formulation MINLP2, the piecewise-linear approximation and relaxation MILP problems, and our NLP formulations for obtaining the piecewise-linear approximations and relaxations. The goals of these experiments are to: (1) compare the accuracy of formulations MINLP1 and MINLP2, (2) compare the computational effort of solving MINLP1 using a state of the art global optimization solver and approximations of MINLP2 using the three MILP formulations PLA, SEC, k-SEC, and (3) measure the quality of the piecewise-linear function approximations and relaxations produced by formulations (3.30) and (3.32), respectively.

We used Gurobi 4.5.1 to solve all MILPs, Conopt 3.14 to solve all non-linear programs to local optimality except formulations (3.30) and (3.32) for which we used Knitro 7.0.0 to exploit the multistart feature. We used BARON 9.3.1 to solve all MINLPs to global optimality. We used the default options for all solvers except

that the LP method of Gurobi was changed to the barrier algorithm for better performance and the NLP solver of BARON was changed to Conopt 3.14 (from MINOS) to avoid some observed numerical issues. Section 3.5 describes the multistart options set for Knitro 7.0.0. We ran all instances with a maximum time limit of 1 hour, relative optimality gap of 0.1% and feasibility tolerance of 10^{-5}. We ran all experiments with a single thread on a 2.30GHz Intel E5-2470 Xeon processor with 128GB RAM.

Sample Application

We conducted our numerical experiments on a multi-period production and distribution planning problem. In this problem, the production facilities ($\mathcal{I}$) produce a mixture of products ($\mathcal{P}$), of which the useful products $\mathcal{P}^+$ are supplied to match the demand of customers ($\mathcal{J}$). The manufacturing process also produces byproducts ($\mathcal{P}^-$) which are not sent to customers, but do incur a processing cost. The development of each facility $i \in \mathcal{I}$ over the planning horizon ($\mathcal{T}$) involves deciding when to open the facility and how much of the product mixture to produce over time. We let d_{jpt} denote the known demand for product $p \in \mathcal{P}^+$, at time period $t \in \mathcal{T}$ for each customer $j \in \mathcal{J}$.

Opening a facility $i \in \mathcal{I}$ at time period $t \in \mathcal{T}$ incurs a fixed cost α_{it}. Each unit of product $p \in \mathcal{P}$ processed at facility $i \in \mathcal{I}$ in time period $t \in \mathcal{T}$ incurs a processing cost of β_{ipt}. The cost of shipping a unit of product $p \in \mathcal{P}^+$ from facility $i \in \mathcal{I}$ to customer $j \in \mathcal{J}$ during time period $t \in \mathcal{T}$ is γ_{ijpt}, and δ_{jpt} represents the per unit penalty cost of unsatisfied demand of product $p \in \mathcal{P}^+$ for customer $j \in \mathcal{J}$.

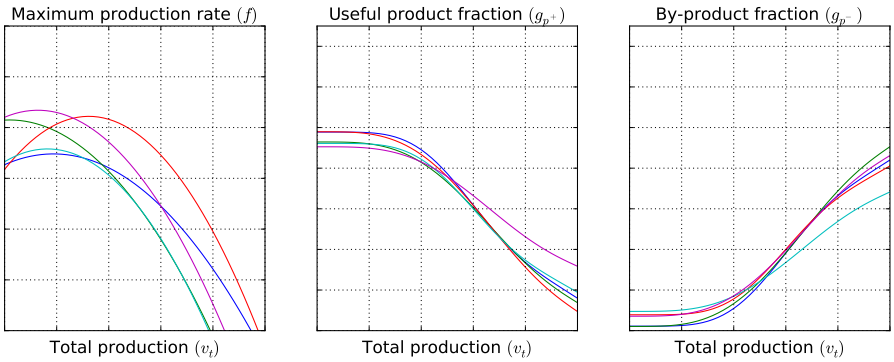
Each facility $i \in \mathcal{I}$ operates according to the production process described in the introduction. Thus, in the discrete-time formulation of this problem, for each facility $i \in \mathcal{I}$, we have decision variables x_{it} and v_{it}, which represent the total mixture production during time period $t \in \mathcal{T}$ and cumulative mixture production up to and including time period $t \in \mathcal{T}$, respectively. The binary decision variables $z_{it}, t \in \mathcal{T}$ determine the time when each facility $i \in \mathcal{I}$ is opened (if at all), and the variables y_{ipt} represent the amount of each product $p \in \mathcal{P} = \mathcal{P}^+ \cup \mathcal{P}^-$ produced during time period $t \in \mathcal{T}$ at facility $i \in \mathcal{I}$.

The facilities are linked by the need to supply demand to the common set of customers $j \in \mathcal{J}$. Let q_{ijpt} define a decision variable that represents the amount of product $p \in \mathcal{P}^+$ transported from production facility $i \in \mathcal{I}$ to customer $j \in \mathcal{J}$ during time period $t \in \mathcal{T}$, and let u_{jpt} be a decision variable representing the unsatisfied demand of product $p \in \mathcal{P}^+$ for customer $j \in \mathcal{J}$ during time period $t \in \mathcal{T}$. Then, the multi-period production and distribution problem is:

$$
\begin{aligned}
\min \quad & \sum_{i\in\mathcal{I}} \Big(\sum_{i\in\mathcal{I}} \alpha_{it} z_{it} + \sum_{i\in\mathcal{I}} \sum_{p\in\mathcal{P}} \beta_{ipt} y_{ipt} + \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}} \gamma_{ijpt} q_{ijpt} + \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}} \delta_{jpt} u_{jpt} \Big) \\
\text{subject to} \quad & \sum_{j\in\mathcal{J}} q_{ijpt} = y_{i,p,t} && i \in \mathcal{I}, p \in \mathcal{P}^+, t \in \mathcal{T} \\
& \sum_{i\in\mathcal{I}} q_{ijpt} = d_{jpt} - u_{jpt}, && j \in \mathcal{J}, p \in \mathcal{P}^+, t \in \mathcal{T} \qquad \text{(PP-X)} \\
& (z_i, x_i, v_i, y_i) \in X_i, && i \in \mathcal{I} \\
& u_{jpt} \geqslant 0, && j \in \mathcal{J}, p \in \mathcal{P}^+, t \in \mathcal{T}
\end{aligned}
$$

Here X_i denotes the set of constraints defining the production process for each

Figure 3.8: Sample production functions used for numerical experiments.



facility $i \in \mathcal{I}$. One can model the production set in this production and distribution problem using either MINLP1 or MINLP2, which results in two different problems denoted by PP-MINLP1 and PP-MINLP2, respectively.

Test Instances

We report tests conducted on randomly generated instances of the sample application problem. All aspects of the dataset, including customer demands and unit costs, the production functions $f_i(\cdot)$ and the product production ratio function $g_{i,p}(\cdot)$ for each facility $i \in \mathcal{I}$ were generated randomly. Figure 3.8 illustrates some sample production functions that were used in our instances. We solved instances categorized by the number of binary variables. We grouped instances as small ($<$ 150 binary variables), and large ($>$200 binary variables). We measured formulation accuracy on small instances and computational impact on large instances.

Table 3.2: Solution statistics for formulation MINLP1 on 12 small instances.

			MINLP1	Δy_{ipt}		Gap to Best
$\|\mathcal{I}\|$	$\|\mathcal{T}\|$	$\|\mathcal{P}\|$	Gap	Max	Avg	MINLP2 Obj
5	5	2	17.5	3.75	0.85	29.9
5	5	2	13.8	2.64	0.60	27.4
5	5	2	15.7	3.02	0.71	24.8
5	10	2	24.5	2.43	0.44	19.0
5	10	2	20.6	2.22	0.54	19.3
5	10	2	27.1	2.37	0.45	17.3
10	10	2	35.1	3.29	0.59	19.4
10	10	2	32.6	2.98	0.55	15.6
10	10	2	35.1	2.27	0.55	17.7
10	15	2	34.8	2.62	0.44	13.5
10	15	2	33.6	2.53	0.45	12.6
10	15	2	35.5	2.40	0.38	11.3

Dataset Generation

We now discuss the procedure used to generate the datasets for the sample applicaiton problem (PP-X).

Production functions We normalized the range of the total production function so that the cumulative production variable v has takes values in $[0, 1]$. We used bounded, concave production functions of the form $f(x) = r_0 + r_1 x + r_2 x^2$ with $r_2 < 0$. We randomly generated $f(\cdot)$ by choosing $r_0 \sim U(15, 25)$, $r_2 \sim U(-50, 0)$ and $r_0 + r_1 + r_2 \sim U(-15, 0)$. This procedure was repeated to generate independently and identically distributed samples of production functions $f_i(\cdot)$ for each facility $i \in \mathcal{I}$. The family of functions generated by this procedure always ensures that $f(\cdot)$ is concave.

For each product $p \in \mathcal{P}$, we must generate either a monotonically increasing or decreasing function. Additionally, we must ensure that $\sum_{p\in\mathcal{P}} g_p(v) = 1, \forall v \in [0,1]$. We simulated such functions by scaling and translating a randomly generated function from a family of functions. One such family of monotonically increasing functions is $\hat{g}(v) = \int_0^v u^k(1-u)^k du / \int_0^1 u^k(1-u)^k du, k = \{1,2,3\}$. We chose this family of functions because (a) it was rich enough to express productions functions of different kinds, (b) these functions are monotonically increasing, and (c) these functions satisfy $\hat{g}(0) = 0$ and $\hat{g}(1) = 1$. We then obtain functions $g_p(v), p \in \mathcal{P}$ by choosing positive real numbers a_p and b_p for $p \in \mathcal{P}$ and then setting

$$g_p(v) = a_p + (b_p - a_p)\hat{g}(v) \quad \forall p \in \mathcal{P}.$$

Since the family of generated functions must satisfy $\sum_{p\in\mathcal{P}} g_p(v) = 1, \forall v \in [0,1]$, we require

$$\sum_{p\in\mathcal{P}^+} a_p + \sum_{p\in\mathcal{P}^-} a_p = 1 \tag{3.33a}$$

$$\sum_{p\in\mathcal{P}^+} b_p + \sum_{p\in\mathcal{P}^-} b_p = 1 \tag{3.33b}$$

where $a_p = g_p(0)$ and $b_p = g_p(1)$. Additionally, we must also ensure that

$$a_p > b_p \quad \forall p \in \mathcal{P}^+ \tag{3.33c}$$

$$a_p < b_p \quad \forall p \in \mathcal{P}^- \tag{3.33d}$$

which makes the useful product ratio functions decreasing and the byproduct ratio function increasing. The following sampling procedure generates product ratio functions which satisfy all the required conditions.

- We sampled $\hat{a}_p \sim U(0.8, 1), \forall p \in \mathcal{P}^+$ and $\hat{a}_p \sim U(0, 0.2), \forall p \in \mathcal{P}^-$.
- We normalized the above samples so that $a_p = \frac{\hat{a}_p}{\sum_{p \in \mathcal{P}} \hat{a}_p}, \forall p \in \mathcal{P}$ which ensures that condition (3.33a) is satisfied.
- Next, we sampled $\hat{b}_p \sim U(0, \frac{a_p}{2})\ \forall p \in \mathcal{P}^+$ and $\hat{b}_p \sim U(2a_p, 1)\ \forall p \in \mathcal{P}^-$ which ensures that conditions (3.33c) and (3.33d) are satisfied.
- We normalized the above samples so that $b_p = \frac{\hat{b}_p}{\sum_{p \in \mathcal{P}} \hat{b}_p}, \forall p \in \mathcal{P}$ which ensures that condition (3.33b) is satisfied.

Customer Demands We generated increasing demand profiles as follows. For each customer $j \in \mathcal{J}$ and product $p \in \mathcal{P}$, we randomly generated a demand time frame uniformly from start-time between $[1, \frac{T}{3}]$ and end-time between $[\frac{2T}{3}, T]$. We generated demands $\{d_{jpt}\} \sim U(0.8, 1.2)$ for each time period within the demand time frame, sorted them in increasing order and scaled them by $\frac{\sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}^+} h_{ip}(1)}{3|\mathcal{J}|}$. Here, $h_{i,p}(1)$ is the cumulative production of each useful product $p \in \mathcal{P}^+$ from a facility $i \in \mathcal{I}$. The scaling ensures that the total demand across all customers is roughly a third of the total possible production of useful products over all facilities.

Costs We used the following procedure to generate the four different types of costs incurred at each facility. This procedure ensured that at the end of the planning

horizon, the total facility opening cost is of the same order as the sum of operating, production and penalty costs. For each customer $j \in \mathcal{J}$, facility $i \in \mathcal{I}$ and product $p \in \mathcal{P}$ during the first time period, we generated operating costs $\beta_{i,p,1} \sim U(0.8, 1.2)$, transportation costs $\gamma_{ijp1} \sim U(0.8, 1.2)$, penalty costs $\delta_{j,p,1} \sim U(16, 24)$ and fixed costs $\alpha_{i1} \sim \frac{\sum_{p \in \mathcal{P}+} h_{ip}(1)}{3|\mathcal{J}|} U(0.8, 1.2)$. We discounted costs at 5% for subsequent time periods.

Inaccuracy and Computational Difficulty of MINLP1

We first demonstrate the disadvantages of the formulation based on MINLP1, in terms of the difficulty in solving the problem, and the inaccuracy of the resulting solution. This experiment is performed on relatively small instances (< 150 binary variables) of our sample application problem.

First, to analyze the computational difficulty of solving these small instances, we use BARON to attempt to solve PP-MINLP1, with a ten hour time limit. None of these instances are solved to optimality, and so we obtain the ending optimality gap, which is defined as $(UB - LB)/UB$, where UB and LB are the best upper and lower bounds, respectively, obtained by BARON in the time limit. Table 3.2 displays statistics about these values for twelve small test instances. Each row in this table corresponds to a different instance, and the first two columns describe the size of the instance in terms of the number of facilities $|\mathcal{I}|$ and the number of time periods $|\mathcal{T}|$. The column 'MINLP1 Gap' provides the ending optimality gaps of these instances. These large remaining gaps indicate that it is not possible to obtain provably near-optimal solutions of MINLP1 using the general-purpose global optimization solver

like BARON.

We next study the quality of the solutions obtained using MINLP1. This is somewhat challenging because MINLP1 and MINLP2 represent two *different* approximations of the underlying problem. In particular, MINLP1 is based on the assumption that the fraction of products produced during period t is equal to $h_p(v_{t-1})$ throughout the period. Since $h_p(v)$ is increasing in v for by-products and decreasing in v for desirable products, this assumption is optimistic, and leads to an over-estimation of the actual obtainable objective value. In contrast, MINLP2 is based on an exact calculation of the amount of each product p produced during each period, so the objective value in MINLP2 is a more accurate representation of the actual cost. Because of this difference, we cannot simply compare the objective value of a solution obtained form MINLP1 to a solution obtained using MINLP2.

To overcome this challenge, we first obtain a solution to PP-MINLP1, say $(z^1, x^1, v^1, y^1, q^1, u^1)$, and then *repair* the solution to obtain a feasible solution to the more accurate formulation PP-MINLP2. The solution $(z^1, x^1, v^1, y^1, q^1, u^1)$ is taken as the best feasible solution to PP-MINLP1 found by BARON within the time limit. To repair the solution, for each facility i, we fix the decision variables z_i^1, x_i^1, v_i^1, and then use the equations (3.12) to exactly calculate the amount of each product $p \in \mathcal{P}$ produced in each period $t \in \mathcal{T}$, given that x_{it}^1 units of the mixture are produced, and denote this amount by $y_{i,p,t}^2$. Then, with these values all fixed, we solve (PP-X), which then amounts to a transportation problem that determines how much of each product to ship from each facility to each customer in all time periods to minimize the transportation costs and penalty costs of unmet demand. This leads to a feasible

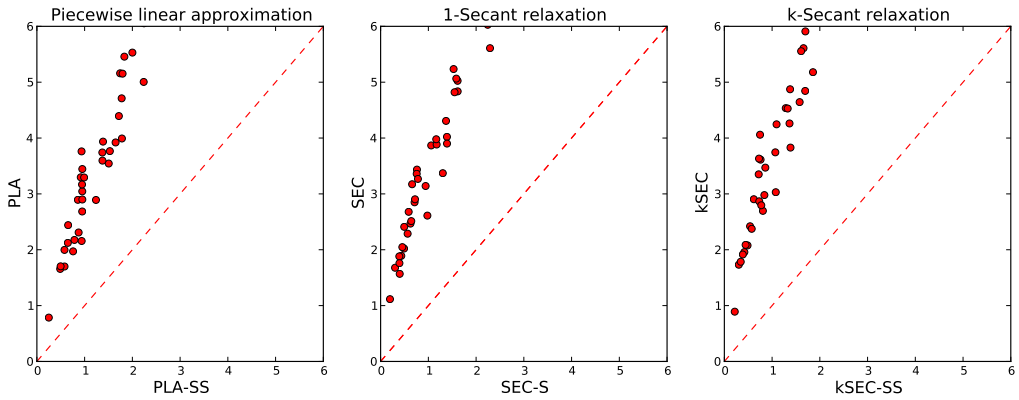
solution to the more accurate formulation PP-MINLP2.

One measure of the inaccuracy of the formulation MINLP1 is the difference between the values y^1_{ipt} obtained directly from solving PP-MINLP2, to the repaired values y^2_{ipt}, which we denote by $\Delta y_{ipt} = |y^1_{ipt} - y^2_{ipt}|$. The columns under the heading Δy_{ipt} in Table 3.2 display the maximum and average of Δy_{ipt} over all facilities, products, and time periods. For comparison purposes, the average and maximum value of the $y_{i,p,t}$ variables (over the solutions obtained in all instances) is 3.43 and 24.6, respectively. The average errors range between 0.38 and 0.85, and the maximum error ranges between 2.22 and 3.75. Thus, the estimates of the product quantities used in formulation MINLP1 are significantly different (at least 10% on average) from the actual.

In the last column of Table 3.2, we give the percentage difference between the cost of the repaired solution to the cost of the best known feasible solution to PP-MINLP2, obtained using methods that directly use the formulation MINLP2, as described in the following sections. Here the percentage gap is calculated as $(A - B)/A$ where A is the cost of the repaired solution and B is the cost of the best known solution. Thus, we see that the best known solutions to PP-MINLP2 are between 11.3% and 29.9% less costly than the repaired solution obtained from solving PP-MINLP1.

Finally, we observe that the gap between the formulation MINLP2 and MINLP1 reduces as the number of time periods ($|\mathcal{T}|$) increases, which is expected as the approximation error of formulatioin MINLP2 decreases as the period length decreases.

Figure 3.9: Effect of formulation strengthening on terminal MILP optimality gaps (%) of the three MILP formulations.



The effect of formulation strengthening on the MILP formulations

In our next experiment, we study the effect of using equations (3.18) and (3.19) to strengthen each of the three MILP formulations PLA-SS, SEC-S, and k-SEC-SS, compared to the formulations PLA, SEC,and k-SEC, which do not use these. These tests are conducted on 36 large instances of PP-MINLP2. These instances have between 15-20 facilities, 15-25 time periods, and 2-6 products, resulting in instances with 200-500 binary variables for all formulations and 200-500 SOS2 sets of variables for the PLA and k-SEC formulations. The piecewise-linear functions used in these MILP approximation all have three line segments that were constructed using the formulations in section 3.4 (see section 3.5 for implementation details). For each of these instances, we solved the MILP formulations with and without strengthening

with a time limit of one hour. We then found the ending optimality gap of the MILP instance, calculated as $(UB - LB)/UB$, where UB and LB are the best upper and lower bounds, respectively, obtained for the MILP formulation within the time limit. Figure 3.9 displays scatter plots of the resulting optimality gaps for the MILP formulations with and without strengthening. When using PLA-SS, SEC-S k-SEC-SS, most instances have ending optimality gap less than 2%. In comparison, when using PLA, SEC, k-SEC most instances have optimality gaps in the range of 2%-7%. Thus, it is clear that strengthening significantly helps the PLA, SEC as well as k-SEC models.

Table 3.3: Average linear programming relaxation gaps for MILP formulations of MINLP2 with and without strengthening while solving 36 large instances of PP.

$\|\mathcal{J}\|$	$\|\mathcal{T}\|$	$\|\mathcal{P}\|$	PLA	PLA-SS	SEC	SEC-S	k-SEC	k-SEC-SS
15	15	2	20.97	1.38	21.07	1.47	21.10	1.47
15	15	4	21.17	1.59	21.23	1.62	21.37	1.67
15	15	6	22.73	1.64	23.10	1.74	23.13	1.91
20	15	2	21.33	1.28	21.30	1.32	21.30	1.33
20	15	4	22.30	1.46	22.47	1.46	22.57	1.53
20	15	6	22.50	1.51	22.80	1.59	22.70	1.78
20	20	2	21.93	1.68	22.07	1.75	22.17	1.77
20	20	4	21.57	2.11	21.87	2.20	22.20	2.36
20	20	6	23.07	2.23	23.57	2.43	23.80	2.50
20	25	2	21.43	2.39	21.70	2.53	21.73	2.53
20	25	4	21.47	2.37	21.70	2.47	22.03	2.60
20	25	6	22.53	2.56	23.07	2.81	23.37	3.06
Arithmetic mean			21.92	1.85	22.16	1.95	22.29	2.04

We can further analyze the improvements obtained from using the strengthened formulations by comparing the quality of the linear programming (LP) relaxations.

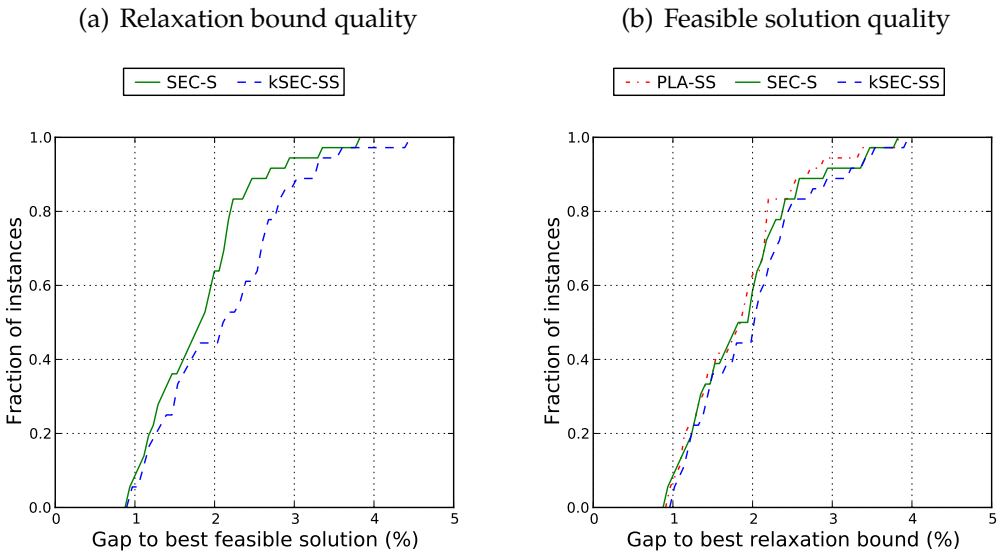
For an MILP instance, the LP relaxation gap is calculated as $(\hat{z} - z^{LP})/\hat{z}$, where z^{LP} is the LP relaxation objective value of the given instance, and $\hat{z}$ is the value of the best known feasible solution (found using *any* formulation for that instance). The LP relaxation gaps using the strengthened and unstrengthened versions of the three MILP formulations are summarized in Table 3.3. Each row in this table corresponds to an average across three different instances with the same size of $|\mathcal{I}|$, $|\mathcal{T}|$ and $|\mathcal{P}|$. The final row provides an average over all 36 instances. We observed more than a ten-fold improvement in the LP relaxation gaps across all three MILP formulation due to equations (3.18) and (3.19). In our remaining experiments, we use only the strengthened versions of the MILP formulations.

Comparing the MILP approximations and relaxations of MINLP2

We next compare the performance of the different piecewise-linear approximations and relaxations that we proposed in Section 3.3 for obtaining lower bounds and feasible solutions to PP-MINLP2. These tests are conducted on the same 36 large instances of PP-MINLP2 used in Section 3.5.

We used the MILP approximations and relaxations to obtain feasible solutions to PP-MINLP2 as follows. First, we solve each of the MILP problems PLA-SS, SEC-S, and k-SEC-SS with a one-hour time limit. Then, for each formulation, we take the best feasible MILP solution found, fix all the binary decision variables $\{z_{it}\ t \in \mathcal{T}\}$ for all facilities $i \in \mathcal{I}$, and then solve the resulting nonlinear program (NLP) using Conopt 3.14. This returns a feasible solution to PP-MINLP2, and hence provides an upper bound on the optimal objective value. The overall best upper bound for

Figure 3.10: Evaluating formulations PLA-SS, SEC-S, k-SEC-SS on large instances PP-MINLP2.



PP-MINLP2 is the minimum of the objective values of each of the feasible solutions found. The formulations SEC and k-SEC are relaxations of MINLP2. We solve these MILP relaxations with a time limit of one hour, and the lower bound on the MILP instance after that time limit is then a lower bound on the optimal objective value of problem PP-MINLP2. For each test instance, the larger of these two lower bounds is the best lower bound we obtain.

Figure 3.10 and Table 3.4 summarize the results of these experiments. We separately analyze the performance of these formulations in terms of quality of the lower bounds and the feasible solutions (upper bounds). Only SEC and k-SEC are guaranteed to provide lower bounds for PP-MINLP2. To compare the quality of these lower bounds, for each instance and each formulation we compute the gap between the lower bound obtained from the formulation and the *best* feasible

Table 3.4: Summary statistics for MILP approximations and relaxations of MINLP2 while solving 36 large instances of PP-MINLP2.

			Average gap to best feasible solution of PP-MINLP2 (%)		Average gap to best bound of PP-MINLP2 (%)		
$\|\mathcal{I}\|$	$\|\mathcal{T}\|$	$\|\mathcal{P}\|$	SEC-S	k-SEC-SS	PLA-SS	SEC-S	k-SEC-SS
15	15	2	1.02	1.01	1.03	1.00	1.09
15	15	4	1.01	1.32	1.06	1.10	1.14
15	15	6	1.81	2.58	1.85	1.83	1.98
15	20	2	1.16	1.14	1.16	1.16	1.21
15	20	4	1.34	1.59	1.36	1.38	1.44
15	20	6	2.02	2.78	2.03	2.06	2.23
20	20	2	1.47	1.42	1.45	1.45	1.58
20	20	4	1.91	2.33	1.91	1.98	2.07
20	20	6	2.58	2.88	2.62	2.60	2.94
25	20	2	1.98	2.00	2.00	2.04	2.15
25	20	4	2.17	2.54	2.17	2.38	2.41
25	20	6	3.27	3.76	3.27	3.55	3.61
Arithmetic mean			1.81	2.11	1.83	1.88	1.99

solution to PP-MINLP2 for that instance (found by any method using the procedure described in section 3.5). Figure 3.10(a) plots the cumulative distribution function of this gap over the 36 test instances for SEC and k-SEC, and the second and third columns of Table 3.4 provide summary statistics. Again, each row refers to an average over 3 instances of the same size. First, we observe that in both cases, the bounds provided are reasonably small, between 1.81% and 2.11% on average, and less than 4% in all instances. Second, we observe that the bounds provided from formulation SEC are slightly better than those obtained from k-SEC, SEC yields average gaps of 1.81% compared to 2.11% from k-SEC. This result is somewhat

counterintuitive, because the feasible region of the k-SEC formulation is a subset of the feasible region of the SEC formulation, so that if both of these formulations were solved to optimality, k-SEC must provide a bound at least as good as SEC. However, these MILP instances are not solved to optimality within the time limit, and so the bounds obtained are the best lower bounds on the respective MILP instances within the time limit. Because k-SEC contains SOS2 variables, these instances are more difficult to solve, and hence have a larger MILP optimality gap after the time limit, which translates to a worse bound on the original problem PP-MINLP2.

For each instance and each MILP formulation we also compute the gap between the feasible solution of PP-MINLP2 found by that formulation to the largest lower bound of PP-MINLP2 obtained for that instance. Figure 3.10(b) plots the cumulative distribution function of this gap over the 36 test instances for SEC and k-SEC, and the last three columns of Table 3.4 provide averages over three instances with the same size. We observe that all formulations provide good solutions for all of the test instances: each formulation provided a solution within 4% of the best known lower bound on all of the instances. However, we observed that formulation PLA-SS is marginally better than SEC-S and k-SEC.

In summary, for these test instances, the SEC-S formulation appears to be preferable to the k-SEC-SS formulation in terms of both the quality of the lower bound obtained within a time limit, and the quality of the feasible solutions obtained. The SEC-S and PLA-SS formulation are complementary: PLA-SS can be used to consistently obtain high quality feasible solutions, whereas SEC-S can be used to

provide a lower bound to evaluate the quality of this solution.

Evaluating approximations of nonlinear functions

In our final experiment, we study the quality of the piecewise-linear approximations and relaxations obtained using the formulations (3.30) and (3.32), respectively. Because these NLPs are nonconvex, we used a multi-start strategy with 100 randomly generated starting solutions to obtain different local optimal solutions. Each NLP is solved to local optimality using Knitro 7.0.0 with an iteration limit of 1000 and of the solutions generated we choose the solution with the best objective value. The starting points were randomly generated between the natural upper and lower bounds of all bounded variables. Starting points for unbounded variables were set to 0 by default by setting the parameter (maxbndrange=0). We conducted numerical experiments on 765 sets of functions which were generated as described in Section 3.5. Each set of functions contained a single production rate function $f(\cdot)$ and between two and six cumulative product production functions $h_p(\cdot)$, leading to a total of 3645 functions. We calculate piecewise-linear approximations and relaxations consisting of three line segments. The average CPU time for solving all a 100 NLP problems (3.32) and (3.30) for a given set of functions was in the order of a few seconds.

We first evaluate the quality of the piecewise-linear approximations produced by formulation (3.30). For a given piecewise-linear approximation $\hat{f}$ of a nonnegative function f, both with domain $[0, M^v]$, we measure the accuracy of the approximation by calculating the *relative function evaluation error* (RFE), evaluated as $\max\{|\hat{f}(v) -$

$f(v)| : v \in [0, M^v]\}/\bar{f}$, where $\bar{f} = \max\{f(v) : v \in [0, M^v]\}$. For each of the 765 sets of functions, we calculated three different sets of piecewise-linear approximations of these functions, and evaluated the RFE value of each approximation of each function in the set. First, we used uniformly spaced breakpoints in the interval $[0, M^v]$, and then optimized the function values $\hat{F}$ or $\hat{H}_p$ to minimize the error with these break points fixed. This strategy can be seen as a naive method for achieving a set of piecewise linear approximations that share the same set of break points. Next, for each set of functions we used the NLP formulation (3.30). Finally, for each *individual* function, we used an adaptation of formulation (3.30) in which only this single function is approximated. This method yields break points that are different for the functions in a set, and hence does not satisfy our goal of using the same set of break points. However, it provides an estimate of the best possible piecewise-linear approximation that could be obtained for each function, if the requirement to share break points were removed. Table 3.5 presents the average, geometric mean, and maximum of the RFE over all these function approximations, taken over all sets and all functions in each set. We observe that using (3.30) yields approximations that are better than using uniform break points (0.96% average RFE compared to 1.21% average RFE). We also observe that the approximations using (3.30) are not much worse than the approximations using separate break points. Thus, we conclude that the loss in accuracy induced by the requirement to share break points across functions within a set is not that significant when using (3.30).

We next evaluate the quality of the piecewise-linear relaxations produced by formulation (3.32). For a given piecewise-linear relaxation $\hat{f}$ of a function f, both

Table 3.5: Average cccuracy of three-segment linear approximations of nonlinear functions on 765 sets of functions (3645 functions total).

		Relative function evaluation error (%)		
$\|\mathcal{P}\|$	# Instances	Separate break points	(3.30)	Uniform break points
2	315	0.78	1.01	1.16
4	225	0.73	0.94	1.21
6	225	0.72	0.91	1.28
Arithmetic mean		0.75	0.96	1.21
Geometric mean		0.74	0.95	1.20
Maximum		1.22	1.53	1.62

Table 3.6: Average accuracy of three-segment linear relaxations of nonlinear functions on 765 sets of functions (3645 functions total).

		Relative area difference (%)	
$\|\mathcal{P}\|$	# Instances	Separate break points	(3.32)
2	315	1.03	1.10
4	225	1.07	1.11
6	225	1.10	1.14
Arithmetic mean		1.06	1.11
Geometric mean		1.05	1.11
Maximum		1.41	1.43

with domain $[0, M^v]$, we measure the accuracy of the relaxation by calculating the *relative area difference* (RAD), evaluated as $|A_f - A_{\hat{f}}|/A_f$, where $A_f = \int_0^{M^v} f(v)dv$ and similarly for $A_{\hat{f}}$. For each set of functions, we obtain two sets of piecewise-linear relaxations. The first set is obtained using (3.32), and the second set is also obtained using (3.32), except that in this case we solve (3.32) separately for each function, allowing the break points to be different for different functions within

a set. Table 3.6 provides the average, geometric mean, and maximum RAD for these two methods over the 765 sets of functions. Once again, we observe that when using (3.32), the requirement that the break points be shared across functions within a set only slightly decreases the quality of the relaxation.

3.6 Concluding remarks

In this chapter, a production planning problem was described with a special structure that the production process creates a mixture of desirable products and undesirable byproducts. A distinguishing feature of this nonconvex MINLP problem is that the fraction of undesirable byproducts increases monotonically as a function of the total mixture production up to that point in time. We present a continuous-time formulation and two discrete-time approximations (MINLP1 and MINLP2) of this problem. A MILP-based approximation (PLA) and two MILP-based relaxations (1-SEC, k-SEC) of this formulation were presented, and modifications to these formulations to improve the linear programming relaxations were derived.

Numerical experiments on small instances demonstrated our proposed formulation MINLP2 yielded solutions up to 30% less costly than those obtained by the natural formulation MINLP1. We found that the strengthening of the MILP formulations had a significant positive impact on the ability of a commercial MILP solver to obtain near-optimal solutions. We demonstrated that, in contrast to the formulation based on MINLP1, using the MILP formulations we were able to obtain good quality feasible solutions, along with lower bounds that verify that these

solutions are near-optimal for this nonconvex MINLP problem. Finally, we found that by using our proposed NLP formulations for obtaining piecewise-linear relaxations and approximations for sets of functions, it is possible to enforce that these piecewise-linear functions share the same set of break points without significantly sacrificing the quality of the approximations.

4 MODELS AND SOLUTION TECHNIQUES FOR PRODUCTION PLANNING PROBLEMS WITH COMPLEX FISCAL TERMS

In strategic planning problems, the modelling of realistic fiscal contracts & business rules like taxes, tariffs and royalties can greatly impact decision making. Optimization models for such strategic planning problems often ignore these business rules and instead use simple linear objective functions like net present value (NPV). In this paper, we develop optimization models, solution techniques, and algorithms for production planning problems in the presence of production sharing contracts (PSC); a type of fiscal contract in which the tax incurred by a contractor is a piecewise-constant function of the internal rate-of-return (IRR). We propose two different solution techniques. The first is a mixed-integer programming (MIP) formulation and the second is a search heuristic based on a novel continuous domain linear programming (LP) formulation. We then propose decomposition methods to compute bounds on the quality of a given solution. Our computational experiments demonstrate the impact of our formulations, solution techniques, and algorithms on a synthetic (but realistic) application.

4.1 Introduction

Sophisticated optimization models are being increasingly used as a tool in large engineering development projects during their design and planning phases. In applications such as hydrocarbon field infrastructure planning [49, 98, 44, 58, 22],

portfolio optimization [77] and production planning [2, 86], decision making during the project planning phase is greatly influenced by business rules like taxes, royalties, tariffs, and other fiscal considerations. It is challenging to formulate and solve optimization problems that can both accurately model these realistic business rules and capture the complex processes that drive the operational decisions. One approach [42, 49, 93] is to retain the complexity in the modeling of operational decisions while simplifying the fiscal models. However, simplified fiscal models may produce substantially worse decisions, in terms of both investment and operations [2, 44, 55]. Since large projects involve multi-billion dollar capital investments, there is a need to develop new models and computational methods that allow decision-makers to incorporate sufficient detail in both operational and fiscal aspects of strategic planning problems. In this work, we consider production sharing contracts (PSC), a fiscal contract commonly used between governments and mineral development industries [28]. PSCs are fiscal legislations wherein the government allocates to a contractor a fixed share of its revenue based on "trigger points," such as cumulative profits, the ratio of profits to costs (called R-factor), or the internal rate of return (IRR). A survey by Deutsche Bank [28] describes a variety of the PSCs implemented in different countries. In this work, we focus on IRR-based PSCs.

IRR-based PSCs. Consider an operational planning problem with decisions spanning a horizon of finite time periods $\mathcal{T} := \{1, 2, \ldots T\}$ that generates a series of cash flows $f_1, f_2, \ldots f_T$. The net present value (NPV) at the end of time period $t \in \mathcal{T}$ can

be computed using

$$h(\hat{q}, f_{[1,t]}) = \sum_{s=1}^{t} \frac{f_s}{(1+\hat{q})^s} \tag{4.1}$$

where $f_{[1,t]} := (f_1, f_2, \ldots, f_t)$ is the vector of cash flows for all time periods $s = 1, 2, \ldots, t$ and $\hat{q} \in (-1, \infty)$ is the discounting rate. At the end of time period $t \in \mathcal{T}$, the internal rate of return (IRR) from the cash flows $f_{[1,t]}$ is a discounting rate q_t that satisfies

$$h(q_t, f_{[1,t]}) = 0. \tag{4.2}$$

(Note that such a q_t need not exist, for example, if all cash flows in $f_{[1,t]}$ have the same sign.) In an IRR-based PSC, the fraction of revenue retained by the contractor during time period $t \in \mathcal{T}$ is a piecewise-constant function of q_{t-1}, evaluated by dividing the range of possible values of q_{t-1} into a set of discrete and disjoint set of K intervals called *profit tranches*. Typically, a PSC allocates a larger share of the revenue to the contractor during time periods t where q_{t-1} is small, since a small value of q_{t-1} corresponds to a stage in the project when the investments have been less profitable. An important aspect of PSCs that we address in this work is the presence of *administrative blocks* (see Figure 4.1). These administrative blocks, also known as "markets" or "ring fences," are entities in the project whose fiscal calculations are grouped together to be completely independent any of the other markets. Section 4.2 provides additional details on IRR-based PSCs.

There are many challenges in developing tractable optimization models for problems involving IRR-based PSCs. One main challenge is to tackle the nonlinearity involved in computing an IRR using (4.2). We show in Section 4.2 that under reason-

able assumptions, the nonlinearity can be eliminated, resulting in a natural Mixed Integer Linear Programming (MIP) formulation for IRR-based PSCs. However, the MIP formulation may be challenging to solve, as it has a weak linear programming relaxation, arising from *variable bounds* (or "big-Ms") necessary to activate and deactive the certain tranche conditions. The complexity of solving the MIP formulation grows significant as the number of markets increases. The complexity of the model grows because its key decision variables are those that determine which of the K possible tranches is associated with each market during each time period. With a single market, the number of possible combinations of such *tranche associations* is small enough to be completely enumerated. But this number grows exponentially with the number of markets, and the combination of the weak LP-relaxation and the combinatorial explosion makes it hard for branch-and-bound algorithms to effectively explore the complete solution space of the MIP formulation.

Related Work. Many authors [98, 44, 58, 22] have explored the use modelling techniques such as MIP and mixed-integer nonlinear programming (MINLP) to incorporate realistic fiscal considerations for strategic planning problems. Of these works, we identify two that we feel are most related to this paper. Van den Heever et al. [98] propose an MINLP model for production planning problems with PSCs. However, they do not consider the modelling of administrative blocks, which is a key challenge for many PSCs. Additionally, the work [98] only considers PSCs where the contractual trigger points are linear functions (e.g. total production) of the operating decision variables, which is not the case for IRR-based PSCs. Gupta and Grossmann [44] propose a series of MIP and MINLP formulations and approximations for

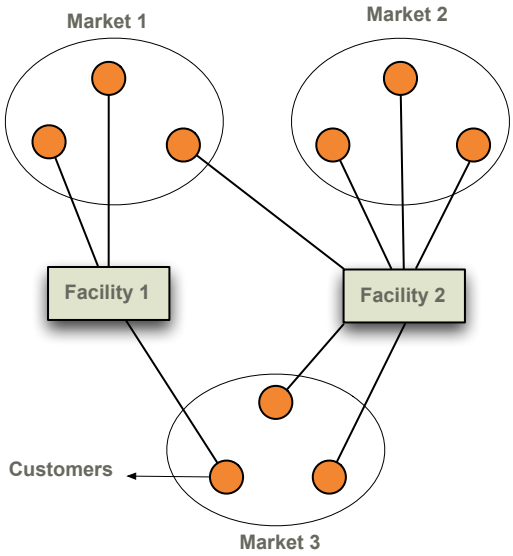


Figure 4.1: A production set with 2 shared facilities and 9 customers distributed between 3 markets.

planning problems involving PSCs. Again, this approach does not easily extend to IRR-based PSCs, in which the trigger points are nonlinear functions of cash flows. The solution techniques proposed in this work are significantly different from any previous work in the literature [98, 44, 58, 22].

Contributions. We highlight three main contributions of this work. First, we propose two formulations for IRR-based PSCs. In Section 4.3, we present a MIP formulation which, under a realistic assumption, eliminates the nonlinearity involved in equation (4.6). This MIP formulation is based on an observation that the profit tranche associated with a market during each time period can be determined by evaluating the function $h(\cdot)$ at a discrete set of K points. Even though the LP-relaxation of this MIP formulation is weak, it can be used to solve problems of

realistic sizes when the number of markets is small.

In order to tackle problems with more markets, we present an alternative formulation in which the decision variables are the tranches associated with each market during each time period. This approach results in an optimization problem with very few decision variables (in a discrete space) but has a nonsmooth objective function. As a second contribution of this work, we extend this formulation for IRR-based PSCs to one in which the decision variables can take continuous values. We then propose a fast and scalable search algorithm that can find feasible solutions of the alternative formulation efficiently. Feasible solutions of this alternative formulation can be used to to determine the profit tranche associated with each market during each time period which can then be used to easily recover a feasible solution of the original MIP formulation. We demonstrate in Section 4.6 that this solution technique is effective in finding feasible solutions for problems with a larger number of markets.

Single-market models for IRR-based PSCs can be solved efficiently using our proposed MIP formulation. We leverage this property in our third contribution; a market-based decomposition scheme. In this scheme, we first reformulate our proposed MIP formulation by making copies, for each market, of all decision variables involving resources that are shared across all markets. Additional constraints are added to the formulation to force the copies to be equal to each other. We then use Lagrangian relaxation to decompose the resulting formulation into independent single-market subproblems. We propose two algorithms to solve the resulting Lagrangian dual problem. The first is a standard subgradient method while the

second is based on an augmented Lagrangian approach known as progressive hedging to this context. Our experiments show that the market-based decomposition approach produces solution bounds (for problems with multiple markets) that are nearly an order of magnitude better than those produced by the MIP formulation. A combination of the search algorithm to quickly find feasible solutions and the market-based decomposition scheme to validate the solution quality constitutes an integrated approach suitable for solving large-scale problems with IRR-based PSCs involving multiple markets.

There has been some recent interest [35] in the design of parameter update rules to obtain solution bounds from progressive hedging. Commonly used parameter update rules [81, 107, 46] focus more on the "consensus" of the copies of the variables in each of the decomposed sub-problems. However, since our goal is to obtain solution bounds, there may not be a cost associated with losing consensus. As a fourth contribution, we propose a simple parameter update rule, for our decomposition algorithm, that is effective in finding good dual solutions for the Lagrangian dual problem.

The remainder of this paper is organized as follows. In Section 4.2, we review notation and provide background on IRR-based PSCs. We present a MIP formulation in Section 4.3. In Section 4.4, we present the alternative solution technique for based on a continuous-domain MIP formulation. We discuss market based decomposition algorithms to obtain solution bounds in Section 4.5. A computational study of the performance of our proposed formulations, solution techniques, and algorithms from Sections 4.3-4.5 is presented in Section 4.6. We close with concluding remarks

in Section 4.7.

4.2 Notation & Background

Consider a multi-period operational planning set X, involving markets $\mathcal{A} = \{1, 2, \ldots A\}$, with a decision horizon spanning a finite set $\mathcal{T} = \{1, 2, \ldots, T\}$ of time periods. During each time period $t = 1, 2, \ldots, T$, operations involving the market $a = 1, 2, \ldots, A$ generate a revenue stream $r_{a,t}$ and require expenses $c_{a,t}$ that result in a series of cash flows $f_{a,t}$. An operational planning problem (involving X) with a linear objective function can be formulated as follows:

$$\max \sum_{a=1}^{A} \sum_{t=1}^{T} \pi_t f_{a,t} \quad \text{subject to } \; f_{a,t} = r_{a,t} - c_{a,t}, \; (\mathbf{c}, \mathbf{r}, \mathbf{x}) \in X,$$

where $\mathbf{x}$ is the vector of decision variables corresponding to resources, about which we say more below. Here, $\pi_t \in \mathbb{R}_+$ is a positive weight on the cash-flows during time period t. If the objective function is the NPV, then π_t has the form $(1 + \hat{\pi})^{-t}$ where $\hat{\pi} \in (-1, \infty)$ is a discounting rate that adjusts for time-value of money. For strategic planning problems with IRR-based PSCs, the objective function can be modelled using a discontinuous and nonconvex function $G(\cdot)$ (known as the PSC function) of the revenues $r_{a,t}$ and expenses $c_{a,t}$. The problem then has the following form:

$$\max G(\mathbf{r}, \mathbf{c}) \quad \text{subject to } \; (\mathbf{c}, \mathbf{r}, \mathbf{x}) \in X. \tag{4.3}$$

In this paper, we propose models, solution techniques and algorithms for the formulation (4.3).

We assume the following structure in the vector $\boldsymbol{x}$. Let $\boldsymbol{x_a}$ denote decision variables corresponding to resources that are local to market a and $\boldsymbol{w} \in W$ denote decision variables associated with resources that are shared across all markets. The full decision vector $\boldsymbol{x}$ has the form $\{\boldsymbol{w}, \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_A}\}$. The variables $\boldsymbol{x_a}$, $a = 1, 2, \ldots, A$ and $\boldsymbol{w}$ may be continuous or discrete and are related by the following *linking* constraints:

$$\sum_{a \in \mathcal{A}} C_a \boldsymbol{x_a} + D\boldsymbol{w} \leqslant d. \tag{4.4}$$

(Here, C_a, $a = 1, 2, \ldots, A$ and D are matrices and d is a vector.) A production model X with such a structure can be written as

$$X = \{\mathbf{c}, \mathbf{r}, \boldsymbol{x}, \boldsymbol{w} : \boldsymbol{w} \in W, (\mathbf{c_a}, \mathbf{r_a}, \boldsymbol{x_a}, \boldsymbol{w}) \in X_a \ \forall a \in \mathcal{A}, \text{and (4.4)}\}, \tag{4.5}$$

where X_a denotes the subset of X that models operations involving only market $a \in \mathcal{A}$. For the production model illustrated in Figure 4.1, the shared and local resources are facilities and customers, respectively. The set W captures constraints that involve only the facilities, whereas X_a captures constraints that involve customers in market $a \in \mathcal{A}$. The linking constraints (4.4) model constraints that involve facilities that are shared between markets. The assumed structure of X is general enough to be useful in a wide range of practical applications.

IRR-Based PSCs

We now describe and analyze the PSC function $G()$ of an IRR-based PSC (4.3). Let $q_{a,t}$ denote the smallest value of the IRR calculated from the cash flows $f_{a,t}$, i.e. the smallest value of $q_{a,t}$ that solves the nonlinear equation

$$h(q_{a,t}, f_{a,[1,t]}) = 0, \quad a = 1, 2, \ldots, A, \quad t = 1, 2, \ldots, T. \tag{4.6}$$

In an IRR-based PSC, the fraction of the revenue $r_{a,t}$ from market a retained by the contractor during time period t is a piecewise-constant function of $q_{a,t}$. The range of possible values of $q_{a,t}$, which is $(-1, \infty)$, is divided into a discrete and disjoint set of profit tranches $\mathcal{K} := \{1, 2 \ldots, K\}$ associated with trigger points λ_k, $k = 1, 2, \ldots, K$. (Typically, K is set to four or five [28].) A market a is associated with profit tranche $k_{a,t} \in \mathcal{K}$ during time period t if $q_{a,t-1} \in (\lambda_k, \lambda_{k+1}]$. The market a is in the first profit tranche for $t = 1$ or if $\sum_{s=1}^{t} f_s < 0$ or if all cash flows in the sequence $f_{a,[1,t]}$ are negative. (Recall that $q_{a,t}$ does not exist if all cash flows in $f_{a,[1,t]}$ are of the same sign.) If all cash flows in $f_{a,[1,t]}$ are positive, then market a is associated with tranche K. The profit tranche $k_{a,t}$ determines the fraction $\mu_{k_{a,t}}$ of the revenue, from market a, retained by the contractor during time period t. The parameters μ_l, $l = 1, 2, \ldots, K$ satisfy the condition

$$\mu_1 > \mu_2 \ldots > \mu_K.$$

Table 4.1 provides an illustration of trigger points λ_k and profit fractions μ_k used in a typical contract. Table 4.2 shows a sample calculation of a planning problem

Table 4.1: Sample parameters used in an IRR-based PSC.

Tranche k	IRR Range (%) $(\lambda_k, \lambda_{k+1}]$	Profit fraction μ_k
1	(-100, 15]	70%
2	(15, 20]	60%
3	(20, 30]	25%
4	(30, ∞)	15%

with 5 time periods involving an IRR-based PSC with 4 profit tranches and 1 market. A procedure to evaluate the PSC function G is presented in Algorithm 1. Here, the PSC function has the form $\sum_{a\in\mathcal{A}} \sum_{t\in\mathcal{T}} \pi_t f_{a,t}$ where $f_{a,t}$ are the cash flows wherein revenue has been adjusted to account for the investor's profit fractions μ_k. The adjusted cash flows can be calculated from the revenues $r_{a,t}$ and expenses $c_{a,t}$ using

$$f_{a,t} = \mu_{k_{a,t}} r_{a,t} - c_{a,t} \quad a = 1, 2, \ldots, A, \quad t = 1, 2, \ldots, T.$$

The dependence of the cash flow $f_{a,t}$ on the index of the tranche $k_{a,t}$ introduces nonlinearities into the model.

Table 4.2: Example of an IRR-based PSC with 1 market, 4 tranches and 5 time periods.

Time (t)	Investment ($c_{1,t}$)	Revenue ($r_{1,t}$)	IRR ($q_{1,t}$)	Tranche ($k_{1,t}$)	Contractor Share ($\mu_{k_{1,t}}$)	Cash Flow ($\mu_{k_{1,t}} r_{1,t} - c_{1,t}$)
0	4000	0	n.a	1	70%	-4000
1	0	2300	-59.8%	1	70%	1610
2	0	6000	24.6%	1	70%	4200
3	0	5000	46.7%	2	60%	3000
4	0	4000	49.0%	4	15%	600

Algorithm 1: Procedure for evaluating $G(\cdot)$.

Data: Function arguments $r_{a,[1,t]}, c_{a,[1,t]}$ and parameters

$\lambda_k, \mu_k \; \forall k \in \mathcal{K}, \pi_t \; \forall t \in \mathcal{T}$

Result: Production sharing function value $G(r_{a,[1,t]}, c_{a,[1,t]})$

$g \leftarrow 0$

for $a = 1, 2, \ldots, A$ **do**

  $f_{a,1} \leftarrow \mu_1 r_{a,1} - c_{a,1}$

  $g \leftarrow g + \pi_1 f_{a,1}$

  $t \leftarrow 1$

  while $t \leqslant T$ **do**

    Solve for $q_{a,t}$ in $\sum_{s=1}^{t} \frac{f_{a,s}}{(1+q_{a,t})^s} = 0$

    if $q_{a,t}$ *exists and* $q_{a,t} \in (\lambda_{k-1}, \lambda_k]$ *for some* $k > 1$ **then**

      $f_{a,t+1} \leftarrow \mu_k r_{a,t+1} - c_{a,t+1}$

    else if $f_{a,t-1} \geqslant 0$ **then**

      $f_{a,t+1} \leftarrow \mu_K r_{a,t+1} - c_{a,t+1}$

    else

      $f_{a,t+1} \leftarrow \mu_1 r_{a,t+1} - c_{a,t+1}$

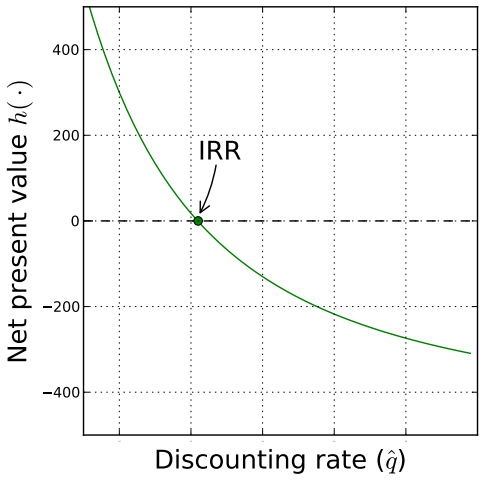
    $g \leftarrow g + \pi_t f_{a,t+1}$

    $t \leftarrow t + 1$

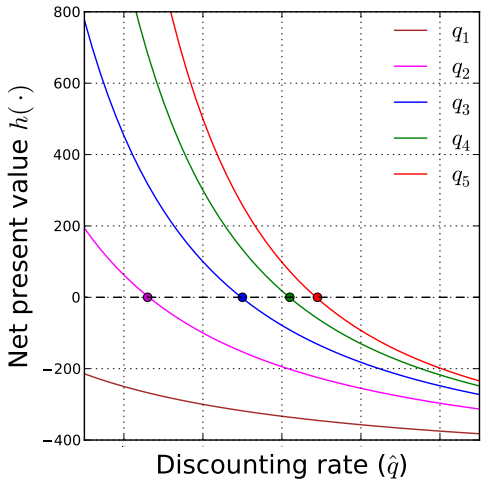
return g

Figure 4.2: Illustration of the assumptions made while modelling IRR-based PSCs

(a) Assumption 1: The NPV function $h(\cdot)$ has at most one root.



(b) Property 2: The sequence of tranches associated with a single market are nondecreasing



Assumptions & Restrictions

We specify here some assumptions about the sequence of cash flows $f_{a,[1,t]}$ that commonly hold in practice and that will be assumed in the remainder of the paper. To avoid an overload of notation, we drop the market subscript a from all quantities in this section.

Assumption 4.1. *Given a series of cash flows* $f_{[1,t]}$*, the equation* $h(q_t, f_{[1,t]}) = 0$ *has at most one unique root. If it has one unique root, then* $f_1 < 0$*.*

Assumption 4.1 is illustrated in Figure 4.2(a). When equation (4.6) has no root, then the cash flows $f_{[1,t]}$ are either all negative or all positive, putting the system in the lowest or the highest profit tranche during time periods $2, 3, \ldots, t$. When equation (4.6) has one root, Assumption 4.1 ensures that there exists a root for

$h(\hat{q}, f_{[1,t]}) = 0$ in the range $[\hat{q}_l, \hat{q}_u]$ if and only if $h(\hat{q}_l, f_{[1,t]}) \geqslant 0$ and $h(\hat{q}_u, f_{[1,t]}) \leqslant 0$. Using Assumption 4.1, we can formulate IRR-based PSCs as a MIP using the fact that a market lies in tranche $k \in \{2, 3, \ldots, K\}$ if

$$h(\lambda_{k-1}, f_{[1,t]}) \geqslant 0 \quad \text{and} \quad h(\lambda_k, f_{[1,t]}) \leqslant 0. \tag{4.7}$$

A sufficient condition for Assumption 4.1 to hold is the single sign-change test [75] with an additional condition that $f_1 < 0$. The single sign-change (SSC) test limits the number of sign changes in the series of cash-flows $f_{[1,t]}$ to exactly one, which in turn limits the number of IRRs to exactly one, using Descartes' rule of signs. The additional condition that $f_1 < 0$ ensures that the sign of the cash flows $f_{[1,t]}$ changes from negative to positive, as required in (4.7). Assumption 4.1 also holds under the Norstrøm condition [71], with the additional requirement that $f_1 < 0$. The Norstrøm condition states that an IRR due to cash flows $f_{[1,t]}$ is unique if the sequence of cumulative cash flows $\{\sum_{s=1}^{t} f_s\}$ has exactly one sign change. Both these sufficient conditions occur quite commonly in practical applications, where large capital investments are made primarily at the early stages of the project.

Our model will enforce that the sequence of tranches associated with each market is non-descreasing in the time periods (see Figure 4.2(b)).

Property 4.2. *If* $\tau_k = \inf\{t \in \mathcal{T} : h(\lambda_k, f_{[1,t]}) > 0\}$ *is the earliest period that the sequence of cash flows* $\{f_{[1,T]}\}$ *induce entering tranche* $k \in \mathcal{K}$*, then solutions of our model will satisfy* $\tau_2 \geqslant \tau_3 \geqslant \cdots \geqslant \tau_K$.

The following proposition shows that enforcing this monotoncity property is

not an unduly restrictive assumption of our model.

Proposition 4.1. *Define the tranche-entry periods* $\tau_k = \inf\{t \in \mathcal{T} : h(\lambda_k, f_{[1,t]}) > 0\}$. *If the SSC holds and* $f_1 < 0$, *then* $\tau_2 \geqslant \tau_3 \geqslant \cdots \geqslant \tau_K$.

Proof. By the SSC assumption, there is at most one IRR. Furthermore, since $f_1 < 0$, if an IRR exists in period t, then an IRR exists for all $s \geqslant t$. Let p be the first period in which an IRR exists, consider any period $t \geqslant p$, and let q_t denote the unique IRR calculated at the end of time period t. We will show that $q_{t+1} \geqslant q_t$, and the result will follow from the fact that the endpoints of the tranche ranges λ_k are increasing in k. With the SSC condition, the form of the NPV function (4.1) implies that if $h(\iota, q_{t+1}) > 0$, then $\iota < q_{t+1}$. Substituting q_t into the NPV function for $f_{[1,t+1]}$, we see that

$$h(q_t, f_{[1,t+1]}) = h(q_t, f_{[1,t]}) + \frac{f_{t+1}}{(1+q_t)^{t+1}} > 0,$$

since $h(q_t, f_{[1,t]}) = 0$ by definition, and the second term is positive. Thus, $h(q_t, f_{[1,t+1]}) > 0$, so $q_t \leqslant q_{t+1}$, which completes the proof. □

Property 4.2 reduces the number of possible combinations of the K tranches associated with a single market during each of the T time periods to $O(T^K)$ from $O(K^T)$. This reduction can have a significant impact on the computational effort required to solve the problem, because in most practical applications, the number of tranche levels K varies between 4 and 6 while the number of time periods T can be significantly greater, for example, $T = 20$ or $T = 25$.

Table 4.3: State of binary variables for a system with three tranches and four time periods

Time Period	Tranche	$b_{k,t}$	$b_{k,t} - b_{k+1,t}$	$\text{sgn}(h_{k,t})$
0	1	1 0 0 0	1 0 0 0	- - - -
1	1	1 0 0 0	1 0 0 0	- - - -
2	1	1 0 0 0	1 0 0 0	- - - +
3	2	1 1 0 0	0 1 0 0	- + + +
4	4	1 1 1 1	0 0 0 1	- + + +

4.3 MIP Formulations

In this section, we build a MIP formulation for the production sharing contract problem (4.3). The formulation relies on a disaggregation of the revenues into different tranches. The first step is to determine the tranches for the cash flow for each time period. To that end, let $h_{a,k,t}$ denote the NPV for market a during time period t calculated using the cash flows $f_{a,[1,t]}$ discounted at an interest rate λ_k

$$h_{a,k,t} = \sum_{s=1}^{t} \frac{f_{a,s}}{(1+\lambda_k)^s}, \quad a = 1,2,\ldots,A, \;\; k = 1,2,\ldots,K, \;\; t = 1,2,\ldots,T. \tag{4.8}$$

Let $b_{a,k,t}$ be a binary variable that is set to one if cash flows for market $a \in \mathcal{A}$ have reached the minimum IRR level associated with a tranche k during time period $t \in \mathcal{T}$. Given minimum and maximum attainable values of the net present value function $h(f_{a,[1,t]}, \lambda_k)$ — $m^h_{k,t}$ and $M^h_{k,t}$, respectively — and using the insight

captured in equation (4.7), we arrive at the inequalities

$$m^h_{a,k,t-1}(1 - b_{a,k,t}) \leqslant h_{a,k,t-1} \leqslant M^h_{a,k,t-1} b_{a,k,t} \quad a \in \mathcal{A},\ \ k = 1, 2, \ldots, K,\ \ t = 2, \ldots, T, \tag{4.9}$$

$$b_{a,k-1,t} \geqslant b_{a,k,t} \qquad a \in \mathcal{A}\ \ k = 2, 3, \ldots, K,\ \ t \in \mathcal{T}, \tag{4.10}$$

$$b_{a,k,1} = 0 \qquad a \in \mathcal{A},\ \ k = 2, 3, \ldots, K, \tag{4.11}$$

$$b_{a,1,t} = 1 \qquad a \in \mathcal{A},\ \ t \in \mathcal{T}. \tag{4.12}$$

The inequalities (4.9) enforce that the variable $b_{akt} = 1$ if and only if $h_{a,k,t-1} \geqslant 0$, which implements the logic described in (4.7), so we can be sure that we have reached tranche k by period t. Table 4.3 illustrates the state of the binary variables for the sample calculation of the IRR-based PSC in Table 4.2 and their relation to the sign of the NPV variables $h_{a,k,t}$.

Next, we enforce the condition from Property 4.2 that the sequence of tranches associated with each market during a time period is nondecreasing:

$$b_{a,k,t-1} \geqslant b_{a,k,t} \quad a \in \mathcal{A},\ \ k = 2, 3, \ldots, K,\ \ t = 2, 3, \ldots, T. \tag{4.13}$$

A key idea in the MIP model is to disaggregate the revenues over the tranches. Then, using the binary variables b_{akt}, the proper contractor profit fraction μ_k can be applied for the cash flow calculation. Mathematically, this leads to the following

constraints:

$$f_{a,t} = \sum_{k \in \mathcal{K}} \mu_k p_{a,k,t} - c_{a,t} \qquad a \in \mathcal{A},\ \ t \in \mathcal{T}, \tag{4.14}$$

$$r_{a,t} = \sum_{k \in \mathcal{K}} p_{a,k,t} \qquad a \in \mathcal{A},\ \ t \in \mathcal{T} \tag{4.15}$$

$$p_{a,k,t} \leqslant M^p_{k,t}(b_{a,k,t} - b_{a,k+1,t}) \quad a \in \mathcal{A},\ \ k = 1, 2, \ldots, K-1,\ \ t \in \mathcal{T} \tag{4.16}$$

$$p_{a,K,t} \leqslant M^p_{k,t} b_{a,K,t} \qquad a \in \mathcal{A},\ \ t \in \mathcal{T}. \tag{4.17}$$

Here, $p_{a,k,t}$ is the disaggregation of the revenue $r_{a,t}$ over the set of profit tranches $\mathcal{K}$ and $M^p_{a,k,t}$ is the maximum revenue attainable from market a while being in tranche k during time period t. The equations enforce that $p_{a,k,t} = r_{a,t}$ if the market $a \in \mathcal{A}$ is in tranche $k \in \mathcal{K}$ during time period $t \in \mathcal{T}$ and $p_{a,k,t} = 0$ otherwise. Put together, the constraints (4.8)-(4.17) formulate an IRR-based PSC for each market $a \in \mathcal{A}$, and we denote this formulation with the set

$$Y_a := \left\{ \mathbf{b_a} \in \{0,1\}^{K \times T},\ \mathbf{c_a} \in \mathbb{R}^{K \times T},\ f_a \in \mathbb{R}^{K \times T}, \mathbf{h_a} \in \mathbb{R}^T,\ \mathbf{p_a} \in \mathbb{R}^T,\ \mathbf{r_a} \in \mathbb{R}^T c \ :\ (4.8)\text{-}(4.17) \right\}.$$

Table 4.4: Problem and solution statistics for a single instance of the MIP formulation in (4.18) for a sample application problem solved using Gurobi 5.0.1 with 2 threads for 7200 seconds.

Markets	Variables (Binary)	Constraints	Time (s)	Optimality Gap (%)	Root LP Gap (%)	Nodes
1	15132 (1457)	22101	680.39	< 0.1	184.3	1323
2	15263 (1514)	22401	1117.4	< 0.1	186.9	2934
3	15389 (1568)	22701	3670.3	< 0.1	189.4	12136
4	15515 (1622)	23001	7200.0	8.24	190.4	28208
5	15646 (1679)	23301	7200.0	34.4	186.3	22381
6	15762 (1727)	23601	7200.0	66.2	194.9	10515

A MIP formulation for (4.3) can now be written as:

$$\begin{aligned} \max \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \pi_t f_{a,t} & \\ \text{subject to } (\mathbf{b}_a, \mathbf{c}_a, \mathbf{f}_a, \mathbf{h}_a, \mathbf{p}_a, \mathbf{r}_a) \in Y_a \quad & a \in \mathcal{A} \\ (\mathbf{c}_a, \mathbf{r}_a, \mathbf{x}_a, \boldsymbol{w}) \in X_a \quad & a \in \mathcal{A} \\ \boldsymbol{w} \in W & \\ \sum_{a \in \mathcal{A}} C_a \mathbf{x}_a + D\boldsymbol{w} \leqslant d. & \end{aligned} \tag{4.18}$$

We illustrate the shortcomings of formulation (4.18) by solving an instance of a sample-application problem, with 1-6 markets using Gurobi with a two-hour time limit. The instances were constructed to contain roughly the same number of decision variables and constraints, but to have a different number of markets. Sec-

tion 4.6 provides additional details about the sample application problem, datasets and system setup used to obtain these results. Table 4.4 provides problem statistics, wall clock time (in seconds), terminal optimality gaps, root LP gaps and the number of nodes explored. The root LP gap reports the quantity $(LP - BFS)/BFS$ as a percentage, where LP is the value of the root LP-relaxation of (4.18) and BFS is objective value of the best found feasible solution. We observed that the root LP gap for these instances was 180%-190% above the best feasible solution. We attribute this weak LP-relaxation to the difficulty in obtaining natural variable upper and lower bounds $m^h_{a,k,t}$ and $M^h_{a,k,t}$ on the NPV variables $h_{a,k,t}$ required for the constraints in (4.9).

An interesting observation from this experiment is that even with a weak LP relaxation, instances of (4.18) can be solved to optimality in a reasonable amount of time when the number of markets is not more than 3. However, the terminal optimality gaps increase significantly as the number of markets increases beyond 3, even though the problem size increases only marginally. Our intuition behind this computational behavior is as follows. The key decision variables in (4.18) are the binary variables $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_A}\}$. For example, if the values of these variables are fixed to integer variables, the resulting "operational" MIP can be solved in a few minutes. By property 4.2, we know that the number of feasible combinations for the set of variables $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_A}\}$ for a problem with A markets is $O(A^{T^K})$. If A is small-enough, then the LP-based branch and bound algorithm can within the time-limit branch enough to essentially enumerate the feasible combinations for the values of the variables $\{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_A}\}$. If A gets larger than 3, the solutions cannot

be enumerated, and the extremely weak LP-relaxation makes it impossible for a branch-and-bound algorithm to effectively reduce the solution space. To summarize: Our results demonstrate that the MIP formulation (4.18) is not sufficient to solve production planning problems with IRR-based PSCs, especially when the number of markets is more than 3. In the following sections, we propose alternative solution techniques and algorithms to find feasible solutions as well as bounds for (4.3).

4.4 Heuristics

In this section, we present a technique to obtain feasible solutions for (4.3). We first reformulate (4.3) as a search problem where the decision variables are *tranche configurations* $\tau_{a,k} \in \{1, 2, \ldots, T\}\ \forall a \in \mathcal{A},\ \forall k = 1, 2, \ldots, K$, denoting the time period at which the market $a \in \mathcal{A}$ changes from tranche level $k-1$ to k. This formulation has $A \times K$ (integer-valued) decision variables, but the objective function can be computed only by solving a MIP involving the production model X, with bounds on the signs of the NPV values implied by the choice of trache configuation. To efficiently search for good tranche configurations, we extend the definition to account for *fractional tranche configurations* that need not be integral. We then propose a continuous domain search algorithm to find good fractional/integral tranche configurations and therefore good feasible solutions for (4.3).

The Fixed-Tranche Problem

Let $\tau_{a,k} = \inf\{t \in \{1,2,\ldots,T\} \ : \ h(\lambda_k, f_{[1,]}at) > 0\}$ denote the time period $t \in \mathcal{T}$ during which the market $a \in \mathcal{A}$ moves from tranche $k-1$ to k where $k = 2,3,\ldots,K$. From Property 4.2, we know that

$$T := \{\boldsymbol{\tau} \in \{1,2,\ldots,T\}^{A\times K} : 1 = \tau_{a,1} \leqslant \tau_{a,2} \leqslant \cdots \leqslant \tau_{a,K}, \ \ a \in \mathcal{A}\}$$

is the set of feasible tranche configurations for (4.3). We define the *fixed-tranche problem* as a MIP formulation in which the tranche configuration $\boldsymbol{\tau}$ is known a priori. Given $\tau_{a,k}$ $\forall a \in \mathcal{A}, k \in \mathcal{K}$, we know the tranche associated with each market a during each time period t, which imposes the following restrictions on the signs on the NPV variables $h_{a,k,t}$:

$$h_{a,k,t} \leqslant 0, \quad a \in \mathcal{A}, \ k = 2,3,\ldots,K, \ t \in \mathcal{T} : t < \tau_{a,k}, \tag{4.19a}$$

$$h_{a,k,t} \geqslant 0, \quad a \in \mathcal{A}, \ k = 2,3,\ldots,K, \ t \in \mathcal{T} : t \geqslant \tau_{a,k}, \tag{4.19b}$$

where $h_{a,k,t}$ is defined using (4.8). Note that constraints (4.19) are equivalent to the constraints (4.9) in the MIP formulation (4.18) when the values of all binary decision variables $\mathbf{b}$ are fixed. The tranche configuration also fixes the fraction $\mu_{a,t}$

retained by the contractor during time t for market a as follows:

$$\mu_{a,t} = \begin{cases} \mu_1, & t < \tau_{a,2} \\ \mu_k, & \tau_{a,k} \leqslant t < \tau_{a,k+1},\ k \in \{2, 3, \dots K-1\} \\ \mu_K, & \tau_{a,K} \leqslant t \end{cases} \tag{4.20}$$

The cash flow $f_{a,t}$ is calculated using these investor revenue fractions:

$$f_{a,t} = \mu_{a,t} r_{a,t} - c_{a,t} \quad \forall a \in \mathcal{A},\ t \in \mathcal{T}. \tag{4.21}$$

Given $\boldsymbol{\tau} \in T$, we can recover a feasible solution for (4.18) by evaluating the following *fixed-tranche* function $g : T \to \mathbb{R}$:

$$\begin{aligned} g(\boldsymbol{\tau}) := \max & \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \pi_t f_{a,t} \\ \text{subject to} \quad & (\mathbf{c}, \mathbf{f}, \mathbf{h}, \mathbf{r}) \in Q(\boldsymbol{\tau}) \\ & (\boldsymbol{c}_a, \boldsymbol{r}_a, \boldsymbol{x}_a, \boldsymbol{w}) \in X_a \quad \forall a \in \mathcal{A} \\ & \boldsymbol{w} \in W \\ & \sum_{a \in \mathcal{A}} C_a \boldsymbol{x}_a + D\boldsymbol{w} \leqslant d \end{aligned} \tag{4.22}$$

where the set $Q(\boldsymbol{\tau})$ is defined by

$$Q(\boldsymbol{\tau}) := \left\{ \mathbf{c} \in \mathbb{R}^{A \times T},\ \mathbf{f} \in \mathbb{R}^{A \times T},\ \mathbf{h} \in \mathbb{R}^{A \times K \times T},\ \mathbf{r} \in \mathbb{R}^{A \times T} : (4.8), (4.19)\text{-}(4.21) \right\}.$$

The optimal tranche configuration $\boldsymbol{\tau} \in \mathrm{T}$ can be computed by solving the nonconvex optimization problem

$$\max g(\boldsymbol{\tau}) \quad \text{subject to } \boldsymbol{\tau} \in \mathrm{T} \tag{4.23}$$

where the decision variables $\boldsymbol{\tau}$ are constrained to lie in a discrete space. By construction of (4.22), it is easy to see that the cost of an optimal solution of (4.23) is exactly equal to the optimal objective of (4.18). Figure 4.3(a) illustrates the function values of $g(\cdot)$ for a problem with $K = 3$ and $A = 1$. The x, y, and z-axes are values for $\tau_{1,2}$, $\tau_{1,3}$, and $g(\boldsymbol{\tau})$, respectively. As the figure illustrates, the search for the optimal tranche configuration from the set T is challenging because the decision variables lie in a discrete space, where direct search methods such as pattern search are typically employed. The results of initial computational experiments using these methods were disappointing. This fact motivates us to extend the formulation in (4.23) to allow *fractional tranche configurations* that can take values in the range $[1, T]$, thus allowing us to use continuous-domain search algorithms.

Continuous-Domain Extension of $Q(\boldsymbol{\tau})$

In this section, we extend the definition $Q(\boldsymbol{\tau})$ to allow for fractional tranche configurations. For each market $a \in \mathcal{A}$, let $\hat{\tau}_{a,k} \in [1, T]$ denote a real-valued quantity indicating the time at which the market $a \in \mathcal{A}$ moves from tranche configurations $k-1$ to k, for $k = 1, 2, \ldots, K$. From Property 4.2, the set of feasible fractional tranche

configurations is:

$$\hat{T} := \{\hat{\boldsymbol{\tau}} \in [1, T]^{A \times K} : 1 = \hat{\tau}_{a,1} \leqslant \hat{\tau}_{a,2} \leqslant \cdots \leqslant \hat{\tau}_{a,K} \; \forall a \in \mathcal{A}\}.$$

In extending $Q(\boldsymbol{\tau})$, we first extend (4.9) using the definition of continuous-time NPV, where compounding is applied continuously, instead of at the end of a time period. Given $\delta \in [0, 1]$, we define the NPV at time $t + \delta$ as follows:

$$\hat{h}(f_{a,[1,t+1]}, \hat{q}, t + \delta) := \sum_{s=1}^{t} \frac{f_{a,s}}{(1 + \hat{q})^s} + (\delta f_{a,t+1}) e^{-\hat{q}_c(\delta + t)} \quad a \in \mathcal{A}. \tag{4.24}$$

Here, the discounting rate $\hat{q}_c = \log(1 + \hat{q})$ accounts for continuous compounding. In (4.24), the NPV at time $t + \delta$ is calculated using discrete time discounting at a rate $\hat{q}$ for the first t time periods, followed by continuous compounding at a rate q_c in the time interval $[t, t + \delta]$, during which we assume that market $a \in \mathcal{A}$ has a cash flow of $\delta f_{a,t+1}$. We can simplify the expression in (4.24) as follows:

$$\hat{h}(f_{a,[1,t+1]}, q, t + \delta) = \sum_{s=1}^{t} \frac{f_{a,s}}{(1 + \hat{q})^s} + \frac{\delta f_{a,t+1}}{(1 + \hat{q})^{t+\delta}}, \quad a \in \mathcal{A}. \tag{4.25}$$

Note that (4.25) is consistent with (4.9) when $\delta \in \{0, 1\}$. Using (4.25), we restrict the signs on the NPV variables $h_{a,k,t}$ during all time periods $t \in \mathcal{T} \backslash \{\lfloor \hat{\tau}_{a,k} \rfloor, \lceil \hat{\tau}_{a,k} \rceil\}$ as follows:

$$h_{a,k,t} \leqslant 0 \quad \forall a \in \mathcal{A},\ k \in \{2 \ldots K\},\ t \in \mathcal{T} : t \leqslant \lfloor \hat{\tau}_{a,k} \rfloor - 1 \tag{4.26a}$$

$$h_{a,k,t} \geqslant 0 \quad \forall a \in \mathcal{A},\ k \in \{2 \ldots K\},\ t \in \mathcal{T} : t \geqslant \lceil \hat{\tau}_{a,k} \rceil. \tag{4.26b}$$

Here, $h_{a,k,t}$ is defined using (4.8). During the time periods $\lfloor \hat{\tau}_{a,k} \rfloor$ and $\lceil \hat{\tau}_{a,k} \rceil$, we use the definition of NPV in (4.25) to restrict $\hat{h}(f_{a,[1,\tau_{a,k}]}, \lambda_k, \hat{\tau}_{a,k}) \leqslant 0$ and $\hat{h}(f_{a,[1,\tau_{a,k}-1]}, \lambda_k, \hat{\tau}_{a,k} - 1) \geqslant 0$. We can express this restriction as follows:

$$h_{a,k,\lfloor \hat{\tau}_{a,k} \rfloor - 1} + \frac{(\hat{\tau}_{a,k} - \lfloor \hat{\tau}_{a,k} \rfloor) f_{a,[1,\lfloor \hat{\tau}_{a,k} \rfloor]}}{(1+\lambda_k)^{\hat{\tau}_{a,k}-1}} \leqslant 0 \quad \forall a \in \mathcal{A}, t \in \mathcal{T} \tag{4.26c}$$

$$h_{a,k,\lfloor \hat{\tau}_{a,k} \rfloor} + \frac{(\hat{\tau}_{a,k} - \lfloor \hat{\tau}_{a,k} \rfloor) f_{a,[1,\lfloor \hat{\tau}_{a,k} \rfloor + 1]}}{(1+\lambda_k)^{\hat{\tau}_{a,k}}} \geqslant 0 \quad \forall a \in \mathcal{A}, t \in \mathcal{T}. \tag{4.26d}$$

The use of constraints (4.26) in the is based on (4.19) in formulation (4.22). This may seem may seem less natural that a formulation in which we can simply restrict $\hat{h}(f_{a,[1,\hat{\tau}_{a,k}]}, \lambda_k, \hat{\tau}_{a,k}) = 0$, but is consistent with (4.22) when the tranche configurations are integral, which is the main purpose of the continuous domain extension of $Q(\boldsymbol{\tau})$.

Next, we use the fractional tranche configuration $\hat{\boldsymbol{\tau}}$ to determine, in advance, the contractor share during time periods $t \in \mathcal{T}$ for each market $a \in \mathcal{A}$ using

$$\hat{\mu}_{a,t}(\bar{\tau}) = \begin{cases} \mu_1, & \bar{\tau} \leqslant \lfloor \hat{\tau}_{a,2} \rfloor \\ \mu_k, & \lceil \hat{\tau}_{a,k} \rceil \leqslant \bar{\tau} < \lfloor \hat{\tau}_{a,k+1} \rfloor,\ k \in \{2, \dots K-1\} \\ \mu_K, & \lceil \hat{\tau}_{a,K} \rceil \leqslant \bar{\tau} \\ \bar{\mu}(\hat{\tau}_{a,k}), & \bar{\tau} = \lfloor \hat{\tau}_{a,k} \rfloor,\ k \in \{2 \dots K\} \end{cases} \tag{4.27}$$

Here, $\hat{\mu}_{a,t}(\bar{\tau})$ is the contractor share at a time $\bar{\tau} \in [1, T]$ and

$$\bar{\mu}_{a,t}(\hat{\tau}) := \sum_{\sigma^L_{a,t}(\hat{\tau})}^{\sigma^R_{a,t}(\hat{\tau})} \mu_k \Big(\min\{\hat{\tau}_{a,k+1}, t+1\} - \max\{\hat{\tau}_{a,k}, t\} \Big)$$

where $\sigma^L_{a,t}(\hat{\tau}) := \max\{k \in K : \hat{\tau}_{a,k} \leqslant t\}$ and $\sigma^H_{a,t}(\hat{\tau}) := \max\{k \in K : \lfloor \hat{\tau}_{a,k} \rfloor \leqslant t\}$. The first three terms of $\hat{\mu}(\cdot)$ in (4.27) determine the share of the revenue retained by the contractor during the time periods $t \in \mathcal{T}$ when market $a \in \mathcal{A}$ lies completely in a tranche $k \in \mathcal{K}$. The only difference between $\hat{\mu}(\cdot)$ and $\mu(\cdot)$ in (4.20) occurs during the time periods $\lceil \hat{\tau}_{a,k} \rceil \in \mathcal{T}$ where the market $a \in \mathcal{A}$ is partially associated with other tranches in $\mathcal{K}$. During these time periods, we calculate the contractor share as a linear combination of the contractor share from profit tranches $\{k_2 \in \mathcal{K} : \lfloor \hat{\tau}_{a,k} \rfloor = \lfloor \hat{\tau}_{a,k_2} \rfloor\}$. The discounting rate $\hat{\mu}(\cdot)$ is based on the intuition that market $a \in \mathcal{A}$ lies in tranche $k_2 - 1 \in \{2 \dots K\}$ during the interval $[\min(\hat{\tau}_{a,k_2}, \lceil \hat{\tau}_{a,k_2} \rceil), \max(\hat{\tau}_{a,k_2-1}, \lfloor \hat{\tau}_{a,k} \rfloor)]$ during which we assume that the contractor retains μ_{k_2-1} of the revenue. Again, we highlight that the calculation of the contractor share in $\hat{\mu}(\cdot)$ is consistent with $\mu(\cdot)$ used in formulation (4.22) when $\hat{\tau}_{a,k}$ is integral.

We can use (4.27) to calculate the cash flow $f_{a,t}$ as follows:

$$f_{a,t} = \hat{\mu}_{a,t}(t) r_{a,t} - c_{a,t} \quad \forall a \in \mathcal{A},\ t \in \mathcal{T} \tag{4.28}$$

We now define the continuous domain extension of the formulation (4.22) for $\hat{\boldsymbol{\tau}} \in \hat{T}$

as

$$\begin{aligned} \hat{g}(\hat{\boldsymbol{\tau}}) := \max \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \pi_t f_{a,t} \qquad & (4.29) \\ \text{subject to} \quad (\mathbf{c}, \mathbf{f}, \mathbf{h}, \mathbf{r}) &\in \hat{Q}(\boldsymbol{\tau}) \\ (\mathbf{c}_a, \mathbf{r}_a, \boldsymbol{x}_a, \boldsymbol{w}) &\in X_a \quad a \in \mathcal{A} \\ \boldsymbol{w} &\in W \\ \sum_{a \in \mathcal{A}} C_a \boldsymbol{x}_a + D\boldsymbol{w} &\leqslant d, \end{aligned}$$

where the set is given by

$$\hat{Q}(\hat{\boldsymbol{\tau}}) := \left\{ \mathbf{c} \in \mathbb{R}^{A \times T}, \mathbf{f} \in \mathbb{R}^{A \times T}, \ \mathbf{h} \in \mathbb{R}^{A \times K \times T}, \ \mathbf{r} \in \mathbb{R}^{A \times T} : (4.8), (4.26)\text{-}(4.28) \right\}.$$

An optimal *fractional tranche configuration* can be computed by solving:

$$\max \hat{g}(\hat{\boldsymbol{\tau}}) \quad \text{subject to} \ \ \hat{\boldsymbol{\tau}} \in \hat{T} \qquad (4.30)$$

Based on the construction of $\hat{Q}(\boldsymbol{\tau})$, it is easy to verify that the following holds.

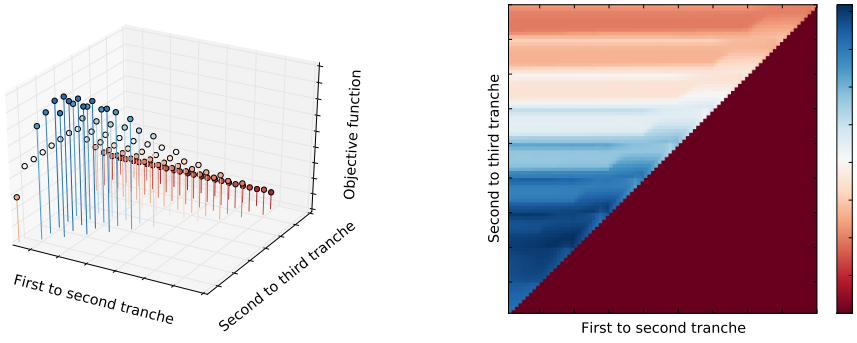
Lemma 4.3. $\hat{Q}(\boldsymbol{\tau}) = Q(\boldsymbol{\tau})$ *and* $\hat{g}(\boldsymbol{\tau}) = g(\boldsymbol{\tau})$ *for* $\boldsymbol{\tau} \in \hat{T}$.

Figure 4.3 illustrates the difference between the functions $g(\cdot)$ and $\hat{g}(\cdot)$ for a problem with a $A = 1$, and $K = 3$. The x and y axes denote $\hat{\tau}_{1,2}$ and $\hat{\tau}_{1,3}$ respectively. The function values of $\hat{g}(\cdot)$ are illustrated using a heat-map where shades of blue indicate a higher function value than shades of red. The lower-right triangular region indicates infeasible solutions because $\hat{\tau}_{a,2} \geqslant \hat{\tau}_{a,3}$. We produced Figure 4.3(b)

by computing $\hat{g}(\cdot)$ on an even grid with a spacing of 0.01 and Gaussian interpolation on the function values.

Figure 4.3: An illustration of the differences between formulations (4.22) and (4.29).

(a) The *fixed-tranche* problem (4.22). (b) The *fixed fractional-tranche* problem (4.29).



Search Algorithm for Fractional Tranche Configurations

We now describe a search algorithm that can be used to generate feasible solutions for (4.3). First, we use an LP-based approximation of $\hat{g}(\cdot)$ to compute function values and search directions efficiently. We then propose an algorithm, motivated by the gradient-projection method [82], to find good fractional tranche configurations. At each iteration, we take a step along a certain search direction and project the resulting point onto the feasible set. The fractional tranche configuration configuration returned by this algorithm is then rounded to obtain a feasible integral tranche configuration, which can then be used to construct feasible solutions for (4.3).

Consider the following LP-based approximation of (4.29), defined for $\widetilde{\boldsymbol{\tau}} \in$

$[1, T]^{A \times K}$, in which the production set X is replaced by its LP-relaxation:

$$
\begin{aligned}
\widetilde{g}(\widetilde{\tau}) := \max \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \pi_t f_{a,t} & \qquad (4.31) \\
\text{subject to } (\mathbf{c}, \mathbf{f}, \mathbf{h}, \mathbf{p}, \mathbf{r}) &\in \hat{Q}(\boldsymbol{\tau}) \\
(\mathbf{c}_a, \mathbf{r}_a, \boldsymbol{x}_a, \boldsymbol{w}) &\in \mathrm{LP}(X_a) \quad a \in \mathcal{A} \\
\boldsymbol{w} &\in \mathrm{LP}(W) \\
\sum_{a \in \mathcal{A}} C_a \boldsymbol{x}_a + D\boldsymbol{w} &\leqslant d \\
\hat{\tau}_{a,k+1} - \hat{\tau}_{a,k} = \widetilde{\tau}_{a,k+1} & \quad a \in \mathcal{A},\ k \in \{2 \dots K-1\} \\
\hat{\tau}_{a,2} - \hat{\tau}_{a,1} = \widetilde{\tau}_{a,2}. & \quad \forall a \in \mathcal{A} \qquad (4.32) \\
\hat{\tau}_{a,1} = 1. & \quad \forall a \in \mathcal{A}
\end{aligned}
$$

Here, the parameters $\widetilde{\boldsymbol{\tau}}$ represent the time elapsed between the transition between tranche levels $k-1$ and k. The change in variables, from $\hat{\boldsymbol{\tau}}$ in (4.29) to $\widetilde{\boldsymbol{\tau}}$ in (4.31), alters the constraints $\hat{\boldsymbol{\tau}} \in \hat{T}$ in problem (4.30) to simple bound constraints $\widetilde{\boldsymbol{\tau}} \in \Omega$, where

$$\Omega := [1, T]^{A \times K}.$$

The change in variables makes the projection of the point $\widetilde{\boldsymbol{\tau}}$ on to the convex set Ω (defined as $P_\Omega(\widetilde{\boldsymbol{\tau}}) = \underset{\hat{\boldsymbol{\tau}} \in \Omega}{\operatorname{argmin}} \|\widetilde{\boldsymbol{\tau}} - \hat{\boldsymbol{\tau}}\|^2$) simple to perform. There are both advantages and limitations to using an LP-based approximation of $\widetilde{g}(\cdot)$. An advantage is $\widetilde{g}(\cdot)$ can be computed easily by solving an LP, which is significantly cheaper than $\hat{g}(\cdot)$ which requires a MIP. Additionally, we can find good search directions, at a point $\widetilde{\boldsymbol{\tau}}$,

by solving many closely related LPs, which require only a few of additional simplex iterations once $\hat{g}(\widetilde{\boldsymbol{\tau}})$ is computed. A disadvantage is that LP(X) may not a suitable proxy for X. The algorithm described here works best when applied to production models with strong LP-relaxations.

We now focus on obtaining good feasible solutions to the following bound-constrained optimization problem:

$$\max_{\widetilde{\boldsymbol{\tau}} \in \Omega} \widetilde{g}(\widetilde{\boldsymbol{\tau}}). \tag{4.33}$$

Given a point $\tilde{\boldsymbol{\tau}}$, we first compute a search direction $\Delta\widetilde{g}(\cdot)$ using

$$\Delta\widetilde{g}_{a,k}(\widetilde{\boldsymbol{\tau}}) = \frac{\widetilde{g}(\widetilde{\boldsymbol{\tau}} + \delta e_{a,k}) - \widetilde{g}(\widetilde{\boldsymbol{\tau}})}{\delta} \quad \forall a \in \mathcal{A},\ k \in \mathcal{K}, \tag{4.34}$$

which is motivated by the forward-difference scheme for approximating gradients of smooth functions. Here, $\delta > 0$ is a small perturbation and $e_{a,k}$ is a unit vector along the coordinate corresponding to market a and tranche k. We can compute $\Delta\widetilde{g}(\widetilde{\boldsymbol{\tau}})$ using (4.34) with $(1 + AK)$ function evaluations of $\widetilde{g}(\cdot)$, each of which require a solution to an LP. Since δ is small, the LPs are closely related and can be solved cheaply using the simplex method. If the function $\widetilde{g}(\cdot)$ is continuous (which is not always the case because the LP in (4.33) may be infeasible for some $\widetilde{\boldsymbol{\tau}}$) then (4.34) is simply a finite-difference approximate gradient of the function $\widetilde{g}(\cdot)$. Although the search direction $\Delta\widetilde{g}_{a,k}(\widetilde{\boldsymbol{\tau}})$ may not necessarily be an ascent direction (since $\widetilde{g}(\cdot)$ is not differentiable), our experiments demonstrate that this heuristic works well in our experiments.

Algorithm 2: A search algorithm for (4.33).

Data: Initial point $\widetilde{\boldsymbol{\tau}}^0$, step size parameters α^0, β, $\alpha_{\min}$, finite difference size δ, termination threshold ϵ_{tol} and iteration limit N

$i \leftarrow 0$
$\delta^i \leftarrow \delta$
repeat
  $\Delta\widetilde{g}_{a,k}(\widetilde{\boldsymbol{\tau}}^i) \leftarrow \frac{\widetilde{g}(\widetilde{\boldsymbol{\tau}}^i + \delta^i e_{a,k}) - \widetilde{g}(\widetilde{\boldsymbol{\tau}}^i)}{\delta^i}$ $a \in \mathcal{A},\ k \in \mathcal{K}$
  while $\widetilde{g}(P_\Omega(\widetilde{\tau}^i + \alpha^i \Delta\widetilde{g}(\widetilde{\boldsymbol{\tau}}^i)) > \widetilde{g}(\widetilde{\boldsymbol{\tau}}^i)$ **do**
    $\alpha^{i+1} \leftarrow \beta\alpha^{i+1}$
    if $\alpha^{i+1} \leqslant \alpha_{\min}$ **then**
      return $\widetilde{\tau}^i$
  $\widetilde{\boldsymbol{\tau}}^{i+1} \leftarrow P_\Omega(\widetilde{\boldsymbol{\tau}}^i + \alpha^{i+1}\Delta\widetilde{g}(\widetilde{\tau}^i))$
  $\alpha^{i+1} \leftarrow \alpha^{i+1}/\beta$
  $\delta^{i+1} \leftarrow \min(\delta, \alpha^{i+1}\|\Delta\widetilde{g}(\widetilde{\tau}^i)\|$
  $i \leftarrow i + 1$
until $\|\Delta\widetilde{g}_c(\widetilde{\boldsymbol{\tau}})\| \leqslant \epsilon$ **or** $i = N$
return τ^i

Given $\widetilde{\boldsymbol{\tau}}^i$, an estimate of a solution of (4.33) at iteration i, our iterative search algorithm (see Algorithm 2) uses $\Delta\widetilde{g}_{a,k}(\widetilde{\boldsymbol{\tau}})$ to update $\widetilde{\boldsymbol{\tau}}^i$ using

$$\widetilde{\boldsymbol{\tau}}^{i+1} = P_\Omega\big(\widetilde{\boldsymbol{\tau}}^i + \alpha^i \Delta\widetilde{g}(\widetilde{\boldsymbol{\tau}}^i)\big), \tag{4.35}$$

where α^i is the step length along the given search direction $\Delta\widetilde{g}_{a,k}(\widetilde{\boldsymbol{\tau}})$. The projection operator $P_\Omega(\cdot)$ is defined in closed form as follows:

$$[P_\Omega(\widetilde{\boldsymbol{\tau}}^i)]_{a,k} = \begin{cases} 0 & \widetilde{\tau}^i_{a,k} \leqslant 1 \\ T & \widetilde{\tau}^i_{a,k} \geqslant T \\ \widetilde{\tau}^i_{a,k} & \text{otherwise.} \end{cases} \qquad a \in \mathcal{A},\ k \in \{2 \dots K\}.$$

We use a backtracking line search strategy to chose α^i. We start with an initial step size α^0 and select an acceptable step length if after a finite number of contractions of α^0 (using parameter $\beta < 1$) the following increase condition is satisfied:

$$\widetilde{g}(P_\Omega(\widetilde{\boldsymbol{\tau}}^i + (\beta)^n \alpha^0 \Delta\widetilde{g}(\widetilde{\boldsymbol{\tau}}^i))) > \widetilde{g}(\widetilde{\boldsymbol{\tau}}^i). \tag{4.36}$$

In our experiments, we enforce the $>$ above by requiring a sufficient increase of 10^{-3} in the function value. The algorithm terminates if one of three conditions are satisfied: (a) we cannot find a suitable step size $\alpha^i > \alpha_{\min}$ that satisfies (4.36); (b) an iteration limit is reached; or (c) $\|\Delta\widetilde{g}_c(\widetilde{\boldsymbol{\tau}}^i)\| \leqslant \epsilon_{tol}$, where

$$[\Delta\widetilde{g}_c(\widetilde{\tau}^i)]_{a,k} := \begin{cases} \min([\Delta\widetilde{g}_c(\widetilde{\boldsymbol{\tau}}^i)]_{a,k}, 0) & \widetilde{\tau}^i_{a,k} \leqslant 1 \\ \max([\Delta\widetilde{g}_c(\widetilde{\boldsymbol{\tau}}^i)]_{a,k}, 0) & \widetilde{\tau}^i_{a,k} \geqslant T \\ \Delta\widetilde{g}(\widetilde{\boldsymbol{\tau}}^i) & 1 < \widetilde{\tau}^i_{a,k} < T, \end{cases} \qquad a \in \mathcal{A},\ k \in \mathcal{K} \tag{4.37}$$

and $\epsilon_{tol} > 0$ is a termination threshold close to zero. The criterion involving $\Delta\widetilde{g}_c(\widetilde{\boldsymbol{\tau}}^i)$ is analogous to termination criterion used in box-constrained convex optimization problems. Notice that we do not use a fixed value of δ while computing search directions using (4.34). Instead, we use an adaptive perturbation parameter

$$\delta^i \leftarrow \min(\delta, \alpha^i \|\Delta\widetilde{g}(\widetilde{\boldsymbol{\tau}}^{i-1})\|),$$

which is always smaller than the size of the step taken by the algorithm.

The final step of our proposed solution technique is to convert the fractional

tranche configuration $\widetilde{\boldsymbol{\tau}}^*$ returned by Algorithm 2 to an integral tranche configuration $\boldsymbol{\tau}^*$ of (4.22) using the following rounding scheme:

$$\tau^*_{a,k} = 1 + \text{round}\Big(\sum_{k' \leqslant k} \widetilde{\tau}^*_{a,k'} \Big) \qquad a \in \mathcal{A},\ k \in \{2 \dots K\}. \tag{4.38}$$

We can use $\boldsymbol{\tau}^*$ to recover a feasible for (4.3) by evaluating $g(\boldsymbol{\tau}^*)$ using (4.22). Although there is no theoretical guarantee that this procedure results in a feasible solution of (4.3), we found that the simple rounding scheme in (4.38) works well in practice.

4.5 Decomposition Methods For Solution Bounds

In this section, we propose a market-based decomposition scheme to obtain solution bounds for IRR-based PSCs. Our approach is based on a reformulation of (4.18) which contains replicates of the shared decision variables $\boldsymbol{w} \in W$. Equality of the replicates is enforced through an additional constraint. A Lagrangian relaxation of the reformulated MIP can then be decomposed into independent subproblems, each involving only a single market. We propose two algorithms to solve the Lagrangian dual problem.

Market-Based Decomposition

Consider the following relaxation of (4.18)

$$\begin{aligned} & \max \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \pi_t f_{a,t} \\ \text{subject to } & (\mathbf{b}_a, \mathbf{c}_a, \mathbf{f}_a, \mathbf{h}_a, \mathbf{p}_a, \mathbf{r}_a) \in Y_a \qquad a \in \mathcal{A} \\ & (\mathbf{c}_a, \mathbf{r}_a, \boldsymbol{x}_a, \boldsymbol{w}) \in \mathrm{LP}(X_a) \quad a \in \mathcal{A} \\ & \boldsymbol{w} \in \mathrm{LP}(W) \\ & \sum_{a \in \mathcal{A}} C_a \boldsymbol{x}_a + D\boldsymbol{w} \leqslant d, \end{aligned} \tag{4.39}$$

where the sets X_a and W are replaced by their corresponding LP-relaxations, but the sets Y_a maintain their current status, to prevent the relaxation in (4.39) from becoming too weak. In (4.39), the constraints involving sets Y_a are decomposable by market. The challenge in decomposing the *entire problem* by market is in the handling of the shared variables $\boldsymbol{w} \in W$ and the linking constraints defined in (4.4) (which is the last constraint in (4.39)). We handle $\boldsymbol{w}$ by introducing replicates $w_a \in W$ for each market $a \in \mathcal{A}$ and enforce the additional equality constraint $\frac{1}{A} \sum_{a' \in \mathcal{A}} w_{a'} = w_a \; \forall a \in \mathcal{A}$. We enforce $(\mathbf{c}_a, \mathbf{r}_a, \boldsymbol{x}_a, \boldsymbol{w}_a) \in \mathrm{LP}(X_a)$ and introduce

the replicates into the linking constraints to obtain the following formulation:

$$\max \sum_{a \in \mathcal{A}} \sum_{t \in \mathcal{T}} \pi_t f_{a,t} \tag{4.40a}$$

$$\text{subject to } (\mathbf{b}_a, \mathbf{c}_a, \mathbf{f}_a, \mathbf{h}_a, \mathbf{p}_a, \mathbf{r}_a) \in Y_a \qquad \forall a \in \mathcal{A} \tag{4.40b}$$

$$(\mathbf{c}_a, \mathbf{r}_a, \mathbf{x}_a, \mathbf{w}_a) \in \mathrm{LP}(X_a) \quad \forall a \in \mathcal{A} \tag{4.40c}$$

$$w_a \in \mathrm{LP}(W) \quad \forall a \in \mathcal{A} \tag{4.40d}$$

$$\frac{1}{A} \sum_{a' \in \mathcal{A}} w_{a'} = w_a \qquad \forall a \in \mathcal{A} \tag{4.40e}$$

$$\sum_{a \in \mathcal{A}} C_a \mathbf{x}_a + D w_a \leqslant d \tag{4.40f}$$

We use *Lagrangian relaxation* to obtain solution bounds for (4.40) by dualizing the constraints (4.40e) and (4.40f). Define $\mathbf{\Theta} := (\Delta_1, \Delta_2, \ldots, \Delta_A, \theta)$ to be the vector of Lagrangian multipliers, where Δ_a are associated with constraints (4.40e) and $\theta \geqslant 0$ are associated with the constraints (4.40f). Without loss of generality, we can assume that $\sum_{a \in \mathcal{A}} \Delta_a = 0$. Let $\widetilde{\mathbf{y}}_a := (\mathbf{b}_a, \mathbf{c}_a, \mathbf{f}_a, \mathbf{h}_a, \mathbf{p}_a, \mathbf{r}_a)$ and $\widetilde{\mathbf{x}}_a := (\mathbf{c}_a, \mathbf{r}_a, \mathbf{x}_a, \mathbf{w}_a)$ denote the vectors of primal variables. Using $\sum_{a \in \mathcal{A}} \Delta_a = 0$, we know that the Lagrangian of (4.40) becomes $\sum_{a \in \mathcal{A}} L_a(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a; \mathbf{\Theta})$ where

$$L_a(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a; \mathbf{\Theta}) := \sum_{t \in \mathcal{T}} \pi_t f_{a,t} - \Delta_a^{\mathsf{T}} w_a - \theta^{\mathsf{T}} \Big(\frac{d}{A} - C_a x_a - \frac{D w_a}{A} \Big).$$

For each market $a \in \mathcal{A}$, we define

$$L_a^D(\boldsymbol{\Theta}) := \max_{\widetilde{\boldsymbol{x}}_a, \boldsymbol{w}_a \widetilde{\boldsymbol{y}}_a} \left\{ L_a(\widetilde{\boldsymbol{x}}_a, \widetilde{\boldsymbol{y}}_a; \boldsymbol{\Theta}) \text{ s.t. } \widetilde{\boldsymbol{x}}_a \in \mathrm{LP}(X_a), \boldsymbol{w}_a \in \mathrm{LP}(W), \widetilde{\boldsymbol{y}}_a \in Y_a \right\}. \tag{4.41}$$

It is clear that for any fixed $\boldsymbol{\Theta}$, the quantity $\sum_{a \in \mathcal{A}} L_a^D(\boldsymbol{\Theta})$ is an upper bound on the optimal objective of (4.40). We can compute $L_a^D(\boldsymbol{\Theta})$ for each market $a \in \mathcal{A}$, independently, by solving a MIP which is easier to solve than a single-market instance of the (4.18) because the binary decision variables in X are relaxed. We can find the best (smallest) solution bound (over the possible range of dual solutions $\boldsymbol{\Theta}$) by solving the following Lagrangian dual problem:

$$G_D^* := \min_{\boldsymbol{\Theta}} \left\{ \sum_{a \in \mathcal{A}} L_a^D(\boldsymbol{\Theta}) : \theta \geqslant 0, \sum_{a \in \mathcal{A}} \Delta_a = 0 \right\}. \tag{4.42}$$

Note that one may chose not to relax the sets X_a and W in the definition of (4.18). This choice may result in a better solution bound, from (4.42), but requires additional computational effort to solve each Lagrangian subproblem in (4.41). The trade-off between solution bound quality and computational effort depends on the strength of $\mathrm{LP}(W)$ and $\mathrm{LP}(X_a)$.

Solving the Lagrangian Dual Problem

We now discuss two algorithms to solve the formulations of the previous section. The first is a textbook *subgradient* algorithm while the second is a regularized decomposition method based on progressive hedging. The regularization term

Algorithm 3: DD+LP: Subgradient Method for (4.40).

Data: An initial set of primal variables $(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a)^0$, $a \in \mathcal{A}$, an initial set of dual multipliers $\Delta_1^0, \Delta_2^0, \ldots, \Delta_A^0$, θ^0 such that $\sum_{a\in\mathcal{A}} \Delta_a^0 = 0$, an initial step size $\eta_w^0 > 0$ and $\eta_s^0 > 0$, an iteration limit $N \geqslant 1$ and a termination tolerance $\epsilon_{tol} > 0$.

Result: A valid upper bound for (4.18)

$i \leftarrow 0$

$G_D^i \leftarrow \infty$

repeat

  Solve (4.41) with $\boldsymbol{\Theta} = \boldsymbol{\Theta}^i$ for $(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a)^{i+1}$, for each $a \in \mathcal{A}$;

  $\Delta_a^{i+1} \leftarrow \Delta_a^i + \delta_v^i \left(w_a^{i+1} - \frac{\sum_{a\in\mathcal{A}} w_a^{i+1}}{A} \right) \quad a \in \mathcal{A}$;

  $\theta_i^{i+1} \leftarrow \theta_i^i + \eta_d^i \left(d - \sum_{a\in\mathcal{A}} C_a x_a^{i+1} - D \sum_{a\in\mathcal{A}} \frac{w_a^{i+1}}{A} \right)_+$;

  $\delta_w^{i+1} \leftarrow \delta_w^0/(i+1)$;

  $\delta_s^{i+1} \leftarrow \delta_s^0/(i+1)$;

  $G_D^{i+1} \leftarrow \min(G_D^i, \sum_{a\in\mathcal{A}} L_a^D(\boldsymbol{\Theta}^i))$;

  $i \leftarrow i+1$;

until $\sum_{a\in\mathcal{A}} \|(\widetilde{\mathbf{y}}_a, \widetilde{\mathbf{x}}_a)^{i+1} - (\widetilde{\mathbf{y}}_a, \widetilde{\mathbf{x}}_a)^i\| + \|\boldsymbol{\Theta}^{i+1} - \boldsymbol{\Theta}^i\| > \epsilon_{tol}$ *and* $i \leqslant N$

return G_D^i

helps compute high quality solution bounds using fewer iterations than the traditional subgradient method, but each iteration is more expensive because it requires solving a set of mixed-integer quadratic programs (MIQPs).

A Subgradient Method

We describe a subgradient algorithm for (4.40) in Algorithm 3. Given an estimate of dual multipliers $\boldsymbol{\Theta}^i$, each iteration solves the problems in (4.41) for $\boldsymbol{\Theta} = \boldsymbol{\Theta}^i$ to obtain $(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a)^{i+1}$, for each $a \in \mathcal{A}$. Next, we update the dual multipliers $\boldsymbol{\Theta}^i = (\Delta_1^i, \Delta_2^2, \ldots, \Delta_A^i, \theta^i)$ by taking a step along the subgradient $\nabla L_a^D(\boldsymbol{\Theta}^i)$, using step sizes $\eta_w^i \geqslant 0$ and $\eta_d^i \geqslant 0$. If the sequence of step sizes $\{\eta_w^i\}$ and $\{\eta_d^i\}$ satisfy the

Algorithm 4: D+LP: Regularized Decomposition Method for (4.40).

Data: An initial set of primal variables $(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a)^0$, $a \in \mathcal{A}$, an initial set of dual multipliers $\Delta_1^0, \Delta_2^2, \ldots, \Delta_A^0, \theta^0$ such that $\sum_{a\in\mathcal{A}} \Delta_a = 0$, an initial step size δ_s^0, an initial penalty parameter ρ^0, penalty decrease rate ρ_d, minimum penalty parameter ρ_m, iteration limit N and a convergence threshold ϵ_{tol}.

Result: A valid lower bound for (4.18)

$i \leftarrow 0$

$\bar{w}^0 \leftarrow (\sum_{a\in\mathcal{A}} w_a^0)/A$

repeat

  Solve (4.43) with $\boldsymbol{\Theta} = \boldsymbol{\Theta}^i$ for $(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a)^{i+1}$, for each $a \in \mathcal{A}$;

  $\bar{w}^{i+1} \leftarrow (\sum_{a\in\mathcal{A}} w_a^{i+1})/A$;

  $\Delta_a^{i+1} \leftarrow \Delta_a^i + \rho^i\left(w_a^{i+1} - \bar{w}^{i+1}\right) \quad a \in \mathcal{A}$;

  $\theta_i^{i+1} \leftarrow \theta_i^i + \eta_d^i\left(d - \sum_{a\in\mathcal{A}} C_a x_a^{i+1} - D\sum_{a\in\mathcal{A}} \frac{w_a^{i+1}}{A}\right)_+$;

  $\delta_s^{i+1} \leftarrow \delta_s^0/(i+1)$;

  $\rho^i \leftarrow \max(\rho_d\rho^i, \rho_m)$;

  $i \leftarrow i+1$;

until $\sum_{a\in\mathcal{A}} \|(\widetilde{\mathbf{y}}_a, \widetilde{\mathbf{x}}_a)^{i+1} - (\widetilde{\mathbf{y}}_a, \widetilde{\mathbf{x}}_a)^i\| + \|\boldsymbol{\Theta}^{i+1} - \boldsymbol{\Theta}^i\| > \epsilon_{tol}$ *and* $i \leqslant N$

$G_D^i \leftarrow \sum_{a\in\mathcal{A}} L_a^D(\boldsymbol{\Theta}^i)$;

return G_D^i

conditions $\sum_i \eta_w^i = \infty$, $\sum_i \eta_d^i = \infty$, $\eta_w^i \to 0$ and $\eta_d^i \to 0$ as $i \to \infty$, then Algorithm 3 converges to the optimal dual solution G_D^* [30]. The algorithm terminates when an iteration limit is reached or when the change in primal and dual variables falls below a threshold ϵ_{tol}.

A Regularized Decomposition Method

One of the disadvantages of the subgradient method discussed above is that it may take many iterations before the dual multipliers converge to a stable accumulation point. To prevent Algorithm 3 from taking an extremely large number of steps, we

regularize each Lagrangian subproblem in (4.41) by introducing a prox-term into the objective:

$$(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a)^{i+1} = \underset{\widetilde{\mathbf{x}}_a, w_a, \widetilde{\mathbf{y}}_a}{\operatorname{argmax}} \big\{ L_a(\widetilde{\mathbf{x}}_a, \widetilde{\mathbf{y}}_a; \boldsymbol{\Theta}) - \frac{\rho^i}{2} \|w_a - \bar{w}^i\|^2 : \widetilde{\mathbf{x}}_a \in \mathrm{LP}(X_a), \boldsymbol{w} \in \mathrm{LP}(W),\ \widetilde{\mathbf{y}}_a \in Y_a \big\} \tag{4.43}$$

where $\bar{w}^i_a = \sum_{a \in \mathcal{A}} w^i_a / A$. Observe that the subproblem is now a concave MIQP which can solved with existing optimization software. After solving the MIQPs in (4.43), we update the multipliers using stepsizes η^i_d (for θ) and ρ^i (for $\Delta_1, \Delta_2, \ldots, \Delta_A$), which is also the penalty parameter in (4.43). Algorithm 4 summarizes the method. Since each subproblem defined in (4.43) is nonconvex, there are no known convergence results for this method. Nevertheless, it has been successfully used to find both primal and dual solutions in the context of stochastic mixed-integer programming [31, 108, 100, 20].

Updating ρ^i. It is well known that the performance of regularized decomposition methods is sensitive to the choice of the penalty parameter ρ^i in (4.43). Many update rules for ρ^i have been considered [81, 107, 46]. One common approach is to use is a constant penalty parameter ρ_c for the entire algorithm. Another possibility is to vary the penalty parameters ρ^i adaptively at each iteration to make performance less dependent on the initial choice of the penalty parameter. Typically, these adaptive schemes chose an increasing sequence of $\{\rho^i\}$ such that $\rho^i \to \infty$. The constant as well as the adaptively increasing scheme try to ensure that $\|\bar{w}^i - (\sum_{a \in \mathcal{A}} w^i_a / A)\|^2 \to 0$

as $i \to \infty$. While consistency of the various copies of variables $w_a \ \forall a \in \mathcal{A}$ is important for producing primal feasible solutions, they may not result in good dual solutions. The update equations in Algorithm 4 suggest that large values of ρ place a large penalty on deviations from consensus. Conversely, smaller values of ρ tend to regularize the dual variables, thereby preventing them from making very large steps. With these considerations in mind, we propose the following update scheme:

$$\rho^{i+1} = \max\{\rho_d \rho^i, \rho_m\} \tag{4.44}$$

where $\rho_d < 1$, and $\rho_m > 0$ is the minimum allowable value for ρ^i. In order to obtain a valid solution bound on the optimal objective of (4.40), we use the dual multipliers $\mathbf{\Theta}^*$ returned by the algorithm to compute $\sum_{a \in \mathcal{A}} L_a^D(\mathbf{\Theta}^*)$ using (4.41).

4.6 Experimental Results

We conducted numerical experiments to test the effectiveness of the MIP formulation (4.18), the continuous domain search algorithm of Section 4.4, and the decomposition methods from Section 4.5. The two main purposes of our experiments are:

- To compare the effectiveness of the MIP formulation (4.18) and Algorithm 2 in obtaining feasible solutions for (4.3); and
- To compare the effectiveness of Algorithms 3 and 4 and the MIP formulation (4.18) in obtaining solution bounds for (4.3).

We report tests conducted on randomly generated instances of a sample application problem. Appendix A provides more details on how the instances were generated. We used Gurobi 5.0.1 (with default options) to solve all LPs, MIPs, and MIQPs to 0.1% tolerance. We ran all experiments using two threads on a Dell R420 with 128G Memory with a 2.66GHz Intel Core2 Quad 2.30G E5-2470 Xeon Chip set.

Sample Application

We conducted numerical experiments on a multi-period production planning problem. In this problem, the shared resources are a set of production facilities $\mathcal{I}$ that manufacture a single product to satisfy the demand of customers $\mathcal{J}$ distributed across markets $\mathcal{A}$. The development of each facility $l \in \mathcal{I}$ over the planning horizon $\mathcal{T}$ involves both discrete and continuous decisions. Discrete decisions include customer and facility opening times, while continuous decisions concern the amount of product distributed from shared facility $l \in \mathcal{I}$ to customer $j \in \mathcal{J}$. The demand $\hat{d}_{jt}$ for the product from each customer $j \in \mathcal{J}$ during time period $t \in \mathcal{T}$ is assumed to be known a priori, and there is a supply limit $\hat{s}_{l,t}$ for each facility $l \in \mathcal{I}$ during each time period $t \in \mathcal{T}$. The set of constraints that define this multi-period production

set are as follows:

$$\sum_{j \in \mathcal{J}_a} \kappa_{l,j} x_{l,j,t} = r_{a,t} \qquad a \in \mathcal{A},\ t \in \mathcal{T} \tag{4.45a}$$

$$\sum_{j \in \mathcal{J}_a} \omega_{l,j} x_{l,j,t} + \sum_{l \in \mathcal{I}} \frac{\gamma_l}{A}(z_{l,t} - z_{l,t-1}) + \sum_{j \in \mathcal{J}_a} \sigma_j (v_{j,t} - v_{j,t-1}) = c_{a,t} \quad a \in \mathcal{A},\ t \in \mathcal{T} \tag{4.45b}$$

$$\sum_{l \in \mathcal{I}} x_{l,j,t} \leqslant \hat{d}_{j,t} v_{j,t} \qquad j \in \mathcal{J},\ t \in \mathcal{T} \tag{4.45c}$$

$$\sum_{j \in \mathcal{J}} x_{l,j,t} \leqslant \hat{s}_{l,t} z_{l,t} \qquad l \in \mathcal{I},\ t \in \mathcal{T} \tag{4.45d}$$

$$z_{l,t} \leqslant z_{l,t+1} \qquad l \in \mathcal{I},\ t \in \mathcal{T} \tag{4.45e}$$

$$v_{j,t} \leqslant v_{j,t+1} \qquad j \in \mathcal{J},\ t \in \mathcal{T}, \tag{4.45f}$$

where the set J_a denotes the set of all customers $j \in \mathcal{J}$ that belong to a market $a \in \mathcal{A}$, the continuous decision variable $x_{l,j,t}$ determines the amount of the product distributed from facility $l \in \mathcal{I}$ to customer $j \in \mathcal{J}$ during time $t \in \mathcal{T}$, $z_{l,t}$ is a binary variable that is set to 1 if the facility $l \in \mathcal{I}$ is open during time $t \in \mathcal{T}$, and $v_{j,t}$ is binary variable set to 1 if supply to a customer $j \in \mathcal{J}$ is open during time period $t \in \mathcal{T}$. The revenues and expenses considered in the model are as follows:

- A fixed cost γ_l for opening the shared facility $l \in \mathcal{I}$ which are evenly distributed across all markets.
- A fixed cost σ_j to initiate supply to the customer $j \in \mathcal{J}$.
- Transportation costs $\omega_{l,j}$ per unit of product supplied from facility $l \in \mathcal{I}$ to customer $j \in \mathcal{J}$.
- Revenue $\kappa_{l,j}$ obtained by shipping the product from facility $l \in \mathcal{I}$ to customer $j \in \mathcal{J}$ during time period $t \in \mathcal{T}$.

The definitions of the various sets introduced in Section 4.3 for the sample application problem in (4.45) are as follows:

$$X = \big\{x \in \mathbb{R}_+^{I\times J\times T},\ v \in \{0,1\}^{J\times T},\ z \in \{0,1\}^{I\times T},\ c \in \mathbb{R}_+^{A\times T}\ r \in \mathbb{R}_+^{A\times T} :\ (4.45)\big\}$$

$$\begin{aligned} X_a = \big\{x \in \mathbb{R}_+^{I\times J_a\times T},\ v \in \{0,1\}^{J_a\times T},\ z \in \{0,1\}^{I\times T},\ \mathbf{c_a} \in \mathbb{R}_+^{T},\ \mathbf{r_a} \in \mathbb{R}_+^{\times T} :\\ \sum_{j\in\mathcal{J}_a} \kappa_{l,j} x_{l,j,t} = r_{a,t} \quad t \in \mathcal{T};\\ \sum_{j\in\mathcal{J}_a} \omega_{l,j} x_{l,j,t} + \sum_{l\in\mathcal{I}} \gamma_l \frac{(z_{l,t} - z_{l,t-1})}{A} + \sum_{j\in\mathcal{J}_a} \sigma_j (v_{j,t} - v_{j,t-1}) = c_{a,t} \quad t \in \mathcal{T};\\ \sum_{l\in\mathcal{I}} x_{l,j,t} \leqslant \hat{d}_{j,t} v_{j,t}, \quad j \in \mathcal{J}_a,\ t \in \mathcal{T};\\ v_{j,t} \leqslant v_{j,t+1} \quad j \in \mathcal{J}_a,\ t \in \mathcal{T}\big\} \end{aligned}$$

$$W = \{z \in \{0,1\}^{I\times T} : z_{l,t} \leqslant z_{l,t+1} \quad l \in \mathcal{I},\ t \in \mathcal{T}\}$$

The constraints (4.45d) are linking constraints which can be rewritten as

$$\sum_{a \in \mathcal{A}} \sum_{j \in \mathcal{J}_a} x_{l,j,t} \leqslant \hat{s}_{l,t} z_{l,t} \quad \forall l \in \mathcal{I}, t \in \mathcal{T}$$

to match the structure in (4.4). Table 4.5 provides an overview of the datasets used in the computational results. For each configuration of facilities, customers and time periods, we generate five datasets. For each dataset, we generate six separate instances of the problem by distributing the set of customers evenly into 1-6 markets; see Appendix A for details.

Table 4.5: Overview of the 90 instances used in the computational experiments.

Facilities	Time Periods	Customers	Markets	# Instances
15	20	40	1-6	5
15	20	50	1-6	5
20	20	50	1-6	5

Comparing Feasible Solutions

In this section, we compare the quality of the solutions obtained using a multi-start variant of Algorithm 2 (denoted as HEU+M) with those obtained using the formulation (4.18) (denoted as TRMIP) for problems with more than 1 market. Both methods ran with a time limit of 2 hours. In addition to comparing the solution quality after 2 hours of computation, we also measured the progress made by the methods after 5 minutes and 30 minutes of computation. For the MIP formulation (4.18), we computed variable bounds $m^h_{a,k,t}$, $M^h_{a,k,t}$ and $M^p_{a,k,t}$

as follows. First, we use $M^x_{i,j,t}$ and $M^c_{a,t}$ to denote the maximum production and maximum expenditure, that is,

$$\begin{aligned} M^x_{i,j,t} &= \min(\hat{s}_{l,t}, d_{j,t}) && l \in \mathcal{I},\ j \in \mathcal{J},\ t \in \mathcal{T} \\ M^c_{a,t} &= \sum_{j \in \mathcal{J}_a} \omega_{l,j} M^x_{l,j,t} + \sum_{l \in \mathcal{I}} \frac{\gamma_l}{A} + \sum_{j \in \mathcal{J}_a} \sigma_j && a \in \mathcal{A},\ t \in \mathcal{T}. \end{aligned}$$

These quantities are then used to compute $m^h_{a,k,t}$, $M^h_{a,k,t}$ and $M^p_{a,k,t}$ using the formulae

$$\begin{aligned} M^p_{a,k,t} &= \sum_{j \in \mathcal{J}_a} \kappa_{i,j,t} M^x_{i,j,t} && a \in A,\ k \in \mathcal{K},\ t \in \mathcal{T} \\ m^h_{a,k,t} &= \sum_{s \leqslant t} \frac{-M^c_{a,s}}{(1+\lambda_k)^s} && a \in \mathcal{A},\ k \in \{2 \dots K\},\ t \in \mathcal{T} \\ M^h_{a,k,t} &= \sum_{s \leqslant t} \frac{M^p_{a,s}}{(1+\lambda_k)^s} && a \in \mathcal{A},\ k \in \{2 \dots K\},\ t \in \mathcal{T}. \end{aligned}$$

For HEU+M, we used a multi-start variant of Algorithm 2, generating each starting point $\boldsymbol{\tau}^0 \in T$ from the following randomized procedure. First, for each market, we solved the following MIP:

$$\begin{aligned} &\max \sum_{t \in \mathcal{T}} \pi_t f_{a,t} \\ \text{subject to } &(\mathbf{b}_a, \mathbf{c}_a, \mathbf{f}_a, \mathbf{h}_a, \mathbf{p}_a, \mathbf{r}_a) \in Y_a \\ &(\mathbf{c}_a, \mathbf{r}_a, \boldsymbol{x}_a, \boldsymbol{w}) \in \mathrm{LP}(X_a), \end{aligned} \tag{4.46}$$

which is a relaxed version of (4.18) without the linking constraints (4.4). (For each market, (4.46) is solved in less than a minute.) Let $b^*_{a,k,t}$ denote the optimal values of the binary variables $b_{a,k,t}$ obtained after by solving (4.46) for a given market $a \in \mathcal{A}$. Using this quantity, we generated a sequence of initial fractional configurations $\{(\boldsymbol{\tau}^0)^n\}$ as follows:

$$(\boldsymbol{\tau}^0_{a,k})^n = \begin{cases} t + \phi^n_a, & \text{when } b^*_{a,k,t} - b^*_{a,k,t-1} = 1 \\ T, & \text{otherwise,} \end{cases} \qquad a \in \mathcal{A},\ k \in \{2 \dots K\}, \tag{4.47}$$

where $\phi^n_a \sim \mathcal{U}(-.05n, .05n)$ is a uniform random variable. The algorithm goes through as many starting points as it can within the two-hour time limit. For each starting point, we searched for a fractional tranche configuration using Algorithm 2 with $\alpha^0 = 0.1$, $\alpha_{\min} = 10^{-3}$, $\delta = 10^{-3}$, $\epsilon = 10^{-3}$, and $N = 20$. We round the terminal fractional tranche configuration using (4.38) to obtain a valid integral tranche configuration, and recover a feasible solution for (4.3) using (4.22). For HEU+M, we include the time taken for each of the steps described above.

Table 4.6 measures solution quality of a method as a percentage gap to the best found feasible solution using any method. We define this gap as (BFS-SOL)/BFS where BFS is the best feasible solution found using any method and SOL is the best feasible solution found using the method in consideration. Each row of Table 4.6 reports a summary statistic (average or maximum) over the 15 instances with the same number of markets at three different time intervals; (a) t=300s, (b) t=1800s and (c) t=7200s. At the end of the two-hour interval, the feasible solutions obtained using HEU+M are at most 2.43% away from the best feasible solution and on average

Table 4.6: Summary statistics for feasible solution quality at three different time intervals; (a) t=300s (b) t = 1800s and (c) t = 7200s

Average gap to the best feasible solution (%)

Markets	TRMIP			HEU+M		
	t=300	t =1800	t =7200	t=300	t =1800	t =7200
2	3.89	0.01	0.00	0.77	0.17	0.15
3	2.76	0.37	0.00	0.70	0.22	0.18
4	5.34	1.02	0.23	0.67	0.27	0.17
5	8.47	1.39	0.30	0.62	0.33	0.10
6	8.23	1.57	0.79	0.68	0.41	0.25
Average	5.74	0.87	0.26	0.69	0.28	0.17

Maximum gap to the best feasible solution (%)

Markets	TRMIP			HEU+M		
	t=300	t =1800	t =7200	t=300	t =1800	t =7200
2	15.12	0.07	0.00	3.12	1.21	1.21
3	8.25	2.20	0.00	1.91	0.86	0.86
4	9.67	4.77	1.01	3.44	2.54	1.83
5	17.75	3.77	1.60	2.51	1.33	0.54
6	23.79	3.30	1.92	2.43	2.43	2.43
Maximum	23.79	4.77	1.92	3.44	2.54	2.43

0.17% away. In contrast, the feasible solutions obtained using TRMIP are at most 1.92% away from the best found feasible solution with an average-case performance of 0.26%.

We make the following observations.

- TRMIP solves each of the 30 instances with less than or equal to 3 markets. For these instances, HEU+M is 0.15-0.18% away from the optimal solution for two and three markets, respectively. For instances with four or more markets, there is a decrease in quality of solutions found using TRMIP. The gap to the

best feasible solution increases from 0.23% for instances with four markets to 0.79% for instances with six markets. By contrast, the performance of HEU+M was independent of the number of markets. Our results suggest that HEU+M is suitable for finding feasible solutions for problems with more markets.

- After 5 minutes of computation, HEU+M consistently outperforms TRMIP, regardless of the number of markets, finding solutions that are on average only 0.67% away from the best known feasible solution. Even in the worst case, HEU+M is only 3.44% away from the best known feasible solution. By contrast, TRMIP could be as much as 23.79% away from the best known feasible solution. We observed similar results after 30 minutes, for instances with three to six markets.

We conclude that HEU+M is the preferred choice for obtaining feasible solutions of (4.3) when the number of markets is greater than three, while TRMIP is preferable when the number of markets is at most three. HEU+M is also suitable for finding high quality feasible solutions quickly.

Comparing Solution Bounds

We now compare the solution bounds obtained using Algorithm 3 (DD+LP), Algorithm 4 (D+LP), and TRMIP for problems with two to six markets. We measured the progress made by each of the algorithms after 30 minutes and two hours of computation.

Table 4.7: Summary statistics for solution bound quality at two different time intervals; (a) t = 1800s and (b) t = 7200s

Average gap to the best feasible solution (%)

Markets	TRMIP		DD+LP		D+LP	
	t=1800	t =7200	t =1800	t=7200	t =1800	t =7200
2	0.23	0.01	2.41	2.13	2.87	2.17
3	14.93	0.03	3.24	2.57	3.34	2.59
4	43.25	11.65	4.49	2.99	3.81	2.92
5	66.43	36.77	6.04	3.54	3.93	3.05
6	81.20	51.86	8.80	4.74	4.60	3.53
Average	41.21	20.06	4.99	3.19	3.71	2.85

Maximum gap to the best feasible solution (%)

Markets	TRMIP		DD+LP		D+LP	
	t=1800	t =7200	t =1800	t=7200	t =1800	t =7200
2	3.25	0.12	3.75	3.39	6.69	3.50
3	40.20	0.43	5.57	3.85	5.97	4.03
4	59.60	24.64	6.26	4.61	6.00	4.33
5	79.00	49.18	10.13	5.90	5.92	4.99
6	104.05	64.10	15.15	8.05	7.03	5.69
Maximum	104.05	64.10	15.15	8.05	7.03	5.69

For DD+LP, we run Algorithm 3 with $\eta_w^0 = 4.5$, $\eta_d^0 = 1$, and $\epsilon_{tol} = 10^{-5}$ until the termination criterion is met or the chosen time limit is reached. The initial dual multipliers $\boldsymbol{\Theta}^0$ are set to zero, while the initial primal variables are set to the solution obtained using HEU+M with a time limit of 300 seconds. In reporting the results, we include the time taken to compute the initial primal and dual solutions. For Algorithm 3 (DD+LP), each iteration generates a set of dual multipliers $\boldsymbol{\Theta}^i$ and a valid solution bound. At the end of each iteration, we keep track of the best (lowest) bound obtained from any of the previous iterations of DD+LP.

We run Algorithm 4 (D+LP) with parameters $\rho^0 = 2$, $\eta_d^0 = 1$, $\epsilon_t = 10^{-3}$, $\rho_d = 0.9$ and $\rho_m = 10^{-2}$ until the termination criterion is met or the time limit is reached. We set the initial dual multipliers and primal variables using the same procedure as that used for DD+LP. For D+LP we compute a valid solution bound, from the dual multipliers, using the procedure described in Section 13. We perform this procedure at $t = 1600$s and $t = 7000$s to ensure that a valid bound is available before the $t = 1800$s and $t = 7200$s time stamps. To eliminate bias, we chose parameters η_w^0, η_d^0, ρ^0, and ρ^d to be the values that yield the best performance at the end of two hours of computation, from a grid of parameters on a training dataset that is independent of the datasets used to report the results in this section. (We generated the training dataset using the same procedure as the other datasets.)

We report quality of solutions produced by TRMIP, DD+LP, and D+LP in Table 4.7, as a percentage gap to the best found feasible solution. We calculate this gap from the formula (BFS-BD)/BFS, where BFS is the best feasible solution found with any method and BD is the best bound obtained by using the method under consideration. Each row in Table 4.7 reports a summary statistic (average or maximum) over fifteen instances measured at the two time intervals of thirty minutes and two hours.

At the end of the two-hour time limit, we observe that both DD+LP and D+LP produce bounds that are on average, 2-5% away from the best known feasible solution, figures that are more than an order of magnitude better than TRMIP. We also observed that D+LP and DD+LP produce bounds that were in 2-9% range at the end of 1800s, significantly better than the bounds obtained by TRMIP, both

for $t = 1800$s and $t = 7200$s. Our results suggest that TRMIP is not suitable for computing solution bounds for (4.3). When comparing the two decomposition algorithms DD+LP and D+LP, we found that the performance of D+LP is more consistent when number of markets increases. Both DD+LP and DD+LP produce bounds of about 2% for instances with fewer than three markets. However, on instances with four or more markets, the average and worst case performance of D+LP improves on that of DD+LP. Most notably, for the six-market instances, D+LP is 33% better than DD+LP in average-case performance. At $t = 1800$s, the bounds produced by D+LP for instances with four to six markets is 50% better than DD+LP for the same computation time.

We conclude that D+LP is the preferred choice of algorithms to obtain solution bounds for problems with three or more markets, while TRMIP is competitive only when the number of markets is two.

4.7 Concluding Remarks

We have proposed formulations, solution techniques, and algorithms for operational planning problems influenced by IRR-based PSCs. We presented a natural MIP formulation for the problem and a search algorithm based on a novel continuous domain formulation. We also proposed market-based decomposition methods to compute solution bounds. Our experiments, on a sample application problem, demonstrated that the MIP formulation was suitable for problems with three or fewer markets. When the number of markets is greater than three, our continuous-

domain search algorithms are suitable for finding good feasible solutions and our decomposition algorithm was suitable for finding solution bounds. We conclude that a combination of the continuous domain search algorithms and market-based decomposition schemes is a suitable integrated approach for operational planning models involving IRR-based PSCs with multiple markets.

5 APPROXIMATING MIP FORMULATIONS FOR COMBINATORIAL PROBLEMS VIA APPROXIMATE LP ROUNDING

Many important problems in applications such as health-care, biology finance, and operations research are NP-hard. Although it is difficult to solve these NP-hard problems exactly, there may still be ways to efficiently compute solutions that are 'good enough'.There has been a lot of successful research in developing effective heuristics to find 'good enough' solutions to NP-hard problems. However, such approaches are less appealing because they do not offer any theoretical bounds on runtime and solution quality. In contrast, the field of *approximation algorithms* is focussed on developing a theoretical framework to design algorithms that find provably near-optimal solutions to NP-hard problems. Good approximation algorithms can get to within a constant factor of an optimal solution. Sometimes, there are even better approximation algorithms that get arbitrarily close to an optimal solution by trading off computational effort with solution quality. Unfortunately, there are still several classes of problems for which no known polynomial-time algorithms get to within a constant factor of an optimal solution. Vazirani [101] provides an overview of the theory and algorithms used in the field of approximation algorithms.

Approximation algorithms for NP-hard problems are either combinatorial or LP-based. The combinatorial algorithms include greedy algorithms, randomized algorithms and reduction to problems with known poly-time algorithms. On the other hand, LP-based algorithms broadly fall into two categories; LP-rounding and primal dual algorithms. In this work, we focus on approximation algorithms

based on LP-rounding. LP-rounding based approximation schemes [101, chs. 12-26] have been successfully used for a wide range of NP-hard problems in applications like machine-learning [85, 112, 57, 84], computer vision [53, 16, 24], natural language processing [14, 54], statistics [7, 99]. An LP-rounding scheme consists of the following three-steps: First, we *construct* an integer (binary) linear program (IP) formulation of a given problem. Second, we *relax* the IP to a linear program (LP) by replacing the constraints $x \in \{0, 1\}$ by $x \in [0, 1]$. We then *solve* the LP. Finally, we *round* the solution of the LP to obtain a feasible solution for the original IP problem and hence a feasible solution for the original problem. LP rounding is known to work well on a problems such as set-cover, set packing, multiway-cut, facility location etc. and comes with theoretical guarantees for runtime and solution quality.

One of the main reasons for the success of LP-rounding schemes has been that many NP-hard problems have natural IP formulations whose objectives are bounded by the LP-relaxation of these formulations. However, the Achilles' heel of LP-rounding is that it requires solutions of LPs of possibly extreme scale. Despite decades of work on LP solvers, including impressive advances during the 1990s, commercial codes such Cplex or Gurobi may not be capable of handling problems of the required scale. In this work, we develop a large-scale approximate LP solver suitable for use in LP-rounding based approximation schemes. Our intuition is that in LP rounding, since we ultimately round the LP to obtain an approximate solution of the combinatorial problem, a crude solution of the LP may suffice. Hence, an approach that can find approximate solutions of large LPs quickly may be suitable,

even if they are inefficient for obtaining highly accurate solutions.

Contributions. We highlight three main technical contributions of this work. First, we build on the theory of approximation algorithms and propose a suitable notion of an approximate LP solution. We define $\hat{x}$ to be an (ϵ, δ) approximate LP solution it is ϵ away (in the infinity norm) from the feasible region of the LP and its objective is a relative factor of δ away from the optimal objective of the LP. We show that for a class of rounding algorithms, used in NP-hard problems like vertex cover, set cover, set packing, and multiway-cut, we can successfully round (ϵ, δ) approximate LP solutions. As a second contribution, we show that one can approximately solve large LPs using a convex quadratic programming (QP) approximation that can be solved efficiently using asynchronous parallel algorithms [70, 59]. The convergence analysis in [89] precisely quantifies the relationship between the QP approximation and the LP. As a third contribution, we derive bounds on runtime as well as worst-case approximation ratios for our rounding schemes. Our experiments demonstrate that on three different classes of combinatorial problems, our approach, called Thetis, can outperform Cplex (a state-of-the-art commercial LP and IP solver) by up to an order of magnitude in runtime, while achieving comparable solution quality.

Related Work. The use of approximate LP solutions for the purposes of LP-rounding is a relatively recent advancement. We identify two works that are most similar to ours. The first is in the computation of the maximum a posteriori probability (MAP) estimate; a task that commonly arises in the field of Bayesian statistics. MAP estimation problems can solved using the LP-relaxation of a combinatorial

problem [85, 79]. Ravikumar et. al [79] proposed approximate LP-rounding schemes for iterative LP solvers to facilitate MAP inference in graphical models. In contrast, we propose to use stochastic descent methods to solve a QP relaxation of the LP; this allows us to take advantage of recent results on asynchronous parallel optimization methods [70, 59]. The second related work is by Manshadi et al. [62], which focuses on parallel LP-rounding based algorithms for packing and covering problems. The main difference between their approach and ours is that, our results apply to more general LP-relaxations, including partitioning problems like multiway-cut. Additionally, the runtime of our algorithm is less sensitive to the approximation error in the LP. For an error ε, the bound on runtime of the algorithm in [62] grows as ε^{-5}, while the bound on our algorithm's runtime grows as ε^{-2}.

Outline. This rest of this chapter is organized as follows. In Section 5.1, we review the theory of LP-rounding based approximation algorithms for NP-hard combinatorial problems. In Section 5.2, we discuss how one can perform LP-rounding with approximate LP solutions. In Section 5.3, we discuss how these approximate LP solutions can be computed using a QP formulation. In Section 5.4, we discuss implementation details of our QP solver. Our numerical experiments, in Section 5.5 test the effectiveness of our approximate LP-rounding schemes on a range of problems of practical interest. We conclude with remarks in Section 5.7.

5.1 Background

For a minimization problem Φ, an algorithm ALG is an α-factor approximation for Φ, for some $\alpha > 1$, if any solution obtained by ALG has a cost at most α times the value of an optimal (lowest cost) solution. The value of α may depend on the input size of the problem. For some problems, like *vertex cover*, there is a constant factor ($\alpha = 2$) approximation scheme [101] while for others, like *set cover* it can be as large as a $O(\log N)$ factor approximation [92] where N is the number of sets.

An LP-rounding based approximation scheme for Φ consists of three main steps. First, we *construct* a binary IP formulation of Φ which we denote as "P". This step is typically easy to do but P is as hard to solve as the original problem Φ. Second, we *relax* the constraints in P to an LP by replacing the binary variables with continuous variables that are restricted to lie in the interval $[0, 1]$. We then *solve* $LP(P)$. The relaxed formulation is LP-relaxation of P. Finally, we *round* the solution of $LP(P)$ to obtain feasible solution to P. The solution constructed after the *round* phase is a feasible integral solution for the original IP formulation and hence an upper bound on the cost of an optimal solution Φ. The optimal objective of the LP-relaxation forms a natural lower bound on the optimal cost of Φ. Any feasible solution of the $LP(P)$ is called a *fractional solution* of the set cover problem. The approximation guarantee is established by comparing the cost of a fractional solution and the cost of a feasible integral solution. We formalize this discussion by introducing the notion of an *integrality gap* of $LP(P)$.

Consider a in instance I of the IP formulation P for the minimization problem Φ. Let $OPT_f(I)$ denote the cost of an optimal fractional solution of instance I of $LP(P)$.

Let $OPT(I)$ denote the cost of an optimal integral solution of the same instance I of P. For any given minimization problem Φ, we define the **integrality gap**, as:

$$\rho := \sup_{I} \frac{OPT(I)}{OPT_f(I)}.$$

as the worst case ratio (for any instance I of the formulation P) between the cost of an optimal solution of $LP(P)$ and an optimal integral solution of P. The above definition suggests that there exists an instance I of Φ such that $OPT(I)$ is at least a factor of ρ away from $OPT_f(I)$. Hence, any scheme that uses $OPT_f(I)$ as a lower bound for $OPT(I)$ cannot guarantee an approximation factor better than ρ. In problems like vertex cover, there exists rounding schemes that achieve approximation factors equal to the integrally gap i.e $\alpha = \rho$.

Problem family	**Approx factor**	**Applications**
Set Covering	f [47]	Classification [13], Multi-object tracking [112]
Set Packing	$es + o(s)$ [8].	MAP-inference [85], Natural language [54]
Multiway-cut	$3/2 - 1/k$ [18].	Computer vision [16], Entity resolution [57].
Graphical Models	Heuristic	Semantic role labeling [84], Clustering [99]

Table 5.1: LP-rounding schemes considered in this chapter. The parameter f refers to the frequency of the most frequent element; s refers to s-column sparse matrices; and k refers to the number of terminals. e refers to the Euluer's constant.

We now review the *construct*,*relax/solve*, and *round* phases of LP-rounding schemes for three combinatorial problems- set covering, set packing, and multiway-cuts. Table 5.1 provides an overview of the approximation factors for the best known LP-rounding schemes along with practical applications for each of these problems.

Set Cover

Given a universe U of n elements, a collection of m subsets $S = \{S_1, \dots S_m\}$, a nonnegative cost function $c : S \to \mathbb{R}_+$. The set cover problem is the search for the lowest cost sub-collection of S that covers the entire universe U. We denote f to be the maximum number of sets in S that any given element belongs to.

In the *construct* phase, we formulate set cover as an IP by introducing a set of binary variables $x_s \in [0, 1], \forall s \in S$. The binary variable is set to 1 if the set $s \in S$ is selected. An IP formulation for the minimum cost set cover problem is:

$$\begin{aligned} & \min \sum_{s \in S} c_s x_s && && (5.1) \\ \text{subject to} \quad & \sum_{s \in S : e \in s} x_s \geqslant 1 && \forall e \in U, \\ & x_s \in \{0, 1\} && \forall s \in S \end{aligned}$$

In the *relax/solve* phase, we relax the integrality conditions on the variables $x_s, \forall s \in S$ in (5.1) to the following:

$$\begin{aligned} & \min \sum_{s \in S} c_s x_s && && (5.2) \\ \text{subject to} \quad & \sum_{s \in S : e \in s} x_s \geqslant 1 && \forall e \in U, \\ & x_s \in [0, 1] && \forall s \in S \end{aligned}$$

In the *round* phase, we generate a valid set cover by simply choosing the set $s \in S$

whose fractional solution $x_s^* \geqslant \frac{1}{f}$. It is easy to see that the set cover generated by such a rounding scheme is of cost no more than f times the cost of the fractional solution. If the fractional solution chosen for rounding is an optimal solution of (5.2), then we arrive at a f-factor approximation scheme for set cover. Algorithm 5 details the f-factor LP-rounding scheme for the set cover problem [47]. Vazirani [101] provides an instance of set cover with an integrality gap of f which shows that Algorithm 5 is tight. A special case of the set cover problem with $f = 2$ is the vertex cover problem.

Algorithm 5: An f-factor LP-rounding based approximation scheme for set cover

Data: A universe U of n elements, a collection of m subsets $S = \{S_1, \ldots S_m\}$, a nonnegative cost function $c : S \to \mathbb{R}_+$.
Result: A valid set cover C whose cost is no more than f times the cost of the minimum cost set cover.

```
Find x*, an optimal LP solution to (5.2).
C = φ
for s ∈ S do
    if x*_s ⩾ 1/f then
        C ← C ∪ s
return C
```

Set Packing

Given a universe U of n elements, a collection of m subsets $S = \{S_1, \ldots S_m\}$, a nonnegative cost function $c : S \to \mathbb{R}_+$. The set packing problem is the search for the maximum cost collection of mutually disjoint sets. Let A denote a matrix whose entries $A_{is} = 1$ if the element $i \in U$ is present in set $s \in S$. We denote x_s as a binary variable that is set to 1 if the set $s \in S$ is chosen in the packing. The set packing

problem can be formulated as:

$$\begin{aligned} & \max \sum_{s \in S} c_s x_s && (5.3) \\ \text{subject to} \quad & \sum_{s \in S} A_{is} x_s \leqslant 1 \quad && \forall i \in U \\ & x_s \in \{0,1\} && \forall s \in S \end{aligned}$$

In its full generality, the set packing problem can be approximated to within a factor of n but is NP-hard to approximate within a factor of $n^{1-\epsilon}$ [45]. However, a commonly considered case for set packing is the *k-column sparse* set packing problem in which every element is present in no more than k sets. Bansal et al. [8] proposed an $ek + o(k)$ factor approximation of the k-column sparse set packing problem based on the following IP formulation which has stronger LP-relaxation than (5.3):

$$\begin{aligned} & \max \sum_{s \in S} c_s x_s && (5.4) \\ \text{subject to} \quad & \sum_{s \in S} A_{is} x_s \leqslant 1 \quad && \forall i \in U, \\ & \sum_{s \in B(i)} x_s \leqslant 1 \quad && \forall i \in U, \\ & x_s \in \{0,1\} && \forall s \in S. \end{aligned}$$

Here, $B(i) := \{s \in S \mid A_{is} > \frac{1}{2}\}$ denotes the set of items marked as "big" because they use up more than half of the set $s \in S$. The constraints $\sum_{s \in B(i)} x_s \leqslant 1$, $\forall i \in U$

ensure that there can be no more than two big sets chosen for a given element $i \in U$. There is a $ek + o(k)$ factor randomized rounding algorithm (see Algorithm 6) due to [8]. The integrality gap of (5.4) is atleast $2k - 1$ [8]. The maximum independent set problem, is a special case of set packing where the k-column sparsity refers to the maximum degree of the graph.

Algorithm 6: A $ek + o(k)$-factor randomized LP-rounding algorithm for set packing

Data: A universe U of n elements, a collection of m subsets $S = \{S_1, \ldots S_m\}$, a nonnegative cost function $c : S \to \mathbb{R}_+$.

Result: A valid packing C whose cost is no less than $ek + o(k)$- times the cost of the maximum cost set packing.

1 Compute the optimal solution x^* of the LP-relaxation of (5.4).

2 Chose set $s \in \mathcal{S}$ with probability $\frac{x^*_s}{k\theta}$. Let $\mathcal{C} \subseteq \mathcal{S}$ denote the chosen sets.

3 For each set $s \in \mathcal{C}$ and element $a \in U$, let $E_{a,s}$ denote the event that the sets $\{s_2 \in \mathcal{C} : w_{a,s_2} > w_{a,s}\}$ have a total weight (with respect to element a) exceeding 1. Mark s for deletion if $E_{a,s}$ occurs for any $a \in s$.

4 Delete all sets from $s \in \mathcal{C}$ that are marked for deletion.

Multiway-Cuts

Let $G(V, E)$ denote a graph where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. Given a subset of vertices $V_1, V_2, \ldots V_k$ called terminals, a k-way cut is a set of edges that disconnects all terminals. Given a nonnegative cost function $c : E \to \mathbb{R}_+$, a k-way cut of minimum cost is the solution to the multiway-cut problem. Călinescu

et al. [18] formulated the multiway-cut problem as:

$$\begin{aligned} \min \frac{1}{2} \sum_{u,v \in E} c_{u,v} \|x_u - x_v\|_1 & \\ \text{subject to} \quad x_v \in \Delta_k \quad & \forall v \in V, \\ x_v \in \{0,1\}^k \quad & \forall v \in V \end{aligned} \tag{5.5}$$

where $\Delta_k := \{x \in \mathbb{R}^k : \sum_{i=1}^{k} x_i = 1,\ x \geqslant 0\}$ is the set of simplex constraints in k dimensions. Although it might appear that the formulation in (5.5) is non-linear, one can easily linearize (5.5) as follows:

$$\begin{aligned} \min \frac{1}{2} \sum_{i=1}^{k} \sum_{u,v} c_{u,v} x^i_{u,v} & \\ \text{subject to} \quad x_v \in \Delta_k \quad & \forall v \in V, \\ x^i_{u,v} \geqslant x^i_v - x^i_u \quad & \forall u, v \in E, i \in \{1 \dots k\}, \\ x^i_{u,v} \geqslant x^i_u - x^i_v \quad & \forall u, v \in E, i \in \{1 \dots k\}, \\ x^i_{u,v} \in [0,1] \quad & \forall u, v \in E, i \in \{1 \dots k\}, \\ x_v \in \{0,1\}^k \quad & \forall v \in V. \end{aligned} \tag{5.6}$$

Algorithm 7 is a $3/2 - 1/k$ factor approximation for multiway-cut problem [18]. Freund and Karloff [34] provide a family of instances whose integrality gap is $8/(7 + \frac{1}{k-1})$ which is the best known lower bound on the integrality gap of the LP-relaxation of (5.6).

Algorithm 7: An $3/2 - 1/k$-factor randomized LP-rounding algorithm for multi-way cut.

Data: A graph $G(V, E)$, a subset of vertices $V_1, V_2, \ldots V_k$ and a nonnegative cost function $c : E \to \mathbb{R}_+$

Result: A minimum cost k-way cut.

1 Compute an optimal solution x of the LP-relaxation of (5.6).
2 $E_i \leftarrow \{(u, v) \in E : x^i_u \neq x^i_v\}\ \forall i \in \{1 \ldots k\}$.
3 $W_i \leftarrow \sum_{u,v \in E_i} c_{u,v} (\sum_{j=1}^{k} x^i_{u,v})\ \forall i \in \{1 \ldots k\}$.
4 Renumber the terminals so that W_k is the largest among $W_1 \ldots W_k$.
5 Pick $\rho \in (0, 1)$ uniformly at random and
$\sigma \in \{(1, 2 \ldots k-1, k), (k-1, k-2 \ldots 1, k)$
6 For $i = 1 \ldots k-1$: $V_{\sigma(i)} \leftarrow B(s_i, \rho) - \bigcup_{j \leqslant i} V_{\sigma(j)}$ where
$B(s_i, \rho) := \{v \in V : x^i_v \geqslant \rho\}$.
7 $V_k \leftarrow V - \cup_{i<k} V_i$.
8 Let C be the set of edges that run between the partitions $V_1 \ldots V_k$.
9 **return** C

5.2 Approximate LP rounding

In this section, we describe how approximate LP solutions can be used in LP-rounding schemes We use the *vertex cover* problem as a running example, for it is the simplest non-trivial setting that exposes the main ideas of this work.

Let $G(V, E)$ denote a graph with vertex set V undirected edges $E \subseteq (V \times V)$. Let c_v denote a nonnegative cost associated with each vertex $v \in V$. A vertex cover of a graph is a subset of V such that each edge $e \in E$ is incident to at least one vertex in this set. The *minimum-cost* vertex cover is the one that minimizes the sum of terms c_v (summed over the vertices v) belonging to the cover. The vertex cover problem

is a special case of the set cover problem and can be formulated as:

$$\min_{x} \sum_{v \in V} c_v x_v \quad \text{subject to} \quad x_u + x_v \geqslant 1 \text{ for } (u,v) \in E \text{ and } x_v \in \{0,1\} \text{ for } v \in V. \tag{5.7}$$

where $x_v \in [0,1], \forall v \in V$ is a binary variable set to 1 if the vertex $v \in V$ is selected in the vertex cover and 0 otherwise. The LP-relaxation of (5.7) is:

$$\min_{x} \sum_{v \in V} c_v x_v \quad \text{subject to} \quad x_u + x_v \geqslant 1 \text{ for } (u,v) \in E \text{ and } x_v \in [0,1] \text{ for } v \in V. \tag{5.8}$$

Algorithm 5 is a 2-factor approximation algorithm for vertex cover. We note here an important property: The rounding algorithm can generate feasible integral solutions while being *oblivious* of whether the fractional solution is the optimal solution of (5.8). We formally define the notion of an *oblivious rounding scheme* as follows.

Definition 5.1. *For a minimization problem Φ with an IP formulation P whose LP relaxation is denoted by LP(P), a γ-factor 'oblivious' rounding scheme converts any feasible point $x_f \in$ LP(P) to an integral solution $x_I \in P$ with cost at most γ times the cost of LP(P) at x_f.*

Given a γ-factor *oblivious algorithm* ALG for Φ, one can construct a γ-factor approximation algorithm for Φ by using ALG to round an *optimal* fractional solution of LP(P). When we have an approximate solution for LP(P) that is feasible for this problem, rounding can produce an α-factor approximation algorithm for Φ for a factor α slightly larger than γ, where the difference between α and γ takes ac-

count of the inexactness in the approximate solution of LP(P). Many LP-rounding schemes, including the ones in Table 5.1, are oblivious.

Approximate Solutions of LPs. Consider the LP in the following standard form:

$$\min\ c^T x \quad \text{subject to}\ \ Ax = b,\ \ x \geqslant 0, \tag{5.9}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m\times n}$ and its corresponding dual

$$\max\ b^T u \quad \text{subject to}\ \ c - A^T u \geqslant 0. \tag{5.10}$$

Let x^* denote an optimal primal solution of (5.9). Let $\hat{x}$ denote an approximate LP solution that may be infeasible and have objective value different from the optimum $c^T x^*$. sub-optimality of $\hat{x}$.

Definition 5.2. *A point $\hat{x}$ is an (ϵ, δ)-approximate solution of the LP (5.9) if $\hat{x} \geqslant 0$ and there exists constants $\epsilon > 0$ and $\delta > 0$ such that*

$$\|A\hat{x} - b\|_\infty \leqslant \epsilon \quad \textit{and} \quad |c^T\hat{x} - c^T x^*| \leqslant \delta |c^T x^*|$$

.

Definition 5.1 refers to approximate LP solutions that are ϵ away from a feasible solution in terms of the infinity norm and a factor of δ away from the optimal objective in relative terms. We show in Lemma 5.3 that a γ-factor oblivious rounding scheme can round a $(0, \delta)$ approximate LP solution to produce a feasible integral

solution whose cost is no more than $\gamma(1+\delta)$ times an optimal solution of the P. The factor γ comes from nature of the rounding algorithm and the extra δ factor arises because the rounding algorithm does not have access to an optimal fractional solution.

Lemma 5.3. *Given an γ-factor oblivious rounding algorithm ALG for a minimization problem Φ based on a $(0,\delta)$ approximate solution of $\mathsf{LP}(\mathsf{P})$, we can construct a $\gamma(1+\delta)$ approximation algorithm for Φ.*

Proof. Let $\hat{x}$ denote a $(0,\delta)$ approximate LP solution of $\mathsf{LP}(\mathsf{P})$ of cost $\mathrm{OPT}(\hat{x})$. Let C denote the cost of the feasible integral solution produced by rounding $\hat{x}$ using ALG. Since ALG is a γ-factor oblivious algorithm, we have that

$$C \leqslant \gamma \mathrm{OPT}(\hat{x}).$$

From the δ sub-optimality of $\hat{x}$, we have that

$$\mathrm{OPT}(\hat{x}) \leqslant (1+\delta)\mathrm{OPT}.$$

Hence, the result $C \leqslant (1+\delta)\mathrm{OPT}$ follows. □

To cope with the infeasibility, we convert an (ϵ,δ) approximate solution to a $(0,\hat{\delta})$ approximate solution where $\hat{\delta}$ is not too large. For the vertex cover problem OPT(5.1), Sridhar et al. [89] shows the following result. (Here, $\Pi_{[0,1]^n}(\cdot)$ denotes projection onto the unit hypercube in $\mathbb{R}^n$.)

Lemma 5.4. *Let $\hat{x}$ be an (ε, δ) approximate solution to the linear program (5.8) with $\varepsilon \in [0, 1)$. Then, $\tilde{x} = \Pi_{[0,1]^n}((1-\varepsilon)^{-1}\hat{x})$ is a $(0, \delta(1-\varepsilon)^{-1})$-approximate solution.*

Since $\tilde{x}$ is a feasible solution for (5.2), the oblivious rounding scheme in Section 5.1 results in an $2(1+\delta)(1-\epsilon)^{-1}$ factor approximation algorithm. In general, constructing $(0, \hat{\delta})$ from (ϵ, δ) approximate solutions requires reasoning about the structure of a particular LP. Sridhar et al. [89] prove statements analogous to Lemma 5.4 for set-packing, set-covering, and multiway-cut problems. We omit the details of these results.

5.3 Computing Approximate Solutions to LPs

Consider the following *regularized quadratic penalty* approximate formulation of the LP defined in (5.9), parameterized by a positive constant β, whose solution is denoted by $x(\beta)$:

$$x(\beta) := \arg\min_{x \geqslant 0} f_\beta(x) := c^T x - \bar{u}^T(Ax - b) + \frac{\beta}{2}\|Ax - b\|^2 + \frac{1}{2\beta}\|x - \bar{x}\|^2, \quad (5.11)$$

where $\bar{u} \in \mathbb{R}^m$ and $\bar{x} \in \mathbb{R}^n$ are arbitrary vectors. In practice, $\bar{u}$ and $\bar{x}$ may be set to zero, or chosen to be approximations to the dual and primal solutions of (5.9), or generated as a part of an overall scheme like the augmented Lagrangian method. The results in [89] show that an exact solution to the QP approximation defined in (5.11) is an approximate solution of (5.9) in accordance with Definition 5.2. In practice, since we cannot solve (5.11) exactly, it is also the case that an approximate solution of (5.11) can still serve as an approximate solution of (5.9).

Perturbation Results

We now discuss the relationship between the unique solution $x(\beta)$ of the QP in (5.11) and an optimal solution of the LP in (5.9). The quality of the approximation in (5.11) depends strongly on the *conditioning* of underlying LP (5.9), a concept that was studied by Renegar [80]. Denoting the data for problem (5.9) by $d := (A, b, c)$, we consider first $\Delta d := (\Delta A, \Delta b, \Delta c)$ of smallest norm such that the linear program defined by $d + \Delta d$ is primal infeasible. The primal condition number δ_P is the infimum of the ratios $\|\Delta d\|/\|d\|$ taken over all such vectors Δd. The dual condition number δ_D is defined analogously. Clearly both δ_P and δ_D are in the range $[0, 1]$; smaller values indicate poorer conditioning. Sridhar et al. [89] prove the following result.

Theorem 5.5 (Theorem 4 of [89])**.** *Suppose that δ_P and δ_D are both positive, and let (x^*, u^*) be any primal-dual solution pair for (5.9), (5.10). If we define $C_* := \max(\|x^* - \bar{x}\|, \|u^* - \bar{u}\|)$, then the unique solution $x(\beta)$ of (5.11) satisfies*

$$\|Ax(\beta) - b\| \leqslant (1/\beta)(1 + \sqrt{2})C_*, \quad \|x(\beta) - x^*\| \leqslant \sqrt{6}C_*.$$

If in addition we have

$$\beta \geqslant \frac{10C_*}{\|d\| \min(\delta_P, \delta_D)},$$

then we have

$$|c^T x^* - c^T x(\beta)| \leqslant \frac{1}{\beta}\left[\frac{25C_*}{2\delta_P\delta_D} + 6C_*^2 + \sqrt{6}\|\bar{x}\|C_*\right].$$

If β is large enough, the unique optimal solution $x(\beta)$ of the QP (5.11) is an (ϵ, δ)

approximate solution for the LP (5.9) where

$$\epsilon = \frac{(1+\sqrt{2})C_*}{\beta} \text{ and } \delta = \frac{1}{\beta}\left[\frac{25C_*}{2\delta_P\delta_D} + 6C_*^2 + \sqrt{6}\|\bar{x}\|C_*\right].$$

In practice, we solve (5.11) approximately, using an algorithm whose complexity depends on the threshold $\bar{\epsilon}$ for which the objective is accurate to within $\bar{\epsilon}$. That is, we seek $\hat{x}$ such that

$$f_\beta(\hat{x}) - f_\beta(x(\beta)) \leqslant \bar{\epsilon}.$$

Since f_β is strongly convex with modulus β^{-1}, we have that

$$\beta^{-1}\|\hat{x} - x(\beta)\|^2 \leqslant f_\beta(\hat{x}) - f_\beta(x(\beta)) \leqslant \bar{\epsilon}.$$

Sridhar et al. [89] show that for an appropriate choice of $\bar{\epsilon}$, we obtain

$$|c^T\hat{x} - c^Tx^*| \leqslant \frac{1}{\beta}\left[\frac{25C_*}{\delta_P\delta_D} + 6C_*^2 + \sqrt{6}\|\bar{x}\|C_*\right], \quad \|A\hat{x} - b\| \leqslant \frac{1}{\beta}\left[(1+\sqrt{2})C_* + \frac{25C_*}{2\delta_P\delta_D}\right].$$

The following result from Sridhar et al. [89] is almost an immediate consequence.

Theorem 5.6 (Theorem 5 of [89])**.** *Suppose that δ_P and δ_D are both positive and let (x^*, u^*) be any primal-dual optimal pair. Suppose that C_* is defined as in Theorem 5.5. Then for any given positive pair (ϵ, δ), we have that $\hat{x}$ satisfies the inequalities in definition 5.2*

provided that β *satisfies the following three lower bounds:*

$$\begin{aligned}
\beta &\geqslant \frac{10C_*}{\|d\| \min(\delta_P, \delta_D)} \\
\beta &\geqslant \frac{1}{\delta |c^T x^*|} \left[\frac{25C_*}{\delta_P \delta_D} + 6C_*^2 + \sqrt{6}\|\bar{x}\| C_* \right] \\
\beta &\geqslant \frac{1}{\epsilon} \left[(1+\sqrt{2})C_* + \frac{25C_*}{2\delta_P \delta_D} \right].
\end{aligned}$$

In summary, the results from Sridhar et al. [89] suggest that if β is large enough, the QP approximation (5.11) can be used to compute (ϵ, δ) optimal solutions for (5.9) where $\epsilon \sim \beta^{-1}$ and $\delta \sim \beta^{-1}$ (ignoring the dependence on δ_P and δ_D). For general LPs of the form (5.9), both δ_D and δ_P can easily be 0. However, this is not the case for the combinatorial problems of interest in this work. Sridhar et al. [89] provide estimates of (δ_P, δ_D) in detail for vertex cover. For an instance of vertex cover with n nodes and m edges, they show that $\delta_P^{-1} \leqslant 2n^{-1/2}(2m+n)^{-1/2}$ and $\delta_D^{-1} = (2m+n)^{-1/2}$. Setting $x = \vec{1}$ and $u = \vec{0}$, gives that $C_* \leqslant \sqrt{m}$ for vertex-cover. Together, we get that $\beta = O(m^{3/2} n (\min\{\varepsilon, \delta |c^T x^*|\})^{-1})$. Similar results can be shown for set covering, set packing and multiway-cut problems.

Solving the QP Approximation

Recently, there has been a lot of interest in developing algorithms and software tailored at solving large QP problems of the form (5.11). Problem sizes in fields like machine learning, social-network analysis, and marketing are already in the giga-byte scale and are predicted to grow even more. High-quality commercial QP solvers, such as Cplex, Gurobi and Knitro are not known to scale well to large

Instance	PV (M)	NNZ (M)	Time (seconds)
frb59-26-1	0.12	0.88	1410.7
frb59-26-2	0.12	0.88	1371.7
frb59-26-3	0.12	0.88	-
Amazon	0.56	2.91	462.8
DBLP	0.52	2.78	427.7
Google+	0.77	5.06	-

Table 5.2: Summary of wall-clock time required by Cplex-QP to solve the QP-approximation (5.11) for the vertex cover problem. Cplex-QP is run with a time limit of one hour and parallelized over 32 cores, with '-' indicating that the code reached the time limit. PV is the number of primal variables while NNZ is the number of nonzeros in the hessian matrix (both in millions).

problems. We present evidence to support this claim in Table 5.2. We solved the QP-approximation (5.11) of the vertex cover problem with $\beta = 0.1$ using Cplex-QP (a state-of-the-art commercial QP solver). Section 5.5 provides details of the system setup and instances used to produce these results. Table 5.2 provides a summary of problem sizes and wall-clock time required to solve these instances. Our results suggest that commercial QP solvers do not scale well for problems of interest in this work.

We propose the use of the SCD to solve (5.11). Several constrained and unconstrained variants of this method have been proposed and analyzed under different conditions [69, 95, 11, 111]. During iteration j of SCD (see Algorithm 6), we choose a coordinate component $i \in \{1, 2, \ldots, n\}$ and take a step along the partial gradient of the coordinate component i of x_j to obtain the next iterate x_{j+1}. The update rule for SCD is given by:

$$[x_{j+1}]_{i(j)} \leftarrow \max\left(0, [x_j]_{i(j)} - (1/L_{max})[\nabla f_\beta(x_j)]_{i(j)}\right)$$

where $L_{max} := \beta(\max_{i=1,2,\ldots,n} A_{:i}^T A_{:i}) + \beta^{-1}$ and $A_{:i}$ denotes the ith column of A. The algorithm terminates when

$$\|x_{j+1} - \max(0, x_j - \nabla f_\beta(x_j))\| \leqslant \epsilon_t$$

where $\epsilon_t > 0$ is a threshold close to 0. The proof of convergence, in expectation, of this extremely simple and basic procedure is a special case of the theory in [59] and depends on the following constants:

$$l := \frac{1}{\beta}, \quad R := \sup_{j=1,2,\ldots} \|x_j - x(\beta)\|_2, \tag{5.12}$$

Here $\mathbb{E}(\cdot)$ denotes expectation over all the random variables $i(j)$ indicating the update indices chosen at each iteration.

Theorem 5.7. *For Algorithm 6 we have*

$$\mathbb{E}\|x_j - x(\beta)\|^2 + \frac{2}{L_{max}}\mathbb{E}(f_\beta(x_j) - f_\beta^*) \leqslant \left(1 - \frac{l}{n(l + L_{max})}\right)^j \left(R^2 + \frac{2}{L_{max}}(f_\beta(x_0) - f_\beta^*)\right),$$

where $f_\beta^* := f_\beta(x(\beta))$. *We obtain high-probability convergence of* $f_\beta(x_j)$ *to* f_β^* *in the following sense: For any* $\eta \in (0, 1)$ *and any small* $\bar{\epsilon}$, *we have*

$$P(f_\beta(x_j) - f_\beta^* < \bar{\epsilon}) \geqslant 1 - \eta$$

Algorithm 8: SCD method for (5.11)

Data: Initial point $x_0 \in \mathbb{R}^n$.
Result: Solution x^* for (5.11).

$j \leftarrow 0$
while *converged* **do**
  Choose $i(j) \in \{1, 2, \ldots, n\}$ randomly with equal probability
  Define x_{j+1} from x_j by setting
  $[x_{j+1}]_{i(j)} \leftarrow P_{\mathcal{C}}([x_j]_{i(j)} - (1/L_{\max})[\nabla f_\beta(x_j)]_{i(j)})$, leaving other components unchanged
  $j \leftarrow j + 1$
return x_j

provided that

$$j \geqslant \frac{n(l + L_{\max})}{l} \left| \log \frac{L_{\max}}{2\eta\bar{\epsilon}} \left(R^2 + \frac{2}{L_{\max}} (f_\beta(x_0) - f_\beta^*) \right) \right|$$

Worst-Case Complexity Bounds For Vertex Cover. We combine the analysis in this section to derive a worst-case complexity bound for our approximate LP solver. Supposing that the columns of A have norm $O(1)$, we have from (5.12) that $l = \beta^{-1}$ and $L_{\max} = O(\beta)$. Theorem 5.7 indicates that we require $O(n\beta^2)$ iterations to solve (5.11) (modulo a log term). For the vertex cover problem, we use the values to β described in Section 5.3 to provide a complexity estimate of $O((nm)^3/\hat{\epsilon}^2)$ (assuming $m > n$), where $\hat{\epsilon} = \min(\varepsilon, \delta c^T x^*)$. In order to obtain the desired accuracy in terms of feasibility and function value of the LP (captured by $\hat{\epsilon}$) we need to solve the QP to within a different, tighter tolerance, quantified by $\bar{\epsilon}$. Both tolerances are related to the choice of penalty parameter β in the QP. Ignoring here the dependence on dimensions, we note the relationships $\beta \sim \hat{\epsilon}^{-1}$ (from theorem 5.6) and $\bar{\epsilon} \sim$

$\beta^{-3} \sim \hat{\epsilon}^3$. Expressing all quantities in terms of $\hat{\epsilon}$, and using theorem 5.7 we see an iteration complexity of $\hat{\epsilon}^{-2}$ for SCD (ignoring log terms). The following lemma is an immediate consequence.

Lemma 5.8. *Given $\delta > 0$ and $\epsilon > 0$, there is a $O((nm)^3/\hat{\epsilon}^2)$ algorithm for a non-trivial instance of the vertex cover problem in* (5.7)*, with an approximation factor $2(1+\delta)(1+\epsilon)^{-1}$ where $\hat{\epsilon} = \min(\varepsilon, \delta c^T x^*)$ and x^* is an optimal solution of* (5.8)*.*

The linear convergence rate of SCD is instrumental in the result in Lemma 5.8. By contrast, standard variants of stochastic-gradient descent (SGD) applied to the QP yield poorer complexity. From Nemirovski et al. [68], it follows that for diminishing-step or constant-step variants of SGD, we see complexity of $\hat{\epsilon}^{-7}$, while for robust SGD, we see $\hat{\epsilon}^{-10}$. (Besides the inverse dependence on $\bar{\epsilon}$ or its square, there is a contribution of order $\hat{\epsilon}^{-2}$ from the conditioning of the QP).

5.4 Implementation Details

In section, we discuss two main enhancements that improve the efficiency of our LP-rounding scheme outlined in Section 5.2 to make it suitable for practical applications. The first is the use of an augmented Lagrangian framework rather than the *one-shot* approximation by the QP in (5.11) and the second is an asynchronous parallel implementation of Algorithm 6.

Augmented Lagrangian

The quadratic-penalty approach can be extended to an augmented Lagrangian approach, in which a sequence of problems of the form (5.11) are solved, with the estimates $\bar{x}$ and $\bar{u}$ of primal and dual solutions and possibly the penalty parameter β are updated between iterations. The advantage of such an approach is that (5.9) can be solved without increasing the penalty parameter β to a very large value, thereby avoiding some of the ill-conditioning observed with quadratic penalty methods. Such a "proximal method of multipliers" for LP was described in [110]. The algorithm generates a sequence $\{\bar{x}_k\}$ and $\{\bar{u}_k\}$ that converge to the optimal primal and dual solutions of the LP. We omit a discussion of the convergence properties of the algorithm here, but note that the quality of solution depends on the values of $\bar{x}$, $\bar{u}$ and β at the last iteration before convergence is declared. By applying Theorem 5.6, we note that the constant C_* is smaller when $\bar{x}$ and $\bar{u}$ are close to the primal and dual solution sets, thus improving the approximation and reducing the need to increase β to a larger value to obtain an approximate solution to the LP of acceptable quality for purposes of rounding.

We now detail the update rules for Augmented Lagrangian method used in this work. Consider the following LP

$$\min_{x \in \mathcal{C}} c^T x \quad \text{subject to } Ax = b \tag{5.13}$$

where $x \in \mathbb{R}^n$ and $\mathcal{C}$ is a polyhedral set. For LP-relaxations of the set-packing and set-covering problems, the set $\mathcal{C}$ is the nonnegative orthant. For the multiway-cut

problem, the set $\mathcal{C}$ may be the k-dimensional simplex Δ_k defined in (5.6). The Augmented Lagrangian method solves (5.13) by recasting it as a sequence of optimization problems of the form

$$x_{j+1} = \underset{x \in \mathcal{C}}{\operatorname{argmin}}\, G_{\beta_j}(x; u_j) := c^T x - u_j^T(Ax-b) + \frac{\beta_j}{2}\|Ax-b\|^2 + \frac{1}{2\beta_j}\|x-x_j\|^2, \quad (5.14)$$

followed by an update step on the dual multipliers given by

$$u_{j+1} = u_k + \beta_j(Ax_{j+1} - b). \quad (5.15)$$

In practice, the subproblems (5.14) are solved inexactly using tolerances that tighten as the algorithm proceeds. We define an inexact solution to the QP in (5.14) as one that satisfies $\|\nabla^c G_{\beta_j}(x^i; u_j)\| \leqslant \epsilon_i$ where

$$\nabla^c G_{\beta_j}(x; u_j) := P_{\mathcal{C}}(x - \nabla G_{\beta_j}(x_j; u_j)) - x_j. \quad (5.16)$$

When $\mathcal{C} = \{x \in \mathbb{R}^n : l \leqslant x \leqslant u\}$ is the set of 'box-constraints', we may also use

$$[\nabla^c G_{\beta_j}(x; u_j)]_i = \begin{cases} \min\left(0, \nabla G_{\beta_j}(x; u_j)\right) & [x]_i \geqslant l_i \\ \max\left(0, \nabla G_{\beta_j}(x; u_j)\right) & [x]_i \leqslant u_i \\ \nabla G_{\beta_j}(x; u_j) & l_i < [x]_i < u_i \end{cases} \qquad \forall i \in \{1 \dots n\}. \quad (5.17)$$

Both the termination criterion in (5.16) and (5.17) are motivated from the optimality conditions of (5.13). Our implementation is illustrated in Algorithm 9 (due to

Algorithm 9: Augmented Lagrangian method for (5.9)

Data: Initial point $x_0 \in \mathbb{R}^n$, $u_o \in \mathbb{R}^m$. Parameters $\mu \in (0,1)$, $\nu \in (0,1)$, $\sigma_0 \geqslant 0$, $\sigma_{red} \in (0,1)$, $\beta_0 \geqslant 0$, $\epsilon_0 \geqslant 0$, $\epsilon_{tol} > 0$, $\beta_{max} > 0$

Result: Solution x^* for (5.9)

$j \leftarrow 0$

while $\|Ax_j - b\|_\infty \leqslant \epsilon_{tol}$ *and* $u_j^T(Ax_j - b) \leqslant \epsilon_{tol}$ **do**

  $\epsilon_j \leftarrow \sigma_j \|Ax_j - b\|$

  $x_{j+1} \leftarrow \underset{x \in \mathcal{C}}{\operatorname{argmin}}\ G_{\beta_j}(x; u_j)$ (solved to tolerance ϵ_j)

  $u_{j+1} \leftarrow u_j + \beta_j(Ax_{j+1} - b)$

  if $\epsilon_j \leftarrow \sigma_j \|Ax_j - b\|$ **then**

    $\sigma_{j+1} \leftarrow \sigma_{red}\sigma_j$

  else

    $\sigma_{j+1} \leftarrow \sigma_j$

  $j \leftarrow j + 1$

  if $\|u_{j+1} - u_j\|_\infty > \mu\|u_j - u_{j-1}\|_\infty$ *or* $\|u_{j+1} - u_j\|_\infty < \epsilon_j$ **then**

    $\beta_{j+1} = \min(\beta/\nu, \beta_{max})$

  else

    $\beta_{j+1} \leftarrow \beta_j$

  $j \leftarrow j + 1$

return x_j

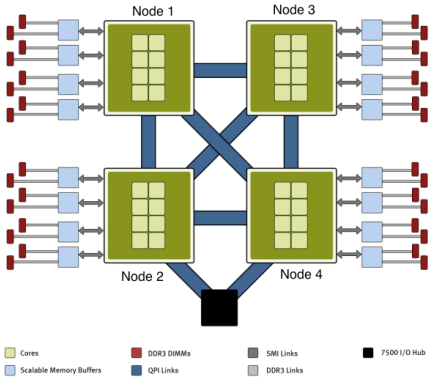
Eckstein and Silva [29]).

Asynchronous Stochastic-Coordinate Descent (ASCD)

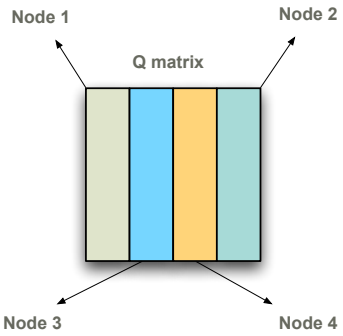
The computational bottleneck in Algorithm 9 is the minimization step (5.14). Every other step consists of simple matrix-vector operations that can easily be executed in parallel. Recent work on asynchronous parallel versions of Algorithm 6 [70, 59] make the entire LP-rounding scheme discussed in this work suitable for execution on multi core, shared-memory architectures. We use an asynchronous variant of Algorithm 6 (due on [59]). In this scheme, each core performs an update step, on a

Figure 5.1: Asynchronous parallel implementation of algorithm 6.

(a) Intel E7-4450 architecture with 40 cores and 4 nodes.



(b) Sharding the hessian matrix of (5.14).



single coordinate, of the vector x that is centrally stored. The update steps across multiple cores are performed in an asynchronous manner. Each thread essentially runs its own version of Algorithm 6 independently of the others, choosing its update component $i(j)$ and updating this component of the shared x. By the time the update is performed, x will have been updated by several other threads. Provided that the number of threads is not too large (according to a certain threshold that depends on n (size of the vector x) and on the diagonal dominance properties of the Hessian matrix) the convergence rate is similar to the serial case, and almost-linear speedup is seen as the number of threads is increased. The algorithm terminates when (5.16) or (5.17) is satisfies. We check for termination every $50n$ iterations.

Our implementation, based on [59], is optimized for the architecture of a 40-core Intel Xeon E7-4450 which is used in our numerical experiments. Each thread is assigned to a single core in our 40-core Intel Xeon architecture. The 40 cores on the Xeon architecture are arranged into four *nodes* (also known as sockets). As

Figure 5.1 illustrates, each node has 10 cores and its own memory. Intel follows a non-uniform memory access (NUMA) design paradigm where each core can access its own local memory faster than the local memory of a core in another node. To exploit this, we *shard* the hessian matrix $Q := A^T A$ of (5.14) into equal slices that are distributed across nodes. The sharding is illustrated in Figure 5.1. We use a *sampling without replacement* strategy where each coordinate, when updated, is not updated again until all other coordinates have already been updated. Further more, each core only updates coordinates of x whose corresponding shard is stored in its own local memory. The order in which the coordinates are updated are shuffled within each shard at the end of each pass over the entire set of coordinates.

5.5 Experiments

Our experiments address the following questions: (1) How does our approach compare to a state-of-the-art commercial solver? (2) Does not knowing the exact LP solution effect the quality of rounded solutions in practice? and (3) Is our approximate LP-rounding scheme useful in graph analysis tasks that arise in practical applications? We give favorable answers to all three questions.

Comparisons with Cplex

We conducted numerical experiments on three different combinatorial problems that commonly arise in graph analysis tasks in machine learning: vertex cover, independent set, and multiway cuts. For each problem, we compared the perfor-

mance of our LP solver against the LP and IP solvers of Cplex (v12.5) (denoted as Cplex-LP and Cplex-IP respectively). The two main goals of this experiment are: (1) to compare the quality of the integral solutions obtained using our approach with the integral solutions from Cplex-IP and (2) compare wall-clock times required by our Thetis and Cplex-LP to solve the LPs constructed in the LP-rounding schemes.

Datasets. Our tasks are based on two families of graphs. The first family of instances (*frb59-26-1* to *frb59-26-3*) was obtained from Bhoslib[1] (Benchmark with Hidden Optimum Solutions); they are considered difficult problems [113]. The second family of instances are social networking graphs obtained from the Stanford Network Analysis Platform (SNAP)[2].

System Setup. Thetis was implemented using a combination of C++ (for Algorithm 6) and Matlab (for the augmented Lagrangian framework). Our implementation of the augmented Lagrangian framework was based on [29]. All experiments were run on a 4 Intel Xeon E7-4450 (40 cores @ 2Ghz) with 256GB of RAM running Linux 3.8.4 with a 15-disk RAID0. Cplex uses 32 (of the 40) cores available in the machine, and for consistency, our implementation was also restricted to 32 cores. Cplex implements presolve procedures that detect redundancy, and substitute and eliminate variables to obtain equivalent, smaller LPs. Since the aim of this experiment is compare the algorithms used to solve LPs, we ran both Cplex-LP and Thetis on the reduced LPs generated by the presolve procedure of Cplex-LP. We use the barrier optimizer option in Cplex-LP. Both Cplex-LP and Thetis were run to

[1] `http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm`
[2] `http://snap.stanford.edu/`

a tolerance of $\epsilon = 0.1$. Additional experiments with Cplex-LP run with its default tolerance options are reported in Table 5.5). All algorithms were provided with a time limit of 3600 seconds excluding the time taken for preprocessing as well as the rounding algorithms used to generate integral solutions from fractional solutions.

Results. We solved the *vertex cover problem* using the approximation algorithm described in section 5.2. We solve the *maximum independent set* problem using a variant of the $es + o(s)$-factor approximation of [8] where s is the maximum degree of a node in the graph. For the multiway-cut problem (with $k = 3$) we use the $3/2 - 1/k$-factor approximation algorithm described in [101]. The details of the transformation from approximate infeasible solutions to feasible solutions are provided in Sridhar et al. [89]. Since the rounding schemes for maximum-independent set and multiway-cut are randomized, we chose the best feasible integral solution from 10 repetitions. The results are summarized in Table 5.3, with additional details in Table 5.4. We now discuss, in detail, the results for each of the problems; vertex cover, independent set, and multiway cut separately.

Vertex Cover. On the Bhoslib instances (*frb59-26-1* to *frb59-26-3*), Cplex-IP produced integral solutions that were within 1% of the documented optimal solutions, but required an hour for each of the instances. In comparison, the integral solutions from Thetis were within 4% of the documented optimal solutions. Although the LP solutions obtained by Thetis were less accurate than those obtained by Cplex-LP, the rounded solutions from Thetis and Cplex-LP are almost exactly the same. In summary, the LP-rounding approaches using Thetis and Cplex-LP obtain integral

Instance	Minimization problems								Maximization problems			
	VC				MC				MIS			
	PV	NNZ	S	Q	PV	NNZ	S	Q	PV	NNZ	S	Q
frb59-26-1	0.12	0.37	2.8	1.04	0.75	3.02	53.3	1.01	0.12	0.38	5.3	0.36
frb59-26-2	0.12	0.37	4.6	1.04	0.75	3.02	52.0	1.00	0.12	0.38	5.4	0.34
frb59-26-3	0.12	0.37	4.9	1.04	0.75	3.02	53.2	1.00	0.12	0.38	4.0	0.37
Amazon	0.39	1.17	8.4	1.23	5.89	23.2	-	0.42	0.39	1.17	7.4	0.82
DBLP	0.37	1.13	8.3	1.25	6.61	26.1	-	0.33	0.37	1.13	8.5	0.88
Google+	0.71	2.14	9.0	1.21	9.24	36.8	-	0.83	0.71	2.14	10.2	0.82

Table 5.3: Summary of wall-clock speedup (in comparison with Cplex-LP) and solution quality (in comparison with Cplex-IP) of Thetis for the LP relaxations of three graph analysis problems. Each code is run with a time limit of one hour and parallelized over 32 cores, with '-' indicating that the code reached the time limit. PV is the number of primal variables while NNZ is the number of nonzeros in the constraint matrix of the LP in standard form (both in millions). S is the speedup, defined as the time taken by Cplex-LP divided by the time taken by Thetis. Q is the ratio of the solution objective obtained by Thetis to that reported by Cplex-IP. For minimization problems (**VC** and **MC**) lower Q is better; for maximization problems (**MIS**) higher Q is better. For MC, a value of $Q < 1$ indicates that Thetis found a better solution than Cplex-IP found within the time limit.

solutions of comparable quality with Cplex-IP — but Thetis is three times faster than Cplex-LP. We observed a similar trend on the large social networking graphs. We were able to recover integral solutions of comparable quality to Cplex-IP, but seven to eight times faster that using LP-rounding with Cplex-LP. We make two additional observations. First, we observed a smaller gap (compared to the Bhoslib instances) between the optimal LP and optimal IP solutions. Second, we recorded unpredictable performance of Cplex-IP on large instances. Notably, Cplex-IP was able to find the optimal solution for the *Amazon* and *DBLP* instances, but timed out on *Google+*, which is of comparable size. On some instances, Cplex-IP outperformed even Cplex-LP in wall clock time, due to specialized presolve strategies.

VC	Cplex IP			Cplex LP			Thetis		
(min)	t (secs)	BFS	Gap(%)	t (secs)	LP	RSol	t (secs)	LP	RSol
frb59-26-1	-	1475	0.7	2.48	767.0	1534	0.88	959.7	1532
frb59-26-2	-	1475	0.6	3.93	767.0	1534	0.86	979.7	1532
frb59-26-3	-	1475	0.5	4.42	767.0	1534	0.89	982.9	1533
Amazon	85.5	1.60×10^5	-	24.8	1.50×10^5	2.04×10^5	2.97	1.50×10^5	1.97×10^5
DBLP	22.1	1.65×10^5	-	22.3	1.42×10^5	2.08×10^5	2.70	1.42×10^5	2.06×10^5
Google+	-	1.06×10^5	0.01	40.1	1.00×10^5	1.31×10^5	4.47	1.00×10^5	1.27×10^5
MC	Cplex IP			Cplex LP			Thetis		
(min)	t (secs)	BFS	Gap(%)	t (secs)	LP	RSol	t (secs)	LP	RSol
frb59-26-1	72.3	346	-	312.2	346	346	5.86	352.3	349
frb59-26-2	561.1	254	-	302.9	254	254	5.82	262.3	254
frb59-26-3	27.7	367	-	311.6	367	367	5.86	387.7	367
Amazon	-	12	NA	-	-	-	55.8	7.3	5
DBLP	-	15	NA	-	-	-	63.8	11.7	5
Google+	-	6	NA	-	-	-	109.9	5.8	5
MIS	Cplex IP			Cplex LP			Thetis		
(max)	t (secs)	BFS	Gap(%)	t (secs)	LP	RSol	t (secs)	LP	RSol
frb59-26-1	-	50	18.0	4.65	767	15	0.88	447.7	18
frb59-26-2	-	50	18.0	4.74	767	17	0.88	448.6	17
frb59-26-3	-	52	13.4	3.48	767	19	0.87	409.2	19
Amazon	35.4	1.75×10^5	-	23.0	1.85×10^5	1.56×10^5	3.09	1.73×10^5	1.43×10^5
DBLP	17.3	1.52×10^5	-	23.2	1.75×10^5	1.41×10^5	2.72	1.66×10^5	1.34×10^5
Google+	-	1.06×10^5	0.02	44.5	1.11×10^5	9.39×10^4	4.37	1.00×10^5	8.67×10^4

Table 5.4: Wall-clock time and quality of fractional and integral solutions for three graph analysis problems using Thetis, Cplex-IP and Cplex-LP. Each code was given a time limit of one hour, with '-' indicating a timeout. BFS is the objective value of the best integer feasible solution found by Cplex-IP. The gap is defined as (BFS−BB)/BFS where BB is the best known solution bound at the end of the time limit. A gap of '-' indicates that the problem was solved to within 0.01% accuracy and NA indicates that Cplex-IP was unable to find a valid solution bound. LP is the objective value of the LP solution, and RSol is objective value of the rounded solution.

Maximum Independent Set. The independent set problem is theoretically one of the hardest known combinatorial problems. Again, we observed that the quality

VC	Cplex-IP			Cplex-LP (default)			Thetis		
(min)	t (secs)	BFS	Gap(%)	t (secs)	LP	RSol	t (secs)	LP	RSol
frb59-26-1	-	1475	0.7	4.59	767.0	1534	0.88	959.7	1532
frb59-26-2	-	1475	0.6	4.67	767.0	1534	0.86	979.7	1532
frb59-26-3	-	1475	0.5	4.76	767.0	1534	0.89	982.9	1533
Amazon	85.5	1.60×10^5	-	21.6	1.50×10^5	1.99×10^5	2.97	1.50×10^5	1.97×10^5
DBLP	22.1	1.65×10^5	-	23.7	1.42×10^5	2.07×10^5	2.70	1.42×10^5	2.06×10^5
Google+	-	1.06×10^5	0.01	60.0	1.00×10^5	1.30×10^5	4.47	1.00×10^5	1.27×10^5
MC	Cplex-IP			Cplex-LP (default)			Thetis ($\epsilon = 0.1$)		
(min)	t (secs)	BFS	Gap(%)	t (secs)	LP	RSol	t (secs)	LP	RSol
frb59-26-1	547.4	346	-	397.0	346	346	5.86	352.3	349
frb59-26-2	71.9	254	-	330.1	254	254	5.82	262.3	254
frb59-26-3	30.3	367	-	400.6	367	367	5.86	387.7	367
Amazon	-	12	NA	-	-	-	55.8	7.28	5
DBLP	-	15	NA	-	-	-	63.8	11.70	5
Google+	-	6	NA	-	-	-	109.9	5.84	5
MIS	Cplex-IP			Cplex-LP (default)			Thetis ($\epsilon = 0.1$)		
(max)	t (secs)	BFS	Gap(%)	t (secs)	LP	RSol	t (secs)	LP	RSol
frb59-26-1	-	50	18.0	4.88	767	16	0.88	447.7	18
frb59-26-2	-	50	18.0	4.82	767	16	0.88	448.6	17
frb59-26-3	-	52	13.4	4.85	767	16	0.87	409.2	19
Amazon	35.4	1.75×10^5	-	25.7	1.85×10^5	1.58×10^5	3.09	1.73×10^5	1.43×10^5
DBLP	17.3	1.52×10^5	-	24.0	1.75×10^5	1.41×10^5	2.72	1.66×10^5	1.34×10^5
Google+	-	1.06×10^5	0.02	68.8	1.11×10^5	9.40×10^4	4.37	1.00×10^5	8.67×10^4

Table 5.5: Wall-clock time and quality of fractional and integral solutions for three graph analysis problems using Thetis, Cplex-IP and Cplex-LP (run to default tolerance). Each code was given a time limit of one hour, with '-' indicating a timeout. BFS is the objective value of the best integer feasible solution found by Cplex-IP. The gap is defined as (BFS−BB)/BFS where BB is the best known solution bound at the end of the time limit. A gap of '-' indicates that the problem was solved to within 0.01% accuracy and NA indicates that Cplex-IP was unable to find a valid solution bound. LP is the objective value of the LP solution, and RSol is objective value of the rounded solution.

of the rounded solutions obtained using Cplex-LP and Thetis were comparable but Thetis was 5-10x faster than Cplex-IP. In terms of solution quality, on the Bhoslib

instances, the integral solutions obtained using Cplex-LP and Thetis were a factor of 3 away from the documented optimal solutions. Even Cplex-IP was unable to solve any of the instances within an hour. The terminal optimality gaps were as large as 20%. On the social networking instances, we observed that the rounded feasible solutions obtained from Thetis were only marginally worse than those obtained using Cplex-IP, but were obtained using an order of magnitude less time.

Multiway-Cut. The LP formulation of the multiway-cut problem was the largest of the three problems discussed in the experimental results. The number of variables in the multiway-cut problem is $O(|E| \times k)$ where $|E|$ is the number of edges in the graph and k is the number of terminals. The terminals were chosen randomly to be in the same connected component of the graph. All instances, excepting Google+, were fully connected. For Google+, 201949 (out of 211186) vertices were connected to the terminals. For all instances, including Google+, all codes were run on the formulation (5.6) built using the entire graph. The LP representation of (5.6) was around ten times larger than vertex cover and independent set. We solved the QP-approximation for the multiway-cut problem using block-SCD, which is variant of Algorithm 6. Each iteration of the block-SCD method updates a block of coordinates. If the block corresponds to the variable x_v, $\forall v \in V$ in (5.6), we perform a projection on to the k-dimensional simplex Δ_k. The simplex projection is necessary to ensure that the approximate LP solution is always feasible for (5.6). We disabled presolve for Thetis to prevent the simplex constraints from being eliminated or altered. We did not disable presolve for Cplex-LP or Cplex-IP. Our results demonstrate that our solver is much more scalable than both Cplex-IP and Cplex-LP. We were over an

Instance	Problem Statistics		Time (seconds)	
	PV	NNZ	Cplex-QP	ASCD
frb59-26-1	0.12	0.88	1410.7	0.34
frb59-26-2	0.12	0.88	1371.7	0.32
frb59-26-3	0.12	0.88	-	0.36
Amazon	0.56	2.91	462.8	0.89
DBLP	0.52	2.78	427.7	0.77
Google+	0.77	5.06	-	2.83

Table 5.6: Summary of wall-clock time required by ASCD and Cplex-QP to solve the QP-approximation (5.11) for the vertex cover problem. Each code is run with a time limit of one hour and parallelized over 32 cores, with '-' indicating that the code reached the time limit. PV is the number of primal variables while NNZ is the number of nonzeros in the hessian matrix (both in millions).

order of magnitude faster than Cplex-LP on the Bhoslib instances with solutions of comparable quality. In fact, LP-rounding recovered the optimal solution on some of the instances. On the SNAP instances, both Cplex-IP and Cplex-LP failed to complete within an hour on any of the instances. Cplex-IP was able to generate feasible solutions using its heuristics, but was able to unable to solve the root-node relaxation on any of the instances.

Effect of LP-solution accuracy on rounded solutions

The results reported in table 5.4 use comparable (fairly loose) tolerances for both Cplex-LP and Thetis. Table 5.7 reports the computational time required by Cplex-LP and Thetis to solve the LP-relaxations of the vertex cover problem to three different values of tolerance i.e. $\epsilon = 1 \times 10^{-1}$, $\epsilon = 1 \times 10^{-3}$ and $\epsilon = 1 \times 10^{-5}$ and the quality of the corresponding integral solution obtained by rounding each of the LP solutions. Our results, on the Bhoslib as well as social networking instances,

Cplex-LP

Instance	$\epsilon = 1 \times 10^{-1}$			$\epsilon = 1 \times 10^{-3}$			$\epsilon = 1 \times 10^{-5}$		
	t(s)	LP	RSol	t(s)	LP	RSol	t(s)	LP	RSol
frb59-26-1	2.48	767.0	1534	4.70	767.0	1534	4.59	767.0	1534
frb59-26-2	3.93	767.0	1534	4.61	767.0	1534	4.67	767.0	1534
frb59-26-3	4.42	767.0	1534	4.62	767.0	1534	4.76	767.0	1534
Amazon	24.8	1.50×10^5	2.04×10^5	21.0	1.50×10^5	1.99×10^5	46.7	1.50×10^5	1.99×10^5
DBLP	22.3	1.42×10^5	2.08×10^5	22.8	1.42×10^5	2.07×10^5	31.1	1.42×10^5	2.06×10^5
Google+	40.1	1.00×10^5	1.31×10^5	61.1	1.00×10^5	1.29×10^5	60.0	1.00×10^5	1.30×10^5

Thetis

Instance	$\epsilon = 1 \times 10^{-1}$			$\epsilon = 1 \times 10^{-3}$			$\epsilon = 1 \times 10^{-5}$		
	t(s)	LP	RSol	t(s)	LP	RSol	t(s)	LP	RSol
frb59-26-1	0.88	959.7	1532	13.7	767.0	1534	13.3	767.0	1534
frb59-26-2	0.86	979.7	1532	14.2	767.0	1534	14.1	767.0	1534
frb59-26-3	0.89	982.9	1533	12.9	767.0	1534	12.9	767.0	1534
Amazon	2.97	1.50×10^5	1.97×10^5	59.5	1.50×10^5	1.99×10^5	50.3	1.50×10^5	1.99×10^5
DBLP	2.70	1.42×10^5	2.06×10^5	39.2	1.42×10^5	2.07×10^5	59.1	1.42×10^5	2.07×10^5
Google+	4.47	1.00×10^5	1.27×10^5	1420.1	1.00×10^5	1.29×10^5	2818.2	1.00×10^5	1.30×10^5

Table 5.7: Wall-clock time and quality of fractional and integral solutions for the vertex cover proble using Thetis and Cplex-LP solved to three different tolerance levels. Each code was given a time limit of one hour, with '-' indicating a timeout. LP is the objective value of the LP solution, and RSol is objective value of the rounded solution.

indicate that computational time required by Cplex-LP to solve the LP-relaxation (5.8) was relatively insensitive to the to tolerance level ϵ. In contrast, Thetis was much faster at finding less accurate solutions. For both methods, the quality of integral solutions obtained using LP-rounding are not sensitive to the choice of tolerance. Our results validated the main claim in the chapter that in LP rounding schemes, since we ultimately round the LP to obtain an approximate solution of the combinatorial problem, a crude solution of the LP may suffice.

Comparing ASCD and Cplex while solving (5.11)**.** We highlight the importance of the ASCD by comparing its performance with Cplex-QP in solving the QP

approximation (5.11) of the vertex cover problem to a tolerance of 1×10^{-3} with $\beta = 1$. Table 5.6 reports problem statistics as well as wall-clock time required by each of the codes. We used the barrier optimizer option for Cplex-QP. ASCD was three orders of magnitude faster than Cplex-QP on the Bhoslib instances and up to 2 orders of magnitude faster than Cplex-QP on the social networking instances. Cplex-QP was unable to solve the QP-approximations, within the time limit, for some of the social networking instances.

5.6 Case Study: LP-rounding in Machine Learning

We now present a case study demonstrating the use of LP-rounding for tasks that occur in the field of machine learning: entity resolution and text chunking. We compare application level quality of the rounded solutions obtained using Thetis, optimal integral solutions obtained using Cplex-IP and the solutions obtained using using Gibbs sampling-based approach [114] which is a state-of-the-art system for these tasks.

Background Graphical models are probabilistic models that are commonly used in the design and analysis of algorithms used in machine learning. In these models, a graph structure encodes a joint probability distribution, where vertices correspond to random variables and edges specify conditional dependencies. In graphical models, *inference* is the problem of computing the posterior distribution of 'hidden' nodes given 'observed' nodes in a graphical model. The Maximum-A-Posteriori (MAP) inference problem is the task of finding the most likely assignment to the

Task	Formulation	PV	NNZ	Method	**P**	**R**	**F1**	Rank
CoNLL	Skip-chain CRF	25M	51M	Cplex-IP	.87	.91	.89	10/13
				Thetis	.87	.90	.89	10/13
				Gibbs Sampling	.86	.90	.88	10/13
TAC-KBP	Factor graph	62K	115K	Cplex-IP	.80	.80	.80	6/17
				Thetis	.79	.79	.79	6/17
				Gibbs Sampling	.80	.80	.80	6/17

Table 5.8: Solution quality of our LP-rounding approach on machine learning tasks. PV is the number of primal variables and NNZ is the number of non-zeros in the constraint matrix of the LP in standard form. The *rank* indicates where each code would have placed, had it been used in the competition.

variables in a graphical model. A factor graph [54] is a specific type of graphical models in which a bipartite hypergraph can be used to expresses how the joint-probability distribution of the variables can be factored into a product of functions that depend only on a subset of the variables. The MAP inference problem, on factor graphs, can be solved using belief propagation or the sum-product algorithm [76]. The MAP estimation problem can also be formulated as an IP. The connection between belief-propagation and the LP-relaxation of the IP formulation was studied by Wainwright et al. [106]. They showed that if the optimal solution of the LP-relaxation is integral, then an optimal solution of the LP-relaxation is a solution to the MAP inference problem. Ravikumar et al. [79] build on this result and show that approximate LP solutions can also be rounded to solve MAP inference on graphical models.

Results We now focus on solving MAP estimation problems using factor graphs on two different tasks: entity linking and text chunking. For entity linking, we use

the TAC-KBP 2010 benchmark[3]. The input graphical model has 12k boolean random variables and 17k factors. For text chunking, we use the CoNLL 2000 shared task[4]. The factor graph contained 47K categorical random variables (with domain size 23) and 100K factors. We use the training sets provided by TAC-KBP 2010 and CoNLL 2000 respectively. For each task, we ran the MAP inference on the factor graph using the LP formulation in [54] and compare the quality of the solutions obtained by Thetis, Cplex-IP and a Gibbs sampling-based approach [114]. For Thetis, we follow the LP-rounding algorithm in [79] to solve the MAP estimation problem. We evaluate the quality of both approaches using the official evaluation scripts and evaluation data set provided by each challenge. Table 5.8 contains a description of the three relevant quality metrics at the application level: precision (P), recall (R), and F1-scores. Table 5.8 demonstrates that our algorithm produces solutions of comparable quality to state-of-the-art approaches for these graph analysis tasks. Our results also indicate that solution quality (at an application level) obtained using Thetis was comparable to Cplex-IP suggesting that, in these instances, an exact solution of the IP does not provide any more value than a 'good enough' solution obtained using Thetis.

5.7 Conclusion

We described Thetis, an LP rounding procedure based on an approximate solver for the LP-relaxation. We derived worst-case runtime and solution quality bounds

[3] http://nlp.cs.qc.cuny.edu/kbp/2010/
[4] http://www.cnts.ua.ac.be/conll2000/chunking/

for our scheme, and demonstrated that our approach was faster than an alternative based on a state-of-the-art LP solver, while producing rounded solutions of comparable quality.

6 CONCLUSIONS AND FUTURE WORK

In this dissertation we discussed several non-convex and combinatorial optimization problems. In Chapter 2, we presented a theoretical and computational comparison of MIP models for PLFs when a binary indicator variable determines if the PLF is required to be evaluated. For a large class of popular MIP formulations, we showed that our proposed formulations were either locally ideal or sharp while standard formulations were not. Our numerical experiments showed that our proposed formulations have significant computational advantages.

In Chapter 3, we considered a production planning problem in which the production process creates a mixture of desirable products and undesirable byproducts. We presented a novel discrete-time MINLP formulation of the problem that was more accurate than a previously considered formulations. We then proposed a MIP approximation and two MIP relaxations of our proposed MINLP formulation. Our numerical experiments demonstrated that our MINLP formulation was 30% more accurate than past work and that our MIP approximations and relaxations were able to find near-optimal solutions for large instances of this non-convex production planning problem.

In Chapter 4, we studied production planning problems influenced by IRR-based PSCs. We presented two approaches for finding feasible solutions: a MIP formulation and a search algorithm based on a novel continuous domain formulation. We also proposed a market-based decomposition scheme to compute solution bounds. Our experiments, on a sample application problem, demonstrated that the

MIP formulation was suitable for problems with 3 or fewer markets. For problems with more than 3 markets, a combination of our search algorithm and market-based decomposition algorithm was preferable. The update rule discussed in Algorithm 4 works well in practice but we are currently not aware of any convergence analysis (even when the function being optimized is convex in the primal variables) that shows that the primal variables, dual variables and solution bound all converge.

In Chapter 5, we study three classes of combinatorial optimization problems, with applications in fields such as machine learning, advertising, statistics, and computer vision. For each of these problems, one can obtain near-optimal solutions by rounding the solution of an LP. We showed that one can recover solutions of comparable quality by rounding an approximate LP solution. We built software that could compute these approximate LP solutions up to an order of magnitude faster than Cplex. We also derived bounds on worst case runtime and solution quality bound of our approximate LP-rounding scheme. Our work on approximate LP-rounding is relatively new. A natural direction for future work is to extend the results derived for covering, packing, and multiway-cut problems to a wider range of LP-rounding schemes. Another possibility is to extend the worst case run-time and solution quality bounds from the QP-approximation to augmented Lagrangian algorithms. A third possibility is to address one of the main disadvantages of LP-rounding: the 'one-shot' nature of this scheme which means that quality of feasible integral solutions cannot be improved beyond an initial rounding of the root LP-relaxation. One possible way to address this issue is to use cutting-plane algorithms that iteratively refine fractional solutions by adding suitable linear

inequalities (called cuts). For classes of combinatorial problems, such as maximum-independent set, there are known families of cuts [63, 87, 61, 5, 6, 40] with desirable theoretical and computational properties. It is known that the addition of only a few carefully chosen inequalities from these families can significantly improve solution quality. Unfortunately, some of the most effective families of cuts require LP basis information which is not easily available using penalty based methods like the one discussed in Chapter 5. Enhancing solution quality of approximate LP-rounding with a general class of cutting plane algorithms could be an important step towards effectively solving large scale combinatorial problems.

BIBLIOGRAPHY

[1] Alguacil, N., A. L. Motto, and A. J. Conejo (2003). Transmission expansion planning: a mixed-integer LP approach. *IEEE Transactions on Power Systems 18*, 1070–1077.

[2] Bagajewicz, M. (2008). On the use of net present value in investment capacity planning models. *Industrial & Engineering Chemistry Research 47*(23), 9413–9416.

[3] Balakrishnan, A. and S. C. Graves (1989). A composite algorithm for a concave-cost network flow problem. *Networks 19*(2), 175–202.

[4] Balas, E. (1979). Disjunctive Programming. In *Annals of Discrete Mathematics 5: Discrete Optimization*, pp. 3–51. North Holland.

[5] Balas, E., S. Ceria, and G. Cornuéjols (1993). A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming 58*(1-3), 295–324.

[6] Balas, E., S. Ceria, G. Cornuéjols, and N. Natraj (1996). Gomory cuts revisited. *Operations Research Letters 19*(1), 1–9.

[7] Bansal, N., A. Blum, and S. Chawla (2004). Correlation clustering. *Machine Learning 56*(1-3), 89–113.

[8] Bansal, N., N. Korula, V. Nagarajan, and A. Srinivasan (2012). Solving packing integer programs via randomized rounding with alterations. *Theory Of Computing 8*(1), 533–565.

[9] Beale, E. M. L. and J. J. H. Forrest (1976). Global optimization using special ordered sets. *Mathematical Programming 10*(1), 52–69.

[10] Beale, E. M. L. and J. A. Tomlin (1970). Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *Operations Research 69*, 447–454.

[11] Beck, A. and L. Tetruashvili (2013). On the convergence of block coordinate descent type methods. *Technical report*.

[12] Belotti, P., J. Lee, L. Liberti, F. Margot, and A. Wächter (2009). Branching and bounds tighteningtechniques for non-convex minlp. *Optimization Methods & Software 24*(4-5), 597–634.

[13] Bien, J. and R. Tibshirani (2009). Classification by set cover: The prototype vector machine. *arXiv preprint arXiv:0908.2284*.

[14] Bodlaender, H. L. (1993). A tourist guide through treewidth. *Technical report RUU-CS 92*.

[15] Borghetti, A., S. Member, C. D'Ambrosio, A. Lodi, and S. Martello (2008). An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on Power Systems 23*(3), 1115–1124.

[16] Boykov, Y. and V. Kolmogorov (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26*, 1124–1137.

[17] Burkard, R. E., H. W. Hamacher, and G. Rote (1991). Sandwich approximation of univariate convex functions with an application to separable convex programming. *Naval Research Logistics 38*, 911–924.

[18] Călinescu, G., H. Karloff, and Y. Rabani (1998). An improved approximation algorithm for multiway cut. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 48–52. ACM.

[19] Camponogara, E., M. P. de Castro, A. Plucenio, and D. J. Pagano (2011). Compressor scheduling in oil fields. *Optimization and Engineering 12*(1-2), 153–174.

[20] Carøe, C. C. and R. Schultz (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters 24*(1), 37–45.

[21] Carrion, M. and J. M. Arroyo (2006). A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems 21*(3), 1371–1378.

[22] Carvalho, M. and J. Pinto (2006). A bilevel decomposition technique for the optimal planning of offshore platforms. *Brazilian Journal of Chemical Engineering 23*(1), 67–82.

[23] Ceria, S. and J. Soares (1999). Convex programming for disjunctive convex optimization. *Mathematical Programming 614*, 595–614.

[24] Chekuri, C., S. Khanna, J. S. Naor, and L. Zosin (2001). Approximation algorithms for the metric labeling problem via a new linear programming for-

mulation. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pp. 109–118. Society for Industrial and Applied Mathematics.

[25] Croxton, K. L., B. Gendron, and T. L. Magnanti (2003). Comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science 49*(9), 1268–1273.

[26] D'Ambrosio, C., J. Lee, and A. Wachter (2009). A global-optimization algorithm for mixed-integer nonlinear programs having separable non-convexity. In A. Fiat and P. Sanders (Eds.), *Algorithms - ESA 2009*, Volume 5757 of *Lecture Notes in Computer Science*, pp. 107–118. Springer Berlin / Heidelberg.

[27] Dantzig, G. (1960). On the significance of solving linear programming problems with some integer variables. *Econometrica, Journal of the Econometric Society 28*(1), 30–44.

[28] Deutsche Bank (2010). Oil & gas for beginners: A guide to the oil and gas industry.

[29] Eckstein, J. and P. J. Silva (2010). A practical relative error criterion for augmented lagrangians. *Mathematical Programming*, 1–30.

[30] Everett, H. (1963). Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research 11*(3), 399–417.

[31] Fan, Y. and C. Liu (2010). Solving stochastic transportation network protection problems using the progressive hedging-based method. *Networks and Spatial Economics 10*(2), 193–208.

[32] Fisher, M. L. (2004). The lagrangian relaxation method for solving integer programming problems. *Management science 50*(12 supplement), 1861–1871.

[33] Frangioni, A., C. Gentile, and F. Lacalandra (2008). Solving unit commitment problems with general ramp constraints. *International Journal of Electrical Power & Energy Systems 30*(5), 316–326.

[34] Freund, A. and H. Karloff (2000). A lower bound of 8/(7+ 1k- 1) on the integrality ratio of the călinescu–karloff–rabani relaxation for multiway cut. *Information Processing Letters 75*(1), 43–50.

[35] Gade, D., S. M. Ryan, J.-P. Watson, and D. L. Woodruff (2013). A new lower bound from progressive hedging. International Conference on Stochastic Programming.

[36] Geissler, B., A. Martin, A. Morsi, and L. Schewe (2012a). Using piecewise linear functions for solving MINLPs. In *Mixed Integer Nonlinear Programming*, Volume 154 of *The IMA Volumes in Mathematics and its Applications*, pp. 287–314. Springer.

[37] Geissler, B., A. Martin, A. Morsi, and L. Schewe (2012b). Using piecewise linear functions for solving MINLPs. In J. Lee and S. Leyffer (Eds.), *Mixed Integer Nonlinear Programming*, Volume 154 of *The IMA Volumes in Mathematics and its Applications*, pp. 287–314. Springer New York.

[38] Geoffrion, A. M. (1977). Objective function approximations in mathematical programming. *Mathematical Programming 13*(1), 23–37.

[39] Goldberg, N., Y. Kim, S. Leyffer, and T. D. Veselka (2012). Adaptively refined dynamic program for linear spline regression. Technical Report Preprint ANL/MCS-P3040-0912, Argonne National Laboratory.

[40] Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society 64*(5), 275–278.

[41] Günlük, O. and J. Linderoth (2010). Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Mathematical programming 124*(1-2), 183–205.

[42] Gupta, V. and I. E. Grossmann (2012a). An efficient multiperiod minlp model for optimal planning of offshore oil and gas field infrastructure. *Industrial & Engineering Chemistry Research 51*(19), 6823–6840.

[43] Gupta, V. and I. E. Grossmann (2012b). An efficient multiperiod minlp model for optimal planning of offshore oil and gas field infrastructure. *Industrial & Engineering Chemistry Research 51*(19), 6823–6840.

[44] Gupta, V. and I. E. Grossmann (2012c). Modeling and computational strategies for optimal development planning of offshore oilfields under complex fiscal rules. *Industrial & Engineering Chemistry Research 51*(44), 14438–14460.

[45] Hastad, J. (1996). Clique is hard to approximate within $n^{1-\epsilon}$. pp. 627–636.

[46] He, B., H. Yang, and S. Wang (2000). Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications 106*(2), 337–356.

[47] Hochbaum, D. S. (1982). Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing 11*(3), 555–556.

[48] Imamoto, A. and B. Tang (2008). Optimal piecewise linear approximation of convex functions. In *Proceedings of the world congress on engineering and computer science*, pp. 1191–1194. Citeseer.

[49] Iyer, R. and I. E. Grossmann (1998). Optimal planning and scheduling of offshore oil field infrastructure investment and operations. *Industrial & Engineering Chemistry Research 37*(4), 1380–1397.

[50] Jeroslow, R. G. and J. K. Lowe (1984). Modelling with integer variables. In *Mathematical Programming at Oberwolfach II*, Volume 22 of *Mathematical Programming Studies*, pp. 167–184. Springer Berlin Heidelberg.

[51] Keha, A. B., I. R. de Farias, and G. L. Nemhauser (2004). Models for representing piecewise linear cost functions. *Operations Research Letters 32*(1), 44–48.

[52] Khachiyan, L. G. (1980). Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics 20*(1), 53–72.

[53] Kleinberg, J. and E. Tardos (2002). Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM (JACM) 49*(5), 616–639.

[54] Kschischang, F. R., B. J. Frey, and H.-A. Loeliger (2001). Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on 47*(2), 498–519.

[55] Laínez, J. M., L. Puigjaner, and G. V. Reklaitis (2009). Financial and financial engineering considerations in supply chain and product development pipeline management. *Computers & Chemical Engineering 33*(12), 1999–2011.

[56] Lee, J. and D. Wilson (2001). Polyhedral methods for piecewise-linear functions i: the lambda method. *Discrete applied mathematics 108*(3), 269–285.

[57] Lee, T., Z. Wang, H. Wang, and S.-w. Hwang (2011). Web scale entity resolution using relational evidence. Technical report, Technical report, Microsoft Research, 2011. Available at http://research. microsoft. com/apps/pubs/default. aspx.

[58] Lin, X. and C. A. Floudas (2003). A novel continuous-time modeling and optimization framework for well platform planning problems. *Optimization and Engineering 4*(1), 65–95.

[59] Liu, J., S. J. Wright, C. Ré, and V. Bittorf (2013, October). An asynchronous parallel stochastic coordinate descent algorithm. Technical report, University of Wisconsin-Madison.

[60] Lodish, L. (1971). CALLPLAN: An interactive salesman's call planning system. *Management Science 18*(4).

[61] Lovász, L. and A. Schrijver (1991). Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization 1*(2), 166–190.

[62] Manshadi, F., B. Awerbuch, R. Gemulla, R. Khandekar, J. Mestre, and M. Sozio (2013). A distributed algorithm for large-scale generalized matching. *Proceedings of the VLDB Endowment*.

[63] Marchand, H., A. Martin, R. Weismantel, and L. Wolsey (2002). Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics 123*(1), 397–446.

[64] Markowitz, H. M. and A. S. Manne (1957). On the solution of discrete programming problems. *Econometrica 25*(1), 84–110.

[65] Martin, A., M. Möller, and S. Moritz (2006). Mixed integer models for the stationary case of gas network optimization. *Mathematical Programming 105*, 563–582.

[66] Martín, M. and I. E. Grossmann (2011). Energy optimization of hydrogen production from lignocellulosic biomass. *Computers & Chemical Engineering 35*(9), 1798–1806.

[67] Meyer, R. R. (1976). Integer and mixed-integer programming models: General properties. *Journal of Optimization Theory and Applications 16*, 191–206.

[68] Nemirovski, A., A. Juditsky, G. Lan, and A. Shapiro (2009). Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization 19*(4), 1574–1609.

[69] Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization 22*(2), 341–362.

[70] Niu, F., B. Recht, C. Ré, and S. J. Wright (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *arXiv preprint arXiv:1106.5730*.

[71] Norstrøm, C. J. (1972). A sufficient condition for a unique nonnegative internal rate of return. *Journal of Financial and Quantitative Analysis 7*(03), 1835–1839.

[72] Orero, S. and M. Irving (1998). A genetic algorithm modelling framework and solution technique for short term optimal hydrothermal scheduling. *Power Systems, IEEE Transactions on 13*(2), 501–518.

[73] Padberg, M. (2000). Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters 27*(1), 1–5.

[74] Padberg, M. and M. P. Rijal (1998). Location, Scheduling, Design and Integer Programming. *Journal of The Operational Research Society 49*, 93–98.

[75] Park, C. S., P. Kumar, and N. Kumar (2004). *Fundamentals of Engineering Economics*. Pearson/Prentice Hall.

[76] Pearl, J. (1982). Reverend Bayes on inference engines: a distributed hierarchical approach. In *National Conference on Artificial Intelligence*, pp. 133–136.

[77] Perold, A. F. (1984). Large-scale portfolio optimization. *Management Science 30*(10), 1143–1160.

[78] Polyak, B. T. (1969). Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics 9*(3), 14–29.

[79] Ravikumar, P., A. Agarwal, and M. J. Wainwright (2010). Message-passing for graph-structured linear programs: Proximal methods and rounding schemes. *The Journal of Machine Learning Research 11*, 1043–1080.

[80] Renegar, J. (1994). Some perturbation theory for linear programming. *Mathenatical Programming, Series A 65*, 73–92.

[81] Rockafellar, R. T. (1976). Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization 14*(5), 877–898.

[82] Rosen, J. B. (1960). The gradient projection method for nonlinear programming. Part I. Linear constraints. *Journal of the Society for Industrial & Applied Mathematics 8*(1), 181–217.

[83] Rote, G. (1992). The convergence rate of the sandwich algorithm for approximating convex functions. *Computing*.

[84] Roth, D. and W. Yih (2005). Integer linear programming inference for conditional random fields. In *Proceedings of the 22nd international conference on Machine learning*, pp. 736–743. ACM.

[85] Sanghavi, S., D. Malioutov, and A. S. Willsky (2007). Linear programming analysis of loopy belief propagation for weighted matching. In *Advances in neural information processing systems*, pp. 1273–1280.

[86] Selot, A., L. K. Kuok, M. Robinson, T. L. Mason, and P. I. Barton (2008). A short-term operational planning model for natural gas production systems. *AIChE Journal 54*(2), 495–515.

[87] Sherali, H. D. and W. P. Adams (1990). A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics 3*(3), 411–430.

[88] Simon, J. and J. Arndt (1980). The shape of the advertising response function. *Journal of Advertising Research*.

[89] Sridhar, S., V. Bittorf, J. Liu, C. Zhang, C. Ré, and S. J. Wright (2013). An approximate, efficient solver for lp rounding. *arXiv preprint arXiv:1106.5730*.

[90] Sridhar, S., J. Linderoth, and J. Leudtke (2013a). Locally ideal formulations for piecewise linear functions with indicator variables. *Operations Research Letters 41*(6), 627 – 632.

[91] Sridhar, S., J. Linderoth, and J. Leudtke (2013b). Models and solution techniques for production planning problems with increasing byproducts. *Optimization Online*.

[92] Srinivasan, A. (1999). Improved approximation guarantees for packing and covering integer programs. *SIAM Journal on Computing 29*(2), 648–670.

[93] Tarhan, B., I. E. Grossmann, and V. Goel (2009). Stochastic programming approach for the planning of offshore oil or gas field infrastructure under decision-dependent uncertainty. *Industrial & Engineering Chemistry Research 48*(6), 3078–3097.

[94] Tawarmalani, M. and N. V. Sahinidis (2004). Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical programming 99*(3), 563–591.

[95] Tseng, P. (2001). Convergence of a block coordinate descent method for non-differentiable minimization. *Journal of Optimization Theory and Applications 109*, 475–494.

[96] Van Den Heever, S. A. and I. E. Grossmann (2000). An iterative aggregation/disaggregation approach for the solution of a mixed-integer nonlinear oilfield infrastructure planning model. *Industrial & engineering chemistry research 39*(6), 1955–1971.

[97] Van den Heever, S. A., I. E. Grossmann, S. Vasantharajan, and K. Edwards (2000). Integrating complex economic objectives with the design and planning of offshore oilfield infrastructures. *Computers & Chemical Engineering 24*(2), 1049–1055.

[98] Van den Heever, S. A., I. E. Grossmann, S. Vasantharajan, and K. Edwards (2001). A lagrangean decomposition heuristic for the design and planning of offshore hydrocarbon field infrastructures with complex economic objectives. *Industrial & engineering chemistry research 40*(13), 2857–2875.

[99] Van Gael, J. and X. Zhu (2007). Correlation clustering for crosslingual link detection. In *IJCAI*, pp. 1744–1749.

[100] Van Slyke, R. M. and R. Wets (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics 17*(4), 638–663.

[101] Vazirani, V. V. (2004). *Approximation algorithms*. springer.

[102] Vielma, J. P., S. Ahmed, and G. Nemhauser (2009). Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research 58*(2), 303–315.

[103] Vielma, J. P., S. Ahmed, and G. Nemhauser (2010). Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. *Operations research 58*(2), 303–315.

[104] Vielma, J. P., A. B. Keha, and G. L. Nemhauser (2008). Nonconvex, lower semicontinuous piecewise linear optimization. *Discrete Optimization 5*(2), 467–488.

[105] Vielma, J. P. and G. L. Nemhauser (2009). Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming 128*(1-2), 49–72.

[106] Wainwright, M. J., T. S. Jaakkola, and A. S. Willsky (2005). Map estimation via agreement on trees: message-passing and linear programming. *Information Theory, IEEE Transactions on 51*(11), 3697–3717.

[107] Wang, S. and L. Liao (2001). Decomposition method with a variable parameter for a class of monotone variational inequality problems. *Journal of optimization theory and applications 109*(2), 415–429.

[108] Watson, J.-P. and D. L. Woodruff (2011). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science 8*(4), 355–370.

[109] Wolsey, L. A. (2000). Integer Programming. *IIE Transactions 32*(273-285), 2–58.

[110] Wright, S. J. (1990). Implementing proximal point methods for linear programming. *Journal of Optimization Theory and Applications 65*(3), 531–554.

[111] Wright, S. J. (2012). Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization 22*(1), 159–186.

[112] Wu, Z., A. Thangali, S. Sclaroff, and M. Betke (2012). Coupling detection and data association for multiple object tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1948–1955. IEEE.

[113] Xu, K. and W. Li (2006). Many hard examples in exact phase transitions. *Theoretical Computer Science 355*(3), 291–302.

[114] Zhang, C. and C. Ré (2013). Towards high-throughput gibbs sampling at scale: A study across storage managers. SIGMOD.

[115] Zoltners, A. and P. Sinha (1980). Integer programming models for sales resource allocation. *Management Science 26*(3), 242–260.