Mixed 0-1 conic quadratic optimization: formulations, convex relaxations and algorithms

by

Andrés Gómez Escobar

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Industrial Engineering and Operations Research

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alper Atamtürk, Chair
Professor Dorit Hochbaum
Professor Peter Bartlett

Summer 2017

Mixed 0-1 conic quadratic optimization: formulations, convex relaxations and algorithms

Andrés Gómez Escobar

Abstract

Mixed 0-1 conic quadratic optimization: formulations, convex relaxations and algorithms

by

Andrés Gómez Escobar

Doctor of Philosophy in Engineering - Industrial Engineering and Operations Research

University of California, Berkeley

Professor Alper Atamtürk, Chair

Conic quadratic functions arise often when modeling uncertainty and risk-aversion, and are used in many fields including finance, machine-learning and robotics. Such functions are convex, and thanks to substantial efforts over the past two decades in developing techniques for convex problems, large conic quadratic optimization problems can be solved efficiently in practice. However, many decision-making problems involving logical choices are discrete in nature, and thus non-convex. Despite considerable improvements in our ability to solve mixed-integer *linear* optimization problems (MILO), their nonlinear and conic counterparts are still poorly understood and considered intractable.

Most of the advances in solving mixed-integer nonlinear optimization (MINLO) were obtained by adapting techniques used for linear discrete optimization. One of the first approaches proposed was to construct a linearization of the nonlinear terms and solving the optimization problem as a MILO, but since such approaches solve a *relaxation* of the original problem, they may fail to find optimal solutions. More recent approaches that have proved successful involve using linear outer approximations with extended formulations, or using mixed-integer rounding and lift-and-project cuts, which where original proposed for MILO. However, such approaches based on previous results for MILO may fail to consider and exploit the specific structure of the nonlinear discrete problems.

In this dissertation, we study the structures specific to nonlinear mixed-integer optimization. Moreover, we propose a variety of novel algorithms for conic discrete optimization. The algorithms, despite exploiting the nonlinear structure of the problems, are similar to algorithms commonly used for the linear case.

In Chapter 2 we study the problem of maximizing a class of nonlinear utility functions over the vertices of an integral polytope, and propose an approximation algorithm for the problem. The algorithm exploits the fact that there exists an optimal solution to the natural convex relaxation of the problem in an edge of the polytope, and rounds the solution to a vertex.

One of the principal approaches for solving discrete optimization problems to optimality are branch-and-bound algorithms. For linear problems, branch-and-bound algorithms are typically implemented using the simplex method, which allows to use warm-starts to efficiently solve the convex subproblems at each node of the branch-and-bound tree. In Chapter 3 we present a simplex method for conic quadratic minimization over polyhedra, and show that the algorithm outperforms existing methods in both convex and discrete instances.

In Chapter 4 we study strong formulations for general mixed-binary conic quadratic optimization. In particular, we give a complete description of the convex hull of the lower level set of a mixed-integer conic quadratic function. The convex hull can be described in an extended formulation using a single conic quadratic constraint and exponentially many linear inequalities. Thus, the inequalities can be implemented as cutting planes using existing techniques. Our computational experiments indicate that the inequalities strengthen the formulation considerably, and often result in order-of-magnitude improvements over commercial software.

In Chapter 5 we consider binary quadratic problems, which are a special case of conic quadratic problems. We propose an approach based on the decomposition of binary quadratic functions into a submodular component and a component with a convex relaxation. Then, by linearizing only the submodular component, we obtain formulations stronger than the natural convex relaxation of the problem and the formulation obtained from the full linearization of the quadratic expression. Preliminary computational experiments indicate that the proposed approach can result in considerable faster branch-and-bound algorithms. Finally, in Chapter 6 we give an overview of the main contributions in the dissertation, and provide promising directions for future research.

A mis papás, Mauricio y Ana Cristina.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my advisor and dissertation chair Alper Atamtürk. Through our many discussions he taught me how to approach research questions, and provided me with many pointers and suggestions that made this dissertation possible. He has also been a constant source of advice, guiding many decisions beyond the doctoral research. But, more importantly, Alper's passion for research and excellence was an inspiration for me during my Ph.D. Without such an advisor, I can only wonder whether I would have worked as hard as I did, or whether I would have enjoyed the process as much as I did.

I would like to give special thanks to professor Dorit Hochbaum. Few professors are as passionate and knowledgeable about discrete optimization as she is, and I am very grateful for her lectures, her support and her mentorship. I would also like to thank professor Peter Bartlett for his support and valuable lessons in statistics. Many thanks to all the faculty and staff that made my experience in Berkeley wonderful. Professor Phil Kaminsky was an exceptional department chair, always available and supportive for students. I thoroughly enjoyed my conversations with professor Candy Yano, both related to research and life. Professors Anil Aswani and Paul Grigas have been very supportive and helpful during my stay at UC Berkeley. I would like to give special thanks to Dr. Deepak Rajan, who was my integer optimization teacher and with whom I had many insightful conversations regarding my research. I am also grateful to the IEOR staff, and in particular to Anayancy Paz, who were always friendly and helpful during my Ph.D.

My time in California has been enhanced thanks to the friends and colleagues I met here. Thanks to my office mates Avinash, Birce, Chen and Hyemin for all the research discussions, coffees and beers that we shared. Thanks to my fellow Ph.D. students Auyon, Carlos, Matt, Quico, Salar, and Yonatan for all the drinks at Triple Rock, board games, soccer games and lunch(o)s/dinners we had over the years. I am grateful for the Colombian friends I met here, in particular to Angie, Ingrid, Juan Sebastián, and Lorena, who made me feel closer to the *tierrita*. I am immensely grateful to Nora and Doug Smith for providing for me a home away from home.

Finally, I would like to thank my friends and family from Colombia. In particular, I would like to thank Andru, Camilo, Juan, Julian, Nathalia and Pablo. And, off course, I would like to thank Ximena, who despite the distance, has been always present in my life.

All of this was possible thanks to the support and love of my father, mother and brother.

Chapter 1

Introduction

The goal of this dissertation is study the structure of conic quadratic problems with discrete variables, and develop new algorithmic tools for such problems. The dissertation consists of four parts. The first and second parts study the maximization and minimization of a conic quadratic objective, respectively, the third part focuses on structural insights for general mixed-binary optimization with second-order cone constraints, and the fourth part deals with binary quadratic optimization problems.

Over the past three decades there have been substantial advances in the capabilities of solving nonlinear and linear discrete optimization problems. On the nonlinear optimization front, using Newton method with self-concordant barrier functions has resulted in efficient solution approaches for constrained convex optimization. Second-order cone optimization (SOCO) in particular has received considerable attention due to its ubiquity in practice. Algorithms tailored for SOCO, including interior point methods, have been studied extensively in the literature, and large-scale SOCO are solved routinely today.

Linear discrete optimization has also been a major field of study in the Operations Research and Computer Science communities. Most discrete optimization problems are $\mathcal{NP}$-hard, and there have been two main approaches to tackle such problems. One approach is to prioritize solution speed and compromise on the quality of the solution found, resulting in approximation algorithms and heuristics. One technique in particular has been successful for designing approximation algorithms for many integer optimization problems. The techniques involves solving a convex relaxation of the problem and round the resulting fractional solution to a suitable integer solution.

The other approach for discrete optimization focuses on solving the problems to opti-

mality, typically resulting in some from of exhaustive enumeration and non-polynomial time algorithms. Among such approaches, branch-and-bound algorithms have been the most effective for solving general mixed-integer programs: by solving convex relaxation at each node of the search tree, they substantially reduce the number of solutions to be explored. State-of-the-art solvers for mixed-integer linear optimization (MILO) leverage the simplex method for linear programming to efficiently solve the linear programming relaxations, and use cutting planes based on the polyhedral structure of the feasible region to further enhance the performance of the algorithm. As a result of the application of these techniques, many optimization problems considered intractable 30 years ago can be solved within seconds today.

Despite the advances in nonlinear and discrete optimization, and despite many relevant applications in finance, machine learning and supply chain problems, our ability to solve mixed-integer nonlinear optimization (MINLO) is still limited. Most of the work to date focuses either on transforming MINLO problems into corresponding MILO problems, or on adapting techniques originally developed for MILO. Both approaches have drawbacks: transformations typically involve some loss of accuracy or require adding additional variables and constraints to the optimization model, increasing the difficulty to solve it; and many tools for MILO do not generalize naturally to MINLO due to the non-polyhedral structure induced by nonlinear functions. In this dissertation we study structures specific to nonlinear discrete optimization, and we address in particular the case of mixed-integer second order cone optimization (MISOCO), i.e., discrete optimization problems with conic quadratic constraints.

In Chapter 2 we study the problem of maximizing a conic quadratic function over a discrete set. This problem arises in a variety of applications, including project management and reinforcement learning. The problem is well understood if the discrete set is a matroid, but no efficient approaches are known for more complicated sets. We propose an approximation algorithm for the case where the discrete set corresponds to the vertices of a polytope. The algorithm first solves a convex relaxation of the optimization problem to find a solution in an edge of the polytope; then it rounds the fractional solution to a suitable vertex. The objective value of the solution found is within 20% of the optimal objective value, which is an improvement over the previous best bound of 37% for matroid polytopes. New methodologies in project scheduling and robust conic quadratic optimization are also proposed, using the approximation algorithm in settings where no other approaches exist in the literature.

Branch-and-bound solvers for mixed-integer linear optimization rely on the dual simplex

method and warm starts to efficiently solve the subproblems at each node of the branch-and-bound tree. Simplex-like methods are also available for convex quadratic problems, but have not been developed for second order conic optimization problems. In Chapter 3 we propose an algorithm with warm start capabilities for minimizing a conic quadratic function subject to linear constraints. The algorithm is suitable to solve the convex subproblems arising in branch-and-bound approaches. In the computational experiments the algorithm outperforms interior point methods when directly used to solve convex problems, and is faster by an order of magnitude in large instances. Moreover, when used with a branch-and-bound algorithm to solve discrete instances, the algorithm also outperforms commercial solvers, which use a polyhedral outer approximation.

Cutting plane methods are among the most effective tools for solving MILO problems to optimality. There is an increasing effort to develop strong formulations for MINLO, but current general purpose solvers for MINLO and MISOCO still do not use many valid inequalities to tighten the relaxations. In Chapter 4 we propose valid inequalities for the lower level set of a mixed-integer conic quadratic function. The inequalities completely describe the convex hull of the considered set when the quadratic term is separable. The inequalities are nonlinear in the original space of variables, but can be implemented as linear cuts in an extended formulation. The computational experiments indicate that the inequalities can improve current commercial solvers by many factors, even in instances with non-separable quadratic terms.

Another approach that has been used successfully in MILO and MINLO is to use extended formulations. By adding a polynomial number of variables and constraints, it is often possible to have stronger formulations than the ones obtained by adding a similar number of valid inequalities. In Chapter 5 we study extended formulations for binary quadratic optimization. We establish connections between classical linear formulations for quadratic problems, submodularity and the minimum cut problem. Then, by decomposing the quadratic functions into a convex component and a submodular component, and using extended formulations to represent the submodular component, we find stronger convex relaxations than alternatives proposed in the literature.

1.1 Notation

Throughout the paper, we use $\mathbb{B}$, $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ to denote the set of binary, natural, integer and real numbers. We use $\mathbb{R}_+$ to denote the nonnegative real numbers. Given a set N, we use $\mathbb{R}^N$ to denote the set of real vectors whose components are indexed by the elements of N. Similar notation is used for sets $\mathbb{Z}$ and $\mathbb{N}$. Given a set $S \subseteq N$ and $v \in \mathbb{R}^n$, we use $v(S)$ to denote $\sum_{i \in S} v_i$.

1.2 Mixed-integer optimization

Given functions $f : \mathbb{R}^{n+m} \to \mathbb{R}$ and $g_i : \mathbb{R}^{n+m} \to \mathbb{R}$, $i = 1, \ldots, \ell$, a mixed-integer optimization problem is an optimization problem of the form

$$\begin{array}{lll} & \min f(x,y) & \\ \text{(MIO)} & \text{s.t. } g_i(x,y) \leq 0, & i = 1, \ldots, \ell \\ & x \in \mathbb{Z}^n, y \in \mathbb{R}^m. & \end{array}$$

If $n = 0$ and all functions f, g_i are convex, then MIO is a *convex optimization problem*. Additionally, if all functions are affine, then MIO is a *linear program* (LP), and if f is a quadratic function and all functions g_i are affine, then MIO is a *quadratic problem* (QP). If $n > 0$ and all functions are affine, then MIO is a MILO, and if some of the functions are nonlinear, then we say that MIO is a MINLO. In this dissertation we focus in problems where all functions are convex. We now discuss some basic definitions and properties for MIO. Most of the material in this section is taken from Nemhauser and Wolsey (1988).

Definition 1. A set $T \subseteq \mathbb{R}^n$ is *convex* if for all $x^1, x^2 \in T$ we have that that $\lambda x^1 + (1-\lambda)x^2 \in T$ for all $0 \leq \lambda \leq 1$.

The set $X = \{(x,y) \in \mathbb{Z}^n \times \mathbb{R}^m : g_i(x,y) \leq 0,\ i = 1, \ldots, \ell\}$ is the *feasible region* of MIO. If $n = 0$, then the feasible region of MIO is convex.

The set $X_c = \{(x,y) \in R^{n+m} : g_i(x,y) \leq 0,\ i = 1, \ldots, \ell\}$ is the convex relaxation of X. We also say that the optimization problem $\min_{(x,y) \in X_c} f(x,y)$ is the convex relaxation of MIO. Since optimization problems over convex sets are in general easier than optimization over discrete sets, strategies to solve MIO often involve solving optimization problems over suitable convex relaxations.

Definition 2. Given a set $X \subseteq \mathbb{R}^n$, a point $x \in \mathbb{R}^n$ is a *convex combination* of points of X if there exists a finite set of points $\{x^i\}_{i=1}^t$ in X and a $\lambda \in \mathbb{R}_+^t$ such that $\sum_{i=1}^t \lambda_i = 1$ and $x = \sum_{i=1}^t \lambda_i x^i$. The *convex hull* of X, denoted $\text{conv}(X)$, is the set of all points that are convex combinations of points in X.

Clearly, for any set X, $\text{conv}(X)$ is convex by definition. As Proposition 1 states, optimization over a set is equivalent to optimization over the convex hull of the set. Thus, in principle, any discrete optimization problem can be transformed into a convex optimization problem.

Proposition 1 (Nemhauser and Wolsey (1988))**.** *Given $X \subseteq \mathbb{Z}^n \times \mathbb{R}^m$ and any $c \in \mathbb{R}^n$, $d \in \mathbb{R}^m$, we have that*

$$\min \{c'x + d'y : (x, y) \in X\} = \min \{c'x + d'y : (x, y) \in \mathit{conv}(X)\}.$$

An important class of problems that have received most of the attention in the literature are problems in which all constraints are linear.

Definition 3. A rational *polyhedron* is a set of the form $\{x \in \mathbb{R}^n : Ax \leq b\}$, where A is a finite rational matrix and b a finite rational vector.

Definition 4. A polyhedron P is bounded if there exists $M \in \mathbb{R}_+$ such that
$P \subseteq \{x \in \mathbb{R}^n : -M \leq x_i \leq M, \text{ for i=1,...,n}\}$. A bounded polyhedron is called a *polytope*.

Proposition 2 (Pulleyblank (1973))**.** *A polyhedron is a convex set.*

Proposition 3 (Nemhauser and Wolsey (1988))**.** *If $P \subseteq \mathbb{R}^{n+m}$ is a rational polyhedron and $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^m)$, then conv$(X)$ is a rational polyhedron.*

There is a rich theory of polyhedra that was developed to solve MILO. Moreover, the natural convex relaxations for MILO are LPs, which are very well understood and have many unique properties (e.g., duality, basic solutions). However, solving MINLO requires handling non-polyhedral sets, which are less understood and lack many of properties that solvers for MILO exploit.

1.3 Solution approaches for MIO

Most optimization problems with discrete variables are $\mathcal{NP}$-hard. Thus, unless $P = NP$, it is not possible to have an algorithm that finds optimal solutions in polynomial time. First we

cover approximation algorithms, which focus on solution speed while providing guarantees on the quality of the solutions. Then we cover the branch-and-bound algorithm, which is the most widely used algorithm to solve MIO to optimality. Finally we focus on cuts, a technique that is often paired with branch-and-bound algorithms to efficiently solve MIO.

Approximation algorithms

We now provide a brief overview of approximation algorithms. The reader is referred to Williamson and Shmoys (2011) and the references therein for an in-depth treatment of the subject.

Definition 5 (Williamson and Shmoys (2011))**.** An *α-approximation algorithm* for MIO is a polynomial-time algorithm that for all instances of the problem produces a solution with objective value within a factor of α of the objective value of the optimal solution.

The value α is called the *performance guarantee*, *approximation ratio* or *approximation factor* of the algorithm. We use the convention that $\alpha > 1$ for minimization problems and $0 < \alpha < 1$ for maximization problems. Thus, for a maximization problem, an α-approximation indicates that the objective value of the solution found by the algorithm is at least α times the optimal objective value.

The value α can be a function of the number of variables in MIO (e.g., $\alpha = \log n$ for minimization, or $\alpha = 1/n$ for maximization). In that case, the performance guarantee of the algorithm degrades as the size of the problem increases. In this dissertation, we are mainly concerned with algorithms where the performance guarantee does not depend on the number of variables (e.g., $\alpha = 1/2$). We say that such algorithms have *constant approximation ratio*.

An important technique in the design of approximation algorithms for MILO is to round the solution found by a suitable convex relaxation. The convex relaxation is often the natural LP relaxation (e.g., Hochbaum 1982, for the set cover problem), although other convex relaxations have also been used (e.g., Goemans and Williamson 1995, using a SDP relaxation for the max-cut problem). However, few rounding approximation algorithms based on convex relaxations have been proposed for MINLO. In Chapter 2 we propose such an algorithm.

Branch-and-bound algorithm

We now present a classic branch-and-bound algorithm for MIO. In the description of the algorithm, $\mathcal{L}$ is a collection of mixed-integer optimization problems $\left\{\text{MIO}^i\right\}$ of the form $\min_{x\in X^i} f(x)$, and associated with each problem in $\mathcal{L}$ is a lower bound $\underline{z}^i \leq z^i_{\text{MIO}}$, where z^i_{MIO} is the optimal objective value of MIO^i. Moreover, let MIO^i_c denote the convex relaxation of MIO^i. Finally, given a set X, we say that $\{X^j\}_{j=1}^k$ is a division of X if $\bigcup_{j=1}^k X^j = X$.

1. `Initialization` $\mathcal{L} \leftarrow \{\text{MIO}\}$, $X^0 \leftarrow X$, $\underline{z}^0 \leftarrow -\infty$, $\bar{z}_{\text{MIO}} \leftarrow \infty$.

2. `Termination test` If $\mathcal{L} = \emptyset$, then the solution x^0 that yielded $\bar{z}_{\text{MIO}}$ is optimal.

3. `Problem selection and relaxation` Select and delete MIO^i from $\mathcal{L}$. Solve its relaxation MIO^i_c. Let z^i_c be the optimal value of the relaxation and x^i_c be an optimal solution if one exists.

4. `Pruning` a. If $z^i_c \geq \bar{z}_{\text{MIO}}$, then go to `Termination test`.

   b. If $x^i_c \notin X^i$, then go to `Division`.

   c. If $x^i_c \in X_i$ and $z^i_c < \bar{z}_{\text{MIO}}$, then $\bar{z}_{\text{MIO}} \leftarrow z^i_c$. Delete all problems from $\mathcal{L}$ with $\underline{z}^i \geq \bar{z}_{\text{MIO}}$. Go to `Termination test`.

5. `Division` Let $\{X^{ij}\}_{j=1}^k$ be a division of X^i. Add problems $\left\{\text{MIO}^{ij}\right\}_{j=1}^k$ to $\mathcal{L}$, where $\underline{z}^{ij} = z^i_c$ for $j = 1, \ldots, k$. Go to `Termination test`.

There are different possible implementations for branch-and-bound algorithms. For MILO, the most common choice is to solve LP-relaxations at each node of the search tree, and to use dichotomies for `Division`: if x_ℓ corresponds to an integer variable and $x^i_{c\ell} = f$ with f fractional, then $X^{i,1} = X \cap \{x \in \mathbb{R}^{n+m} : x^i_{c\ell} \leq \lfloor f \rfloor\}$ and $X^{i,2} = X \cap \{x \in \mathbb{R}^{n+m} : x^i_{c\ell} \geq \lceil f \rceil\}$. Moreover, after solving the LP-relaxation MIO^i_c, the optimal basis is also stored. Then, using the simplex method with the basis as a warm start, optimization of the child problems $\left\{\text{MIO}^{ij}_c\right\}_{j=1}^k$ can be done efficiently.

There is no "typical" branch-and-bound algorithm for MINLO. In particular, there is no consensus on the best convex relaxation to use. One approach is to use the natural convex relaxation, obtained by relaxing the integrality constraints. However, due to the absence of the simplex method or a similar method with warm starts for nonlinear optimization, such approach results in high computational times. An alternative is to use a linear outer

approximation instead of the natural convex relaxation. This approach can use the simplex method, but the relaxations are weaker, resulting in less effective pruning and a larger search tree; moreover, additional steps to refine the approximation need to be incorporated in the branch-and-bound algorithm. In Chapter 3 we show that for a class of SOCO an simplex-based algorithm can be used, resulting in both strong relaxations and fast solution times.

Cuts and strong formulations

Strong convex relaxations are critical to the performance of branch-and-bound algorithms. With stronger relaxations, the algorithm can prune more effectively and less subproblems need to be solved. In particular, if the convex hull of the feasible region of MIO is known and the objective function is linear, then a branch-and-bound algorithm would require solving a single convex problem. Therefore, branch-and-bound algorithm are often paired with techniques that dynamically strengthen the convex relaxations.

Definition 6. The inequality $h(x) \leq 0$ is called a *valid inequality* for X if it is satisfied by all points in X.

There is a rich theory of valid inequalities for the case where X is a polyhedron. Since in that case $\text{conv}(X)$ is a polyhedron, it is sufficient to study linear valid inequalities. The valid inequalities required to describe $\text{conv}(X)$ are called *facets* or *facet-defining* inequalities. Moreover, there are a number of ways to identify and systematically generate facet-defining inequalities. Typically, an exponential number of *facet-defining* inequalities is required to describe the convex hull of X. Nevertheless, optimization over $\text{conv}(X)$ can be done in polynomial time whenever the *separation problem* can be solved in polynomial time (Grötschel et al. 1981).

Definition 7 (Separation)**.** Given a set $X \subseteq \mathbb{R}^n$ and a point $\bar{x} \in \mathbb{R}^n$, the *separation problem* consists in deciding whether $\bar{x} \in X$ and, if $\bar{x} \notin X$, producing a valid inequality for X such that $h(\bar{x}) > 0$.

The reader is referred to Nemhauser and Wolsey (1988) and the references therein for an in-depth treatment of valid inequalities for polyhedra.

In most practical applications, there is not a complete description of $\text{conv}(X)$ available, and the separation problem cannot be solved exactly. Nevertheless, adding valid inequalities may significantly improve the strength of the convex relaxation. Thus, most commercial

branch-and-bound algorithms for MILO generate valid inequalities on the fly as *cuts*. A common approach to decide which inequalities to use consists in adding the most violated inequalities: given a family of valid inequalities $\{h_j(x) \leq 0\}_{j \in J}$ available to the algorithm and a fractional point $\bar{x} \notin X$, a most violated inequality $h_j(x) \leq 0$ can be found by solving the optimization problem

$$\max_{j \in J} h_j(x). \tag{1.1}$$

Since $|J|$ is often exponential in the number of variables, problem (1.1) may be $\mathcal{NP}$-hard and, in that case, it is typically solved using heuristics.

Using valid inequalities has resulted in a considerable improvement of branch-and-bound algorithms (Bixby 2012). However, since non-polyhedral sets arising in MINLO are not well understood, solvers for MINLO currently use only a limited number of valid inequalities. We address this problem in Chapter 4, and propose valid inequalities for general conic quadratic constraints.

1.4 Submodular functions

Submodularity is an important concept in discrete optimization, akin to convexity or concavity in continuous optimization.

Definition 8. Given a finite set $N = \{1, \ldots, n\}$, and a real-valued function f on the subsets of N,

1. f is *nondecreasing* if $f(S) \leq f(T)$ for $S \subseteq T \subseteq N$.
2. f is *monotone* if either f or $-f$ is nondecreasing.
3. f is *submodular* if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for all $S, T \subseteq N$.

Submodular function can also be characterized as functions that exhibit diminishing returns.

Proposition 4 (Nemhauser et al. (1978))**.** *f is submodular if and only if*

$$f(S \cup \{j\}) - f(S) \geq f(T \cup \{j\}) - f(T), \quad \textit{for all } j \in N \textit{ and } S \subseteq T \subseteq N \setminus \{j\}.$$

Edmonds (1970) was the first to formally study the properties of submodular functions. In particular, he established a correspondence between submodular functions and a class of polyhedra.

Definition 9. Given a submodular function f on N with $f(\emptyset) = 0$, the polyhedron

$$EP(f) = \left\{ \pi \in \mathbb{R}^n : \sum_{j \in S} \pi_j \leq f(S) \text{ for } S \subseteq N \right\}$$

is the *extended polymatroid* associated with f, and let $\Pi(f)$ denote the set of extreme points of $EP(f)$.

Edmonds (1970) shows that optimization of linear functions over extended polymatroids, i.e.,

$$\max_{\pi \in EP(f)} c'\pi, \tag{1.2}$$

can be solved by the greedy algorithm.

Proposition 5 (Edmonds (1970))**.** *Assume without loss of generality that $c_1 \geq c_2 \geq \ldots \geq c_n \geq 0$ (if $c_n < 0$ then the problem is unbounded), and let $S^j = \{1, \ldots, j\}$ with $S^0 = \emptyset$. Then an optimal solution π^* of (1.2) is $\pi_j = f(S^j) - f(S^{j-1})$ for $1 \leq j \leq n$.*

From Proposition 5 we obtain a characterization of $\Pi(f)$. In particular $\pi \in \Pi(f)$ if and only if $\pi_i = f(S^j) - f(S^{j-1})$, where $S^{(0)} = \emptyset$ and $S^j = \{(1), \ldots, (j)\}$ for some permutation $((1), (2), \ldots, (n))$ of N.

Now consider the convex lower envelope of a submodular function f, given by $\text{conv}(K)$ where $K = \{(x, t) \in \{0, 1\}^n \times \mathbb{R} : f(x) \leq t\}$. Since there is a natural correspondence between the extended polymatroid $EP(f)$ and the set of valid inequalities for K (Atamtürk and Narayanan 2008), it follows that the convex hull of K is described by bound constraints and inequalities corresponding to the extreme points of the extended polymatroid.

Corollary 1. *Given any submodular function f and $K = \{(x, t) \in \{0, 1\}^n \times \mathbb{R} : f(x) \leq t\}$,*

$$conv(K) = \{(x, t) \in [0, 1]^n \times \mathbb{R} : \pi' x \leq t, \ \forall \pi \in \Pi(f)\}.$$

Since the work of Edmonds (1970), submodularity has played a key role in the design of algorithms for combinatorial problems. In particular, there are numerous polynomial time algorithms for unconstrained submodular minimization (Schrijver 2000, Iwata et al. 2001, Grötschel et al. 2012, Orlin 2009, Iwata and Nagano 2009), and a number of approximation algorithms for maximization of a monotone submodular function over matroid constraints (Nemhauser et al. 1978, Fisher et al. 1978, Calinescu et al. 2011) and knapsack

constraints(Sviridenko 2004, Kulik et al. 2009), and the maximization of non-monotone submodular functions (Buchbinder et al. 2012). Moreover, submodularity has also been used to find strong formulations for $\mathcal{NP}$-hard optimization problems (Atamtürk and Narayanan 2008, 2009, Ahmed and Atamtürk 2011, Atamtürk and Bhardwaj 2015, Atamtürk and Bhardwaj 2017, Zhang et al. 2017).

We conclude this section by providing some classes of submodular functions.

Proposition 6 (Nemhauser et al. (1978))**.**

1. *Affine functions, i.e., $f(S) = c_0 + \sum_{i \in S} c_i$, are submodular.*
2. *Given $f_1, \ldots, f_k$ submodular functions on N and nonnegative $\alpha_1, \ldots, \alpha_k$, the function $h(S) = \sum_{i=1}^{k} \alpha_i f_i(S)$ is submodular.*
3. *Given a monotone submodular function f and a concave function $g : \mathbb{R}_+ \to \mathbb{R}$, the function $h(S) = g(f(S))$ is submodular.*

In particular, Proposition 6 implies that functions of the form $h(S) = \sum_{i \in S} a_i + g\left(\sum_{i \in S} c_i\right)$ with $a \in \mathbb{R}^N$, $c \in \mathbb{R}^N_+$ and g concave are submodular. In Chapters 2 and 4 we propose different techniques for optimization problems with functions of that form.

Chapter 2

Approximations for conic quadratic maximization

2.1 Introduction

For a rational polytope $X \subseteq \mathbb{R}^n$, let $V_X \subseteq X$ denote the set of vertices of X. We consider the discrete optimization problem

$$\max_{x \in V_X} f(x) := c'x + g(d'x), \tag{2.1}$$

where c and d are rational vectors in $\mathbb{R}^n$ and $g : \mathbb{R} \to \mathbb{R}$ is a monotone concave function. We refer to f as the utility function. Problem (2.1) includes combinatorial optimization problems, where X is an integral polytope, e.g., trees, flows, matchings, with an objective function that exhibits diminishing returns. The concave utility function f is often used to model probabilistic objectives.

In Section 2.2 we present five applications of problem (2.1) involving different utility functions. The first one is in modeling reliability of a parallel system, where g takes a negative exponential form. The second application is in the assortment planning application, where g is the multinomial logit probability function. The third application is on estimating project duration with stochastic task durations. We propose an improvement over PERT (project evaluation and review technique) by solving a maximum value-at-risk problem of the form (2.1) to determine a critical path, where g is the square root function. As the fourth application, we present a class of robust conic quadratic optimization problems, where g is the square root function. Finally, for the fifth application, we present a combinatorial

multi-armed bandit algorithm, and g is again the square root function.

Connections to submodularity

Independently, the square root function also arises in approximating submodular functions. Given a non-negative, monotone, submodular function h, which is accessible through either a value oracle or random sampling, Goemans et al. (2009) and Balcan and Harvey (2010) consider constructing an approximation $\hat{h}$ such that $\hat{h}(S) \leq h(S) \leq \alpha\hat{h}(S)$ for $S \subseteq \{1, \cdots, n\}$[1]. They give a construction of a function of the form $\hat{h}(S) = \sqrt{\sum_{i \in S} d_i}$ with an approximation ratio of $O(\sqrt{n} \log n)$ that uses a polynomial number of queries to h, and show that the approximation ratio is close to the best possible.

Note that when $V_X \subseteq \{0,1\}^n$ and $d \geq 0$, the objective function f is submodular. Maximization of submodular functions over special structured binary polytopes has received considerable attention. Fisher et al. (1978) show that when X is a matroid polytope and the objective is any non-negative, monotone submodular function accessible through a value oracle, the *greedy* algorithm yields a 1/2-approximation, and Nemhauser et al. (1978) prove that the approximation ratio of the greedy algorithm is $(1 - e^{-1})$ if X is the uniform matroid. Other $(1 - e^{-1})$-approximation algorithms have been given when the objective is non-negative, monotone and X is either a down-monotone polytope defined with a single knapsack constraint (Sviridenko 2004) or an arbitrary matroid (Calinescu et al. 2011), and a $(1 - e^{-1} - \epsilon)$-approximation algorithm is known for the case when X is a down-monotone polytope defined with multiple knapsack constraints (Kulik et al. 2009). For a non-monotone objective, Buchbinder et al. (2012) give a 1/2-approximation algorithm for the unconstrained case, and Vondrák et al. (2011) give a 0.325-approximation algorithm for down-monotone polytopes. Ahmed and Atamtürk (2011) study the polytope induced by problem (2.1) when $V_X = \{0,1\}^n$, $d \geq 0$ and g is strictly concave and increasing, and use submodularity to derive a strong formulation for exact algorithms. Atamtürk and Narayanan (2009) give valid inequalities for the lower level set of non-decreasing f, whereas Atamtürk and Bhardwaj (2015) give valid inequalities for the lower level set of non-increasing f.

In this work we do not make use of submodularity, but exploit the structure of the function f to derive an approximation algorithm for *any* polytope X. The algorithm rounds an optimal solution to the continuous relaxation $\max_{x \in X} f(x)$ to a vertex of X. When $c'x$

[1]When h is accessible through sampling, the condition $\hat{h}(S) \leq h(S) \leq \alpha\hat{h}(S)$ needs to hold only in most of the sets with high probability.

and $g(d'x)$ are non-negative on V_X and g is monotone, we show that the gap of the continuous relaxation is at most 100%, and the approximation ratio of the algorithm is 1/2. Moreover, when g is the square root function, the gap is at most 25%, and the approximation ratio improves to 4/5. Both of these bounds are tight.

Exploiting the structure of the utility function f leads to a number of advantages compared to relying on submodularity alone. For the square root case and monotone f, the approximation ratio of 4/5 is better than $(1 - e^{-1}) \approx 0.63$. In the general case, the proposed approximation algorithm can be used with arbitrary polytopes. We give examples with path and assignment polytopes, which are neither down-monotone nor matroids. Moreover, we do not require that f be monotone – even if g is monotone, the function f may be non-monotone – and the approximation ratios of 4/5 for the square root or 1/2 for the general case are better than 0.325. Moreover, unlike approximation algorithms based on submodularity, we also get a tight upper bound on the optimal objective value (which is used in the robust conic quadratic optimization application discussed in Section 2.2).

The chapter is organized as follows. In Section 2.2 we give applications in reliability modeling, assortment planning, in estimating project duration with stochastic times ,in robust optimization and reinforcement learning that motivate problem (2.1). In Section 2.3 we prove $\mathcal{NP}$-hardness of (2.1) for simple polytopes. In Section 2.4 we give a high-level description of the approximation algorithm and show that the gap of the continuous relaxation of problem (2.1) is tight. In Section 2.5 we propose efficient implementations of the proposed approximation algorithms, and compare with alternatives found in the literature. In Section 2.6 we illustrate the empirical performance of the approximation algorithms through computational experiments for varying utility functions and polytopes. In Section 2.7 we conclude the chapter with a few final remarks.

2.2 Applications

Reliability modeling

Given a parallel system with components N, where each component has an independent failure/malfunction probability q_i, $i \in N$, the reliability of the system is the probability that not all components malfunction simultaneously, i.e., $1 - \prod_{i=1}^{n} q_i$.

Now given a set of candidate components N with revenue r_i and malfunction probability q_i, $i \in N$, consider the problem of finding a subset $S \subseteq N$ with a revenue and reliability

tradeoff. Letting $x_i = 1$ if component i is selected and 0 otherwise, the problem can be formulated as

$$\begin{aligned}&\max_{x\in V_X} \sum_{i\in N} r_i x_i + \beta \left(1 - \prod_{i\in N} q_i^{x_i}\right)\\ =&\max_{x\in V_X} \sum_{i\in N} r_i x_i + \beta \left(1 - \exp\Big(\sum_{i\in N} x_i \ln(q_i)\Big)\right),\end{aligned} \tag{2.2}$$

where $\beta > 0$ is the weight given to the reliability and X is an integral 0-1 polytope with additional restrictions (e.g. a cardinality constraint). Note that (2.2) is a special case of problem (2.1), where $g(d'x) = \beta\,(1 - \exp(-d'x))$ is the negative exponential function and $d_i = -\ln(q_i)$.

Problems of the form of (2.2) arise when considering how to allocate resources either to defend a parallel system from threats such as terrorist attacks or cyber attacks, or to attack a series system to ensure that the system is disrupted. Bier et al. (2005) and Hausken (2008) study continuous versions of such problems, in which they derive the Karush-Kuhn-Tucker conditions explicitly, and Levitin and Hausken (2008) consider a discrete version when all components are identical. Formulations similar to (2.2) also arise as substructures of more complicated systems (Ahmed and Papageorgiou 2013, Gen and Yun 2006).

Assortment with multinomial logit choice model and fixed costs

Given a set of products N, consider the problem of choosing an assortment $S \subseteq N$ satisfying a set of constraints so as to maximize the profit. In this context consumer preferences are commonly modeled with a multinomial logit (MNL) choice model (Van Ryzin and Mahajan 1999, Chong et al. 2001). In the MNL choice model the utility of products $i \in N$ are modeled as $u_i = \mu_i + \zeta_i$, where $\mu_i \in \mathbb{R}$ is a known parameter, and ζ_i are i.i.d. standard Gumbel random variables. For an assortment S, the probability that a customer purchases item $i \in S$ is given by

$$p_i(S) = \frac{e^{\mu_i}}{1 + \sum_{j\in S} e^{\mu_j}}.$$

Consider an online advertiser that earns a variable profit c_i for displaying ad $i \in N$ as well as constant profit β if the ad is clicked. Constant profit margins have been considered by Van Ryzin and Mahajan (1999) among others. Then, an optimal set of up to k ads to

display in order to maximize the expected profit is formulated as

$$\max_{S:\ |S|\leq k} \sum_{i\in S}(c_i + \beta p_i(S)).$$

More generally, the problem is stated as

$$\max_{x\in V_X} c'x + \beta\frac{d'x}{1+d'x}, \tag{2.3}$$

which is a special case of problem (2.1) with $g(d'x) = \beta\frac{d'x}{1+d'x}$.

Various versions of the assortment problem with multinomial logit choice model have been studied in the literature (Rusmevichientong et al. 2010, Sen et al. 2015). For many of these the constraint set X_V is the uniform matroid (corresponding to maximum cardinality constraint). Note that formulation (2.3) allows, among others, differential pricing and probabilities according to the position of the ad when X is the assignment polytope.

PERT and VaR critical paths

Given a set of activities V with duration $\ell_i \geq 0$ for $i \in V$ and a set A of dependencies between pairs of activities, the Critical Path Method (CPM) computes the completion time of the overall project by finding a longest path on the directed acyclic graph $G = (V, A)$. The duration of the project is the length of the longest path, referred to as the *critical path*, between a source node s representing the start time of the first activity and a destination node t representing the completion of the last activity. The activities on the critical path are monitored carefully as delays in those activities result in delays in the completion time of the project.

The Project Evaluation and Review Technique (PERT) is a simple and prevalent generalization of CPM to incorporate the uncertainty in activity durations. The traditional PERT computes the deterministic critical path using the expected duration of the activities, and then makes an assessment of the probability of completion time of the project based on the deterministic critical path (e.g. Nahmias 2001, Chapter 9). Since the uncertainty is not taken into account when computing the deterministic critical path, the estimation of PERT can be poor. In order to incorporate the uncertainty, we consider an alternative where we compute a path with maximum value-at-risk.

The Value-at-Risk at confidence level $\alpha \in (0.5, 1)$ of duration D is

$$\mathrm{VaR}_\alpha(D) = \sup\{\ell \in \mathbb{R} : \mathbf{Pr}(D \leq \ell) \leq \alpha\}. \tag{2.4}$$

Observe that the value-at-risk of any path is at most the value at risk of the project. Let $\mathcal{P}$ be the set of feasible paths, let D_p be the duration of a path $p \in \mathcal{P}$ and let $D^* = \max_{p\in\mathcal{P}} D_p$ be the duration of the project. Since $\mathbf{Pr}(D^* \le \ell) \le \mathbf{Pr}(D_p \le \ell)$ for any $p \in \mathcal{P}$, we have

$$\mathrm{VaR}_\alpha(D^*) = \sup\{\ell \in \mathbb{R} : \mathbf{Pr}(D^* \le \ell) \le \alpha\} \ge \sup\{\ell \in \mathbb{R} : \mathbf{Pr}(D_p \le \ell) \le \alpha\} = \mathrm{VaR}_\alpha(D_p).$$

A path with the largest value-at-risk at confidence level α thus provides the best lower bound on the value-at-risk of the project. We call such a path a VaR_α-critical path.

Let $x_{ij} = 1$ if arc $(i,j) \in A$ belongs to the path and 0 otherwise, $\delta^+(i)$ denote the set incoming arcs to i and $\delta^-(i)$ denote the set of outgoing arcs from i. If the durations are independent and normally distributed with mean μ_{ij} and variance σ_{ij}^2, then the VaR_α-critical path corresponds to the optimal solution to

$$\begin{aligned}
&\max \sum_{(i,j)\in A} \mu_{ij}x_{ij} + \Phi^{-1}(\alpha)\sqrt{\sum_{(i,j)\in A} \sigma_{ij}^2 x_{ij}} \\
&\text{s.t.} \sum_{(j,i)\in\delta^+(i)} x_{ji} - \sum_{(i,j)\in\delta^-(i)} x_{ij} = \begin{cases} -1 & \text{if } i = s \\ 1 & \text{if } i = t \\ 0 & \text{if } i \in V \setminus \{s,t\} \end{cases} \\
&\quad x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A,
\end{aligned} \tag{2.5}$$

where Φ is the c.d.f of the standard normal distribution. Problem (2.5) is a case of problem (2.1), where the objective is submodular but the feasible region is neither down-monotone nor a matroid.

Example

Figure 2.1 illustrates a network with four activities. Using the traditional PERT method, we find that the (deterministic) critical path is given by activities $(s,2)$ and $(2,t)$, and the duration of the project is estimated to be exactly 2 at any confidence level. On the other hand, for $\delta \ge 0.08$, the VaR-critical path is given by activities $(s,3)$ and $(3,t)$, with corresponding value-at-risk $1.8 + 2.77\delta$ at 97.5% confidence level. The $\mathrm{VaR}_{0.975}$-critical path provides a better assessment of the risk of the project duration.

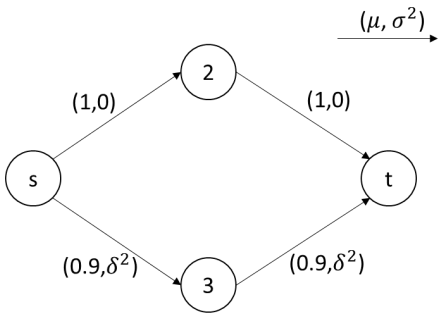


Figure 2.1: PERT network with four activities.

Robust optimization with conic quadratic objective

Given a feasible region $Z \subseteq \mathbb{R}^p$ and matrix $Q \in \mathcal{S}_+^p$, consider the problem with the conic quadratic objective

$$\min_{z \in Z} c'z + \sqrt{z'Qz}. \tag{2.6}$$

Problem (4.7) arises when minimizing value-at-risk as a mean-risk objective (e.g. Ahmed 2006, Atamtürk and Narayanan 2008). The reader is referred to Lobo et al. (1998) and Alizadeh and Goldfarb (2003) for other applications conic quadratic optimization.

We describe a robust formulation of problem (4.7), generalizing the approach given by Bertsimas and Sim (2003, 2004) for linear objectives. Let c_0 and $Q_0 \succeq 0$ be the nominal mean and covariance and $N = \{1, \ldots, n\}$ be a set of potential events, each of which may increase the mean and covariance by $c_i \geq 0$ and $Q_i \succeq 0,\ i \in N$. For example, Q_i may represent the increase in volatility and correlations in financial markets for a particular stress scenario. Then $c(S) = c_0 + \sum_{i \in S} c_i$ and $Q(S) = Q_0 + \sum_{i \in S} Q_i$ are the mean and covariance when events $S \subseteq N$ are realized. The goal of the robust optimization is to find a solution that minimizes the worst objective given that only a small number, $k \leq n$, events are realized, i.e.,

$$\min_{z \in Z} \max_{S \subseteq N : |S| \leq k} c(S)'z + \sqrt{z'Q(S)z}. \tag{2.7}$$

Letting

$$X = \left\{ x \in [0,1]^n : \sum_{i=1}^{n} x_i \leq k \right\}, \tag{2.8}$$

problem (4.37) can be equivalently stated as

$$\min_{z\in Z}\max_{x\in V_X} c_0'z + \sum_{i\in N}(c_i'z)\,x_i + \sqrt{z'Q_0z + \sum_{i\in N}(z'Q_iz)\,x_i}. \tag{2.9}$$

Observe that for a given value of z the inner maximization problem is of the form (2.1) with $g(x) = \sqrt{\xi + x}$, where $\xi \geq 0$ does not depend on x.

Combinatorial multi-armed bandits

The combinatorial multi-armed bandit problem is a sequential decision problem. Given a set $N = \{1, 2, \cdots, n\}$ of items, each with a stochastic weight with expected value w_i and support in $[0, 1]$, the goal is to choose, at each round $t = 1, 2, \cdots$, at set $S^t \subseteq N$ satisfying a set of constraints such that $\sum_{i\in S^t} w_i$ is maximized. If the expected weight vector w is known a priori, then the best stategy is to choose $S^* = \arg\max_S \sum_{i\in S} w_i$ at each round. We consider the case when the expected weights are initially unknown, and at each stage of the sequential problem, the decision maker plays a feasible $S^t \subseteq N$ and observes realizations $\tilde{w}_i^t$ for $i \in S^t$. A policy for this problem is evaluated in terms of the cumulative regret $R(T)$ with respect to choosing, at each stage, the best set in hindsight:

$$R(T) = \sum_{t=1}^{T} r^t,$$

where $r^t = \sum_{i\in S^*} w_i - \sum_{i\in S^T} w_i$ is the regret in round t.

Let x_i^t be 1 if item i is played at round t and 0 otherwise, $d_i^t = \sum_{\tau=0}^{t-1} x_i^\tau$ be the total number of times item i has been played by round t, $\tilde{w}_i^t$ be the random realization of item i at round t, $\hat{w}_i^t = \frac{1}{d_i^t}\sum_{\tau=0}^{t-1} \tilde{w}_i^\tau x_i^\tau$ be the observed sample average of the weight of item i by round t. Given $0 < \delta < 1$, Algorithm 1 describes a policy for the combinatorial multi-armed bandit. The policy is an adaptation of the *Confidence Ball 2* policy of Dani et al. (2008) and of the *Uncertainty Ellipsoid* policy of Rusmevichientong and Tsitsiklis (2010).

The policy uses the principle of optimism under uncertainty to handle the exploration and exploitation tradeoff: at each round t, we mantain an unbiased estimate $\hat{w}^t$ of the unknown weight vector w. A natural confidence region for w is the ellipsoid

$$B^t = \left\{\nu \in \mathbb{R}^n : \sum_{i=1}^{n} (\hat{w}_i^t - \nu_i)^2 d_i^t \leq \beta_t\right\}$$

centered on $\hat{w}^t$, in which the length of each axis is inversely proportional to the number of times we have sampled the corresponding item. Then, at each stage, we play the set that has the most optimistic weight $\nu \in B^t$.

Algorithm 1 Multi-armed bandit policy.

Input: $0 < \delta < 1$, probability; N, item set of cardinality n.

initialization:
$t \leftarrow 1$
$\forall i \in N: \; x_i^0 \leftarrow 1, \tilde{w}_i^0 \leftarrow 1$ ▷ $\hat{w}_i^1 = 1, d_i^1 = 1$

loop
  $\beta_t \leftarrow \max\left\{64n(\ln t)\ln(t^2/\delta), n\left(\frac{8}{3}\ln(t^2/\delta)\right)^2\right\}$.
  $B^t = \{\nu \in \mathbb{R}^n : \sum_{i=1}^n (\hat{w}_i^t - \nu_i)^2 d_i^t \leq \beta_t\}$
  Play $S_t = \arg\max_S \max_{\nu \in B_t} \sum_{i \in S} \nu_i \Leftrightarrow \arg\max_S \sum_{i \in S} \hat{w}_i^t + \sqrt{\beta_t \sum_{i \in S} \frac{1}{d_i^t}}$
  Observe $\tilde{w}$, update d^{t+1} and $\hat{w}^{t+1}$
  $t \leftarrow t + 1$.
end loop

Using an analysis similar to Dani et al. (2008), we can prove upper bounds on the regret after T iterations.

Theorem 1 (Problem independent upper bound)**.** *The regret R(T) of Algorithm 1 is with high probability at most $O^*(n\sqrt{T})$, where the O^* notation hides a polylogarithmic dependence on T. More precisely,*

$$P\left(\forall T, R(T) \leq 2\sqrt{Tn\beta_T \ln T}\right) \geq 1 - \delta.$$

Theorem 2 (Problem dependent upper bound)**.** *Let Δ be the regret of the best suboptimal feasible solution. The regret R(T) of Algorithm 1 is then with high probability at most $O(\frac{n^2}{\Delta}\ln^3 T)$. More precisely,*

$$P\left(\forall T, R(T) \leq \frac{4n\beta_T \ln T}{\Delta}\right) \geq 1 - \delta.$$

For the sake of completeness, we include a proof of the high probability bounds in Appendix A.1.

2.3 Complexity

In this section we show that problem (2.1) is $\mathcal{NP}$-hard for the uniform matroid polytope, the path polytope, and the assignment polytope. Note that the results are valid even when g is given explicitly.

Proposition 7. *For $c \geq 0$, $d \geq 0$ and any strictly concave function g, problem*

$$\max_{x\in\{0,1\}^n} \left\{ c'x + g(d'x) : \sum_{i=1}^{n} x_i = k \right\} \tag{2.10}$$

is $\mathcal{NP}$-hard.

Proof. Ahmed and Atamtürk (2011) show that the *unconstrained* problem

$$\max_{x\in\{0,1\}^n} -c'x + g(d'x) \tag{2.11}$$

is $\mathcal{NP}$-hard when g is the negative exponential function. We first extend their proof for *any* strictly concave function. The proof is by reduction from PARTITION: Given positive numbers c_i, $i \in N$, PARTITION calls for a set $S \subseteq N$ such that $\sum_{i\in S} c_i = \sum_{i\in N\setminus S} c_i$. If $\sum_{i\in N} c_i = 0$, then $S = \emptyset$ is a trivial answer. Otherwise, let $y^* = \arg\max_{y\in\mathbb{R}} \{g(y) - g(1)y\}$, which can be computed in polynomial time since it is the solution of a convex problem (note that if g is differentiable, y^* can be computed in closed form). If necessary, by negating and/or scaling the data, we may assume without loss of generality that $\sum_{i\in N} c_i = 2y^*$. Then, there exists $S \subseteq N$ with the desired property if and only if

$$\max_{x\in\{0,1\}^n} -g(1)c'x + g(c'x) = -g(1)y^* + g(y^*),$$

which is true if and only if $c'x = y^*$.

Next we show that (2.10) is $\mathcal{NP}$-hard. Given an instance of (2.11), let $\bar{c} = \max_{i\in N} c_i$. Then problem (2.11) can be written as

$$\begin{aligned}
&\max_{S\subseteq N} -\bar{c}\,|S| + \sum_{i\in S}(\bar{c} - c_i) + g\left(\sum_{i\in S} d_i\right) \\
&= \max_{k=0,\cdots,n} \left(-\bar{c}k + \max_{x\in\{0,1\}^n} \left\{ (\bar{c} - c)'x + g(d'x) : \sum_{i=1}^{n} x_i = k \right\} \right),
\end{aligned}$$

which can be solved by solving n problems of the form (2.10). □

We will prove $\mathcal{NP}$-hardness of problem (2.1) for special polytopes by reduction from problem (2.10).

Uniform matroid polytope.

The uniform matroid polytope is given by

$$X = \left\{ x \in [0,1]^n : \sum_{i=1}^{n} x_i \leq k \right\}, \tag{2.12}$$

with $k \in \mathbb{N}$.

Proposition 8. *Problem* (2.1) *is $\mathcal{NP}$-hard when X is the uniform matroid polytope and g is strictly concave.*

Proof. Observe that if $c, d \geq 0$, g is non-decreasing and X is the uniform matroid polytope, then any optimal solution x^* satisfies

$$\sum_{i=1}^{n} x_i = k \tag{2.13}$$

and is also an optimal solution for (2.10). In general, given an instance of (2.10), let

$$\bar{d} = \min_{j=1,\ldots,n} \left\{ c_j + g\left(\sum_{i=1}^{n} d_i \right) - g\left(-d_j + \sum_{i=1}^{n} d_i \right) \right\}$$

and consider the optimization problem

$$\max_{x \in \{0,1\}^n} \left\{ \sum_{i=1}^{n} (c_i - \bar{d}) x_i + g\left(\sum_{i=1}^{n} d_i x_i \right) : \sum_{i=1}^{n} x_i \leq k \right\}. \tag{2.14}$$

Observe that any optimal solution x^* of (2.14) satisfies (2.13) by construction. Moreover, the objective value of any solution satisfying (2.13) is $c'x + g(d'x) - \bar{d}k$ (where $\bar{d}k$ is a constant), and therefore we see that x^* is an optimal solution of (2.10).

□

Path polytope.

The path polytope on an acyclic directed graph is described in (2.5).

Proposition 9. *Problem* (2.1) *is $\mathcal{NP}$-hard when X is the path polytope and g is strictly concave.*

Proof. Consider an instance of problem (2.10). We construct an equivalent path instance on a directed graph $G = (V, A)$ with $O(n^2)$ vertices and arcs. Let $V = \{(i, j) \in \mathbb{Z}^2 : 1 \leq i \leq n + 1, 0 \leq j \leq k\}$ be the set of vertices. G has an arc for the vertex pair (i, j), $(i + 1, j)$ with value $(0, 0)$, which corresponds to setting $x_i = 0$ in problem (2.10); G has an arc for pair (i, j), $(i + 1, j + 1)$ with value (c_i, d_i), which corresponds to setting $x_i = 1$ in problem (2.10). Note that a path between $(1, 0)$ and (i, j) corresponds to a choice of $x_1, \ldots, x_{i-1}$ such that $\sum_{l=1}^{i-1} x_l = j$, thus paths between $(1, 0)$ and $(n + 1, k)$ correspond to feasible solutions of problem (2.10). Therefore, we have that the set of optimal solutions of problem (2.10) correspond to set of longest paths between $(1, 0)$ and $(n + 1, k)$.

□

Assignment polytope.

The $m \times m$ assignment polytope is given by

$$X = \left\{ x \in \mathbb{R}^{m \times m} : \sum_{i=1}^{m} x_{ij} = 1, \sum_{j=1}^{m} x_{ij} = 1, x_{ij} \geq 0 \right\}.$$

Proposition 10. *Problem* (2.1) *is $\mathcal{NP}$-hard when X is the assignment polytope and g is strictly concave.*

Proof. Consider an instance of problem (2.10). We construct an equivalent $n \times n$ assignment instance. Let x_{ij} have objective values (c_j, d_j) for $i = 1, \cdots, k$, and values $(0, 0)$ for $i = k + 1, \cdots, n$. Note that given any assignment we can construct a feasible solution $\bar{x}$ to problem (2.10) with same objective value: set $\bar{x}_j = 1$ if $x_{ij} = 1$ for some $i \leq k$, and set $\bar{x}_j = 0$ otherwise. Moreover, any feasible solution to (2.10) corresponds to at least one assignment by construction. Therefore, we get that an optimal solution to (2.10) can be obtained by finding an optimal assignment. □

2.4 Approximation analysis

In this section we discuss approximation algorithms for problems (2.1) and (2.9) that use the continuous relaxation

$$\max_{x \in X} c'x + g(d'x). \tag{2.15}$$

First we describe an algorithm for the maximization problem (2.1), next we describe the approach used for the robust problem (2.9), then we prove constant approximation ratios for both methods, and finally we give a simple extension for unbounded polyhedra.

Approximation algorithm

The approximation algorithm for (2.1) consists of rounding an optimal solution to the continuous relaxation (2.15) to particular extreme points X if the continuous optimal solution is not an extreme point itself.

Proposition 11. *If there exists an optimal solution to problem* (2.15)*, then there exists an optimal solution on an edge of* X*.*

Proof. Let x_0 be an optimal solution to (2.15). Consider the linear optimization problem

$$\begin{aligned} &\max c'x \\ &\text{s.t. } d'x = d'x_0 \\ &\quad x \in X. \end{aligned} \tag{2.16}$$

Observe that $\bar{X} = X \cap \{x \in \mathbb{R}^n : d'x = d'x_0\}$ is a nonempty polytope. Let x^* be an optimal extreme point of $\bar{X}$. Note that x^* is also an optimal solution to problem (2.15). Let F_0 be the zero-dimensional face of $\bar{X}$ defined by active constraints at x^*. Removing constraint (2.16) from F_0 results in a face F_1 of X of dimension at most 1. Therefore $x^* \in F_1$ and F_1 is either an extreme point or an edge of X, as desired. □

If the face F_1 described in Proposition 11 has zero-dimension, then F_1 is an optimal solution to problem (2.1) and the approximation algorithm returns it as the solution. Otherwise, F_1 is an edge of X, and the approximation algorithm computes the two extreme points of F_1 and returns the one with the best objective value.

Proposition 12. *Given a rational polytope* X *and a rational optimal solution* x_0 *to the continuous problem* (2.15)*, there exists a polynomial-time algorithm that finds an optimal solution* x^* *of problem* (2.15) *on an edge* E *of* X *and, if* x^* *is not an extreme point of* X*, two extreme points* x_1 *and* x_2 *of* E*.*

Proof. Computing F_1 as described in Proposition 11 requires finding an optimal extreme point to a linear program, which can be done using a polynomial time algorithm for linear

programs that finds an (interior) optimal primal-dual pair, and then using the polynomial algorithm of Megiddo (1991) to find an optimal extreme point. Therefore, computing x^* on an edge E of X can be done in polynomial time. If x^* is not an extreme point of X, the two extreme points of E can be found similarly by solving two auxiliary linear programs by converting tight inequalities defining F_1 into equalities. □

The approximation algorithm is displayed in Algorithm 2. We first solve the continuous relaxation (line 1) and compute an optimal x^* on an edge of X. If optimal x^* is a vertex, then we have found an optimal solution to the discrete problem (lines 3-4). Otherwise, we compute the two extreme points of the edge (line 6) and then we select the best vertex (line 7).

Algorithm 2 Approximation algorithm.

Input: X, polytope; f, objective function with $f(x) = c'x + g(d'x)$.
Output: x, a vertex of X.

1: $x_0 \leftarrow \max_{x \in X} f(x)$
2: Compute an optimal x^* on edge E.
3: **if** x^* is vertex **then**
4: $\quad x \leftarrow x^*$
5: **else**
6: $\quad (x_1, x_2) \leftarrow \texttt{vertices}(E)$
7: $\quad x \leftarrow \arg\max_{\{x_i : i=1,2\}} f(x_i)$
8: **end if**
9: **return** x

Corollary 2. *If the convex problem* (2.15) *can be solved in polynomial time, then Algorithm 2 is a polynomial-time algorithm.*

Interior point algorithms can be used to solve convex problems in polynomial time if used with a self-concordant barrier functions (Nemirovski and Todd 2008). In particular, if g is a negative exponential, logarithmic or power function, then (2.15) can be solved in polynomial time (see chapters 5.3.1 and 5.3.2 of Nesterov and Nemirovskii (1994)).

Approximation for robust conic quadratic programs

Note that in order to have an approximation algorithm to problem (2.9), an *upper bound* to problem (2.1) needs to be computed (instead of a lower bound). To this end, we propose

to solve the *approximate robust optimization problem*

$$\min_{z\in Z}\max_{x\in X} c_0'z+\sum_{i\in N}(c_i'z)\,x_i+\sqrt{z'Q_0z+\sum_{i\in N}(z'Q_iz)\,x_i}, \tag{2.17}$$

where the feasible region of the inner maximization problem is relaxed from V_X to X. We evaluate the strength of formulation (2.17) in terms of the *translated approximation ratio*

$$\Delta=\frac{\omega_c-\omega_n}{\omega_d-\omega_n}, \tag{2.18}$$

where ω_n, ω_c and ω_d are the optimal objective values of the nominal problem (4.7), the continuous relaxation (2.17) and the discrete problem (2.9), respectively. Note that when $\omega_n \geq 0$, a translated approximation ratio Δ implies a Δ-approximation algorithm in the usual sense for the minimization problem (2.9)[2].

Approximation ratio

We now characterize the approximation ratio of Algorithm 2. In this section we assume that $c'x \geq 0$ and $g(d'x) \geq 0$ over V_X. If x^* is a vertex of X, then the algorithm is exact. Now let x_1 and x_2 be two vertices of X such that $x^* = (1-\lambda)x_1 + \lambda x_2$ with $0 < \lambda < 1$, $a_i = c'x_i$ and $b_i = d'x_i$, $i = 1, 2$. Without loss of generality, assume that $g(b_1) \leq g(b_2)$. We can bound the gap between the optimal objective of (2.15) and the best extreme point solution by

$$\begin{aligned}\rho(\lambda) :=& \frac{f\left((1-\lambda)x_1+\lambda x_2\right)-\max\left\{f(x_1),f(x_2)\right\}}{\max\left\{f(x_1),f(x_2)\right\}}\\ =& \frac{a_1+\lambda(a_2-a_1)+g\left(b_1+\lambda(b_2-b_1)\right)-\max\left\{a_1+g(b_1),a_2+g(b_2)\right\}}{\max\left\{a_1+g(b_1),a_2+g(b_2)\right\}}.\end{aligned} \tag{2.19}$$

For any $\lambda \in [0, 1]$ we have

$$\max\left\{a_1+g(b_1),a_2+g(b_2)\right\} \geq a_1+g(b_1)+\lambda\left(a_2+g(b_2)-a_1-g(b_1)\right), \tag{2.20}$$

and

$$\max\left\{a_1+g(b_1),a_2+g(b_2)\right\} \geq a_2+g(b_2) \geq g(b_2). \tag{2.21}$$

Using (2.20) in the numerator of (2.19), and (2.21) in the denominator, we get

$$\rho(\lambda) \leq \frac{g\left(b_1+\lambda(b_2-b_1)\right)-g(b_1)-\lambda(g(b_2)-g(b_1))}{g(b_2)} =: \rho_+(\lambda). \tag{2.22}$$

[2]It is not possible to have an approximation algorithm for (2.9) when $\omega_n < 0$: it is possible to construct non-trivial instances where $\omega_d = 0$, and every other feasible solution is arbitrarily bad in comparison.

Remark 1. A sufficient condition for $a_2 \geq 0$ is that $c \geq 0$ and x is nonnegative. More generally, if $c'x \geq -k$ for $x \in V_X$ and $k \geq 0$, we can use (2.22) to bound the gap of the problem

$$\max_{x \in V_X} k + c'x + g(d'x). \tag{2.23}$$

Lemma 1. *If g is concave, non-decreasing, differentiable, for any $\ell \leq b_1$,*

$$\rho(\lambda) \leq \frac{g\left(\ell + \lambda(b_2 - \ell)\right) - g(\ell) - \lambda(g(b_2) - g(\ell))}{g(b_2)}.$$

Proof. Since g is non-decreasing, as by assumption $g(b_1) \leq g(b_2)$, we have $b_1 \leq b_2$. Taking the derivative of (2.22) with respect to b_1, we get that

$$\rho'_+(\lambda) = \frac{1}{g(b_2)}(1-\lambda)\big(g'\left(b_1 + \lambda(b_2 - b_1)\right) - g'(b_1)\big).$$

Since g is concave, g' is non-increasing and $g'\left(b_1 + \lambda(b_2 - b_1)\right) - g'(b_1) \leq 0$. The function ρ_+ is then non-increasing in b_1, and setting $b_1 = \ell$, we get the upper bound. □

Lemma 2. *If g is concave, non-increasing, differentiable, for any $\ell \geq b_2$,*

$$\rho(\lambda) \leq \frac{g\left(\ell + \lambda(b_2 - \ell)\right) - g(\ell) - \lambda(g(b_2) - g(\ell))}{g(b_2)}.$$

Proof. The proof is analogous to the proof of Lemma 1. □

We now give approximation ratios for different forms of function g.

Proposition 13 (Root function)**.** *If $g(z) = z^p$ with $0 < p < 1$ and $d'x \geq 0$ for all $x \in V_X$, the approximation ratio of Algorithm 2 is*

$$\frac{1}{1 + \left(\frac{1}{p}\right)^{\frac{p}{p-1}}(1-p)}.$$

Proof. We use Lemma 1 with $\ell = 0$ to get

$$\rho(\lambda) \leq \frac{(\lambda b_2)^p - \lambda b_2^p}{b_2^p} = \lambda^p - \lambda. \tag{2.24}$$

Expression (2.24) is maximized at $\lambda^* = \left(\frac{1}{p}\right)^{\frac{1}{p-1}}$, and $\rho(\lambda^*) \leq \left(\frac{1}{p}\right)^{\frac{p}{p-1}}(1-p)$. The continuous relaxation has then a gap of at most $\left(\frac{1}{p}\right)^{\frac{p}{p-1}}(1-p)$ and the result follows. □

Corollary 3. *If $g(z) = \sqrt{z}$ and $d'x \geq 0$ for all $x \in V_X$, Algorithm 2 has 4/5 approximation ratio.*

Corollary 4. *The translated approximation ratio for problem* (2.9) *satisfies* $\Delta \leq 1.25$.

Remark 2. The gap of the continuous relaxation of the square root function is tight: consider the case in which X has only two extreme points x_1 and x_2, with $(a_1, b_1) = (1, 0)$ and $(a_2, b_2) = (0, 1)$; the value of an optimal extreme point solution is 1, while the optimal solution to the continuous relaxation is $0.75x_1 + 0.25x_2$, with value 1.25. We can similarly prove that the approximation ratios of 0.72 and 0.68 (approximately) of the cubic and quartic roots are tight.

Proposition 14 (Monotone function)**.** *When g is nonnegative and monotone, Algorithm 2 has an approximation ratio 1/2.*

Proof. From (2.22), we have that

$$\begin{aligned}
\rho(\lambda) \leq & \frac{g\left(b_1 + \lambda(b_2 - b_1)\right) - g(b_1) - \lambda(g(b_2) - g(b_1))}{g(b_2)} \\
\leq & \frac{g(b_2) - g(b_1) - \lambda(g(b_2) - g(b_1))}{g(b_2)} && ((g\left(b_1 + \lambda(b_2 - b_1)\right) \leq g(b_2) \text{ by monotonicity})) \\
\leq & \frac{g(b_2) - g(b_1)}{g(b_2)} && ((\text{since } g(b_1) \leq g(b_2) \text{ w.l.o.g})) \\
= & 1 - \frac{g(b_1)}{g(b_2)} \leq 1.
\end{aligned}$$

The continuous relaxation has therefore a gap of at most 100%. In other words, selecting the better of the two extreme points solutions results in a 1/2-approximation guarantee. □

Example: Exponential utility

Consider the exponential utility function, $g(z) = 1 - e^{-z}$. Consider the case where X has only two extreme points x_1 and x_2, with $(a_1, b_1) = (1, 0)$ and $(a_2, b_2) = (0, b)$, and let $\lambda^* = -\frac{1}{b} \ln\left(\frac{1-e^{-b}}{b}\right)$. It can be shown that the gap $\rho(\lambda^*)$ defined in (2.19) approaches 1 as b goes to infinity. For instance, when $b = 100$, $\rho(\lambda^*) \approx 0.94$.

Example: Logarithmic utility

Consider the logarithmic utility function, $g(z) = \ln(1+z)$. Consider the case where X has only two extreme points x_1 and x_2, with $(a_1, b_1) = (\ln(1+b), 0)$ and $(a_2, b_2) = (0, b)$, and let $\lambda^* = \frac{1}{\ln(1+b)} - \frac{1}{b}$. It can be shown that the gap approaches 1 as b goes to infinity. For instance, when $b = 10^{40}$, $\rho(\lambda^*) \approx 0.94$.

Example: MNL probability function

Consider the MNL probability function, $g(z) = 1 - \frac{1}{1+z}$. Consider the case where X has only two extreme points x_1 and x_2, with $(a_1, b_1) = (\frac{b}{1+b}, 0)$ and $(a_2, b_2) = (0, b)$, and let $\lambda^* = \frac{\sqrt{1+b}-1}{b}$. It can be shown that the gap approaches 1 as b goes to infinity. For instance, when $b = 1000$, $\rho(\lambda^*) \approx 0.94$.

Unbounded polyhedra with integrality constraints

When X is an unbounded polyhedron, the solution x^* given by Proposition 11 may lie on an extreme ray, in which case it is not possible to find two extreme points. We consider, in this section, the particular case when X is a polyhedron with integral extreme points and rational rays and the feasible region consists of all integer points of X.

In this case we can still find an optimal solution to the continuous relaxation x^* on a face F_1 with dimension at most one as before (Proposition 11). If F_1 is an extreme point or an edge, then the algorithm is identical to the polytope case. If F_1 is a ray, let x_1 be its extreme point. In this case, we let $x_2 = x_1 + \gamma(x^* - x_1)$, where γ is the least common multiple of the denominators of the entries of $(x^* - x_1)$. Since x^* is a convex combination of x_1 and x_2, the analysis of Section 2.4 holds.

2.5 Implementation

The general algorithm proposed in Section 2.4, despite being polynomial, may not be efficient in practice. In this section we discuss improvements of Algorithm 2 that exploit additional information on X or function g, and we also give an explicit formulation of problem (2.17).

0-1 polytopes

In many of the applications of problem (2.1), including those discussed in Section 2.2, X is a binary polytope, i.e., a polytope with 0-1 valued vertices. Given an optimal solution x^* on an edge of the polytope, we describe a simple $O(n)$ procedure for finding the extreme points x_1 and x_2, provided that $x^* \neq \frac{x_1+x_2}{2}$. Observe that because x^* is a convex combination of only two binary extreme points, i.e., $x^* = \lambda x_1 + (1-\lambda)x_2$, each component of x^* is either 0, 1, λ or $1-\lambda$ for some $\lambda \in (0,1)$.

Denote by $x(i)$ the i-th coordinate of vector x. Then for each $i = 1, \ldots, n$, one of the following cases is true:

$x^*(i) = 0$ Let $x_1(i) = 0$ and $x_2(i) = 0$.

$x^*(i) = \lambda$ Let $x_1(i) = 1$ and $x_2(i) = 0$.

$x^*(i) = 1 - \lambda$ Let $x_1(i) = 0$ and $x_2(i) = 1$.

$x^*(i) = 1$ Let $x_1(i) = 1$ and $x_2(i) = 1$.

Note that if $\lambda = 1/2$, it is not obvious how to round the components to 0 and 1 consistently for each component. From the previous observation, the condition $x^* \neq \frac{x_1+x_2}{2}$ is equivalent to having no $x_i^* = \frac{1}{2}$, which is easily verifiable.

Lagrangian relaxation

In many cases there may be efficient algorithms for maximizing a linear function over X, while directly solving the nonlinear problem (2.15) may be computationally more challenging. Here we give a Lagrangian relaxation approach that exploits an oracle for the linear problem.

Suppose g is concave and increasing, the inverse h of g is differentiable and the derivative h' has an inverse h'^{-1}. Then

$$\begin{aligned} \max_{x \in X} f(x) &= \max_{x \in X} c'x + g(d'x) \\ &= \max_{x \in X,\ t \in \mathbb{R}} \{c'x + t : t \leq g(d'x)\} \\ &= \max_{x \in X,\ t \in \mathbb{R}} \{c'x + t : h(t) \leq d'x\}. \end{aligned} \tag{2.25}$$

Let $y \geq 0$ be the dual variable associated with the constraint $h(t) \leq d'x$. Since h is convex, the Lagrangian dual with respect to this constraint has no duality gap. We obtain

$$\max_{x \in X} f(x) = \min_{y \geq 0} \max_{x \in X, t \in \mathbb{R}} (c' + yd')x + t - yh(t) = \min_{y \geq 0} \theta(y). \tag{2.26}$$

Note that $\theta(y)$ is a one-dimensional convex function, which can be optimized using line search methods.

We now describe how to evaluate $\theta(y)$. Taking derivatives in (2.26) with respect to t, we find that $1 - yh'(t) = 0$, or $t = h'^{-1}\left(\frac{1}{y}\right)$. Replacing in (2.26), we obtain

$$\theta(y) = h'^{-1}\left(\frac{1}{y}\right) - yh\left(h'^{-1}\left(\frac{1}{y}\right)\right) + \max_{x \in X}(c' + yd')x,$$

which can be computed efficiently using the linear oracle.

Example: Square root function

For $g(z) = \beta\sqrt{\xi + z}$, we have that $h(z) = \left(\frac{z}{\beta}\right)^2 - \xi$, $h'(z) = \frac{2z}{\beta^2}$, $h'^{-1}(z) = \frac{\beta^2 z}{2}$ and

$$\theta(y) = \frac{\beta^2}{4y} + y\xi + \max_{x \in X}(c' + yd')x.$$

Example: Exponential utility

For $g(z) = 1 - e^{-z}$, we have that $h(z) = -\ln(1 - z)$, $h'(z) = \frac{1}{1-z}$, $h'^{-1}(z) = 1 - \frac{1}{z}$ and

$$\theta(y) = 1 - y + y\ln(y) + \max_{x \in X}(c' + yd')x.$$

Example: Logarithmic utility

For $g(z) = \ln(1 + z)$, we have that $h(z) = e^z - 1$, $h'(z) = e^z$, $h'^{-1}(z) = \ln(z)$ and

$$\theta(y) = y - \ln(y) - 1 + \max_{x \in X}(c' + yd')x.$$

Example: MNL probability function

For $g(z) = \frac{z}{1+z}$, we have that $h(z) = \frac{z}{1-z}$, $h'(z) = \frac{1}{(1-z)^2}$, $h'^{-1}(z) = 1 - \frac{1}{\sqrt{z}}$ and

$$\theta(y) = (1 - \sqrt{y})^2 + \max_{x \in X}(c' + yd')x.$$

Proposition 15. *Problem* (2.26) *can be solved with* $O\left(\ln\left(\frac{g'(d'x_L)}{\epsilon}\right)\right)$ *calls to the linear optimization oracle, where* $\epsilon > 0$ *is the precision and* $x_L = \arg\max_{x\in X} c'x$.

Proof. Let $y^* = \arg\min_{y\geq 0}\theta(y)$, and let x^* and t^* be optimal solutions to (2.25). Since $t^* = g(d'x^*)$ and $t^* = h'^{-1}\left(\frac{1}{y^*}\right)$, we have that $y^* = \frac{1}{h'(g(d'x^*))}$. Using the inverse function theorem, we get that $y^* = g'(d'x^*)$. Moreover, $c'x_L \geq c'x^*$ and $d'x_L \leq d'x^*$ and, since g is concave, g' is non-increasing and in particular $g'(d'x^*) \leq g'(d'x_L)$. Therefore $0 \leq y^* \leq g'(d'x_L)$, and solving problem (2.26) using golden section search requires

$$\left\lceil \frac{\ln\left(\frac{g'(d'x_L)}{\epsilon}\right)}{-\ln(0.618)} \right\rceil + 2$$

calls to the linear oracle. □

Remark 3. Let $y^* = \arg\min_{y\geq 0}\theta(y)$. If $\max_{x\in X}(c + y^*d)'x$ has a unique optimal solution x^*, then x^* is an optimal solution to problems (2.15) and (2.1). Otherwise, if $\max_{x\in X}(c+y^*d)'x$ has optimal extreme point solutions x_1 and x_2, then there exists an optimal solution x^* to problem (2.15) which can be written as a convex combination of x_1 and x_2. Moreover, note that each evaluation of $\theta(y)$ yields a vertex of X. Therefore, instead of selecting the best vertex among x_1 and x_2, we can select the best among all evaluated vertices, resulting in a practical improvement of Algorithm 2.

Remark 4. The Lagrangian relaxation provides an upper bound on the optimality gap, given by

$$\left|\frac{V_{\text{cont}} - V_{\text{int}}}{V_{\text{int}}}\right|, \tag{2.27}$$

where $V_{\text{cont}} = \min_{y\geq 0}\theta(y)$ is the value of the continuous relaxation and V_{int} is the objective value of the best extreme point found.

Approximate robust formulation

In this section we discuss how to solve the approximate robust conic quadratic optimization problem

$$\begin{aligned}(\text{ARCQO}) \quad \zeta &:= \min_{z\in Z}\max_{x\in\mathbb{R}^n}\left\{c_0'z+\sum_{i\in N}(c_i'z)\,x_i+\sqrt{z'Q_0z+\sum_{i\in N}(z'Q_iz)\,x_i} : Ax\le b, x\ge 0\right\}\\ &= \min_{z\in Z, y\ge 0}\left\{\frac{1}{4y}+c_0'z+yz'Q_0z+\max_{Ax\le b, x\ge 0}\sum_{i=1}^{n}(c_i'z+yz'Q_iz)\,x_i\right\},\end{aligned}$$

where the equality follows from the analysis in the previous section. Let w be the dual variables associated with the inner maximization problem. Using standard LP duality, we get that

$$\begin{aligned}&\zeta = \min \frac{1}{4y}+c_0'z+yz'Q_0z+b'w\\ (\text{ARCQO}') \qquad &\text{s.t } A_i'w\ge c_i'z+yz'Q_iz && i=1,\dots,n\\ &\quad z\in Z, y\ge 0, w\ge 0,\end{aligned}$$

where A_i' denotes the transpose of the i-th column of A. We now substitute $y = 1/\hat{y}$, introduce additional nonnegative variables v_i, $i = 0,\dots,n$, and enforce the rotated second order cone constraints $z'Qz \le \hat{y}v_i$ to get the equivalent formulation

$$\begin{aligned}&\zeta = \min \frac{1}{4}\hat{y}+c_0'z+v_0+b'w\\ (\text{ARCQO}'') \qquad &\text{s.t } A_i'w\ge c_i'z_i+v_i && i=1,\dots,n\\ &\quad z'Q_iz\le \hat{y}v_i && i=0,\dots,n\\ &\quad z\in Z, \hat{y}\ge 0, w\ge 0, v\ge 0.\end{aligned}$$

A case of particular interest is when Z is a SOCP-representable convex set.

Proposition 16. *If Z is SOCP-representable, then the approximate robust conic quadratic optimization problem (ARCQO) is SOCP-representable.*

Note that there are efficient interior point algorithms for the special case of SOCPs (e.g. Nesterov and Todd 1998).

Remark 5. Unlike the linear case studied in Bertsimas and Sim (2004), the *exact* robust counterpart (2.9) is $\mathcal{NP}$-hard. Note that most robust counterparts of conic quadratic programs are $\mathcal{NP}$-hard, and *safe tractable approximations* are used instead (Ben-Tal et al. 2009,

chapters 5-7). In our case, the approximation ratio is constant (1.25), and the approximation belongs to the same complexity class as the nominal problem (4.7).

Computational complexity

We now discuss the theoretical computational complexity of the approximation algorithm for specific polytopes, and compare it with the existing approaches in the literature.

Computational complexity for the uniform matroid

The greedy algorithm, with complexity $O(nk)$, is a well-known approach to tackle a monotone submodular maximization problem over the uniform matroid. We now argue that the Lagrangean version of the approximation algorithm described in 2.5 may be preferable in large instances.

First note that maximizing a linear function over the uniform matroid can be done in $O(n)$ time using quickselect. The complexity of the approximation algorithm is thus $O\left(n \ln\left(\frac{g'(d'x_L)}{\epsilon}\right)\right)$. Note that ϵ corresponds to the required precision of a lagrangean multiplier related with the nonlinear term, and *does not depend on n or k.* Moreover, for the exponential utility, logarithmic utility and MNL probability function we have that $g'(d'x) \leq 1$ whenever $d'x \geq 0$ (a similar dimension-independent upper bound can be obtained for the square root function if $\xi > 0$). Therefore, we see in small instances with high precision (i.e., $k < \ln \frac{1}{\epsilon}$) the greedy algorithm is preferable, but in instances with large values for k and n the proposed approximation algorithm is faster.

Badanidiyuru and Vondrák (2014) proposed another algorithm for maximizing a submodular function over the uniform matroid that does not depend on k. However, the complexity of their algorithm is $O\left(\frac{n}{\epsilon} \ln \frac{n}{\epsilon}\right)$, which is worse than the complexity of our approximation algorithm for all values of n.

Computational complexity for matroids

We consider the case when X is a matroid which is known through an independence oracle. There are randomized algorithms for maximizing a monotone submodular function, which rely on *the multilinear extension*, but such algorithms are computationally expensive. For example, the algorithm of Calinescu et al. (2011) runs in $\tilde{O}(n^8)$ time, and the authors comment that the high complexity is due to the number of samples necessary to achieve

high probability bounds. A more efficient algorithm was proposed by Badanidiyuru and Vondrák (2014), with complexity $O\left(\frac{1}{\epsilon^4}nk\ln^2\frac{n}{\epsilon}+T\left(\frac{1}{\epsilon^2}n\ln\frac{n}{\epsilon}+\frac{1}{\epsilon}k^2\right)\right)$, where k is the maximum cardinality of a feasible solution and T is the time complexity of the independence oracle. Observe that the time complexity of the Lagrangean version of our approximation algorithm is better: finding a maximum weight independent set (linear oracle) can be done in $O\left(n\ln n+Tn\right)$ using the greedy algorithm, and the overall complexity of the algorithm is thus $O\left((n\ln n+Tn)\ln\frac{1}{\epsilon}\right)$.

Computational complexity for 0-1 down-monotone polytopes

Vondrák et al. (2011) propose a general framework for maximizing a submodular function over down-monotone polytopes using *contention resolution schemes*, which are general randomized rounding techniques that convert a fractional solution obtained from solving a relaxation into a feasible integer solution while preserving the quality of the solution. However, finding such a resolution scheme (which depends on fractional solution found) is computationally challenging, even when the polytope is a matroid: it requires solving an LP which is only known through a separation oracle (using the ellipsoid method, which is known to perform poorly in practice), and in which each evaluation of the separation oracle is noisy (if ε is the maximum violation allowed for a given constraint, then we require $poly(1/\varepsilon)$ calls to the separation oracle to guarantee feasibility with high probability). In contrast, using the rounding technique for 0-1 polytopes described in Section 2.5, we *deterministically* recover a feasible solution in only $O(n)$ time.

General case

Most approximation algorithms in the literature rely on *rounding down* fractional solutions, and such algorithms do not guarantee feasibility if the polytope is not down-monotone (to the best of our knowledge there is no other approximation algorithm for general polyhedra). In contrast our approach exploits the structure of the objective function, but it makes no assumption about the polyhedron besides the existence of an efficient optimization oracle.

2.6 Computational experiments

In this section we study the empirical performance of the approximation algorithm. First we test the approximation algorithm in instances for which there are other approximation

methods in the literature; next we test the approximation algorithm in the assignment polytope; then we test the approximation algorithm for the PERT application; then we test the approximation algorithm for the robust optimization application; and finally we test the algorithm in the combinatorial multi-armed bandit context. Note that for the assignment polytope, PERT, robust optimization and multi-armed applications, methods based on submodularity are not applicable. All experiments are conducted on one thread of a Dell computer with a 2.2 GHz Intel®Core™ i7-2670QM CPU and 8 GB main memory.

Experiments for the uniform matroid polytope

In this section we conduct computational experiments over the uniform matroid, comparing the proposed approximation algorithm and the greedy algorithm. We solve the problems with the Lagrangian approach described in Section 2.5, using golden section for the line search and *quickselect* as the linear oracle.

Each coefficient c_i is generated uniformly in $[0, 1]$ and scaled so that $\sum_{i=1}^{n} c_i = 1$. Each coefficient d_i is generated uniformly in $[0, 1/c_i]$, and scaled so that $\sum_{i=1}^{n} d_i = 1$. The scaling ensures that the weights of the linear and nonlinear parts of the objective are similar, and the coefficients are generated so that there is a tradeoff between the linear and nonlinear contributions. Both of these choices contribute to making the instances more challenging. We set $k = n/10$ and we use $g(x) = 1 - e^{-d'x}$, as in the reliability problem described in Section 2.2[3]. Table 2.1 shows, for different values of n, the time required by the approximation algorithm and the greedy algorithm to solve the instances in milliseconds and the gap between the solutions found, computed as

$$\frac{f^*_{\text{Greedy}} - f^*_{\text{Approx.}}}{f^*_{\text{Greedy}}},$$

where f^*_{Greedy} and $f^*_{\text{Approx.}}$ are the objective values of the solutions found by the greedy algorithm and the approximation algorithm, respectively. Each column represents the average over five instances generated with the same parameters. We also computed an upper bound of the gap with respect of the optimal solution using (2.27), and the gaps are under 0.1% for both approaches.

We see that the solutions generated by both approaches are very similar in terms of the objective value. Yet, the proposed approximation algorithm is faster than the greedy algorithm in all cases, and is considerably so for the larger instances.

[3]Experiments with other functions yield similar results.

Table 2.1: Results in the uniform matroid polytope.

	$n = 100$	$n = 1,000$	$n = 10,000$	$n = 100,000$	$n = 500,000$
Time approximation (ms)	2	4	26	102	305
Time greedy (ms)	3	18	1,184	112,533	2,819,163
Gap	-1.5×10^{-16}	2.4×10^{-15}	-3.1×10^{-15}	8.1×10^{-9}	1.7×10^{-8}

As we observed in Section 2.5, the time complexity of existing approximation algorithms for other down-monotone polytopes is much larger than the complexity of the greedy algorithm. The results for the uniform matroid polytope suggest the performance of the approximation algorithm may be orders of magnitude faster than existing approaches in other polytopes in practice.

Experiments for the assignment polytope

In this section we conduct computational experiments over the $n \times n$ assignment polytope. We solve the problems with the Lagrangian approach described in Section 2.5, using golden section for the line search and the the *Hungarian method* as the linear oracle. The coefficients are generated as in the uniform matroid polytope case, and we use $g(x) = \frac{d'x}{1+d'x}$, as in the assortment optimization problem.

Table 2.2 shows, for different values of n, the time required by the approximation algorithm to solve the instances in milliseconds and the upper bound of the optimality gap given by (2.27). Each column represents the average over five instances generated with the same parameters.

Table 2.2: Results in the assignment polytope.

	$n = 3$	$n = 10$	$n = 100$	$n = 1000$
Time approximation (ms)	1	2	66	2,892
Gap	0.24%	0.01%	0.00%	0.00%

We observe that in problems with $n \geq 100$ the solutions to approximation algorithm are very close to optimal, and in the smaller instances the average gap is at most 0.24%. In both cases the approximation the reported gap is far from the worst case bound. Moreover

the approximation algorithm can solve instances with $n = 1,000$ (i.e. 1,000,000 variables) in under three seconds.

PERT

The networks for the critical path experiments are constructed as follows. Let $V = \{0, 1, \cdots, r\}$ and p be the density parameter. For each pair (i, j), $i < j$, construct an arc with probability p. For each arc the expected duration c_{ij} is drawn from $U[0.5(j-i), 1.5(j-i)]$, and the standard deviation $\sqrt{d_{ij}}$ from $U[0.25c_{ij}, 0.75c_{ij}]$.

The goal is to find the critical 0-r path that best approximates the Value-at-Risk at confidence level α of the completion time of the project (as defined in (2.4)). We compare the two approaches described in Section 2.2:

Det-critical We use the Value-at-Risk of the deterministic critical path.

VaR-critical We use the Lagrangian version of the approximation algorithm to solve problem (2.5), and use the resulting path to approximate the Value-at-Risk.

For each instance we compute the simulated Value-at-Risk (VaR_{sim}) of the completion time of the project using Monte Carlo simulation with 2,000 replications. In each replication we generate samples for the duration of each arc according to the normal distributions. We then compute $\text{VaR}_\alpha(D) = \sup\{\ell \in \mathbb{R} : \mathbf{Pr}(D \leq \ell) \leq 1 - \alpha\}$ using binary search on ℓ, and we evaluate $\mathbf{Pr}(D \leq \ell)$ by the fraction of replications in which $D \leq \ell$ (where D is the length of the critical path).

Table 2.3 presents the results for instances with 50 nodes. It shows the density p; the solution approach used; and for varying confidence levels the estimated Value-at-Risk $\text{VaR}_{\text{est.}}$, and the gap between $\text{VaR}_{\text{est.}}$ and VaR_{sim}, computed as

$$\frac{\text{VaR}_{\text{sim.}} - \text{VaR}_{\text{est.}}}{\text{VaR}_{\text{sim.}}}.$$

Each row represents the average over five instances generated with the same parameters. In all cases, the solution of approximation algorithm is within 0.1% of optimal (computed using (2.27)) and computing the VaR_α-critical path takes less than 0.1 seconds.

The computational experiments suggest that explicitly using the risk information to select the critical path results in paths that better estimate the Value-at-Risk of the project compared to using the deterministic critical path by PERT: the gap between the estimated

Table 2.3: Duration estimates in networks with 50 nodes.

Density	Method	90% CL		97.5% CL		99% CL	
		Est.	**Gap**	**Est.**	**Gap**	**Est.**	**Gap**
0.3	Det	10,137	26.4%	12,031	16.5%	13,203	14.6%
	VaR	10,811	20.2%	13,008	9.8%	14,375	7.1%
	Sim	13,008	-	14,414	-	15,469	-
0.5	Det	10,002	33.4%	11,563	30.3%	12,344	30.1%
	VaR	11,699	22.2%	14,453	12.9%	15,625	11.5%
	Sim	15,029	-	16,593	-	17,656	-
0.8	Det	10,479	35.4%	12,383	30.2%	13,281	29.2%
	VaR	12,036	25.8%	15,078	15.0%	16,250	13.3%
	Sim	16,230	-	17,734	-	18,750	-
Average	**Det**		**31.7%**		**25.7%**		**24.6%**
	VaR		**22.7%**		**12.6%**		**10.6%**

and true Value-at-Risk of the project is reduced by almost half. Note that the quality of the approximation for both approaches decreases as the density of the network increases, and the VaR_α-critical path is comparatively better for higher values of confidence.

These experiments on estimating project duration indicate that the approximation algorithm is effective in finding optimal solutions to VaR-critical path problem.

Robust portfolio optimization

We illustrate the method for robust conic quadratic programs proposed in Section 2.4 using a canonical finance application. Given a set of assets $N = \{1, \ldots, n\}$, with expected return μ and covariance matrix Σ, the portfolio with minimum value-at-risk can be found by solving the optimization problem

$$\min_{z \in Z} \; -\mu' z + \beta \sqrt{z' \Sigma z},$$

where $Z = \left\{ z \in \mathbb{R}^n_+ : \sum_{i \in N} z_i = 1 \right\}$ is the set of long-only budget-constrained portfolios.

For simplicity we test the case where Σ is a diagonal matrix with entries $\Sigma_{ii} = \sigma_i^2$. In the robust version, we consider the events that may reduce the expected return of asset i to $\mu_i - c_i$ and increase the variance of it to $\sigma_i^2 + d_i$. The decision-maker wishes to select a robust portfolio, given that k of the events happen simultaneously.

For the particular case of diagonal Σ and each event corresponding to a single variable, the approximate robust formulation in Section 2.5 can be further simplified to

$$\begin{aligned}
\min\ & \frac{1}{4}\hat{y} - \sum_{i\in N}\mu_i z_i + \sum_{i\in N}\sigma_i^2 v_i + kw \\
\text{s.t } & w \geq c_i z_i + d_i v_i && i = 1,\dots,n \\
& z_i^2 \leq \hat{y} v_i && i = 1,\dots,n \\
& z \in Z, \hat{y} \geq 0, w \geq 0, v \geq 0.
\end{aligned}$$

In order to compute the optimality gap, we solve the robust problem (2.9) exactly by enumerating all extreme points of the uniform matroid polytope, i.e.,

$$\begin{aligned}
\min\ & t \\
\text{s.t. } & t \geq -\sum_{i\in N}(\mu_i - c_i x_i) z_i + \sqrt{\sum_{i\in N}(\sigma_i^2 + d_i x_i) z_i^2} \quad \forall x \in V_X \\
& z \in Z, t \in \mathbb{R}
\end{aligned} \tag{2.28}$$

which requires $\binom{n}{k}$ conic quadratic constraints.

We solve both formulations using CPLEX 12.6.2. In our experiments we use $k = 3$, $\beta = 2$, μ_i is drawn from $U[0, 1]$, σ_i is drawn from $U[0, 2\mu_i]$ (thus risky assets have on average high expected return) and c_i and $\sqrt{d_i}$ are drawn form $U[0, 2]$. Table 2.4 presents the results. It shows, for n between 10 and 70, the time required to solve the problems for both formulations in milliseconds, and the value of the approximation ratio (2.18). Each column represents the average over five instances generated with the same parameters. For instances with $n > 70$, formulation (2.28) is impractical due to its high memory requirements.

Table 2.4: Robust portfolio results.

	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$	$n = 70$
Time approximation (ms)	8	10	17	35	36	39	59
Time exact (ms)	59	1,092	7,904	33,252	101,113	266,131	601,215
Δ	1.000	1.008	1.004	1.001	1.000	1.000	1.000

We see that the approximate robust formulation is very accurate in practice, finding optimal solutions (with $\Delta = 1$) in most of the cases. Its performance is far from worst case bound of 1.25. We also observe that the approximate robust problem scales very well in practice.

Multi-armed bandit

We consider the multi-armed bandit problem where the weight of each arc follows a Bernoulli distribution whose parameter is uniformly generated on $[0, 1]$. To evaluate a policy in terms of regret, we simulate the learning process: at each step we play a solution according to the policy, then we generate random variates from the Bernoulli distributions of each item, and finally we use the generated variates to update the parameters of the policy. We compare the regret of Algorithm 1 using the approximation algorithm to solve the subproblems (NlUCB), with other approaches found in the literature. In particular the test the alternatives proposed by Chen et al. (2013) (CombUCB) and György et al. (2006) (Exp). Note that algortihm Exp is designed for the more general adverserial case, but is limited to the path polytope.

We summarize the results in Table 2.5. It shows, from left to right: the number of variables in the problem; the policy used to solve the problem; the average regret after 20,000 simulation iterations. Each row represents the average over 5 simulations with the same parameters. Figure 2.2 presents the average cumulative regret of NlUCB and CombUCB as a function of the iteration for the path instances of size 760 (others sizes have similar behavior).

Table 2.5: Regret after 20,000 iterations.

Variables	**Policy**	**Regret**
40	NlUCB	356
	CombUCB	1,6885
	Exp	25,188
180	NlUCB	747
	CombUCB	8,018
	Exp	79,304
760	NlUCB	3,669
	CombUCB	31,670
	Exp	175,237

In all cases, NlUCB achieves lower regret than the other methods.

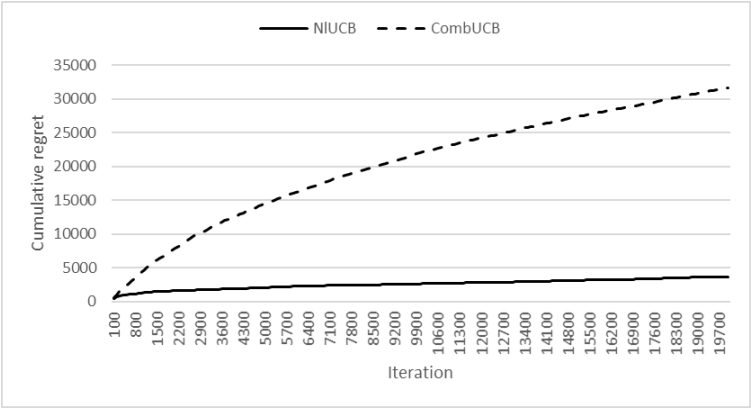


Figure 2.2: Cumulative regret of App and CombUCB in instances with 760 variables.

2.7 Conclusions

In this chapter we consider the problem of maximizing a class of concave utility functions over the extreme points of a polytope, which is $\mathcal{NP}$-hard for many classes of polytopes. Such problems naturally arise in combinatorial problems in which the feasible region is composed of the extreme points of an integral polytope. We exploit the property that there exists a solution of the continuous relaxation of the problem on an edge of the polytope to develop an approximation algorithm. The algorithm requires either an oracle for the continuous relaxation of the nonlinear problem, or an oracle for linear programs over the polytope, and is polynomial time under mild assumptions on the function g and the polytope X. We prove that the proposed approach is a 1/2-approximation. When the concave function in the objective is the square root function, the proposed approach is a 4/5-approximation. We also propose a simple 1.25-approximation algorithm for a class of robust conic quadratic minimization problems. Computational experiments suggest that both approaches find solutions with very small optimality gap, are much more efficient than alternatives found in the literature in some cases, and are the first approximation algorithms proposed for other cases.

Chapter 3

Simplex QP method for conic quadratic minimization

3.1 Introduction

Consider the minimization of a conic quadratic function over a polyhedron, i.e.,

$$\text{(CO)} \quad \min_{x \in \mathbb{R}^n} \left\{ c'x + \Omega\sqrt{x'Qx} : x \in X \right\},$$

where $c \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n \times n}$ is a symmetric positive semidefinite matrix, $\Omega > 0$, and $X \subseteq \mathbb{R}^n$ is a rational polyhedron. We denote by CDO the discrete counterpart of CO with integrality restrictions: $X \cap \mathbb{Z}^n$. CO and CDO are frequently used to model utility with uncertain objectives as in parametric value-at-risk minimization (El Ghaoui et al. 2003), portfolio optimization (Atamtürk and Jeon 2017), and robust counterparts of linear programs with an ellipsoidal objective uncertainty set (Ben-Tal and Nemirovski 1998, 1999, Ben-Tal et al. 2009).

Note that CO includes LP and QP as special cases. The simplex method (Dantzig et al. 1955, Wolfe 1959, Van de Panne and Whinston 1964) is still the most widely used algorithm for LP and QP, despite the fact that polynomial interior point algorithms (Karmarkar 1984, Nesterov and Nemirovskii 1994, Nemirovski and Scheinberg 1996) are competitive with the simplex method in many large-scale instances. Even though non-polynomial, the simplex method has some distinct advantages over interior point methods. Since the simplex method iterates over bases, it is possible to carry out the computations with high accuracy and little cost, while interior point methods come with a trade-off between precision and efficiency. Moreover, an optimal basis returned by the simplex method is useful for sensitivity analysis, while interior point methods do not produce such a basis unless an additional "crashing"

procedure is performed (e.g. Megiddo 1991). Finally, if the parameters of the problem change, re-optimization can often be done very fast with the simplex method starting from a primal or dual feasible basis, whereas warm starts with interior point methods have limitations (Yildirim and Wright 2002, Çay et al. 2017). In particular, fast re-optimization with the dual simplex method is crucial when solving discrete optimization problems with a branch-and-bound algorithm.

CO is a special case of SOCO (Lobo et al. 1998, Alizadeh and Goldfarb 2003), which can be solved by polynomial-time interior points algorithms (Alizadeh 1995, Nesterov and Todd 1998, Ben-Tal and Nemirovski 2001a). Although CO can be solved by a general conic quadratic solver, we show in this chapter that iterative QP algorithms scale much better. In particular, simplex-based QP algorithms allowing warm starts perform orders of magnitude faster than interior point methods for CO.

For the discrete counterpart CDO, a number of different approaches are available for the special case with a diagonal Q matrix: Ishii et al. (1981) give a polynomial time for optimization over spanning trees; Bertsimas and Sim (2004) propose an approximation algorithm that solves series of linear integer programs; Atamtürk and Narayanan (2008) give a cutting plane algorithm utilizing the submodularity of the objective for the binary case; Atamtürk and Narayanan (2009) use parametric linear programming for the binary case with a cardinality constraint.

The aforementioned approaches do not extend to the non-diagonal case or to general feasible regions, which are obviously $\mathcal{NP}$-hard as quadratic and linear integer optimization are special cases. The branch-and-bound algorithm is the method of choice for general CDO. However, branch-and-bound algorithms that repeatedly employ a nonlinear programming (NLP) solver at the nodes of the search tree are typically hampered by the lack of effective warm starts. Borchers and Mitchell (1994) and Leyffer (2001) describe NLP-based branch-and-bound algorithms, and they give methods that branch without solving the NLPs to optimality, reducing the computational burden for the node relaxations. On the other hand, LP-based branch-and-bound approaches employ linear outer approximations of the nonlinear terms. This generally results in weaker relaxations at the nodes, compared to the NLP approaches, but allows one to utilize warm starts with the simplex method. Therefore, one is faced with a trade-off between the strength of the node relaxations and the solve time per node. A key idea to strengthen the node relaxations, as noted by Tawarmalani and Sahinidis (2005), is to use extended formulations. Atamtürk and Narayanan (2007) describe mixed-integer rounding inequalities in an extended formulation for conic quadratic

integer programming. Vielma et al. (2016) use an extended formulation for conic quadratic optimization that can be refined during branch-and-bound, and show that an LP-based branch-and-bound using the extended formulations typically outperforms the NLP-based branch-and-bound algorithms. The reader is referred to Belotti et al. (2013) for an excellent survey of the solution methods for mixed-integer nonlinear optimization.

In this chapter, we reformulate CO through the perspective of its objective function and give algorithms that solve a sequence of closely related QPs. Utilizing the simplex method, the solution to each QP is used to warm start the next one in the sequence, resulting in a small number of simplex iterations and fast solution times. Moreover, we show how to incorporate the proposed approach in a branch-and-bound algorithm, efficiently solving the continuous relaxations to optimality at each node and employing warm starts with the dual simplex method. Our computational experiments indicate that the proposed approach outperforms the state-of-the-art algorithms for convex as well as discrete cases.

The rest of the chapter is organized as follows. In Section 3.2 we give an alternative formulation for CO using the perspective function of the objective. In Section 3.3 we present coordinate descent and accelerated bisection algorithms that solve a sequence of QPs. In Section 3.4 we provide computational experiments, comparing the proposed methods with state-of-the-art barrier and other algorithms.

3.2 Formulation

In this section we present a reformulation of CO using the perspective function of its objective. Let $X = \{x \in \mathbb{R}^n : Ax = b,\ x \geq 0\}$ be the feasible region of problem CO. For convex quadratic $q(x) = x'Qx$, consider the function $h : \mathbb{R}^{n+1} \to \mathbb{R}_+ \cup \{\infty\}$ defined as

$$h(x,t) = \begin{cases} \frac{x'Qx}{t} & \text{if } t > 0, \\ 0 & \text{if } x'Qx = 0, t = 0, \\ +\infty & \text{otherwise.} \end{cases}$$

Observe that

$$\begin{aligned} &\min \left\{ c'x + \Omega\sqrt{x'Qx} : x \in X \right\} \\ =&\min \left\{ c'x + \frac{\Omega}{2}h(x,t) + \frac{\Omega}{2}t : x \in X,\ t = \sqrt{x'Qx} \right\} \\ \geq\ & \zeta, \end{aligned}$$

where

$$\text{(PO)} \quad \zeta = \min \left\{ c'x + \frac{\Omega}{2} h(x,t) + \frac{\Omega}{2} t : x \in X,\ t \geq 0 \right\}.$$

We will show that problems CO and PO have, in fact, the same optimal objective value and that there is a one-to-one correspondence between the optimal primal-dual pairs of both problems.

Proposition 17. *Problem PO is a convex optimization problem.*

Proof. It suffices to observe that h is the closure of the *perspective function* $tq(x/t)$ of the convex quadratic function $q(x)$, and is therefore convex (e.g. Hiriart-Urruty and Lemaréchal 2013, p. 160). Since all other objective terms and constraints of PO are linear, PO is a convex optimization problem. □

Proposition 18. *Problems CO and PO are equivalent.*

Proof. If $t > 0$, the objective function of problem PO is continuous and differentiable, and since the feasible region is a polyhedron and the problem is convex, its KKT points are equivalent to its optimal solutions. The KKT conditions of PO are

$$\begin{aligned} Ax = b,\ x \geq 0,\ t \geq 0 & \\ -c' - \frac{\Omega}{t} x'Q = \lambda' A - \mu & \qquad (3.1) \\ \frac{\Omega}{2t^2} x'Qx - \frac{\Omega}{2} = 0 & \qquad (3.2) \\ \mu \geq 0 & \\ \mu' x = 0, & \end{aligned}$$

where λ and μ are the dual variables associated with constraints $Ax = b$ and $x \geq 0$, respectively. Note that $t > 0$ and (3.2) imply that $t = \sqrt{x'Qx}$. Substituting $t = \sqrt{x'Qx}$ in (3.1), one arrives at the equivalent conditions

$$\begin{aligned} Ax = b,\ x \geq 0 & \\ -c' - \frac{\Omega}{\sqrt{x'Qx}} x'Q = \lambda' A - \mu & \qquad (3.3) \\ t = \sqrt{x'Qx} & \qquad (3.4) \\ \mu \geq 0 & \\ \mu' x = 0. & \end{aligned}$$

Ignoring the redundant variable t and equation (3.4), we see that these are the KKT conditions of problem CO. Therefore, any optimal primal-dual pair for PO with $t > 0$ is an

optimal primal-dual pair for CO. Similarly, we see that any optimal primal-dual pair of problem CO with $x'Qx > 0$ gives an optimal primal-dual pair of problem PO by setting $t = \sqrt{x'Qx}$. In both cases, the objective values match.

On the other hand, if $t = 0$, then PO reduces to problem

$$\min_{x \in \mathbb{R}^n} \{c'x : Ax = b, x \geq 0, x'Qx = 0\},$$

which corresponds to CO with $x'Qx = 0$, and hence they are equivalent. □

Since they are equivalent optimization problems, we can use PO to solve CO. In particular, we exploit the fact that, for a fixed value of t, PO reduces to a QP.

3.3 Algorithms

For simplicity, assume that PO has an optimal solution; hence, X is nonempty and may be assumed to be bounded. Consider the one-dimensional optimal value function

$$g(t) = \min_{x \in X} c'x + \frac{\Omega}{2} h(x,t) + \frac{\Omega}{2} t. \tag{3.5}$$

As X is nonempty and bounded, g is real-valued and, by Proposition 17, it is convex. Throughout, $x(t)$ denotes an optimal solution to (3.5).

In this section we describe two algorithms for PO that utilize a QP oracle. The first one is a coordinate descent approach, whereas, the second one is an accelerated bisection search algorithm on the function g. Finally, we discuss how to exploit the warm starts with the simplex method to solve convex as well as discrete cases.

Coordinate descent algorithm

Algorithm 3 successively optimizes over x for a fixed value of t, and then optimizes over t for a fixed value of x. Observe that the optimization problem in line 4 over x is a QP, and the optimization in line 5 over t has a closed form solution: by simply setting the derivative to zero, we find that $t_{i+1} = \sqrt{x_{i+1}{}'Qx_{i+1}}$.

First observe that the sequence of objective values $\left\{c'x_i + \frac{\Omega}{2t_i} x_i' Q x_i + \frac{\Omega}{2} t_i\right\}_{i \in \mathbb{N}}$ is non-increasing. Moreover, the dual feasibility KKT conditions for the QPs in line 4 are of the form

$$-c' - \frac{\Omega}{t_i} x_{i+1}{}'Q = \lambda' A - \mu. \tag{3.6}$$

Algorithm 3 Coordinate descent.

Input: X polyhedron; Q psd matrix; c cost vector; $\Omega > 0$
Output: Optimal solution x^*

1: **Initialize** $t_0 > 0$ ▷ e.g. $t_0 = 1$
2: $i \leftarrow 0$ ▷ iteration counter
3: **repeat**
4: $\quad x_{i+1} \leftarrow \arg\min_{x \in X} \left\{ c'x + \frac{\Omega}{2t_i} x'Qx + \frac{\Omega}{2} t_i \right\}$ ▷ solve QP
5: $\quad t_{i+1} \leftarrow \arg\min_{t \geq 0} \left\{ c'x_{i+1} + \frac{\Omega}{2t} x_{i+1}'Qx_{i+1} + \frac{\Omega}{2} t \right\}$ ▷ $t_{i+1} = \sqrt{x_{i+1}'Qx_{i+1}}$
6: $\quad i \leftarrow i + 1$
7: **until** stopping condition is met
8: **return** x_i

Let $\|\cdot\|$ be a norm and suppose that the QP oracle finds feasible primal-dual pairs with $\epsilon > 0$ tolerance with respect to $\|\cdot\|$. In particular x_{i+1} in line 4 violates (3.6) by at most ϵ, i.e.,

$$\left\| -c' - \frac{\Omega}{t_i} x_{i+1}'Q - \lambda'A + \mu \right\| \leq \epsilon.$$

Proposition 19 below states that, at each iteration of Algorithm 3, we can bound the violation of the dual feasibility condition (3.3) corresponding to the original problem CO. The bound depends only on the precision of the QP oracle ϵ, the relative change of t in the last iteration $\frac{\Delta_i}{t_i}$, where $\Delta_i = t_{i+1} - t_i$, and the gradient of the function $f(x) = \Omega\sqrt{x'Qx}$ evaluated at the new point x_{i+1}.

Proposition 19 (*Dual feasibility bound*)**.** *A pair* (x_{i+1}, t_{i+1}) *in Algorithm 3 satisfies*

$$\left\| -c' - \Omega \frac{x_{i+1}'Q}{\sqrt{x_{i+1}'Qx_{i+1}}} - \lambda'A + \mu \right\| \leq \epsilon + \frac{|\Delta_i|}{t_i} \cdot \|\nabla f(x_{i+1})\|$$

Proof.

$$\begin{aligned}
&\left\| -c' - \Omega \frac{x_{i+1}'Q}{\sqrt{x_{i+1}'Qx_{i+1}}} - \lambda' A + \mu \right\| \\
=&\left\| -c' - \Omega \frac{x_{i+1}'Q}{t_i + \Delta_i} - \lambda' A + \mu \right\| \\
=&\left\| -c' - \Omega \frac{x_{i+1}'Q}{t_i} - \Omega x_{i+1}'Q \left(\frac{1}{t_i + \Delta_i} - \frac{1}{t_i} \right) - \lambda' A + \mu \right\| \\
=&\left\| -c' - \Omega \frac{x_{i+1}'Q}{t_i} - \lambda' A + \mu + \Omega \left(\frac{\Delta_i}{t_i \cdot t_{i+1}} \right) x_{i+1}'Q \right\| \\
\leq&\epsilon + \left\| \Omega \frac{\Delta_i}{t_i} \cdot \frac{x_{i+1}'Q}{t_{i+1}} \right\| = \epsilon + \Omega \frac{|\Delta_i|}{t_i} \cdot \left\| \frac{x_{i+1}'Q}{\sqrt{x_{i+1}'Qx_{i+1}}} \right\|.
\end{aligned}$$

□

Let t^* be a minimizer of g on $\mathbb{R}_+$. We now show that the sequence of values of t produced by Algorithm 3, $\{t_i\}_{i\in\mathbb{N}}$, is monotone and bounded by t^*.

Proposition 20 (*Monotonicity*)**.** *If $t_i \leq t^*$, then $t_{i+1} = \sqrt{x_{i+1}'Qx_{i+1}}$ satisfies $t_i \leq t_{i+1} \leq t^*$. Similarly, if $t_i \geq t^*$, then $t_i \geq t_{i+1} \geq t^*$.*

Proof. If $t_i \leq t^*$, then $\frac{\Omega}{2t_i} \geq \frac{\Omega}{2t^*}$. It follows that $x(t_{i+1})$ is a minimizer of an optimization problem with a larger coefficient for the quadratic term than $x(t^*)$, and therefore $x_{i+1}'Qx_{i+1} = t_{i+1}^2 \leq t^{*2} = x^{*\prime}Qx^*$, and $t_{i+1} \leq t^*$. Moreover, the inequality $t_i \leq t_{i+1}$ follows from the convexity of the one-dimensional function g and the fact that function g is minimized at t^*, and that $g(t_{i+1}) \leq g(t_i)$. The case $t_i \geq t^*$ is similar. □

Since the sequence $\{t_i\}_{i\in\mathbb{N}}$ is bounded and monotone, it converges to a supremum or infimum. Thus $\{t_i\}_{i\in\mathbb{N}}$ is a Cauchy sequence, and $\lim_{i\to\infty} \Delta_i = 0$. Corollaries 5 and 6 below state that Algorithm 3 converges to an optimal solution. The cases where there exists a KKT point for PO (i.e., there exists an optimal solution with $t^* > 0$) and where there are no KKT points are handled separately.

Corollary 5 (Convergence to a KKT point)**.** *If PO has a KKT point, then Algorithm 3 converges to a KKT point.*

Proof. By convexity, the set of optimal solutions to (3.5) is an interval, $[t_\ell, t_u]$. Since by assumption there exists a KKT point, we have that $t_u > 0$. The proof is by cases, depending on the value of t_0 in line 1 of Algorithm 3.

Case $t_\ell \leq t_0 \leq t_u$ Since t_0 is optimal, we have by Proposition 20 that $t_1 = t_0$. Since $\Delta_0 = 0$ and $t_0 = \sqrt{x_{i+1}'Qx_{i+1}} > 0$, we have that $\|\nabla f(x_{i+1})\| < \infty$ in Proposition 19, and $\frac{|\Delta_i|}{t_i} \cdot \|\nabla f(x_{i+1})\| = 0$.

Case $t_0 < t_\ell$ We have by Proposition 20 than for all $i \in \mathbb{N}$, $t_i = \sqrt{x_i' Q x_i} \geq t_0 > 0$. Therefore, there exists a number M such that $\frac{1}{t_i} \|\nabla f(x_{i+1})\| < M$ for all $i \in \mathbb{N}$, and we find that $\frac{|\Delta_i|}{t_i} \cdot \|\nabla f(x_{i+1})\| \xrightarrow{\Delta_i \to 0} 0$.

Case $t_0 > t_u$ We have by Proposition 20 than for all $i \in \mathbb{N}$, $t_i = \sqrt{x_i' Q x_i} \geq t_u > 0$. Therefore, there exists a number M such that $\frac{1}{t_i} \|\nabla f(x_{i+1})\| < M$ for all $i \in \mathbb{N}$, and we find that $\frac{|\Delta_i|}{t_i} \cdot \|\nabla f(x_{i+1})\| \xrightarrow{\Delta_i \to 0} 0$.

Therefore, in all cases, Algorithm 3 convergences to a KKT point by Proposition 19. □

Corollary 6 (Convergence to 0)**.** *If $t^* = 0$ is the unique optimal solution to $\min\{g(t) : t \in \mathbb{R}_+\}$, then for any $\xi > 0$ Algorithm 3 finds a solution $(\bar{x}, \bar{t})$, where $\bar{t} < \xi$ and $\bar{x} \in \arg\min \left\{c'x : \sqrt{x'Qx} = \bar{t}, x \in X\right\}$.*

Proof. The sequence $\{t_i\}_{i \in \mathbb{N}}$ converges to 0 (otherwise, by Corollary 5, it would converge to a KKT point). Thus, $\lim_{i \to \infty} \sqrt{x_i' Q x_i} = 0$ and all points obtained in line 4 of Algorithm 3 satisfy $x_{i+1} \in \arg\min \left\{c'x : \sqrt{x'Qx} = t_{i+1}, x \in X\right\}$. □

We now discuss how to initialize and terminate Algorithm 3, corresponding to lines 1 and 7, respectively.

Initialization.

The algorithm may be initialized by an arbitrary $t_0 > 0$. Nevertheless, when a good initial guess on the value of t^* is available, t_0 should be set to that value. Moreover, observe that setting $t_0 = \infty$ results in a fast computation of x_1 by solving an LP.

Stopping condition.

Proposition 19 suggests a good stopping condition for Algorithm 3. Given a desired dual feasibility tolerance of $\delta > \epsilon$, we can stop when $\epsilon + \frac{|\Delta_i|}{t_i} \cdot \|\nabla f(x_{i+1})\| < \delta$. Alternatively, if $\exists k$ s.t. $\max_{x \in X} \|\nabla f(x)\| \leq k < \infty$, then the simpler $\left|\frac{\Delta_i}{t_i}\right| \leq \frac{\delta - \epsilon}{k}$ is another stopping condition. For instance, a crude upper bound on $\nabla f(x) = \Omega\left\|\frac{x'Q}{\sqrt{x'Qx}}\right\|$ can be found by maximizing/minimizing the numerator $x'Q$ over X and minimizing $x'Qx$ over X. The latter minimization is guaranteed to have a nonzero optimal value if $0 \notin X$ and Q is positive definite.

Bisection algorithm

Algorithm 4 is an accelerated bisection approach to solve PO. The algorithm maintains lower and upper bounds, $t_{\min}$ and $t_{\max}$, on t^* and, at each iteration, reduces the interval $[t_{\min}, t_{\max}]$ by at least half. The algorithm differs from the traditional bisection search algorithm in lines 7–11, where it uses an acceleration step to reduce the interval by a larger amount: by Proposition 20, if $t_0 \leq t_1$ (line 7), then $t_0 \leq t_1 \leq t^*$, and therefore t_1 is a higher lower bound on t^* (line 8); similarly, if $t_0 \geq t_1$, then t_1 is an lower upper bound on t^* (lines 9 and 10). Intuitively, the algorithm takes a "coordinate descent" step as in Algorithm 3 after each bisection step. Preliminary computations show that the acceleration step reduces the number of steps as well as the overall solution time for the bisection algorithm by about 50%.

Algorithm 4 Accelerated bisection.

Input: X polyhedron; Q psd matrix; c cost vector; $\Omega > 0$
Output: Optimal solution x^*

1: **Initialize** $t_{\min}$ and $t_{\max}$ ▷ ensure $t_{\min} \leq t^* \leq t_{\max}$
2: $\hat{z} \leftarrow \infty$ ▷ best objective value found
3: **repeat**
4: $t_0 \leftarrow \frac{t_{\min}+t_{\max}}{2}$
5: $x_0 \leftarrow \arg\min_{x \in X} \left\{ c'x + \frac{\Omega}{2t_0} x'Qx + \frac{\Omega}{2} t_0 \right\}$ ▷ solve QP
6: $t_1 \leftarrow \sqrt{x_0'Qx_0}$
7: **if** $t_0 \leq t_1$ **then** ▷ accelerate bisection
8: $t_{\min} \leftarrow t_1$
9: **else**
10: $t_{\max} \leftarrow t_1$
11: **end if**
12: **if** $c'x_0 + \Omega\sqrt{x_0'Qx_0} \leq \hat{z}$ **then** ▷ update the incumbent solution
13: $\hat{z} \leftarrow c'x_0 + \Omega\sqrt{x_0'Qx_0}$
14: $\hat{x} \leftarrow x_0$
15: **end if**
16: **until** stopping condition is met
17: **return** $\hat{x}$

Initialization.

In line 1, $t_{\min}$ can be initialized to zero and $t_{\max}$ to $x_{LP}'Qx_{LP}$, where x_{LP} is an optimal solution to the LP relaxation $\min_{x \in X} c'x$.

Stopping condition.

There are different possibilities for the stopping criterion in line 16. Note that if we have numbers t_m and t_M such that $t_m \leq t^* \leq t_M$, then $c'x(t_M) + \Omega\sqrt{x(t_m)'Qx(t_m)}$ is a lower bound on the optimal objective value $c'x^* + \Omega\sqrt{x^{*\prime}Qx^*}$. Therefore, in line 5, a lower bound z_l on the objective function can be computed, and the algorithm can be stopped when the gap between $\hat{z}$ and z_l is smaller than a given threshold. Alternatively, stopping when $\frac{|t_1-t_0|}{t_0} \cdot \Omega \left\| \frac{x_0{}'Q}{\sqrt{x_0{}'Qx_0}} \right\| < \delta - \epsilon$ provides a guarantee on the dual infeasibility as in Proposition 19.

Warm starts

Although any QP solver can be used to run the coordinate descent and bisection algorithms, simplex methods for QP are particularly effective as they allow warm starts for small changes in the model parameters in iterative applications. This is the main motivation for the QP based algorithms presented above.

Warm starts with primal simplex for convex optimization

All QPs solved in Algorithms 3–4 have the same feasible region and only the objective function changes in each iteration. Therefore, an optimal basis for a QP is primal feasible for the next QP solved in the sequence, and can be used to warm start a primal simplex QP solver.

Warm starts with dual simplex for discrete optimization

When solving discrete counterparts of CO with a branch-and-bound algorithm one is particularly interested in utilizing warm starts in solving convex relaxations at the nodes of the search tree. In a branch-and-bound algorithm, children nodes typically have a single additional bound constraint compared to the parent node.

For this purpose, it is also possible to warm start Algorithm 3 from a dual feasible basis. Let (x^*, t^*) be an optimal solution to PO and B^* be an optimal basis. Consider a new problem

$$\min \left\{ c'x + \frac{\Omega}{2t} x'Qx + \frac{\Omega}{2} t : x \in \bar{X},\ t \geq 0 \right\}, \tag{3.7}$$

where the feasible set $\bar{X}$ is obtained from X by adding new constraints. Note that B^* is a dual feasible basis for (3.7) when $t = t^*$. Therefore, Algorithm 3 to solve problem (3.7) can be warm started by initializing $t_0 = t^*$ and using B^* as the initial basis to compute x_1

with a dual simplex algorithm. The subsequent QPs can be solved using the primal simplex algorithm as described above.

3.4 Computational experiments

In this section we report on computational experiments with solving convex CO and its discrete counterpart CDO with the algorithms described in Section 3.3. The algorithms are implemented with CPLEX Java API. We use the simplex and barrier solvers of CPLEX version 12.6.2 for the computational experiments. All experiments are conducted on a workstation with a 2.93GHz Intel®Core$^{\text{TM}}$ i7 CPU and 8 GB main memory using a single thread.

Test problems

We test the algorithms on two types of data sets. For the first set the feasible region is described by a cardinality constraint and bounds, i.e., $X = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = b,\ \mathbf{0} \leq x \leq \mathbf{1}\}$ with $b = n/5$. For the second data set the feasible region consists of the path polytope of an acyclic grid network. For discrete optimization problems we additionally enforce the binary restrictions $x \in \mathbb{B}^n$.

For both data sets the objective function $q(x) = c'x + \Omega\sqrt{x'Qx}$ is generated as follows: Given a rank parameter r and density parameter α, Q is the sum of a low rank factor matrix and a full rank diagonal matrix; that is, $Q = F\Sigma F' + D$, where

- D is an $n \times n$ diagonal matrix with entries drawn from Uniform$(0, 1)$.
- $\Sigma = HH'$ where H is an $r \times r$ matrix with entries drawn from Uniform$(-1, 1)$.
- F is an $n \times r$ matrix in which each entry is 0 with probability $1 - \alpha$ and drawn from Uniform$(-1, 1)$ with probability α.

Each linear coefficient c_i is drawn from Uniform$(-2\sqrt{Q_{ii}}, 0)$.

Experiments with convex problems

In this section we present the computational results for convex instances. We compare the following algorithms:

ALG1 Algorithm 3.

ALG2 Algorithm 4.

BAR CPLEX' barrier algorithm (the default solver for convex conic quadratic problems).

For algorithms ALG1 and ALG2 we use CPLEX' primal simplex algorithm as the QP solver.

Optimality tolerance

As the speed of the interior point methods crucially depends on the chosen optimality tolerance, it is prudent to first compare the speed vs the quality of the solutions for the algorithms tested. Here we study the impact of the optimality tolerance in the solution time and the quality of the solutions for CPLEX' barrier algorithm BAR and simplex QP-based algorithm ALG1. The optimality tolerance of the barrier algorithm is controlled by the QCP convergence tolerance parameter ("BarQCPEpComp"), and in Algorithm 3, by the stopping condition $\frac{|\Delta_i|}{t} \leq \delta$.

In both cases, a smaller optimality tolerance corresponds to a higher quality solution. We evaluate the quality of a solution as $\texttt{optgap} = |(z_{\min} - z)/z_{\min}|$, where z is the objective value of the solution found by an algorithm with a given tolerance parameter and $z_{\min}$ is the objective value of the solution found by the barrier algorithm with tolerance 10^{-12} (minimum tolerance value allowed by CPLEX). Table 3.1 presents the results for different tolerance values for a 30×30 convex grid instance with $r = 200$, $\alpha = 0.1$, and $\Omega = 1$. The table shows, for varying tolerance values and for each algorithm, the quality of the solution, the solution time in seconds, the number of iterations, and QPs solved (for ALG1). We highlight in bold the default tolerance used for the rest of the experiments presented in this chapter. The tolerance value 10^{-7} for the barrier algorithm corresponds to the default parameter in CPLEX.

First observe that the solution time increases with reduced optimality tolerance for both algorithms. With lower tolerance, while the barrier algorithm performs more iterations, ALG1 solves more QPs; however, the total number of simplex iterations barely increases. For ALG1 the changes in the value of t are very small between QPs, and the optimal bases of the QPs are thus the same. Therefore, using warm starts, the simplex method is able to find high precision solutions inexpensively. ALG1 achieves much higher precision an order of magnitude faster than the barrier algorithm. For the default tolerance parameters used in our computational experiments, Algorithm 3 is several orders of magnitude more precise than the barrier algorithm.

Table 3.1: The effect of optimality tolerance.

Tolerance	BAR			ALG1			
	optgap	time	#iter	optgap	time	#iter	#QP
10^{-1}	8.65×10^{-2}	29.9	10	5.48×10^{-5}	3.2	835	4
10^{-2}	8.77×10^{-3}	41.5	15	3.24×10^{-7}	4.2	844	6
10^{-3}	6.98×10^{-4}	54.6	23	2.60×10^{-9}	4.3	844	8
10^{-4}	5.52×10^{-5}	62.9	27	2.12×10^{-11}	4.7	844	10
10^{-5}	3.72×10^{-6}	66.8	29	$\mathbf{6.80 \times 10^{-13}}$	**5.2**	**844**	**12**
10^{-6}	7.12×10^{-7}	69.6	30	5.32×10^{-13}	5.4	844	13
10^{-7}	$\mathbf{2.04 \times 10^{-8}}$	**72.0**	**32**	5.15×10^{-13}	6.0	844	15
10^{-8}	2.65×10^{-9}	74.0	33	5.15×10^{-13}	6.2	844	17
10^{-9}	2.42×10^{-10}	75.9	34	5.15×10^{-13}	6.6	844	19
10^{-10}	1.97×10^{-11}	78.7	35	5.15×10^{-13}	7.0	844	21
10^{-11}	9.61×10^{-12}	79.6	36	5.15×10^{-13}	7.4	844	23
10^{-12}	0	89.6	39	5.15×10^{-13}	7.8	844	25

Effect of the nonlinearity parameter Ω.

We now study the effect of changing the nonlinearity parameter Ω. Tables 3.2 and 3.3 show the total solution time in seconds, the total number of simplex or barrier iterations, and the number of QPs solved in cardinality (1000 variables) and path instances (1760 variables), respectively. Each row represents the average over five instances for a rank (r) and density(α) configuration and algorithm used. For each parameter choice the fastest algorithm is highlighted in bold.

First observe that in both data sets the barrier algorithm is the slowest: it is 3.5 and 6 times slower than the simplex QP-based methods for the cardinality instances, and is up to 15 times slower for the path instances. The barrier algorithm does not appear to be too sensitive to the nonlinearity parameter Ω, whereas the simplex QP-based methods are faster for smaller Ω.

The number of simplex iterations in ALG1 increases with the nonlinearity parameter Ω. Indeed, the initial problem solved by ALG1 is an LP (corresponding to $\Omega = 0$), so as Ω increases the initial problem becomes a worse approximation, and more work is needed to converge to an optimal solution. Also note that Algorithm 4 requires fewer QPs to be solved, but as a result it benefits less from warm starts (it requires more simplex iterations per QP than ALG1). Indeed, in ALG2 the value of t changes by a larger amount at each iteration (with respect to ALG1), so the objective function of two consecutive QPs changes

Table 3.2: The effect of nonlinearity (cardinality instances).

r	α	Method	$\Omega = 1$			$\Omega = 2$			$\Omega = 3$		
			time	#iter	#QP	time	#iter	#QP	time	#iter	#QP
100	0.1	ALG1	1.0	22	20	1.1	53	24	1.3	104	29
		ALG2	**0.8**	**41**	**14**	**0.9**	**95**	**15**	**0.9**	**150**	**15**
		BAR	4.6	16	-	4.9	24	-	5.2	26	-
100	0.5	ALG1	1.1	33	23	1.1	69	24	1.5	144	37
		ALG2	**0.8**	**60**	**14**	**0.9**	**125**	**15**	**0.9**	**200**	**15**
		BAR	4.5	21	-	5.1	25	-	5.8	29	-
200	0.1	ALG1	0.9	33	19	1.1	73	25	1.2	110	25
		ALG2	**0.8**	**49**	**14**	**0.9**	**126**	**14**	**0.9**	**172**	**14**
		BAR	4.7	22	-	4.5	22	-	5.1	25	-
200	0.5	ALG1	1.0	48	22	1.1	99	22	1.2	151	25
		ALG2	**0.9**	**94**	**14**	**0.9**	**179**	**14**	**1.0**	**233**	**15**
		BAR	4.4	21	-	4.9	24	-	5.2	26	-
avg		ALG1	1.0	34	21	1.1	73	24	1.3	127	29
		ALG2	**0.8**	**61**	**14**	**0.9**	**131**	**15**	**0.9**	**189**	**15**
		BAR	4.3	20	-	4.9	24	-	5.3	27	-

Table 3.3: The effect of nonlinearity (path instances).

r	α	Method	$\Omega = 1$			$\Omega = 2$			$\Omega = 3$		
			time	#iter	#QP	time	#iter	#QP	time	#iter	#QP
100	0.1	ALG1	**4.4**	**940**	**12**	**6.4**	**1,307**	**16**	**7.5**	**1,505**	**18**
		ALG2	4.8	1,283	11	**6.4**	**1,637**	**13**	**7.5**	**1,865**	**14**
		BAR	68.4	26	-	56.7	21	-	46.3	16	-
100	0.5	ALG1	**4.7**	**902**	**14**	7.4	1,191	21	8.3	1,391	21
		ALG2	4.9	1,148	12	**6.5**	**1,474**	**13**	**7.7**	**1,772**	**14**
		BAR	54.3	19	-	48.8	16	-	47.4	16	-
200	0.1	ALG1	4.5	836	14	**5.7**	**1,053**	**15**	**7.3**	**1,220**	**18**
		ALG2	**4.4**	**932**	**12**	6.0	1,377	13	7.4	1,671	13
		BAR	63.7	25	-	49.8	18	-	54.5	20	-
200	0.5	ALG1	**4.1**	**858**	**12**	**5.5**	**1,048**	**15**	**6.8**	**1,237**	**16**
		ALG2	4.4	978	12	6.0	1,363	13	7.5	1,626	13
		BAR	70.5	26	-	60.2	21	-	52.4	18	-
avg		ALG1	**4.4**	**884**	**13**	**6.2**	**1,150**	**17**	**7.5**	**1,338**	**18**
		ALG2	4.6	1,086	12	**6.2**	**1,463**	**13**	**7.5**	**1,734**	**13**
		BAR	64.2	24	-	53.9	19	-	50.2	17	-

by a larger amount.

Effect of the dimension

Table 3.4 presents a comparison of the algorithms for the convex cardinality instances with sizes 400, 800, 1600, and 3200. Each row represents the average over five instances, as before, generated with parameters $r = 200$, $\alpha = 0.1$, and $\Omega = 2$. Additionally, Figure 3.1 shows the solution time for each algorithm and the speed-up factor of the simplex QP-based algorithms compared to the barrier algorithm as a function of the dimension (n).

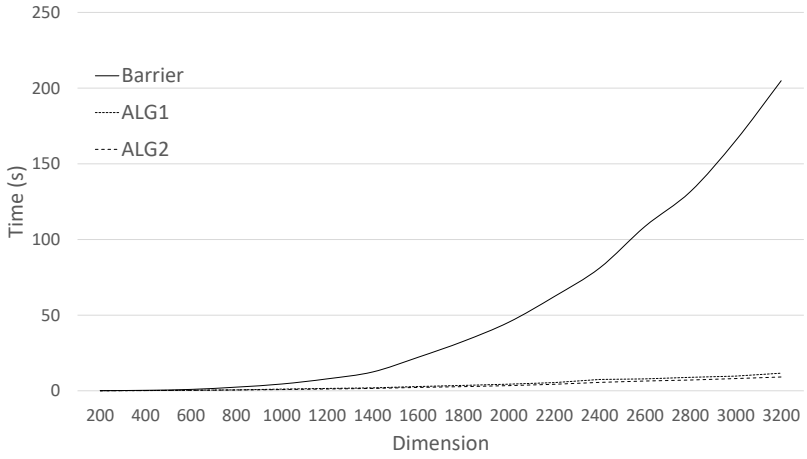
Table 3.4: The effect of dimension (cardinality instances).

Method	$n = 400$			$n = 800$			$n = 1600$			$n = 3200$		
	time	#iter	#QP	time	#iter	#QP	time	#iter	#QP	time	#iter	#QP
ALG1	**0.2**	**43**	**20**	0.6	65	19	2.8	75	25	11.7	104	25
ALG2	**0.2**	**73**	**14**	**0.5**	**116**	**14**	**2.2**	**129**	**15**	**9.1**	**175**	**15**
BAR	0.3	21	-	2.4	22	-	22.1	27	-	204.9	30	-

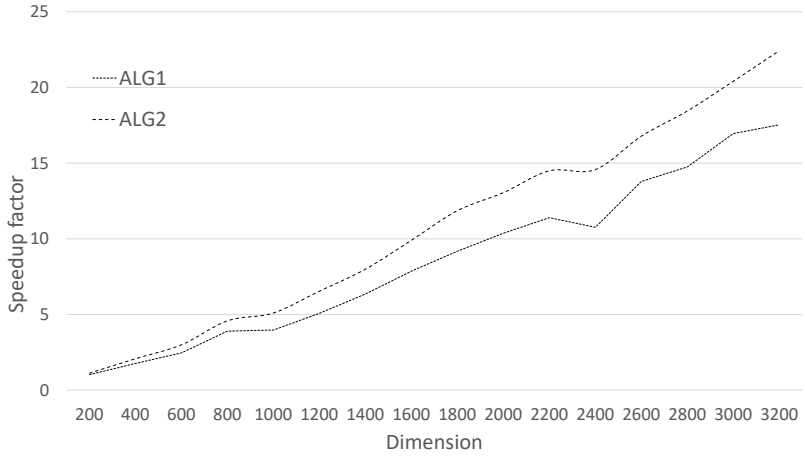
Observe in Table 3.4 that the number of QPs solved with the simplex-based algorithms does not depend on the dimension. The number of simplex iterations, however, increases with the dimension. For $n = 400$ all algorithms perform similarly and the problems are solved very fast. However, as the dimension increases, the simplex-based algorithms outperform the barrier algorithm, often by many factors. For $n = 3200$, the fastest simplex-based algorithm ALG2 is more than 20 times faster than the barrier algorithm. Similar results are obtained for other parameter choices and for the path instances as well. In summary, the simplex-based algorithms scale better with the dimension, and are faster by orders of magnitude for large instances.

Discrete instances

In this section we describe our experiments with the discrete counterpart CDO. As of version 12.6.2 of CPLEX, it is not possible to employ a user-defined convex solver such as Algorithm 3 at the nodes of the CPLEX' branch-and-bound algorithm. Therefore, in order to test the proposed approach for CDO, we implement a rudimentary branch-and-bound algorithm described in Appendix A.2. The algorithm uses a maximum infeasibility rule for branching, and does not employ presolve, cutting planes, or heuristics. We test the following configurations:



(a) Solution time as a function of dimension.



(b) Speed-up as a function of dimension.

Figure 3.1: Barrier vs the simplex QP-based algorithms.

BBA1 Branch-and-bound algorithm in Appendix A.2 using Algorithm 3 as the convex solver. The first QP at each node (except the root node) is solved with CPLEX dual simplex method using the parent dual feasible basis as a warm start (as mentioned in Section 3.3) and all other QPs are solved with CPLEX primal simplex method using the basis from the parent node QP as a warm start.

BBBR Branch-and-bound algorithm in Appendix A.2, using CPLEX barrier algorithm as the convex solver. This configuration does not use warm starts.

CXBR CPLEX branch-and-bound algorithm with barrier solver, setting the branching rule to maximum infeasibility, the node selection rule to best bound, and disabling presolve, cuts and heuristics. In this setting CPLEX branch-and-bound algorithm is as close as possible to our branch-and-bound algorithm.

CXLP CPLEX branch-and-bound algorithm with LP outer approximations, setting the branching rule to maximum infeasibility, the node selection rule to best bound, and disabling presolve, cuts and heuristics. In this setting CPLEX branch-and-bound algorithm is as close as possible to our branch-and-bound algorithm.

CXLPE CPLEX branch-and-bound algorithm with LP outer approximations, setting the branching rule to maximum infeasibility, the node selection rule to best bound, and disabling cuts and heuristic. Since presolve is activated, CPLEX uses extended formulations described in Vielma et al. (2016). Besides presolve, all other parameters are set as in CXLP.

CXD CPLEX default branch-and-bound algorithm with LP outer approximations.

In all cases the time limit is set to two hours.

Table 3.5 presents the results for discrete cardinality instances with 200 variables and Table 3.6 for the discrete path instances with 1,740 variables (30×30 grid). Each row represents the average over five instances with varying rank and density parameters, and algorithm. The tables show the solution time in seconds, the number of nodes explored in the branch-and-bound tree, the end gap after two hours as percentage, and the number of instances that are solved to optimality for varying values of Ω. For each instance class we highlight in bold the algorithm with the best performance.

First of all, observe that the difficulty of the instances increases considerably for higher values of Ω due to higher integrality gap. The problems corresponding to high values of the density parameter α are also more challenging.

Performance of CPLEX branch-and-bound

Among CPLEX branch-and-bound algorithms, CXD is the best choice when $\Omega \geq 2$. Configuration CXD is much more sophisticated than the other configurations, so a better performance is expected. However, note that for $\Omega = 1$ configuration CXD is not necessarily the best. In particular in the path instances (Table 3.6) CXLP and CXLPE are 2.3 times

Table 3.5: Comparison for discrete cardinality instances.

r	α	Method	$\Omega = 1$				$\Omega = 2$				$\Omega = 3$			
			time	nodes	egap	#s	time	nodes	egap	#s	time	nodes	egap	#s
100	0.1	BBA1	**1**	**156**	**0.0**	**5**	**26**	**3,271**	**0.0**	**5**	652	68,318	0.0	5
		BBBR	16	156	0.0	5	349	3,270	0.0	5	4,664	43,695	0.1	3
		CXBR	35	276	0.0	5	513	3,497	0.0	5	5,260	32,782	0.2	2
		CXLP	34	9,562	0.0	5	7,200	209,576	0.7	0	7,200	244,911	2.2	0
		CXLPE	2	374	0.0	5	81	7,640	0.0	5	2,629	111,293	0.0	5
		CXD	3	368	0.0	5	37	5,152	0.0	5	**604**	**58,076**	**0.0**	**5**
100	0.5	BBA1	**1**	**87**	**0.0**	**5**	**51**	**6,274**	**0.0**	**5**	**1,323**	**140,874**	**0.0**	**5**
		BBBR	10	87	0.0	5	686	6,274	0.0	5	6,134	56,394	0.3	1
		CXBR	24	183	0.0	5	1,027	6,734	0.0	5	6,399	39,710	0.4	1
		CXLP	294	26,957	0.0	5	7,200	229,641	0.8	0	7,200	263,810	2.3	0
		CXLPE	2	349	0.0	5	191	14,737	0.0	5	4,967	215,292	0.1	3
		CXD	3	373	0.0	5	144	16,070	0.0	5	3,300	245,251	0.0	5
200	0.1	BBA1	**1**	**247**	**0.0**	**5**	**20**	**3,259**	**0.0**	**5**	**388**	**55,248**	**0.0**	**5**
		BBBR	24	247	0.0	5	321	3,259	0.0	5	4,573	39,647	0.1	3
		CXBR	52	460	0.0	5	540	3,711	0.0	5	5,295	34,090	0.2	2
		CXLP	221	17,205	0.0	5	7,200	208,874	0.6	0	7,200	230,304	2.0	0
		CXLPE	4	473	0.0	5	122	6,064	0.0	5	3,376	111,205	0.0	4
		CXD	4	360	0.0	5	52	6,413	0.0	5	1,062	67,577	0.0	5
200	0.5	BBA1	**4**	**674**	**0.0**	**5**	**170**	**24,636**	**0.0**	**5**	**1,140**	**156,632**	**0.0**	**5**
		BBBR	77	674	0.0	5	2,106	17,743	0.0	4	5,590	47,725	0.2	2
		CXBR	104	680	0.0	5	2,452	15,816	0.0	4	6,127	38,973	0.3	1
		CXLP	3,514	120,007	0.1	4	7,200	212,082	1.0	0	7,200	240,445	2.3	0
		CXLPE	18	1,461	0.0	5	1,722	61,593	0.0	4	5,020	198,891	0.2	2
		CXD	16	1,612	0.0	5	1,068	75,098	0.0	5	4,647	299,723	0.1	4
avg		BBA1	**2**	**291**	**0.0**	**20**	**67**	**9,360**	**0.0**	**20**	**876**	**105,268**	**0.0**	**20**
		BBBR	32	291	0.0	20	865	7,637	0.0	19	5,240	46,865	0.2	9
		CXBR	54	400	0.0	20	1,133	7,440	0.0	19	5,770	36,389	0.3	6
		CXLP	1,016	43,433	0.0	19	7,200	215,043	0.8	0	7,200	244,867	2.2	0
		CXLPE	7	664	0.0	20	529	22,508	0.0	19	3,998	159,170	0.1	14
		CXD	7	678	0.0	20	325	25,683	0.0	20	2,403	167,657	0.0	19

faster than CXD. This result suggests that in simple instances the additional features used by CXD (e.g. cutting planes and heuristics) may be hurting the performance.

The extended formulations result in much stronger relaxations in LP based branch-and-bound and, consequently, the number of branch-and-bound nodes required with CXLPE is only a small fraction of the number of nodes required with CXLP. However, CXLPE requires

Table 3.6: Comparison for discrete path instances.

r	α	Method	$\Omega = 1$				$\Omega = 2$				$\Omega = 3$			
			time	nodes	egap	#s	time	nodes	egap	#s	time	nodes	egap	#s
100	0.1	BBA1	**256**	**145**	**0.0**	**5**	3,616	1,774	0.0	5	**7,200**	**2,988**	**5.4**	**0**
		BBBR	3,577	91	0.2	4	7,200	184	4.9	0	7,200	236	11.7	0
		CXBR	7,200	67	20.8	0	7,200	79	∞	0	7,200	129	∞	0
		CXLP	533	2,428	0.0	5	7,200	24,776	4.5	0	7,200	20,099	15.0	0
		CXLPE	713	315	0.0	5	6,337	2,432	1.9	1	7,200	2,837	23.2	0
		CXD	1,309	164	0.0	5	**3,361**	**1,176**	**0.0**	**5**	7,200	2,644	6.4	0
100	0.5	BBA1	**589**	**353**	**0.0**	**5**	4,799	2,317	0.4	3	**6,472**	**2,698**	**4.6**	**1**
		BBBR	6,071	134	0.7	2	7,200	175	4.9	0	7,200	162	11.5	0
		CXBR	7,200	24	∞	0	7,200	56	∞	0	7,200	70	∞	0
		CXLP	1,132	6,187	0.0	5	7,200	23,671	4.4	0	7,200	16,851	12.8	0
		CXLPE	903	607	0.0	5	6,207	2,906	2.4	1	7,200	3,128	16.1	0
		CXD	1,532	267	0.0	5	5,189	2,123	0.3	4	7,200	2,645	5.6	0
200	0.1	BBA1	**149**	**77**	**0.0**	**5**	**1823**	**1,075**	**0.0**	**5**	**6,411**	**3,401**	**2.5**	**1**
		BBBR	3,245	76	0.0	5	7,200	171	2.4	0	7,200	180	10.5	0
		CXBR	7,200	34	∞	0	7,200	45	∞	0	7,200	68	∞	0
		CXLP	436	1,548	0.0	5	7,200	30,265	2.9	0	7,200	20,579	12.4	0
		CXLPE	487	188	0.0	5	4,565	1,681	1.0	3	7,200	2,402	17.7	0
		CXD	1,965	106	0.0	5	4,481	1,453	0.4	4	7,200	2,532	4.2	0
200	0.5	BBA1	**292**	**196**	**0.0**	**5**	**3,862**	**2,337**	**0.3**	**4**	**7,200**	**3,703**	**3.7**	**0**
		BBBR	4,826	113	0.2	3	7,200	173	3.9	0	7,200	176	12.7	0
		CXBR	7,200	20	∞	0	7,200	51	∞	0	7,200	89	∞	0
		CXLP	859	4,989	0.0	5	7,200	28,007	4.4	0	7,200	18,873	13.3	0
		CXLPE	923	399	0.0	5	5,730	2,363	1.7	3	7,200	2,691	17.9	0
		CXD	2,028	177	0.0	5	4,752	1,899	0.3	4	7,200	2,775	6.3	0
avg		BBA1	**322**	**193**	**0.0**	**20**	**3,525**	**1,876**	**0.2**	**17**	**6,821**	**3,198**	**4.1**	**2**
		BBBR	4,430	103	0.3	14	7,200	176	4.0	0	7,200	189	11.6	0
		CXBR	7,200	36	∞	0	7,200	58	∞	0	7,200	89	∞	0
		CXLP	740	3,788	0.0	20	7,200	26,680	4.1	0	7,200	19,101	13.4	0
		CXLPE	757	377	0.0	20	5,710	2,346	1.8	8	7,200	2,765	18.7	0
		CXD	1,708	178	0.0	20	4,446	1663	0.3	17	7,200	2,650	5.6	0

more time to solve each branch-and-bound node, due to the higher number of variables and the additional effort needed to refine the LP outer approximations. For the cardinality instances, CXLPE is definitely the better choice and is faster by orders of magnitude. For the path instances, however, CXLP is not necessarily inferior: when $\Omega = 1$ CXLP is competitive with CXLPE, and when $\Omega = 3$ CXLP performs better.

The barrier-based branch-and-bound CXBR, in general, performs poorly. For the cardinality instances, it outperforms CXLP but is slower than the other algorithms. For the path instances it has the worst performance, often struggling to find even a single feasible solution (resulting in infinite end gaps).

Performance of BBA1

Note that BBA1 and BBBR are very simple and differ only by the convex node solver. BBA1 is faster than BBBR by an order of magnitude. BBA1 is also considerably faster than the simplest CPLEX branch-and-bound algorithms CXBR and CXLP.

We see that BBA1 outperforms CXLPE (which uses presolve and extended formulations) in all instances. Observe that in the cardinality instances with $\Omega = 1, 2$ and path instances with $\Omega = 1$, BBA1 requires half the number of nodes (or less) compared to CXLPE to solve the instances to optimality (since the relaxations solved at each node are stronger), which translates into faster overall solution times. In the more difficult instances BBA1 is able to solve more instances to optimality, and the end gaps are smaller.

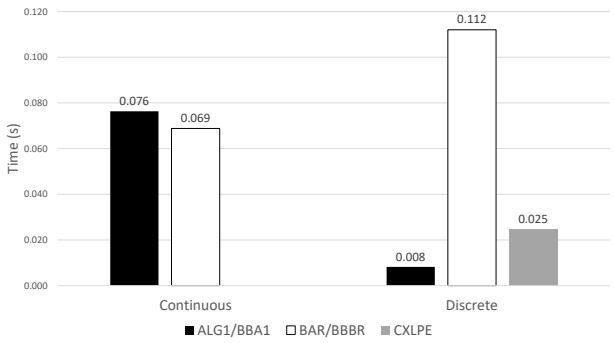
Despite the fact that BBA1 is a rudimentary branch-and-bound implementation, it is faster than default CPLEX in most of the cases. Indeed, BBA1 is the better choice in 21 of the instance classes considered, while CXD is better in only 2. Moreover, in the instances where CXD is better the difference between the algorithms is small (around 10% difference in solution times), while in the other instances BBA1 is often faster by many factors. We observe that CXD is comparatively better for the instances with a low factor rank ($r = 100$), and BBA1 is comparatively better for the instances with a high factor rank ($r = 200$).

Warm starts

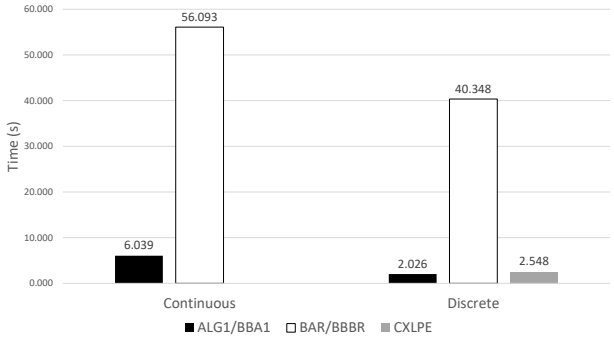
Algorithm BBA1 is faster than BBBR in part due to a faster convex solver (as observed in the results for convex instances), and in part due to node warm starts. To quantify the impact of warm starts, we plot in Figure 3.2 the *time per node* (computed as solution time divided by the number of branch-and-bound nodes) for BBA1, BBBR and CXLPE, and also plot the solution time for the corresponding convex instances with solvers ALG1 and BAR[1].

For the small cardinality instances with 200 variables, Algorithm 3 is slightly worse than the barrier algorithm to solve the convex relaxations; however, it is 15 times faster than

[1]The time per node is similar for all combinations of parameters Ω, r and α, and thus we plot the average over all parameters.



(a) Cardinality instances



(b) Path instances

Figure 3.2: Time per node.

barrier when used in branch-and-bound due to the node warm starts from dual feasible solutions. For the larger path instances with 1,740 variables, Algorithm 3 is 10 times faster than the barrier algorithm to solve the convex relaxations, and is about 20 times faster for the discrete instances. Thus node warm starts make the algorithm twice as fast. Finally, observe that the solve time per node for BBA1 is smaller compared to CXLPE: the proposed simplex-based algorithm is thus as effective as the simplex method for extended formulations in exploiting warm starts. Moreover, it solves the nonlinear convex relaxations at each node to optimality, whereas CXLPE solves its LP relaxation. The improved lower bounds lead to significantly small search trees.

We conclude that Algorithm 3 is indeed suitable for branch-and-bound algorithms since it benefits from node warms starts from the parent nodes, resulting in a significant improvement in solution times.

3.5 Conclusions

We consider minimization problems with a conic quadratic objective and linear constraints, which are natural generalizations of linear programming and quadratic programming. Using the perspective function we reformulate the objective and propose simplex QP-based algorithms that solve a quadratic program at each iteration. Computational experiments indicate that the proposed algorithms are faster than interior point methods by orders of magnitude, scale better with the dimension of the problem, return higher precision solutions, and, most importantly, are amenable to warm starts. Therefore, they can be embedded in branch-and-bound algorithms quite effectively.

Chapter 4

Polymatroid cuts for conic quadratic minimization

4.1 Introduction

Mixed-integer second order cone optimization problems can be written as

$$\begin{aligned} & \min\ f'x + g'z \\ \text{(MISOCO)} \quad & \text{s.t. } \sqrt{x'Q_i x} \le z_i, && i = 1, \dots, \ell \qquad (4.1) \\ & (x, z) \in X \subseteq \mathbb{Z}^n_+ \times \mathbb{R}^m_+ \times \mathbb{R}^\ell_+ \end{aligned}$$

where $Q_i \succeq 0$ for $i = 1, \dots, \ell$. We refer to inequalities (4.1) as the second order conic constraints. Many design and estimation optimization problems are modeled as MISOCO (Lobo et al. 1998, Alizadeh and Goldfarb 2003, Atamtürk et al. 2012). In particular, second order conic constraints are frequently used to model probabilistic optimization with Gaussian distributions (Birge and Louveaux 2011) and robust optimization problems with ellipsoidal uncertainty sets (Ben-Tal and Nemirovski 1998, 1999, Ben-Tal et al. 2009).

Linear mixed-integer optimization (MILO) is a special case of MISOCO. Strong formulations have proven to be one of the critical components in solving MILO, and state-of-the-art solvers for MILO employ a variety of valid inequalities as cutting planes. However, relatively few classes of strong valid inequalities are known to strengthen the convex relaxations of MISOCO and, more generally, nonlinear mixed-integer optimization.

General valid inequalities for convex nonlinear and/or conic mixed-integer optimization include intersection cuts, disjunctive cuts, and lift-and-project cuts (Ceria and Soares 1999, Stubbs and Mehrotra 1999). Çezik and Iyengar (2005) discuss Gomory cuts for general

conic optimization problems. Atamtürk and Narayanan (2010) give conic MIR cuts for conic mixed-integer optimization and Atamtürk and Narayanan (2011) study lifting for conic mixed-integer optimization. Dadush et al. (2011) investigate the split closure of a convex set. Belotti et al. (2015) study the intersection of a convex set and a linear disjunction. Kılınç et al. (2010) and Bonami (2011) discuss the separation of split cuts using outer approximations and nonlinear programming, respectively. Kılınç-Karzan and Yıldız (2015) study disjunctions on the second order cone.

Another stream of research involves generating strong cuts by exploiting structured sets as it is common for the linear integer case. Although the applicability of such cuts is restricted to certain classes of sets, they tend to be far more effective than the general cuts that ignore any special structure. Aktürk et al. (2009, 2010) give second-order representable perspective cuts for a nonlinear scheduling problem with variable upper bounds, which are generalized further by Günlük and Linderoth (2010). Ahmed and Atamtürk (2011) give strong lifted inequalities for maximizing a submodular concave utility function. Atamtürk and Narayanan (2009), Atamtürk and Bhardwaj (2015) study binary knapsack sets defined by a single second-order conic constraint. Modaresi et al. (2016) derive closed form intersection cuts for a number of structured sets.

To goal of the current chapter is to contribute to the understanding of convex hull of simple conic mixed-integer sets that form the building blocks of more general constraint sets as relaxations. In a related paper, Atamtürk and Narayanan (2008) give *extended polymatroid inequalities* for second order conic constraints (4.1) with diagonal Q_i matrices on binary variables, and show that these inequalities describe the convex hull in that case. In this chapter, we first extend their results to the mixed-binary case and show that a nonlinear generalization of the polymatroid inequalities is sufficient to describe the convex hull for the mixed-binary case with unbounded continuous variables. We then show how additional constraints, in particular, the upper bounds on the continuous variables, can be used to further generalize and strengthen the first class of inequalities. Interestingly, although the inequalities are derived for the diagonal case, they can be applied to the non-diagonal case as well through a suitable relaxation. Computational experiments indicate that the derived inequalities are quite effective for the diagonal and as well as the non-diagonal cases.

We should note that utilizing the diagonal entries of matrices is standard for constructing convex relaxations in quadratic optimization (e.g. Poljak and Wolkowicz 1995, Anstreicher

2012). In particular, for $x \in \{0,1\}^n$, we have

$$x'Qx \leq z \iff x'(Q - \text{diag}(a))x + a'x \leq z$$

with $a \in \mathbb{R}^n$ such that $Q - \text{diag}(a) \succeq 0$. This transformation is based on the ideal (convex hull) representation of the separable quadratic term $x'\text{diag}(a)x$ as $a'x$ for $x \in \{0,1\}^n$.

A similar approach is also available for convex quadratic optimization with indicator constraints. For $x \in \{0,1\}^n$ and $y \in \mathbb{R}^n$ s.t. $\ell \circ x \leq y \leq u \circ x$, we have

$$y'Qy \leq z \iff y'(Q - \text{diag}(a))y + a't \leq z,\ y_i^2 \leq x_i t_i$$

with $t \in \mathbb{R}^n_+$ (e.g. Aktürk et al. 2009, Günlük and Linderoth 2010). This transformation is based on the ideal representation of each quadratic term $a_i y_i^2$ subject to indicator constraints as a linear term $a_i t_i$ along with a rotated cone constraint $y_i^2 \leq x_i t_i$.

Since in the conic quadratic constraint (4.1), the terms are not separable even for the diagonal case, simple transformations as in the quadratic cases above are not sufficient to arrive at an ideal formulation. We show that it is necessary to exploit the submodularity of the underlying set function to arrive at the ideal representations.

The rest of the chapter is organized as follows. In Section 4.2 we review the results of Atamtürk and Narayanan (2008). In Section 4.3 we give the complete convex hull description of a single mixed-binary conic constraint. In Section 4.5 we study mixed-binary conic constraints with upper bounds on the continuous variables. In Section 4.6 we show how to include additional constraints to generalize and strengthen the inequalities. In Section 4.7 we report on a computational study done to test the effectiveness of the proposed inequalities for solving MISOCO, including instances with non-diagonal matrices.

4.2 Previous work

Let x denote an n-dimensional vector of binary variables, y denote an m-dimension vector of continuous variables, and c and d be *nonnegative* vectors of dimension n and m, respectively, $\sigma \geq 0$ be a constant. Define $N = \{1, \ldots, n\}$ and $M = \{1, \ldots, m\}$.

For $p > 1$ and $y \geq 0$, we study a p-order conic constraint of the form

$$\sqrt[p]{\sum_{i \in N} c_i x_i^p + \sum_{i \in M} d_i y_i^p} \leq z, \tag{4.2}$$

where the second order conic constraint corresponds to the case $p = 2$. Throughout, instead of the convex inequality (4.2) we will use

$$\sqrt[p]{\sum_{i\in N} c_i x_i + \sum_{i\in M} d_i y_i^p} \leq z. \tag{4.3}$$

Constraint (4.3) is equivalent to (4.2) over binary x, but it is stronger over the continuous relaxation of x since $x_i^p = x_i$ for $x_i \in \{0, 1\}$, but $x_i^p < x_i$ for $x_i \in (0, 1)$. Inequality (4.3) is concave in x, but convex in y. We will exploit both the concavity on x and the convexity on y.

Previous work

In this section we state, without proof, the main results of Atamtürk and Narayanan (2008) for the set

$$K_\sigma = \left\{ (x, z) \in \{0, 1\}^n \times \mathbb{R}_+ : \sqrt[p]{\sigma + \sum_{i\in N} c_i x_i} \leq z \right\}.$$

For a given a permutation $((1), (2), \ldots, (n))$ of N, let

$$\begin{aligned} \sigma_{(k)} &= \sigma + \sum_{i=1}^{k-1} c_{(i)}, \text{ and} \\ \pi_{(k)} &= \sqrt[p]{c_{(k)} + \sigma_{(k)}} - \sqrt[p]{\sigma_{(k)}}. \end{aligned} \tag{4.4}$$

and define the *extended polymatroid inequality* as

$$\sqrt[p]{\sigma} + \sum_{i=1}^{n} \pi_{(i)} x_{(i)} \leq z. \tag{4.5}$$

Let Π_σ be the set of such coefficient vectors π for *all* permutations of N.

Proposition 21 (Convex hull of K_σ)**.**

$$conv(K_\sigma) = \left\{ (x, z) \in [0, 1]^n \times \mathbb{R}_+ : \sqrt[p]{\sigma} + \pi' x \leq z, \ \ \forall \pi \in \Pi_\sigma \right\}.$$

The set function defining K_σ is submodular; therefore, Π_σ form the extreme points of an extended polymatroid. Since the maximization of a linear function over an extended polymatroid can be solved by the greedy algorithm (Edmonds 1970), a point $\bar{x} \in \mathbb{R}_+^n$ can be separated from conv(K_σ) via the greedy algorithm by sorting $\bar{x}_i$ in non-increasing order in $O(n \log n)$.

Proposition 22 (Separation)**.** *A point $\bar{x} \notin conv(K_\sigma)$ such that $\bar{x}_{(1)} \geq \bar{x}_{(2)} \geq \ldots \geq \bar{x}_{(n)}$ is separated from $conv(K_\sigma)$ by inequality* (4.5)*.*

Atamtürk and Narayanan (2008) also consider a mixed-integer extension and give valid inequalities for the mixed-integer set

$$L_\sigma = \left\{ (x, y, z) \in \{0,1\}^n \times [0,1]^m \times \mathbb{R}_+ : \sqrt[p]{\sigma + \sum_{i \in N} c_i x_i + \sum_{i \in M} d_i y_i^p} \leq z \right\}.$$

Without loss of generality, the upper bounds of the continuous variables in L_σ are set to one by scaling. For $T \subseteq M$, define $d(T) := \sum_{i \in T} d_i$.

Proposition 23 (Valid inequalities for L_σ)**.** *For $T \subseteq M$ inequalities*

$$\sqrt[p]{\sigma + \sum_{i \in T} d_i y_i^p} + \pi' x \leq z, \quad \pi \in \Pi_{\sigma + d(T)} \tag{4.6}$$

are valid for L_σ.

Inequalities (4.6) are obtained by setting the subset T of the continuous variables to their upper bounds and relaxing the rest and they dominate any inequality of the form

$$\sqrt[p]{\sigma + \sum_{i \in T} d_i y_i^p} + \xi' x \leq z$$

with $\xi \in \mathbb{R}^n$.

Finally, note that the optimization of a linear function over L_σ:

$$\min\{a'x + b'y + z \ : \ (x, y, z) \in L_\sigma\} \tag{4.7}$$

is solvable in polynomial time: For a fixed value of x, problem (4.7) reduces to a (convex) conic quadratic optimization problem in y that can be solved easily. On the other hand, for fixed a value of y problem (4.7) reduces to a submodular minimization problem that can be solved by the greedy algorithm (Shen et al. 2003). Without loss of generality, assume that $c_i > 0$ for all i, as otherwise x_i can be set to either 0 or 1, depending on the sign of a_i. Index the binary variables so that $\frac{a_1}{c_1} \leq \ldots \leq \frac{a_n}{c_n}$ (breaking ties arbitrarily) and let $S_i = \{1, \ldots, i\}$ for $i = 1, 2, \ldots, n$. There exists an optimal solution (x^*, y^*) to (4.7) such that $x_k^* = 1$ if $k \in S_i$ for some $i = 1, \ldots, n$, and $x_k^* = 0$ otherwise. Thus, problem (4.7) can be solved by fixing the binary variables according to sets S_i one at a time and then solving the remaining conic quadratic optimization problem in polynomial time.

4.3 Conic constraint with unbounded continuous variables

In this section we consider the mixed-integer set

$$H_\sigma = \left\{ (x, y, z) \in \{0,1\}^n \times \mathbb{R}^{m+1}_+ : \sqrt[p]{\sigma + \sum_{i \in N} c_i x_i + \sum_{i \in M} d_i y_i^p} \leq z \right\}.$$

Note that H_σ is the relaxation of L_σ by dropping the upper bounds on the continuous variables y. Thus, the only class of valid inequalities of type (4.6) are the extended polymatroid inequalities

$$\sqrt[p]{\sigma} + \pi' x \leq z, \quad \forall \pi \in \Pi_\sigma$$

from the "binary-only" relaxation by letting $T = \emptyset$. Here, we define a new class of *nonlinear* valid inequalities for H_σ and prove that they are sufficient to define its convex hull.

Consider the inequalities

$$\sqrt[p]{\left(\sqrt[p]{\sigma} + \pi' x\right)^p + \sum_{i \in M} d_i y_i^p} \leq z, \quad \pi \in \Pi_\sigma. \tag{4.8}$$

Proposition 24. *Inequalities* (4.8) *are valid for* H_σ.

Proof. Consider the extended formulation of H_σ given by

$$\widehat{H}_\sigma = \left\{ (x, y, z, s) \in \{0,1\}^n \times \mathbb{R}^{m+2}_+ : \sqrt[p]{s^p + \sum_{i \in M} d_i y_i^p} \leq z, \sqrt[p]{\sigma + \sum_{i \in N} c_i x_i} \leq s \right\}.$$

The validity of inequalities (4.8) for H_σ follows directly from the validity of the extended polymatroid inequality $\sqrt[p]{\sigma} + \pi' x \leq s$, $\pi \in \Pi_\sigma$ (Proposition 21) for $\widehat{H}_\sigma$. □

Remark 6. For $M = \emptyset$ inequalities (4.8) reduce to the extended polymatroid inequalities (4.5).

Remark 7. Although inequalities (4.8) are nonlinear in the original space of variables, they can be represented as linear inequalities in the extended formulation $\widehat{H}_\sigma$. Such a representation is desirable when they are used as cutting planes in branch-and-cut algorithms.

Remark 8. Since inequalities (4.8) correspond to extended polymatroid inequalities in an extended formulation, the separation for them is the same as in the binary case and can be done by sorting in $O(n \log n)$ (Proposition 22).

Proposition 25. *Inequalities* (4.8) *and the bound constraints describe* $conv(H_\sigma)$.

Proof. Consider the optimization of an arbitrary linear function over the convex relaxation of $\widehat{H}_\sigma$:

$$\begin{aligned} & \min \ -a'x - b'y + rz && (4.9)\\ \text{(P1)} \qquad & \text{s.t.} \ \sqrt[p]{s^p + \sum_{i\in M} d_i y_i^p} \le z && (4.10)\\ & \sqrt[p]{\sigma} + \pi' x \le s, \quad \forall \pi \in \Pi_\sigma && (4.11)\\ & x \in [0,1]^n, y \in \mathbb{R}^m_+, z \ge 0, s \ge 0. && (4.12) \end{aligned}$$

Note that the constraint $\sqrt[p]{\sigma + \sum_{i\in N} c_i x_i} \le s$ in $\widehat{H}_\sigma$ is implied by inequalities (4.11). We prove that for any linear objective (P1) is either unbounded or has an optimal solution that is integer in x.

Without loss of generality, we can assume that $r > 0$ (if $r < 0$ then the problem is unbounded, and if $r = 0$ then (P1) reduces to a linear program over an integral polyhedron), $r = 1$ (by scaling), $a_i, b_i > 0$ (otherwise $x_i = 0$ or $y_i = 0$ in any optimal solution), and $d_i = 1$ for all $i \in M$ (by scaling y_i). Eliminating the variable z from (P1) we rewrite the problem as

$$\begin{aligned} & \min \ -a'x - b'y + \sqrt[p]{s^p + \sum_{i\in M} y_i^p} \\ \text{(P2)} \qquad & \text{s.t.} \ \sqrt[p]{\sigma} + \pi' x \le s, \quad \forall \pi \in \Pi_\sigma \\ & x \in [0,1]^n, y \in \mathbb{R}^m_+, s \ge 0. \end{aligned}$$

Let $\mu \in \mathbb{R}^m_+$ be the dual variables for constraints $y \ge 0$. From the KKT conditions of (P2) with respect to y, we see that

$$-\mu_k = b_k - \left(s^p + \sum_{i\in M} y_i^p\right)^{\frac{1-p}{p}} y_k^{p-1}, \qquad \forall k \in M.$$

However, the complementary slackness conditions $y_k \mu_k = 0$ imply that $\mu_k = 0$ for all k, as otherwise $-\mu_k = b_k$ contradicts with the assumption that $b_k > 0$. Therefore, it holds that

$$y_k = \sqrt[p-1]{b_k} \cdot \sqrt[p]{s^p + \sum_{i\in M} y_i^p}, \quad \forall k \in M.$$

Defining $\beta = \sum_{i=1}^m b_i^{\frac{p}{p-1}}$, we have

$$\sum_{i\in M} b_i y_i = \beta \sqrt[p]{s^p + \sum_{i\in M} y_i^p}$$

and

$$\sum_{i\in M} y_i^p = \beta\left(s^p + \sum_{i\in M} y_i^p\right). \tag{4.13}$$

Observe that if $\beta > 1$, equality (4.13) cannot be satisfied, and the feasible (P2) is dual infeasible, therefore, unbounded. Moreover, if $\beta = 1$ then either the problem is unbounded or $s = 0$ in any optimal solution, which implies that $x = 0$ and all optimal solutions are integral in x. Finally, if $\beta < 1$, we deduce from (4.13) that

$$\sum_{i\in M} y_i^p = \frac{\beta}{1-\beta} s^p.$$

Replacing the summands in the objective, we rewrite (P2) as

$$\begin{array}{cc} & \min\ -a'x + (1-\beta)^{\frac{p-1}{p}}\, s \\ \text{(P3)} & \text{s.t. } \sqrt[p]{\sigma} + \pi' x \le s, \quad \forall \pi \in \Pi_\sigma \\ & x \in [0,1]^n, s \ge 0. \end{array}$$

As $\beta < 1$, (P3) has an optimal solution and, by Proposition 21, it is integral in x. □

4.4 Rotated cone constraints

In this section we focus on rotated cone constraints. Consider the mixed-integer set

$$\begin{aligned} R_\sigma &= \left\{(x,y,w,z) \in \{0,1\}^n \times \mathbb{R}^{m+2}_+ : \sigma + \sum_{i\in N} c_i x_i + \sum_{i\in M} d_i y_i^2 \le 4wz\right\} \\ &= \left\{(x,y,w,z) \in \{0,1\}^n \times \mathbb{R}^{m+2}_+ : \sqrt{\sigma + \sum_{i\in N} c_i x_i + \sum_{i\in M} d_i y_i^2 + (w-z)^2} \le w + z\right\}. \end{aligned}$$

Observe that, even when $M = \emptyset$, the set R_σ is defined by a second-order cone constraint with unbounded continuous variables. Thus, previous results were not applicable to R_σ. On the other hand, the inequalities discussed in Section 4.3 (with $p = 2$) can be used directly. In particular, inequalities (4.8) reduce to

$$\sqrt{\left(\sqrt{\sigma} + \pi' x\right)^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2} \le w + z, \quad \pi \in \Pi_\sigma. \tag{4.14}$$

We can also write inequalities (4.14) in rotated cone form,

$$(\sqrt{\sigma} + \pi' x)^2 + \sum_{i\in M} d_i y_i^2 \le 4wz, \quad \pi \in \Pi_\sigma.$$

Observe that the second-order cone constraint defining R_σ has additional structure that is not exploited by inequalities (4.14), namely the presence of the continuous variables w and z in both sides of the inequality. However, as Proposition 26 states, inequalities (4.14) are actually sufficient to describe the convex hull of R_σ.

Proposition 26. *Inequalities* (4.14) *and bound constraints completely describe the convex hull of* R_σ.

Proof. Consider the optimization of an arbitrary linear function over the convex relaxation of $\widehat{R}_\sigma$, using a similar extended formulation as in Proposition 24:

$$
\begin{aligned}
& \min\ a'x + b'y + pw + qz \\
(P_R) \qquad & \text{s.t.} \ \sqrt{s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2} \le w + z \\
& \sqrt{\sigma} + \pi' x \le s, \quad \forall \pi \in \Pi_\sigma \\
& x \in [0,1]^n, y \in \mathbb{R}^m_+, w \ge 0, z \ge 0, s \ge 0.
\end{aligned} \tag{4.15}
$$

Without loss of generality, we can assume that $p > 0$ and $q > 0$ (if $p < 0$ or $q < 0$ then the problem is unbounded, and if $p = 0$ or $q = 0$ then (P_R) reduces to a linear program over an integral polyhedron). Moreover observe that if $w = 0$ or $z = 0$ in an optimal solution, then $x = 0$ and the optimization problem has an integral optimal solution. Thus, we can assume that $z > 0$ and $q > 0$ and we infer from KKT conditions with respect to w and z that

$$
-p = -\lambda + \lambda \frac{w-z}{\sqrt{s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2}} \tag{4.16}
$$

$$
-q = -\lambda - \lambda \frac{w-z}{\sqrt{s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2}}, \tag{4.17}
$$

where λ is the dual variable associated with constraint (4.15). We deduce from (4.16) that $w - z = \frac{\lambda - p}{\lambda} \sqrt{s^2 + \sum\limits_{i\in M} d_i y_i^2 + (w-z)^2}$, and we deduce from (4.17) that

$$
w - z = \frac{q - \lambda}{\lambda} \sqrt{s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2}. \tag{4.18}
$$

Thus, we find that $\lambda = \frac{p+q}{2}$.

Moreover, we obtain from (4.18) that

$$\begin{aligned}(w-z)^2 &= \left(\frac{q-\lambda}{\lambda}\right)^2 \left(s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2\right) \\ &= \left(\frac{q-p}{q+p}\right)^2 \left(s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2\right) \\ &= \beta \left(s^2 + \sum_{i\in M} d_i y_i^2\right), \end{aligned} \tag{4.19}$$

where $\beta = \frac{\left(\frac{q-p}{q+p}\right)^2}{1-\left(\frac{q-p}{q+p}\right)^2}$. Therefore, we have that

$$\sqrt{s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2} = \sqrt{1+\beta}\sqrt{s^2 + \sum_{i\in M} d_i y_i^2}. \tag{4.20}$$

Moreover, since in any optimal solution of (P_R) constraint (4.15) is binding, we have that

$$w + z = \sqrt{1+\beta}\sqrt{s^2 + \sum_{i\in M} d_i y_i^2}. \tag{4.21}$$

Multiplying equality (4.16) by w in both sides, and multiplying equality (4.17) by z in both sides, we find that

$$\begin{aligned} pw + qz &= \lambda(w+z) - \lambda \frac{(w-z)^2}{\sqrt{s^2 + \sum_{i\in M} d_i y_i^2 + (w-z)^2}} \\ &= \lambda\sqrt{1+\beta}\sqrt{s^2 + \sum_{i\in M} d_i y_i^2} - \lambda \frac{\beta\left(s^2 + \sum_{i\in M} d_i y_i^2\right)}{\sqrt{1+\beta}\sqrt{s^2 + \sum_{i\in M} d_i y_i^2}} \quad \text{((4.19), (4.20),(4.21))} \\ &= \lambda \frac{s^2 + \sum_{i\in M} d_i y_i^2}{\sqrt{1+\beta}\sqrt{s^2 + \sum_{i\in M} d_i y_i^2}} \\ &= \frac{\lambda}{\sqrt{1+\beta}}\sqrt{s^2 + \sum_{i\in M} d_i y_i^2}. \end{aligned}$$

Therefore, we see that problem (P_R) reduces to

$$(P_R\text{'}) \qquad \begin{aligned} \min\ & a'x + b'y + \frac{p+q}{2\sqrt{1+\beta}}\sqrt{s^2 + \sum_{i\in M} d_i y_i^2} \\ \text{s.t. } & \sqrt{\sigma} + \pi' x \le s, \quad \forall \pi \in \Pi_\sigma \\ & x \in [0,1]^n, y \in \mathbb{R}^m_+, s \ge 0, \end{aligned}$$

which has an integral optimal solution (Proposition 25). □

4.5 Conic constraint with bounded continuous variables

In this section we study the set L_σ, the generalization of K_σ with upper bounded continuous variables. As Example 1 illustrates, $\text{conv}(L_\sigma)$ is significantly more difficult to describe than $\text{conv}(H_\sigma)$.

Example 1. Consider the three-dimensional set given by

$$L^2_\sigma = \left\{(x, y, z) \in \{0, 1\} \times [0, 1] \times \mathbb{R}_+ : \sqrt{\sigma + cx + dy^2} \leq z\right\}.$$

We show in Appendix A.3 that

$$\text{conv}(L^2_\sigma) = \left\{(x, y, z) \in [0, 1] \times [0, 1] \times \mathbb{R}_+ : g(x, y) \leq z\right\},$$

where

$$g(x,y) = \begin{cases} g_1(x,y) = \sqrt{\left(\sqrt{\sigma} + x(\sqrt{c+\sigma} - \sqrt{\sigma})\right)^2 + dy^2} & \text{if } y \leq x + (1-x)\sqrt{\frac{\sigma}{\sigma+c}} \\ g_2(x,y) = \sqrt{\sigma(1-x)^2 + d(y-x)^2} + x\sqrt{\sigma + c + d} & \text{otherwise.} \end{cases}$$

Observe that the inequality $g_1(x, y) \leq z$ is a particular case of (4.8). The difficulties arise with the function g_2:

(a) The discrete and continuous variables are tied together in the term $\sqrt{\sigma(1-x)^2 + d(y-x)^2}$.

(b) The inequality $g_2(x, y) \leq z$ is not valid. In particular, it cuts off the feasible point $(x, y, z) = (1, 0, \sqrt{\sigma + c})$. Moreover, the inequality $g_2(x, y) \leq z$ cuts off portions of $\text{conv}(L^2_\sigma)$ whenever $y \leq x + (1-x)\frac{\sqrt{\sigma}}{\sqrt{\sigma+c}}$.

(c) The condition $y \leq x + (1-x)\frac{\sqrt{\sigma}}{\sqrt{\sigma+c}}$ depends both on x and y.

Figure 4.1 shows functions g_1 and g_2 for a fixed value of x, and illustrates point (b) above. We see that the function g_2 is always "above" the function g_1, and cuts the convex hull of L^2_σ (the shaded region) whenever $y \leq x + (1-x)\frac{\sqrt{\sigma}}{\sqrt{\sigma+c}}$.

We now give valid inequalities for $\text{conv}(L_\sigma)$. For $T \subseteq M$, consider the inequalities

$$\sqrt[p]{\left(\sqrt[p]{\sigma + \sum_{i\in T} d_i y_i^p} + \pi' x\right)^p + \sum_{i \in M\setminus T} d_i y_i^p} \leq z, \quad \pi \in \Pi_{\sigma+d(T)}. \tag{4.22}$$

Proposition 27. *Inequalities* (4.22) *are valid for* L_σ.

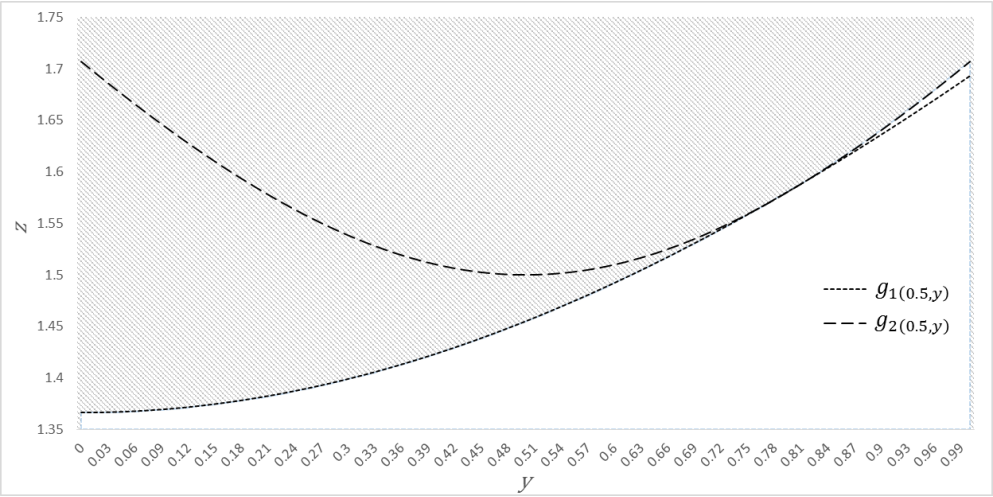


Figure 4.1: Funcs. g_1, g_2 with $\sigma = d = 1$, $c = 2$, restricted to $x = 0.5$.

Proof. For $T \subseteq M$, let

$$L_\sigma(T) = \left\{ (x,y) \in \{0,1\}^n \times [0,1]^m, s \geq 0 : \sqrt[p]{\sigma + \sum_{i \in N} c_i x_i + \sum_{i \in T} d_i y_i^2} \leq s \right\},$$

and consider the extended formulation of L_σ given by

$$\widehat{L}_\sigma = \left\{ (x,y,s) \in L_\sigma(T), z \geq 0 : \sqrt[p]{s^p + \sum_{i \in M \setminus T} d_i y_i^p} \leq z \right\}.$$

The validity of inequalities (4.22) for L_σ follows from the validity of

$$\sqrt[p]{\sigma + \sum_{i \in T} d_i y_i^p} + \pi' x \leq s, \quad \pi \in \Pi_{\sigma + d(T)} \tag{4.23}$$

for $L_\sigma(T)$ (Proposition 27). □

Remark 9. If $T = \emptyset$, then inequalities (4.22) coincide with inequalities (4.8). If $T = M$, then inequalities (4.22) coincide with inequalities (4.6). If $T \subset M$, then inequalities (4.22) dominate inequalities (4.6).

Remark 10. Inequalities (4.22) are convex, since they correspond to the projection of convex inequalities (4.23) in an extended formulation.

Example 2. Consider the set

$$L_0^6 = \left\{ (x, y, z) \in \{0,1\}^4 \times [0,1]^2 \times \mathbb{R}_+ : \sqrt{\sum_{i=1}^{4} x_i + y_1^2 + y_2^2} \leq z \right\}.$$

For the permutation (1,2,3,4) inequalities (4.22) are

$$\begin{array}{ll} T = \emptyset: & \sqrt{(x_1 + 0.41x_2 + 0.32x_3 + 0.27x_4)^2 + y_1^2 + y_2^2} \leq z, \\ T = \{1\}: & \sqrt{(0.41x_1 + 0.32x_2 + 0.27x_3 + 0.24x_4 + y_1)^2 + y_2^2} \leq z, \\ T = \{1,2\}: & 0.32x_1 + 0.27x_2 + 0.24x_3 + 0.21x_4 + \sqrt{y_1^2 + y_2^2} \leq z. \end{array}$$

Observe that for $T = \emptyset$ and $T = \{1\}$, the resulting inequalities dominate the corresponding inequalities obtained from (4.6), given by $x_1 + 0.41x_2 + 0.32x_3 + 0.27x_4 \leq z$ and $0.41x_1 + 0.32x_2 + 0.27x_3 + 0.24x_4 + y_1 \leq z$, respectively.

Example 1 (Continued). We obtain from (4.22) the valid inequality

$$g_3(x, y) = \sqrt{\sigma + dy^2} + x\left(\sqrt{\sigma + c + d} - \sqrt{\sigma + d}\right) \leq z$$

for L_σ^2. Observe that if $\sigma = 0$, then $g_1(x, y) \leq z$, $g_3(x, y) \leq z$ and the bound constraints give a complete description of $\text{conv}(L_\sigma^2)$ since

$$g_3(x, y) = \sqrt{d}y + x\left(\sqrt{c + d} - \sqrt{d}\right) = \sqrt{d}\,(|y - x|) + x\sqrt{\sigma + c + d} = g_2(x, y)$$

whenever $y \geq x + (1 - x)\sqrt{\frac{\sigma}{\sigma+c}} = x$. If $\sigma > 0$, then $g_3(x, y) \leq z$ is valid and provides an approximation of $\text{conv}(L_\sigma^2)$ (Figure 4.2).

4.6 Strengthened polymatroid inequalities

The polymatroid inequalities of Sections 4.3, 4.4 and 4.5 use the conic constraint and the bounds of the variables. In this section we show how to strengthen the polymatroid inequalities using additional constraints. In particular, given *any* mixed-integer set $X \subseteq \{0,1\}^n \times \mathbb{R}_+^m$, we consider the generalization

$$G_\sigma = \left\{ (x, y) \in X, z \geq 0 : \sqrt[p]{\sigma + \sum_{i \in N} c_i x_i + \sum_{i \in M} d_i y_i^p} \leq z \right\}.$$

First, in Section 4.6 we describe a lifting procedure for obtaining valid inequalities for G_σ, where computing each coefficient requires solving an integer optimization problem. Then, in Section 4.6 we discuss how the strengthened polymatroid inequalities can be efficiently implemented in practice.

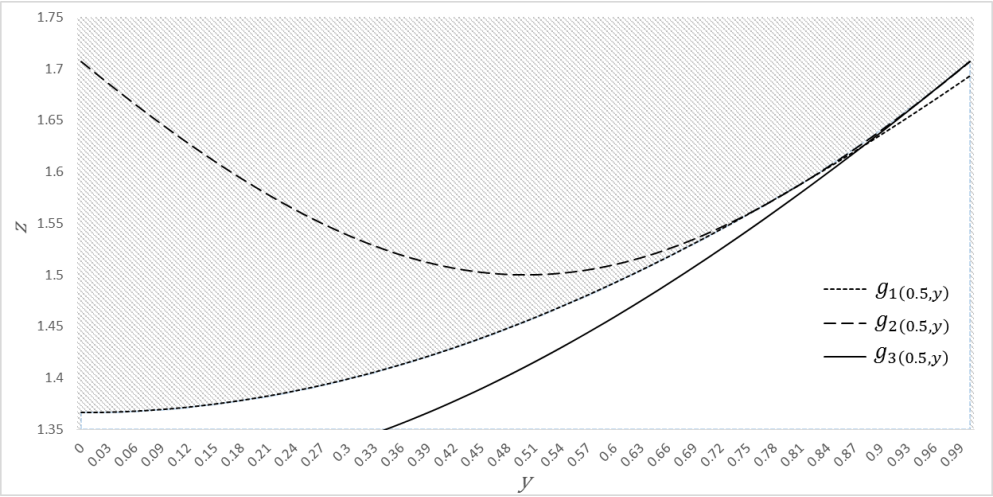


Figure 4.2: Functions g_1, g_2, g_3 with $\sigma = d = 1$, $c = 2$, restricted to $x = 0.5$.

Valid inequalities for G_σ

For a given a permutation $((1), (2), \ldots, (n))$ of N and $T \subseteq M$, let

$$h_k(x, y) = \sigma + \sum_{i=1}^{k-1} c_{(i)} x_{(i)} + \sum_{i \in T} d_i y_i^p$$

$$\bar{\sigma}_{(k)} = \max \left\{ h_k(x, y) : (x, y) \in X, x_k = 1 \right\}, \text{ and} \tag{4.24}$$

$$\rho_{(k)} = \begin{cases} \sqrt[p]{c_{(k)} + \bar{\sigma}_{(k)}} - \sqrt[p]{\bar{\sigma}_{(k)}} & \text{if } \bar{\sigma}_{(k)} < \infty \\ 0 & \text{otherwise.} \end{cases} \tag{4.25}$$

Consider the inequality

$$\sqrt[p]{\left(\sqrt[p]{\sigma + \sum_{i \in T} d_i y_i^p} + \sum_{i=1}^{n} \rho_{(i)} x_{(i)} \right)^p + \sum_{i \in M \setminus T} d_i y_i^p} \leq z. \tag{4.26}$$

Proposition 28. *Inequalities* (4.26) *are valid for* G_σ.

Proof. Let

$$G_\sigma(T) = \left\{ (x, y) \in X, s \geq 0 : \sqrt[p]{\sigma + \sum_{i \in N} c_i x_i + \sum_{i \in T} d_i y_i^p} \leq s \right\},$$

and consider the extended formulation of G_σ given by

$$\hat{G_\sigma} = \left\{ (x, y, s) \in G_\sigma(T), z \geq 0 : \sqrt[p]{s^p + \sum_{i \in M \setminus T} d_i y_i^p} \leq z \right\}.$$

To prove the validity of (4.26) for G_σ, it is sufficient to show that

$$\sqrt[p]{\sigma + \sum_{i \in T} d_i y_i^p} + \sum_{i=1}^{n} \rho_{(i)} x_{(i)} \leq s \tag{4.27}$$

is valid for $G_\sigma(T)$. In particular, we prove by induction that

$$\sqrt[p]{\sigma + \sum_{i \in T} d_i y_i^p} + \sum_{i=1}^{k} \rho_{(i)} x_{(i)} \leq \sqrt[p]{\sigma + \sum_{i=1}^{k} c_{(i)} x_{(i)} + \sum_{i \in T} d_i y_i^p} \tag{4.28}$$

for all $(x, y) \in X$ and $k = 0, \ldots, n$.

Base case: $k = 0$ Inequality (4.28) holds trivially.

Inductive step Let $(\bar{x}, \bar{y}) \in X$, and suppose inequality (4.28) holds for $k-1$. Observe that if $\bar{x}_{(k)} = 0$ or $\rho_{(k)} = 0$, then inequality (4.28) clearly holds for k. Therefore, assume that $\bar{x}_{(k)} = 1$ and $\bar{\sigma}_{(k)} < \infty$. We have

$$\begin{aligned}
\sqrt[p]{\sigma + \sum_{i=1}^{k} c_{(i)} \bar{x}_{(i)} + \sum_{i \in T} d_i \bar{y}_i^p} &= \sqrt[p]{h_k(\bar{x}, \bar{y}) + c_{(k)}} \\
&= \sqrt[p]{h_k(\bar{x}, \bar{y})} + \left(\sqrt[p]{h_k(\bar{x}, \bar{y}) + c_{(k)}} - \sqrt[p]{h_k(\bar{x}, \bar{y})} \right) \\
&\geq \sqrt[p]{h_k(\bar{x}, \bar{y})} + \left(\sqrt[p]{\bar{\sigma}_{(k)} + c_{(k)}} - \sqrt[p]{\bar{\sigma}_{(k)}} \right) \qquad (4.29) \\
&\geq \sqrt[p]{\sigma + \sum_{i \in T} d_i \bar{y}_i^p} + \sum_{i=1}^{k} \rho_{(i)} \bar{x}_{(i)}, \qquad (4.30)
\end{aligned}$$

where (4.29) follows from $\bar{\sigma}_{(k)} \geq h_k(\bar{x}, \bar{y})$ (by definition of $\bar{\sigma}_{(k)}$) and from the concavity of the root function, and (4.30) follows from $\sqrt[p]{h_k(\bar{x}, \bar{y})} \geq \sqrt[p]{\sigma + \sum_{i \in T} d_i \bar{y}_i^p} + \sum_{i=1}^{k-1} \rho_{(i)} \bar{x}_{(i)}$ (induction hypothesis) and from the definition of $\rho_{(k)}$. □

Example 2 (Continued). Let $X^6 = \left\{ (x, y) \in \{0,1\}^4 \times [0,1]^2 : \sum_{i=1}^{4} x_i + y_1 + y_2 \leq 3 \right\}$ and consider the set $G_0^6 = L_0^6 \cap X^6$. For the permutation (1,2,3,4) inequalities (4.26) are

$$
\begin{aligned}
T = \emptyset: &\quad \sqrt{(x_1 + 0.41x_2 + 0.32x_3 + 0.32x_4)^2 + y_1^2 + y_2^2} \leq z, \\
T = \{1\}: &\quad \sqrt{(0.41x_1 + 0.32x_2 + 0.32x_3 + 0.32x_4 + y_1)^2 + y_2^2} \leq z, \\
T = \{1,2\}: &\quad 0.32x_1 + 0.32x_2 + 0.32x_3 + 0.32x_4 + \sqrt{y_1^2 + y_2^2} \leq z.
\end{aligned}
$$

Observe that, in all cases, the resulting inequalities dominate the corresponding inequalities obtained from (4.22).

Remark 11. If $T = \emptyset$ and $X = \{0,1\}^n \times \mathbb{R}^m_+$, then inequalities (4.26) reduce to inequalities (4.8). If $T = \emptyset$ and $X \subset \{0,1\}^n \times \mathbb{R}^m_+$, then inequalities (4.26) dominate inequalities (4.8).

Remark 12. If $X = \{0,1\}^n \times [0,1]^m$, then inequalities (4.26) reduce to inequalities (4.22). If $X \subset \{0,1\}^n \times [0,1]^m$, then inequalities (4.26) dominate inequalities (4.22).

Remark 13. For the case of the pure-binary set defined by a cardinality constraint, i.e., $X = \{x \in \{0,1\}^n : \sum_{i=1}^n x_i \leq k\}$, inequalities (4.26) coincide with the inequalities proposed in Yu and Ahmed (2015).

Computational efficiency

Note that computing each coefficient of inequality (4.26) requires solving the integer optimization problem (4.24), which may not be practical in most cases. However, observe from Remarks 11 and 12 that solving the optimization problem over *any* relaxation of X results in valid inequalities at least as strong as the ones resulting from using only the bounds constraints.

In particular, assume in problem (4.24) that for $i \in T$ there exists $u_i \geq 0$ such that $y_i \leq u_i$ (otherwise the problem is unbounded and $\rho_i = 0$) and $u_i = 1$ (by scaling). Moreover let X_P be a polytope such that $X \subseteq X_P$. Convex constraints can also be included in X_P by using a suitable linear outer approximation (Ben-Tal and Nemirovski 2001b, Tawarmalani and Sahinidis 2005, Hijazi et al. 2013, Vielma et al. 2016, Lubin et al. 2016).

Given X_P, the approximate coefficients

$$
\begin{aligned}
&\hat{\rho}_{(k)} = \sqrt[p]{c_{(k)} + \hat{\sigma}_{(k)}} - \sqrt[p]{\hat{\sigma}_{(k)}}, \text{ with} \qquad (4.31) \\
&\hat{\sigma}_{(k)} = \sigma + \max \left\{ \sum_{i=1}^{k-1} c_{(i)} x_{(i)} + \sum_{i \in T} d_i y_i : (x,y) \in X_P, x_k = 1 \right\}
\end{aligned}
$$

can be computed efficiently by solving a linear program. Moreover, the linear program required to compute $\hat{\sigma}_{(k)}$ differs from the one required for $\hat{\sigma}_{(k-1)}$ in two bound constraints, corresponding to $x_{(k-1)}$ and $x_{(k)}$, and one objective coefficient, corresponding to $x_{(k-1)}$. Therefore, using the simplex method with warm starts, each $\hat{\sigma}_{(k)}$ can be computed efficiently, using only a small number of simplex pivots.

4.7 Computational experiments

In this section we report computational experiments performed to test the effectiveness of the polymatroid inequalities in solving MISOCO problems with a branch-and-cut algorithm. All experiments are done using CPLEX 12.6.2 solver on a workstation with a 2.93GHz Intel®Core™ i7 CPU and 8 GB main memory and with a single thread. The time limit is set to two hours and CPLEX' default settings are used unless specified otherwise. The inequalities are added only at the root node using callback functions.

Instances with bounded continuous variables

In this section we test the effectiveness of the polymatroid inequalities (4.8) and (4.22) in solving optimization problems of the form

$$\min\{-a'x - b'y + \Omega z : (x, y, z) \in L_\sigma\} \tag{4.32}$$

with $\sigma = 0$ and compare them with default CPLEX with no user cuts. For two numbers $\ell < u$, let $U[\ell, u]$ denote the continuous uniform distribution between ℓ and u. The data for the model is generated as follows: $a_i \sim U[0, 1]$, $\sqrt{c_i} \sim U[0.85a_i, 1.15a_i]$ for $i \in N$, $b_j \sim U[0, 1]$, $\sqrt{d_j} \sim U[0.85b_j, 1.15b_j]$ for $j \in M$, and Ω is the solution[1] of

$$-a(N) - b(M) + \Omega\sqrt{c(N) + d(M)} = 0.$$

Inequalities (4.8) are added as linear cuts in an extended formulation, as described in Remark 7. For $p = 2$, inequalities (4.22) are of the form $f(x, y) \leq z$, where

$$f(x, y) = \sqrt{\left(\sqrt{\sigma + \sum_{i \in T} d_i y_i^2} + \pi' x\right)^2 + \sum_{i \in M \setminus T} d_i y_i^2}.$$

As only linear inequalities can be added through callbacks in CPLEX (as of version 12.6.2), we utilize the gradient inequalities for (4.22). Thus, given a fractional solution $(\bar{x}, \bar{y})$, we add the linear underestimator $g(x, y) \leq z$, where

$$g(x, y) = f(\bar{x}, \bar{y}) + \nabla_x f(\bar{x})'(x - \bar{x}) + \nabla_y f(\bar{y})'(y - \bar{y}).$$

[1]This choice of Ω ensures that the linear and nonlinear components are well-balanced, resulting in challenging instances with large integrality gap.

In particular, we have that

$$g(x,y) = \psi + \frac{1}{\psi}\left(\eta\pi'(x-\bar{x}) + \zeta\sum_{i\in T} d_i\bar{y}_i(y_i-\bar{y}_i) + \sum_{i\in M\setminus T} d_i\bar{y}_i(y_i-\bar{y}_i)\right),$$

where

$$\begin{aligned}\eta &= \sqrt{\sigma + \sum_{i\in T} d_i\bar{y}_i^2} + \pi'\bar{x},\\ \zeta &= \frac{\eta}{\sqrt{\sigma + \sum_{i\in T} d_i\bar{y}_i^2}},\\ \psi &= \sqrt{\eta^2 + \sum_{i\in M\setminus T} d_i\bar{y}_i^2}.\end{aligned}$$

A greedy heuristic is used to choose $T \subseteq M$ for inequalities (4.22): if $\bar{y}$ satisfies $\bar{y}_{(1)} \geq \bar{y}_{(2)} \geq \ldots \geq \bar{y}_{(m)}$, then we check for violation inequalities for each T_i of the form $T_i = \{(1),(2),\ldots,(i)\}$ for $i = 0,\ldots,m$. When adding the gradient inequalities corresponding to (4.22), CPLEX' barrier algorithm is found to be more effective than using the default setting to solve the subproblems of the branch-and-bound tree. Therefore, we report the results for inequalities (4.22) with the barrier algorithm.

Table 4.1 presents the results for $n = 100$. Each row represents the average over five instances generated with the same parameters and shows the number of continuous variables (m), the initial gap (`igap`), the root gap improvement (`rimp`), the number of nodes explored (`nodes`), the time elapsed in seconds (`time`), and the end gap (`egap`)[in brackets, the number of instances solved to optimality (#)]. The initial gap is computed as $\texttt{igap} = \frac{t_{\text{opt}} - t_{\text{relax}}}{|t_{\text{opt}}|} \times 100$, where t_{opt} is the objective value of the best feasible solution at termination and t_{relax} is the objective value of the continuous relaxation. The end gap is computed as $\texttt{egap} = \frac{t_{\text{opt}} - t_{\text{bb}}}{|t_{\text{opt}}|} \times 100$, where t_{bb} is the objective value of the best lower bound at termination. The root improvement is computed as $\texttt{rimp} = \frac{t_{\text{root}} - t_{\text{relax}}}{t_{\text{opt}} - t_{\text{relax}}} \times 100$, where t_{root} is the value of the continuous relaxation after adding the valid inequalities to the formulation.

We observe in Table 4.1 that the use inequalities (4.8), which do not exploit the upper bounds of the continuous variables, close 80.0% of the initial gap on average, but the gap improvement does not translate to better solution times or end gaps. On the other hand, inequalities (4.22), which exploit the upper bounds of the continuous variables, close 99% of the initial gap on average. This improves the performance of the algorithm substantially, reducing the average solution time by half and the end gap from from 5.4% to 0.7%.

Table 4.1: Experiments with bounded continuous variables.

m	igap	cpx				inequality (4.8)				inequality (4.22) (barrier)			
		rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]
20	1,554.7	0.0	283,747	420	0.0[5]	90.4	19,976	628	0.0[5]	99.5	316	25	0.0[5]
50	724.6	0.0	1,887,926	2,223	0.0[5]	79.4	1,206,283	5,770	65.4[1]	98.8	1,635	857	0.0[5]
100	267.8	0.0	982,945	5,343	16.1[2]	70.1	615,494	7,200	54.6[0]	98.7	1,506	2,959	2.0[3]
Average		**0.0**	**1,051,539**	**2,662**	**5.4[12]**	**80.0**	**613,918**	**4,533**	**40.0[6]**	**99.0**	**1,152**	**1,280**	**0.7[13]**

Instances with a cardinality constraint

In this section we test the value of strengthening the polymatroid inequalities utilizing additional problem constraints. To do so, we solve optimization problems with a cardinality constraint:

$$\min_{x\in\{0,1\}^n} \left\{ -a'x + \Omega\sqrt{c'x} : \sum_{i=1}^{n} x_i \leq k \right\}, \tag{4.33}$$

where a and c are generated as in Section 4.7 and $\Omega = \Phi^{-1}(\alpha)$, where Φ is the cumulative distribution function of the normal distribution and $\alpha \in \{0.95, 0.975, 0.99\}$. We set $n = 200$, and set k to be 15%, 20% and 25% of the total number of variables. Inequalities (4.5) and (4.26) are compared with default CPLEX. The coefficients of inequalities (4.26) are computed using linear programming with warm starts as outlined in Section 4.6—observe that, in this case, the coefficients (4.31) coincide with (4.25) since the feasible region is an integral polytope.

Table 4.2: Experiments with cardinality constraints.

k	α	igap	cpx				inequality (4.5)				inequality (4.26)			
			rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]
30	0.95	4.4	23.7	7,150,715	2,528	0.3[4]	36.6	3,754,826	2,073	0.4[4]	48.9	2,614,446	1,510	0.2[4]
	0.975	7.2	7.2	13,632,197	6,120	1.8[1]	23.7	9,573,199	5,945	1.8[1]	39.8	9,235,158	5,797	1.0[1]
	0.99	11.9	4.0	16,867,459	7,200	5.0[0]	14.7	10,899,169	7,200	5.7[0]	31.6	13,328,370	7,200	4.1[0]
Average			**11.6**	**12,550,124**	**5,283**	**2.4[5]**	**25.0**	**8,075,731**	**5,073**	**2.6[5]**	**40.1**	**8,392,658**	**4,836**	**1.8[5]**
40	0.95	1.9	20.7	6,235,270	1,674	0.1[4]	70.5	620,389	261	0.0[5]	75.0	90,179	62	0.0[5]
	0.975	3.3	9.6	13,961,488	4,360	0.4[3]	49.2	3,268,824	2,122	0.2[4]	57.3	2,729,459	1,557	0.2[4]
	0.99	5.6	6.0	15,334,782	6,738	1.8[1]	30.0	6,110,571	6,149	1.7[1]	42.6	5,222,829	5,799	1.2[1]
Average			**12.1**	**11,843,847**	**4,257**	**0.8[8]**	**49.9**	**3,333,261**	**2,844**	**0.6[10]**	**58.3**	**2,680,821**	**2,472**	**0.5[10]**
50	0.95	1.0	8.9	270,852	72	0.0[5]	93.3	249	2	0.0[5]	93.3	98	2	0.0[5]
	0.975	1.6	8.0	3,882,494	1,045	0.0[5]	81.3	316,625	221	0.0[5]	84.4	198,916	92	0.0[5]
	0.99	2.8	7.9	14,835,539	4,600	0.3[3]	57.3	4,695,268	3,480	0.2[3]	64.3	983,894	1,537	0.2[4]
Average			**8.3**	**6,329,628**	**1,906**	**0.1[13]**	**77.3**	**1,670,714**	**1,234**	**0.1[13]**	**80.7**	**394,293**	**544**	**0.1[14]**

Table 4.2 presents the results for each value of k and α. We see that for instances with $k = 50$, using inequalities (4.5) or (4.26) results in gap improvement of more than 75% and faster solutions times than default CPLEX. In particular, using inequalities (4.26) results in solutions times that are four times faster than default CPLEX on average. As expected, for instances with tighter cardinality constraints, inequalities (4.26), which exploit the cardinality constraints, are more effective than inequalities (4.5) in reducing the solution times as well as end gaps. On the other hand, when the cardinality constraint is loose, the effectiveness of both classes of inequalities improve.

Instances with non-diagonal quadratic term and cardinality constraint

Although the inequalities in this chapter are developed for the diagonal case of the conic inequalities (4.1), they can, nevertheless, be used for the general non-diagonal case as well through a relaxation. Consider an optimization problem of the form

$$\min_{x\in\{0,1\}^n} \left\{ -a'x + \Omega\sqrt{x'Qx} : \sum_{i=1}^{n} x_i \leq k \right\}, \tag{4.34}$$

with $Q = D + Q_0$, where $Q_0 \succeq 0$, $D \succeq 0$ and D is diagonal. Given a general matrix $Q \succeq 0$, matrices Q_0 and D can be computed using the smallest eigenvalue (Frangioni and Gentile 2006) or solving an SDP (Frangioni and Gentile 2007). Alternatively, in many large-scale instances Q is a covariance matrix built through a *factor model*, in which case D is the diagonal matrix with the specific variances, $Q_0 = XFX'$, where $X \in \mathbb{R}^{n\times r}$ is the exposure matrix and $F \in \mathbb{R}^{r\times r}$ is the factor covariance matrix. Either way, given Q_0 and D, problem (4.34) can be reformulated as

$$\min_{(x,y)\in\{0,1\}^n\times\mathbb{R}_+} \left\{ -a'x + \Omega\sqrt{\sum_{i=1}^{n} D_{ii}x_i + y^2} : \sum_{i=1}^{n} x_i \leq k, \sqrt{x'Q_0x} \leq y \right\},$$

and the polymatroid inequalities can be applied to the diagonal objective.

In the computational experiments we generate the data using a factor model. Let $F = GG'$, with $G \in \mathbb{R}^{r\times r}$ and $G_{ij} \sim U[-1, 1]$, $X_{ij} \sim U[0, 1]$ with probability 0.2 and $X_{ij} = 0$ otherwise, $D_{ii} \sim U[0, \delta\bar{q}]$, where $\delta \geq 0$ is a diagonal dominance parameter and $\bar{q} = \frac{1}{N}\sum_{i\in N} Q_{0ii}$, and $a_i \sim U[0.85\sqrt{Q_{ii}}, 1.15\sqrt{Q_{ii}}]$. The parameter Ω is set as in Section 4.7. We let $n = 200$, $r = 40$ and k equal to 10%, 15%, and 20% of the number of the variables.

The effectiveness of inequalities (4.8) and (4.26) are compared with default CPLEX. The inequalities are added using an extended formulation as described in Remark 7.

Table 4.3: Experiments with the non-diagonal case ($\delta = 0.5$).

k	α	igap	cpx				inequality (4.8)				inequality (4.26)			
			rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]
20	0.95	1.7	22.6	9,557	74	0.0[5]	53.3	3,957	23	0.0[5]	55.6	2,367	17	0.0[5]
	0.975	3.0	21.3	33,468	242	0.0[5]	53.5	13,316	86	0.0[5]	55.9	5,839	40	0.0[5]
	0.99	5.2	15.2	164,568	1,845	0.0[5]	52.8	80,735	730	0.0[5]	55.3	23,577	269	0.0[5]
Average			**19.7**	**69,198**	**720**	**0.0[15]**	**53.2**	**32,669**	**280**	**0.0[15]**	**55.6**	**10,594**	**109**	**0.0[15]**
30	0.95	0.8	15.5	7,115	57	0.0[5]	53.3	1,656	11	0.0[5]	52.4	1,159	9	0.0[5]
	0.975	1.3	14.9	18,901	135	0.0[5]	53.1	2,800	20	0.0[5]	54.0	2,095	15	0.0[5]
	0.99	2.3	5.7	76,675	1,005	0.0[5]	61.1	8,265	48	0.0[5]	62.1	5,131	30	0.0[5]
Average			**12.0**	**34,230**	**399**	**0.0[15]**	**55.8**	**4,240**	**26**	**0.0[15]**	**56.2**	**2,795**	**18**	**0.0[15]**
40	0.95	0.4	23.3	2,910	18	0.0[5]	48.5	611	6	0.0[5]	50.5	577	6	0.0[5]
	0.975	0.7	20.0	4,216	30	0.0[5]	54.3	884	7	0.0[5]	55.5	839	7	0.0[5]
	0.99	1.1	13.5	46,030	514	0.0[5]	55.9	2,493	18	0.0[5]	56.7	2,144	14	0.0[5]
Average			**18.9**	**17,719**	**187**	**0.0[15]**	**52.9**	**1,329**	**10**	**0.0[15]**	**54.2**	**1,187**	**9**	**0.0[15]**

Table 4.4: Experiments with the non-diagonal case ($\delta = 1.0$).

k	α	igap	cpx				inequality (4.8)				inequality (4.26)			
			rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]	rimp	nodes	time	egap[#]
20	0.95	2.9	21.6	64,283	927	0.0[5]	55.1	14,984	165	0.0[5]	59.1	6,233	68	0.0[5]
	0.975	5.0	15.5	240,224	3,975	0.4[3]	44.4	189,826	3,390	0.4[3]	50.9	102,053	1,915	0.1[4]
	0.99	9.0	6.4	378,116	7,200	2.2[0]	35.7	477,553	7,200	1.9[0]	43.1	430,707	5,966	0.6[2]
Average			**14.5**	**227,541**	**4,034**	**0.9[8]**	**45.1**	**227,454**	**3,585**	**0.8[8]**	**51.0**	**179,664**	**2,650**	**0.2[11]**
30	0.95	1.1	17.1	32,629	316	0.0[5]	77.2	1,082	12	0.0[5]	78.2	682	10	0.0[5]
	0.975	2.0	12.5	150,756	2,046	0.1[4]	72.9	12,202	107	0.0[5]	75.5	4,896	39	0.0[5]
	0.99	3.5	10.5	258,866	3,679	0.5[3]	67.8	115,507	1,510	0.1[4]	70.6	59,106	511	0.0[5]
Average			**13.4**	**147,417**	**2,014**	**0.2[12]**	**72.6**	**42,930**	**543**	**0.0[14]**	**74.8**	**21,561**	**187**	**0.0[15]**
40	0.95	0.6	23.9	6,522	64	0.0[5]	72.3	270	9	0.0[5]	74.8	192	8	0.0[5]
	0.975	1.0	24.0	31,022	414	0.0[5]	71.0	823	12	0.0[5]	72.1	695	11	0.0[5]
	0.99	1.6	17.6	122,568	2,907	0.2[3]	73.9	4,416	37	0.0[5]	75.1	2,543	26	0.0[5]
Average			**21.8**	**53,371**	**1,128**	**0.1[13]**	**72.4**	**1,836**	**19**	**0.0[15]**	**74.0**	**1,143**	**15**	**0.0[15]**

Tables 4.3 and 4.4 present the results for different choices of the diagonal dominance parameter δ[2]. Observe that adding inequalities (4.8) or (4.26) closes the initial gaps by 45% to 75%, resulting in significant performance improvement over default CPLEX. In particular, using inequalities (4.26) for instances with $k = 20$ leads to seven times speed-up with $\delta = 0.5$ and two times speed-up with $\delta = 1$) and lower end gaps. Moreover, for instances with

[2]Intuitively, if $\delta = 0.5$ then the factors explain 80% of the variance in the problem; if $\delta = 1.0$, then the factors explain 66% of the variance in the problem.

$k \geq 30$ using inequalities (4.26) results in at least an order-of-magnitude speed-up over default CPLEX. As in the previous section, inequalities (4.26), exploiting the cardinality constraint, are more effective than (4.8). The impact of both inequalities increases with higher diagonal dominance.

Binary fractional programming instances

Consider the binary fractional linear program

$$\max_{x\in\{0,1\}^n} \left\{ \sum_{i=1}^{m} \gamma_i \frac{\sum_{j=1}^{n} \rho_{ij}\nu_{ij}x_j}{\nu_{i0} + \sum_{j=1}^{n} \nu_{ij}x_j} : \sum_{j=1}^{n} x_j \leq k \right\}. \tag{4.35}$$

Problem (4.35) arises for example in assortment optimization under the mixed multinomial logit model. In this setting n is the number of products being offered and m is the number of customer classes, where each class is assumed to choose a product according to a multinomial logit model with parameters specific to the class. Parameter ρ_{ij} is revenue obtained by selling product j to customer class i, ν_{ij} is the preference associated for a customer of class i of buying product j, ν_{i0} is the no-purchase preference for a customer of class i, γ_i represents the proportion of customers that belong to class i, and k is the maximum number of products that can be offered. The goal is to maximize the total product obtained from selling products.

Following the work of Sen et al. (2015), problem (4.35) can be transformed into an equivalent minimization problem. Let $\bar{\rho}_i = \max_{j=1,\ldots,n} \rho_{ij}$, and observe that we can write (4.35) as

$$\begin{aligned} \sum_{i=1}^{m} \gamma_i \bar{\rho}_i - \min \; & \sum_{i\in M} \gamma_i \frac{\nu_{i0}\bar{p}_i + \sum_{j\in N} \nu_{ij}\left(\bar{p}_i - p_{ij}\right) x_j}{\nu_{i0} + \sum_{j\in N} \nu_{ij}x_j} \\ \text{s.t.} \; & \sum_{j\in N} x_j \leq k \\ & x \in \{0,1\}^N. \end{aligned}$$

We now propose a MISOCO formulation different from the one proposed in Sen et al.

(2015), allowing us to use inequalities (4.14). Introduce new variables

$$\begin{aligned} z_i &:= \frac{\nu_{i0}\bar{p}_i + \sum_{j\in N} \nu_{ij}\left(\bar{p}_i - p_{ij}\right) x_j}{\nu_{i0} + \sum_{j\in N} \nu_{ij} x_j} \\ s_i^2 &:= \nu_{i0}\bar{p}_i + \sum_{j\in N} \nu_{ij}\left(\bar{p}_i - p_{ij}\right) x_j \\ w_i &:= \nu_{i0} + \sum_{j\in N} \nu_{ij} x_j, \end{aligned}$$

and reformulate the problem as

$$\begin{aligned} \min\ & \sum_{i\in M} \gamma_i z_i \\ \text{s.t.}\ & \sum_{j\in N} x_j \le k \\ & s_i^2 \le w_i z_i && \forall i \in M \\ & \sqrt{\nu_{i0}\bar{p}_i + \sum_{j\in N} \nu_{ij}\left(\bar{p}_i - p_{ij}\right) x_j} \le s_i && \forall i \in M \qquad (4.36) \\ & w_i = \nu_{i0} + \sum_{j\in N} \nu_{ij} x_j && \forall i \in M \\ & x \in \{0,1\}^N, z \in \mathbb{R}_+^M, s \in \mathbb{R}_+^M, w \in \mathbb{R}_+^M. \end{aligned}$$

Note that constraint (4.36) can be strengthened using the polymatroid inequalities proposed in this chapter.

We test the inequalities in the instances with $n = 200$ and $m = 20$ used in Sen et al. (2015). We compare the MILO formulation and conic formulation with McCormick estimators used in Sen et al. (2015), and the conic formulation with strengthened polymatroid inequalities proposed in this chapter, using a time limit of 600 seconds. Table 4.5 shows the results for different values of the parameters ν_0 and k. The initial gap and root improvement shown in the table correspond to the gap of the classic MILO formulation (with no McCormick inequalities) and the improvement with respect to the MILO formulation.

We see that MISOCO formulations perform much better than MILO formulations, especially in instances with tight cardinality constraint. In particular, instances with $k \le 20$ cannot be solved in 600 seconds using the MILO formulation and results in large end gaps; 19 out of 20 instances can be solved to optimality using the conic formulation proposed in Sen et al. (2015), with an average time of 77 seconds; and using polymatroid inequalities, all

Table 4.5: Assortment optimization with 200 products and $m = 20$.

ν_0	k	igaps	MILO+McCormick				MISOCO+McCormick				inequality (4.26)			
			rimp	nodes	time[#]	egap	rimp	nodes	time[#]	egap	rimp	nodes	time[#]	egap
5	10	50.9	76.4	2,294	600[0]	10.8	100.0	67	24[5]	0.0	100.0	11	8[5]	0.0
	20	18.0	69.4	5,921	600[0]	4.8	99.4	125	148[4]	0.1	100.0	8	13[5]	0.0
	50	0.9	66.7	16,350	267[3]	0.1	100.0	8	127[4]	0.0	100.0	30,344	162[5]	0.0
	100	0.0	100.0	1	3[5]	0.0	100.0	0	123[4]	0.0	100.0	28,600	335[5]	0.0
10	10	46.8	70.9	2,109	600[0]	11.6	100.0	36	26[5]	0.0	100.0	1	7[5]	0.0
	20	39.8	78.1	3,831	600[0]	7.9	100.0	101	33[5]	0.0	100.0	5	11[5]	0.0
	50	5.6	71.4	14,088	600[0]	1.2	100.0	242	40[5]	0.0	100.0	5	11[5]	0.0
	100	0.0	100.0	0	2[5]	0.0	100.0	0	4[5]	0.0	100.0	13,112	147[5]	0.0
Average			4.2	4,460	328[13]	3.6	99.9	78	66[37]	0.0	100.0	70	66[40]	0.0

20 instances with low cardinality can be solved to optimality, with an average time of less than 10 seconds. Among the MISOCO formulations, we observe that the conic formulation proposed in Sen et al. (2015) performs better in instances with large cardinality, and the formulation proposed in this section performs better in instances with low cardinality. Finally, the formulation we propose is the only formulation that is able to solve all 40 instances in less than 600 seconds.

Robust conic instances

We now consider the discrete version of the robust SOCO discussed in Chapter 2. For the sake of completeness we recall the problem and formulation.

We describe a robust formulation of a conic quadratic minimization problem,

$$\min_{z \in Z} a'z + \sqrt{z'Qz}.$$

.

Let a_0 and $Q_0 \succeq 0$ be the nominal mean and covariance and $M = \{1, \ldots, m\}$ be a set of potential events, each of which may increase the mean and covariance by $a_i \geq 0$ and $Q_i \succeq 0,\ i \in M$. Then $a(S) = a_0 + \sum_{i \in S} a_i$ and $Q(S) = Q_0 + \sum_{i \in S} Q_i$ are the mean and covariance when events $S \subseteq M$ are realized. The goal of the robust optimization is to find a solution that minimizes the worst objective given that only a small number, $k \leq m$, events are realized, i.e.,

$$\min_{x \in X} \max_{S \subseteq M : |S| \leq k} a(S)'x + \sqrt{x'Q(S)x}. \tag{4.37}$$

Solving the inner maximization problem for a fixed value of x is $\mathcal{NP}$-hard. However, in Chapter 2 we showed that the *approximate* formulation

$$\begin{aligned}
\min\ & \frac{1}{4}y + a_0'x + t_0 + kw & & \\
\text{s.t. } & a_i'x + t_i \leq w & i \in M & \\
& x'Q_0x \leq yt_0 & & (4.38)\\
& x'Q_ix \leq yt_i & i \in M & (4.39)\\
& x \in X, y \geq 0, t \geq 0, w \geq 0 & & (4.40)
\end{aligned}$$

provides solutions for the original problem with tight guarantees. In particular, *the price of robustness*, i.e., the additional objective cost incurred by the robust formulation, is at most 25% more than the cost of the optimal solution.

In Chapter 2 we considered the case when the feasible region X is convex. However, in many practical applications, the region X may be discrete. In particular, when the variables are restricted to be binary, we can use polymatroid inequalities on constraints (4.38)-(4.39) to strengthen the formulations. In particular, if $Q_0 = D_0 + U_0$ where $D_0 \succeq 0$, $U_0 \succeq 0$ and D_0 is diagonal, we can write constraint (4.38) in an extended formulation as

$$\begin{aligned}
\sum_{i=1}^{n} D_{0_{ii}}x_i + s^2 &\leq yt_i & (4.41)\\
\sqrt{x'U_0x} &\leq s\\
s &\geq 0,
\end{aligned}$$

and apply the inequalities presented in Section 4.4.

We model a decision-maker that seeks a path with minimal value-at-risk and robust to interruptions in the arcs (corresponding to traffic incidents or attacks by an adversary). The feasible region in our computational experiments is thus given by path constraints in a 40×40 grid network. The nominal costs a_0 and Q_0 are generated as in the computational experiments with non-diagonal covariance matrix. There is a potential event corresponding to each arc in the problem, and each event results in an increase of the expected duration and variance of that arc: thus we have that $a_i \sim U[0, 2a_{0i}]e^i$, where e^i is the vector which has value 1 in the i-th position and 0 elsewhere, and the matrix Q_i satisfies

$$Q_{ijk} = \begin{cases} \sim U[0, 2Q_{0ii}] & \text{if } i = j = k\\ = & \text{otherwise.} \end{cases}$$

Table 4.6 shows the results for different values of α. We can use that using strengthened polymatroid cuts results in a better root improvement of 55% - compared to 30% achieved by default CPLEX-, and help solving all problems to optimality in an average of 2,672 seconds. The other configurations, on the other hand, are unable to solve the more difficult problems within a time limit of two hours. Thus we conclude that the polymatroid inequalities presented in this Chapter are helpful for tackling robust discrete conic quadratic problems.

Table 4.6: Path with 1,600 vertices and $k = 4$.

α	igaps	cpx				+ polymatroid				+ strengthened poly			
		rimp	nodes	time[#]	egap	rimp	nodes	time[#]	egap	rimp	nodes	time[#]	egap
0.950	22.6	35.1	63,533	3,124[4]	0.6	44.3	72,322	5,220[3]	1.2	56.8	17,057	917[5]	0.0
0.975	24.1	30.2	95,337	4,239[4]	0.8	41.1	87,697	7,200[0]	3.5	55.2	53,022	2,648[5]	0.0
0.990	25.7	26.6	153,481	7,200[0]	2.2	37.9	80,160	7,200[0]	7.6	53.5	102,578	4,452[5]	0.0
Average		30.6	104,117	4,854[8]	1.2	41.1	80,060	6,540[3]	4.1	55.2	57,552	2,672[15]	0.0

Chapter 5

Submodularity in 0-1 quadratic optimization

5.1 Introduction

In this Chapter we consider the binary quadratically constrained quadratic optimization (BQCQO) problem

$$\text{(BQCQO)} \qquad \begin{array}{ll} \min & a_0'x + x'Q_0x \\ \text{s.t.} & a_k'x + x'Q_kx \leq b_k, \qquad k = 1, \ldots, \ell \\ & x \in \{0,1\}^n, \end{array}$$

where Q_0 and Q_k, $i = k, \ldots, n$ are positive semi-definite matrices. The positive semi-definite assumption is without loss of generality since otherwise terms of the form $\gamma(x_i^2 - x_i)$ with $\gamma \geq 0$ can be added to make the resulting quadratic functions convex (Hammer and Rubin 1970). Problem BQCQO arises often in practice in network problems, with both minimum and maximum cut problems being special cases of unconstrained BQCQO, and in optimization under uncertainty, where the quadratic functions correspond to covariance matrices. Moreover, most combinatorial problems can be modeled as binary linear optimization problems, which is a special case of BQCQO. Our study is based on the structure of the discrete set given by the lower level set of a quadratic function

$$H_Q = \{(x,t) \in \{0,1\}^n \times \mathbb{R} : x'Qx \leq t\}.$$

Different approaches have been suggested to formulate and solve BQCQO. One approach involves the *linearization* of the quadratic terms by introducing additional variables, thus

transforming BQCQO into a MILO. One of the best known linearization techniques (Glover and Woolsey 1974, Hansen 1979) is to replace each term x_i^2 by x_i, to replace each bilinear term $x_i x_j$ by a new variable z_{ij}, and to add the linear constraints $z_{ij} \leq x_i$, $z_{ij} \leq x_j$ and $z_{ij} \geq x_i + x_j - 1$. The linear formulation coincides with the convex underestimators proposed by McCormick (1976) for non-convex optimization. Other linearization techniques have also been proposed that require adding only $O(n)$ additional linear variables (Chaovalitwongse et al. 2004, Adams and Forrester 2005, Sherali and Smith 2007). Another approach is the *Reformulation-Linearization Technique* (RLT) proposed in Adams and Sherali (1986) and Sherali and Adams (1990), which constructs a hierarchy of linear formulations which ultimately results in a convex-hull description of the problem, but each subsequent linear formulation in the hierarchy requires additional variables and constraints. In general there is a tradeoff between the size and the strength of a formulation (Adams et al. 2004), as formulations that use less variables typically result in easier to solve convex subproblems in a branch-and-bound algorithm, but also result in weaker convex relaxations an more branch-and-bound nodes need to be explored to prove optimality.

Linearization of the quadratic functions may result in weak formulations or require a prohibitively large amount of additional variables. In such cases it may be referable to use the natural nonlinear convex relaxation of H_Q, where the binary constraint is replaced by bound constraints. A standard approach to improve the convex formulation is to decompose the matrix Q into two matrices $Q = D + R$, where D is a diagonal matrix with nonnegative entries and $R \succeq 0$ (Poljak and Wolkowicz 1995, Anstreicher 2012). Then observe that for x binary we have that

$$x'Qx \leq t \Leftrightarrow \sum_{i=1}^{n} D_{ii} x_i + x'Rx \leq t \tag{5.1}$$

since $x_i = x_i^2$. Moreover, for all $0 < x_i < 1$ we have that $x_i^2 < x_i$, and we see that formulation (5.1) is stronger.

A different line of research has focused on indetifying and exploiting submodularity in discrete optimization. In the seminal work of Edmonds (1970), key properties of submodular functions and their connections with linear optimization were studied. Since then, submodularity has played a key role in the design of approximation algorithms for maximization problems (Nemhauser et al. 1978, Fisher et al. 1978, Sviridenko 2004, Calinescu et al. 2011, Buchbinder et al. 2012), of polynomial time algorithms for a class of submodular optimization problems (Schrijver 2000, Iwata et al. 2001, Grötschel et al. 2012, Orlin 2009, Iwata and Nagano 2009), and of strong formulations for $\mathcal{NP}$-hard optimization problems (Atamtürk

and Narayanan 2008, 2009, Ahmed and Atamtürk 2011, Atamtürk and Bhardwaj 2015, Zhang et al. 2017).

In this paper we suggest an alternative decomposition to the one proposed in Poljak and Wolkowicz (1995), Anstreicher (2012), where we decompose the quadratic function into a submodular function and a convex function. In particular, we write $Q = U + R$ where $R \succeq 0$ and $U_{ij} \leq 0$ for all $i, j = 1, \ldots, n$, $i \neq j$, and exploit submodularity of the quadratic function

$$f_U(x) = x'Ux$$

to strengthen the formulations. Similar decompositions were used by Atamtürk and Bhardwaj (2017) and Zhang et al. (2017), but they used submodularity to derive a family of *extended polymatroid inequalities*. Since the number of such inequalities is factorial in the number of variables, they implement them using a cutting plane algorithm. In contrast, we use an extended formulation which is more compact -it requires $O(n^2)$ variables and constraints- and is as strong. Our main contributions are:

1. We establish connections between submodularity, classical linearization techniques (McCormick inequalities) and the minimum-cut problem.

2. We formally establish the strength of the proposed decomposition $Q = U + R$ with respect to the natural convex relaxation. In previous works (Atamtürk and Bhardwaj 2017, Zhang et al. 2017) the proposed decomposition was used but the problem of determining whether it is stronger was not addressed. In the current work we show that in general it can be weaker, but that for diagonally dominant matrices U the decomposition is guaranteed to be stronger.

3. Using McCormick inequalities, we get more compact formulations that Atamtürk and Bhardwaj (2017), Zhang et al. (2017). Moreover, despite that using McCormick inequalities may result in weak formulations, by applying them to only a submodular term in the quadratic expression, we get formulations that are guaranteed to be stronger than the natural convex relaxation and as strong as the formulations of Atamtürk and Bhardwaj (2017), Zhang et al. (2017).

4. We show how to derive stronger formulations by using additional constraints of the optimization problem.

The rest of this chapter is organized as follows. In Section 5.2 we review submodular functions, and the connections between quadratic submodular functions and the minimum

cut problem. In Section 5.3 we study strong formulations for submodular quadratic functions, and establish connections between extended polymatroid inequalities, the McCormick inequalities and minimum cut formulations. In Section 5.4 we give sufficient conditions in which the proposed decomposition results in stronger formulations. In Section 5.5 we show how extended polymatroid inequalities can be strengthened using additional constraints of the optimization problem. In Section 5.6 we present preliminary computational experiments. In Section 5.7 we show how the results can be extended to address conic quadratic optimization, and in Section 5.8 we conclude the chapter.

5.2 Preliminaries

Binary quadratic functions, submodularity and minimum cut problems

Given a matrix $U_{n\times n}$, consider the function $f_U : \{0,1\}^n \to \mathbb{R}$ given by

$$f_U(x) = x'Ux.$$

Fisher et al. (1978) showed that the function f_U is submodular if and only the matrix U has non-positive off-diagonal entries, i.e., $U_{ij} \leq 0$ for all $i, j = 1, \ldots, n$ with $i \neq j$. We refer to matrices U corresponding to submodular functions as *submodular matrices*.

Definition 10. We say that matrix U is a submodular matrix if $U_{ij} \leq 0$ for all $i, j = 1, \ldots, n$ with $i \neq j$.

Binary quadratic optimization with submodular matrices and minimum cut problems are closely related. Given a graph $G = (V, A)$, vectors $a \in \mathbb{R}^V$ and $c \in \mathbb{R}^A_+$, consider the minimum cut problem

$$\min \left\{ \sum_{i\in V} a_i x_i + \sum_{(i,j)\in A} c_{ij} y_{ij} : x_i - x_j \leq y_{ij}, \forall (i,j) \in A,\ x \in \{0,1\}^V, y \in \{0,1\}^A \right\}. \quad (5.2)$$

Observe that, in any optimal solution, we have that $c_{ij}y_{ij} = c_{ij}x_i(1 - x_j)$. Thus we that problem (5.2) is equivalent to

$$\min_{x\in\{0,1\}^n} x'Ux, \quad (5.3)$$

where $U_{ii} = a_i + \sum_{(i,j)\in A} c_{ij}$, $U_{ij} = -c_{ij}$ for $(i,j) \in A$, and $U_{ij} = 0$ otherwise.

Example 3. Consider the graph with four vertices and four arcs shown in Figure 5.1. Suppose $a_1 = \ldots = a_4 = 0$ and $c_{12} = c_{13} = c_{24} = c_{34} = 1$. Then any cut with source set S has value $x'Ux$, where $x_i = 1$ if $i \in S$ and

$$U = \begin{pmatrix} 2 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

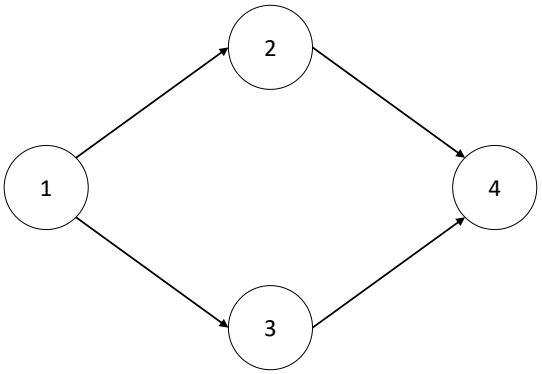


Figure 5.1: Cut example

Remark 14. Observe that the matrix corresponding to a given cut is not unique. Given any submodular matrix U, any matrix $\bar{U}$ such that $U_{ii} = \bar{U}_{ii}$ and $U_{ij} + U_{ji} = \bar{U}_{ij} + \bar{U}_{ji}$ results in the same quadratic function.

5.3 Strong formulations for submodular quadratic functions

In this section we consider the set

$$H_U = \{x \in \{0,1\}^n, t \in \mathbb{R} : f_U(x) \leq t\},$$

where $f_U(x) = x'Ux$ and U is a submodular matrix. For simplicity we assume that the matrix U is symmetric.

One approach to find strong formulations for H_U is to exploit the submodularity of f_U and add extended polymatroid inequalities for the form

$$\pi' x \leq t, \ \pi \in \Pi(f_U) \tag{5.4}$$

to the formulation. This is the approach used by Atamtürk and Bhardwaj (2017) and Zhang et al. (2017). For the particular case of quadratic submodular functions and for a permutation $((1),(2),\ldots,(n))$ of N, the coefficients of the inequality are given by

$$\pi_{(k)} = U_{(k),(k)} + 2\sum_{i=1}^{k-1} U_{(i),(k)}. \tag{5.5}$$

Note that the formulation with extended polymatroid inequalities involves a factorial number of constraints, requiring the use of a cutting plane algorithm. We now show that for the case of submodular quadratic functions, classical linearization techniques may be preferable.

Extended formulations for submodular quadratic constraints

We now give two extended formulations of H_U. The first formulation uses McCormick inequalities and is given by

$$\begin{aligned}
&\sum_{i=1}^{n} U_{ii}x_i + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} U_{ij}z_{ij} \le t \\
\text{(QE)}\qquad &z_{ij} \le x_i,\ z_{ij} \le x_j, && \forall i = 1,\ldots,n,\ j = i+1,\ldots,n, \\
&x_i \in \{0,1\}, z_{ij} \ge 0, && \forall i = 1,\ldots,n,\ j = i+1,\ldots,n.
\end{aligned}$$

Observe that since $U_{ij} \le 0$ for all $i \ne j$, only upper bounds of variables z_{ij} need to be included in the formulation and we omit constraints of the form $x_i + x_j \le 1 + z_{ij}$.

The second formulation corresponds to reformulating H_U as a cut problem. Given any matrix $C_{n\times n}$ such that $C_{ij}+C_{ji} = -U_{ij}-U_{ji}$ with $C_{ij} \ge 0$ for all $i \ne j$, and $C_{ii} = U_{ii} - \sum_{\substack{j=1\\ j\ne i}}^{n} C_{ij}$, let

$$\begin{aligned}
&\sum_{i=1}^{n} C_{ii}x_i + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\ne i}}^{n} C_{ij}y_{ij} \le t \qquad (5.6)\\
\text{(QC)}\qquad &x_i - x_j \le y_{ij}, && \forall i,j = 1,\ldots,n,\ i \ne j \\
&x_i \in \{0,1\},\ y_{ij} \ge 0, && \forall i,j = 1,\ldots,n,\ i \ne j.
\end{aligned}$$

Proposition 29. *Formulations QE and QC are equivalent.*

Proof. Substitute $y_{ij} = x_i - z_{ij}$. Then $x_i - x_j \leq y_{ij} \Leftrightarrow z_{ij} \leq x_j$ and $y_{ij} \geq 0 \Leftrightarrow z_{ij} \leq x_i$. Moreover,

$$\sum_{i=1}^{n} C_{ii}x_i + \sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n} C_{ij}y_{ij} = \sum_{i=1}^{n}\left(C_{ii} + \sum_{\substack{j=1\\j\neq i}}^{n} C_{ij}\right)x_i - \sum_{i=1}^{n}\sum_{j=i+1}^{n}\left(C_{ij}z_{ij} + C_{ji}z_{ji}\right).$$

We can assume without loss of generality that $z_{ij} = z_{ji}$ since $C_{ij} \geq 0$ and $C_{ji} \geq 0$, and we see that QC and QE are equivalent. □

We now show that formulation QE is at least as strong as the formulation obtained by adding all inequalities (5.4).

Proposition 30. *Formulation QE implies all extended polymatroid inequalities* (5.4).

Proof. Let $x \in [0,1]^n$, assume without loss of generality that $x_1 \geq x_2 \geq \ldots \geq x_n$, and let $\pi \in \Pi(f_U)$ be the extreme point corresponding to the permutation $(1, 2, \ldots, n)$. We find that

$$\begin{aligned}
\sum_{i=1}^{n} U_{ii}x_i + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} U_{ij}z_{ij} \geq & \sum_{i=1}^{n} U_{ii}x_i + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} U_{ij}\min\{x_i, x_j\} && (5.7)\\
= & \sum_{i=1}^{n} U_{ii}x_i + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} U_{ij}x_j && (5.8)\\
= & \sum_{j=1}^{n}\left(U_{jj} + 2\sum_{i=1}^{j-1} U_{ij}\right)x_j\\
= & \pi' x,
\end{aligned}$$

where inequality (5.7) follows from $U_{ij} \leq 0$ and $z_{ij} \leq \min\{x_i, x_j\}$, and (5.8) follows from the assumption $x_i \geq x_j$ for $i < j$. Thus we see that formulation QE implies the most violated extended polymatroid inequality $\pi' x \leq t$, and therefore implies all such inequalities. □

Observe that since the additional constraints of QE correspond to the *closure problem* (Picard 1976), it follows that QE and QC give a complete convex hull description of H_U in an extended formulation. Moreover, using extended polymatroid inequalities yields the convex hull of H_U in the original space of variables (Proposition 1). However formulations QE and QC are more compact, requiring only $O(n^2)$ additional variables and constraints. If inequalities (5.4) are implemented using a cut-and-branch algorithm, only a small subset of the inequalities would be added, and formulations QE and QC would result in stronger relaxations in all nodes of the branch-and-bound tree.

5.4 Decomposition schemes

As mentioned in the introduction, given a general quadratic constraint

$$x'Qx \leq t, \tag{5.9}$$

we propose to decompose matrix $Q = U + R$ where U is submodular and $R \succeq 0$ and equivalently write (5.9) as

$$f_U(x) + x'Rx \leq t.$$

Then, as explained in Section 5.3, we use an extended formulation based on McCormick inequalities to represent the lower level set of f_U. However it is not true that for any choice of U the resulting formulation is stronger than the natural convex relaxation. Moreover, although we use McCormick inequalities in our formulation, it is not true that McCormick's linearization is stronger than the natural convex relaxation. Example 4 illustrates these concepts.

Example 4. Consider the matrix $Q = \begin{pmatrix} 1 & -0.5 & -0.5 \\ -0.5 & 1 & 0.7 \\ -0.5 & 0.7 & 1 \end{pmatrix}$, and let $\bar{x}' = \begin{pmatrix} 0.5 & 0.5 & 0.5 \end{pmatrix}$. We now study the strength of different relaxations of (5.9) for at $\bar{x}$. First observe that $\bar{x}'Q\bar{x} = 0.6$. Moreover,

$$\sum_{i=1}^{n} Q_{ii}\bar{x}_i + \sum_{i \neq j : Q_{ij} \leq 0} Q_{ij} \min\{\bar{x}_i, \bar{x}_j\} + \sum_{i \neq j : Q_{ij} > 0} Q_{ij} \max\{\bar{x}_i + \bar{x}_j - 1, 0\} = 0.5,$$

and we see that the McCormick linearization is weaker at $\bar{x}$. Now let $U = \begin{pmatrix} 0 & -1.5 & -1.5 \\ -1.5 & 0 & -0.3 \\ -1.5 & -0.3 & 0 \end{pmatrix}$ and $R = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$. Clearly, U is submodular , $R \succeq 0$ and $Q = U + R$, but

$$\sum_{i=1}^{n} U_{ii}\bar{x}_i + \sum_{i \neq j} U_{ij} \min\{\bar{x}_i, \bar{x}_j\} + x'Rx = -1.05,$$

and we see that, for this choice of matrix U and vector $\bar{x}$, the resulting decomposition using formulation QE is very weak.

Sufficient conditions for strong decompositions

We now show that if U is chosen to be diagonally dominant, then the resulting formulations are stronger for all $x \in [0,1]^n$. First we introduce a third extended formulation for H_U similar to QC but involving a quadratic constraint:

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{n} U_{ij}\right) x_i^2 - \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} U_{ij} y_{ij}^2 \leq t \tag{5.10}$$

$$\text{(QCS)} \qquad \begin{aligned} & x_i - x_j \leq y_{ij}, && \forall i,j = 1,\dots,n,\ i \neq j \\ & x_i \in \{0,1\},\ y_{ij} \geq 0, && \forall i,j = 1,\dots,n,\ i \neq j. \end{aligned}$$

Proposition 31. *Formulation QCS and the formulation induced by the quadratic constraint $x'Ux \leq t$ are equivalent.*

Proof. Let $x \in [0,1]^n$, assume without loss of generality that $x_1 \geq x_2 \geq \ldots \geq x_n$. We can assume without loss of generality that variables y_{ij} in (5.10) satisfy $y_{ij} = (x_i - x_j)^+$. We have that

$$\begin{aligned} \sum_{i=1}^{n}\left(\sum_{j=1}^{n} U_{ij}\right) x_i^2 - \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} U_{ij} y_{ij}^2 &= \sum_{i=1}^{n}\left(\sum_{j=1}^{n} U_{ij}\right) x_i^2 - \sum_{i=1}^{n}\sum_{j=i+1}^{n} U_{ij}(x_i - x_j)^2 \\ &= \sum_{i=1}^{n} U_{ii} x_i^2 + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} U_{ij} x_i x_j = x'Ux, \end{aligned}$$

where the first equality follows by substituting $y_{ij} = x_i - x_j$ for $i < j$ and $y_{ij} = 0$ for $i > j$, and the second inequality follows from rearranging terms. □

Corollary 7. *If U is diagonally dominant, then formulations QE and QC are stronger than the formulation induced by the quadratic constraint $x'Ux \leq t$.*

Proof. By Proposition 31, we need to prove that QE and QC are stronger than QCS, and by Proposition 29, it is sufficient to show than QC is stronger than QCS. Since U is diagonally dominant, we have that $\sum_{j=1}^{n} U_{ij} \geq 0$, thus for $0 \leq x_i \leq 1$ we have that $x_i^2 \sum_{j=1}^{n} U_{ij} \leq x_i \sum_{j=1}^{n} U_{ij}$. Moreover, for $0 \leq y_{ij} \leq 1$ we have that $-U_{ij} y_{ij}^2 \leq -U_{ij} y_{ij}$. Therefore the left hand side of (5.6) is greater or equal than the left hand side of (5.10). □

Example 4 (Continued). Consider the decomposition given by $U = \begin{pmatrix} 0.9 & -0.3 & -0.3 \\ -0.3 & 0.3 & 0 \\ -0.3 & 0 & 0.3 \end{pmatrix}$ and $R = \begin{pmatrix} 0.1 & -0.2 & -0.2 \\ -0.2 & 0.7 & 0.7 \\ -0.2 & 0.7 & 0.7 \end{pmatrix}$. It can be shown that $R \succeq 0$, and clearly U is a diagonally

dominant submodular matrix and $Q = U + R$. Moreover we have that

$$\sum_{i=1}^{n} U_{ii}\bar{x}_i + \sum_{i \neq j} U_{ij} \min\{\bar{x}_i, \bar{x}_j\} + x'Rx = 0.675,$$

and we see that, for this choice of matrix U and vector $\bar{x}$, the resulting decomposition is stronger than the McCormick linearization and the natural convex relaxation.

5.5 Strengthening extended polymatroid inequalities for quadratic functions

We now show how to strengthen extended polymatroid inequalities using additional constraints of the optimization problem. The inequalities are similar from the strengthened polymatroid inequalities presented in Chapter 4.

Given any set $X \subseteq \{0,1\}^n$ and a submodular symmetric matrix U, consider the set

$$L_U = \{(x, z) \in X \times \mathbb{R}_+ : x'Ux \leq z\}. \tag{5.11}$$

Let $(((1), (2), \ldots, (n))$ be a permutation of $\{1, \ldots, n\}$, and let

$$\zeta_{(k)} = \min \left\{ \sum_{i=1}^{k-1} U_{(i),(k)} x_{(i)} : x \in X, x_{(k)} = 1 \right\}. \tag{5.12}$$

Proposition 32. *The inequalities*

$$\sum_{i=1}^{n} (U_{(i)(i)} + 2\zeta_{(i)}) x_{(i)} \leq z \tag{5.13}$$

are valid for L_U.

Proof. We prove that

$$\sum_{i=1}^{k} (U_{(i)(i)} + 2\zeta_{(i)}) x_{(i)} \leq \sum_{i=1}^{k} \sum_{j=1}^{k} U_{(i)(j)} x_{(i)} x_{(j)}, \quad \forall x \in X,$$

by induction on k.

Base case $k = 1$ Trivial.

Inductive step $k \to k+1$ If $x_{(k+1)} = 0$, the results follows directly from the induction hypothesis. Otherwise, for any $x \in X$ with $x_{(k+1)} = 1$ we have that

$$\begin{aligned}
\sum_{i=1}^{k+1}(U_{(i)(i)} + 2\zeta_{(i)})x_{(i)} &= \sum_{i=1}^{k}(U_{(i)(i)} + 2\zeta_{(i)})x_{(i)} + U_{(k+1)(k+1))} + 2\zeta_{(k+1)} \\
&\leq \sum_{i=1}^{k}(U_{(i)(i)} + 2\zeta_{(i)})x_{(i)} + U_{(k+1)(k+1)} + 2\sum_{i=1}^{k} U_{(i)(k)}x_{(i)} \qquad (5.14) \\
&\leq \sum_{i=1}^{k}\sum_{j=1}^{k} U_{(i)(j)}x_{(i)}x_{(j)} + U_{(k+1)(k+1)} + 2\sum_{i=1}^{k} U_{(i)(k)}x_{(i)} \qquad (5.15) \\
&= \sum_{i=1}^{k+1}\sum_{j=1}^{k+1} U_{(i)(j)}x_{(i)}x_{(j)},
\end{aligned}$$

where (5.14) follows from the definition of ζ and (5.15) follows from the induction hypothesis. □

Remark 15. If $X = \{0,1\}$, then inequalities (5.13) reduce to the extended polymatroid inequalities (5.4). If $X \subset \{0,1\}$, then inequalities (5.13) dominate (5.4).

Remark 16. We can use any relaxation of X, and in particular a linear programming relaxation, to compute lower bounds on the coefficients ζ efficiently.

5.6 Computational experiments

In this section we report preliminary computational experiments in solving BQCQO in a branch-and-bound algorithm, using the proposed decomposition. All experiments are conducted using Gurobi 7.5 solver on a workstation with a 2.93GHz Intel®Core™ i7 CPU and 8 GB main memory and with a single thread. The time limit is set to two hours and Gurobi's default settings are used.

The instances considered are randomly generated binary quadratic optimization problems of the form

$$\min\ x'Qx \qquad (5.16)$$

$$\text{s.t.} \sum_{i=1}^{n} a_i x_i \geq b \qquad (5.17)$$

$$\sum_{i=1}^{n} x_i = k \qquad (5.18)$$

$$x \in \{0,1\}^n. \qquad (5.19)$$

Problems of the form (5.16)-(5.19) arise often in decision-making under uncertainty. Given a choice of n items with return vector $r \sim \mathcal{N}(a, Q)$, the decision maker wishes to minimizes risk (5.16) while guaranteeing a given expected return (5.17), and satisfying additional constraints (5.18).

For simplicity we consider diagonal dominant matrices Q. Observe that in such cases a decomposition can be computed efficiently, where

$$R_{ij} = \begin{cases} \sum\limits_{k \neq i : Q_{ik} > 0} Q_{ij} & \text{if } i = j \\ Q_{ij} & \text{if } j \neq i \text{ and } Q_{ij} > 0 \\ 0 & \text{if } j \neq i \text{ and } Q_{ij} \leq 0, \end{cases}$$

and $U = Q - R$. Observe that in such cases both matrices Q and R are diagonally dominant and in particular $R \succeq 0$.

In the computational experiments we generate matrix Q mimicking a factor model, i.e. $Q = \Sigma + D$ with $\Sigma = XFX'$, where $X \in \mathbb{R}^{n \times m}$ is the exposure matrix, $F \in \mathbb{R}^{m \times m}$ is the factor covariance matrix and D is a diagonal matrix with the specific variances. In our experiments matrix Σ is generated with $m = 20$, $F = GG'$, with $G \in \mathbb{R}^{m \times m}$ and $G_{ij} \sim U[-1, 1]$, $X_{ij} \sim U[0, 1]$ with probability 0.2 and $X_{ij} = 0$ otherwise. To guarantee diagonal dominance, we set $\Sigma_{ii} = \sum_{j \neq i} |\Sigma_{ij}|$ for all $i = 1, \ldots, n$. Each diagonal element of D is given by $D_{ii} \sim U[0, \delta \bar{q}]$, where $\delta \geq 0$ is a diagonal dominance parameter, $\bar{q} = \frac{1}{N} \sum_{i \in N} \Sigma_{ii}$. The vector of expected return is generated as $a_i \sim U[0.85\sqrt{Q_{ii}}, 1.15\sqrt{Q_{ii}}]$, so the expected return of an item is proportional to its risk. We set $k = 0.15n$ and $b = \beta \sum_{i=1}^{n} a_i$, where β is a return parameter.

We test three different formulations for the objective:

def. The natural convex relaxation, $x'Qx$.

diag. The diagonal decomposition (5.1), $\sum_{i=1}^{n} D_{ii} x_i + x'\Sigma x$.

sub. The proposed submodular decomposition, $f_U(x) + x'Rx$, implemented with the extended formulation QE.

Tables 5.1 and 5.2 present the results for different problem sizes n. Each row represents the average over five instances generated with the same parameters and shows for each formulation and for different values of the parameters δ and β, the initial gap (`igap`), the the number of nodes explored (`nodes`), the time elapsed in seconds (`time`), and the end gap

(`egap`)[in brackets, the number of instances solved to optimality (#)]. The initial gap is computed as `igap` $= \frac{t_{\text{opt}} - t_{\text{relax}}}{|t_{\text{opt}}|} \times 100$, where t_{opt} is the objective value of the best feasible solution at termination and t_{relax} is the objective value of the continuous relaxation.

Table 5.1: Experiments with $n = 100$.

δ	β	def.				diag.				sub.			
		igap	nodes	time	egap[#]	igap	nodes	time	egap[#]	igap	nodes	time	egap[#]
	0.15	69.1	22,492,560	7,200	8.7[0]	63.2	22,519,280	7,200	8.7[0]	57.2	33	0	0.0[5]
0.1	0.17	65.2	20,455,440	7,200	8.9[0]	60.3	20,214,080	7,200	0.0[0]	52.9	114	1	0.0[5]
	0.20	52.9	14,971,898	5,202	0.9[3]	49.8	15,117,790	5,225	0.9[3]	40.6	132	1	0.0[5]
Average		**62.4**	**19,306,633**	**6,534**	**6.2[3]**	**57.8**	**19,283,717**	**6,542**	**6.[3]**	**50.2**	**93**	**1**	**0.0[5]**
	0.15	70.9	17,600,914	6,100	4.0[2]	49.9	17,714,534	6,081	4.0[2]	43.1	19	0	0.0[5]
0.5	0.17	66.9	17,912,420	6,622	4.3[1]	48.0	17,776,260	6,618	4.3[1]	40.7	118	1	0.0[5]
	0.20	52.7	4,807,298	1,553	0.0[5]	39.5	4,800,138	1,553	0.0[5]	31.5	59	1	0.0[5]
Average		**63.5**	**13,440,211**	**4,758**	**2.8[8]**	**45.8**	**13,430,311**	**4,751**	**2.8[8]**	**38.5**	**66**	**1**	**0.0[15]**
	0.15	72.1	13,337,372	4,818	2.0[3]	41.5	13,324,100	4,842	2.0[3]	34.0	73	1	0.0[5]
1.0	0.17	68.0	11,883,686	4,621	1.5[3]	38.8	12,035,878	4,660	1.5[3]	31.5	42	1	0.0[5]
	0.20	53.4	1,779,981	515	0.0[5]	31.1	1,770,680	506	0.0[5]	24.3	98	0	0.0[5]
Average		**64.5**	**9,000,346**	**3,318**	**1.2[11]**	**37.1**	**9,043,553**	**3,336**	**1,2[11]**	**29.9**	**71**	**1**	**0.0[15]**

Table 5.2: Experiments with $n = 400$.

δ	β	def				diag.				sub.			
		igap	nodes	time	egap[#]	igap	nodes	time	egap[#]	igap	nodes	time	egap[#]
	0.15	68.2	895,789	7,200	51.0[0]	62.0	901,037	7,200	51.0[0]	55.5	189	30	0.0[5]
0.1	0.17	64.2	980,722	7,200	49.8[0]	59.0	1,007,364	7,200	49.8[0]	51.3	734	100	0.0[5]
	0.20	52.6	1,892,852	7,200	38.1[0]	49.1	1,863,882	7,200	38.1[0]	40.0	1,416	269	0.0[5]
Average		**61.7**	**1,252,331**	**7,200**	**46.3[0]**	**56.7**	**1,256,577**	**7,200**	**46.3[0]**	**49.0**	**780**	**135**	**0.0[15]**
	0.15	69.7	1,040,322	7,200	37.9[0]	47.0	1,041,994	7,200	37.9[0]	40.0	176	11	0.0[5]
0.5	0.17	65.7	1,115,298	7,200	37.0[0]	45.6	1,105,936	7,200	37.0[0]	38.0	493	34	0.0[5]
	0.20	52.8	2,185,410	7,200	28.6[0]	38.3	2,175,204	7,200	28.6[0]	30.5	899	75	0.0[5]
Average		**62.7**	**1,451,028**	**7,200**	**34.5[0]**	**43.6**	**1,442,933**	**7,200**	**34.5[0]**	**36.2**	**523**	**41**	**0.0[15]**
	0.15	70.9	1,196,390	7,200	29.2[0]	37.7	1,197,232	7,200	29.2[0]	30.1	296	10	0.0[5]
1.0	0.17	67.1	1,276,054	7,200	27.8[0]	35.9	1,278,420	7,200	27.8[0]	28.7	124	13	0.0[5]
	0.20	53.9	2,350,680	7,200	21.5[0]	29.7	2,350,224	7,200	21.5[0]	23.3	265	21	0.0[5]
Average		**63.9**	**1,606,682**	**7,200**	**26.2[0]**	**34.5**	**1,608,258**	**7,200**	**26.2[0]**	**27.4**	**228**	**15**	**0.0[15]**

Observe that `diag.` results in better inital gaps than `def.`, and `sub.` results in better initial gaps than `diag.`. Thus, using the more general submodular matrices instead of diagonal matrices in the decomposition indeed results in stronger formulation. We also see that there is no noticeable difference in the performance of Gurobi's branch-and-bound algorithm for formulations `def.` and `diag.` despite the different convex relaxations; a

possible explanation is that Gurobi's by default decomposes binary quadratic functions using a diagonal component.

There is a dramatic improvement in the performance of Gurobi's branch-and-bound algorithm when using configuration `sub.`: instances that cannot be solved in two hours of branch-and-bound can now be solved in under one second for $n = 100$, and in under two minutes for $n = 400$. The performance improvement is due in part to the strength of the convex relaxation. Moreover, since binary linear optimization is better understood than binary quadratic optimization, using a tight linear representation for the submodular quadratic function allows Gurobi to use more tools than what would be possible otherwise.

5.7 Extensions to conic quadratic optimization

We now show how the ideas presented in this chapter can be used in conic quadratic optimization with binary variables. Consider a conic quadratic constraint of the form

$$\sqrt{x'Qx} \leq s \tag{5.20}$$

where $Q = U + R$ where U is a symmetric submodular matrix such that $x'Ux \geq 0$ and $R \succeq 0$. Then, introducing a new variable $t \geq 0$, we can write (5.20) in an extended formulation as

$$\sqrt{t^2 + x'Rx} \leq s$$
$$\sqrt{x'Ux} \leq t.$$

Thus, in this section, we study the set

$$K = \left\{(x,t) \in \{0,1\}^n \times \mathbb{R} : \sqrt{x'Ux} \leq t\right\}.$$

If the function f_U is non-decreasing, i.e.,

$$U_{ii} + 2\sum_{\substack{j=1 \\ j\neq i}}^{n} U_{ij} \geq 0, \ \forall i = 1, \ldots, n,$$

then the function $g(x) = \sqrt{x'Ux}$ is submodular (Zhang et al. 2017). Then extended polymatroid inequalities $\pi'x \leq t$, $\pi \in \Pi(g)$ can be added to strengthen the formulation. However, as illustrated in Section 5.3, using valid inequalities in an extended formulation may result in stronger formulations.

We propose a class of quadratic submodular inequalities. Given a permutation $((1),(2),\ldots,(n))$ of N, define for $k=1,\ldots,n$ and $\ell=1,\ldots,k-1$

$$\begin{aligned}
\sigma_{(k)} &= \sum_{i=1}^{k-1} U_{(i)(i)} + 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k-1} U_{(i)(j)} \\
\Lambda_{(k)(k)} &= \sqrt{\sigma_{(k)} + U_{(k)(k)}} - \sqrt{\sigma_{(k)}} \\
\Lambda_{(k)(\ell)} = \Lambda_{(\ell)(k)} &= \frac{1}{2}\left(\sqrt{\sigma_{(k)} + U_{(k)(k)} + 2\sum_{i=1}^{\ell} U_{(i)(k)}} - \sqrt{\sigma_{(k)} + U_{(k)(k)} + 2\sum_{i=1}^{\ell-1} U_{(i)(k)}}\right)
\end{aligned}$$

and consider the inequality

$$f_\Lambda(x) := x'\Lambda x \le t. \tag{5.21}$$

Proposition 33. *Inequality* (5.21) *is valid for* K.

Proof. Let $x \in \{0,1\}^n$. We have that

$$\begin{aligned}
\sqrt{x'Ux} =& \sqrt{\sum_{i=1}^{n} U_{(i)(i)}x_{(i)} + 2\sum_{i=1}^{n}\sum_{j=i+1}^{n} U_{(i)(j)}x_{(i)}x_{(j)}} \\
=& \sum_{k=1}^{n}\left(\sqrt{\sum_{i=1}^{k} U_{(i)(i)}x_{(i)} + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k} U_{(i)(j)}x_{(i)}x_{(j)}} - \sqrt{\sum_{i=1}^{k-1} U_{(i)(i)}x_{(i)} + 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k-1} U_{(i)(j)}x_{(i)}x_{(j)}}\right) \\
=& \sum_{k=1}^{n}\left(\sqrt{U_{(k)(k)} + 2\sum_{i=1}^{k} U_{(i)(k)}x_{(i)} + \sum_{i=1}^{k-1} U_{(i)(i)}x_{(i)} + 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k-1} U_{(i)(j)}x_{(i)}x_{(j)}}\right. \\
& \left. - \sqrt{\sum_{i=1}^{k-1} U_{(i)(i)}x_{(i)} + 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k-1} U_{(i)(j)}x_{(i)}x_{(j)}}\right) x_{(k)} \\
\ge& \sum_{k=1}^{n}\left(\sqrt{U_{(k)(k)} + 2\sum_{i=1}^{k} U_{(i)(k)}x_{(i)} + \sigma_{(k)}} - \sqrt{\sigma_{(k)}}\right) x_{(k)} = \sum_{k-1}^{n} \psi_k(x)x_{(k)}
\end{aligned} \tag{5.22}$$

where the last inequality follows from concavity of the square root function, $U_{(k)(k)} + 2\sum_{i=1}^{k} U_{(i)(k)}x_{(i)} \ge 0$ and $\sigma_{(k)} \ge \sum_{i=1}^{k-1} U_{(i)(i)}x_{(i)} + 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k-1} U_{(i)(j)}x_{(i)}x_{(j)}$ (since f_U is non-decreasing). Moreover we have that for all $k=1,\ldots,n$, function $\psi_k(x) + \sqrt{\sigma_{(k)}}$ is the composition of a concave function and a monotone modular function (plus a constant),

and is therefore submodular. Therefore we have that

$$\begin{aligned}\psi_k(x) + \sigma_k \geq & \sqrt{U_{(k)(k)} + \sigma_{(k)}} + \sum_{\ell=1}^{k} \left(\sqrt{U_{(k)(k)} + \sigma_{(k)} + 2\sum_{i=1}^{\ell} U_{(i)(k)}} \right. \\ & \left. - \sqrt{U_{(k)(k)} + \sigma_{(k)} + 2\sum_{i=1}^{\ell-1} U_{(i)(k)}} \right) x_\ell. \\ = & \sqrt{U_{(k)(k)} + \sigma_{(k)}} + 2\sum_{\ell=1}^{k} \Lambda_{(\ell)(k)} x_\ell.\end{aligned}$$

Replacing in (5.22), we get the desired result. □

Since the right hand side of inequalities (5.21) is a submodular quadratic function, the inequality can be strengthened further (Section 5.3).

Proposition 34. *The extended polymatroid inequalities corresponding to permutation* $((1), (2), \ldots, (n))$ *of functions* f_Λ *and* $g(x) = \sqrt{x'Ux}$ *are the same.*

Proof. The $\pi \in \Pi(f_\Lambda)$ be the extreme point of $EP(f_\Lambda)$ corresponding to permutation $((1), (2), \ldots, (n))$. Then from (5.5) we have that

$$\begin{aligned}\pi_{(k)} = & \Lambda_{(k)(k)} + 2\sum_{\ell=1}^{k-1} \Lambda_{(\ell)(k)} \\ = & \sqrt{\sigma_{(k)} + U_{(k)(k)}} - \sqrt{\sigma_{(k)}} \\ & + \sum_{\ell=1}^{k-1} \left(\sqrt{\sigma_{(k)} + U_{(k)(k)} + 2\sum_{i=1}^{\ell} U_{(i)(k)}} - \sqrt{\sigma_{(k)} + U_{(k)(k)} + 2\sum_{i=1}^{\ell-1} U_{(i)(k)}} \right) \\ = & \sqrt{\sigma_{(k)} + U_{(k)(k)} + 2\sum_{i=1}^{k-1} U_{(i)(k)}} - \sqrt{\sigma} \\ = & \sqrt{\sum_{i=1}^{k} U_{(i)(i)} + 2\sum_{i=1}^{k}\sum_{j}^{k} U_{(i)(j)}} - \sqrt{\sum_{i=1}^{k-1} U_{(i)(i)} + 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k-1} U_{(i)(j)}}.\end{aligned}$$

It is easy to check that it coincides with the extreme point of $EP(g)$ corresponding to the same permutation. □

5.8 Conclusions

In this chapter we study how to exploit submodularity in binary quadratic optimization. We show that, once a quadratic submodular component as been identified, using classical linearization techniques give the convex hull description of the submodular component. Moreover, when the submodular component corresponds to a quadratic function with a diagonally dominant matrix, we obtain tighter formulations than the natural convex relaxation and the formulation resulting from fully linearizing the quadratic expression. Our computational experiments indicate that the approach proposed can result in considerable improvements when solving BQCQO with a branch-and-bound algorithm.

Chapter 6

Conclusion

In this dissertation we consider diverse classes of mixed-integer nonlinear optimization problems, with focus on optimization problems with conic quadratic functions. We study both maximization and minimization of conic quadratic objectives with linear constraints, and optimization problems with quadratic and conic quadratic constraints. We propose algorithms that account both for the non-linearity of the problem and discrete variables. The algorithms share many similarities with classic algorithms for MILO. We now recount our main contributions and provide directions for future research.

In Chapter 2 we study the problem of maximizing a class of nonlinear functions over the vertices of polyhedra. We prove that there exists an optimal solution to the natural convex relaxation of the problem, and that by rounding the solution to a suitable vertex we obtain a 1/2-approximation algorithm. When the nonlinear function corresponds to a conic quadratic function, the approximation ratio is improved to 4/5. The approximation algorithm proposed is the first approximation proposed for polyhedra that are not down-monotone. Moreover, our algorithm fits the paradigm of rounding fractional solutions obtained from convex relaxations that has proven successful for MILO but has not been exploited for MINLO. Finally, using the natural convex relaxation, we propose approximation algorithms for a class of robust SOCO, extending previous results for robust LP.

In Chapter 3 we study the problem of minimizing a conic quadratic function over polyhedra. We rewrite the objective function using the perspective function of a quadratic function, and show how to solve the optimization problem by solving a sequence of QPs. We show that the resulting algorithm, when using the simplex method to solve the QPs, scales better than the barrier algorithm used by commercial software and can be an order of magnitude faster in larger instances. Moreover, when used in a branch-and-bound algorithm, our ap-

proach outperforms all alternatives used by commercial software, including SOCO-based branch-and-bound and LP-based branch-and-bound with extended formulations.

In Chapter 4 we study the structure of a single mixed-binary conic quadratic constraint with separable quadratic term. We give the convex-hull description of the set considered, and show how the inequalities can be implemented in commercial software using a single conic quadratic inequality and linear cuts. We also give the complete convex hull description of the set defined by a rotated cone constraint with binary and continuous variables. We extend the previous results to consider upper bounds on the continuous variables and other constraints of the optimization problem. In our computational experiments we show that using the proposed inequalities may result in orders-of-magnitude improvements with respect to commercial software.

In Chapter 5 we propose a decomposition strategy for binary quadratic optimization. The decomposition aims to exploit submodular component of a general quadratic function. By using the connections between quadratic submodular functions and the minimum cut problem, we find a strong extended formulation for the submodular component requiring $O(n^2)$ additional variables and constraints. We also give sufficient conditions for the proposed decomposition to be stronger than the natural convex relaxation of the problem. Our preliminary computational experiments suggest that the proposed approach can be very effective for solving binary quadratic problem to optimality in a branch-and-cut algorithm.

In this dissertation we give four different algorithmic approaches, each suited for a specific problem considered. In our computational experiments we demonstrate how each algorithm by itself improves the existing solution approaches. However, our current ability to solve nonlinear discrete optimization problems is still limited compared to our ability to solve their linear counterparts. Bridging the gap between linear and nonlinear discrete optimization requires combining and integrating different techniques for MINLO. Promising ideas for future research include integrating algorithms with warm-starts (Chapter 3) with valid inequalities (Chapter 4) in a branch-and-cut algorithm; and using the ideas given in Chapter 2 to efficiently produce feasible solutions in a branch-and-bound algorithm. Decomposition strategies like the one proposed in Chapter 5 can be used in presolve to further strengthen the convex relaxations. Developing branching and node selection strategies for MINLO, and integrating them with other existing techniques is another direction worth exploring.

Most of the work to date in MINLO has focused in the pure-binary case, which is simpler due to the underlying polyhedral structure. Furthermore, only simple feasible regions have been considered to date. In Chapter 4 we begin studying MISOCO with binary and contin-

uous variables and general feasible regions, and illustrate both its similarities and differences with the pure-binary case. We hope that the work presented here leads to the study of other structured problems often encountered in practice. In particular, understanding the structure of conic constraints with knapsack constraints or a fixed-charge network structure is of particular interest, due to the numerous applications in fields such as finance or machine learning.

Appendix A

Appendix

A.1 Regret bound for Algorithm 1

Let $x^* = \arg\min_{x \in V_X} w'x$ be the best solution, so the regret in round t is $r^t = w'x^* - w'x^t$. Recall that the confidence region at time t is the ellipsoid

$$B_t = \left\{ \nu \in \mathbb{R}^n : \sum_{i=1}^{n} (\nu_i - \hat{w}_i^t)^2 d_i^t \leq \beta_t \right\},$$

and in each round the play the solution corresponding to

$$\max_{x \in V_X, \nu \in B_t} \nu' x.$$

To prove the high probability regret bounds, we prove first that if $w \in B_t$ for all t, then the sum of the squares of the regret cannot be large. Then we show that the probability that the true weight w belongs to the confidence region B_t for all t is high. Finally we use these facts to prove the regret bounds.

Sum of squares regret bound

Proposition 35. *For all $v \in B_t$ and $x \in X$,*

$$|v'x - \hat{w}'x| \leq \sqrt{\beta_t \sum_{i=1}^{n} \frac{x_i}{d_i^t}}.$$

Proof.

$$\begin{aligned}
|v'x - \hat{w}'x| &= \left| \sum_{i=1}^{n} (v_i - \hat{w}_i) \sqrt{d_i^t} \frac{x_i}{\sqrt{d_i^t}} \right| \\
&\leq \sqrt{\sum_{i=1}^{n} (v_i - \hat{w}_i)^2 d_i^t} \sqrt{\sum_{i=1}^{n} \frac{x_i^2}{d_i^t}} && \text{(Cauchy-Schwarz)} \\
&\leq \sqrt{B_t} \sqrt{\sum_{i=1}^{n} \frac{x_i}{d_i^t}}
\end{aligned}$$

□

Proposition 36. *If $w \in B_t$ for all t, then*

$$\sum_{t=1}^{T} \left(r^t\right)^2 \leq 4n\beta_T(\ln T + 1).$$

Proof. Given t, let $v_t \in B_t$ be the vector that maximizes $v'x_t$. Since $w \in B_t$, we have $w'x^* \leq v_t'x_t$ and

$$\begin{aligned}
r^t &= w'x^* - w'x^t \\
&\leq (v_t - w)'x^t \\
&= (v_t - \hat{w}^t)'x^t + (\hat{w}^t - w)'x^t \\
&\leq 2\sqrt{\beta_t \sum_{i=1}^{n} \frac{x_i}{d_i^t}}. && \text{(Proposition 35)}
\end{aligned}$$

We have then

$$\begin{aligned}
\sum_{t=1}^{T} \left(r^t\right)^2 &\leq 4\beta_T \sum_{t=1}^{T} \sum_{i=1}^{n} \frac{x_i}{d_i^t} \\
&\leq 4n\beta_T \sum_{t=1}^{T} \frac{1}{t} \\
&\leq 4n\beta_T(\ln T + 1).
\end{aligned}$$

□

Remark 17. Note that Proposition 36 gives a stronger bound than the corresponding $8n\beta_T \ln T$ of Theorem 6 of Dani et al. (2008).

Ellipsoid growth

Let $Z_t = \sum_{i=1}^{n}(w_i - \hat{w}_i^t)^2 d_i^t$ be the squared distance between the true weight vector and the center of the confidence region. In this section we study the growth of Z_t. Recall that $d_i^1 = 1$ and $\hat{w}_i^1 = 1$ for all $i \in N$.

Proposition 37. *For all t,*

$$Z_t \leq n + 2\sum_{\tau=1}^{t-1}\sum_{i=1}^{n} \frac{(w_i - \hat{w}_i^\tau)\, d_i^\tau\, (w_i - \tilde{w}_i^\tau)\, x_i^\tau}{d_i^{\tau+1}} + \sum_{\tau=1}^{t-1}\sum_{i=1}^{n} \frac{(w_i - \tilde{w}_i^\tau)^2\, x_i^\tau}{d_i^{\tau+1}}. \tag{A.1}$$

Proof. We prove the result by induction. For the inductive step, we will use the identities

$$\begin{aligned} d_i^{t+1} &= d_i^t + x_i^t, \\ \hat{w}_i^{t+1} &= \frac{\hat{w}_i^t d_i^t + \tilde{w}_i^t x_i^t}{d_i^{t+1}}. \end{aligned}$$

Consider:

$$\begin{aligned} Z_{t+1} &= \sum_{i=1}^{n} \left(w_i - \hat{w}_i^{t+1}\right)^2 d_i^{t+1} \\ &= \sum_{i=1}^{n} \left(w_i - \frac{\hat{w}_i^t d_i^t + \tilde{w}_i^t x_i^t}{d_i^{t+1}}\right)^2 d_i^{t+1} \\ &= \sum_{i=1}^{n} \frac{\left(w_i d_i^t - \hat{w}_i^t d_i^t + w_i x_i^t - \tilde{w}_i^t x_i^t\right)^2}{d_i^{t+1}} \\ &= \sum_{i=1}^{n} \left(\left(w_i - \hat{w}_i^t\right)^2 \frac{{d_i^t}^2}{d_i^{t+1}} + 2\frac{(w_i - \hat{w}_i^t)\, d_i^t\, (w_i - \tilde{w}_i^t)\, x_i^t}{d_i^{t+1}} + \frac{\left(w_i - \tilde{w}_i^t\right)^2 x_i^t}{d_i^{t+1}} \right) \\ &\leq Z_t + 2\sum_{i=1}^{n} \frac{(w_i - \hat{w}_i^t)\, d_i^t\, (w_i - \tilde{w}_i^t)\, x_i^t}{d_i^{t+1}} + \sum_{i=1}^{n} \frac{\left(w_i - \tilde{w}_i^t\right)^2 x_i^t}{d_i^{t+1}}. \end{aligned}$$

By induction, it follows that

$$Z_t \leq Z_1 + 2\sum_{\tau=1}^{t-1}\sum_{i=1}^{n} \frac{(w_i - \hat{w}_i^\tau)\, d_i^\tau\, (w_i - \tilde{w}_i^\tau)\, x_i^\tau}{d_i^{\tau+1}} + \sum_{\tau=1}^{t-1}\sum_{i=1}^{n} \frac{(w_i - \tilde{w}_i^\tau)^2\, x_i^\tau}{d_i^{\tau+1}}.$$

Finally, $Z_1 = \sum_{i=1}^{n}(w_i - \hat{w}_i^1)^2 d_i^1 \leq n$, completing the proof. □

Note that the last term in equation (A.1) is bounded by

$$\sum_{\tau=1}^{t-1}\sum_{i=1}^{n} \frac{x_i^\tau}{d_i^{\tau+1}}(\tilde{w}_i^\tau - w_i)^2 \leq n\sum_{\tau=2}^{t} \frac{1}{\tau} \leq n \ln t.$$

Concentration

To bound the remaining term, define the indicator $E_t = \begin{cases} 1 & \text{if } Z_\tau \leq \beta_\tau \text{ for all } \tau \leq t, \\ 0 & \text{otherwise} \end{cases}$, and consider the random variable

$$M_t = 2E_t \sum_{i=1}^{n} x_i^t \frac{d_i^t}{d_i^{t+1}} (\hat{w}_i^t - w_i)(\tilde{w}_i^t - w_i).$$

Proposition 38. *M_t is a martingale difference sequence with respect to the sequence of plays $\mathcal{H}_t$.*

Proof.

$$\mathbb{E}\left[M_t | \mathcal{H}_t\right] = 2E_t \sum_{i=1}^{n} x_i^t \frac{d_i^t}{d_i^{t+1}} (\hat{w}_i^t - w_i) \mathbb{E}\left[(\tilde{w}_i^t - w_i)\right] = 0,$$

since $x^1, \cdots, x^t$, $\hat{w}^1, \cdots, \hat{w}^t$ and $E_1, \cdots, E_t$ are fully determined by the story $\mathcal{H}_t$, and $\mathbb{E}\left[(\tilde{w}_i^t - w_i)\right] = 0$ for all $i \in N$. □

Proposition 39. *Given $\delta \in (0, 1)$,*

$$P\left(\forall t, \sum_{\tau=1}^{t-1} M_\tau \leq \frac{\beta_t}{2}\right) \geq 1 - \delta.$$

Before proving Proposition 39, we introduce a theorem from Freedman (1975).

Theorem 3 (Freedman)**.** *Suppose $M_1, \cdots, M_t$ is a martingale difference sequence, and b is a uniform upper bound on the steps M_i. Let V denote the sum of conditional variances,*

$$V = \sum_{\tau=1}^{t} Var(M_\tau | M_1, \cdots, M_{\tau-1}).$$

Then, for every $a_1, a_2 > 0$,

$$P\left(\sum_{\tau=1}^{t} M_\tau \geq a_1 \text{ and } V \leq a_2\right) \leq \exp\left(\frac{-a_1^2}{2a_2 + 2a_1 b/3}\right).$$

Proof of Proposition 39. Note that the steps M_t are bounded by $\sqrt{n\beta_t}$:

$$\begin{aligned} |M_t| &= 2E_t \left| \sum_{i=1}^{n} x_i^t \frac{d_i^t}{d_i^{t+1}} (\hat{w}_i^t - w_i)(\tilde{w}_i^t - w_i) \right| \\ &\leq 2\sqrt{E_t \sum_{i=1}^{n} d_i^t (\hat{w}_i^t - w_i)^2} \sqrt{\sum_{i=1}^{n} x_i^{t2} \frac{d_i^t}{d_i^{t+1^2}} (\tilde{w}_i^t - w_i)^2} && \text{(Cauchy-Schwarz)} \\ &\leq 2\sqrt{\beta_t}\sqrt{n} = \sqrt{\beta_t n}. && \text{(Proposition 35)} \end{aligned}$$

Moreover, the sum of conditional variances is bounded by $4\beta_t n \ln t$:

$$\begin{aligned}
\sum_{\tau=1}^{t} \mathrm{Var}(M_\tau | M_1, \cdots, M_{\tau-1}) \leq & 4 \sum_{\tau=1}^{t} E_\tau \left(\sum_{i=1}^{n} d_i^\tau (\hat{w}_i^\tau - w_i)^2 \right) \left(\sum_{i=1}^{n} x_i^{\tau 2} \frac{d_i^\tau}{d_i^{\tau+1^2}} (\tilde{w}_i^\tau - w_i)^2 \right) \\
\leq & 4 \sum_{\tau=1}^{t} E_\tau \beta_\tau \left(\sum_{i=1}^{n} x_i^\tau \frac{d_i^\tau}{d_i^{\tau+1^2}} \right) \\
\leq & 4\beta_t \sum_{i=1}^{n} \sum_{\tau=1}^{t} E_\tau x_i^\tau \frac{1}{d_i^{\tau+1}} \\
\leq & 4\beta_t n \ln(\max\{\tau \leq t : E_\tau = 1\}) \\
\leq & 4\beta_t n \ln t.
\end{aligned}$$

Since the sum of conditional variances is always bounded, we can apply Theorem 3 with $a_1 = \frac{\beta_t}{2}$, $a_2 = 4\beta_T n \ln T$:

$$\begin{aligned}
P\left(\sum_{\tau=1}^{t} M_\tau \geq \frac{\beta_t}{2} \right) = & P\left(\sum_{\tau=1}^{t} M_\tau \geq \frac{\beta_t}{2} \text{ and } V \leq 4\beta_T n \ln T \right) \\
\leq & \exp\left(\frac{-\left(\frac{\beta_t}{2}\right)^2}{8\beta_T n \ln T + \frac{2}{3}\frac{\beta_t}{2}\sqrt{n\beta_t}} \right) \\
= & \exp\left(\frac{-\beta_t}{32 n \ln T + \frac{4}{3}\sqrt{n\beta_t}} \right) \\
\leq & \max\left\{ \exp\left(\frac{-\beta_t}{64 n \ln T} \right), \exp\left(\frac{-3\sqrt{\beta_t}}{8\sqrt{n}} \right) \right\} \\
\leq & \frac{\delta}{t^2}. \qquad \text{(Definition of } \beta_t\text{)}
\end{aligned}$$

Finally, we apply union bound to get

$$\begin{aligned}
P\left(\sum_{\tau=1}^{t-1} M_\tau \geq \frac{\beta_t}{2} \text{ for some } t \right) \leq & \sum_{t=1}^{\infty} P\left(\sum_{\tau=1}^{t-1} M_\tau \geq \frac{\beta_t}{2} \right) \\
\leq & \sum_{t=2}^{\infty} \frac{\delta}{t^2} \\
\leq & \delta \left(\frac{\pi^2}{6} - 1 \right) \leq \delta.
\end{aligned}$$

□

Remark 18. Note that the bound of the conditional variances is tighter than the bound of $8\beta_t n \ln t$ given in the proof of Lemma 14 of Dani et al. (2008). We can therefore use a smaller β_t than in Dani et al. (2008).

Confidence

Proposition 40. *With high probability, the true weight vector is contained in B_t for all t. In particular,*

$$P\left(\forall t,\ E_t = 1\right) = P\left(\forall t,\ \sum_{i=1}^{n}(w_i - \hat{w}_i^t)^2 d_i^t \leq \beta_t\right) \geq 1 - \delta.$$

Proof. We prove by induction that, if the event described in Proposition 39 holds, then $\sum_{i=1}^{n}(w_i - \hat{w}_i^t)^2 d_i^t \leq \beta_t$ for all t. Note that, since $\sum_{i=1}^{n}(w_i - \hat{w}_i^1)^2 d_i^1 \leq n \leq \beta_1$, $E_1 = 1$ .Now assume $\sum_{\tau=1}^{t-1} M_\tau \leq \frac{\beta_t}{2}$. Then, by inductive hypothesis, $E_\tau = 1$ for all $\tau < t$, and we have

$$\begin{aligned}
\sum_{i=1}^{n}(w_i - \hat{w}_i^t)^2 d_i^t &\leq n + \sum_{\tau=1}^{t-1} M_\tau + \sum_{\tau=1}^{t-1}\sum_{i=1}^{n} \frac{x_i^\tau}{d_i^{\tau+1}}(\tilde{w}_i^\tau - w_i)^2 && \text{(Proposition 37)} \\
&\leq n + \frac{\beta_t}{2} + n \ln t \\
&\leq \beta_t.
\end{aligned}$$

We have shown $E_t = 1$, completing the induction. □

Regret bounds

We now prove the high probability bounds for the regret.

Proof of Theorem 1.

$$\begin{aligned}
R(T) =& \sum_{t=1}^{T} r_t \\
\leq& \sqrt{T \sum_{t=1}^{T} r_t^2} && \text{(Cauchy-Schwarz)} \\
\leq& 2\sqrt{Tn\beta_T \ln T}. && \text{(Proposition 36)}
\end{aligned}$$

□

Proof of Theorem 2. Note that, for all t, either $r_t = 0$ or $r_t \geq \Delta$. Either way, $r_t \leq \frac{r_t^2}{\Delta}$. We

have then

$$\begin{aligned} R(T) &= \sum_{t=1}^{T} r_t \\ &\leq \sum_{t=1}^{T} \frac{r_t^2}{\Delta} \\ &\leq \frac{4n\beta_T \ln T}{\Delta}. \end{aligned} \qquad \text{(Proposition 36)}$$

□

A.2 Branch-and-bound algorithm

Algorithm 5 describes the branch-and-bound algorithm used in computations. Throughout the algorithm, we maintain a list L of the nodes to be processed. Each node is a tuple (S, B, lb), where S is the subproblem, B is a basis for warm starting the continuous solver and lb is a lower bound on the objective value of S. In line 3 list L is initialized with the root node. For each node, the algorithm calls a continuous solver (line 9) which returns a tuple $(x, \bar{B}, z)$, where x is an optimal solution of S, $\bar{B}$ is the corresponding optimal basis and z is the optimal objective value (or ∞ if S is infeasible). The algorithm then checks whether the node can be pruned (lines 10-11), x is integer (lines 12-15), or it further branching is needed (lines 16-18).

We now describe the specific implementations of the different subroutines. For branching (line 17) we use the maximum infeasibility rule, which chooses the variable x_i with value v_i furtherest from an integer (ties broken arbitrarily). The subproblems $S_{\leq}$ and $S_{\geq}$ in line 18 are created by imposing the constraints $x_i \leq \lfloor v_i \rfloor$ and $x_i \geq \lceil v_i \rceil$, respectively. The `PULL` routine in line 5 chooses, when possible, the child of the previous node which violates the bound constraint by the least amount, and chooses the node with the smallest lower bound when the previous node has no child nodes. The list L is thus implemented as a sorted list ordered by the bounds, so that the `PULL` operation is done in $O(1)$ and the insertion is done in $O(\log |L|)$ (note that in line 18 we only add to the list the node that is not to be processed immediately). A solution x is assumed to be integer (line 12) when the values of all variables are within 10^{-5} of an integer. Finally, the algorithm is terminated when $\frac{ub - lb_{best}}{|lb_{best} + 10^{-10}|} \leq 10^{-4}$, where lb_{best} is the minimum lower bound among all the nodes in the tree.

Algorithm 5 Branch-and-bound algorithm

Input: P, discrete minimization problem
Output: Optimal solution x^*

1: $ub \leftarrow \infty$ ▷ Upper bound
2: $x^* \leftarrow \emptyset$ ▷ Best solution found
3: $L \leftarrow \{(P, \emptyset, -\infty)\}$ ▷ list of nodes L initialized with the original problem
4: **while** $L \neq \emptyset$ **do**
5: $(S, B, lb) \leftarrow \texttt{PULL}(L)$ ▷ select and remove one element from L
6: **if** $lb \geq ub$ **then**
7: **go to** line 4
8: **end if**
9: $(x, \bar{B}, z) \leftarrow \texttt{SOLVE}(S, B)$ ▷ solve continuous relaxation
10: **if** $z \geq ub$ **then** ▷ if S is infeasible then $z = \infty$
11: **go to** line 4 ▷ prune by infeasibility or bounds
12: **else if** x is integer **then**
13: $ub \leftarrow z$ ▷ update incumbent solution
14: $x^* \leftarrow x$
15: **go to** line 4 ▷ prune by integer feasibility
16: **else**
17: $(S_{\leq}, S_{\geq}) \leftarrow \texttt{BRANCH}(x)$ ▷ create two subproblems
18: $L \leftarrow L \cup \{(S_{\leq}, \bar{B}, z), (S_{\geq}, \bar{B}, z)\}$ ▷ add the subproblems to L
19: **end if**
20: **end while**
21: **return** x^*

The maximum infeasibility rule is chosen due to its simplicity. The other rules and parameters correspond to the ones used in CPLEX branch-and-bound algorithm in default configuration.

A.3 Convex hull of L^2_σ

A point (x, y, z) belongs to $\text{conv}(L^2_\sigma)$ if and only if there exist $x_1, x_2, y_1, y_2, z_1, z_2, \lambda$ such that the system

$$x = (1-\lambda)x_1 + \lambda x_2 \tag{A.2}$$
$$y = (1-\lambda)y_1 + \lambda y_2 \tag{A.3}$$
$$z = (1-\lambda)z_1 + \lambda z_2 \tag{A.4}$$
$$z_1 \geq \sqrt{\sigma + dy_1^2} \tag{A.5}$$
$$z_2 \geq \sqrt{\sigma + c + dy_2^2} \tag{A.6}$$
$$0 \leq y_1, y_2 \leq 1,\ x_1 = 0,\ x_2 = 1 \tag{A.7}$$

is feasible. Observe that from (A.2) and (A.7) we can conclude that $\lambda = x$. Also observe that from (A.2), (A.5) and (A.6) we have that

$$\begin{aligned} & z = (1-x)z_1 + xz_2 \\ \Leftrightarrow\ & z \geq (1-x)\sqrt{\sigma + dy_1^2} + x\sqrt{\sigma + c + dy_2^2}. \end{aligned}$$

Therefore, the system is feasible if and only if

$$z \geq \min_{y_1, y_2} (1-x)\sqrt{\sigma + dy_1^2} + x\sqrt{\sigma + c + dy_2^2} \tag{A.8}$$
$$\text{s.t. } y = (1-x)y_1 + xy_2 \qquad (\gamma)$$
$$y_1 \leq 1 \qquad (\alpha_1)$$
$$y_2 \leq 1 \qquad (\alpha_2)$$
$$y_1 \geq 0 \qquad (\beta_1)$$
$$y_2 \geq 0, \qquad (\beta_2)$$

and let γ, α and β be the dual variables of the optimization problem above. From KKT conditions for variables y_1 and y_2 we find that

$$\begin{aligned} -(1-x)\frac{dy_1}{\sqrt{\sigma + dy_1^2}} &= \gamma(1-x) + \alpha_1 - \beta_1 \\ -x\frac{dy_2}{\sqrt{\sigma + c + dy_2^2}} &= \gamma x + \alpha_2 - \beta_2 \\ \implies \frac{y_1}{\sqrt{\sigma + dy_1^2}} + \bar{\alpha}_1 - \bar{\beta}_2 &= \frac{y_2}{\sqrt{\sigma + c + dy_2^2}} + \bar{\alpha}_2 - \bar{\beta}_2, \end{aligned} \tag{A.9}$$

where $\bar{\alpha}, \bar{\beta}$ correspond to α and β after scaling. We can deduce from (A.9) and complementary slackness that $y_1, y_2 > 0$ (unless $y = 0$) and that $y_1 \leq y_2$. Therefore, in an optimal solution either $0 < y_1, y_2 < 1$ (and $\bar{\alpha} = \bar{\beta} = 0$) or $y_2 = 1$ (and $\bar{\alpha}_2 \geq 0$). If $\bar{\alpha} = \bar{\beta} = 0$, then

$$y_1^* = y\frac{\sqrt{\sigma}}{x\sqrt{c+\sigma} + (1-x)\sqrt{\sigma}} \quad \text{and}$$
$$y_2^* = y\frac{\sqrt{c+\sigma}}{x\sqrt{c+\sigma} + (1-x)\sqrt{\sigma}}$$

satisfy conditions (A.9) and (A.3). Moreover, if

$$\begin{aligned} & y_2^* \leq 1 \\ \Leftrightarrow\ & y \leq \frac{x\sqrt{c+\sigma} + (1-x)\sqrt{\sigma}}{\sqrt{c+\sigma}} = x + (1-x)\sqrt{\frac{\sigma}{c+\sigma}}, \end{aligned}$$

then y_1^*, y_2^* also satisfy bound constraints, and thus correspond to an optimal solution to the optimization problem. Replacing in (A.8), we find that

$$z \geq \sqrt{\left(\sqrt{\sigma} + x(\sqrt{c+\sigma} - \sqrt{\sigma})\right)^2 + dy^2}$$

when $y \leq x + (1-x)\sqrt{\frac{\sigma}{\sigma+c}}$. On the other hand, if $y_1^* > 1$, an optimal solution to the optimization problem is given by $\bar{y}_2 = 1$ and $\bar{y}_1 = \frac{y-x}{1-x}$. Replacing in (A.8)

$$z \geq \sqrt{\sigma(1-x)^2 + d(y-x)^2} + x\sqrt{\sigma + c + d}$$

when $y \geq x + (1-x)\sqrt{\frac{\sigma}{\sigma+c}}$.

Bibliography

Adams, W. P. and Forrester, R. J. (2005). A simple recipe for concise mixed 0-1 linearizations. *Operations research letters*, 33:55–61.

Adams, W. P., Forrester, R. J., and Glover, F. W. (2004). Comparisons and enhancement strategies for linearizing mixed 0-1 quadratic programs. *Discrete Optimization*, 1:99–120.

Adams, W. P. and Sherali, H. D. (1986). A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32:1274–1290.

Ahmed, S. (2006). Convexity and decomposition of mean-risk stochastic programs. *Mathematical Programming*, 106:433–446.

Ahmed, S. and Atamtürk, A. (2011). Maximizing a class of submodular utility functions. *Mathematical Programming*, 128:149–169.

Ahmed, S. and Papageorgiou, D. J. (2013). Probabilistic set covering with correlations. *Operations Research*, 61:438–452.

Aktürk, M. S., Atamtürk, A., and Gürel, S. (2009). A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, 37:187–191.

Aktürk, M. S., Atamtürk, A., and Gürel, S. (2010). Parallel machine match-up scheduling with manufacturing cost considerations. *Journal of Scheduling*, 13:95–110.

Alizadeh, F. (1995). Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51.

Alizadeh, F. and Goldfarb, D. (2003). Second-order cone programming. *Mathematical programming*, 95:3–51.

Anstreicher, K. M. (2012). On convex relaxations for quadratically constrained quadratic programming. *Mathematical programming*, 136:233–251.

Atamtürk, A., Berenguer, G., and Shen, Z.-J. (2012). A conic integer programming approach to stochastic joint location-inventory problems. *Operations Research*, 60:366–381.

Atamtürk, A. and Bhardwaj, A. (2015). Supermodular covering knapsack polytope. *Discrete Optimization*, 18:74–86.

Atamtürk, A. and Bhardwaj, A. (2017). Network design with probabilistic capacities. BCOL Research Report 16.01, UC Berkeley.

Atamtürk, A. and Jeon, H. (2017). Lifted polymatroid for mean-risk optimization with indicator variables. BCOL Research Report 17.01, UC Berkeley.

Atamtürk, A. and Narayanan, V. (2007). Cuts for conic mixed-integer programming. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 16–29. Springer Berlin Heidelberg.

Atamtürk, A. and Narayanan, V. (2008). Polymatroids and mean-risk minimization in discrete optimization. *Operations Research Letters*, 36:618–622.

Atamtürk, A. and Narayanan, V. (2009). The submodular 0-1 knapsack polytope. *Discrete Optimization*, 6:333–344.

Atamtürk, A. and Narayanan, V. (2010). Conic mixed-integer rounding cuts. *Mathematical Programming*, 122:1–20.

Atamtürk, A. and Narayanan, V. (2011). Lifting for conic mixed-integer programming. *Mathematical programming*, 126:351–363.

Badanidiyuru, A. and Vondrák, J. (2014). Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1497–1514. SIAM.

Balcan, M. and Harvey, N. J. A. (2010). Learning submodular functions. *CoRR*, abs/1008.2159.

Belotti, P., Góez, J. C., Pólik, I., Ralphs, T. K., and Terlaky, T. (2015). A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization. In *Numerical Analysis and Optimization*, pages 1–35. Springer.

Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., and Mahajan, A. (2013). Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1131.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust optimization*. Princeton University Press.

Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization. *Mathematics of operations research*, 23:769–805.

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations research letters*, 25:1–13.

Ben-Tal, A. and Nemirovski, A. (2001a). *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS-SIAM Series on Optimization. SIAM, Philadelphia.

Ben-Tal, A. and Nemirovski, A. (2001b). On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26:193–205.

Bertsimas, D. and Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical programming*, 98:49–71.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations research*, 52:35–53.

Bier, V. M., Nagaraj, A., and Abhichandani, V. (2005). Protection of simple series and parallel systems with components of different values. *Reliability Engineering & System Safety*, 87:315 – 323.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Bixby, R. E. (2012). A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, pages 107–121.

Bonami, P. (2011). Lift-and-project cuts for mixed integer convex programs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 52–64. Springer.

Borchers, B. and Mitchell, J. E. (1994). An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers & Operations Research*, 21:359–367.

Buchbinder, N., Feldman, M., Naor, J., and Schwartz, R. (2012). A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 649–658.

Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. (2011). Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40:1740–1766.

Çay, S. B., Pólik, I., and Terlaky, T. (2017). Warm-start of interior point methods for second order cone optimization via rounding over optimal Jordan frames. ISE Technical Report 17T-006, Lehigh University.

Ceria, S. and Soares, J. (1999). Convex programming for disjunctive convex optimization. *Mathematical Programming*, 86:595–614.

Çezik, M. T. and Iyengar, G. (2005). Cuts for mixed 0-1 conic programming. *Mathematical Programming*, 104:179–202.

Chaovalitwongse, W., Pardalos, P. M., and Prokopyev, O. A. (2004). A new linearization technique for multi-quadratic 0–1 programming problems. *Operations Research Letters*, 32:517–522.

Chen, W., Wang, Y., and Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In Dasgupta, S. and Mcallester, D., editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 151–159. JMLR Workshop and Conference Proceedings.

Chong, J.-K., Ho, T.-H., and Tang, C. S. (2001). A modeling framework for category assortment planning. *Manufacturing & Service Operations Management*, 3:191–210.

Dadush, D., Dey, S. S., and Vielma, J. P. (2011). The split closure of a strictly convex body. *Operations Research Letters*, 39:121–126.

Dani, V., Hayes, T. P., and Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. In *COLT*.

Dantzig, G. B., Orden, A., and Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5:183–196.

Edmonds, J. (1970). Submodular functions, matroids, and certain polyhedra. In Guy, R., Hanani, H., Sauer, N., and Schönenheim, J., editors, *Combinatorial Structures and Their Applications*, pages 69–87. Gordon and Breach.

El Ghaoui, L., Oks, M., and Oustry, F. (2003). Worst-case value-at-risk and robust portfolio optimization: A conic programming approach. *Operations Research*, 51:543–556.

Fisher, M., Nemhauser, G., and Wolsey, L. (1978). An analysis of approximations for maximizing submodular set functions II. In Balinski, M. and Hoffman, A., editors, *Polyhedral Combina-*

torics, volume 8 of *Mathematical Programming Studies*, pages 73–87. Springer Berlin Heidelberg.

Frangioni, A. and Gentile, C. (2006). Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming*, 106:225–236.

Frangioni, A. and Gentile, C. (2007). Sdp diagonalizations and perspective cuts for a class of nonseparable miqp. *Operations Research Letters*, 35:181–185.

Freedman, D. A. (1975). On tail probabilities for martingales. *The Annals of Probability*, 3:pp. 100–118.

Gen, M. and Yun, Y. (2006). Soft computing approach for reliability optimization: State-of-the-art survey. *Reliability Engineering & System Safety*, 91:1008 – 1026. Special Issue - Genetic Algorithms and ReliabilitySpecial Issue - Genetic Algorithms and Reliability.

Glover, F. and Woolsey, E. (1974). Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Operations research*, 22:180–182.

Goemans, M. X., Harvey, N. J. A., Iwata, S., and Mirrokni, V. S. (2009). Approximating submodular functions everywhere. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 535–544.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42:1115–1145.

Grötschel, M., Lovász, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197.

Grötschel, M., Lovász, L., and Schrijver, A. (2012). *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media.

Günlük, O. and Linderoth, J. (2010). Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Mathematical programming*, 124:183–205.

György, A., Linder, T., and Ottucsák, G. (2006). The shortest path problem under partial monitoring. In Lugosi, G. and Simon, H., editors, *Learning Theory*, volume 4005 of *Lecture Notes in Computer Science*, pages 468–482. Springer Berlin Heidelberg.

Hammer, P. L. and Rubin, A. A. (1970). Some remarks on quadratic programming with 0-1 variables. *Revue française d'informatique et de recherche opérationnelle. Série verte*, 4:67–79.

Hansen, P. (1979). Methods of nonlinear 0-1 programming. *Annals of Discrete Mathematics*, 5:53–70.

Hausken, K. (2008). Strategic defense and attack for series and parallel reliability systems. *European Journal of Operational Research*, 186:856 – 881.

Hijazi, H., Bonami, P., and Ouorou, A. (2013). An outer-inner approximation for separable mixed-integer nonlinear programs. *INFORMS Journal on Computing*, 26:31–44.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2013). *Convex Analysis and Minimization Algorithms I: Fundamentals*, volume 305. Springer Science & Business Media.

Hochbaum, D. S. (1982). Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on computing*, 11:555–556.

Ishii, H., Shiode, S., Nishida, T., and Namasuya, Y. (1981). Stochastic spanning tree problem. *Discrete Applied Mathematics*, 3:263–273.

Iwata, S., Fleischer, L., and Fujishige, S. (2001). A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM (JACM)*, 48:761–777.

Iwata, S. and Nagano, K. (2009). Submodular function minimization under covering constraints. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 671–680. IEEE.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311. ACM.

Kılınç, M., Linderoth, J., and Luedtke, J. (2010). Effective separation of disjunctive cuts for convex mixed integer nonlinear programs. *Optimization Online*.

Kılınç-Karzan, F. and Yıldız, S. (2015). Two-term disjunctions on the second-order cone. *Mathematical Programming*, 154:463–491.

Kulik, A., Shachnai, H., and Tamir, T. (2009). *Maximizing Submodular Set Functions Subject to Multiple Linear Constraints*, chapter 60, pages 545–554.

Levitin, G. and Hausken, K. (2008). Protection vs. redundancy in homogeneous parallel systems. *Reliability Engineering & System Safety*, 93:1444–1451.

Leyffer, S. (2001). Integrating SQP and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization & Applications*, 18:295–309.

Lobo, M. S., Vandenberghe, L., Boyd, S., and Lebret, H. (1998). Applications of second-order cone programming. *Linear algebra and its applications*, 284:193–228.

Lubin, M., Yamangil, E., Bent, R., and Vielma, J. P. (2016). Polyhedral approximation in mixed-integer convex optimization. *arXiv preprint arXiv:1607.03566*.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part iconvex underestimating problems. *Mathematical programming*, 10:147–175.

Megiddo, N. (1991). On finding primal- and dual-optimal bases. *INFORMS Journal on Computing*, 3:63–65.

Modaresi, S., Kılınç, M. R., and Vielma, J. P. (2016). Intersection cuts for nonlinear integer programming: Convexification techniques for structured sets. *Mathematical Programming*, 155:575–611.

Nahmias, S. (2001). *Production and Operations Analysis*. McGraw Hill.

Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization. Interscience Series in Discrete Mathematics and Optimization*.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14:265–294.

Nemirovski, A. and Scheinberg, K. (1996). Extension of Karmarkar's algorithm onto convex quadratically constrained quadratic problems. *Mathematical Programming*, 72:273–289.

Nemirovski, A. and Todd, M. (2008). Interior-point methods for optimization. *Acta Numerica*, 17:191–234.

Nesterov, Y. and Nemirovskii, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics.

Nesterov, Y. E. and Todd, M. J. (1998). Primal-dual interior-point methods for self-scaled cones. *SIAM Journal on optimization*, 8:324–364.

Orlin, J. B. (2009). A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118:237–251.

Picard, J.-C. (1976). Maximal closure of a graph and applications to combinatorial problems. *Management science*, 22:1268–1272.

Poljak, S. and Wolkowicz, H. (1995). Convex relaxations of (0, 1)-quadratic programming. *Mathematics of Operations Research*, 20:550–561.

Pulleyblank, W. (1973). Faces of matching polyhedra. Ph.D. Thesis, University of Waterloo.

Rusmevichientong, P., Shen, Z.-J. M., and Shmoys, D. B. (2010). Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research*, 58:1666–1680.

Rusmevichientong, P. and Tsitsiklis, J. N. (2010). Linearly parameterized bandits. *Mathematics of Operations Research*, 35:395–411.

Schrijver, A. (2000). A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80:346–355.

Sen, A., Atamtürk, A., and Kaminsky, P. (2015). A conic integer programming approach to constrained assortment optimization under the mixed multinomial logit model. Research Report BCOL.15.06, IEOR, University of California–Berkeley.

Shen, Z.-J. M., Coullard, C., and Daskin, M. S. (2003). A joint location-inventory model. *Transportation science*, 37:40–55.

Sherali, H. D. and Adams, W. P. (1990). A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3:411–430.

Sherali, H. D. and Smith, J. C. (2007). An improved linearization strategy for zero-one quadratic programming problems. *Optimization Letters*, 1:33–47.

Stubbs, A. R. and Mehrotra, S. (1999). A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86:515–532.

Sviridenko, M. (2004). A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32:41 – 43.

Tawarmalani, M. and Sahinidis, N. V. (2005). A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249.

Van de Panne, C. and Whinston, A. (1964). Simplicial methods for quadratic programming. *Naval Research Logistics Quarterly*, 11:273–302.

Van Ryzin, G. and Mahajan, S. (1999). On the relationship between inventory costs and variety benefits in retail assortments. *Management Science*, 45:1496–1509.

Vielma, J. P., Dunning, I., Huchette, J., and Lubin, M. (2016). Extended formulations in mixed integer conic quadratic programming. *Mathematical Programming Computation*.

Vondrák, J., Chekuri, C., and Zenklusen, R. (2011). Submodular function maximization via the multilinear relaxation and contention resolution schemes. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 783–792. ACM.

Williamson, D. P. and Shmoys, D. B. (2011). *The design of approximation algorithms*. Cambridge university press.

Wolfe, P. (1959). The simplex method for quadratic programming. *Econometrica: Journal of the Econometric Society*, pages 382–398.

Yildirim, E. A. and Wright, S. J. (2002). Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization*, 12:782–810.

Yu, J. and Ahmed, S. (2015). Polyhedral results for a class of cardinality constrained submodular minimization problems. *Discrete Optimization*.

Zhang, Y., Jiang, R., and Shen, S. (2017). Ambiguous chance-constrained bin packing under mean-covariance information.