Summer 2010

Convex relaxations in nonconvex and applied optimization

Jieqiu Chen
University of Iowa

CONVEX RELAXATIONS IN NONCONVEX AND APPLIED OPTIMIZATION

by

Jieqiu Chen

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Business Administration
in the Graduate College of
The University of Iowa

July 2010

Thesis Supervisor: Associate Professor Samuel Burer

ABSTRACT

Traditionally, *linear programming* (LP) has been used to construct convex relaxations in the context of branch and bound for determining global optimal solutions to nonconvex optimization problems. As *second-order cone programming* (SOCP) and *semidefinite programming* (SDP) become better understood by optimization researchers, they become alternative choices for obtaining convex relaxations and producing bounds on the optimal values. In this thesis, we study the use of these convex optimization tools in constructing strong relaxations for several nonconvex problems, including 0-1 integer programming, nonconvex box-constrained quadratic programming (BoxQP), and general *quadratic programming* (QP).

We first study a SOCP relaxation for 0-1 integer programs and a sequential relaxation technique based on this SOCP relaxation. We present desirable properties of this SOCP relaxation, for example, this relaxation cuts off all fractional extreme points of the regular LP relaxation. We further prove that the sequential relaxation technique generates the convex hull of 0-1 solutions asymptotically.

We next explore nonconvex quadratic programming. We propose a SDP relaxation for BoxQP based on relaxing the first- and second-order KKT conditions, where the difficulty and contribution lie in relaxing the second-order KKT condition. We show that, although the relaxation we obtain this way is equivalent to an existing SDP relaxation at the root node, it is significantly stronger on the children nodes in a branch-and-bound setting.

New advance in optimization theory allows one to express QP as optimizing a linear function over the convex cone of completely positive matrices subject to linear constraints, referred to as completely positive programming (CPP). CPP naturally admits strong semidefinite relaxations. We incorporate the first-order KKT conditions of QP into the constraints of QP, and then pose it in the form of CPP to obtain a strong relaxation. We employ the resulting SDP relaxation inside a finite branch-and-bound algorithm to solve the QP. Comparison of our algorithm with commercial global solvers shows potential as well as room for improvement.

The remainder is devoted to new techniques for solving a class of large-scale linear programming problems. First order methods, although not as fast as second-order methods, are extremely memory efficient. We develop a first-order method based on Nesterov's smoothing technique and demonstrate the effectiveness of our method on two machine learning problems.

Abstract Approved: _____
Thesis Supervisor

_____
Title and Department

_____
Date

CONVEX RELAXATIONS IN NONCONVEX AND APPLIED OPTIMIZATION

by

Jieqiu Chen

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Business Administration
in the Graduate College of
The University of Iowa

July 2010

Thesis Supervisor: Associate Professor Samuel Burer

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Jieqiu Chen

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Business Administration at the July 2010 graduation.

Thesis Committee: _____
Samuel Burer, Thesis Supervisor

_____
Kurt Anstreicher

_____
Nick Street

_____
Jeffrey Ohlmann

_____
Pavlo Krokhmal

To my parents: Chen, Dajun and Zhou, Ze

ACKNOWLEDGEMENTS

I owe my deepest gratitude to my advisor Samuel Burer. It is his guidance and support that enables me to finish this thesis. Sam is the most effective teacher I have encountered in my academic life. During my first year at the University of Iowa, I took Sam's Linear Programming class, which I found was one of the easiest graduate classes. It is not because the subject of linear program itself was easy, but that Sam had explained complicated theories in a way easy for students to understand that made the class easy. In addition to Linear Programming, Sam also taught me numerous other things: from basic concepts of semidefinite programming to how to use SeDuMi to solve an optimization problem, and from breaking down hard math problems into easier ones to how to make a research presentation. I am especially thankful for that he corrected the grammar errors in my thesis and has always been helpful when I face difficulties with research. I could not have imagined having a better advisor for my research.

I would like to thank my dissertation committee members, Kurt Anstreicher, Nick Street, Jeff Ohlmann, and Pavlo Krokhmal for their critical questions and invaluable suggestions that improved the thesis. I am indebted to Pavlo for his short-term mentorship that inspired my interests in second-order cone programming. My sincere thanks go to Kurt, Jeff and Nick for their helpful advice and encouragement during my course of study. I would also like to express my appreciation to various professors at the University of Iowa, Ann Campbell, Barry Thomas, Richard Dykstra, Kasturi

Varadarajan, as well as Dr. Miguel Anjos at the University of Waterloo for being supportive on my career.

I would like to thank many past and present fellow PhD students, Justin Goodson, Nick Leifker, Kaan Ataman, Xin Ying Qiu, Yi Zhang, and Chuanjie Liao. Special thanks to Hui Chen, who has given me advice and support on many occasions. I am also grateful to the PhD program coordinator, Renea Jay, and our department secretary, Barb Carr, for their assistance that made my PhD life hassle free.

Many thanks to my friends at Iowa City, Beverly Robalino, Zhejia Ling, Tingting Xiao, Tingting Que, Ke Yang, Yi Jiang, Fei Su, and Shanshan Zhao, for their companionship and the fun they brought to my life. I will always remember the wonderful times we spent together.

Last but not the least, I would like to thank my father and mother for their unconditional love and support that allow me to go this far. I would also like to express my gratitude to my boyfriend, Bolin Ding, for his love, encouragement, and faith in me.

ABSTRACT

Traditionally, *linear programming* (LP) has been used to construct convex relaxations in the context of branch and bound for determining global optimal solutions to nonconvex optimization problems. As *second-order cone programming* (SOCP) and *semidefinite programming* (SDP) become better understood by optimization researchers, they become alternative choices for obtaining convex relaxations and producing bounds on the optimal values. In this thesis, we study the use of these convex optimization tools in constructing strong relaxations for several nonconvex problems, including 0-1 integer programming, nonconvex box-constrained quadratic programming (BoxQP), and general *quadratic programming* (QP).

We first study a SOCP relaxation for 0-1 integer programs and a sequential relaxation technique based on this SOCP relaxation. We present desirable properties of this SOCP relaxation, for example, this relaxation cuts off all fractional extreme points of the regular LP relaxation. We further prove that the sequential relaxation technique generates the convex hull of 0-1 solutions asymptotically.

We next explore nonconvex quadratic programming. We propose a SDP relaxation for BoxQP based on relaxing the first- and second-order KKT conditions, where the difficulty and contribution lie in relaxing the second-order KKT condition. We show that, although the relaxation we obtain this way is equivalent to an existing SDP relaxation at the root node, it is significantly stronger on the children nodes in a branch-and-bound setting.

New advance in optimization theory allows one to express QP as optimizing a linear function over the convex cone of completely positive matrices subject to linear constraints, referred to as completely positive programming (CPP). CPP naturally admits strong semidefinite relaxations. We incorporate the first-order KKT conditions of QP into the constraints of QP, and then pose it in the form of CPP to obtain a strong relaxation. We employ the resulting SDP relaxation inside a finite branch-and-bound algorithm to solve the QP. Comparison of our algorithm with commercial global solvers shows potential as well as room for improvement.

The remainder is devoted to new techniques for solving a class of large-scale linear programming problems. First order methods, although not as fast as second-order methods, are extremely memory efficient. We develop a first-order method based on Nesterov's smoothing technique and demonstrate the effectiveness of our method on two machine learning problems.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF ALGORITHMS

Algorithm

CHAPTER 1
INTRODUCTION

In this thesis, we study the use of convex optimization for solving several non-convex problems and one application of nonsmooth convex optimization. As stated in Boyd and Vandenberghe (2004), a convex optimization problem is one of the form

$$\begin{aligned} \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq b_i, \quad i = 1, \ldots, m, \end{aligned} \tag{1.1}$$

where the functions $f_0, \ldots, f_m : \Re^n \to \Re$ are convex, i.e., satisfy

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

for all $x, y \in \Re^n$ and all α, β with $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$. If $f_0, \ldots, f_m$ are linear, then (1.1) is a linear programming (LP) problem. LP is one of the most well-studied fields within optimization and has mature theories, e.g., duality theory, and solutions techniques, such as the simplex method and interior-point methods, as well as countless applications (Dantzig, 1998; Wright, 1997; Vanderbei, 2007). Modern linear programming technology is so powerful that solving a linear program of reasonable size is easy. Although we cannot say that the task of solving a general convex optimization problem is as easy as solving a linear program, at least theoretically a similar story can be said about (1.1): it is solvable within a number of operations that is a polynomial of the problem dimensions (provided the functions satisfy certain assumptions). In fact, Nesterov and Nemirovskii (1994) have established interior-point polynomial algorithms that solve a large class of convex optimization problems.

If at least one of $f_0, \ldots, f_m$ is not convex, then (1.1) becomes nonconvex optimization problem, which in general is known to be $\mathcal{NP}$-hard. However, such problems arise naturally in diverse areas such as engineering product design, Nash equilibria in economics, traffic equilibria in transportation, supply chain management, etc.; see Pintér (1995) for a collection of real world nonconvex optimization problems. There are various methodological approaches for nonconvex problems: local optimization methods, exact methods, and meta heuristics.

Among a variety of exact methods, branch-and-bound is one of the most widely used approaches. Branch-and-bound works by subdividing the entire search space or feasible region into smaller spaces (the branching process) while maintaining upper and lower bounds on the objective function being optimized (the bounding feature). Recursive branching defines a tree structure, whose nodes are the subdivided spaces. For a minimization problem, a relaxation of the problem is usually solved at each node get a lower bound on the optimal value. A global upper bound (GUB) is used to fathom nodes whose lower bounds exceed the GUB. For a node that cannot be fathomed, it is subdivided and the resulting children nodes are added to the candidate list.

Traditionally, LP relaxation is probably the primary choice for relaxation in solving nonconvex optimization problems, which is evidenced by the extensive use of linear programming relaxations in mixed integer programming and some of the most popular nonconvex optimization solvers, such as BARON (Sahinidis, 1996; Tawarmalani and Sahinidis, 2002). Using LP as the relaxation technique has many bene-

fits; LP is very well understood by optimization researchers and modern LP solvers are very powerful. With the rapid development of convex optimization, several new problem classes have become standard: second-order cone programming (SOCP), semidefinite programming (SDP), etc. These new problem classes have similar duality theory as LP. Moreover, LP, SOCP, and SDP can be viewed under a unified conic programming framework and be treated by interior-point methods (Nesterov and Nemirovskii (1994)). On the application side, SOCPs have been applied to many science and engineering problems (Lobo et al. (1998)); SDPs have been used to get bounds on combinatorial optimization problems (Goemans, 1998), and to solve engineering problems (Vandenberghe and Boyd (1999)). On the computational side, fast software packages have been developed for solving these problem classes, for example, MOSEK (MOSEK, Inc. (2007)) for solving SOCP, and SeDuMi (Sturm (1999)) for solving SDP. These advancs, both on the theoretical and computational sides, have led to the use of SOCP and SDP as alternative relaxations for nonconvex optimization problems.

In this thesis, we mainly explore the use of convex relaxations, such as SOCP and SDP relaxations, for nonconvex problems. The idea is to get tight bounds from these relaxations and use them inside a branch-and-bound framework. Instead of studying general nonconvex optimization problems, we focus on several special cases: (1) quadratic programming (QP) problems, i.e., f_0 is quadratic and its Hessian is not positive semidefinite, and all other function are linear; (2) 0-1 integer programming problems: all functions are linear and some components of the variable x are binary.

The thesis is not limited to the construction of convex relaxations, but also studies a modern technique for solving large-scale convex optimization problems and their applications.

The remainder of the thesis is organized as follows. Chapter 2 investigates an SOCP-based sequential relaxation technique for 0-1 integer programs; Chapter 5 develops a first-order method based on a smoothing technique to solve a class of large-scale linear programs; Chapter 3 studies semidefinite relaxations of nonconvex, box-constrained quadratic programming (BoxQP); related to Chapter 3, Chapter 4 studies how to combine a finite branch and bound method and a new SDP relaxation to solve general nonconvex QP; In Chapter 5, we investigate a first-order method that is suitable for a class of large-scale linear programming problems and apply it to machine learning; finally, Chapter 6 summarizes the contribution of this thesis, and outlines future research.

The following four paragraphs motivate and outline Chapters 2 to 5.

1.1 A p-Cone Sequential Relaxation Procedure for 0-1 Integer Programs

Beyond the basic linear programming (LP) relaxation of the feasible set of 0-1 integer programs, many authors have considered general techniques for achieving tighter relaxations (Gomory, 1963; Sherali and Adams, 1990; Lovász and Schrijver, 1991; Balas et al., 1993; Kojima and Tunçel, 2000a,b; Lasserre, 2001; Parrilo, 2003; Bienstock and Zuckerberg, 2004). One recurring theme is to lift the feasible set

of 0-1 integer programs into a higher dimensional space, where a convex relaxation is constructed, and then to project this relaxation back into the original space of variables, thus obtaining a new relaxation, which is possibly tighter than the basic LP relaxation. The choice of lifting and relaxation determines the strength of the new relaxation. In all previous works, LP and SDP relaxations have been used in the higher dimensional space.

With the development of second-order cone programming (SOCP) both in theory and practical implementation, one may wonder if SOCP or more generally p-order cone programming ($1 \leq p \leq \infty$) can be used in a sequential relaxation technique, which has its own theoretical and practical advantages. In this chapter, we address this question by introducing a sequential relaxation technique based on p-order cone programming. We prove that our technique generates the convex hull of 0-1 solutions asymptotically. Although our technique does not converge in finite iterations for finite p, we prove that for $p = 2$ solving the relaxation obtained by applying the technique one time enjoys a better theoretical iteration complexity than that of other p, including $p = \infty$. In addition, we show that our method generalizes and subsumes several existing methods. For example, when $p = \infty$, our method corresponds to the well-known procedure of Lovász and Schrijver (Lovász and Schrijver (1991)) based on linear programming. Computational considerations of our technique are also discussed.

1.2 Relaxing the Optimality Conditions of Box QP

In finding a global solution to box-constrained nonconvex quadratic program, Vandenbussche and Nemhauser (2005b,a) branch on first-order KKT conditions and solve an LP relaxation at each node of the branch-and-bound tree. Extending the first-order KKT-branching to general quadratic programming, Burer and Vandenbussche (2006a, 2008) develop a finite branch-and-bound scheme where a SDP relaxation is solved at each node. Chapter 3 of this thesis also investigates SDP relaxations of nonconvex box-constrained quadratic programs and their use in a branch-and-bound scheme. However, it differs from the previous works in two aspects: (i) the SDP relaxations are constructed based on the second-order KKT conditions; (ii) the branching method does not branch on the KKT conditions but subdivides the feasible region instead.

First, we derive the first- and second-order necessary optimality conditions of BoxQP and prove an equivalent form of the second-order condition that turns out to be critical in constructing the semidefinite relaxation. Second, we linearize the quadratic terms in the first- and second-order conditions to obtain two semidefinite relaxations (SDP_{12}) and (SDP_2), where the former incorporates both the first- and second-order optimality conditions while the latter only incorporates the second-order condition. We then recall a basic semidefinite relaxation (SDP_0) due to Shor and establish that (SDP_0), (SDP_{12}) and (SDP_2) are all equivalent. However, in the context of branch-and-bound to determine a global optimal solution, we empirically demonstrate that (SDP_2) is significantly stronger than (SDP_0) in the sense that the number

of nodes required by (SDP_2) in the branch-and-bound tree is less than that required by (SDP_0). An effective branching strategy is also developed for the branch-and-bound scheme.

1.3 Combining Finite Branch-and-Bound with Doubly Nonnegative Relaxations for QP

Burer (2006) has recently shown that a large class of quadratic programming problems can be modeled by so-called completely positive programs (CPP), which minimize a linear function over the convex cone of completely positive matrices subject to linear constraints. Relaxing the completely positive matrices naturally leads to semidefinite relaxations, referred to as doubly nonnegative relaxations. As a following work, Burer (2010) presents a computationally efficient algorithm for solving the doubly nonnegative relaxations. This algorithm can be used as a sub-routine inside a branch-and-bound scheme to globally solve QPs.

The goal of this work is to integrate the afore-mentioned sub-routine with a finite branch-and-bound method (Burer and Vandenbussche (2008)) to solve a general QP with linear constraints only. In order to combine the algorithm with the finite branch-and-bound together, we first reformulate the general QP such that it explicitly incorporates its first-order KKT system into its constraints. By doing this, the QP we consider can be modeled by CPP, which explicitly models the complementarity constraint. In addition, formulating the KKT system allows us to do finite branching on the complementarity condition. One key technical issue in the reformulation is

to bound the dual variables because the sub-routine used to solve doubly nonnegative relaxations requires a finite bound on the input variables. We test the finite branch-and-bound method obtained this way on 77 GLOBALIB problems and several BoxQP problems. The computational results indicate potential as well as room for improvement for our method.

1.4 A First-Order Smoothing Technique for a Class of Large-Scale LPs

Chapter 5 of this thesis deals with a class of LPs instead of nonconvex problems. Modern LP solvers are so sophisticated and robust that it is almost always the case that a LP can be solved as long as there is sufficient computer memory. However, a lot of LPs arising in machine learning and data mining are so large that they cannot be solved by off-the-shelf LP solvers because of memory limitations of the computer. Chapter 5 investigates a smoothing technique that is extremely memory efficient and thus is suitable for large-scale problems.

The smoothing technique we use was first proposed in Nesterov (2005b,a) and has been applied, for example, to solve LPs, SDPs and other convex nonsmooth problems. The LP we consider has an unbounded feasible set and cannot be directly cast as the structured convex problem required by the smoothing technique. Our work features a transformation of the unbounded feasible set into a bounded one. This transformation requires a parameter that bounds the optimal value from above. As a result, the iteration complexity of solving the LP depends on the parameter. We

design a scheme that dynamically updates the parameter to speed up the convergence.

We apply the smoothing technique to two machine learning problems: one is a linear programming based ranking problem (Ataman (2007)), and the other is 1-norm SVMs (Zhu et al. (2003); Mangasarian (2006)). For the ranking problem, we compare the smoothing technique with the subgradient method employed by Ataman (2007); for 1-norm SVMs, we compare it with the generalized Newton method developed by Mangasarian (2006) on large-scale 1-norm SVMs. The smoothing technique shows faster convergence than the other methods and is more robust than the generalized Newton method.

Portions of this thesis are joint work with Dr. Samuel Burer.

CHAPTER 2
A P-CONE SEQUENTIAL RELAXATION PROCEDURE FOR 0-1 INTEGER PROGRAMS

This chapter has been published in Burer and Chen (2009a).

2.1 Introduction

Consider solving the 0-1 integer program

$$\begin{aligned} \min \ & c^T x \\ \text{s.t.} \ & a_i^T x \le b_i \quad \forall \ i = 1, \ldots, m \\ & x \in \{0,1\}^n. \end{aligned} \tag{2.1}$$

Beyond the basic linear programming (LP) relaxation P of the feasible set of (2.1), many authors have considered general techniques for achieving tighter relaxations (Gomory, 1963; Sherali and Adams, 1990; Lovász and Schrijver, 1991; Balas et al., 1993; Kojima and Tunçel, 2000a,b; Lasserre, 2001; Parrilo, 2003; Bienstock and Zuckerberg, 2004). One recurring theme is to lift the feasible set of (2.1) into a higher dimensional space, where a convex relaxation is constructed, and then to project this relaxation back into the original space of variables, thus obtaining a relaxation P^1, which is possibly tighter than P. The choice of lifting and relaxation determines the strength of P^1. In all previous works, LP and semidefinite (SDP) relaxations have been used in the higher dimensional space. Kim and Kojima Kim and Kojima (2003) suggest a modification of the SDPs which only enforces positive semidefiniteness on 2×2 principal submatrices, which can be modeled with second-order cone

programming (SOCP) (see also Kim and Kojima (2001); Kim et al. (2003)).

In Lovász and Schrijver (1991), Balas et al. (1993), and Kojima and Tunçel (2000a,b), the idea of sequential relaxation is also introduced. Stated simply, the idea is to repeat the lift-and-project procedure on the tighter relaxation P^1, thus obtaining an even tighter relaxation P^2. If P^k denotes the k-th relaxation obtained inductively, a fundamental question is whether $\{P^k\}$ converges to P^{01}, the convex hull of the feasible set of (2.1). In each study referenced above, the answer is positive; in fact, $P^k = P^{01}$ for all $k \geq n$. Computationally, one can typically optimize over P^k in polynomial time as long as k is constant with respect to n. Kojima and Tunçel (2000a,b) apply their techniques to a much broader class of quadratically constrained problems and show asymptotic convergence to the convex hull of solutions.

Sherali and Adams (1990), Lasserre (2001), and Parrilo (2003) do not explicitly employ the idea of sequential relaxation. Rather, they lift to ever higher dimensions before projecting, a technique which is analogous to sequential relaxation. Here, too, the authors show that lifting to a finite dimension (dependent in some manner on n) achieves P^{01}. Similar to the work of Kojima and Tunçel (2000a,b), these authors' techniques can be applied to more general problem classes, but one may have to lift "infinitely" to achieve the convex hull of solutions.

Although these lift-and-project procedures are very powerful theoretically, they present significant computational challenges, even after a single iteration. One must deal with more variables in the higher dimensional space as well as additional constraints introduced by lifting. For example, after one iteration of the LP-based

procedure of Balas et al. (1993), the resulting LP contains $2n$ variables and $2m+1$ constraints assuming that the constraints $a_i^T x \leq b_i$ already imply the bounds $0 \leq x_j \leq 1$. The procedure of Lovász and Schrijver (1991) requires even more variables and constraints. After one iteration, their LP-based procedure has $\mathcal{O}(n^2)$ variables and $\mathcal{O}(nm)$ linear constraints, and their SDP-based procedure contains an additional semidefinite constraint on an order $n \times n$ matrix. In both of these particular cases, computational progress has been achieved by exploiting the structure of constraints generated by these procedures (Balas and Perregaard, 2003; Burer and Vandenbussche, 2006b).

The purpose of this chapter is to explore p-order cone programming (POCP) relaxations, which include second-order cone programming (SOCP) relaxations when $p = 2$. Our interest in POCP arises from the fact that POCP is becoming a well-understood tool in convex optimization (Xue and Ye, 2000; Andersen et al., 2002; Glineur and Terlaky, 2004; Krokhmal and Soberanis, 2008). Moreover, there are currently several high quality implementations for SOCP (Tütüncü et al., 2001; Sturm, 1999; MOSEK, Inc., 2007), and in fact POCP can be formulated exactly via SOCP (Ben-Tal and Nemirovski, 2001; Alizadeh and Goldfarb, 2003; Krokhmal and Soberanis, 2008). Our hope is that, by introducing POCP relaxations, we might discover new lift-and-project procedures that have their own theoretical and computational advantages.

In this chapter, we introduce a family of lift-and-project procedures parameterized by $p \in [1, \infty]$ and prove that each asymptotically yields P^{01}, the convex hull

of 0-1 solutions (Theorem 2.4.1). A feature of this family of procedures is the ability to lift and project with respect to different subsets of variables at different iterations. Although we do not achieve finite convergence in general so that our procedure is weaker than existing methods in this sense, we do observe theoretical advantages. In particular, the theoretical iteration complexity of solving the POCP relaxations via interior-point methods is minimized when $p = 2$ at which the iteration complexity is an order of magnitude less than solving existing LP and SDP relaxations (Corollary 2.3.6). In addition, our family of procedures unifies existing approaches. For example, when $p = \infty$, we recover the LP-based procedure of Lovász and Schrijver (1991).

In Section 2.2, we describe the basic p-order cone lift-and-project procedure and give an alternate derivation of it. This is just one iteration of the entire sequential relaxation approach, which is described in Section 2.4. We then compare and contrast our procedure with three existing approaches: Lovász and Schrijver (1991), Kojima and Tunçel (2000a,b), and Balas et al. (1993). In particular, we point out that our method includes the LP based lift-and-project procedure of Lovász and Schrijver (1991) and the relaxation of Balas et al. (1993) as special cases.

In Section 2.3, we study fundamental properties of the p-order cone procedure. In particular, we examine duality properties and two types of monotonicity. For example, one monotonicity property establishes that the strength of the procedure increases with p, so that the strength is maximized at $p = \infty$. We also study the iteration complexity of solving the resultant p-order cone relaxation via interior-point methods, where it is shown that the lowest iteration complexity is obtained for $p = 2$.

Following these results is the main technical result of the chapter (Theorem 2.3.15): the p-order cone procedure, when applied to a generic compact, convex set P, cuts off all fractional extreme points of P. Theorem 2.3.15 is motivated and proven in three steps. We first show the result holds when P is a polytope, which is the easiest case. Then we establish the result when P is a ball, which finally allows us to prove the theorem for general P.

In Section 2.4, we describe the sequential relaxation approach based on repetition of the p-order cone procedure and prove that it generates the convex hull of 0-1 solutions asymptotically (Theorem 2.4.1). An explicit example is provided to show that, in general, the iterated procedure may indeed require an infinite number of repetitions to converge.

In Section 2.5, we consider computational issues associated with the p-cone sequential relaxation procedure — particularly with optimizing over the first-iteration relaxation. We compare the SOCP relaxation ($p = 2$, lowest theoretical complexity) to the LP relaxation of Lovász-Schrijver ($p = \infty$, tightest relaxation). As it turns out, even though the SOCP relaxation enjoys a much lower theoretical iteration complexity, the LPs solve more quickly and produce better bounds.

Finally, in Section 2.6, we conclude with final remarks.

2.2 Relaxation Procedure and Comparisons

In this section, we first introduce notation and terminology and then formally describe our p-cone lift-and-project procedure, comparing and contrasting it with the

methods of Lovász and Schrijver (1991), Kojima and Tunçel (2000a,b), and Balas et al. (1993).

2.2.1 Notation and terminology

$\Re^n$ refers to n-dimensional Euclidean space, and $\Re^{n\times n}$ is the set of real, $n \times n$ matrices. We let $e_i \in \Re^n$ represent the i-th unit coordinate vector and $e \in \Re^n$ represent the vector of all ones. We denote by $[n]$ the set $\{1, 2, \ldots, n\}$. For $\mathcal{J} \subseteq [n]$ an index set, $x_{\mathcal{J}} \in \Re^{|\mathcal{J}|}$ is defined as the vector composed of entries of x that are indexed by $\mathcal{J}$. Similarly, given a matrix $A \in \Re^{n\times n}$, $A_{\mathcal{J}\cdot}$ represents the $|\mathcal{J}| \times n$ matrix composed of the rows of A indexed by $\mathcal{J}$. Diag(x) denotes the diagonal matrix with diagonal x, and $A \succeq 0$ means that the matrix A is symmetric positive semidefinite; its dimension will be clear from context. Finally, given a set S defined over variables (x, y), $\text{proj}_x(S)$ denotes the projection of S onto the coordinates in x.

For $p \geq 1$, the usual p-norm on $\Re^n$ is defined as

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

We also define when $p = \infty$ $\|x\|_\infty := \max_{i=1}^n |x_i|$. Associated with $p \in [1, \infty]$ is q such that $p^{-1} + q^{-1} = 1$. Both the p-norm and q-norm give rise to closed, convex cones in $\Re^{1+n}$:

$$\mathcal{K}_p := \{(x_0, x) \in \Re^{1+n} : x_0 \geq \|x\|_p\}$$

$$\mathcal{K}_q := \{(y_0, y) \in \Re^{1+n} : y_0 \geq \|y\|_q\}.$$

It is well known that $\mathcal{K}_q$ is dual to $\mathcal{K}_p$, i.e.,

$$\mathcal{K}_q = \{(y_0, y) \in \Re^{1+n} : y_0 x_0 + y^T x \geq 0 \ \ \forall\ (x_0, x) \in \mathcal{K}_p\},$$

which is written as $K_q = K_p^*$; see for example page 51 of Boyd and Vandenberghe (2004). Important special cases occur when $p = 2$ or $p = \infty$. When $p = 2$, $K_p = K_q$, i.e., K_p and K_q are self-dual. When $p = \infty$, $q = 1$, and both K_p and K_q are polyhedral cones.

2.2.2 Relaxation procedure

For the purposes of generality particularly with regards to Section 2.4, we consider a slightly different form of the feasible set of the integer program (2.1), the only difference being that the linear constraints are indexed by an arbitrary set $\mathcal{I}$ (possibly infinite):

$$F := \left\{ x \in \{0,1\}^n : a_i^T x \leq b_i \ \ \forall\ i \in \mathcal{I} \right\}.$$

This semi-infinite representation for F, as opposed to a finite one, does not affect the theoretical exposition of the p-cone procedure though it may pose computational issues. Associated with F is its continuous convex relaxation

$$P := \{x \in \Re^n : a_i^T x \leq b_i \ \forall\ i \in \mathcal{I}\},$$

obtained by relaxing the integrality requirements on x. We assume P is contained in $[0,1]^n$, for example, by including bounds on x via explicit constraints $a_i^T x \leq b_i$. So P is compact convex.

We wish to generate a compact convex relaxation of F, which is tighter than P. Unless stated otherwise, we assume throughout this section the fixed choice of $p \in [1, \infty]$ and $\emptyset \neq \mathcal{J} \subseteq [n]$. We will denote the proposed convex relaxation as

$N_{(p,\mathcal{J})}(P)$, or more often simply as $N(P)$. Defining

$$P^{01} := \text{conv}(F),$$

our goal is to produce $N(P)$ such that $P^{01} \subseteq N(P) \subseteq P$.

Our first step is to lift F into a higher dimensional space. We will make use of the following simple key geometric proposition.

Proposition 2.2.1. *Define* $r := \sqrt[p]{|\mathcal{J}|}/2$ *and* $d := e/2 \in \Re^{|\mathcal{J}|}$. *Then* $x_{\mathcal{J}} \in \{0,1\}^{|\mathcal{J}|}$ *implies* $\|x_{\mathcal{J}} - d\|_p \leq r$.

It is important to keep in mind that r depends on p and $|\mathcal{J}|$ and that d depends on $|\mathcal{J}|$. This proposition establishes the existence of a family of p-balls circumscribing the integer points $\{0,1\}^{|\mathcal{J}|}$. In fact, $p' \geq p$ implies that the p'-ball is contained in the p-ball (see Proposition 2.3.9), with $p = \infty$ corresponding to the convex hull $[0,1]^{|\mathcal{J}|}$ of the integer points.

Using Proposition 2.2.1, F can be rewritten redundantly as

$$F = \{x \in \Re^n : x = x^2,\ a_i^T x \leq b_i \ \ \forall\ i \in \mathcal{I},\ \|x_{\mathcal{J}} - d\|_p \leq r\}.$$

We note that

$$\left.\begin{array}{l} a_i^T x \leq b_i \\ \|x_{\mathcal{J}} - d\|_p \leq r \end{array}\right\} \quad \Longrightarrow \quad \left\|(b_i - a_i^T x)(x_{\mathcal{J}} - d)\right\|_p \leq r(b_i - a_i^T x), \qquad (2.2)$$

which in turn implies

$$F = \left\{x \in \Re^n : x = x^2,\ \left\|b_i x_{\mathcal{J}} - x_{\mathcal{J}} x^T a_i - (b_i - a_i^T x)d\right\|_p \leq r(b_i - a_i^T x) \ \ \forall\ i \in \mathcal{I}\right\}$$

since $b_i - a_i^T x$ is kept nonnegative. Next, introducing an $n \times n$ matrix variable X satisfying $X = xx^T$ and defining

$$\hat{F} := \left\{ (x, X) \in \Re^n \times \Re^{n \times n} : \begin{array}{c} X = xx^T \qquad \mathrm{diag}(X) = x \\ \left\| b_i x_{\mathcal{J}} - X_{\mathcal{J}.} a_i - (b_i - a_i^T x) d \right\|_p \leq r(b_i - a_i^T x) \ \ \forall\, i \in \mathcal{I} \end{array} \right\}$$

we see that $F = \mathrm{proj}_x(\hat{F})$, i.e., $\hat{F}$ is the lifted version of F. In addition, dropping the nonconvex constraint $X = xx^T$ from $\hat{F}$, we obtain a convex relaxation of $\hat{F}$:

$$\hat{P} := \left\{ (x, X) \in \Re^n \times \Re^{n \times n} : \begin{array}{c} \mathrm{diag}(X) = x \\ \left\| b_i x_{\mathcal{J}} - X_{\mathcal{J}.} a_i - (b_i - a_i^T x) d \right\|_p \leq r(b_i - a_i^T x) \ \ \forall\, i \in \mathcal{I} \end{array} \right\}.$$

Finally, we define $N(P)$ as the projection of $\hat{P}$:

$$N(P) := \mathrm{proj}_x(\hat{P})$$

The desired property of $N(P)$ is immediate.

Proposition 2.2.2. *$P^{01} \subseteq N(P) \subseteq P$.*

Proof. $P^{01} \subseteq N(P)$ by construction. Moreover, the definition of $\hat{P}$ implies that every $x \in N(P)$ satisfies $r(b_i - a_i^T x) \geq 0$ for all $i \in \mathcal{I}$. Since $r > 0$, this implies $x \in P$. So $N(P) \subseteq P$. □

Using the assumption that the constraints $a_i^T x \leq b_i$ include the bounds $0 \leq x_j \leq 1$ explicitly and the fact that $N(P) \subseteq P$ is bounded, one can see from the definition of $\hat{P}$ that $\mathrm{proj}_{(x, X_{\mathcal{J}.})}(\hat{P})$ is bounded and hence compact convex. Since the rows $X_{j.}$ for $j \notin \mathcal{J}$ are not constrained in $\hat{P}$ except for the entries X_{jj}, it is also clear that

$$N(P) = \mathrm{proj}_x \left(\mathrm{proj}_{(x, X_{\mathcal{J}.})}(\hat{P}) \right).$$

Since the projection of compact convex sets are compact convex, we conclude that $N(P)$ is compact convex.

Just like P, $N(P)$ has its own semi-infinite outer description, which can be the basis of lift-and-project applied to $N(P)$ itself. This will be the main idea behind the iterated procedure of Section 2.4.

As an example, consider the feasible set

$$F = \left\{ x \in \{0,1\}^2 : \begin{array}{c} -x_1 \leq 0 \\ -x_2 \leq 0 \\ x_1 + 2x_2 \leq 2.5 \\ 3x_1 + x_2 \leq 2.5 \end{array} \right\} = \Big\{ (0,0), (0,1) \Big\} \tag{2.3}$$

so that

$$P^{01} = \Big\{ (0, x_2) \in \Re^2 : 0 \leq x_2 \leq 1 \Big\}.$$

Let $\mathcal{J} = \{1, 2\}$. In Figure 2.1, we illustrate the p-cone procedure by depicting the four sets

$$P \supseteq N_{(1,\mathcal{J})}(P) \supseteq N_{(2,\mathcal{J})}(P) \supseteq N_{(\infty,\mathcal{J})}(P)$$

containing P^{01}. Figure 2.1 was drawn by determining a collection of points on the boundary of each set via a collection of linesearch procedures. For each of the four sets, a single linesearch started at $(0, 0)$ and moved into the first quadrant at an angle $\theta \in [0, \pi/2]$. The point on the boundary was precisely the point where the linesearch left the set. The linesearch was repeated for a sufficiently fine grid on $[0, \pi/2]$ for each of the four sets.

Recall that P is the continuous LP relaxation of F; in the figure, it is the largest set. The next largest is $N_{(1,\mathcal{J})}$, a polyhedral set since $p = 1$. $N_{(2,\mathcal{J})}$ is the projection of a second-order cone set and hence has a curved boundary. Finally, the

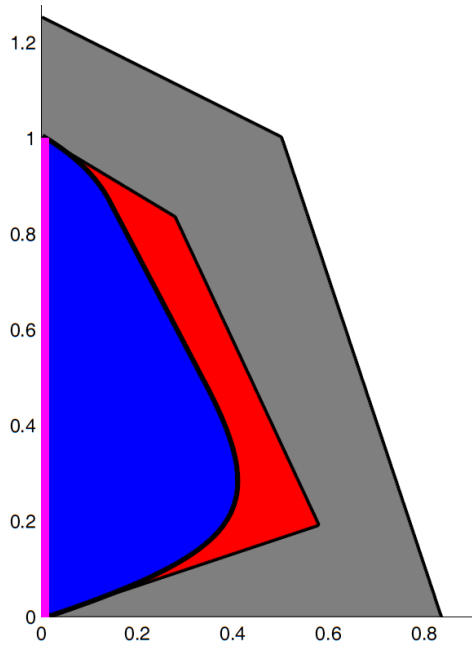

Figure 2.1: The four sets $P \supseteq N_{(1,\mathcal{J})}(P) \supseteq N_{(2,\mathcal{J})}(P) \supseteq N_{(\infty,\mathcal{J})}(P)$ relative to the example feasible set F in (2.3), where $\mathcal{J} = \{1, 2\}$. Note that $N_{(\infty,\mathcal{J})}(P) = P^{01}$ in this example.

depicted line segment between $(0,0)$ and $(0,1)$ is $N_{(\infty,\mathcal{J})}$, which equals P^{01} in this example.

2.2.3 A different derivation

In the derivation of $N(P)$, we have relied on the implication (2.2), which can be thought of as replacing two constraints by their product respecting nonnegativity. We now show that one can obtain an alternate representation of the right-hand side of (2.2) via an alternate representation of $\|x_{\mathcal{J}} - d\|_p \leq r$. Multiplication of constraints is still the key idea.

Consider $\mathcal{K}_p$ and $\mathcal{K}_q$ as described in Section 2.2.1. Because $\mathcal{K}_q = \mathcal{K}_p^*$, it holds that

$$\begin{aligned} \|x_{\mathcal{J}} - d\|_p \leq r \quad &\Longleftrightarrow \quad (r, x_{\mathcal{J}} - d) \in \mathcal{K}_p \\ &\Longleftrightarrow \quad vr + u^T(x_{\mathcal{J}} - d) \geq 0 \quad \forall\ (v,u) \in \mathcal{K}_q. \end{aligned} \tag{2.4}$$

Proposition 2.2.3. *For a given $x \in \Re^n$, the right-hand side of (2.2) holds if and only if*

$$(b_i - a_i^T x)\left(vr + u^T(x_{\mathcal{J}} - d)\right) \geq 0 \quad \forall\ (v,u) \in \mathcal{K}_q. \tag{2.5}$$

Proof. ($\Rightarrow$): The right-hand side of (2.2) implies $b_i - a_i^T x \geq 0$. If $b_i - a_i^T x = 0$, then clearly (2.5) holds. On the other hand, if $b_i - a_i^T x > 0$, then dividing the right-hand side of (2.2) by $b_i - a_i^T x$ shows $\|x_{\mathcal{J}} - d\|_q \leq r$, which in turn implies $vr + u^T(x_{\mathcal{J}} - d) \geq 0$ for all $(v,u) \in \mathcal{K}_q$ by (2.4). Now multiplying with $b_i - a_i^T x$ implies (2.5).

($\Leftarrow$): If $b_i - a_i^T x = 0$, then the right-hand side of (2.2) holds trivially. On the

other hand, if $b_i - a_i^T x \neq 0$, then for any nonzero u the two inequalities

$$(b_i - a_i^T x)\left(\|u\|_q r + u^T(x_{\mathcal{J}} - d)\right) \geq 0$$

$$(b_i - a_i^T x)\left(\|u\|_q r - u^T(x_{\mathcal{J}} - d)\right) \geq 0$$

together imply $b_i - a_i^T x > 0$. As a consequence, $vr + u^T(x_{\mathcal{J}} - d) \geq 0$ for all $(v, u) \in \mathcal{K}_q$, which means $\|x_{\mathcal{J}} - d\|_q \leq r$ by (2.4), which in turn implies the right-hand side of (2.2). □

An immediate consequence of Proposition 2.2.3 is that $\hat{P}$ defined in the derivation of $N(P)$ can be equivalently expressed using the semi-infinite collection of inequalities

$$(b_i - a_i^T x)(vr - u^T d) + b_i u^T x_{\mathcal{J}} - u^T X_{\mathcal{J}.} a_i \geq 0 \qquad \forall \;\; (i, (v, u)) \in \mathcal{I} \times \mathcal{K}_q,$$

thus providing an equivalent definition of $N(P)$.

2.2.4 Comparison with existing approaches

In the derivation of $N(P)$, we did not use the full strength of the relationship $X = xx^T$ in the relaxation $\hat{P}$. In particular, we could have also imposed in $\hat{P}$ the following two convex conditions, which are implied by $X = xx^T$:

$$X = X^T \qquad \text{and} \qquad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0. \tag{2.6}$$

If we had imposed these, then $N(P)$ would be even tighter. However, we have purposely not imposed them because they are not necessary for the theoretical convergence of the iterated procedure in Section 2.4. In practice, one would certainly want

to impose as many constraints that can be handled efficiently. In particular, imposing symmetry $X = X^T$ can be useful to eliminate variables.

We mention the conditions (2.6) here because they facilitate comparison with existing lift-and-project methods in the following subsections.

2.2.4.1 Lovász-Schrijver

The LP-based approach of Lovász and Schrijver (1991) is derived like ours except that the following lifted and relaxed sets serve in the place of our $\hat{F}$ and $\hat{P}$:

$$\hat{F}_{\text{LS}} = \left\{ (x, X) \in \Re^n \times \Re^{n \times n} : \begin{array}{c} X = xx^T \qquad \text{diag}(X) = x \\ (b_i - a_i^T x) x_k \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n] \\ (b_i - a_i^T x)(1 - x_k) \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n] \end{array} \right\}$$

$$\hat{P}_{\text{LS}} = \left\{ (x, X) \in \Re^n \times \Re^{n \times n} : \begin{array}{c} X = X^T \qquad \text{diag}(X) = x \\ b_i x_k - X_{k\cdot} a_i \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n] \\ (b_i - a_i^T x) - (b_i x_k - X_{k\cdot} a_i) \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n] \end{array} \right\}.$$

We next establish relations between $\hat{F}_{\text{LS}}$ and $\hat{F}$, $\hat{P}_{\text{LS}}$ and $\hat{P}$.

Proposition 2.2.4. *Let $p = \infty$ and $\mathcal{J} = [n]$. If the p-cone lift-and-project procedure also enforces the symmetry condition of (2.6), then $\hat{F} = \hat{F}_{\text{LS}}$ and $\hat{P} = \hat{P}_{\text{LS}}$.*

Proof. Note that $r = 1/2$ with $p = \infty$ and $\mathcal{J} = [n]$. It suffices to show that the conditions

$$\left\| b_i x - x x^T a_i - (b_i - a_i^T x) d \right\|_p \leq r (b_i - a_i^T x) \ \ \forall\ i \in \mathcal{I}$$

of $\hat{F}$ are equivalent to the conditions

$$\begin{array}{c} (b_i - a_i^T x) x_k \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n] \\ (b_i - a_i^T x)(1 - x_k) \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n] \end{array} \tag{2.7}$$

of $\hat{F}_{\text{LS}}$. By Proposition 2.2.3, the conditions of $\hat{F}$ can be replaced by

$$(b_i - a_i^T x) \left(v r + u^T (x - d) \right) \geq 0 \quad \forall\ (i, (v,u)) \in \mathcal{I} \times \mathcal{K}_1.$$

Since $\mathcal{K}_1$ is finitely generated by $\{(1, \pm e_1), \ldots, (1, \pm e_n)\}$, we obtain

$$(b_i - a_i^T x)\left(r + e_k^T(x - d)\right) \geq 0 \quad \forall\ (i,k) \in \mathcal{I} \times [n]$$

$$(b_i - a_i^T x)\left(r - e_k^T(x - d)\right) \geq 0 \quad \forall\ (i,k) \in \mathcal{I} \times [n],$$

which reduce to (2.7), as desired. □

The next theorem follows directly.

Theorem 2.2.5. *Let $p = \infty$ and $\mathcal{J} = [n]$, and suppose the p-cone lift-and-project procedure enforces the symmetry condition of (2.6). Then the p-cone procedure reduces to the LP-based Lovász-Schrijver lift-and-project procedure.*

Lovász and Schrijver (1991) also proposed an SDP-based procedure, which enforces the semidefiniteness condition of (2.6) in $\hat{P}_{\text{LS}}$. Just as the p-cone procedure with symmetry replicates the LP-based Lovász-Schrijver procedure, the p-cone procedure with (2.6) replicates the SDP-based Lovász-Schrijver procedure.

Theorem 2.2.6. *Let $p = \infty$ and $\mathcal{J} = [n]$, and suppose the p-cone lift-and-project procedure enforces the symmetry and semidefiniteness conditions of (2.6). Then the p-cone procedure reduces to the SDP-based Lovász-Schrijver lift-and-project procedure.*

2.2.4.2 Kojima-Tunçel

Kojima and Tunçel (2000a,b) present their method as a direct extension of the approach of Lovász and Schrijver (1991) to general quadratic optimization problems. Their approach essentially reduces to that of Lovász-Schrijver in the case of 0-1 integer programming — with one important difference, which we explain next. This

difference, in particular, will have relevance to our discussion and proofs in Section 2.4.

As discussed in the previous subsection, the Lovász-Schrijver approach is based on lifting with respect to the collection of constraints

$$(b_i - a_i^T x)x_k \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n]$$

$$(b_i - a_i^T x)(1 - x_k) \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n]$$

In contrast, Kojima and Tunçel (2000a) (see Section 6, page 767, third full paragraph) lift with respect to the larger, extended collection

$$(b_i - a_i^T x)x_k \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n]$$

$$(b_i - a_i^T x)(1 - x_k) \geq 0 \ \ \forall\ (i,k) \in \mathcal{I} \times [n]$$

$$(b_i - a_i^T x)(b_h - a_h^T x) \geq 0 \ \ \forall\ (i,h) \in \mathcal{I} \times \mathcal{I}$$

$$x_j x_k \geq 0 \ \ \forall\ (j,k) \in [n] \times [n]$$

$$x_j(1 - x_k) \geq 0 \ \ \forall\ (j,k) \in [n] \times [n]$$

$$(1 - x_j)(1 - x_k) \geq 0 \ \ \forall\ (j,k) \in [n] \times [n],$$

which reduces to lifting with respect to

$$(b_i - a_i^T x)(b_h - a_h^T x) \geq 0 \ \ \forall\ (i,h) \in \mathcal{I} \times \mathcal{I},$$

since P implies the constraints $0 \leq x_k \leq 1$ by assumption. For more insight on this point, please refer to Section 2.3.3 for a discussion on the monotonicity properties of lift-and-project procedures.

This broader lifting guarantees the Kojima-Tunçel approach is at least as strong as the Lovász-Schrijver approach and at least as strong as our approach for $p = \infty$ and $\mathcal{J} = [n]$.

We remark that Lovász and Schrijver (1991) did consider the broader lifting of Kojima and Tunçel (2000a,b) but chose not to focus on it for algorithmic reasons. This point is explained in detail by Kojima and Tunçel (2000a,b).

2.2.4.3 Balas-Ceria-Cornuéjols

The approach of Balas et al. (1993) can also be viewed as a special case of our approach. The authors choose a single index j and then apply the lift-and-project procedure outlined above in Section 2.2.2, replacing $\hat{F}$ and $\hat{P}$ by the following:

$$\hat{F}_{\text{BCC}} = \left\{ (x, X) \in \Re^n \times \Re^{n \times n} : \begin{array}{c} X = xx^T \qquad \text{diag}(X) = x \\ (b_i - a_i^T x) x_j \geq 0 \ \ \forall\, i \in \mathcal{I} \\ (b_i - a_i^T x)(1 - x_j) \geq 0 \ \ \forall\, i \in \mathcal{I} \end{array} \right\}$$

$$\hat{P}_{\text{BCC}} = \left\{ (x, X) \in \Re^n \times \Re^{n \times n} : \begin{array}{c} \text{diag}(X) = x \\ b_i x_j - X_{j\cdot} a_i \geq 0 \ \ \forall\, i \in \mathcal{I} \\ (b_i - a_i^T x) - (b_i x_j - X_{j\cdot} a_i) \geq 0 \ \ \forall\, i \in \mathcal{I} \end{array} \right\}.$$

We point out two important details. First, $\hat{P}_{\text{BCC}}$ does not enforce the symmetry condition $X = X^T$ of (2.6). Second, because all rows $X_{k\cdot}$ for $k \neq j$ are unconstrained except for the equation $X_{kk} = x_k$, $\hat{P}_{\text{BCC}}$ may be reduced to

$$\hat{P}_{\text{BCC}} = \left\{ (x, y) \in \Re^n \times \Re^n : \begin{array}{c} y_j = x_j \\ b_i x_j - a_i^T y \geq 0 \ \ \forall\, i \in \mathcal{I} \\ (b_i - a_i^T x) - (b_i x_j - a_i^T y) \geq 0 \ \ \forall\, i \in \mathcal{I} \end{array} \right\}$$

without affecting the projection onto x.

We claim that the Balas-Ceria-Cornuéjols approach is a special case of our method with $\mathcal{J} = \{j\}$ and arbitrary p.

Proposition 2.2.7. *Let $p \in [1, \infty]$ and $\mathcal{J} = \{j\}$ for some fixed index j. It holds that $\hat{F}_{\text{BCC}} = \hat{F}$ and $\hat{P}_{\text{BCC}} = \hat{P}$.*

Proof. It suffices to show that the conditions

$$\left\| b_i x_j - x_j x^T a_i - (b_i - a_i^T x) d \right\|_p \leq r(b_i - a_i^T x) \quad \forall\, i \in \mathcal{I} \tag{2.8}$$

of $\hat{F}$ are equivalent to the conditions

$$\begin{aligned} (b_i - a_i^T x) x_j &\geq 0 \quad \forall\, i \in \mathcal{I} \\ (b_i - a_i^T x)(1 - x_j) &\geq 0 \quad \forall\, i \in \mathcal{I} \end{aligned}$$

of $\hat{F}_{\text{BCC}}$. Noting that $r = 1/2$ and $d = 1/2$ and that the p-norm is applied to a scalar in this case, (2.8) can be rewritten as

$$\left| (b_i - a_i^T x) \left(x_j - \frac{1}{2} \right) \right| \leq \frac{1}{2} (b_i - a_i^T x) \quad \forall\, i \in \mathcal{I},$$

which is clearly equivalent to the conditions of $\hat{F}_{\text{BCC}}$. □

We thus have the following theorem.

Theorem 2.2.8. *Suppose $p \in [1, \infty]$ and $\mathcal{J} = \{j\}$ for some fixed index j. Then the p-cone lift-and-project procedure reduces to the LP-based Balas-Ceria-Cornuéjols lift-and-project procedure.*

2.3 Duality, Complexity, Monotonicity, and Fractional Extreme Points

In this section, we examine fundamental properties of the p-cone lift-and-project procedure outlined in Section 2.2.2. The first main result, proved in Section 2.3.2, establishes the theoretical iteration complexity of optimizing over $N_{(p,\mathcal{J})}(P)$.

The second main result, proved below in Section 2.3.4.3, is that $N(P)$ contains no extreme points of P having fractional entries in $\mathcal{J}$, a result which will prove critical in Section 2.4.

Unless stated otherwise, we assume throughout this section that the pair $(p, \mathcal{J})$ is fixed, and we use the short notation $N(P)$ in place of $N_{(p,\mathcal{J})}(P)$. We also assume throughout that $\mathcal{J} = \{1, \ldots, |\mathcal{J}|\}$, i.e., $\mathcal{J}$ specifies the first $|\mathcal{J}|$ variables in x; this is for notational simplicity only.

2.3.1 Duality

Consider the relaxation $\min\{c^T x : x \in N(P)\}$ of the 0-1 integer program $\min\{c^T x : x \in F\}$. Its explicit p-cone representation is

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & \operatorname{diag}(X) = x \\ & \left(r(b_i - a_i^T x), b_i x_{\mathcal{J}} - X_{\mathcal{J}.} a_i - (b_i - a_i^T x) d\right) \in \mathcal{K}_p \qquad \forall \;\; i \in \mathcal{I}, \end{aligned} \tag{2.9}$$

where $(x, X) \in \Re^n \times \Re^{n \times n}$. It can be derived that the associated dual is

$$\begin{aligned} \max \quad & \sum_{i \in \mathcal{I}} b_i (d^T u^i - r v_i) \\ \text{s.t.} \quad & \sum_{i \in \mathcal{I}} \left((d^T u^i - r v_i) a_i + b_i \begin{pmatrix} u^i \\ 0 \end{pmatrix} \right) + \lambda = c \\ & \sum_{i \in \mathcal{I}} \begin{pmatrix} u^i \\ 0 \end{pmatrix} a_i^T + \operatorname{Diag}(\lambda) = 0 \\ & (v_i, u^i) \in \mathcal{K}_q \qquad \forall \;\; i \in \mathcal{I}, \end{aligned} \tag{2.10}$$

where the dual variables are $\lambda \in \Re^n$ and $(v_i, u^i) \in \Re^{1+|\mathcal{J}|}$ for all $i \in \mathcal{I}$. To illustrate the dual derivation without going deep into the details, we prove weak duality between (2.9) and (2.10) in the following proposition.

Proposition 2.3.1. *Suppose $|\mathcal{I}| < \infty$, i.e., P is a polytope. The dual of the p-cone relaxation (2.9) is the q-cone optimization (2.10). In particular, weak duality holds. If, in addition, both (2.9) and (2.10) have interior feasible solutions, then strong duality holds.*

Proof. We prove weak duality to illustrate the dual nature of (2.9) and (2.10). The strong duality result is standard (see Boyd and Vandenberghe (2004)). Let (x, X) be feasible for (2.9) and let $(\lambda, (v_i, u^i))$ be feasible for (2.10). Also, let $s_i := b_i - a_i^T x$. Then

$$
\begin{aligned}
c^T x - \sum_{i\in\mathcal{I}} & b_i (d^T u^i - r v_i) \\
&= \left(\sum_{i\in\mathcal{I}} \left((d^T u^i - r v_i) a_i + b_i \begin{pmatrix} u^i \\ 0 \end{pmatrix} \right) + \lambda \right)^T x - \sum_{i\in\mathcal{I}} b_i (d^T u^i - r v_i) \\
&= \sum_{i\in\mathcal{I}} (r v_i - d^T u^i) s_i + \sum_{i\in\mathcal{I}} b_i \, x^T \begin{pmatrix} u^i \\ 0 \end{pmatrix} + \lambda^T x \\
&= \sum_{i\in\mathcal{I}} \left(r s_i v_i + \left[b_i x_{\mathcal{J}} - s_i d \right]^T u^i \right) + \lambda^T x \\
&= \sum_{i\in\mathcal{I}} \left(r s_i v_i + \left[b_i x_{\mathcal{J}} - s_i d \right]^T u^i \right) - \left(\sum_{i\in\mathcal{I}} \begin{pmatrix} u^i \\ 0 \end{pmatrix} a_i^T \right) \bullet X \\
&= \sum_{i\in\mathcal{I}} \left(r s_i v_i + \left[b_i x_{\mathcal{J}} - s_i d - X_{\mathcal{J}\cdot} a_i \right]^T u^i \right) \\
&\geq \sum_{i\in\mathcal{I}} 0 = 0.
\end{aligned}
$$

□

Related to the primal and dual problems (2.9) and (2.10), we consider the following question and derive a duality result: given $\bar{x} \in P \subseteq [0,1]^n$, is $\bar{x} \in N(P)$? To answer this question, we must determine whether or not the set

$$\left\{ X \in \Re^{n \times n} : \begin{array}{c} \mathrm{diag}(X) = \bar{x} \\ (r\bar{s}_i, b_i \bar{x}_{\mathcal{J}} - X_{\mathcal{J}.} a_i - \bar{s}_i d) \in \mathcal{K}_p \ \ \forall\ i \in \mathcal{I} \end{array} \right\} \tag{2.11}$$

is empty, where $\bar{s}_i := b_i - a_i^T \bar{x}$. This question is in turn related to the following set by Proposition 2.3.2 below:

$$\left\{ \left(\lambda, (v_i, u^i)\right) \in \Re^n \times \mathcal{K}_q^{|\mathcal{I}|} : \begin{array}{c} \sum_{i \in \mathcal{I}} \begin{pmatrix} u^i \\ 0 \end{pmatrix} a_i^T + \mathrm{Diag}(\lambda) = 0 \\ \bar{x}^T \lambda + \sum_{i \in \mathcal{I}} \left(r\bar{s}_i\, v_i + (b_i \bar{x}_{\mathcal{J}} - \bar{s}_i d)^T u^i \right) < 0 \end{array} \right\}. \tag{2.12}$$

Proposition 2.3.2. *Suppose $|\mathcal{I}| < \infty$, i.e., P is a polytope. Let $\bar{x} \in P$, and define $\bar{s}_i := b_i - a_i^T \bar{x}$ for all $i \in \mathcal{I}$. Then (2.11) is empty, i.e., $\bar{x} \notin N(P)$, if and only if (2.12) is nonempty.*

Proof. We first argue that, when feasible, the set

$$\left\{ \left(\lambda, (v_i, u^i)\right) : \sum_{i \in \mathcal{I}} \begin{pmatrix} u^i \\ 0 \end{pmatrix} a_i^T + \mathrm{Diag}(\lambda) = 0 \right\}$$

has nonempty interior with respect to the cones $\mathcal{K}_q \ni (v_i, u^i)$. This follows because we may arbitrarily increase each v_i without affecting the matrix equation. The proposition is now a straightforward application of the conic version of Farkas' lemma (Anderson and Nash, 1987). □

Sets of the form (2.12) will be crucial for our analysis in the remainder of Section 2.3 and in Section 2.4. For ease of reference and in order to facilitate the derivation of various results, we now establish definitions, notations, and basic results related to (2.12). We still assume fixed $(p, \mathcal{J})$.

Let $\hat{A} \in \Re^{\hat{m}\times n}$, $\hat{b} \in \Re^{\hat{m}}$, and $\hat{x} \in \Re^n$ be given, where $\hat{m} < \infty$. Also define the polyhedron

$$P(\hat{A}, \hat{b}) := \left\{ x \in \Re^n : \hat{A}x \leq \hat{b} \right\}.$$

Proposition 2.3.3. *With the above definitions, it holds that $\hat{x} \notin N(P(\hat{A}, \hat{b}))$ if and only if*

$$\mathcal{C}(\hat{A}, \hat{b}, \hat{x}) := \left\{ \left(\lambda, (v_i, u^i)\right) \in \Re^n \times \mathcal{K}_q^n : \begin{array}{c} \begin{pmatrix} U \\ 0 \end{pmatrix} \hat{A} + \mathrm{Diag}(\lambda) = 0 \\ \hat{x}^T\lambda + \hat{x}_{\mathcal{J}}^T U\hat{b} + (\hat{b} - \hat{A}\hat{x})^T(r\,v - U^T d) < 0 \end{array} \right\}$$

is nonempty, where $U = (u^1, \ldots, u^n) \in \Re^{|\mathcal{J}|\times n}$ and $v = (v_1, \ldots, v_n)^T \in \Re^n$.

Proof. This is just a restatement of Proposition 2.3.2 for the generic polyhedron $P(\hat{A}, \hat{b})$. □

In addition, we define a *proposed canonical solution* associated with $\mathcal{C}(\hat{A}, \hat{b}, \hat{x})$ when $\hat{A}$ is square and invertible:

$$U(\hat{A}) := \left(u^1(\hat{A}), \ldots, u^n(\hat{A})\right) := [\hat{A}^{-1}]_{\mathcal{J}\cdot}, \tag{2.13a}$$

$$v(\hat{A}) := \left(\|u^1(\hat{A})\|_q, \ldots, \|u^n(\hat{A})\|_q\right)^T, \tag{2.13b}$$

$$\lambda := \begin{pmatrix} -e^T & 0 \end{pmatrix}^T, \tag{2.13c}$$

where e is the all-ones vector of length $|\mathcal{J}|$.

Proposition 2.3.4. *If $\hat{A}$ is square and invertible, then the proposed canonical solution $(U, v, \lambda) := (U(\hat{A}), v(\hat{A}), \lambda)$ given by (2.13) is feasible for $\mathcal{C}(\hat{A}, \hat{b}, \hat{x})$ if and only if*

$$-e^T\hat{x}_{\mathcal{J}} + \hat{x}_{\mathcal{J}}^T U\hat{b} + (\hat{b} - \hat{A}\hat{x})^T(r\,v - U^T d) < 0$$

Proof. By construction, (U, v, λ) satisfies all of the conditions for membership in $\mathcal{C}(\hat{A}, \hat{b}, \hat{x})$ except possibly the strict inequality. □

2.3.2 Iteration complexity

For the discussion in this subsection, we assume that $|\mathcal{I}| < \infty$, i.e., P is a polytope.

The general interior-point methodology of Nesterov and Nemirovskii (1994) can be used to derive iteration complexity results for solving the p-cone relaxation (2.9) and/or its dual (2.10). Stated with respect to (2.9), the key result is as follows:

Theorem 2.3.5 (Nesterov and Nemirovskii (1994))**.** *Suppose that (2.9) is interior feasible with finite optimal value v^*. Let a polynomial-time self-concordant barrier for $\mathcal{K}_p$ with barrier parameter θ_p, an interior feasible solution (x^0, X^0), and a tolerance $\epsilon > 0$ be given. Then there exists an algorithm ("short-step primal-only interior-point algorithm"), which delivers a solution (x^*, X^*) satisfying $c^T x^* - v^* < \epsilon$ within $\mathcal{O}(\sqrt{\theta_p |\mathcal{I}|} \log(\epsilon^{-1}(c^T x^0 - v^*)))$ iterations, each of which takes polynomial time.*

As is evident from the theorem, the key ingredient determining the iteration complexity of the interior-point algorithm is the barrier parameter θ_p. It is well known that there exists a self-concordant barrier for $\mathcal{K}_2$ with $\theta_2 = 2 = \mathcal{O}(1)$, and when $p \neq 2$, Andersen et al. (2002) show the existence of a self-concordant barrier for $\mathcal{K}_p$ such that $\theta_p = 4|\mathcal{J}| = \mathcal{O}(|\mathcal{J}|)$. This implies the following corollary.

Corollary 2.3.6. *With respect to Theorem 2.3.5, (2.9) can be solved in $\mathcal{O}(\sqrt{|\mathcal{I}|})$ iterations when $p = 2$ and $\mathcal{O}(\sqrt{|\mathcal{J}||\mathcal{I}|})$ iterations otherwise.*

It is interesting that the iteration complexity does not depend on $|\mathcal{J}|$ when $p = 2$.

This corollary illustrates that, among all relaxations as p varies in $[1, \infty]$, the

second-order cone relaxation has the lowest overall theoretical iteration complexity. In addition, when $p = q = 2$, one can also apply the stronger algorithmic framework of Nesterov and Todd (1997) for homogeneous self-dual cones to obtain long-step primal-dual path-following algorithms, which have high quality practical implementations.

From a theoretical point of view, then, one may be interested only in the relaxations when $p = 2$ (lowest iteration complexity) and $p = \infty$ (strongest relaxation and same iteration complexity as all other $p \neq 2$). Of course, what happens in practice may differ from theory, as we will see in Section 2.5. To close this subsection, we remark that, for $p = 1$ and $p = \infty$, the iteration complexity given by Corollary 2.3.6 matches the iteration complexity obtained if one first formulates (2.9) as its standard LP representation and then applies a LP interior-point method to that representation.

2.3.3 Two types of monotonicity

Monotonicity is a relatively simple, yet important, property of the p-cone procedure outlined in Section 2.2.2. In fact, there are two types of monotonicity though both are derived from the same principle. The first involves the effect of the p-cone procedure on P and its subsets for fixed $(p, \mathcal{J})$, while the second involves the effect on P under different values of p.

The monotonicity properties that we wish to prove for $N_{(p,\mathcal{J})}(P)$ stem directly from the derivation of the p-cone procedure and particularly from the fact that F, $\hat{F}$, and $\hat{P}$ are defined with respect to the inequalities

$$(b_i - a_i^T x)\left(r - \|x_{\mathcal{J}} - d\|_p\right) \geq 0 \qquad \forall\ i \in \mathcal{I}; \tag{2.14}$$

see also (2.2). It is evident that any strengthening of these inequalities in the representations of F, $\hat{F}$, and $\hat{P}$ can yield a corresponding strengthening of $N_{(p,\mathcal{J})}(P)$ around P^{01}. This is the key observation for the following two monotonicity properties.

The first monotonicity property involves strengthening the portion $b_i - a_i^T x$ of (2.14):

Proposition 2.3.7. *Let $p \in [1, \infty]$ and $\emptyset \neq \mathcal{J} \subseteq [n]$ be fixed. Suppose Q is a convex set such that $F \subseteq Q \subseteq P$. Then $P^{01} \subseteq N_{(p,\mathcal{J})}(Q) \subseteq N_{(p,\mathcal{J})}(P) \subseteq P$.*

Proof. The inclusions $P^{01} \subseteq N(Q)$ and $N(P) \subseteq P$ are derived directly from Proposition 2.2.2. To prove $N(Q) \subseteq N(P)$, we simply note that, with respect to $N(Q)$, the sets F, $\hat{F}$, and $\hat{P}$ are based on the inequalities

$$(g_\ell - f_\ell^T x)\left(r - \|x_{\mathcal{J}} - d\|_p\right) \geq 0 \qquad \forall\ \ell \in \mathcal{L},$$

where $Q = \left\{x \in \Re^n : f_\ell^T x \leq g_\ell \ \ \forall\ \ell \in \mathcal{L}\right\}$. Since $Q \subseteq P$, these inequalities are clearly a strengthening of (2.14), and so $N(Q) \subseteq N(P)$. □

The second monotonicity property involves strengthening the portion $r - \|x_{\mathcal{J}} - d\|_p$ of (2.14) and requires the following lemma:

Lemma 2.3.8. *Let $p' \geq 1$, and suppose $v \in \Re^s$ satisfies $\|v\|_{p'}^{p'} \leq s$. Then $\|v\|_p^p \leq s$ for all $p \in [1, p']$.*

Proof. Without loss of generality, we replace v by its component-wise absolute value, i.e., we assume $v_j \geq 0$ for all j.

As a function of p (v fixed), $f(p) := \|v\|_p^p = \sum_{j=1}^s v_j^p$ is convex, and so its maximum over $[1, p']$ occurs at 1 or p'. So to prove the lemma it suffices to show $f(1) \leq s$, i.e.,

$$\sum_{j=1}^{s} v_j \leq s \qquad \Longleftrightarrow \qquad \left(\sum_{j=1}^{s} v_j \right)^{p'} \leq s^{p'}.$$

Next, $g(a) := a^{p'}$ is convex for nonnegative a since $p' \geq 1$. In particular,

$$\begin{aligned} \left(\sum_{j=1}^{s} v_j \right)^{p'} &= g\left(\sum_{j=1}^{s} v_j \right) = g\left(\sum_{j=1}^{s} \frac{1}{s} \cdot s v_j \right) \\ &\leq \sum_{j=1}^{s} \frac{1}{s} g(s v_j) = \sum_{j=1}^{s} s^{p'-1} v_j^{p'} = s^{p'-1} \sum_{j=1}^{s} v_j^{p'} \\ &= s^{p'-1} \|v\|_{p'}^{p'} \leq s^{p'}, \end{aligned}$$

as desired. □

Proposition 2.3.9. *Let $\emptyset \neq \mathcal{J} \subseteq [n]$ and $1 \leq p \leq p' \leq \infty$ be given. Define $r := \sqrt[p]{|\mathcal{J}|}/2$, $r' := \sqrt[p']{|\mathcal{J}|}/2$, and $d = e/2 \in \Re^{|\mathcal{J}|}$ in accordance with Proposition 2.2.1. If $x \in \Re^n$ satisfies $\|x_{\mathcal{J}} - d\|_{p'} \leq r'$, then $\|x_{\mathcal{J}} - d\|_p \leq r$. As a consequence, $N_{(p',\mathcal{J})}(P) \subseteq N_{(p,\mathcal{J})}(P)$.*

Proof. Regarding the first statement of the proposition, we can rearrange the desired implication as

$$\|2x_{\mathcal{J}} - e\|_{p'}^{p'} \leq |\mathcal{J}| \qquad \Longrightarrow \qquad \|2x_{\mathcal{J}} - e\|_p^p \leq |\mathcal{J}|,$$

which holds because of Lemma 2.3.8. Next, the inclusion $N_{(p',\mathcal{J})}(P) \subseteq N_{(p,\mathcal{J})}(P)$ follows because (2.14) is strengthened when p and r are replaced by p' and r'. □

2.3.4 Elimination of fractional extreme points

We introduce the following definition:

Definition 2.3.10. *Let $\emptyset \neq \mathcal{J} \subseteq [n]$ be given. For any $\bar{x} \in \Re^n$, we say that $\bar{x}$ is $\mathcal{J}$-fractional if the subvector $\bar{x}_{\mathcal{J}}$ is contained in $[0,1]^{|\mathcal{J}|}$ and has one or more fractional entries.*

In this subsection, we prove that $N(P)$ contains no $\mathcal{J}$-fractional points, which are extreme in P. Said differently, we show that $N(P)$ cuts off all $\mathcal{J}$-fractional extreme points of P.

We start with the case that $|\mathcal{I}| < \infty$, i.e., P is a polytope. Then we extend the ideas to balls. Finally, we use the analysis with balls to show our main result that, no matter the geometric structure of P, all $\mathcal{J}$-fractional points of P are cut off by $N(P)$.

2.3.4.1 Polytopes

When $|\mathcal{I}| < \infty$, P is a polytope, and since P is bounded in $[0,1]^n$, we know that $|\mathcal{I}| > n$ and that P has extreme points. We prove the following proposition:

Proposition 2.3.11. *Suppose $|\mathcal{I}| < \infty$, i.e., P is a polytope, and $\bar{x}$ is a $\mathcal{J}$-fractional extreme point of P. Then $\bar{x} \notin N(P)$.*

We give two proofs since we feel that both are instructive. The first is a direct proof that the set (2.11) is empty, which implies $\bar{x} \notin N(P)$; see the discussion in Section 2.3.1. The second proof follows the approach of Propositions 2.3.3 and 2.3.4.

Proof. We must show (2.11) is empty, where $\bar{s}_i := b_i - a_i^T \bar{x}$. Because $\bar{x}$ is an extreme point of P, there exists $\mathcal{T} \subseteq \mathcal{I}$ of size n such that $\bar{s}_i = 0$ for all $i \in \mathcal{T}$ and the set $\{a_i : i \in \mathcal{T}\}$ is linearly independent. Hence, any X in (2.11) must satisfy, for all $i \in \mathcal{T}$,

$$\begin{aligned}(0, b_i \bar{x}_{\mathcal{J}} - X_{\mathcal{J}.} a_i) \in \mathcal{K}_p \quad &\iff \quad X_{\mathcal{J}.} a_i = b_i \bar{x}_{\mathcal{J}} \\ &\iff \quad X_{\mathcal{J}.} a_i = (a_i^T \bar{x}) \bar{x}_{\mathcal{J}} \\ &\iff \quad (X_{\mathcal{J}.} - \bar{x}_{\mathcal{J}} \bar{x}^T) a_i = 0.\end{aligned}$$

By the linear independence of $\{a_i : i \in \mathcal{T}\}$, it follows that X must satisfy $X_{\mathcal{J}.} = \bar{x}_{\mathcal{J}} \bar{x}^T$ with $\operatorname{diag}(X) = \bar{x}$. However, since $\bar{x}$ is $\mathcal{J}$-fractional, these conditions are inconsistent. So in fact (2.11) is empty. □

Proof. This proof assumes without loss of generality that $\mathcal{J} = \{1, \ldots, |\mathcal{J}|\}$ in line with (2.12). Let $\mathcal{T}$ be defined as in the previous proof; we assume for simplicity that $\mathcal{T} = [n]$.

We first apply Propositions 2.3.3 and 2.3.4 with $(\hat{A}, \hat{b}, \hat{x}) := (A_{\mathcal{T}.}, b_{\mathcal{T}}, \bar{x})$. Consider the proposed canonical solution $(U, v, \lambda) := (U(\hat{A}), v(\hat{A}), \lambda)$ given by (2.13). Note that $\hat{b} - \hat{A}\hat{x} = 0$ and $U\hat{b} = \hat{x}_{\mathcal{J}}$. By Proposition 2.3.4, (U, v, λ) is feasible for $\mathcal{C}(\hat{A}, \hat{b}, \hat{x})$ since

$$\begin{aligned}-e^T \hat{x}_{\mathcal{J}} + \hat{x}_{\mathcal{J}}^T U\hat{b} + (\hat{b} - \hat{A}\hat{x})^T (r\, v - U^T d) &= -e^T \hat{x}_{\mathcal{J}} + \hat{x}_{\mathcal{J}}^T U\hat{b} + 0^T (r\, v - U^T d) \\ &= -e^T \hat{x}_{\mathcal{J}} + \hat{x}_{\mathcal{J}}^T \hat{x}_{\mathcal{J}} = -e^T \bar{x}_{\mathcal{J}} + \bar{x}_{\mathcal{J}}^T \bar{x}_{\mathcal{J}} \\ &< 0,\end{aligned}$$

where the inequality follows because $\bar{x}$ is $\mathcal{J}$-fractional. So, by Proposition 2.3.3, $\bar{x} \notin N(P(\hat{A}, \hat{b}))$.

Since $P(\hat{A}, \hat{b}) \supseteq P$, it holds by the monotonicity property of Proposition 2.3.7 that $N(P(\hat{A}, \hat{b})) \supseteq N(P)$. Hence, $\bar{x} \notin N(P)$ as desired. □

2.3.4.2 Balls

In the previous subsection, we showed that, if P is a polytope, then $N(P)$ cuts off all $\mathcal{J}$-fractional extreme points from P. The proof used the fact that every extreme point in a polytope corresponds to n linearly independent active constraints. For general P, however, extreme points do not necessarily correspond to n active constraints. For example, if P is a ball in the interior of $[0,1]^n$, then all extreme points of P have exactly one active constraint in the semi-infinite LP representation of P. In this subsection, we study balls to establish that $N(P)$ does in fact cut off all $\mathcal{J}$-fractional extreme points in this case as well. To avoid notational confusion with the P defined in Section 2.2.2, however, we will use B to denote the ball under investigation.

Let B be a ball centered at $h \in \Re^n$ with radius $R > 0$, i.e.,

$$\begin{aligned} B &:= \{x : \|x - h\|_2 \leq R\} \\ &= \{x : w^T(x - h) \leq R \ \ \forall \ w \text{ s.t. } \|w\|_2 = 1\}. \end{aligned} \tag{2.15}$$

In keeping with the development of the p-cone procedure, we could just as well assume that B is the intersection of a ball and $[0,1]^n$, but this is actually not necessary for the result that we present. Moreover, the analysis is a bit simpler without the assumption.

The result is as follows:

Proposition 2.3.12. *Suppose B is a ball given by (2.15) for some center $h \in \Re^n$ and radius $R > 0$. Suppose $\bar{x}$ is a $\mathcal{J}$-fractional extreme point of B. Then $\bar{x} \notin N(B)$.*

The proof of Proposition 2.3.12, although related to the proof of Proposition 2.3.11 for polytopes, is technically different. The fundamental difference is that, for balls, we have only one active constraint at $\bar{x}$, whereas for polytopes, we have n linearly independent ones. Nevertheless, the idea of the proof below is to carefully select n linearly independent constraints, which are *nearly* active at $\bar{x}$. By analyzing those constraints, we see that they have the effect of cutting off $\bar{x}$, yielding a proof similar in spirit to that of Proposition 2.3.11.

Proof. Since $\bar{x}$ is extreme, there exists some $\bar{w}$ with $\|\bar{w}\|_2 = 1$ such that $\bar{w}^T(\bar{x}-h) = R$. In fact, $\bar{w} = R^{-1}(\bar{x} - h)$ since $\|\bar{x} - h\|_2 = R$. Related to $\bar{w}$, we define two additional vectors $\eta, \beta \in \Re^n$. First, let η be any vector having all nonzero entries such that $\eta^T \bar{w} \neq 0$. For example, η could be taken as a small perturbation of $\bar{w}$ itself. Second, define $\beta := \eta^{-1}$, whose components are the inverses of the components of η.

For small $\theta > 0$, define the following collection of n vectors, each of which is a unit-length perturbation of $\bar{w}$:

$$\hat{a}_i := \ell_i^{-1} \left((1-\theta)\bar{w} + \theta \beta_i \, e_i \right) \qquad \forall \;\; i = 1, \ldots, n,$$

where

$$\ell_j := \|(1-\theta)\bar{w} + \theta \beta_j \, e_j\|_2.$$

It is important to note that both $\hat{a}_i$ and ℓ_i depend on θ even though our notation does not reflect this. Note also that $\ell_j > 0$ for $\theta \approx 0$ and so $\hat{a}_i$ is well-defined. Using the notation of Propostions 2.3.3 and 2.3.4, we define

$$\hat{A} := (\hat{a}_1, \ldots, \hat{a}_n)^T$$

$$\hat{b} := Re + \hat{A}h$$

and consider the polyhedron

$$P(\hat{A}, \hat{b}) = \{x : \hat{A}x \leq \hat{b}\} = \left\{x : \hat{a}_i^T(x - h) \leq R \ \ \forall\, i = 1, \ldots, n\right\},$$

which contains B since its defining inequalities are a subset of those defining B. Our proof strategy will be to verify $\bar{x} \notin N(P(\hat{A}, \hat{b}))$ for $\theta \approx 0$ via Proposition 2.3.3 by showing that $\mathcal{C}(\hat{A}, \hat{b}, \bar{x})$ is nonempty. Since $N(B) \subseteq N(P(\hat{A}, \hat{b}))$ due to monotonicity (see Proposition 2.3.7), this will imply $\bar{x} \notin N(B)$ as desired.

To show $\mathcal{C}(\hat{A}, \hat{b}, \bar{x}) \neq \emptyset$, we now consider Proposition 2.3.4 and, in particular, show that the proposed canonical solution $(U, v, \lambda) := (U(\hat{A}), v(\hat{A}), \lambda)$ given by (2.13) satisfies

$$-e^T\bar{x}_{\mathcal{J}} + \bar{x}_{\mathcal{J}}^T U\hat{b} + (\hat{b} - \hat{A}\bar{x})^T(r\, v - U^T d) < 0. \tag{2.16}$$

The proposed canonical solution (U, v, λ) is well-defined because

$$\hat{A} = \mathrm{Diag}(\ell^{-1} \circ \beta)\left((1 - \theta)\eta\bar{w}^T + \theta I\right)$$

is invertible. In particular, its inverse via the Sherman-Morrison-Woodbury formula is

$$\hat{A}^{-1} = \theta^{-1}\left[I - \left(\frac{1 - \theta}{\theta + (1 - \theta)\bar{w}^T\eta}\right)\eta\bar{w}^T\right]\mathrm{Diag}(\ell \circ \eta) \tag{2.17}$$

Note that the denominator $\theta + (1-\theta)\bar{w}^T\eta$ is nonzero for sufficiently small θ since $\bar{w}^T\eta \neq 0$ by construction. So we now investigate the left-hand side of (2.16) and show that its limit as $\theta \to 0_+$ exists and is less than 0, which suffices to establish that $\mathcal{C}(\hat{A}, \hat{b}, \bar{x}) \neq \emptyset$ for $\theta \approx 0$. We remind the reader that $\hat{A}$ and $\hat{b}$ depend on θ, and hence so do U and v.

We first show that $\hat{A}^{-1}e$ equals $\bar{w} + \mathcal{O}(\theta)$. In other words, as $\theta \to 0_+$, $\hat{A}^{-1}e$ approaches $\bar{w}$ such that the error $\hat{A}^{-1} - \bar{w}$ goes to 0 at least as fast as θ itself. A Taylor series expansion of ℓ about $\theta = 0$ shows that

$$\ell = (1-\theta)e + \theta\,\beta \circ \bar{w} + \mathcal{O}(\theta^2),$$

and so $\ell \circ \eta = (1-\theta)\eta + \theta\bar{w} + \mathcal{O}(\theta^2)$ since $\beta = \eta^{-1}$. In addition, using $\bar{w}^T\bar{w} = 1$, we have

$$\bar{w}^T(\ell \circ \eta) = (1-\theta)\bar{w}^T\eta + \theta + \mathcal{O}(\theta^2).$$

Therefore, from (2.17) and the preceding equations,

$$\begin{aligned}
\hat{A}^{-1}e &= \theta^{-1}\left[I - \left(\frac{1-\theta}{\theta + (1-\theta)\bar{w}^T\eta}\right)\eta\bar{w}^T\right](\ell \circ \eta) \\
&= \theta^{-1}\left[\ell \circ \eta - \left(\frac{1-\theta}{\theta + (1-\theta)\bar{w}^T\eta}\right)\eta \cdot \bar{w}^T(\ell \circ \eta)\right] \\
&= \theta^{-1}\left[\ell \circ \eta - \left(\frac{1-\theta}{\theta + (1-\theta)\bar{w}^T\eta}\right)\left(\theta + (1-\theta)\bar{w}^T\eta + \mathcal{O}(\theta^2)\right)\eta\right] \\
&= \theta^{-1}\left[\ell \circ \eta - (1-\theta)\left(1 + \mathcal{O}(\theta^2)\right)\eta\right] \\
&= \theta^{-1}\left[\ell \circ \eta - (1-\theta)\eta + \mathcal{O}(\theta^2)\right] \\
&= \theta^{-1}\left[(1-\theta)\eta + \theta\bar{w} - (1-\theta)\eta + \mathcal{O}(\theta^2)\right] \\
&= \bar{w} + \mathcal{O}(\theta),
\end{aligned}$$

where the fourth equality follows from

$$\frac{\mathcal{O}(\theta)^2}{\theta + (1-\theta)\bar{w}^T\eta} = \mathcal{O}(\theta^2),$$

which holds since $\bar{w}^T\eta \neq 0$.

With $\hat{A}^{-1}e = \bar{w} + \mathcal{O}(\theta)$ now established, it follows that

$$\begin{aligned}
\hat{A}^{-1}\hat{b} = \hat{A}^{-1}(Re + \hat{A}h) &= R\hat{A}^{-1}e + h \\
&= R\bar{w} + h + \mathcal{O}(\theta), \\
\hat{A}^{-1}(\hat{b} - \hat{A}\bar{x}) = \hat{A}^{-1}(Re + \hat{A}(h - \bar{x})) &= R\bar{w} + h - \bar{x} + \mathcal{O}(\theta) \\
&= R \cdot R^{-1}(\bar{x} - h) + h - \bar{x} + \mathcal{O}(\theta) \\
&= \mathcal{O}(\theta),
\end{aligned}$$

where we have used the fact that $\bar{w} = R^{-1}(\bar{x} - h)$. From (2.13), we have $U\hat{b} = R\bar{w}_{\mathcal{J}} + h_{\mathcal{J}} + \mathcal{O}(\theta)$, $U(\hat{b} - \hat{A}\bar{x}) = \mathcal{O}(\theta)$, and $v^T(\hat{b} - \hat{A}\bar{x}) = \mathcal{O}(\theta)$. Thus, the left-hand side of (2.16) is

$$\begin{aligned}
-e^T\bar{x}_{\mathcal{J}} + \bar{x}_{\mathcal{J}}^T &\left(R\bar{w}_{\mathcal{J}} + h_{\mathcal{J}} + \mathcal{O}(\theta)\right) + \mathcal{O}(\theta) \\
&= -e^T\bar{x}_{\mathcal{J}} + \bar{x}_{\mathcal{J}}^T\left(R \cdot R^{-1}(\bar{x}_{\mathcal{J}} - h_{\mathcal{J}}) + h_{\mathcal{J}}\right) + \mathcal{O}(\theta) \\
&= -e^T\bar{x}_{\mathcal{J}} + \bar{x}_{\mathcal{J}}^T\bar{x}_{\mathcal{J}} + \mathcal{O}(\theta).
\end{aligned}$$

As desired, the limit as $\theta \to 0_+$ of the left-hand side of (2.16) equals $-e^T\bar{x}_{\mathcal{J}} + \bar{x}_{\mathcal{J}}^T\bar{x}_{\mathcal{J}}$, which is negative since $\bar{x}$ is $\mathcal{J}$-fractional. □

We will actually use a feature of the above proof again for the proof of Theorem 2.4.1 in Section 2.4. So we record this result for easy reference.

Corollary 2.3.13. *Let B and $\bar{x}$ be as in Proposition 2.3.12. Then there exists a polyhedron $P(\hat{A}, \hat{b}) := \{x \in \Re^n : \hat{A}x \leq \hat{b}\}$, for some $(\hat{A}, \hat{b})$, such that $P(\hat{A}, \hat{b}) \supseteq B$ and $\bar{x} \notin N(P(\hat{A}, \hat{b})) \supseteq N(B)$.*

Proof. The desired polyhedron is $P(\hat{A}, \hat{b})$, depending on small $\theta > 0$, in the proof of Proposition 2.3.12. □

2.3.4.3 The general case

We now show that $N(P)$ cuts off all $\mathcal{J}$-fractional extreme points of P. The basic idea is that, given a $\mathcal{J}$-fractional extreme point $\bar{x} \in P$, there exists a ball $B \supseteq P$ such that $\bar{x}$ is also a $\mathcal{J}$-fractional extreme point of B. Thus, by Proposition 2.3.12 and the monotonicity property of Proposition 2.3.7, $\bar{x} \notin N(B) \supseteq N(P)$.

We first establish the existence of the ball B just described. A similar result has been used in Kojima and Tunçel (2000a,b).

Proposition 2.3.14. *Let $\bar{x}$ be a $\mathcal{J}$-fractional extreme point of P. Then there exists a ball B such that $P \subseteq B$ and $\bar{x}$ is a $\mathcal{J}$-fractional extreme point of B.*

Proof. This proposition is just a simple application of standard convex analysis. Recall that P is compact convex. Hence, there exists a hyperplane $H := \{x : \alpha^T x = \beta\}$ supporting P at $\bar{x}$, i.e., $\bar{x} \in H$ and $P \setminus \{\bar{x}\} \subseteq H_{++} := \{x : \alpha^T x > \beta\}$. We also define $H_+ := \{x : \alpha^T x \geq \beta\}$.

Next, given $\gamma > 0$, we define a ball $B(\gamma)$ dependent on $\bar{x}$ and α:

$$B(\gamma) := \{x : \|x - (\bar{x} + \gamma\alpha)\|_2 \leq \gamma\|\alpha\|_2\}\,.$$

It is easy to check that $B(\gamma) \subseteq H_+$ and that $\bar{x}$ is an extreme point of $B(\gamma)$. Furthermore, for every $x \in H_{++}$, there exists sufficiently large γ such that $x \in B(\gamma)$. Hence, because $P \setminus \{\bar{x}\} \subseteq H_{++}$ is bounded, there exists sufficiently large γ such that $P \setminus \{\bar{x}\} \subseteq B(\gamma)$, and so $P \subseteq B(\gamma)$. For any such large γ, we can take $B := B(\gamma)$ to achieve the proposition. □

The proof of the ensuing theorem then follows from Proposition 2.3.14.

Theorem 2.3.15. *Let $\bar{x}$ be a $\mathcal{J}$-fractional extreme point of P. Then $\bar{x} \notin N(P)$.*

Proof. Let B be the ball of Proposition 2.3.14. Then, by Proposition 2.3.12, $\bar{x} \notin N(B)$. Since $N(B) \supseteq N(P)$ by monotonicity of Proposition 2.3.7, $\bar{x} \notin N(P)$. □

2.4 Iterated Procedure and Convergence

So far we have discussed how the p-cone procedure produces $N_{(p,\mathcal{J})}(P)$ from P for a given $(p, \mathcal{J})$. Because $N(P)$ is compact convex with its own semi-infinite outer description which may or may not be known explicitly, we may conceptually apply the p-cone procedure — perhaps for a different choice of $(p, \mathcal{J})$ — to $N(P)$ itself. In fact, we may repeat the p-cone procedure *ad infinitum*. A key question is whether the resultant sequence of compact convex sets converges to P^{01}.

More formally, let $\{(p_k, \mathcal{J}^k)\}_{k\geq 1}$ be a sequence of choices $p_k \in [1, \infty]$ and $\emptyset \neq \mathcal{J}^k \subseteq [n]$, and define $N^1(P) := N_{(p_1,\mathcal{J}^1)}(P)$ and $N^k(P) := N_{(p_k,\mathcal{J}^k)}(N^{k-1}(P))$ for all $k > 1$. We then ask whether $\lim_{k\to\infty} N^k(P)$ equals P^{01}.

Lovász and Schrijver (1991), Kojima and Tunçel (2000a,b), and Balas et al. (1993) have all considered the same question for their own procedures. In particular,

one may interpret Lovász and Schrijver (1991) as taking $p_k = \infty$ and $\mathcal{J}^k = [n]$ for all k; they show finite convergence after n iterations, i.e., $N^n(P) = P^{01}$. Recall that the method of Kojima and Tunçel (2000a,b), when applied to 0-1 programs, essentially reduces to that of Lovász and Schrijver (1991); so they take the same p_k and $\mathcal{J}^k$. Finally, one may interpret Balas et al. (1993) as choosing p_k arbitrarily and $\mathcal{J}^k$ a single element in $[n]$. The authors prove that, if $\mathcal{J}^1 \cup \cdots \cup \mathcal{J}^n = [n]$, then $N^n(P) = P^{01}$.

We show in Theorem 2.4.1 below that the iterated p-cone procedure converges asymptotically for arbitrary $\{p_k\}_{k=1}^{\infty}$ as long as each index $j \in [n]$ appears infinitely often in the sequence $\{\mathcal{J}^k\}_{k=1}^{\infty}$. Before stating and proving the theorem, we discuss a few items.

First, we have mentioned that the approach of Kojima and Tunçel (2000a,b) obtains asymptotic convergence in general. It is reasonable to ask if their approach or proof techniques may somehow subsume ours and hence prove convergence of our procedure. However, this is not the case since their asymptotic analysis uses *all* valid "rank-2" quadratic inequalities, i.e., valid inequalities $(b_i - a_i^T x)(b_h - a_h^T x) \geq 0$ obtained by multiplying *any* pair of valid linear inequalities $b_i - a_i^T x \geq 0$ and $b_h - a_h^T x \geq 0$ for P. In contrast, our approach and analysis require only a partial subset of such inequalities, which are obtained by multiplying valid linear inequalities of the p-cone constraint $\|x_{\mathcal{J}} - d\|_p \leq r$ with the inequalities $b_i - a_i^T x \geq 0$ defining P; see Section 2.2.4.2 for more discussion. In this sense, one can think of our approach as proving asymptotic convergence under weaker conditions than those used by Kojima

and Tunçel (2000a,b) although the sets we consider are less general than those studied by Kojima and Tunçel.

Second, after the proof of Theorem 2.4.1, we provide an example where an infinite number of iterations is required to converge. So our p-cone successive relaxation procedure does not have finite convergence in general. Of course, for specific sequences $\{(p_k, \mathcal{J}^k)\}_{k=1}^{\infty}$, it may be possible to prove finiteness as with Lovász and Schrijver (1991) and Balas et al. (1993).

Third, we suspect that obtaining a rate of asymptotic convergence is difficult. This perceived difficulty stems from the methodology used to prove Theorem 2.3.15, which establishes that $\mathcal{J}$-fractional extreme points are cut off by the p-cone procedure. To establish a rate of convergence, it seems necessary to establish how "deep" these cuts are with respect to P^{01}, but the methodology of Theorem 2.3.15 uses the existence of cuts with no quantitative knowledge of their strength.

We are now ready to state and prove the theorem.

Theorem 2.4.1. *Let $\{(p_k, \mathcal{J}^k)\}_{k\geq 1}$ be a sequence of choices $p_k \in [1, \infty]$ and $\emptyset \neq \mathcal{J}^k \subseteq [n]$, which give rise to compact, convex sets $N^k(P) \supseteq P^{01}$ via the definitions $N^1(P) := N_{(p_1,\mathcal{J}^1)}(P)$ and $N^k(P) := N_{(p_k,\mathcal{J}^k)}(N^{k-1}(P))$ for all $k > 1$. Then $N^k(P) \supseteq N^{k+1}(P)$ so that $\lim_{k\to\infty} N^k(P)$ exists and equals $\cap_{k\geq 1} N^k(P)$. In addition, if $\cup_{k\geq \bar{k}} \mathcal{J}^k = [n]$ for all $\bar{k}$, then $\lim_{k\to\infty} N^k(P) = P^{01}$.*

Proof. Since each $N^k(P)$ is compact and convex and contained in $N^{k-1}(P)$, $\lim_{k\to\infty} N^k(P)$ exists and equals $Z := \cap_{k=1}^{\infty} N^k(P)$. This proves the first part of the theorem.

To prove the second part, we first claim that every extreme point of Z is

integer. Suppose for contradiction that $\bar{z}$ is a fractional extreme point of Z, and let j be any index where $\bar{z}_j$ is fractional. Next, let $\mathcal{S} := \{\mathcal{J} \subseteq [n] : j \in \mathcal{J}\}$. Theorem 2.3.15 implies $\bar{z} \notin N_{(1,\mathcal{J})}(Z)$ for all $\mathcal{J} \in \mathcal{S}$.

A continuity argument, which we prove two paragraphs below, implies that, for each $\mathcal{J} \in \mathcal{S}$, there exists $k_{\mathcal{J}}$ large enough so that $\bar{z} \notin N_{(1,\mathcal{J})}(N^{k-1}(P))$ for all $k \geq k_{\mathcal{J}}$. Define $\hat{k} := \max\{k_{\mathcal{J}} : \mathcal{J} \in \mathcal{S}\}$. In particular, consider $k \geq \hat{k}$ such that $j \in \mathcal{J}^k$. Since $\mathcal{J}^k \in \mathcal{S}$, it holds that $\bar{z} \notin N_{(1,\mathcal{J}^k)}(N^{k-1}(P))$. By the monotonicity property of Proposition 2.3.9, it also holds that $\bar{z} \notin N_{(1,\mathcal{J}^k)}(N^{k-1}(P)) \supseteq N_{(p_k,\mathcal{J}^k)}(N^{k-1}(P)) = N^k(P)$, which contradicts the statement $\bar{z} \in Z$. Hence, we conclude that every extreme point of Z is integer.

Since $P^{01} \subseteq Z$ by construction, it thus follows that $Z = P^{01}$.

Now we prove the continuity argument from above, i.e., for each $\mathcal{J} \in \mathcal{S}$, we prove the existence of $k_{\mathcal{J}}$ large enough so that $\bar{z} \notin N_{(1,\mathcal{J})}(N^{k-1}(P))$ for all $k \geq k_{\mathcal{J}}$. So let $\mathcal{J} \in \mathcal{S}$ be fixed. Since $\bar{z}$ is a $\mathcal{J}$-fractional extreme point of Z, by Proposition 2.3.14 there exists a ball $B \supseteq Z$ such that $\bar{z}$ is a $\mathcal{J}$-fractional extreme point of B. Furthermore, by Corollary 2.3.13, there exists a polyhedron

$$P(\hat{A}, \hat{b}) := \{x \in \Re^n : \hat{A}x \leq \hat{b}\},$$

for some $(\hat{A}, \hat{b})$, containing B such that $\bar{z} \notin N_{(1,\mathcal{J})}(P(\hat{A}, \hat{b}))$. By Proposition 2.3.3, this is equivalent to $\mathcal{C}(\hat{A}, \hat{b}, \bar{z}) \neq \emptyset$. For small $\epsilon > 0$, this in turn implies $\mathcal{C}(\hat{A}, \hat{b} + \epsilon e, \bar{z}) \neq \emptyset$ or, equivalently, $\bar{z} \notin N_{(1,\mathcal{J})}(P(\hat{A}, \hat{b} + \epsilon e))$.

Since $N^{k-1}(P)$ converges to $Z \subseteq P(\hat{A}, \hat{b})$, for any $\epsilon > 0$, there exists k_ϵ large enough so that $N^{k-1}(P) \subseteq P(\hat{A}, \hat{b} + \epsilon e)$ for all $k \geq k_\epsilon$. Then for small ϵ and large k,

it follows that $\bar{z} \notin N_{(1,\mathcal{J})}(N^{k-1}(P))$ by the previous paragraph and monotonicity. $\square$

We now give an example, which shows that the p-cone successive relaxation procedure may require an infinite number of iterations to converge.

To construct the example, we first analyze the behavior of a particular class of 2-dimensional polytopes under a single iteration of the procedure with $p < \infty$ and $\mathcal{J} = [2]$. For any $a > 1$, consider the following two-dimensional polytope:

$$P(a) := \left\{ (x_1, x_2) \geq 0 : \begin{array}{l} ax_1 + x_2 \leq a \\ x_1 + ax_2 \leq a \end{array} \right\}.$$

Note that $P(a)$ is symmetric about the line $x_1 = x_2$ and has the four vertices $(0,0)$, $(0,1)$, $(1,0)$ and $(a/(a+1), a/(a+1))$, the last of which is greater than $(1/2, 1/2)$ but smaller than $(1,1)$. In particular, $P(a)$ is contained in $[0,1]^2$, and its integer convex hull is

$$P^{01} = \{(x_1, x_2) \geq 0 : \ x_1 + x_2 \leq 1\}.$$

Proposition 2.4.2. *Let $a > 1$ be given. For any $p < \infty$ and $\mathcal{J} = [2]$, there exists $a' \in (1, a)$ such that*

$$P(a') \subseteq N_{(p,\mathcal{J})}\left(P(a)\right).$$

Proof. It suffices to show the existence of $1/2 < \delta < 1$ such that $(\delta, \delta) \in N(P(a))$ since then

$$\operatorname{conv}\{(0,0), (0,1), (1,0), (\delta,\delta)\} = P\left(\frac{\delta}{1-\delta}\right)$$

is contained in $N(P(a))$. In this case, we may take $a' = \delta/(1-\delta)$, which necessarily satisfies $a' \in (1, a)$.

So let $1/2 < \delta < 1$ be arbitrary. By the discussion prior to Proposition 2.3.2, $(\delta, \delta) \in N(P(a))$ if and only if the set (2.11) with $x_1 = x_2 = \delta$ is nonempty. In our case, after eliminating $X_{11} = X_{22} = \delta$, the relevant set is

$$\hat{P}_\delta := \left\{ (0,0) \leq (X_{12}, X_{21}) \leq (\delta, \delta) : \begin{array}{c} \left\| \begin{pmatrix} (a-1)\delta - aX_{21} - \frac{1}{2}\beta \\ X_{12} + \frac{1}{2}\beta \end{pmatrix} \right\|_p \leq r\beta \\ \left\| \begin{pmatrix} (a-1)\delta - aX_{12} - \frac{1}{2}\beta \\ X_{21} + \frac{1}{2}\beta \end{pmatrix} \right\|_p \leq r\beta \end{array} \right\},$$

where $r = \sqrt[p]{2}/2$ and $\beta := a - \delta(a+1)$. It then sufficies to show that the set

$$\begin{aligned} \hat{P}'_\delta &:= \hat{P}_\delta \cap \{(X_{12}, X_{21}) : X_{12} = X_{21}\} \\ &= \left\{ 0 \leq y \leq \delta : \left\| \begin{pmatrix} (a-1)\delta - ay - \frac{1}{2}\beta \\ y + \frac{1}{2}\beta \end{pmatrix} \right\|_p \leq r\beta \right\} \end{aligned}$$

is nonempty.

We next consider the closely related set

$$\hat{P}'_{1/2} := \left\{ 0 \leq y \leq \frac{1}{2} : \left\| \begin{pmatrix} (a-1)/4 - ay \\ (a-1)/4 + y \end{pmatrix} \right\|_p \leq \frac{r}{2}(a-1) \right\},$$

which is gotten by substituting $\delta \leftarrow 1/2$ in the definition of $\hat{P}'_\delta$. Within this convex set, note that $y = 0$ is feasible and makes the cone constraint active. Since $a > 1$, it is then easy to see that increasing y to a small positive number satisfies the cone constraint strictly. In other words, $\hat{P}'_{1/2}$ has nonempty interior.

By continuity, it then follows that for δ sufficiently close to $1/2$, $\hat{P}'_\delta$ is nonempty, as desired. □

With Proposition 2.4.2 in hand, we can construct the example having infinite convergence. For any $a > 1$, consider the sequence $N^k(P(a))$ from Theorem 2.4.1 based on any choice $\{(p_k, \mathcal{J}^k)\}_{k \geq 0}$ satisfying $p_k < \infty$ and $\mathcal{J}^k = [2]$ for all k. By

Proposition 2.4.2, the monotonicity property of Proposition 2.3.7, and induction there exists $a_k > 1$ such that $P(a_k) \subseteq N^k(P(a))$ for all k. Hence, $N^k(P(a)) \neq P^{01}$ for all k, which ensures convergence only in the limit.

Note that this example also shows infinite convergence even if symmetry $X = X^T$ is enforced in the p-cone procedure since symmetry actually is enforced in the proof of Proposition 2.4.2.

2.5 Computational Considerations

For fixed $\mathcal{J}$, a single application of the p-cone lift-and-project procedure gives rise to a family of relaxations of P^{01} parameterized by p. From the monotonicity property of Proposition 2.3.9, we know that the larger p is, the tighter the corresponding relaxation. So $p = \infty$ is the tightest. On the other hand, we have shown in Corollary 2.3.6 that optimizing over the $p = 2$ relaxation induces the lowest theoretical iteration complexity — in fact, an order of magnitude less than all other p, which themselves share the same iteration complexity. Thus, one may be particularly interested in the cases $p = 2$ (second-order cone programming) and $p = \infty$ (linear programming).

In this section, we computationally test these cases using state-of-the-art SOCP and LP software. We had hypothesized that the lower iteration complexity combined with the high quality of modern SOCP software would make solving $p = 2$ quicker than solving $p = \infty$ — perhaps much quicker so as to justify the loss in relaxation quality. However, as described next, we have observed that solving $p = \infty$ is faster with better bounds.

We conduct experiments on two sets of problems. The first set includes 8 instances of the maximum stable set problem from the Center for Discrete Mathematics and Theoretical Computer Science (Johnson and Trick, 1996). For a graph G with vertex set $[n]$ and edge set $E \subseteq [n] \times [n]$, the (unweighted) maximum stable set problem is

$$\alpha := \max\{e^T x \mid x_i + x_j \leq 1,\ (i,j) \in E,\ x \in \{0,1\}^n\}.$$

Table 2.1 contains a basic description of the instances.

Name	n	$\lvert E\rvert$
johnson8-2-4	28	168
MANN-a9	45	72
hamming6-2	64	192
keller4	171	5100
brock200_1	200	5066
san200_07_1	200	5970
sanr200_07	200	6032
c-fat200-1	200	18366

Table 2.1: Description of the stable set instances.

We set $\mathcal{J} = [n]$ and solve both the $p = 2$ and $p = \infty$ relaxations. We also enforce the symmetry condition $X = X^T$ of (2.6) so as to eliminate about half of the variables in X. The SOCPs were solved using MOSEK 5.0, and the LPs were solved using both CPLEX 9.0 and MOSEK 5.0. Pre-solving was turned off for all solvers, and computations were performed under the Linux operating system with a single 2.8 GHz AMD Opteron processor and 4 GB of RAM.

Regarding the solution of the LPs, we used CPLEX to solve the dual form (2.10) using the dual simplex method, which gave better results than, for example, solving (2.9) with the dual simplex method. On the other hand, MOSEK's LP solver optimizes (2.9) and (2.10) simultaneously using a primal-dual interior-point method.

Table 2.2 compares α, the bounds, and the solution times (in seconds). The values for those cells containing "$*$" were unavailable due to the solvers running out of memory.

Name	LP value	α	Bounds		Times		
			$p=\infty$	$p=2$	$p=\infty$		$p=2$
					CPLEX	MOSEK	
johnson8-2-4	14	4	9.33	12.2	9.00e−02	1.30e−01	2.40e−01
MANN-a9	22.5	16	18	20.5	1.76e+00	2.40e−01	1.15e+00
hamming6-2	32	32	32	32	2.39e+01	1.91e+00	1.00e+01
keller4	85.5	11	57	80.9	2.14e+03	1.44e+03	4.67e+03
brock200_1	100	21	66.7	95.1	3.36e+04	8.09e+03	1.18e+04
san200_07_1	100	30	66.7	95.1	1.00e+05	9.00e+02	3.05e+04
sanr200_07	100	18	66.7	95	5.76e+04	6.93e+03	1.21e+04
c-fat200-1	100	12	*	*	*	*	*

Table 2.2: The bounds and times (in seconds) for solving the $p = \infty$ and $p = 2$ relaxations of the stable set instances from Table 2.1. Each LP is solved using two methods: the dual simplex method (CPLEX) and the primal-dual interior-point method (MOSEK). An asterisk ($*$) indicates that the corresponding solver ran out of memory. A time limit of 100,000 seconds is enforced for each run.

Table 2.2 clearly shows the overall superiority of the LP relaxation as solved

by MOSEK for the maximum stable set problems. The $p = \infty$ relaxation can be solved faster and the bounds are better as well. Still, it is worth noting that the SOCP relaxations solve more quickly than the LP relaxations via the dual simplex method.

The second set of test problems includes 7 mixed-integer programming problems from MIPLIB 2003 Achterberg et al. (2006). A description of the problems is shown in Table 2.3. Specifically, Rows and Columns give the number of constraints and variables of those problems. There are two types of variables, binary and continuous, whose numbers are listed under the last two columns of Table 2.3. The column Non-zero indicates the number of non-zero entries in the constraints. We note that the problems contain no general integer variables. These problems are among some of the smallest problems of MIPLIB 2003, but not all of them are considered easy to solve. For example, *markshare1* is small in size but is classified as "hard" in MIPLIB 2003.

Name	Rows	Colums	Non-zero	Binary	Continuous
markshare1	6	62	312	50	12
pk1	45	86	915	55	31
pp08a	136	240	480	64	176
pp08aCUTS	246	240	839	64	176
modglob	291	422	968	98	324
danoint	664	521	3232	56	465
qiu	1192	840	3432	48	792

Table 2.3: Description of selected instances from MIPLIB 2003

Just as with the stable set instances, we set $\mathcal{J}$ equal to the index set of binary variables and solve the $p = \infty$ and $p = 2$ relaxations using CPLEX 9.0 and MOSEK 5.0. The computing environment is similar, the only difference being that we turn on pre-solve for both CPLEX and MOSEK to avoid out-of-memory failures. The bounds of the relaxations and the times in seconds are reported in Table 2.4 along with the optimal and LP-relaxation values. Note that these are minimization problems. The

Name	LP value	IP value	Bounds			Times		
			$p = \infty$		$p = 2$	$p = \infty$		$p = 2$
			CPLEX	MOSEK		CPLEX	MOSEK	
markshare1	0	1	0	0	0	9.12e+00	1.45e+00	4.02e+00
pk1	1.47e−09	1.10e+01	0	0	3.43e−03	1.58e+02	9.10e+00	2.50e+01
pp08a	2.75e+03	7.35e+03	6.40e+03	6.40e+03	3.11e+03	2.38e+02	3.21e+01	3.91e+01
pp08aCUTS	5.48e+03	7.35e+03	6.77e+03	6.77e+03	5.61e+03	7.19e+02	8.13e+01	1.49e+02
modglob	2.04e+07	2.07e+07	–	2.06e+07	2.04e+07	1.00e+05	3.10e+03	4.28e+03
danoint	6.26e+01	6.57e+01	–	6.28e+01	6.27e+01	1.00e+05	4.74e+02	4.53e+04
qiu	-9.32e+02	-1.33e+02	-3.08e+02	-3.08e+02	-8.37e+02	3.95e+04	1.06e+02	2.19e+02

Table 2.4: The bounds and times (in seconds) for solving $p = \infty$ and $p = 2$ relaxations of selected instances from MIPLIB 2003. The relaxations of $p = \infty$ were solved using both CPLEX and MOSEK. A time limit of 100,000 seconds was enforced and "–" means the solver did not find the optimal value when time limit was reached.

conclusion we draw from Table 2.4 is similar to that of Table 2.2: overall, the $p = \infty$ relaxation outperforms the $p = 2$ relaxation in terms of both bounds and CPU times.

Although these computational results do not align with the theoretical result of Corollary 2.3.6 that solving the $p = 2$ relaxation has a lower iteration complexity, we are hopeful that improvements in SOCP software may make the p-cone procedure

more competitive in the future. We also feel that these results are a testament to the high quality of current LP software.

2.6 Conclusions

In this chapter, we have introduced lift-and-project procedures for 0-1 integer programming based on p-order cone programming. From a theoretical point of view, our approach generalizes and unifies several existing methods, which have been based on linear and semidefinite programming. Asymptotic convergence of the repeated application of our procedure has also been established, and for $p = 2$, when applying one iteration of the p-cone procedure, our method enjoys a theoretical iteration complexity, which is an order of magnitude faster than existing lift-and-project techniques. From the computational point of view, solving the SOCP corresponding to $p = 2$ is not competitive with solving the LP for $p = \infty$. Overall, we feel that the p-cone procedure makes a solid theoretical contribution to the literature on lift-and-project procedures, with possible computational improvements in the future as SOCP solvers become more efficient.

We conclude with a final observation. We have mentioned in the Introduction that Kim and Kojima Kim and Kojima (2003) have derived SOCP relaxations from SDP lift-and-project relaxations by replacing semidefiniteness with just the semidefiniteness of 2×2 principal submatrices, which can be modeled by SOCP. In addition, Kim and Kojima (2001) and Kim et al. (2003) use SDP lift-and-project relaxations to generate valid convex quadratic constraints, which are SOCP-representable and

then enforced in place of semidefiniteness. When $p = 2$, the approach in this chapter is different. In particular, our method is not derived from semidefiniteness. In fact, irrespective of p, semidefiniteness can be applied in our procedure to further enhance its strength, and so the above SOCP ideas can also be applied to our procedure as well for any p.

CHAPTER 3
RELAXING THE OPTIMALITY CONDITIONS OF BOX QP

This chapter has been published in Burer and Chen (2009b).

3.1 Introduction

In this chapter, we study semidefinite programming (SDP) relaxations for the fundamental problem of minimizing a nonconvex quadratic function over a box:

$$\min \left\{ \frac{1}{2} x^T Q x + c^T x \ : \ 0 \leq x \leq e \right\}, \tag{3.1}$$

where $x \in \Re^n$, $Q \in \Re^{n \times n}$, $c \in \Re^n$, and $e \in \Re^n$ is the all-ones vector. Without loss of generality, we assume Q is symmetric. If Q is not positive semidefinite (as we assume in this chapter), then (3.1) is NP-hard (Pardalos and Vavasis, 1991).

There are numerous methods for solving (3.1) and more general nonconvex quadratic programs, including local methods (Gould and Toint, 2002) and global methods (Pardalos, 1991). For a survey of methods to solve (3.1) globally, see De Angelis et al. (1997) as well as Vandenbussche and Nemhauser (2005b,a) and Burer and Vandenbussche (2006a).

Critical to any global optimization method for (3.1) is the ability to relax it into a convex problem, one which hopefully provides a tight lower bound on the optimal value with low computational cost. One standard approach is to linearize the quadratic term $x_i x_j$ via a single variable X_{ij} and then to enforce implied linear constraints, which link X_{ij} with x_i and x_j, e.g., $0 \leq X_{ij} \leq \min\{x_i, x_j\}$ (Sherali and Adams, 1997). The resulting relaxation is a linear program. A second approach also

linearizes the terms $x_i x_j$ — by introducing a symmetric matrix variable X to replace the aggregate xx^T — but then includes the valid semidefinite inequality $X \succeq xx^T$ to obtain an SDP relaxation.

In this chapter, we focus on SDP relaxations of (3.1) rather than linear ones. In principle, it is always possible to combine linear and semidefinite approaches (yielding better bounds with added computational costs; see Anstreicher (2007)), but the goal of this chapter is to improve SDP relaxations.

Our approach is to consider semidefinite relaxations of (3.1), which incorporate the standard first- and second-order necessary optimality conditions for (3.1). Vandenbussche and Nemhauser (2005b,a) and Burer and Vandenbussche (2006a) have previously considered linear and semidefinite relaxations, respectively, involving only the first-order conditions. The contributions of the current chapter are to demonstrate how also to incorporate the second-order conditions and to illustrate the positive effects of doing so.

We point out that Nesterov (2000) has considered incorporating the second-order conditions into SDP relaxations of quadratic optimization over p-norm boxes for $2 \leq p < \infty$, i.e., $\{x : \|x\|_p^p \leq 1\}$. However, Nesterov strongly uses the fact that the function $\|x\|_p^p$ is smooth for $p \in [2, \infty)$. Our case (p equal to ∞) is wholly different because of the lack of smoothness.

The chapter is organized as follows. In Section 3.2, we review the first- and second-order optimality conditions of (3.1). In particular, we show how to express the second-order conditions without explicit knowledge of the inactive constraints. This

will prove to be a critical ingredient in constructing semidefinite relaxations involving the second-order conditions. In Section 3.3, we review the basic semidefinite relaxation of (3.1) due to Shor (1987) and then introduce a semidefinite relaxation, which incorporates the first- and second-order optimality conditions. We also construct a relaxation based only on the second-order conditions.

We will call the three relaxations just mentioned (SDP_0), (SDP_{12}), and (SDP_2), respectively. The subscript indicates the type of "order" information incorporated in the relaxation. By construction, it will hold that (SDP_{12}) is at least as strong as (SDP_2), which is at least as strong as (SDP_0). On the other hand, (SDP_{12}) requires the largest solution time, while (SDP_0) requires the smallest one.

Continuing in Sections 3.4 and 3.5, we study the relationship of these three relaxations. In Section 3.4, we prove the surprising, somewhat negative result that all three achieve the same optimal value. (On the positive side, the proof establishes several interesting analytical properties of (SDP_0), which are of independent interest.) Despite this equivalence, Section 3.5 demonstrates positively that, in the context of branch-and-bound to globally solve (3.1), (SDP_{12}) and (SDP_2) are significantly stronger than (SDP_0), when each is appropriately tailored for use at any node of the tree. Our computational experiments are described in detail in Section 3.5, including a new, effective branching strategy.

3.1.1 Notation and terminology

In this section, we introduce some of the notation that will be used throughout the chapter. $\Re^n$ refers to n-dimensional Euclidean space; $\Re^{n\times n}$ is the set of real, $n \times n$ matrices. We let $e_i \in \Re^n$ represent the i-th unit vector. For a set $\mathcal{I}$ in a particular ground set, $\mathcal{I}^c$ is its complement in that ground set. The norm of a vector $v \in \Re^n$ is denoted by $\|v\| := \sqrt{v^T v}$. For a vector v and an index set $\mathcal{I}$, $v_{\mathcal{I}}$ is defined as the vector composed of entries of v that are indexed by $\mathcal{I}$. Also, given a matrix $A \in \Re^{n\times n}$, $A_{\mathcal{I}\mathcal{I}}$ is defined as the matrix composed of entries of A whose rows and columns are indexed by $\mathcal{I}$. We denote by A^j and A_i the j-th column and i-th row of A, respectively. The notation $\mathrm{diag}(A)$ is defined as the vector, which is the diagonal of A, while $\mathrm{Diag}(v)$ denotes the diagonal matrix with diagonal v. The inner product of two matrices $A, B \in \Re^{n\times n}$ is defined as $A \bullet B := \mathrm{trace}(A^T B)$. Given two vectors $x, v \in \Re^n$, we denote their Hadamard product by $x \circ v \in \Re^n$, where $[x \circ v]_j = x_j v_j$; an analogous definition applies to the Hadamard product of matrices. Finally, $A \succeq 0$ means matrix A is positive semidefinite, and $A \succ 0$ means A is positive definite.

3.2 Optimality Conditions

In this section, we first state the standard first- and second-order necessary optimality conditions for (3.1) involving the set of inactive constraints. Then we derive an expression for the second-order conditions that does not explicitly require knowledge of the inactive constraint set.

For any x satisfying $0 \leq x \leq e$, define the following sets of inactive constraints:

$$\mathcal{I}_0(x) := \{i : x_i > 0\}$$

$$\mathcal{I}_1(x) := \{i : x_i < 1\}$$

$$\mathcal{I}(x) := \{i : 0 < x_i < 1\} = \mathcal{I}_0(x) \cap \mathcal{I}_1(x).$$

Note that $\mathcal{I}(x)^c = \mathcal{I}_0(x)^c \cup \mathcal{I}_1(x)^c$ indexes the active constraints at x. Let $y, z \in \Re^n$ denote the Lagrange multipliers for the constraints $e - x \geq 0$ and $x \geq 0$, respectively. For fixed y, z, the Lagrangian of (3.1) is defined as

$$L(x; y, z) := \frac{1}{2} x^T Q x + c^T x - z^T x - y^T (e - x).$$

With these definitions, the necessary optimality conditions for (3.1) are

$$0 \leq x \leq e \tag{3.2a}$$

$$\nabla_x L(x; y, z) = Qx + c - z + y = 0 \tag{3.2b}$$

$$y \geq 0, \ z \geq 0 \tag{3.2c}$$

$$z_i = 0, \ y_j = 0 \qquad \forall \ i \in \mathcal{I}_0(x), \ \forall \ j \in \mathcal{I}_1(x) \tag{3.2d}$$

$$v^T \nabla^2_{xx} L(x; y, z) v = v^T Q v \geq 0 \qquad \forall \ v \in V(x) \tag{3.2e}$$

where

$$V(x) := \{ \ v \ : \ e_i^T v = 0 \ \ \forall \ \ i \in \mathcal{I}_0(x)^c, \ \ -e_j^T v = 0 \ \ \forall \ \ j \in \mathcal{I}_1(x)^c \ \}$$

$$= \{ \ v \ : \ v_i = 0 \ \ \forall \ \ i \in \mathcal{I}(x)^c \ \}$$

is the null space of the Jacobian of the active constraints. By eliminating z and

employing other straightforward simplifications, we can rewrite and label (3.2) as

$$0 \leq x \leq e \qquad \text{(primal feasibility)} \tag{3.3a}$$

$$Qx + c + y \geq 0,\ y \geq 0 \qquad \text{(dual feasibility)} \tag{3.3b}$$

$$x \circ (Qx + c + y) = 0,\ y \circ (e - x) = 0 \qquad \text{(complementary slackness)} \tag{3.3c}$$

$$Q_{\mathcal{I}(x)\mathcal{I}(x)} \succeq 0. \qquad \text{(local convexity)} \tag{3.3d}$$

Now we give an equivalent form of the local convexity condition (3.3d), which does not explicitly involve knowledge of $\mathcal{I}(x)$.

Proposition 3.2.1. *Given x, define*

$$w := x \circ (e - x). \tag{3.4}$$

Then the local convexity condition (3.3d) at x is equivalent to

$$Q \circ ww^T \succeq 0. \tag{3.5}$$

Proof. For notational convenience, we write $\mathcal{I}$ for $\mathcal{I}(x)$ and D for $\mathrm{Diag}(x)$. We first show the equivalence of (3.3d) and the inequality

$$(I - D)DQD(I - D) \succeq 0.$$

Assume (3.3d) holds. By definition, $D(I - D)$ is a diagonal matrix such that, for all $i \in \mathcal{I}(x)^c$, the i-th diagonal entry is 0. For any v, define $\tilde{v} := D(I - D)v$. Then $\tilde{v}_i = 0$ for all $i \in \mathcal{I}^c$ and

$$v^T(I - D)DQD(I - D)v = \tilde{v}^T Q\tilde{v} = \tilde{v}_{\mathcal{I}}^T Q_{\mathcal{I}\mathcal{I}}\tilde{v}_{\mathcal{I}} \geq 0.$$

So $(I-D)DQD(I-D)$ is positive semidefinite. Conversely, assume $(I-D)DQD(I-D) \succeq 0$. Since $D_{\mathcal{I}\mathcal{I}} > 0$ and $[I-D]_{\mathcal{I}\mathcal{I}} > 0$, for any partial vector $\tilde{v}_{\mathcal{I}}$, there exists some v such that the full vector $\tilde{v} := D(I-D)v$ extends $\tilde{v}_{\mathcal{I}}$ and also satisfies $\tilde{v}_{\mathcal{I}^c} = 0$. So

$$\tilde{v}_{\mathcal{I}}^T Q_{\mathcal{I}\mathcal{I}} \tilde{v}_{\mathcal{I}} = \tilde{v}^T Q \tilde{v} = v^T (I-D)DQD(I-D)v \geq 0,$$

which establishes (3.3d).

Now, the equivalence of (3.3d) and $Q \circ ww^T \succeq 0$ follows from

$$(I-D)DQD(I-D) = \operatorname{Diag}(w) Q \operatorname{Diag}(w) = Q \circ ww^T.$$

□

It follows from Proposition 3.2.1 that (3.1) can be reformulated as the following quadratic semidefinite program, which does not depend explicitly on knowledge of the inactive constraints:

$$\min \left\{ \frac{1}{2} x^T Q x + c^T x \ : \quad \text{(3.3a–3.3c)} \quad \text{(3.4)} \quad \text{(3.5)} \right\}. \tag{3.6}$$

3.3 Semidefinite Relaxations

In this section, we first present the basic semidefinite relaxation of (3.1) due to Shor (1987). Then we introduce semidefinite relaxations of the new formulation (3.6).

3.3.1 Shor's bounded relaxation (SDP_0)

As is standard in the SDP literature (see for example Shor (1987)), we can use the non-convex equality $X = xx^T$ to represent (3.1) in the equivalent form

$$\min \left\{ \frac{1}{2} Q \bullet X + c^T x \ : \ 0 \le x \le e, \ \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0, \ X = xx^T \right\}.$$

By dropping the constraint $X = xx^T$, we obtain the relaxation due to Shor:

$$\min \left\{ \frac{1}{2} Q \bullet X + c^T x \ : \ 0 \le x \le e, \ \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \right\}. \tag{3.7}$$

However, the following well known fact about (3.7) is easy to prove:

Proposition 3.3.1. *If $Q \not\succeq 0$, then the optimal value of (3.7) is $-\infty$.*

The reason why (3.7) is unbounded when $Q \not\succeq 0$ is that there is too much freedom for X. We can fix the problem of unboundedness by including some valid linear constraints implied by $X = xx^T$ and $0 \le x \le e$, e.g., $\mathrm{diag}(X) \le x$ (Sherali and Adams, 1997). Adding $\mathrm{diag}(X) \le x$ to (3.7), we get a bounded relaxation for (3.1):

$$\min \left\{ \frac{1}{2} Q \bullet X + c^T x \ : \ 0 \le x \le e, \ \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0, \ \mathrm{diag}(X) \le x \right\}. \tag{SDP_0}$$

In particular, the optimal solution set of (SDP_0) is nonempty. We consider (SDP_0) to be the smallest, simplest semidefinite relaxation of (3.1).

We remark that Ye (1999) has derived an approximation algorithm for quadratic programming over the box $\{x : -e \le x \le e\}$, which is simply a shifted and scaled version of (3.1). The main tool used by Ye is the equivalent version of (SDP_0) for the case $\{x : -e \le x \le e\}$.

3.3.2 Relaxations (SDP_{12}) and (SDP_2) of the optimality conditions

To relax (3.6), we consider the matrix

$$\begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix}^T = \begin{pmatrix} 1 & x^T & y^T & w^T \\ x & xx^T & xy^T & xw^T \\ y & yx^T & yy^T & yw^T \\ w & wx^T & wy^T & ww^T \end{pmatrix} \succeq 0$$

and its linearized version

$$M = \begin{pmatrix} 1 & x^T & y^T & w^T \\ x & X & M_{xy}^T & M_{xw}^T \\ y & M_{xy} & Y & M_{yw}^T \\ w & M_{xw} & M_{yw} & W \end{pmatrix} \succeq 0.$$

We can relax the quadratic constraints (3.3c), (3.4) and (3.5) via M. For example, consider the j-th entry of $x \circ (Qx + c + y) = 0$ from (3.3c), which is

$$x_j(Q_j x + c_j + y_j) = 0.$$

Relaxing it via M, we have

$$Q_j X^j + c_j x_j + [M_{xy}]_{jj} = 0.$$

So, $x \circ (Qx + c + y) = 0$ is relaxed in total as

$$\mathrm{diag}(QX) + c \circ x + \mathrm{diag}(M_{xy}) = 0.$$

Constraints (3.4) and (3.5) can be relaxed in a similar way. Hence, we obtain the

following SDP relaxation of (3.6), which we call (SDP_{12}):

$$\begin{aligned}
\min \quad & \frac{1}{2}Q \bullet X + c^T x && \text{(3.8a)}\\
\text{s.t.} \quad & 0 \le x \le e, \ \ \text{diag}(X) \le x && \text{(3.8b)}\\
& Qx + c + y \ge 0, \ \ y \ge 0 && \text{(3.8c)}\\
& \text{diag}(QX) + c \circ x + \text{diag}(M_{yx}) = 0, \ \ y - \text{diag}(M_{yx}) = 0 && \text{(3.8d)}\\
& w = x - \text{diag}(X) && \text{(3.8e)}\\
& Q \circ W \succeq 0 && \text{(3.8f)}\\
& M \succeq 0. && \text{(3.8g)}
\end{aligned}$$

We point out that $\text{diag}(X) \le x$ in (3.8b) is not a relaxation of a particular constraint in (3.6). Rather, it is added to prevent (SDP_{12}) from being unbounded as with (SDP_0).

We also study a relaxed version of (SDP_{12}), which we call (SDP_2):

$$\begin{aligned}
\min \quad & \frac{1}{2}Q \bullet X + c^T x && \text{(3.9a)}\\
\text{s.t.} \quad & 0 \le x \le e, \ \ \text{diag}(X) \le x && \text{(3.9b)}\\
& w = x - \text{diag}(X) && \text{(3.9c)}\\
& Q \circ W \succeq 0 && \text{(3.9d)}\\
& \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 && \text{(3.9e)}\\
& \begin{pmatrix} 1 & w^T \\ w & W \end{pmatrix} \succeq 0. && \text{(3.9f)}
\end{aligned}$$

In essence, (SDP_2) maintains the minimal set of constraints from (SDP_{12}), which still explicitly relax the second-order optimality conditions. We are particularly interested

in (SDP_2) because it captures the second-order optimality information (a main focus of this chapter) and because its dimension is significantly lower than that of (SDP_{12}). Table 3.1 compares the sizes of the three SDPs.

	SDP_0	SDP_2	SDP_{12}
# variables	$(n^2+3n)/2$	n^2+3n	$9(n^2+n)/2$
# linear constraints	$3n$	$4n$	$8n$
# semidefinite constraints	1	3	2
sizes of semidefinite constraints	$n+1$	n, $n+1$, $n+1$	n, $3n+1$

Table 3.1: Comparison of the sizes of (SDP_0), (SDP_2), and (SDP_{12})

3.4 Equivalence of the SDP Relaxations

3.4.1 Equivalence of (SDP_0) and (SDP_2)

In this section, we establish the equivalence of (SDP_0) and (SDP_2). We will use the following generic result:

Lemma 3.4.1. *Consider two related optimization problems:*

$$(A) \qquad \min\{\ f(x)\ : x \in P\ \}$$

$$(B) \qquad \min\{\ f(x)\ : x \in P,\ y \in R(x)\ \},$$

where P represents the feasible set for (A) and $R(x)$ defines the set of constraints (related to x) that y must satisfy. Let x^* be an optimal solution of (A) and suppose $R(x^*) \neq \emptyset$. Then any (x^*, y) with $y \in R(x^*)$ is optimal for (B). Therefore, the optimal value of (B) equals that of (A).

Since the feasible set of (B) is more restrictive than that of (A), the optimal value of (B) is greater than or equal to that of (A). The inclusion $y \in R(x^*)$ and the fact that the objective value does not depend on y together imply that (A) and (B) achieve the same optimal value.

Our first step is to prove the following property of (SDP_0) at optimality.

Lemma 3.4.2. *Let (x^*, X^*) be an optimal solution of (SDP_0), and define $\mathcal{J} := \{i : X^*_{ii} < x^*_i\}$. Then $Q_{\mathcal{J}\mathcal{J}} \succeq 0$.*

Proof. The argument is based on examining an optimal solution for the dual of (SDP_0). It can be easily verified that the dual is

$$\begin{array}{ll} \max & \lambda - e^T y \\ \text{s.t.} & S = \dfrac{1}{2}\begin{pmatrix} -\lambda & (c+y-z-v)^T \\ c+y-z-v & Q + 2\,\mathrm{Diag}(v) \end{pmatrix} \succeq 0 \\ & y, z, v \geq 0, \end{array}$$

where y, z, v, S, and λ are, respectively, the multipliers for $e - x \geq 0$, $x \geq 0$, $x - \mathrm{diag}(X) \geq 0$, $(1, x^T; x, X) \succeq 0$, and the constraint associated with fixing the top-left entry of $(1, x^T; x, X)$ to 1.

Note that both (SDP_0) and its dual have nonempty interior. Specifically, the point

$$(x, X) = \left(\frac{1}{2}e, \frac{1}{4}ee^T + \varepsilon I\right)$$

is interior feasible for (SDP_0) for all $\varepsilon \in (0, 1/4)$. In addition, taking v sufficiently positive, λ sufficiently negative, and y, z positive such that $y - z - v$ has sufficiently small norm yields an interior solution of the dual with $S \succ 0$. Because both problems

have interiors, strong duality holds. For the remainder of the proof, we let (x, X) and (λ, y, z, v, S) denote specific optimal solutions of the primal and dual.

Due to complementary slackness, $(x - \operatorname{diag}(X)) \circ v = 0$. So $v_{\mathcal{J}} = 0$, and it follows from $S \succeq 0$ that

$$[Q + 2\operatorname{Diag}(v)]_{\mathcal{J}\mathcal{J}} = Q_{\mathcal{J}\mathcal{J}} \succeq 0.$$

□

Theorem 3.4.3. *Let (x^*, X^*) be an optimal solution of (SDP$_0$), and define $w^* := x^* - \operatorname{diag}(X^*)$ and $W^* := w^*(w^*)^T$. Then (x^*, X^*, w^*, W^*) is an optimal solution of (SDP$_2$) with the same optimal value.*

Proof. For notational convenience, we drop the $*$ superscripts. By Lemma 3.4.1, we need only prove that (x, X, w, W) is feasible for (SDP$_2$), and to do so requires the verification of (3.9d) since all the other constraints of (SDP$_2$) are satisfied by construction.

Let $\mathcal{J}$ be defined as in Lemma 3.4.2. Then $w_{\mathcal{J}} > 0$, $w_{\mathcal{J}^c} = 0$, and $Q_{\mathcal{J}\mathcal{J}} \succeq 0$. Note that $[Q \circ W]_{ij} = 0$ if $i \in \mathcal{J}^c$ or $j \in \mathcal{J}^c$. So $Q \circ W = Q \circ ww^T \succeq 0$ is equivalent to $Q_{\mathcal{J}\mathcal{J}} \circ \left(w_{\mathcal{J}} w_{\mathcal{J}}^T\right) = \operatorname{Diag}(w_{\mathcal{J}}) Q_{\mathcal{J}\mathcal{J}} \operatorname{Diag}(w_{\mathcal{J}}) \succeq 0$, which is true because $Q_{\mathcal{J}\mathcal{J}} \succeq 0$ and $w_{\mathcal{J}} > 0$. This proves (3.9d) and hence the theorem. □

3.4.2 Equivalence of (SDP$_0$) and (SDP$_{12}$)

In the last subsection, we have proved that (SDP$_2$) is equivalent to (SDP$_0$). In this subsection, we show that even (SDP$_{12}$) is equivalent to (SDP$_0$). We start by

proving some properties of (SDP_0), which will facilitate the proof of equivalence later in Section 3.4.2.2 but are also of independent interest.

3.4.2.1 Additional properties of (SDP_0)

We will show that every optimal solution (x^*, X^*) of (SDP_0) satisfies the following two inequalities:

$$\text{diag}(QX) + c \circ x \leq 0 \tag{3.10a}$$

$$Qx + c - (\text{diag}(QX) + c \circ x) \geq 0. \tag{3.10b}$$

In other words, (3.10a) and (3.10b) are redundant for (SDP_0) in the sense that enforcing these inequalities does not change the optimal solution set. This knowledge will help us establish the equivalence between (SDP_{12}) and (SDP_0) in the next subsection.

To prove (3.10), we start by examining paths of solutions in the feasible set of (SDP_0). Given any feasible (x, X), consider two paths of emanating from (x, X) and depending on a specified index i. Each path is parameterized by $\alpha \geq 0$. We define $(x_1(\alpha), X_1(\alpha))$ and $(x_2(\alpha), X_2(\alpha))$ by

$$\begin{aligned}
x_1(\alpha) &:= x - \alpha x_i e_i \\
X_1(\alpha) &:= X - \alpha e_i \left(X^i\right)^T - \alpha X^i e_i^T + \alpha^2 X_{ii} e_i e_i^T \\
x_2(\alpha) &:= x + \alpha e_i \\
X_2(\alpha) &:= X + \alpha e_i x^T + \alpha x e_i^T + \alpha^2 e_i e_i^T.
\end{aligned}$$

Furthermore, for any $\beta \in [0, 1]$, we consider a third path $(x(\alpha), X(\alpha))$, which is a

convex combination of $(x_1(\alpha), X_1(\alpha))$ and $(x_2(\alpha), X_2(\alpha))$:

$$\begin{aligned} x(\alpha) &:= \beta x_1(\alpha) + (1-\beta)x_2(\alpha) \\ X(\alpha) &:= \beta X_1(\alpha) + (1-\beta)X_2(\alpha). \end{aligned}$$

Our intent is to examine conditions on α and β such that $(x(\alpha), X(\alpha))$ is feasible for (SDP_0). We will also be interested in the objective value at $(x(\alpha), X(\alpha))$:

$$f(\alpha) := \frac{1}{2} Q \bullet X(\alpha) + c^T x(\alpha)$$

Proposition 3.4.4. *Let an index i be specified. Given (x, X) feasible for (SDP_0) and $\beta \in [0,1]$, $(x(\alpha), X(\alpha))$ satisfies the following properties:*

(i) $x(\alpha)$ differs from x only in the i-th entry, and $x(\alpha)_i = x_i + \alpha(1 - \beta - \beta x_i)$;

(ii) $\text{diag}(X(\alpha)) - x(\alpha)$ differs from $\text{diag}(X) - x$ only in the i-th entry, and

$$[\text{diag}(X(\alpha)) - x(\alpha)]_i =$$

$$(1 - \alpha\beta)(X_{ii} - x_i) + \alpha\left[\alpha(\beta X_{ii} + 1 - \beta) + 2(1-\beta)x_i - (\beta X_{ii} + 1 - \beta)\right];$$

(iii) $\begin{pmatrix} 1 & x(\alpha)^T \\ x(\alpha) & X(\alpha) \end{pmatrix}$ *is positive semidefinite.*

Moreover,

$$f'(0) = \beta(-\left(Q^i\right)^T X^i - c_i x_i) + (1-\beta)(\left(Q^i\right)^T x + c_i).$$

Proof. It is straightforward to verify the formulas for $x(\alpha)_i$, $[\text{diag}(X(\alpha)) - x(\alpha)]_i$, and $f'(0)$. Now we show that (iii) holds. From the definition of $(x_1(\alpha), X_1(\alpha))$, we see

$$\begin{pmatrix} 1 & x_1(\alpha)^T \\ x_1(\alpha) & X_1(\alpha) \end{pmatrix} = \begin{pmatrix} 1 & 0^T \\ 0 & I - \alpha e_i e_i^T \end{pmatrix} \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \begin{pmatrix} 1 & 0^T \\ 0 & I - \alpha e_i e_i^T \end{pmatrix} \succeq 0.$$

Furthermore, by using the Schur complement theorem twice, we have

$$\begin{aligned}
\begin{pmatrix} 1 & x_2(\alpha)^T \\ x_2(\alpha) & X_2(\alpha) \end{pmatrix} \succeq 0 &\iff \begin{pmatrix} 1 & (x+\alpha e_i)^T \\ x+\alpha e_i & X + \alpha e_i x^T + \alpha x e_i^T + \alpha^2 e_i e_i^T \end{pmatrix} \succeq 0 \\
&\iff (X + \alpha e_i x^T + \alpha x e_i^T + \alpha^2 e_i e_i^T) - (x+\alpha e_i)(x+\alpha e_i)^T \succeq 0 \\
&\iff X - xx^T \succeq 0 \\
&\iff \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0,
\end{aligned}$$

which is true due to the feasibility of (x, X). Therefore, (iii) follows because $(1, x(\alpha)^T; x(\alpha), X)$ is a convex combination of positive semidefinite matrices. □

The following corollaries are easy to establish.

Corollary 3.4.5. *Let (x, X) be feasible for (SDP$_0$), and let $\beta = 1$. For a specified index i, $(x(\alpha), X(\alpha))$ is feasible for all $\alpha \in [0, 1]$, and $f'(0) = -\left(Q^i\right)^T X^i - c_i x_i$.*

Corollary 3.4.6. *Let (x, X) be feasible for (SDP$_0$) with $x_i < 1$, which guarantees $1 + X_{ii} - 2x_i > 0$. Also let $\beta = 1/2$. For a specified index i, $(x(\alpha), X(\alpha))$ is feasible for all $\alpha \in \left[0, \frac{1+X_{ii}-2x_i}{1+X_{ii}}\right]$, and $f'(0) = \frac{1}{2}[Qx + c - (\mathrm{diag}(QX) + c \circ x)]_i$.*

We are now ready to prove that every optimal solution (x^*, X^*) of (SDP$_0$) satisfies (3.10). We need just one additional lemma, whose proof is a straightforward adaptation of the proof of proposition 3.2 in Burer and Vandenbussche (2006a):

Lemma 3.4.7. *Let (x, X) be feasible for (SDP$_0$). Then $x_i = 1$ implies $X^i = x$.*

Theorem 3.4.8. *Let (x^*, X^*) be optimal for (SDP$_0$). Then (x^*, X^*) satisfies the inequalities (3.10).*

Proof. We prove the following equivalent statement: suppose feasible (x, X) does not satisfy (3.10); then (x, X) is not optimal. We break the condition of not satisfying (3.10) into three subcases: (i) $[\text{diag}(QX) + c \circ x]_i > 0$ for some i; (ii) $[Qx + c - (\text{diag}(QX)+c\circ x)]_i < 0$ for some i and $x_i < 1$; and (iii) $[Qx+c-(\text{diag}(QX)+c\circ x)]_i < 0$ for some i and $x_i = 1$.

In case (i), Corollary 3.4.5 implies the existence of a feasible path emanating from (x, X) with decreasing objective. Hence, (x, X) is not optimal. Case (ii) follows similarly from Corollary 3.4.6.

Finally, we show that case (iii) actually cannot occur. Suppose $x_i = 1$. Then by Lemma 3.4.7,

$$[Qx + c - \text{diag}(QX) - c \circ x]_i = \left(Q^i\right)^T (x - X^i) + c_i(1 - x_i) = 0,$$

which is incompatible with (iii). □

3.4.2.2 Proof of equivalence

Note that (SDP_{12}) is more constrained than (SDP_0). By Lemma 3.4.1, it suffices to construct a feasible solution to (SDP_{12}) based on (x^*, X^*) to establish the equivalence of (SDP_0) and (SDP_{12}).

We construct the solution for (SDP_{12}) by defining

$$
\begin{aligned}
y &:= -(\operatorname{diag}(QX^*) + c \circ x^*) && \text{(3.11a)}\\
Y &:= yy^T + \epsilon I, && \text{(3.11b)}\\
w &:= x^* - \operatorname{diag}(X^*),\ W := ww^T && \text{(3.11c)}\\
M_{xw} &:= wx^{*T},\ M_{yw} := wy^T && \text{(3.11d)}\\
M &:= \begin{pmatrix} 1 & x^{*T} & y^T & w^T \\ x^* & X^* & M_{xy}{}^T & M_{xw}{}^T \\ y & M_{xy} & Y & M_{yw}{}^T \\ w & M_{xw} & M_{yw} & W \end{pmatrix}, && \text{(3.11e)}
\end{aligned}
$$

where $\epsilon > 0$ is a sufficiently large constant (more details below). Note that we have not specified M_{xy} yet; we will do so below.

We must check that the solution specified is indeed feasible for (SDP_{12}), which requires checking (3.8b)–(3.8g). Obviously, (3.8b) is satisfied by (x^*, X^*). It follows from (3.10a) and (3.10b) that (3.8c) is satisfied by (x^*, X^*, y). The constraint (3.8e) is satisfied by definition, and Theorem 3.4.3 illustrates that (3.8f) is satisfied. It remains to show that (3.8d) and (3.8g) hold. These will depend on the choice of ϵ and M_{xy}.

To prove (3.8d) and (3.8g), we exhibit an M_{xy} such that $\operatorname{diag}(M_{xy}) = y$ and $M \succeq 0$. We first require the following lemma and proposition:

Lemma 3.4.9. *For an optimal solution (x^*, X^*) of (SDP$_0$), if $X^{i*} = x_i^* x^*$, then $y_i(1 - x_i^*) = 0$, where y is defined as in (3.11a).*

Proof. We drop the superscripts $*$ to simplify notation. If $x_i = 1$, then $y_i(1 - x_i) = 0$, and if $x_i = 0$, then $y_i = -((Q^i)^T X^i + c_i x_i) = -x_i((Q^i)^T x + c_i) = 0$. If $0 < x_i < 1$, we show $y_i = 0$. Let $g_i := (Q^i)^T x + c_i$. We know $y_i = -x_i g_i \geq 0$, and so $g_i \leq 0$. On

the other hand, by (3.10b), $g_i + y_i = (1 - x_i)g_i \geq 0$, and so $g_i \geq 0$. Hence, $g_i = 0$, which ensures $y_i = 0$. □

Proposition 3.4.10. *Let (x^*, X^*) be an optimal solution of (SDP_0), and define y as in (3.11a). Then there exists $A \in \Re^{n\times n}$ such that*

$$\operatorname{diag}(A(X^* - x^* x^{*T})) = y \circ (e - x^*).$$

Proof. We drop the superscripts $*$ to simplify notation. We show equivalently that there exists a solution A to the system of equations

$$A_i(X - xx^T)^i = y_i(1 - x_i) \quad \forall\ i = 1, \ldots, n.$$

Note that the n equations just listed are separable; so we consider each i separately. If $(X - xx^T)^i \neq 0$, it is obvious that there exists a solution A_i; just take A_i equal to

$$\frac{y_i(1 - x_i)}{\|(X - xx^T)^i\|^2} \left[(X - xx^T)^i\right]^T.$$

On the other hand, if $(X - xx^T)^i = 0$, i.e., $X^i = x_i x$, then we know by Lemma 3.4.9 that $y_i(1 - x_i) = 0$ and thus A_i can be any vector. □

We define

$$M_{xy} := yx^{*T} + A(X^* - x^* (x^*)^T),$$

where A is any matrix as in Proposition 3.4.10. Then $\operatorname{diag}(M_{xy}) = y \circ x^* + y \circ (e - x^*) = y$, which ensures that (3.8d) is satisfied. Finally, it remains to show that (3.8g) holds, i.e., $M \succeq 0$, for this choice of M_{xy}.

In the following, we drop the superscripts $*$ to simplify notation. Note that

$$M = \begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \\ w \end{pmatrix}^T + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & X - xx^T & (M_{xy} - yx^T)^T & 0 \\ 0 & M_{xy} - yx^T & \epsilon I & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and so it suffices to show

$$\begin{pmatrix} X - xx^T & (M_{xy} - yx^T)^T \\ M_{xy} - yx^T & \epsilon I \end{pmatrix} = \begin{pmatrix} X - xx^T & \big(A(X - xx^T)\big)^T \\ A(X - xx^T) & \epsilon I \end{pmatrix} \succeq 0.$$

By the Schur complement theorem, this holds if and only if

$$(X - xx^T) - \epsilon^{-1}(X - xx^T)A^T A(X - xx^T) \succeq 0. \tag{3.12}$$

Consider the following straightforward lemma:

Lemma 3.4.11. *Suppose $R, S \succeq 0$. Then there exists $\delta > 0$ small enough such that $R - \delta S \succeq 0$ if and only if* $\text{Null}(R) \subseteq \text{Null}(S)$.

Because the null space of $X - xx^T$ is contained in the null space of $(X - xx^T)A^T A(X - xx^T)$, the lemma implies the existence of $\epsilon > 0$ large enough so that (3.12) holds. Taking such ϵ, we conclude that (3.8g) is satisfied.

Overall, we have shown that definition (3.11) — along with the definitions of M_{xy} and ϵ — is feasible for (SDP_{12}), which means (SDP_0) and (SDP_{12}) are equivalent by Lemma 3.4.1.

3.5 Comparison of SDP Relaxations Within Branch-and-Bound

In Section 3.4, we have shown that the three SDP relaxations (SDP_0), (SDP_2) and (SDP_{12}) are equivalent. In this section, we empirically compare these relaxations

in the context of branch-and-bound for solving (3.1) globally, where the relaxations can have different effects on subdivided boxes. Our experiments on randomly generated problems illustrate the strength of the bounds produced by (SDP_2) and (SDP_{12}) over those produced by (SDP_0) in this context. Our approach will be to focus on the comparison of (SDP_0) and (SDP_2), while briefly commenting on (SDP_{12}) towards the end.

We would like to point out that our intention here is *not* to develop a branch-and-bound method for (3.1), which outperforms all other techniques. Rather, our primary goal is comparing (SDP_0), (SDP_{12}), and (SDP_2) in order to gauge the effect of incorporating optimality conditions (particularly the second-order conditions) into SDP relaxations for (3.1).

3.5.1 Branch-and-bound for box QP

The branch-and-bound algorithm we consider recursively subdivides the entire box $\{x \in \Re^n : 0 \leq x \leq e\}$ into smaller and smaller boxes and solves an appropriately tailored SDP relaxation — either (SDP_0) or (SDP_2) — on these smaller boxes. Lower bounds obtained from these relaxations are compared with a global upper bound to fathom as many small boxes as possible from consideration. When fathoming is not possible for a specific small box, that box is further subdivided. Moreover, the global upper bound is improved (whenever possible) throughout the course of the algorithm.

We measure the performance of the branch-and-bound algorithm in two ways: the total number of nodes in the branch-and-bound tree and the total time to complete

the entire branch-and-bound process. The number of nodes is affected by the quality of the lower bounds. Our main comparison will be the lower bound calculations based on either (SDP_0) or (SDP_2). Since the branching strategy also affects the number of nodes, we will investigate two different branching strategies as well.

Before discussing our algorithm design choices below, we first present the SDP relaxations on the small boxes, which are modified appropriately from the corresponding versions on the entire box. Suppose the current node of the branch-and-bound tree corresponds to the box

$$\{\ x \in \Re^n :\ l \leq x \leq u\ \}.$$

Then the SDP relaxation that corresponds to (SDP_0) is

$$\begin{aligned}
\min \quad & \frac{1}{2} Q \bullet X + c^T x && \text{(3.13a)}\\
\text{s.t.} \quad & l \leq x \leq u && \text{(3.13b)}\\
& \operatorname{diag}(X) - (l+u) \circ x + l \circ u \leq 0 && \text{(3.13c)}\\
& \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0. && \text{(3.13d)}
\end{aligned}$$

The constraint $\operatorname{diag}(X) - (l+u) \circ x + l \circ u \leq 0$ is obtained by relaxing the valid inequality

$$x \circ x - (l+u) \circ x + l \circ u = (x-l) \circ (x-u) \leq 0.$$

Note that, when $l = 0$ and $u = e$, this constraint is just $\operatorname{diag}(X) \leq x$. So this inequality plays the role of bounding the diagonal of X on the smaller boxes.

To derive the relaxation corresponding to (SDP_2) on the smaller box, we first

introduce the following notation depending on the bounds (l, u):

$$B_1 := \{i : \ell_i = 0 \text{ and } u_i = 1\},$$

$$B_2 := \{i : \ell_i = 0 \text{ and } u_i < 1\},$$

$$B_3 := \{i : \ell_i > 0 \text{ and } u_i = 1\},$$

$$B_4 := \{i : \ell_i > 0 \text{ and } u_i < 1\}.$$

Note that $B_1 \cup B_2 \cup B_3 \cup B_4 = \{1, \ldots, n\}$. Now consider the following specialization of Proposition 3.2.1:

Proposition 3.5.1. *Given x satisfying $0 \leq l \leq x \leq u \leq e$, define $w \in \Re^n$ by*

$$\begin{aligned} w_{B_1} &:= x_{B_1} \circ (e_{B_1} - x_{B_1}) \\ w_{B_2} &:= x_{B_2} \\ w_{B_3} &:= e_{B_3} - x_{B_3} \\ w_{B_4} &:= e_{B_4}. \end{aligned} \tag{3.14}$$

Then the local convexity condition (3.3d) at x is equivalent to

$$Q \circ ww^T \succeq 0.$$

Proof. Recall that the proof of Proposition 3.2.1 established that the second-order condition for (3.1) at any $0 \leq x \leq e$ is equivalent to

$$(I - D)DQD(I - D) \succeq 0, \tag{3.15}$$

where I is the identity matrix and $D = \mathrm{Diag}(x)$. For fixed i, if $l_i > 0$, then we know the i-th diagonal of D is strictly positive, and so we can replace $D_{ii} = x_i$ with $D_{ii} = 1$

in the inner two D's of (3.15) without affecting semidefiniteness. Similarly, if $u_i < 1$, then we know that the i-th diagonal entry of $I - D$ is positive and hence can be replaced by 1 without affecting semidefiniteness in (3.15). If $l_i > 0$ and $u_i < 1$, then both replacements can be made. Using arguments similar to the proof of Proposition 3.2.1, the resulting matrix $(I-D)DQD(I-D)$ after replacements is equal to $Q \circ ww^T$, where w is given by (3.14). □

With Proposition 3.5.1 in hand, the relaxation corresponding to (SDP_2) on the smaller box is

$$\begin{aligned}
\min \quad & \frac{1}{2} Q \bullet X + c^T x && \text{(3.16a)}\\
\text{s.t.} \quad & l \le x \le u && \text{(3.16b)}\\
& \operatorname{diag}(X) - (l+u) \circ x + l \circ u \le 0 && \text{(3.16c)}\\
& w_{B_1} = x_{B_1} - \operatorname{diag}(X_{B_1 B_1}) && \text{(3.16d)}\\
& w_{B_2} = x_{B_2} && \text{(3.16e)}\\
& w_{B_3} = e_{B_3} - x_{B_3} && \text{(3.16f)}\\
& w_{B_4} = e_{B_4} && \text{(3.16g)}\\
& W_{B_2 B_2} = X_{B_2 B_2} && \text{(3.16h)}\\
& W_{B_2 B_3} = x_{B_2} e_{B_3}^T - X_{B_2 B_3} && \text{(3.16i)}\\
& W_{B_3 B_3} = e_{B_3} e_{B_3}^T - x_{B_3} e_{B_3}^T - e_{B_3} x_{B_3}^T + X_{B_3 B_3} && \text{(3.16j)}\\
& W^{B_4} = w e_{B_4}^T && \text{(3.16k)}\\
& Q \circ W \succeq 0, \quad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0, \quad \begin{pmatrix} 1 & w^T \\ w & W \end{pmatrix} \succeq 0. && \text{(3.16l)}
\end{aligned}$$

Note that the constraints (3.16d–3.16g) give rise to new constraints (3.16h–3.16k) between W and x, w, and X.

We now address the major design choices for the branch-and-bound algorithm:

- **Bounding**. We will compare strategies involving two types of lower bounds: those given by (3.13) and those given by (3.16). A single run of the branch-and-bound algorithm on a single instance will employ one or the other — or a combination of both, i.e., first (3.13) and then a switch to (3.16) (more details are given in the next subsection).

  For the global upper bound, we experimented with two ways to improve it at each node of the tree: (a) locally solve the small box QP at each node via MATLAB's `quadprog` function; (b) simply take the objective value $\frac{1}{2}(x^*)^T Q x^* + c^T x^*$ corresponding to the optimal x^* obtained from the lower bound calculation. Option (b) was a bit quicker, but at each node, the time for either (a) or (b) was dominated by the lower bound calculation. On the other hand, compared to (b), (a) generally resulted in fewer nodes in the tree and thus saved time overall. So we use (a) throughout the computations.

- **Branching**. We consider two branching strategies.

  The first branching strategy — which we will call *simple* — is the standard "bisection via longest edge" (see, for example, Horst and Tuy (1993)). Consider the small box $\{x \in \Re^n : l \leq x \leq u\}$, which has been selected for branching. We select the longest edge of this box to branch on. More specifically, we choose the index i such that $u_i - l_i$ is the largest among all dimensions. If there is a

tie, the smallest such index is chosen. By applying this strategy, we subdivide the box into two smaller boxes:

$$\left\{ x \in \Re^n \ : \ l \leq x \leq u - \frac{1}{2}(u_i - l_i)e_i \right\},$$
$$\left\{ x \in \Re^n \ : \ l + \frac{1}{2}(u_i - l_i)e_i \leq x \leq u \right\}.$$

The second branching strategy — which we will call *advanced* — is more sophisticated and involves two ingredients:

- It is well known that SDP relaxations such as (3.13) and (3.16) enforce a fairly weak approximation of $X = xx^T$ in the interior of $[l, u]$ and a fairly strong one near the boundary. Hence, if $\bar{x}$ is an optimal solution returned by (3.13) or (3.16) and if $\bar{x}_i \in (l_i, u_i)$ for some index i, then branching on i via the intervals $[l_i, \bar{x}_i]$ and $[\bar{x}_i, u_i]$ results in two relaxations with $\bar{x}$ on the boundary, thus strengthening the approximation of $X = xx^T$ precisely where needed and increasing the chances that $\bar{x}$ will be cut off in the relaxations. A similar logic guides the "most fractional" branching rule of integer programming.
- For theoretical validity of the branch-and-bound algorithm, the branching strategy must subdivide all boxes in such a way that the longest edge of all unfathomed boxes tends to 0 in the limit. This is indeed the most basic property of "bisection via longest edge."

So we design the second branching strategy as a combination of "most fractional" and "bisection via longest edge." For a given feasible solution $l \leq \bar{x} \leq u$,

our strategy calculates, for each $i = 1, \ldots, n$, the values

$$\alpha_i = \frac{u_i - \bar{x}_i}{u_i - l_i} \cdot \frac{\bar{x}_i - l_i}{u_i - l_i} \in [0, 1/4],$$

$$\beta_i = u_i - l_i \in [0, 1]$$

and then selects for branching the i such that $\alpha_i \beta_i$ is maximum. A large α_i favors "most fractional," while a large β_i favors "longest edge." In particular, if all $u_i - l_i$ are equal, then the strategy reduces to selecting the most fractional variable, whereas if one edge is significantly longer than the others, it will necessarily be selected for branching.

By applying this strategy, the resulting two small boxes are as follows:

$$\{\ x \in \Re^n\ :\ l \leq x \leq u - (u_i - \bar{x}_i)e_i\ \},$$

$$\{\ x \in \Re^n\ :\ l + (\bar{x}_i - l_i)e_i \leq x \leq u\ \}.$$

- **Node Selection**. We use a best-bound (breadth-first) strategy for selecting the next node to solve in the branch-and-bound tree.
- **Fathoming Tolerance**. A relative optimality tolerance is used for fathoming. For a given tolerance tol, a node with lower bound L is fathomed if $(U - L)/\max\{1, \frac{1}{2}(|U| + |L|)\} < tol$, where U is the current global upper bound. In our experiments, we set $tol = 10^{-3}$.

3.5.2 Implementation and results

For $n = 20$, we generated 100 instances of random data (Q, c) (entries uniform in $[-1, 1]$, which ensured $Q \not\succeq 0$ in all cases) and solved these instances using the

branch-and-bound scheme outlined above. In particular, all instances were solved three times, each time with a different choice of lower bound calculation and branching strategy. The three choices were:

(i) lower bounds by (SDP_0) and simple branching strategy;

(ii) lower bounds by (SDP_0) and advanced branching strategy;

(iii) lower bounds by (SDP_2) and advanced branching strategy.

(Please refer to the previous subsection for an explanation of the *simple* and *advanced* branching strategies). We will refer to these three choices as *test scenarios*. The goals of analyzing these three particular test scenarios are:

- to compare the two branching strategies via scenarios (i) and (ii) (see Figure 3.1);
- to compare (SDP_0) and (SDP_2) via scenarios (ii) and (iii) (see Figure 3.2).

The algorithm was coded in MATLAB (version 7.3, release 2006b) and all SDP relaxations were setup and solved using YALMIP (Lfberg, 2004) and SeDuMi (version 1.1) (Sturm, 1999). All computations were performed on an Intel Pentium D Processor running at 3.2 GHz with 2,048 KB cache and 4 GB RAM under the Linux operating system.

Figure 3.1 contains a log-log plot depicting the number of nodes required by all instances in test scenarios (i) and (ii). For each of the 100 problem instances, a single point is plotted with its x-coordinate equal to the number of nodes under scenario (i) and its y-coordinate equal to the number of nodes under scenario (ii). Also depicted is the "$y = x$" dotted line, which divides the plot into two regions. In particular, a

point plotted in the lower-right region indicates an instance that required fewer nodes under the advanced branching strategy of scenario (ii). Similar to Figure 3.1, Figure 3.2 compares the number of nodes required under test scenarios (ii) and (iii). Also depicted in a separate plot are the CPU times (in seconds). Both plots contain the $y = x$ dotted line for reference.

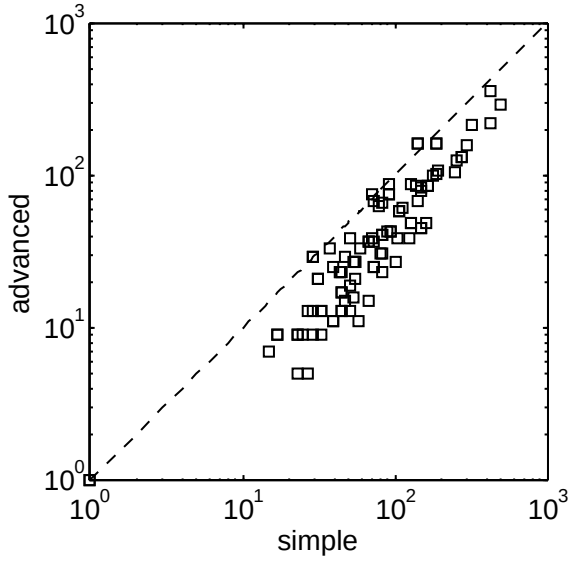

Figure 3.1: Number of nodes required under test scenarios (i) and (ii). This demonstrates that the *advanced* branching strategy reduces the number of nodes significantly compared to the *simple* branching strategy.

Our key interpretations of the figures are as follows:

- Figure 3.1: Since nearly all points are plotted in the lower-right region, the advanced branching strategy is clearly better than the simple branching strategy

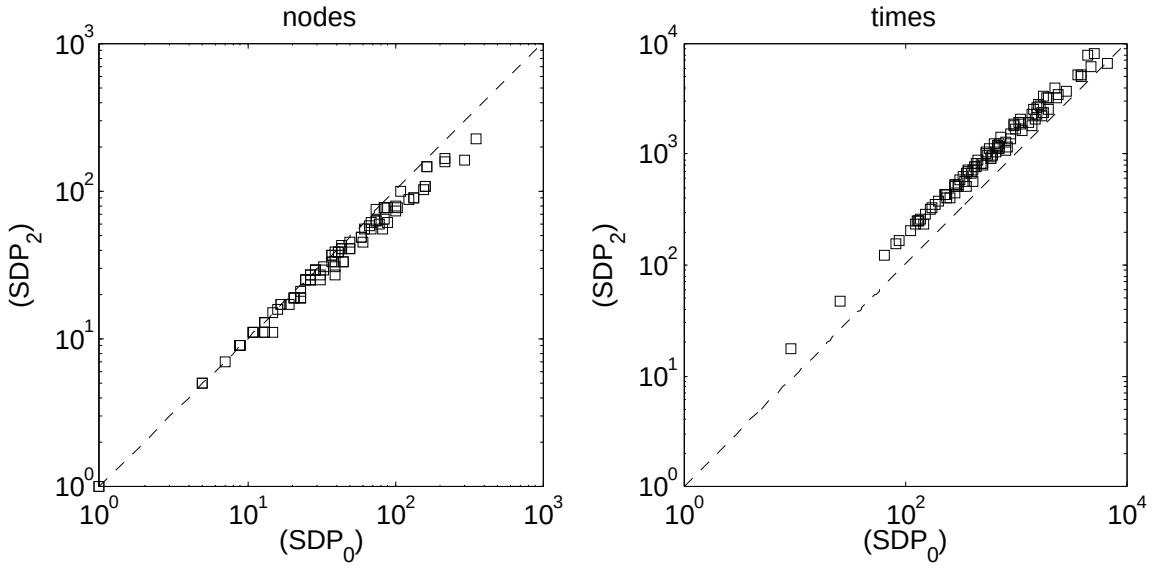

Figure 3.2: Number of nodes and CPU times (seconds) required under test scenarios (ii) and (iii). This demonstrates that (SDP_2) results in fewer nodes compared to (SDP_0). However, the overall CPU time incurred by (SDP_2) is greater.

in terms of number of nodes. In particular, the number of nodes was reduced by more than 44% on average by using the advanced branching strategy. Furthermore, since (SDP_0) was used as the relaxation for lower bounds in both scenarios, the time per node for each instance was essentially the same, so that the reduction in nodes resulted in a real time reduction of about 44% as well.

- Figure 3.2: In all runs, the number of nodes required by (SDP_2) is no more than the number required by (SDP_0), which indicates that (SDP_2) provides stronger lower bounds than (SDP_0). In particular, on average the number of nodes required by (SDP_2) is 21% less than that of (SDP_0). However, the overall CPU times required for the entire branch-and-bound process are higher using

(SDP_2). So (SDP_0) is the overall winner.

In light of Figure 3.2, we wondered if some intelligent combination of (SDP_0) and (SDP_2) during the branch-and-bound procedure might perform better in terms of overall CPU time than using (SDP_0) only. Hopefully, we could reduce the number of nodes significantly — a benefit of (SDP_2) — while keeping the time small— a benefit of (SDP_0). We devised a strategy, which employs (SDP_0) early in the branch-and-bound tree, and then switches to (SDP_2) later in the tree when its stronger bounds may be useful for fathoming. Specifically, our strategy is the following:

> At the beginning of branch-and-bound, all lower bounds are calculated with (SDP_0) by default. At each node after the solution of (SDP_0), the optimal solution (x^*, X^*) is extracted and used to construct $W^*_{B_2B_2}$ and $W^*_{B_3B_3}$ according to (3.16h) and (3.16j), respectively. If either $Q_{B_2B_2} \circ W^*_{B_2B_2}$ or $Q_{B_3B_3} \circ W^*_{B_3B_3}$ are not positive semidefinite, then it follows that (x^*, X^*) cannot be part of a feasible solution for (SDP_2) at that node. In other words, solving (SDP_2) will cut off (x^*, X^*). In these cases, we flag all future descendants of the current node and calculate the lower bound via (SDP_2) for those descendants.

This combination of (SDP_0) was implemented as a new test scenario:

(iv) lower bounds by the above combination of (SDP_0) and (SDP_2) and advanced branching strategy (see Figure 3.3).

Compared to test scenario (ii), test scenario (iv) required 15% fewer nodes on average, which is again a testament to the strength of (SDP_2). In addition, the points in the

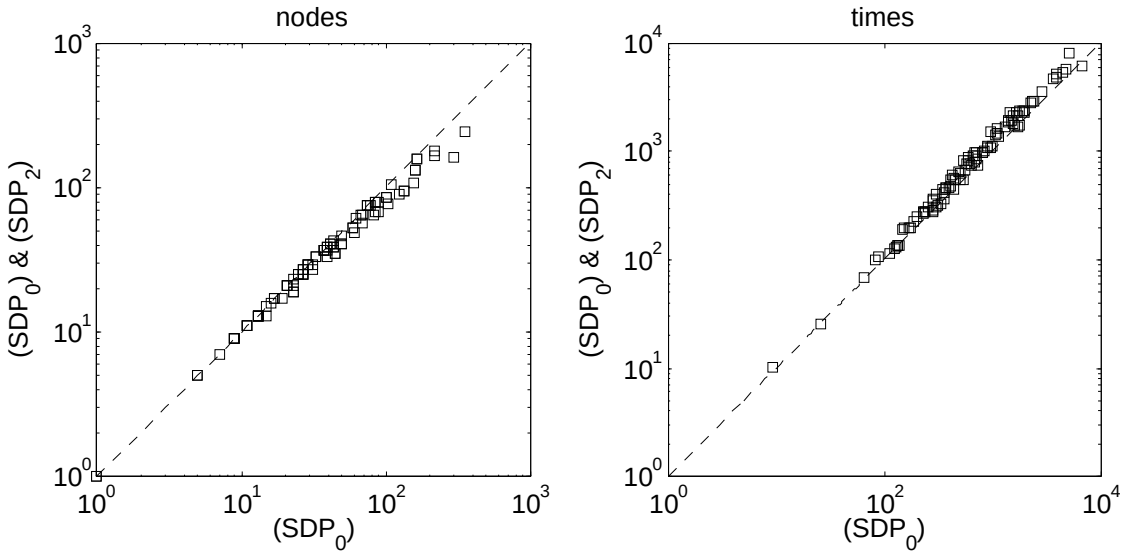

Figure 3.3: Number of nodes and CPU times (seconds) required under test scenarios (ii) and (iv). Compared to scenario (iii) in Figure 3.2, less time is required by scenario (iv), but still scenario (iv) requires more time than scenario (ii).

time plot of Figure 3.3 are shifted closer to the "$y = x$" line compared to the time plot of Figure 3.2, which is an indication of less time used than in scenario (iii). (Keep in mind that both Figures 3.2 and 3.3 share test scenario (ii) as the basis of comparison.) So our strategy of combining (SDP_0) and (SDP_2) was successful in reducing the number of nodes compared to (ii) and reducing the times compared to (iii). In fact, in the time plot, there were actually 6 instances below the "$y = x$" line, indicating that (iv) used less time than (ii) in these cases. However, on average the CPU times for scenario (iv) were still more than (ii), indicating that our strategy was not fully successful as hoped.

3.5.3 Some additional tests

For completeness, we compared (SDP_{12}) with (SDP_0) and (SDP_2) in the context of branch-and-bound. (SDP_{12}) on a smaller box $\{x \in \Re^n : l \leq x \leq u\}$ has all the constraints of (3.16) as well as the first order constraints (3.8c) and (3.8d). In addition, for a particular index i, if $x_i < 1$, then we fix $y_i = 1$; if $x_i > 0$, then we fix $[Qx + c + y]_i = 0$. Both of these rules are based on the complementary slackness condition (3.3c). We conducted the tests for the same 100 problems of size $n = 20$ above with the advanced branching strategy. The results were as follows: (SDP_{12}) required the fewest nodes among all three relaxations but required the longest times. In fact, by using (SDP_{12}) on average the number of nodes is reduced by 65% compared with (SDP_0) and 56% compared with (SDP_2).

We also considered the sensitivity of (SDP_2) with respect to the spectral structure of Q. From each of the randomly generated Q's in the above experiments, which tended to be invertible and well conditioned, we created a new ill-conditioned, indefinite $\tilde{Q}$ by forcing some of Q's eigenvalues to zero. Specifically, let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{20}$ be the eigenvalues of Q and V be the corresponding matrix of eigenvectors. We define

$$\tilde{\lambda}_i = \begin{cases} 0 & \text{if } 6 \leq i \leq 15 \\ \lambda_i & \text{otherwise} \end{cases}$$

and $D := \text{Diag}([\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_{20}]^T)$. Then

$$\tilde{Q} := VDV^T.$$

We found that the new $\tilde{Q}$'s were relatively easier to solve than the original Q's. On average, the number of nodes required to solve the new problems with (SDP_2)

were 25% less than needed to solve the original problems and the times were 86% less. (SDP_0) enjoyed similar node and time improvements on these problems. We unfortunately do not have a hypothesis as to why the $\tilde{Q}$'s were easier to solve, but the fact that they were easier for both (SDP_0) and (SDP_2) would seem to suggest that, in some sense, the problems themselves became easier irrespective of the algorithm.

3.6 Conclusion

In this chapter, we have introduced new semidefinite relaxations of box-constrained quadratic programming: (SDP_{12}) and (SDP_2). (SDP_{12}) is based on relaxing both the first- and second-order necessary optimality conditions; (SDP_2) is similar except that it only incorporates second-order information. (SDP_2) has been our main focus since the first-order conditions have been studied in previous papers. We have compared these two relaxations with a basic semidefinite relaxation (SDP_0) and established the theoretical result that all three relaxations achieve the same optimal value.

Relaxing the standard second-order necessary optimality conditions is one of the main theoretical ideas of this chapter. This task is non-trivial since it implicitly involves knowledge of the active/inactive constraint set at a general point. In the future, it may be possible to extend this technique to other problems, e.g., quadratic programming over the simplex, leading to stronger SDP relaxations in other contexts.

We have also empirically compared (SDP_0) and (SDP_2) in the context of branch-and-bound and demonstrated that (SDP_2) on subdivided boxes is significantly stronger, which indicates that the incorporation of second-order information in SDP

relaxations can help globally solve Box QP. In particular, fewer branch-and-bound nodes are required by (SDP_2) compared to (SDP_0), although overall (SDP_2) uses more time. Future advances in SDP software may allow (SDP_2) to be solved faster, so that the benefits of its node reduction may also be reflected in overall CPU times.

CHAPTER 4
COMBINING FINITE BRANCH-AND-BOUND WITH DOUBLY NONNEGATIVE RELAXATIONS FOR QP

4.1 Introduction

We consider the problem of finding global solutions to a general quadratic programming (QP) problem with linear and bound constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}x^T Hx + f^T x \\ \text{s.t.} \quad & Ax \leq b \\ & A_{eq}\, x = b_{eq} \\ & l \leq x \leq u, \end{aligned} \tag{QP}$$

where $H \in \Re^{n\times n}$, $f \in \Re^n$, $A \in \Re^{m\times n}$, $b \in \Re^m$ and $A_{eq} \in \Re^{m_0\times n}$, $b_{eq} \in \Re^{m_0}$. $l \in \Re^n$ and $u \in \Re^n$ are lower and upper bounds on x, and are allowed to be infinite. However, we assume that all the constraints together imply finite lower and upper bounds on x. We also assume that (QP) has an interior point and A_{eq} has full rank, which will be needed in Section 4.3.2. The particular form of this quadratic program conforms with the input form required by the MATLAB function quadprog, which finds local optimal solutions to (QP). We adopt the same input form as quadprog because our goal is to create a global solver for (QP) using MATLAB as the interface.

We solve (QP) via a finite branch-and-bound method proposed in Burer and Vandenbussche (2008), and the relaxations used in the branch-and-bound scheme are a special semidefinite relaxation (Burer (2010)). We combine these existing technolo-

gies together and utilize them to solve the general quadratic program (QP). Specifically, we would like to reformulate (QP) as the following quadratic program with equality constraints, nonnegative constraints and complementarity conditions:

$$\begin{aligned} \min \quad & \frac{1}{2} x_3^T H_3 x_3 + f_3^T x_3 \\ \text{s.t.} \quad & A_3 x_3 = b_3 \\ & x_3 \geq 0 \\ & \left[x_3 x_3^T \right]_E = 0 \end{aligned} \tag{NQP}$$

where the subscript 3 distinguishes these data and variable from those in (QP). The complementarity condition $[x_3 x_3^T]_E = 0$ encodes the product of the i-th and j-th entry of x_3 is zero for all $(i, j) \in E$. It has been shown recently by Burer (2009) that (NQP) is equivalent to so-called completely positive programs, which optimizes a linear function over the convex cone of completely positive matrices subject to linear constraints. These completely positive programs have natural semidefinite relaxations. Burer (2010) develops a sub-routine that effectively solve such type of semidefinite relaxations, and embed the sub-routine inside branch-and-bound to hence solve (NQP). Also Burer (2010) has implemented such an approach to effectively solve the BoxQP and the quadratic assignment problem. This chapter serves as an extension to this work. In earlier work, the implementation has been tailored to the structure of the particular problem, either the BoxQP or the quadratic assignment problem. Here our goal is to develop a general implementation for (QP) with wider applicability.

The focus of this chapter is on how to model (QP) as (NQP). This includes (i) reformulating (QP) by incorporating the KKT conditions of (QP) into its constraint set (Section 4.3.1), which allows finite branching; (ii) bounding the dual variables associated with the reformulation because the algorithm we will utilize requires finite bounds on the dual variables (Section 4.3.2). The rest of this chapter is organized as follows. In Section 4.2, we briefly review related work in Burer and Vandenbussche (2008) and Burer (2010). In Section 4.4, we detail our preliminary computational experiments on 77 benchmark instances of (QP).

4.2 More Background

In this section, we briefly review the finite branch-and-bound method (Burer and Vandenbussche (2008)) and the SDP relaxation (Burer (2010)) mentioned earlier.

4.2.1 The Finite Branch-and-Bound Method

The finite branch-and-bound method proposed in Burer and Vandenbussche (2008) works by enforcing the first-order KKT conditions through branching. Specifically, the authors start with a general QP (4.1)–(4.2), incorporate its first-order KKT

system (4.3)–(4.5) into the QP's feasible set as follows:

$$\max \quad \frac{1}{2}x^T Q x + c^T x \tag{4.1}$$

$$\text{s.t.} \quad Ax \le b,\ x \ge 0 \tag{4.2}$$

$$A^T y - z = Qx + c,\ (y, z) \ge 0 \tag{4.3}$$

$$\left.\begin{array}{ll} (b - Ax)_j = 0, & \forall j \in F^{(b-Ax)} \\ y_j = 0, & \forall j \in F^y \\ x_i = 0, & \forall i \in F^x \\ z_i = 0, & \forall i \in F^z \end{array}\right\} \tag{4.4}$$

$$F^{(b-Ax)} \cup F^y = \{1, \ldots, m\},\ F^x \cup F^z = \{1, \ldots, n\}, \tag{4.5}$$

where $F^{(b-Ax)}, F^y, F^x, F^x$ are four index sets, and m and n are the row and column dimensions of A. Note that (4.3) is the gradient condition and (4.4)–(4.5) are in fact the complementarity condition. The idea is to relax (4.5) and only enforce partial complementarity condition via (4.4):

$$(b - Ax)_j\, y_j = 0 \ \ \forall j \in F^{(b-Ax)} \cup F^y \subseteq \{1, \ldots, m\}$$

$$x_i\, z_i = 0 \ \ \forall i \in F^x \cup F^z \subseteq \{1, \ldots, n\}.$$

Then branching on a node involves selecting an index $j \in \{1, \ldots, m\} \backslash F^{(b-Ax)} \cup F^y$ or $i \in \{1, \ldots, n\} \backslash F^x \cup F^z$, and creating two children nodes by adding j (suppose j is selected) to $F^{(b-Ax)}$ at one node and adding j to F^y at the other node. By branching this way $F^{(b-Ax)} \cap F^y = \emptyset$ and $F^x \cap F^z = \emptyset$ is maintained at all nodes. For example, the root node is $F^{(b-Ax)}, F^y, F^x, F^x$ all being empty sets. At any leaf node, both (4.4) and (4.5) hold and thus the complementarity condition is satisfied.

At any nodes, (4.1)–(4.4) is still a nonconvex problem and thus a convex relaxation is constructed and then solved to compute upper bounds on the optimal value.

There are many choices for the convex relaxations, and a particular SDP relaxation is constructed in Burer and Vandenbussche (2008). With that SDP relaxation, the authors show that any leaf nodes can be pruned and thus their branch-and-bound scheme is correct and finite.

4.2.2 Doubly Nonnegative Programs

Burer (2009) has shown that (NQP) is equivalent to the so-called completely positive programs, which naturally has the following SDP relaxation (Doubly Nonnegative Programs):

$$\begin{aligned} \min \quad & \frac{1}{2} C \bullet Y \\ \text{s.t.} \quad & A_3 x_3 = b_3, \ \operatorname{diag}(A_3 X A_3^T) = b_3^2 \\ & X_E = 0 \\ & Y = \begin{pmatrix} 1 & x_3^T \\ x_3 & X \end{pmatrix} \succeq 0, \ Y \geq 0, \end{aligned} \tag{DNP}$$

where $C := \begin{pmatrix} 0 & f_3^T \\ f_3 & H_3 \end{pmatrix}$. It is obvious that the derivation of (DNP) is quite different from the SDP relaxations in Burer and Vandenbussche (2008), which is based on (4.1)–(4.4). The major contribution of Burer (2010) is an algorithm that solve (DNP) efficiently. That algorithm requires finite upper bounds on the variable x_3 as input. Therefore, in Section 4.3.2 we will discuss how to bound some dual variables, which are sub-components of x_3 (see Section 4.3.1). We do not review the detail of the algorithm since we can treat it as an available sub-routine that computes valid lower bounds for (DNP) and embed it inside the finite branch-and-bound method.

4.3 Reformulation and Bounding

In this section, we first discuss the several steps required to turn (QP) into (NQP). They involve remodeling all the constraints as equality constraints, shifting and scaling the bounds on the primal variables, and formulating the KKT system of (QP). Then we show how to bound the dual variable that arise when formulating the KKT system.

4.3.1 Reformulation

Without loss of generality, we assume l and u are finite. Otherwise, we can find the assumed finite bounds through LP pre-processing. Specifically, if a lower bound (upper bound) is needed for x_i, we solve the following LP to find l_i (or u_i).

$$\min \{x_i \text{ (or } -x_i) \ : \ Ax \leq b, \ A_{eq}x = b_{eq}, \ l \leq x \leq u\}.$$

First, we reformulate (QP) as a problem with equality and bound constraints only. Define a slack variable $s := b - Ax$ such that $Ax + s = b$ becomes an equality constraint. Obviously, s has computable finite bounds; let us assume that $0 \leq s \leq u_s$. We reformulate (QP) as an equivalent problem (QP_1) without inequality constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}x_1^T H_1 x_1 + f_1^T x_1 \qquad (\text{QP}_1) \\ \text{s.t.} \quad & A_1 x_1 = b_1 \\ & l_1 \leq x_1 \leq u_1, \end{aligned}$$

where

$$H_1 := \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix}, \quad f_1 := \begin{pmatrix} f \\ 0 \end{pmatrix}$$
$$A_1 := \begin{pmatrix} A & I \\ A_{eq} & 0 \end{pmatrix}, \quad b_1 := \begin{pmatrix} b \\ b_{eq} \end{pmatrix}$$
$$x_1 := \begin{pmatrix} x \\ s \end{pmatrix}, \quad l_1 := \begin{pmatrix} l \\ 0 \end{pmatrix}, \quad u_1 := \begin{pmatrix} u \\ u_s \end{pmatrix}.$$

Assume $u_1 - l_1 > 0$, otherwise it is straightforward to remove the fixed variables. Define

$$x_2 := (x_1 - l_1) \circ (u_1 - l_1)^{-1}$$
$$H_2 := \mathrm{Diag}(u_1 - l_1) H_1 \,\mathrm{Diag}(u_1 - l_1)$$
$$f_2 := \mathrm{Diag}(u_1 - l_1)\,(H_1 l_1 + f_1)$$
$$c_2 := \frac{1}{2} l_1^T H_1 l_1 + f_1^T l_1$$
$$A_2 := A_1 \,\mathrm{Diag}(u_1 - l_1)$$
$$b_2 := b_1 - A_1 l_1,$$

where "$\circ$" represents Hadamard product in the first definition. Now we can shift and scale the variable x_1 such that its lower and upper bounds are 0 and e, respectively, where e is a vector of all ones. With this transformation, (QP_1) is equivalent to (QP_2):

$$\min \quad \frac{1}{2} x_2^T H_2 x_2 + f_2^T x_2 + c_2 \tag{QP_2}$$
$$\text{s.t.} \quad A_2 x_2 = b_2 \tag{4.6}$$
$$0 \le x_2 \le e. \tag{4.7}$$

Note that there is now a constant term c_2 in the objective.

Next, we add the first order KKT conditions of (QP_2) to its constraints set. This incorporates more information into the problem and does not change the optimal solution set. However, the number of variables increases. We introduce dual variables y, z and λ for the constraints $A_2x_2 = b_2$, $x_2 \geq 0$ and $x_2 \leq e$, respectively. Then the KKT system is:

$$H_2x_2 + f_2 + \lambda = A_2^T y + z \tag{4.8}$$

$$x_2 \circ z = 0,\ (e - x_2) \circ \lambda = 0 \tag{4.9}$$

$$z \geq 0,\ \lambda \geq 0. \tag{4.10}$$

We would like to pre-calculate finite bounds for the dual variables y, z and λ. Multiplying x_2^T on both sides of (4.8) yields

$$x_2^T H_2 x_2 + f_2^T x_2 + x_2^T \lambda = x_2^T A_2^T y + x_2^T z.$$

Using (4.6) and (4.9), we simplify the above equality to

$$x_2^T H_2 x_2 + f_2^T x_2 + e^T \lambda = b_2^T y. \tag{4.11}$$

We claim that (4.6), (4.7), (4.8), (4.10) and (4.11) imply finite bounds on the dual variables, i.e., there exist finite l_y, u_y, u_z and u_λ such that

$$l_y \leq y \leq u_y$$

$$0 \leq z \leq u_z$$

$$0 \leq \lambda \leq u_\lambda.$$

We defer the proof of the claim to Section 4.3.2.

Without loss of generality, we assume that $u_y - l_y > 0$, $u_z > 0$ and $u_\lambda > 0$[1]. Then we shift the dual variables to its lower bounds and scale their upper bounds to be ones:

$$y_2 := (y - l_y) \circ (u_y - l_y)^{-1}$$

$$z_2 := z \circ u_z^{-1}$$

$$\lambda_2 := \lambda \circ u_\lambda^{-1}.$$

Finally, define a new variable x_3 that packs the original primal and dual variables together and the data that allows us to state (QP_2) in terms of x_3:

$$x_3 := \begin{pmatrix} x_2 \\ s_2 \\ z_2 \\ \lambda_2 \\ y_2 \end{pmatrix}, \quad H_3 := \begin{pmatrix} H_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad f_3 := \begin{pmatrix} f_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$A_3 := \begin{pmatrix} A_2 & 0 & 0 & 0 & 0 \\ -H_2 & 0 & \mathrm{Diag}(u_z) & -\mathrm{Diag}(u_\lambda) & A_2^T \mathrm{Diag}(u_y - l_y) \\ I & I & 0 & 0 & 0 \end{pmatrix}, \quad b_3 := \begin{pmatrix} b_2 \\ f_2 - A_2^T l_y \\ e \end{pmatrix}.$$

With the above definitions, we arrive at the following QP in the form of (NQP):

$$\begin{aligned} \min \quad & \frac{1}{2} x_3^T H_3 x_3 + f_3^T x_3 + c_2 && (\mathrm{QP}_3) \\ \text{s.t.} \quad & A_3 x_3 = b_3 && (4.12) \\ & x_3 \geq 0 \\ & \left[x_3 x_3^T \right]_E = 0 \end{aligned}$$

[1]If the assumption does not hold, then we keep the fixed variables in the KKT system without shifting and scaling them. We do not remove these fixed variables because we need to keep the complementarity condition (4.9).

where E encodes that $x_2 \circ z_2 = 0$ and $s_2 \circ \lambda_2 = 0$. (4.12) now encapsulates the equality and bound constraint of (QP_2) as well as the KKT condition (4.11), and implicitly implies that x_3 is bounded above by e.

4.3.2 Finite Bounds on Dual Variables

Now we prove the claim regarding the boundedness of the dual variables, that is:

Proposition 4.3.1. *(4.6), (4.7), (4.8), (4.10) and (4.11) implies finite bounds on y, z and λ.*

We will use the same proof technique used as in Proposition 2.2 and Proposition 3.1 of Burer and Vandenbussche (2008). Define

$$P := \left\{ x_2 \in \Re^{n+m} : \ A_2 x_2 = b_2, \ 0 \leq x_2 \leq e \right\}.$$

$$R := \left\{ (\triangle\lambda, \triangle z) \geq 0, \triangle y \text{ free} : \ A_2^T \triangle y + \triangle z = \triangle\lambda, \ e^T \triangle\lambda = b_2^T \triangle y \right\}.$$

Similar to the proof in Burer and Vandenbussche (2008), it can be shown that R is the recession cone of (4.6), (4.7), (4.8), (4.10) and (4.11). For interested reader, please refer to Burer and Vandenbussche (2008) for proof technique. We omit the proof here.

Recall that we assume: (i)(QP) has an interior point, i.e., there exists x^0 such that $Ax^0 < b$, $A_{eq}x^0 = b_{eq}$ and $l < x^0 < u$; (ii) A_{eq} has full rank. Given the reformation performed in Section 1.1, (i) implies that P contains an interior point while (ii) implies that A_2 has full rank. In order to prove Proposition 4.3.1, it suffices to show that R only contains trivial point under these two assumptions:

Proof. Consider primal LP $\max\{0 : x_2 \in P\}$ and its dual

$$\min\{e^T\lambda - b^T y : \ A_2^T y + z = \lambda, \ (z, \lambda) \geq 0\}.$$

Duality tells us that $e^T\lambda - b^T y = 0$. By assumption, P contains an interior point x_2^0 such that $0 < x_2^0 < e$. Obviously, x_2^0 is an optimal solution to the primal problem. Complementarity conditions imply that $(z, \lambda) = (0, 0)$ since $0 < x_2^0 < e$. Now the constraint $A_2^T y + z = \lambda$ in the dual problem simplifies to $A_2^T y = 0$. By assumption A_2 has full rank, and thus $y = 0$, which completes the proof. □

To compute l_y, u_y, u_λ, and u_z, we again solve linear programs. Note that among all the constraints that bound the dual variables, (4.11) is nonlinear, and so we use reformulation and convexification technique (Sherali and Tuncbilek (1995)) to relax it. To compute the bounds for the k-th component of y, we solve the following LP:

$$\begin{aligned} \min \ / \ \max \quad & y_k \\ \text{s.t.} \quad & (4.6), \ (4.7), \ (4.8), \ (4.10) \\ & H_2 \bullet X + f_2^T x_2 + e^T\lambda = b_2^T y \\ & 0 \leq X_{i,j} \leq \min\{x_i, x_j\}, \quad \forall (i,j) \\ & 1 - x_i - x_j + X_{i,j} \geq 0, \quad \forall (i,j). \end{aligned} \tag{4.13}$$

We compute the upper bounds for z and λ via solving the same linear programs but modifying the objective functions.

4.3.3 Connection with Section 2

The key idea of Section 2.1 is to branch on complementarity condition. In our case, this is to branch on the complementarity condition encoded in $[x_3 x_3^T]_E = 0$:

$$x_2 \circ z_2 = 0, \qquad s_2 \circ \lambda_2 = 0.$$

At each node of the tree, we enforce partial complementarity similar in fashion to (4.4). Interestingly, (DNP) also has the effect of enforcing the complementarity condition via $X_E = 0$, which is some additional pressure on the complementarity condition that is not present in the SDP relaxation for (4.1)–(4.5).

The fact that x_3 is bounded above by e is crucial when we solving (DNP) at each node of the branch-and-bound tree. Recall that the algorithm to solve (DNP) requires finite upper bounds on x_3.

4.4 Preliminary Computational Experiments

In this section, we document a preliminary test on our branch-and-bound method. In particular, we tested the branch-and-bound scheme on 77 quadratic problems from GLOBALLIB[2] and on several BoxQP problems. The pre-processing and the finite branch-and-bound scheme are implemented in MATLAB. CPLEX is used to solve all the linear programs that arise when bounding the variables. We use the code supplied with Burer (2010) to solve the semidefinite relaxations of (QP_3), where the code was implemented in C.

We compare our method with BARON (Sahinidis and Tawarmalani (2010)),

[2]Available online at http://www.gamsworld.org/global/globallib.htm.

one of the state-of-the-art global solvers, both theoretically and computationally. First, BARON is designed for finding global solutions to general nonlinear and mixed-integer nonlinear programs and while here we only concern general quadratic programming problem. Although both BARON and our technique in essence use the methodology of branch-and-bound, BARON utilizes a rectangular subdivision scheme, which theoretically could produce a tree with infinite number of nodes. In contrast, our method branches on the complementarity conditions, as described in Section 4.2.1. The relaxations used by BARON are based on using *factorable programming techniques* (see references in Sahinidis (2000)) while we use semidefinite relaxations.

Here we do not intend to perform a comprehensive comparison with BARON because our implementation is still preliminary while BARON is a sophisticated software. We only compare with BARON on several instances from GLOBALLIB and on several BoxQP instances, in order to gauge the performance of our method. The computational results of our method are shown in Table 4.1 and 4.3.

The times listed in Table 4.1 are in seconds and were rounded to the nearest integers. For some of the instances, pre-processing fixed all the variables, i.e., the lower bounds obtained via solving linear programs are identical to the upper bounds. Thus the global solutions were returned immediately without any nodes being solved. All the instances in Table 4.1 were solved to within 0.00001 of the global optimal value except st_cqpjk2, where the problem has a poor formulation and caused numerical difficulties (the difference between the returned GUB and the optimal value is 0.189).

We observe that the branch-and-bound scheme has some difficulties on several

Instance	Time	Nodes	Instance	Time	Nodes	Instance	Time	Nodes
ex2_1_1	7	17	st_e26	1	1	st_pan1	3	3
ex2_1_2	2	0	st_fp1	6	17	st_pan2	6	17
ex2_1_3	12	3	st_fp2	2	0	st_ph1	3	1
ex2_1_4	2	0	st_fp3	11	1	st_ph2	4	3
ex2_1_5	20	7	st_fp4	5	3	st_ph3	4	3
ex2_1_6	12	7	st_fp5	20	7	st_ph10	1	0
ex2_1_7	6658	1581	st_fp6	12	7	st_ph11	7	17
ex2_1_8	38	13	st_fp7a	2530	487	st_ph12	9	21
ex2_1_9	3	1	st_fp7b	679	157	st_ph13	13	9
ex2_1_10	87	25	st_fp7c	2023	493	st_ph14	16	13
immun	1631	1787	st_fp7d	963	195	st_ph15	2	1
meanvar	3	3	st_fp7e	6644	1581	st_ph20	4	1
st_bpaf1a	11	1	st_glmp_fp1	3	1	st_phex	2	1
st_bpaf1b	17	7	st_glmp_fp2	10	9	st_qpc_m0	1	1
st_bpk1	3	1	st_glmp_fp3	3	1	st_qpc_m1	3	1
st_bpk2	3	1	st_glmp_kk90	3	1	st_qpc_m3a	11	3
st_bpv2	2	1	st_glmp_kk92	3	1	st_qpc_m3b	15	3
st_bsj2	3	1	st_glmp_kky	6	1	st_qpc_m3c	5	0
st_bsj3	2	3	st_glmp_ss1	5	1	st_qpc_m4	5	0
st_bsj4	15	27	st_glmp_ss2	6	7	st_qpk1	1	1
st_cqpf	2	1	st_ht	4	11	st_qpk2	178	83
st_cqpjk2	2	1	st_iqpbk1	8	5	st_qpk3	12536	1761
st_e22	5	1	st_iqpbk2	8	3	st_rv1	9	5
st_e23	1	1	st_jcbpaf2	26	7	st_rv2	107	19
st_e24	2	1	st_jcbpafex	1	1	st_z	3	1
st_e25	4	1	st_kr	2	1			

Table 4.1: Computational Results of 77 Quadratic Instances in GLOBALLIB. Measures include CPU times (including pre-processing) and the number of nodes in branch-and-bound ("Nodes" column).

of the instances, where it spent more than 1 hour to prove global optimality. We tested these "hard" instances on global optimization solver BARON provided by NEOS sever for optimization[3]. We extracted similar measures from the output of BARON and summarize them in Table 4.2. We acknowledge that BARON was significantly faster on these instances where it only need several seconds to prove global optimality.

We also tested several BoxQP instances (found in Burer and Vandenbussche (2005)) using both BARON and our branch-and-bound algorithm, referred to as

[3]Available at http://www-neos.mcs.anl.gov/.

Instance	Time	Nodes
ex2_1_7	0	1
immun	0.07	17
st_fp7a	0.02	1
st_fp7b	0.06	1
st_fp7c	0.05	1
st_fp7d	0.12	1
st_fp7e	0.42	15
st_qpk3	1.65	235

Table 4.2: BARON Results on Several "Hard" Problems

QUADPROGBB. We report the results in Table 4.3. Among these instances, QUADPROGBB significantly outperforms BARON on two of the instances, 'spar050-030-3' and 'spar050-040-3'. If we compare the number of nodes needed by BARON and that of QUADPROGBB, we see that in general the latter requires many fewer nodes. Note that we break down the total CPU time of QUADPROG into pre-processing time (t_{pre}) and the time spent by branch-and-bound ($t_{B\&B}$). The total pre-processing time takes about 79% of the total CPU time on these instances, which indicates that we need a better technique to compute the bounds of the dual variables and cut down the associated time.

We point out several limitations of the above comparisons: (i) BARON has very sophisticated pre-processing techniques that tightens the constraints (see Sahinidis and Tawarmalani (2010) for more details) while the pre-processing we presented did not tighten the constraints set; (ii) the computing environment of running BARON is unknown because we called it through NEOS sever, which makes the comparison even more difficult; (iii) BARON is in C and Fortran while a large component of the

	BARON		QUADPROGBB			
Instance	Nodes	Time	Nodes	t_{pre}	$t_{B\&B}$	Time
spar020-100-1	31	1.7	3	4	3	7
spar030-080-2	51	4.09	1	14	2	16
spar040-030-1	1	1.18	3	44	12	56
spar050-030-1	153	37.13	1	136	14	150
spar050-030-2	901	82.03	3	138	32	170
spar050-030-3	1547	**334.93**	5	134	51	**185**
spar050-040-3	1169	**1000.03**	3	143	22	**165**
spar060-020-1	1121	392.78	3	358	40	398
spar060-020-3	1917	454.27	13	383	188	571

Table 4.3: Comparison of BARON and QUADPROGBB on several BoxQP instances. We break QUADPROGBB's time into pre-processing time (t_{pre}) and branch-and-bound time ($t_{B\&B}$). All of the time are in seconds.

branch-and-bound algorithm is implemented in MATLAB.

4.5 Conclusion and Future Work

This chapter investigates how to integrate a finite branch-and-bound method (Section 4.2.1) and the so-called doubly nonnegative programs (Section 4.2.2) to solve (QP). The idea is to re-formulate (QP) as (NQP), which naturally admits (DNP) as relaxations. The reformulation involves explicitly incorporating the first-order KKT system into the constraints set, which allows us to branch on the complementarity condition. We addressed the issue of bounding the dual variables associated with the KKT system. We have performed preliminary test on our implementation of this technique on 77 QP problems from GLOBALLIB and 9 BoxQP instances. The results show the correctness of this method. Comparing our results with global solver

BARON indicates our method has room for improvement. We discuss future directions in the subsequent paragraphs.

On the theoretical side, we need to understand the SDP relaxation (DNP) better. How does (DNP) compare with those SDP relaxations obtained by directly relaxing (QP) in terms of strength? The reformulation performed in Section 4.3 that leads to (DNP) increases the size of the problem significantly, which should be justified by a stronger SDP relaxation.

On the computational side, it would be interesting to investigate methods that improve the performance of our technique. First, we could include techniques that tighten the feasible set of (QP), reducing the number of constraints and the number of variables. Reduced constraints and variables will result in a smaller KKT system and also fewer complementarity conditions to check. Second, as indicated by the results of Table 4.3, it is important to have a more time efficient way to estimate the bounds for the variables x, y, z and λ as the pre-processing time are primarily spent on solving linear programs like (4.13). Third, we could investigate other node selection rules in branch-and-bound. Currently, the node selection rule is to select the most violated complementarity and branch on the associated index. We could consider node selection rules that take into account the structure of the KKT system formulated, in the hope of reducing the total number of nodes.

CHAPTER 5
A FIRST-ORDER SMOOTHING TECHNIQUE FOR A CLASS OF LARGE-SCALE LPS

5.1 Introduction

In this chapter, we investigate a first-order smoothing technique to solve large-scale instances of the following linear programming (LP) problem:

$$\begin{aligned} \min \quad & c^T\alpha + w^T\xi \qquad \text{(P)} \\ \text{s.t.} \quad & A\alpha - b \leq \xi \\ & \alpha,\ \xi \geq 0 \\ & \alpha_{\mathcal{B}} \leq d, \end{aligned}$$

where $\alpha \in \Re^n$ and $\xi \in \Re^m$ are the decision vectors and $A \in \Re^{m\times n}$, $b \in \Re^m$, $c \in \Re^n_+$ and $w \in \Re^m_+$ are the data. $\mathcal{B}$ is an index subset of $[n]$. We assume that $c_{\mathcal{B}} = 0$ and the corresponding part of α has to be bounded above by a vector $d > 0$. The set $\mathcal{B}$ may be empty, in which case c is a positive vector. This problem's optimal value is bounded below by zero and thus has an optimal solution. ξ could be interpreted as an error, allowing some of the constraints $A\alpha - b \leq 0$ to be violated. The term $w^T\xi$ in the objective serves to minimize such violations. If ξ is fixed to 0, then the above formulation would be similar to the standard form linear programming (LP).

The motivation for studying (P) is the observation that several machine learning problems are LPs with the structure of (P). One such example is the linear program based ranking formulation introduced by Ataman (2007) to rank instances

with binary outputs. The 1-norm support vector machine (1-norm SVM, Zhu et al. (2003); Mangasarian (2006)) can also be modeled as (P). We are interested in solving large instances of (P) because of two reasons: (i) small instances can be solved efficiently by current LP solvers and memory is not an issue; for problems where A is large and dense, using simplex or interior-point methods might not be feasible due to memory limits; (ii) the machine learning problems above are often applied to large data sets making A large. In applications that involve kernel matrices, such as the popular RBF kernels (Hsu et al. (2003)), A is large and often completely dense.

Currently there are several approaches to solve (P). As mentioned above, standard approaches are the simplex method or interior-point methods. For applications in machine learning and data mining, however, (P) is often too large. For example, Ataman (2007) reported that a moderately-sized ranking problem formulated as (P) would cause CPLEX to run out of memory on standard PC. Mangasarian (2006) formulated a class of LPs, which includes (P) as a special case. The author posed the problem as the unconstrained minimization of a convex differentiable piecewise-quadratic objective function and solved it using a generalized Newton method. Each iteration of his method requires computing the inverse of an $m \times m$ matrix, which is prohibitive for large scale problems.

Another approach is to treat (P) as the equivalent nonsmooth problem

$$\begin{aligned} \min \quad & c^T\alpha + w^T\left(A\alpha - b\right)^+ \\ \text{s.t.} \quad & \alpha \geq 0,\ \alpha_{\mathcal{B}} \leq d \end{aligned} \tag{NS}$$

where $(A\alpha - b)^+$ denotes the nonnegative part of the vector $A\alpha - b$, and then to

solve (NS) using nonsmooth techniques such as the standard subgradient method. Compared with the simplex method and interior point methods, the subgradient method requires much less memory (basically the memory to store A). It also has very cheap computational costs at each iteration (basically A times a vector). The main drawback of the subgradient method is slow convergence: its worst case iteration complexity is $O(1/\epsilon^2)$ (Nesterov (2004)), where ϵ is the absolute error tolerance. Another drawback of the subgradient method is that it does not admit a systematic way of estimating the dual objective value of a general non-smooth problem[1], i.e., we only know the primal objective during each iteration. Therefore, it is difficult to measure the progress of the subgradient method.

In this chapter, we propose to use Nesterov's first-order smoothing method (Nesterov, 2005b,a) to solve (P). The smoothing method has a worst case iteration complexity of $O(1/\epsilon)$, a magnitude faster than the subgradient method. At the same time, its computational cost per iteration and memory requirements are comparable to the subgradient method's. Researchers have successfully applied the smoothing method to several large-scale problems. Hoda et al. (2007) apply the smoothing technique to approximate Nash equilibria of large sequential two-player, zero-sum games. In a related paper, Gilpin et al. (2007) demonstrate that a tailored implementation of the smoothing technique allows them to approximate the Nash equilibria of sequential

[1]Although there exist several schemes to recover the primal feasible solution of a quite general LP when solving the Lagrangian dual of such LP via the subgradient optimization methods (see Sherali and Choi (1996); Anstreicher and Wolsey (2009)), what the subgradient method solves here is the "primal" problem and not quite the same.

games four orders of magnitude larger than previous algorithms. Smoothing technique also have applications in semidefinite programming and general convex optimization (d'Aspremont, 2008; Nesterov, 2007; Lan et al., 2009).

Nesterov's first-order smoothing method applies to the following generic problem:

$$\min\{f(x) \ : \ x \in Q\}, \tag{5.1}$$

where f is a continuous convex function with a certain structure and Q is a (simple) compact convex set (see Section 2). In order to apply the smoothing method, we will need some transformation of (P) or (NS) to get a compact convex feasible region since one pre-requisite of the smoothing technique is the assumption of bounded feasible sets. We show that the unbounded feasible region of (P) can be bounded from the knowledge of an upper bound θ on the optimal value θ^* of (P). We prove that the iteration complexity of the smoothing method is proportional to θ/ϵ. We then show that it is possible to adjust the smoothing algorithm such that it dynamically updates θ as it obtains better bounds on θ^*. To the best of our knowledge, this is the first application of the smoothing technique to solve problems with unbounded feasible sets.

This chapter is organized as follows. We summarize the major ingredients of the smoothing technique in Section 2 to facilitate later discussion. In Section 3, we show how to convert (P) into a problem of the form (5.1) and specify the major components of the smoothing technique, such as the choice of the so-called prox-functions and derivations of various parameters. In Section 4, we demonstrate how

to update the upper bound θ and discuss how it speeds up the smoothing method. Finally, in Section 5, we present two machine learning applications of smoothing technique and compare it with two existing methods.

5.2 Nesterov's Smoothing Method

In this section, we review some of the major ingredients of Nesterov's smoothing technique: the excessive gap condition and convergence rate. We focus on the concepts and results that will be used in our study and leave out technical details. First, we introduce some notations and definitions that will be used throughout this chapter.

5.2.1 Notation and terminology

Let E denote a finite-dimensional real vector space, possibly with an index. This space is equipped with a norm $\|\cdot\|$, which has the same index as the corresponding space. Let A be a linear operator: $E_1 \to E_2$. Define the operator norm of A, induced by the norms $\|\cdot\|_1$ and $\|\cdot\|_2$, as

$$\|A\|_{1,2} = \max_{\|x\|_1=1} \max_{\|u\|_2=1} \langle Ax, u\rangle,$$

where $\langle\cdot,\cdot\rangle$ refers to inner product. Note $\|\cdot\|_1$ and $\|\cdot\|_2$ do not necessarily represent the standard l_1-norm and l_2-norm; the subscripts are indices only. This operator norm has the following property,

$$\|A\|_{1,2} = \max_{\|x\|_1=1} \|Ax\|_2^* = \max_{\|u\|_2=1} \|A^T u\|_1^*,$$

where $\|\cdot\|^*$ denotes the dual norm associated with $\|\cdot\|$ and is defined as

$$\|z\|^* := \max\{z^T x : \|x\| \leq 1\}.$$

We use A_i to denote the i-th row of A and A^j the j-th column of A. For a vector $v \in \Re^n$, $v^{-1} \in \Re^n$ denotes the vector whose components are the inverse of the components of v. We let Diag(v) represent the diagonal matrix with diagonal v. Finally, we use e to represent a vector of all ones. The dimension of e may differ but should be clear from the context.

5.2.2 A primal-dual smoothing method

Consider the following functions $f(x)$ and $\phi(u)$:

$$f(x) = \hat{f}(x) + \max_{u \in Q_2}\{\langle \bar{A}x, u\rangle - \hat{\phi}(u)\} \tag{5.2}$$

$$\phi(u) = -\hat{\phi}(u) + \min_{x \in Q_1}\{\langle \bar{A}x, u\rangle + \hat{f}(x)\}, \tag{5.3}$$

where Q_1 and Q_2 are simple compact convex sets in finite dimensional Euclidean spaces E_1 and E_2, respectively, $\bar{A}$ is a linear operator mapping E_1 to E_2, and $\hat{f}(x)$ and $\hat{\phi}(u)$ are continuous convex functions on Q_1 and Q_2, respectively. Thus, $f(x)$ is convex and $\phi(u)$ is concave, but they are *not* necessarily differentiable. For any $\bar{x} \in Q_1$ and $\bar{u} \in Q_2$ we have

$$\phi(\bar{u}) \leq f(\bar{x}) \tag{5.4}$$

because

$$\begin{aligned}\phi(\bar{u}) &= -\hat{\phi}(\bar{u}) + \min_{x\in Q_1}\{\langle \bar{A}x, \bar{u}\rangle + \hat{f}(x)\} \\ &\leq -\hat{\phi}(\bar{u}) + \langle \bar{A}\bar{x}, \bar{u}\rangle + \hat{f}(\bar{x}) \\ &\leq \hat{f}(\bar{x}) + \max_{u\in Q_2}\{\langle \bar{A}\bar{x}, u\rangle - \hat{\phi}(u)\} \\ &= f(\bar{x}).\end{aligned}$$

The smoothing technique uses a primal-dual approach to simultaneously solve (5.1) and

$$\max\{\phi(u) : u \in Q_2\}, \tag{5.5}$$

where by Fenchel duality (see Borwein and Lewis (2006)) (5.5) is the dual problem of (5.1) and there is no duality gap.

The primal-dual approach does not directly deal with (5.2) and (5.3). Instead it works with the following "smoothed" versions of (5.2) and (5.3):

$$f_{\mu_2}(x) = \hat{f}(x) + \max_{u\in Q_2}\{\langle \bar{A}x, u\rangle - \hat{\phi}(u) - \mu_2 d_2(u)\} \tag{5.6}$$

$$\phi_{\mu_1}(u) = -\hat{\phi}(u) + \min_{x\in Q_1}\{\langle \bar{A}x, u\rangle + \hat{f}(x) + \mu_1 d_1(x)\}, \tag{5.7}$$

where μ_i is a positive *smoothness* parameter, and $d_i(\cdot)$ is a *prox-function* on the set Q_i, which means $d_i(\cdot)$ is continuous and strongly convex on Q_i, $i = 1, 2$. A strongly convex function $d(\cdot)$ on a closed convex set Q has the following property for some $\sigma > 0$:

$$d(x) \geq d(x_*) + \frac{1}{2}\sigma\|x - x_*\|^2, \quad x \in Q, \tag{5.8}$$

where $x_* = \arg\min_{x \in Q} d(x)$. The purpose of introducing these prox-functions is to smooth $f(x)$ and $\phi(u)$. The resultant $f_{\mu_2}(x)$ and $\phi_{\mu_1}(u)$ are differentiable and their gradients are Lipschitz-continuous. When μ_1 and μ_2 are small, $f_{\mu_2} \approx f$ and $\phi_{\mu_1} \approx \phi$.

By definition we have $f_{\mu_2}(x) \leq f(x)$ and $\phi(u) \leq \phi_{\mu_1}(u)$. For sufficiently large μ_1 and μ_2, it can be shown that there exists some $\bar{x} \in Q_1$ and $\bar{u} \in Q_2$ satisfying the following excessive gap condition (EGC):

$$f_{\mu_2}(\bar{x}) \leq \phi_{\mu_1}(\bar{u}). \tag{EGC}$$

(EGC) is like a switched version of (5.4), which ensures that the primal-dual gap is bounded above, as stated in the following lemma.

Lemma 5.2.1. *(Nesterov (2005a)) Let $\bar{x} \in Q_1$ and $\bar{u} \in Q_2$ satisfy (EGC). Then*

$$0 \leq f(\bar{x}) - \phi(\bar{u}) \leq \mu_1 D_1 + \mu_2 D_2 \tag{5.9}$$

where $D_1 := \max_{x \in Q_1} d_1(x)$ and $D_2 := \max_{u \in Q_2} d_2(u)$.

In addition to Lemma 5.2.1, the smoothing technique features three other important ingredients:

(i) a procedure that calculates an initial $(x^0, u^0, \mu_1^0, \mu_2^0)$ satisfying (EGC), i.e., $f_{\mu_2^0}(x^0) \leq \phi_{\mu_1^0}(u^0)$.

(ii) given $(x^k, u^k, \mu_1^k, \mu_2^k)$ satisfying (EGC), a procedure that generates $(x^{k+1}, u^{k+1}, \mu_1^{k+1}, \mu_2^{k+1})$ satisfying (EGC) as well.

(iii) the procedure in (ii) ensures that $\mu_i^{k+1} \leq \mu_i^k,\ i = 1, 2$, where one of the two inequalities is strict, and also $\mu_i^k \longrightarrow 0$, $i = 1, 2$.

(i) and (ii) generate a sequence $\{(x^k, u^k, \mu_1^k, \mu_2^k)\}$ that satisfies (EGC) for each k, and at the same time shrinks μ_1^k or μ_2^k in each iteration. Because of Lemma 5.2.1 and (iii), the primal-dual gap is going to zero as μ_1^k and μ_2^k go to zero, that is,

$$0 \leq f(x^k) - \phi(u^k) \leq \mu_1^k D_1 + \mu_2^k D_2 \longrightarrow 0. \tag{5.10}$$

As long as (EGC) is maintained, (5.10) will hold for all k.

Theorem 5.2.2. *(Nesterov (2005a)) Given $\epsilon > 0$, there is an algorithm based on the smoothing technique that produces a pair $(x^N, u^N) \in Q_1 \times Q_2$ such that*

$$0 \leq f(x^N) - \phi(u^N) \leq \epsilon$$

in

$$N = \frac{4\|\bar{A}\|_{1,2}}{\epsilon}\sqrt{\frac{D_1 D_2}{\sigma_1 \sigma_2}}$$

iterations.

In each iteration of the algorithm, we need to update $(x^k, u^k, \mu_1^k, \mu_2^k)$, which requires solving several sub-problems in the form of the inner max problem in (5.6) or the inner min problem in (5.7). Therefore, the solutions of the these max and min problems should be easily computable. We omit the generic scheme here; we will describe the specific algorithm with respect to our problem in Section 5.3.3.

5.3 Applying the Smoothing Technique

In this section, we show how to convert (NS) into the standard form required by the smoothing technique. After the conversion, we specify each ingredient including our choices of the prox-functions, the calculation of parameters for the smoothing

technique, and the iteration complexity for solving (NS). In the end, we detail the algorithm.

5.3.1 Reformulation

The primal feasible set is unbounded. However, employing the smoothing technique requires the primal/dual feasible set of (NS) to be bounded and simple. We first show that the primal feasible set can be bounded, and then through a simple change of variables, we transform the feasible set of (NS) into the combination of a simplex, denoted as $\triangle := \{x \geq 0 : \ e^T x = 1\}$, and a box, denoted as $\square := \{x : 0 \leq x \leq e\}$. In Section 5.3.2, we show how to find closed-form solutions to some sub-problems of the smoothing technique, which rely on the fact that the feasible sets are simple ones built upon the simplex and the box.

Suppose θ is a valid upper bound on the optimal value of (P), i.e.,

$$c^T\alpha^* + w^T(A\alpha^* - b)^+ \leq \theta,$$

where α^* denotes an optimal solution of (P). Recall parameter w is nonnegative. Thus, $w^T(A\alpha^* - b)^+$ is nonnegative, and the following inequality, which bounds the primal feasible set, is satisfied by the optimal solution α^*:

$$c^T\alpha \leq \theta. \tag{5.11}$$

Recall that $\alpha_{\mathcal{B}} \leq d$, $c_{\mathcal{B}} = 0$ and the remaining elements of c are positive, i.e. $c_{\bar{\mathcal{B}}} > 0$, where $\bar{B} := \{1, \ldots, n\}\backslash\mathcal{B}$. Define a new variable $x \in \Re^{n+1}_+$ that has one larger

dimension than α:

$$x_{\mathcal{B}} := \mathrm{Diag}\left(d^{-1}\right)\alpha_{\mathcal{B}} \in \square$$
$$x_{\mathcal{S}} := \frac{1}{\theta}\begin{pmatrix}\mathrm{Diag}\left(c_{\bar{\mathcal{B}}}\right)\alpha_{\bar{\mathcal{B}}} \\ \theta - c^T\alpha\end{pmatrix} \in \triangle,$$

where the first part of x is inside a box with dimension $|\mathcal{B}|$, the second part is inside a simplex with dimension $n{+}1{-}|\mathcal{B}|$, and the index set $\mathcal{S} := \{1\ldots n{+}1\}\backslash\mathcal{B} = \bar{B}\cup\{n{+}1\}$.

To match the structure of the primal objective (5.2), we re-state $w^T(A\alpha - b)^+$ as a maximization problem as follows

$$\begin{aligned} w^T(A\alpha - b)^+ &= e^T\Big(w \circ (A\alpha - b)\Big)^+ \\ &= \max_{u\in\square}\Big\{\langle w \circ (A\alpha - b), u\rangle\Big\} \\ &= \max_{u\in\square}\Big\{\langle w \circ (A\alpha)\,, u\rangle - (w\circ b)^T u\Big\}, \end{aligned}$$

where $\circ$ refers to Hadamard product. In addition, we define $\hat{A}$, $\hat{e}$ and $\hat{b}$ by properly scaling the original data:

$$\hat{A}_{\bullet\mathcal{S}} := \begin{pmatrix}\mathrm{Diag}(w)A_{\bullet\bar{\mathcal{B}}}\,\mathrm{Diag}(c_{\bar{\mathcal{B}}}^{-1}) & 0\end{pmatrix}$$
$$\hat{A}_{\bullet\mathcal{B}} := \frac{1}{\theta}\,\mathrm{Diag}(w)A_{\bullet\mathcal{B}}\,\mathrm{Diag}(d)$$
$$\hat{b} := w \circ b$$
$$\hat{e}_{\mathcal{B}} := 0, \hat{e}_{\bar{\mathcal{B}}} := e, \hat{e}_{n+1} := 0.$$

With the above definition, the objective function of (NS) can be stated in terms of x:

$$c^T\alpha + w^T(A\alpha - b)^+ = \theta\left[\hat{e}^T x + \max_{u\in\square}\left\{\langle\hat{A}x, u\rangle - \frac{1}{\theta}\hat{b}^T u\right\}\right], \tag{5.12}$$

and thus (NS) is equivalent to the following problem

$$\min\left\{f(x;\theta) := \hat{e}^T x + \max_{u\in\square}\left\{\langle \hat{A}x, u\rangle - \frac{1}{\theta}\hat{b}^T u\right\} :\ x_{\mathcal{S}} \in \Delta,\ x_{\mathcal{B}} \in \square\right\}. \tag{SP}$$

Based on the "primal-dual" structure of (5.2) and (5.3), we immediately have the "dual" problem

$$\max\left\{\phi(u;\theta) := -\frac{1}{\theta}\hat{b}^T u + \min_{x_{\mathcal{S}}\in\Delta, x_{\mathcal{B}}\in\square}\left\{\langle \hat{A}x, u\rangle + \hat{e}^T x\right\} :\ u \in \square\right\}. \tag{SD}$$

The smoothing technique we described in Section 5.2.2 can be used to solve (SP) and (SD). For any primal feasible solution $\bar{x}$ obtained from the smoothing technique, we can recover a feasible solution $\bar{\alpha}$ to (NS). Note that by (5.12), the objective $f(\bar{x};\theta)$ needs to be scaled by θ in order to recover the objective value of (NS). This property will have an influence on the iteration complexity of the smoothing technique. In particular, the primal error (the difference between the primal objective value and the optimal value) of (NS) is the primal error of (SP) scaled by θ.

Lemma 5.3.1. *Let $p(\alpha)$ denote the primal objective function of (NS), and x^* be an optimal solution of (SP). Recall that θ^* denotes the optimal value of (NS). Suppose $\bar{x}$ is feasible to (SP) and $\bar{\alpha}$ is the corresponding feasible solution to (NS). Then we have:*

$$p(\bar{\alpha}) - \theta^* = \theta\Big(f(\bar{x};\theta) - f(x^*;\theta)\Big). \tag{5.13}$$

Proof. By (5.12), $p(\bar{\alpha}) = \theta f(\bar{x};\theta)$ and $\theta^* = \theta f(x^*;\theta)$. □

5.3.2 Specifications

In this subsection, we discuss in detail each ingredient of the smoothing technique. The choices of norm and the prox-function are critical decisions. We select the $L1$-norm and the entropy distance function as the prox-function (see Nesterov (2005b)) for the primal space, and the $L2$-norm and a "distance squared" quadratic function for the dual space:

$$\|x\|_1 := \sum_{i=1}^{n} |x^{(i)}|, \qquad d_1(x) := \ln(|\mathcal{S}|) + |\mathcal{B}| \cdot \exp(-1) + \sum_{i=1}^{n+1} x^{(i)} \ln x^{(i)} \tag{5.14}$$

$$\|u\|_2 := \sqrt{\sum_{j=1}^{m} (u^{(j)})^2}, \quad d_2(u) := \frac{1}{2} \sum_{j=1}^{m} \left(u^{(j)} - \frac{1}{2} \right)^2. \tag{5.15}$$

Under the above choices, we calculate the parameters that determine the iteration complexity of the smoothing technique:

$$D_1 = \max_x \{ d_1(x) : x_{\mathcal{S}} \in \Delta,\ x_{\mathcal{B}} \in \square \} = \ln(|\mathcal{S}|) + |\mathcal{B}| \cdot \exp(-1), \quad \sigma_1 = 1,$$

the derivation of which can be found in Lemma 3 of Nesterov (2005b). Prox-function $d_1(\cdot)$ achieves its minimum at $x_0 = e/(n+1)$, where $d_1(x_0) = 0$. It is easy to verify that

$$D_2 = \max_u \{ d_2(u) : u \in \square_m \} = \frac{m}{8}, \quad \sigma_2 = 1,$$

and $d_2(\cdot)$ achieves its minimum at $u_0 = \frac{1}{2}e$. The operator norm of $\hat{A}$ is thus

$$\begin{aligned}
\|\hat{A}\|_{1,2} &= \max_u \left\{ \|\hat{A}^T u\|_1^* : \|u\|_2 = 1 \right\} \\
&= \max_u \left\{ \max_{i \in \{1,\ldots,n+1\}} \left\{ \langle \hat{A}^i, u \rangle \right\} : \|u\|_2 = 1 \right\} \\
&= \max_{i \in \{1,\ldots,n\}} \left\{ \max_u \left\{ \langle \hat{A}^i, u \rangle : \|u\|_2 = 1 \right\} \right\} \\
&= \max_{i \in \{1,\ldots,n\}} \|\hat{A}^i\|_2,
\end{aligned}$$

where $\hat{A}^i$ denotes the i-th column of $\hat{A}$. The second equality follows because the dual norm of the l_1-norm is the l_∞-norm, and the third equality follows from the fact that $\hat{A}$'s last column is zero.

Now with all the parameters computed, we are ready to state the iteration complexity of solving (SP) and (SD).

Proposition 5.3.2. *Using Nesterov's smoothing technique, for any $\epsilon > 0$, we obtain a pair of solutions (x^N, u^N) to (SP) and (SD) such that*

$$0 \leq f(x^N; \theta) - \phi(u^N; \theta) \leq \epsilon$$

in

$$N(\theta) = \frac{1}{\epsilon} \left(\max_{i \in \{1,\ldots,n\}} \|\hat{A}^i\|_2 \right) \sqrt{2 \left[\ln(|\mathcal{S}|) + |\mathcal{B}| \cdot \exp(-1) \right] \cdot m} \tag{5.16}$$

iterations.

Proof. The result is obtained by applying Theorem 5.2.2. □

Since $A_{\bullet \mathcal{B}}$ depends on θ, it is possible that $\max_{i \in \{1,\ldots,n\}} \|\hat{A}^i\|_2$, the maximum column norm of $\hat{A}$, depends on θ. If indeed, then the number of iterations N is

proportional to $1/\sqrt{\theta}$, and so we write the total number of iterations as a function of θ.

We comment that different combinations of norm and prox-function other than (5.14) and (5.15) may lead to different parameter values and thus different iteration complexities. We considered several alternate choices of norms and prox-functions for the dual space, and the choice (5.15) gives us the lowest iteration complexity among those considered. For example, we could choose the L_1-norm and the same prox-function as $d_1(x)$ for the dual space; the resultant iteration complexity is $O(\sqrt{m})$ times larger than (5.16), which is much worse than the current one if m is large.

In Proposition 5.3.2, the iteration complexity is stated with respect to (SP). Now we state the iteration complexity with respect to (NS).

Proposition 5.3.3. *Using Nesterov's smoothing technique, for any $\epsilon > 0$, we obtain a solution $\bar{\alpha}$ to (NS) such that $0 \leq p(\bar{\alpha}) - \theta^* \leq \epsilon$ in*

$$N' := \theta N(\theta)$$

iterations, where $N(\theta)$ is given by (5.16) and θ^* is the optimal value.

Proof. By Proposition 5.3.2, in N' iterations, we obtain a solution $(\bar{x}, \bar{u})$ such that the primal-dual gap is small enough:

$$0 \leq f(\bar{x}; \theta) - \phi(\bar{u}; \theta) \leq \frac{\epsilon}{\theta}.$$

We then can construct $\bar{\alpha}$ feasible to (NS). Thus, by (5.13), we have

$$0 \leq p(\bar{\alpha}) - \theta^* = \theta\Big(f(\bar{x}; \theta) - f(x^*; \theta)\Big) \leq \theta\Big(f(\bar{x}; \theta) - \phi(\bar{u}; \theta)\Big) \leq \epsilon.$$

□

For notational convenience, we drop the θ from $N(\theta)$. If N does not depend on θ, then N' is proportional to θ; otherwise, N' is proportional to $\sqrt{\theta}$ since $N(\theta)$ is proportional to $1/\sqrt{\theta}$. In both cases, the smaller θ is, the better the iteration complexity. Thus, obtaining a good bound on θ is important for solving (NS). We will discuss in Section 5.4 a strategy that dynamically updates θ within the framework of the smoothing technique and subsequently reduces the iteration complexity for solving (NS).

Next we discuss the sub-problems associated with our choice of prox-functions. These sub-problems will be solved repeatedly in the algorithm presented in Section 5.3.3, and thus it is important to have closed-form solutions to them. The sub-problems are the min and max problems presented within the following smoothed versions of primal and dual objective functions; recall (5.6) and (5.7):

$$f_{\mu_2}(x;\theta) = \hat{e}^T x + \max_{u\in\square_m} \left\{ \langle \hat{A}x, u\rangle - \frac{1}{\theta}\hat{b}^T u - \mu_2 d_2(u) \right\}$$

$$\phi_{\mu_1}(u;\theta) = -\frac{1}{\theta}\hat{b}^T u + \min_{x_{\mathcal{S}}\in\Delta,\ x_{\mathcal{B}}\in\square} \left\{ \langle \hat{A}x, u\rangle + \hat{e}^T x + \mu_1 d_1(x) \right\},$$

where $d_1(x)$ and $d_2(u)$ are given in (5.14) and (5.15). Both the sub-problems have closed-form solutions as described in the following two lemmas.

Lemma 5.3.4. *Assume $\mu > 0$. The solution of the problem*

$$\min_{x_{\mathcal{S}}\in\Delta,\ x_{\mathcal{B}}\in\square} \left\{ -\sum_{i=1}^{n} s^{(i)}x^{(i)} + \mu \sum_{i=1}^{n} x^{(i)} \ln x^{(i)} \right\}$$

is given by

$$x^{(i)} = \begin{cases} \frac{\exp\left(s^{(i)}/\mu\right)}{\sum_{j=1}^{|\mathcal{J}|}\exp\left(s^{(j)}/\mu\right)} & , i \in \mathcal{S} \\ \mathrm{proj}_{[0,1]}\exp\left(s^{(i)}/\mu\right) & , i \in \mathcal{B} \end{cases},$$

where $\mathrm{proj}_{[0,1]}(y)$ *projects* y *to the nearest point between* 0 *and* 1.

Proof. The objective function is separable in $\mathcal{S}$ and $\mathcal{B}$. For optimizing this function over the simplex, see Lemma 4 in Nesterov (2005b). For optimizing this function over the box, we compute the point at which the first-order derivative vanishes and then project that point back to the feasible region. □

Lemma 5.3.5. *Assuming* $\mu > 0$*, the solution to the problem*

$$\min_{u \in \square_m} \left\{ -\sum_{j=1}^{m} s^{(j)} u^{(j)} + \mu \sum_{j=1}^{m} \left(u^{(j)} - \frac{1}{2} \right)^2 \right\}$$

is given by

$$u^{(j)} = \mathrm{proj}_{[0,1]}\left(\frac{1 + s^{(j)}/\mu}{2} \right), \qquad j = 1, \ldots, m,$$

where $\mathrm{proj}_{[0,1]}(y)$ *projects* y *to the nearest point between* 0 *and* 1.

Proof. Observe that

$$-\sum_{j=1}^{m} s^{(j)} u^{(j)} + \mu \sum_{j=1}^{m} \left(u^{(j)} - \frac{1}{2} \right)^2 = \mu \sum_{j=1}^{m} \left(\left(u^{(j)}\right)^2 - \left(s^{(j)}/\mu + 1\right) u^{(j)} + \frac{1}{4} \right).$$

Now the problem reduces to solving m one-dimensional quadratic problems:

$$\min_{0 \le u^{(j)} \le 1} \left(u^{(j)}\right)^2 - \left(s^{(j)}/\mu + 1\right) u^{(j)} + \frac{1}{4}, \quad j = 1, \ldots, m,$$

whose solutions are given as above. □

5.3.3 Algorithm

The algorithmic scheme presented by Nesterov (2005a,b) is generic and thus problem-specific parameters for our problem have been calculated in Section 5.3.1 and 5.3.2. In this subsection, we explicitly state the scheme with respect to (SP) and (SD). Define $\texttt{sargmin}(d_i, \cdot) : \Re^n \to Q_i$ as

$$\texttt{sargmin}(d_i, s) := \arg\min_{x \in Q_i} \left\{ -s^T x + d_i(x) \right\}$$

for $i = 1,\ 2$. In this algorithm, there are three functions: (i) INITIAL initializes all the parameters and the primal-dual solution (x^0, u^0); (ii) UPDATE1 is the primal update; (iii) UPDATE2 is the dual update. UPDATE1 and UPDATE2 are symmetric but entail different sub-problems.

Algorithm 5.1 INITIAL

Input: Data $(\epsilon, m, n, \hat{A}, \hat{b}, \hat{e}, \theta)$

Output: Initialized parameters (μ_1^0, μ_2^0) and solutions (x^0, u^0) that satisfy (EGC)

1: $D_1 = \ln(n+1)$, $D_2 = m/8$, $\sigma_1 = \sigma_2 = 1$, $\|\hat{A}\|_{1,2} = \max_{i \in \{1,\ldots,n\}} \|\hat{A}^i\|_2$

2: $\mu_1^0 = 2\,\|\hat{A}\|_{1,2}\sqrt{\frac{D_2}{\sigma_1\sigma_2 D_1}}$, $\mu_2^0 = \|\hat{A}\|_{1,2}\sqrt{\frac{D_1}{\sigma_1\sigma_2 D_2}}$

3: $\bar{x} = \texttt{sargmin}(d_1, 0)$

4: $u^0 = \texttt{sargmin}\left(d_2, \frac{1}{\mu_2^0}(\hat{A}\bar{x} - \hat{b}/\theta)\right)$

5: $x^0 = \texttt{sargmin}\left(d_1, \ln(\bar{x}) + e - \frac{\mu_2^0}{\|\hat{A}\|_{1,2}^2}(\hat{A}^T u^0 + \hat{e})\right)$

Primal update when k is even:

Algorithm 5.2 UPDATE1: primal update

Input: Current solution (x, u) and parameters $(\mu_1, \mu_2, \tau, \theta)$

Output: $(x^+, u^+, \mu_1^+, \mu_2^+)$ that satisfy (EGC)

1: $\bar{x} = \texttt{sargmin}\left(d_1, -\frac{1}{\mu_1}(\hat{A}^T u + \hat{e})\right)$

2: $\hat{x} = (1-\tau)x + \tau\bar{x}$

3: $\bar{u} = \texttt{sargmin}(d_2, \frac{1}{\mu_2}(\hat{A}\,\hat{x} - \hat{b}/\theta))$

4: $\hat{x} = \texttt{sargmin}\left(d_1, \ln(\bar{x}) + e - \frac{\tau}{(1-\tau)\mu_1}(\hat{A}^T\bar{u} + \hat{e})\right)$

5: $x^+ = (1-\tau)x + \tau\hat{x}$

6: $u^+ = (1-\tau)u + \tau\bar{u}$

7: $\mu_1^+ = (1-\tau)\mu_1$, $\mu_2^+ = \mu_2$

Dual update when k is odd:

Algorithm 5.3 UPDATE2: dual update

Input: Current solution (x, u) and parameters $(\mu_1, \mu_2, \tau, \theta)$

Output: $(x^+, u^+, \mu_1^+, \mu_2^+)$ that satisfy (EGC)

1: $\bar{u} = \texttt{sargmin}(d_2, \frac{1}{\mu_2}(\hat{A}\,x - \hat{b}/\theta))$

2: $\hat{u} = (1-\tau)u + \tau\bar{u}$

3: $\bar{x} = \texttt{sargmin}(d_1, -\frac{1}{\mu_1}(\hat{A}^T\hat{u} + \hat{e}))$

4: $\hat{u} = \texttt{sargmin}(d_2, \frac{\tau}{(1-\tau)\mu_2}(\hat{A}\,\bar{x} - \hat{b}/\theta) + \bar{u} - \frac{1}{2}e)$

5: $x^+ = (1-\tau)x + \tau\bar{x}$

6: $u^+ = (1-\tau)u + \tau\hat{u}$

7: $\mu_2^+ = (1-\tau)\mu_2$, $\mu_1^+ = \mu_1$

The SMOOTH algorithm is then as follows:

Algorithm 5.4 SMOOTH

Input: (i) Data $(m, n, \hat{A}, \hat{b}, \hat{e}, \theta)$; (ii) Subroutines `sargmin`$(d_1, \cdot)$ and `sargmin`$(d_2, \cdot)$

Output: $(x^{N'}, u^{N'})$

1: $(x^0, u^0, \mu_1^0, \mu_2^0) =$ INITIAL $(m, n, \hat{A}, \hat{b}, \hat{e}, \theta)$
2: **for** $k = 0, 1, \ldots, N' - 1$ **do**
3: $\quad \tau = \frac{2}{k+3}$
4: $\quad$ **if** k is even **then**
5: $\quad\quad (x^{k+1}, u^{k+1}, \mu_1^{k+1}, \mu_2^{k+1}) =$ UPDATE1$(x^k, u^k, \mu_1^k, \mu_2^k, \tau, \theta)$
6: $\quad$ **else**
7: $\quad\quad (x^{k+1}, u^{k+1}, \mu_1^{k+1}, \mu_2^{k+1}) =$ UPDATE2$(x^k, u^k, \mu_1^k, \mu_2^k, \tau, \theta)$

We have the following comments regarding the SMOOTH algorithm:

- According to Lemma 5.3.4 and Lemma 5.3.5, the sub-problems have closed-form solutions and can be solved very quickly. The most time-consuming operations are thus the matrix-vector multiplications $\hat{A}\bar{x}$ and $\hat{A}^T\bar{u}$.
- In the above algorithm, θ is treated as an input parameter for UPDATE1 and UPDATE2. We will discuss a simple procedure that dynamically updates θ in the next section and the above algorithm will be valid even with changing values of θ.

5.4 Speeding Up the Convergence

As shown in Proposition 5.3.3, the iteration complexity of obtaining an ϵ-solution of (NS) is proportional to θ/ϵ, where θ is an upper bound on the optimal value of (NS). As an input parameter, the smaller θ is, the better the iteration complexity,

and the best possible θ is the optimal value θ^*. We obtain new information about θ^* as the smooth algorithm progresses; in particular, the improved primal objective value gives a better upper bound on θ^*. To take advantage of this information, we consider updating θ dynamically within Algorithm 5.4.

Suppose we have a better bound $\theta^+ \in [\theta^*, \theta)$ available after running Algorithm 5.4 for K iterations, where $K < N' = \theta N$ and N, defined in (5.16), depends on the accuracy ϵ. With θ^+ on hand, it may be worthwhile to restart the algorithm and input θ^+ for the new run, if the number of iterations required by the new run plus the number of iterations have been run is smaller than the iteration estimate under θ, i.e., $K + \theta^+ N < \theta N$ holds. Even better than this, it turns out we can improve the iteration complexity to $\max\{K, \theta^+ N\}$ without restarting the algorithm, provided that the condition (EGC) is carefully maintained when updating the parameter θ to θ^+. To achieve the result, we need Lemma 5.4.1 below.

Let $\theta > \theta^*$ be the initial upper bound on the optimal value. Let k be the iteration count. Then intuitively if the error tolerance ϵ is small enough, there exists some k such that $f(x^k; \theta) < 1$ since $f(x^*; \theta) = \theta^*/\theta < 1$. Define $\theta^k = \theta f(x^k; \theta)$. Based on the relationship (5.13), we know $\theta^k \geq \theta^*$, and so θ^k is a valid upper bound on θ^*. The following lemma shows the existence of $\theta^+ \in (\theta^*, \theta)$ such that $(x^k, u^k, \mu_1^k, \mu_2^k)$ satisfies (EGC) with respect to θ^+.

Lemma 5.4.1. *Suppose feasible solutions $(x^k, u^k, \mu_1^k, \mu_2^k)$ satisfy (EGC) strictly with respect to θ, i.e., $f_{\mu_2^k}(x^k; \theta) < \phi_{\mu_1^k}(u^k; \theta)$. If in addition $f(x^k; \theta) < 1$, then there exists*

$\theta^+ \in (\theta^*, \theta)$ such that $(x^k, u^k, \mu_1^k, \mu_2^k)$ also satisfy (EGC) for θ^+, i.e.,

$$f_{\mu_2^k}(x^k; \theta^+) \leq \phi_{\mu_1^k}(u^k; \theta^+). \tag{5.17}$$

Proof. $f(x^k; \theta) < 1$ and (5.13) together imply $\theta^k \in [\theta^*, \theta)$. If

$$f_{\mu_2^k}(x^k; \theta^k) < \phi_{\mu_1^k}(u^k; \theta^k),$$

simply let $\theta^+ = \theta^k$. Otherwise, if the above inequality does not hold, by the fact that $f_{\mu_2^k}(x^k; \theta)$ and $\phi_{\mu_1^k}(u^k; \theta)$ are continuous functions of θ, there exists $\theta^+ \in (\theta^k, \theta)$ such that (x^k, u^k) satisfies (EGC) for μ_1^k, μ_2^k, i.e., (5.17) holds. □

For the solution x^k to (SP), we can recover a solution α^k to (NS). From now on, we use p^k to represent $p(\alpha^k)$ for notational convenience. By (5.10), we have

$$f(x^k; \theta) - \phi(u^k; \theta) \leq \mu_1^k D_1 + \mu_2^k D_2, \quad k = 1, 2, \ldots,$$

where $\left\{\mu_1^k D_1 + \mu_2^k D_2\right\}_{k=1}^{\infty}$ is independent of θ. Define $U^k := \mu_1^k D_1 + \mu_2^k D_2$. Thus by (5.13), the primal error at iteration k is bounded above by θU^k because

$$p^k - \theta^* = \theta\Big(f(x^k; \theta) - f(x^*; \theta)\Big) \leq \theta\Big(f(x^k; \theta) - \phi(u^k; \theta)\Big) \leq \theta U^k, \quad k = 1, 2, \ldots \tag{5.18}$$

On the other hand, Lemma 5.4.1 and Lemma 5.2.1 imply

$$f(x_1^k; \theta^+) - \phi(u_1^k; \theta^+) \leq U^k$$

since the excessive gap condition (EGC) holds for θ^+. It follows from (5.12) that $p^k = \theta^+ f(x^k; \theta^+)$. Then we have

$$p^k - \theta^* \leq \theta^+\Big(f(x_1^k; \theta^+) - \phi(u_1^k; \theta^+)\big] \leq \theta^+ U^k, \tag{5.19}$$

where the first inequality follows because $\theta^+\phi(u_1^k;\theta^+)$ is a valid dual value and thus is a lower bound on θ^*. Comparing (5.18) and (5.19), we see that the upper bound on the primal error at iteration k has been improved from $\theta\, U^k$ to $\theta^+\, U^k$. Since the smoothing technique guarantees that (EGC) is maintained in the next iteration as long as it holds at the current iteration, switching θ to θ^+ means we can bound the primal error as follows:

$$p^l - \theta^* \leq \theta^+ U^l, \quad \forall l = k, k+1, \ldots$$

In other words, updating θ with θ^+ helps speed up the convergence as the subsequent primal error is bounded above by the new sequence $\left\{\theta^+ U^l\right\}_{l=k}^{\infty}$. Note that the solutions generated in subsequent iterations $\left\{(x^l, u^l)\right\}_{l=k+1}^{\infty}$ would be different than those generated with parameter θ because the parameter θ^+ affects the subsequent solution generated by UPDATE1 and UPDATE2, which has θ^+ as input parameter. Based on the above analysis, we have the following proposition:

Proposition 5.4.2. *If a new upper bound $\theta^+ \in [\theta^*, \theta)$ is available for Algorithm 5.4 and satisfies (5.17) for $k = K$, the number of iterations required to obtain an ϵ-solution to (NS) reduces from θN to* $\max\{K,\ \theta^+ N\}$.

We have the following comments:

- We demonstrated the existence of a θ^+ in Lemma 5.4.1 but do not have a closed-form formula for it. One simple procedure of obtaining θ^+ is as follows: periodically check the conditions of Lemma 5.4.1; then do the following:

  (i) $\theta^+ := \theta f(x^k;\theta)$

(ii) While $f_{\mu_2}(\bar{x};\theta^+) > \phi_{\mu_1}(\bar{u};\theta^+)$

$\theta^+ := 1/2\ (\theta^+ + \theta)$.

(i) and (ii) will not affect other parts of the SMOOTH algorithm and can be put into the algorithm conveniently.

- In order to maintain the excessive gap condition, it is easy to see that θ^+ should be chosen in a neighborhood of θ and so the reduction of the number of iterations by updating θ may not be big. However, the procedure of updating θ can be performed repeatedly. As the algorithm converges, the updated parameter value becomes better and better approximation of θ^* and thus the cumulative improvements might be significant.

We close this section with an example that illustrates the effects of dynamically updating θ. This example's data is created using the linear programming based ranking formulation from Ataman (2007) and the dataset "Dermatology" from Asuncion and Newman (2007). The resulting data matrix A in (P) has dimension 6760×358. We did three runs of the SMOOTH algorithm with the following variants:

(i) initial input of trivial $\theta = w^T(-b)^+$; run the SMOOTH algorithm without updating θ.

(ii) initial input of trivial $\theta = w^T(-b)^+$; run the SMOOTH algorithm while dynamically updating θ.

(iii) initial input of $\theta = \theta^*$, where the optimal value θ^* has been calculated using a standard LP solver; there is no update on θ because the bound is already optimal.

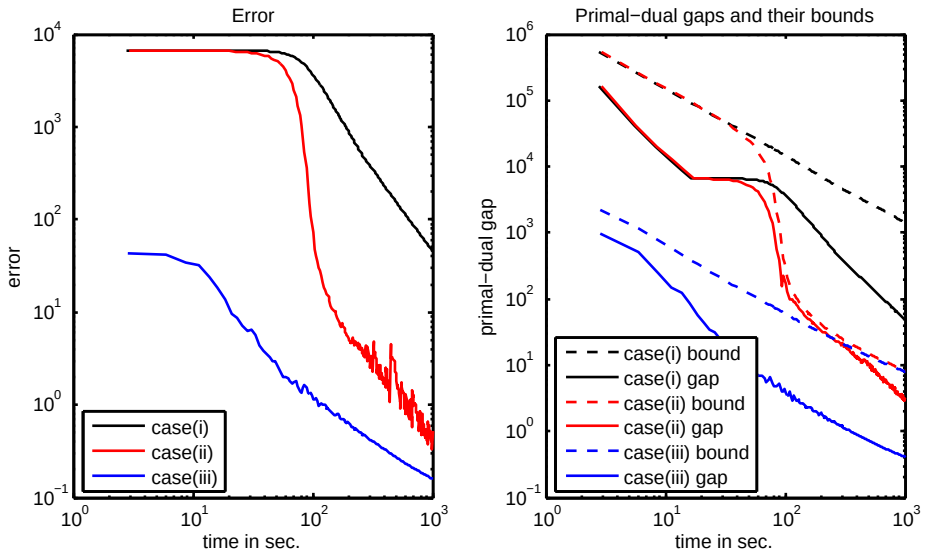

Figure 5.1: Comparison of running SMOOTH algorithm under cases (i) – (iii). The left subplot shows the change of the error $p(\alpha) - \theta^*$ over time; the right subplot shows how the primal-dual gap changes over time and compares it with the corresponding upper bound

Figure 5.1 shows the results in log-log scale. The first subplot shows how the primal error $p^k - \theta^*$ change over time. It can be seen from the figure that dynamically updating θ reduces the error significantly and its performance is almost as good as case (iii). We also comment that in this particular example $\theta^* \ll w^T(-b)^+$ and thus the starting value of the error for case (iii) is far smaller than that of cases (i) and (ii). Note that (i) and (ii) are identical until the point where θ is updated. The second subplot demonstrates how the primal-dual gap $\theta\Big(f(x^k;\theta) - \phi(u^k;\theta)\Big)$ changes over time for the three cases. It also plots the upper bound θU^k to give us a visual illustration of how the upper bound affects the primal-dual gap. The

solid lines depict the primal-dual gap under these cases and the dotted lines are the corresponding upper bounds. Note that for this example, the upper bound associated with case (ii) changes over time because θ is being updated and as time goes on this bound is very close to the bound of case (iii).

5.5 Applications and Computational Experiments

In this section, we study the computational performance of Algorithm 5.4 by comparing it with the subgradient optimization method and a generalized Newton method proposed by Mangasarian (2006) on two machine learning applications, respectively. The first application is a linear programming based ranking problem (LPR); the second application is 1-norm support vector machines (see Mangasarian (2006)).

5.5.1 Linear Programming Ranking Problem

The linear programming based ranking method (Ataman (2007)) is a new ranking method for problems with binary outputs and is reported to perform better than SVM-based ranking algorithms. At its core, it is the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{l \in X} \alpha_l + C \sum_{i \in X^+,\ j \in X^-} w_{i,j} \xi_{i,j} \qquad \text{(LPR)} \\ \text{s.t.} \quad & \sum_{l \in X} y_l \left[k(x_i, x_l) - k(x_j, x_l) \right] \alpha_l \geq 1 - \xi_{i,j} \quad \forall\ i \in X^+, j \in X^- \\ & \alpha \geq 0, \xi \geq 0, \end{aligned}$$

where $\alpha \in \Re^{|X|}$ and $\xi \in \Re^{|X^+|\times|X^-|}$ are the decision vectors and all others are data. It is assumed that $C > 0$. The entries of A are calculated as follows:

$$A_{i\times j,l} = -y_l \left[k(x_i, x_l) - k(x_j, x_l)\right], \quad \forall\ i \times j \in X^+ \times X^-, l \in X$$

where $(x_l, y_l), \forall\ l \in X$ is an instance in the training set X, $y_l \in \{1, -1\}$ is the class label for that instance, X^+, X^- represent the set of points with positive,and negative labels, respectively, and $k(\cdot, \cdot)$ is a chosen kernel function, for example, RBF kernel function. Define

$$m := |X^+| \times |X^-|$$

$$n := |X| = |X^+| + |X^-|,$$

and so $A \in \Re^{m\times n}$. Now LPR can be modeled by (P) with data

$$c = e \in \Re^n, \quad w = Ce \in \Re^m, \quad b = -e \in \Re^m.$$

Since every entry of A is the difference of two kernel functions, A is usually extremely dense.

In this computational study, we will use 14 datasets from Asuncion and Newman (2007) and prepare them in the same way as in Ataman (2007). Table 5.1 describes the dimensions and percentage of non-zeros in A. The last column of Table 5.1 shows the storage size of A in megabytes for each instance. We point out that among the problems in Table 5.1, 'cancer' and 'diabetes' cannot be solved by CPLEX (via either primal or dual LP formulation) on a machine of 4GB RAM without some other special treatment of the memory.

Name	m	n	Non-zeros(%)	Size(MB)
wine	6240	178	11%	0.2
iris	5000	150	100%	5.4
glass	5365	214	100%	8.3
ntyroid	5550	215	98%	7.3
sonar	10767	208	100%	16.4
derma	6760	358	93%	6.6
heart	18000	270	100%	32.9
ecoli	14768	336	100%	36.3
spectf	24638	351	93%	20.0
ion	28350	351	100%	72.7
liver	29000	345	100%	60.6
boston	21984	506	100%	50.2
cancer	75684	569	100%	314.6
diabetes	134000	768	96%	409.1

Table 5.1: Dimensions of matrix A and its percentage of non-zeros (instances are ordered increasingly by the number of non-zeros)

As mentioned in introduction, it is not easy to obtain a primal-dual gap for the subgradient method. Thus, we compare the two method under two circumstances:

- For smaller problems where the optimal value can be easily obtained, e.g., via solving (P) directly, we first compute the optimal value θ^* and then supply this information to both methods. In particular, we use θ^* as the initial upper bound for the smoothing technique; for the subgradient method, we employ the Polyak's step size rule (Boyd and Mutapcic, 2008), which requires θ^*. There are two purposes of doing this: first, both methods achieve the best possible convergence rate when supplied with the optimal values as described and it is interesting to understand how they perform under the idealized situation; second, the primal error, i.e., the difference between the current primal objective

value and the optimal value, being reduced to within an error tolerance serves as a convenient stopping criteria since the subgradient method does not have a convenient stopping criteria. The stopping criteria for both methods is that the best found objective value is within $\epsilon = 1$. Also we impose a time limit of 18000 seconds.

- For large data sets where θ^* is unavailable, we compare the best objective values found by these two methods for running them for a fixed amount of time (18000 seconds). In this case, the method that obtains better objective values has a faster convergence rate. Now we allow the smoothing technique to dynamically update θ, and we use $\theta = p(0) = w^T(-b)^+$ as the initial value. For the subgradient method, we adopt the code from Ataman (2007), which is an implementation of the incremental subgradient methods in Nedic and Bertsekas (2001).

Table 5.2 present the experiment results. The smoothing technique performs better than the subgradient method on 10 out of the 14 instances. For the smaller data sets, the smoothing technique was significantly faster on four smaller instances: iris, glass, ntyroid, and heart. For larger instances where both ran approximately 18000 seconds, the smoothing technqiue was able to find better objective values, with the only exception of spectf.

We also plot the primal errors of both methods versus the CPU time for the data set glass, to illustrate that the convergence results here are consistent with what the theory says. Figure 5.2 shows that the smoothing technique started with a worser

	Time (in sec.)			Best Obj.	
Dataset	SMOOTH	SUBG.	θ^*	SMOOTH	SUBG.
wine	111	39	77.82	78.82	78.76
iris	1104	7001	3317.72	3318.72	3318.72
glass	1296	18001	2850.45	2851.45	2852.00
ntyroid	1232	4153	1694.65	1695.65	1695.65
sonar	2136	1055	191.85	192.85	192.85
derma	119	7	28.71	29.67	29.36
heart	5687	18004	1240.92	1241.92	1249.52
ecoli	18000	18004	8421.05	8422.60	8453.03
spectf	18003	18004	1580.99	1586.30	1582.94
ion	18000	18001	2578.49	2583.15	2634.61
liver	18001	18006	11339.9	11370.30	11641.30
boston	18001	18001	889.88	890.94	899.03
cancer	18000	18000	—	17989.09	27520.79
diabetes	18000	18000	—	14327.73	28070.29

Table 5.2: Comparison of Algorithm 5.4 (SMOOTH) and the subgradient method (SUBG.) when applied to 14 linear ranking problems. Times are in seconds and rounded to the nearest integers.

primal objective value than the subgradient method, but has a faster convergence rate. This figure together with the results in Table 5.2 also suggest that the subgradient method could be a better choice for applications where finding a good solution quickly is important.

5.5.2 1-Norm Support Vector Machines

Support Vector Machines (SVMs) are popular machine learning techniques for classification and regression. There are 1-norm SVMs and 2-norm SVMs. Both types of SVMs have the same constraints with objectives having the form of $loss + penalty$, but the norms used in the penalty part are different. There are many algorithms

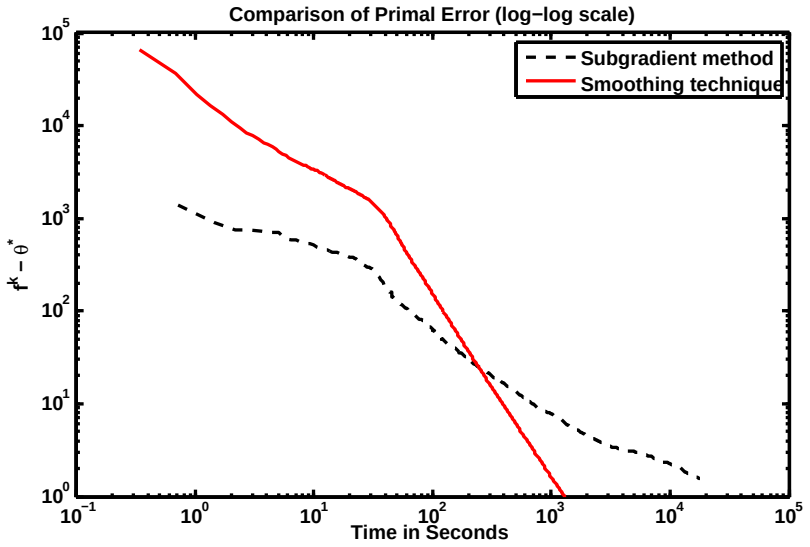

Figure 5.2: Primal errors versus time (log-log scale) for the smoothing technique and the subgradient method.

available for solving large-scale 2-norm SVMs and (see Woodsend and Gondzio (2009) for references and literature review). Existing solution approaches for 1-norm SVMs include linear programming techniques and a series of similar generalized Newton methods (Fung and Mangasarian (2004) and Mangasarian (2006)), which solve small size problems very fast and reliably (Mangasarian (2006)). Thus here we mainly concern large-scale instances that cannot be readily solved by off-the-shelf LP due to the sizes of these problems.

In this computational study, we compare the performances of Algorithm 5.4 and the generalized Newton method proposed in Mangasarian (2006) on several very large data sets. The latter method has similar memory requirement as the smoothing technique, and so we are mainly concerned with the speed and the reliability of the

two schemes. Both these methods are implemented as MATLAB programs, where the latter is adopted from the author's web site[2].

We selected six standard data sets (adult, covtype, usps, ijcnn1, w5a[3] and magic[4]) and randomly sampled 8000 or the maximum number of available data points, whichever is smaller, from these data sets. We then created six large RBF kernel matrixes using the sampled data points[5]. We present the computational results in Table 5.3. The stopping criteria for the smoothing technique is the relative primal-dual gap r as defined below being smaller than 0.1%:

$$r := \frac{p - d}{\max\{1, 1/2(|p| + |d|)\}}, \tag{5.20}$$

where p and d represent the primal and dual objective value, respectively.

We observe that SMOOTH finished satisfying the stopping criteria for all instances except 'usps', where the relative primal-dual gap is 21%. For NEWTONLP, only 'usps' and 'magic' are finished satisfying the stopping criteria, i.e. the norm of the gradient is smaller than 1e−5. Also we observe that NEWTONLP behaves unpredictably as there is no trend of diminishing in terms of the norm of the gradient

[2]http://www.cs.wisc.edu/dmi/svm/lpsvm/. The code on this web site is for Fung and Mangasarian (2004). Since the methods in Fung and Mangasarian (2004) and Mangasarian (2006) are similar, a few minor changes can turn the code of the former to an implementation of the latter.

[3]Available at http://www.csie.ntu.edu.tw/∼ cjlin/libsvmtools/datasets/

[4]Available at http://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope

[5]The best parameter combination (C, γ) is determined beforehand through a 10-fold cross-validation on smaller data sets, where γ is the parameter of the RBF kernel (see Hsu et al. (2003)). Then γ was used to create the large RBF kernel matrices and C was used as the penalty parameter in the objective.

	Time (in sec.)		Best Objectives	
Dataset	SMOOTH	NEWTONLP	SMOOTH	NEWTONLP
adult	18000	18000	1.24e+02	1.47e+06
covtype	9676	18000	1.78e+02	2.01e+07
usps	18000	17560	3.01e+03	2.39e+03
ijcnn1	181	18000	1.19e+04	4.56e+06
w5a	181	18000	1.64e+03	2.77e+07
magic	898	7227	7.00e−02	7.27e−02

Table 5.3: Comparison of Algorithm 5.4 (SMOOTH) and the generalized Newton scheme (NEWTONLP) in terms of CPU times and the best objective values found. The stopping criteria for SMOOTH is the relative primal-dual gap r, defined in (5.20), is smaller than 0.1%; the stopping criteria for NEWTONLP is that the norm of the gradient is smaller than 1e−5. The time limit is set to be 18000 seconds.

in the generalized Newton method, which results in that the best objective values found by NEWTONLP are much worse than those found by SMOOTH in four of the instances. Overall, SMOOTH is a more reliable method compared with NEWTONLP for large problems.

5.6 Conclusion

In this chapter, we have developed a first-order smoothing technique for solving (P) and the equivalent (NS). To the best of our knowledge, this is the first application of Nesterov's smoothing technique to LPs with unbounded feasible set. We show that the iteration complexity of this smoothing technique depends on the parameter θ, which arises when bounding the feasible set. We estimate θ to be an upper bound on θ^*, the optimal value of (NS). Since smaller θ means better iteration complexity,

we have designed a scheme that dynamically updates the value of θ as the algorithm obtains more information about θ^*, resulting in faster convergence. This idea may be extended to other convex nonsmooth problems with unbounded feasible set.

This smoothing technique is designed for large-scale instances of (P). We have applied the smoothing technique to two large-scale problems in machine learning: the linear programming ranking problem and 1-norm support vector machines. We demonstrate the effectiveness of our technique by comparing it with two existing methods, the subgradient method and the generalized Newton method, respectively. Our computational experience also indicates the smoothing technique is more robust than the generalized Newton Method.

CHAPTER 6
CONCLUSION

In this thesis, we present new convex relaxations for several classes of problems in the field of nonconvex optimization, and a smoothing technique for large-scale linear programming problems. Linear programming relaxations have been primary considerations for obtaining bounds on nonconvex optimization problems under a branch-and-bound framework because they are well-understood tools. With the development of convex programming and the emergence of SOCP and SDP as standard problem classes in convex optimization, we try to develop new relaxations based on new advances in convex programming and hope to obtain tighter bounds. In this conclusion, we highlight the major contributions of our exploration in this line of research and list them by chapters.

In Chapter 2, we study a sequential relaxation technique for 0-1 integer programs. We use p-balls to relax 0-1 integer points, and then perform a lift-and-project procedure to obtain relaxations based on p-order cone programming, including SOCP as a special case. Previously, only LP and SDP have been used to construct the relaxations in the higher dimensional space. Thus, this work contributes to the literature by using SOCP as the relaxation tool. We establish that this new relaxation has many nice properties, e.g., it eliminates any fractional extreme points of a general convex set. We show that repeated applications of the lift-and-project procedure, resulting in a sequential relaxation technique, generates the convex hull of the feasible set asymptotically. In addition, our method generalizes and subsumes several existing

methods, including Lovász-Schrijver relaxation based on LP. Geometrically, p-balls can be used to relax many nonconvex sets, and thus the approach of constructing p-order cone relaxations based on p-balls could be potentially useful to other nonconvex optimization problems.

In Chapter 3, we study SDP relaxations for box-constrained quadratic programs. Unlike previous SDP relaxations based on first-order KTT conditions, our SDP relaxations incorporate the second-order KTT conditions, which is not trivial and is one of the major contributions of this work. We experimentally demonstrate that our relaxation is significantly stronger than a basic SDP relaxation due to Shor, in the sense that the branch-and-bound tree based on our relaxation is smaller.

In Chapter 4, we aim to solve general QP via a finite branch-and-bound scheme in conjunction with advanced relaxations based on SDP. For this purpose, we explicitly incorporate the first-order KKT condition into the constraints of QP and models it as NQP, a quadratic program with complementarity constraints. We utilize an existing SDP relaxations for NQP and its solution technique, which requires finite bounds on all the variables. We show how to obtain the finite bounds via solving related linear programs. We perform a preliminary comparison of our branch-and-bound scheme with BARON, an state-of-the-art global solver, where we find BARON is faster than our method on many problem instances. However, our method is very competitive on BoxQP problems. As a future work, we would like to explore methods to improve the following aspects of our scheme: pre-processing, branching rules, tightening the feasible set, etc. Ultimately, our goal is to have a comprehensive comparison

with BARON on benchmark problems.

In Chapter 5, we investigate a first-order smoothing technique to solve large-scale instances of a class of LPs, which has many applications in the field of machine learning. Although modern linear programming technology is very advanced, solving large-scale LPs are still a challenging task because of the limitation of memory on computers. Here our goal is to develop a first-order method that requires much less memory and is thus suitable for large-scale problems. We apply Nesterov's smoothing technique to the problem on hand. Our main contribution includes (i) one critical step to transform the unbounded feasible set of our problem into a bounded one using a parameter that bounds the optimal value; and (ii) the design of a scheme that dynamically updates the parameter to speed up the convergence. We apply the smoothing technique to two machine learning problems and obtain favorable computational results compared to two other existing techniques. This work contributes to the field of convex optimization by showing the potential of nonsmooth techniques in solving large-scale convex problems.

REFERENCES

Achterberg, T., T. Koch, and A. Martin (2006). MIPLIB 2003. *Operations Research Letters 34*(4), 1–12. See `http://miplib.zib.de`.

Alizadeh, F. and D. Goldfarb (2003). Second-order cone programming. *Math. Program. 95*(1, Ser. B), 3–51. ISMP 2000, Part 3 (Atlanta, GA).

Andersen, E. D., C. Roos, and T. Terlaky (2002). Notes on duality in second order and p-order cone optimization. *Optimization 51*(4), 627–643.

Anderson, E. J. and P. Nash (1987). *Linear programming in infinite-dimensional spaces*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester: John Wiley & Sons Ltd. Theory and applications, A Wiley-Interscience Publication.

Anstreicher, K. M. (2007, May). Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. Manuscript, University of Iowa.

Anstreicher, K. M. and L. A. Wolsey (2009). Two "well-known" properties of subgradient optimization. *Math. Program. 120*(1), 213–220.

Asuncion, A. and D. Newman (2007). UCI machine learning repository.

Ataman, K. (2007). *Learning to rank by maximizing the AUC with linear programming for problems with binary output*. Ph. D. thesis, University of Iowa.

Balas, E., S. Ceria, and G. Cornuéjols (1993). A lift-and-project cutting plan algorithm for mixed 0-1 programs. *Mathematical Programming 58*, 295–324.

Balas, E. and M. Perregaard (2003). A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Math. Program. 94*(2-3, Ser. B), 221–245. The Aussois 2000 Workshop in Combinatorial Optimization.

Ben-Tal, A. and A. Nemirovski (2001). *Lectures on modern convex optimization*. MPS/SIAM Series on Optimization. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM). Analysis, algorithms, and engineering applications.

Bienstock, D. and M. Zuckerberg (2004). Subset algebra lift operators for 0-1 integer programming. *SIAM J. Optim. 15*(1), 63–95 (electronic).

Borwein, J. M. and A. S. Lewis (2006). *Convex analysis and nonlinear optimization* (Second ed.). CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, 3. New York: Springer. Theory and examples.

Boyd, S. and A. Mutapcic (2008, April). Subgradient methods. in lecture notes for EE364b, Stanford University, Winter 2006–07.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Burer, S. (2006, October). On the copositive representation of binary and continuous nonconvex quadratic programs. Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA, USA. Revised January 2007 and July 2007. Forthcoming.

Burer, S. (2009). On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming 120*(2), 479–495.

Burer, S. (2010). Optimizing a polyhedral-semidefinite relaxation of completely positive programs. *Mathematical Programming Computation 2*(1), 1–19.

Burer, S. and J. Chen (2009a). A p-cone sequential relaxation procedure for 0-1 integer programs. *Optimization Methods Software 24*(4-5), 523–548.

Burer, S. and J. Chen (2009b). Relaxing the optimality conditions of box qp. Technical report. *Computational Optimization and Applications, DOI 10.1007/s10589-009-9273-2*.

Burer, S. and D. Vandenbussche (2005, June). A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA, USA. Revised April 2006 and June 2006. To appear in *Mathematical Programming*.

Burer, S. and D. Vandenbussche (2006a, November). Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. Manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA, USA. Revised June 2006 and July 2006. To appear in *Computational Optimization and Applications*.

Burer, S. and D. Vandenbussche (2006b). Solving lift-and-project relaxations of binary integer programs. *SIAM J. Optim. 16*(3), 726–750 (electronic).

Burer, S. and D. Vandenbussche (2008). A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Math. Program. 113*(2, Ser. A), 259–282.

Dantzig, G. B. (1998). *Linear Programming and Extensions*. Princeton University Press.

d'Aspremont, A. (2008). Smooth optimization with approximate gradient. *SIAM J. on Optimization 19*(3), 1171–1183.

De Angelis, P., P. Pardalos, and G. Toraldo (1997). Quadratic programming with box constraints. In I. M. Bomze, T. Csendes, R. Horst, and P. Pardalos (Eds.), *Developments in Global Optimization*, pp. 73–94. Kluwer Academic Publishers.

Fung, G. and O. L. Mangasarian (2004, July). A feature selection newton method for support vector machine classification. *Computational Optimization and Applications 28*(2), 185–202.

Gilpin, A., S. Hoda, J. Peña, and T. Sandholm (2007). Gradient-based algorithms for finding nash equilibria in extensive form games. In *WINE*, pp. 57–69.

Glineur, F. and T. Terlaky (2004). Conic formulation for l_p-norm optimization. *J. Optim. Theory Appl. 122*(2), 285–307.

Goemans, M. (1998). Semidefinite programming and combinatorial optimization. *Documenta Mathematica Extra Volume ICM 1998*(III), 657–666.

Gomory, R. E. (1963). An algorithm for integer solutions to linear programs. In R. Graves and P. Wolfe (Eds.), *Recent Advances in Mathematical Programming*, pp. 269–302. McGraw-Hill.

Gould, N. I. M. and P. L. Toint (2002). Numerical methods for large-scale non-convex quadratic programming. In *Trends in industrial and applied mathematics (Amritsar, 2001)*, Volume 72 of *Appl. Optim.*, pp. 149–179. Dordrecht: Kluwer Acad. Publ.

Hoda, S., A. Gilpin, and J. Peña (2007). Smoothing techniques for computing nash equilibria of sequential game. Working Paper, Tepper School of Business, Carnegie Mellon Universit.

Horst, R. and H. Tuy (1993). *Global optimization* (Second ed.). Berlin: Springer-Verlag. Deterministic approaches.

Hsu, C. W., C. C. Chang, and C. J. Lin (2003). A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei. http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

Johnson, D. and M. Trick (1996). *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. American Mathematical Society.

Kim, S. and M. Kojima (2001). Second order cone programming relaxation of non-convex quadratic optimization problems. *Optim. Methods Softw. 15*(3-4), 201–224.

Kim, S. and M. Kojima (2003). Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. *Comput. Optim. Appl. 26*(2), 143–154.

Kim, S., M. Kojima, and M. Yamashita (2003). Second order cone programming relaxation of a positive semidefinite constraint. *Optim. Methods Softw. 18*(5), 535–541.

Kojima, M. and L. Tunçel (2000a). Cones of matrices and successive convex relaxations of nonconvex sets. *SIAM J. Optim. 10*(3), 750–778.

Kojima, M. and L. Tunçel (2000b). Discretization and localization in successive convex relaxation methods for nonconvex quadratic optimization. *Math. Program. 89*(1, Ser. A), 79–111.

Krokhmal, P. and P. Soberanis (2008). Risk optimization with p-order conic constraints: A linear programming approach. Working paper, University of Iowa, Iowa City, IA, USA.

Lan, G., Z. Lu, and R. D. C. Monteiro (2009). Primal-dual first-order methods with $o(1/\epsilon)$ iteration-complexity for cone programming. *Mathematical Programming*, 1436–4646. (Online).

Lasserre, J. B. (2001). Global optimization with polynomials and the problem of moments. *SIAM J. Optim. 11*(3), 796–817.

Lfberg, J. (2004). Yalmip : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan.

Lobo, M., L. Vandenberghe, S. Boyd, and H. Lebret (1998). Applications of second-order cone programming. *Linear Algebra and its Applications 284*, 193–228.

Lovász, L. and A. Schrijver (1991). Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization 1*, 166–190.

Mangasarian, O. L. (2006). Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research 7*, 1517–1530.

MOSEK, Inc. (2007). *The MOSEK optimization tools manual 5.0.*

Nedic, A. and D. P. Bertsekas (2001). Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization 12*(1), 109–138.

Nesterov, Y. (2000). Global quadratic optimization via conic relaxation. In R. Saigal, L. Vandenberghe, and H. Wolkowicz (Eds.), *Handbook of Semidefinite Programming*. Kluwer Academic Publishers.

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization: A Basic Course*. Boston, MA: Kluwer Academic.

Nesterov, Y. (2005a). Excessive gap technique in nonsmooth convex minimization. *SIAM J. on Optimization 16*(1), 235–249.

Nesterov, Y. (2005b). Smooth minimization of non-smooth functions. *Math. Program. 103*(1), 127–152.

Nesterov, Y. (2007, July). Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming 110*(2), 245–259.

Nesterov, Y. E. and A. S. Nemirovskii (1994). *Interior-Point Polynomial Algorithms in Convex Programming*. Philadelphia: Society for Industrial and Applied Mathematics.

Nesterov, Y. E. and M. J. Todd (1997). Self-scaled barriers and interior-point methods for convex programming. *Math. Oper. Res. 22*(1), 1–42.

Pardalos, P. (1991). Global optimization algorithms for linearly constrained indefinite quadratic problems. *Computers and Mathematics with Applications 21*, 87–97.

Pardalos, P. M. and S. A. Vavasis (1991). Quadratic programming with one negative eigenvalue is NP-hard. *J. Global Optim. 1*(1), 15–22.

Parrilo, P. A. (2003). Semidefinite programming relaxations for semialgebraic problems. *Math. Program. 96*(2, Ser. B), 293–320. Algebraic and geometric methods in discrete optimization.

Pintér, J. D. (1995). *Global Optimization in Action*. Dordrecht: Kluwer.

Sahinidis, N. V. (1996). BARON: a general purpose global optimization software package. *J. Glob. Optim. 8*, 201–205.

Sahinidis, N. V. (2000). *Branch And Reduce Optimization Navigator, Users Manual, Version 4.0*.

Sahinidis, N. V. and M. Tawarmalani (2010). *BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual.

Sherali, H. D. and W. P. Adams (1990). A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discrete Math. 3*(3), 411–430.

Sherali, H. D. and W. P. Adams (1997). *A Reformulation-Linearization Technique (RLT) for Solving Discrete and Continuous Nonconvex Problems*. Kluwer.

Sherali, H. D. and G. Choi (1996). Recovery of primal solutions when using subgradient optimization methods to solve lagrangian duals of linear programs. *Operations Research Letters 19*(3), 105 – 113.

Sherali, H. D. and C. H. Tuncbilek (1995). A reformulation-convexification approach for solving nonconvex quadratic programming problems. *J. Global Optim. 7*, 1–31.

Shor, N. (1987). Quadratic optimization problems. *Soviet Journal of Computer and Systems Science 25*, 1–11. Originally published in *Tekhnicheskaya Kibernetika*, 1:128–139, 1987.

Sturm, J. F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Softw. 11/12*(1-4), 625–653.

Tawarmalani, M. and N. Sahinidis (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications* (First ed.). Springer.

Tütüncü, R. H., K. C. Toh, and M. J. Todd (2001, August). *SDPT3: A Matlab software package for semidefinite-quadratic-linear programming, version 3.0*. Available from `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`.

Vandenberghe, L. and S. Boyd (1999). Applications of semidefinite programming. *Applied Numerical Mathematics 29*, 283–299.

Vandenbussche, D. and G. Nemhauser (2005a). A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming 102*(3), 559–575.

Vandenbussche, D. and G. Nemhauser (2005b). A polyhedral study of nonconvex quadratic programs with box constraints. *Mathematical Programming 102*(3), 531–557.

Vanderbei, R. J. (2007). *Linear Programming: Foundations and Extensions* (third ed.). New York: Springer.

Woodsend, K. and J. Gondzio (2009). Exploiting separability in large-scale linear support vector machine training. *Computational Optimization and Applications*. http://www.springerlink.com/content/N04W25255753056M.

Wright, S. J. (1997). *Primal-dual interior-point methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM).

Xue, G. and Y. Ye (2000). An efficient algorithm for minimizing a sum of p-norms. *SIAM J. Optim. 10*(2), 551–579 (electronic).

Ye, Y. (1999). Approximating quadratic programming with bound and quadratic constraints. *Math. Program. 84*(2, Ser. A), 219–226.

Zhu, J., S. Rosset, T. Hastie, and R. Tibshirani (2003). 1-norm support vector machines. In *Neural Information Processing Systems*, pp. 16. MIT Press.