General Heuristics for Nonconvex Quadratically Constrained Quadratic Programming

Jaehyun Park   Stephen Boyd

March 22, 2017

Abstract

We introduce the *Suggest-and-Improve* framework for general nonconvex quadratically constrained quadratic programs (QCQPs). Using this framework, we generalize a number of known methods and provide heuristics to get approximate solutions to QCQPs for which no specialized methods are available. We also introduce an open-source Python package `QCQP`, which implements the heuristics discussed in the paper.

Contents

3

1 Introduction

In this paper we introduce the *Suggest-and-Improve* heuristic framework for general non-convex quadratically constrained quadratic programs (QCQPs). This framework can be applied to general QCQPs for which there are no available specialized methods. We only briefly mention global methods for solving QCQPs in §1.4, as the exponential running time of these methods makes them unsuitable for medium- to large-scale problems. Our main focus, instead, will be on polynomial-time methods for obtaining approximate solutions.

1.1 Quadratically constrained quadratic programming

A quadratically constrained quadratic program (QCQP) is an optimization problem that can be written in the following form:

$$\begin{array}{ll} \text{minimize} & f_0(x) = x^T P_0 x + q_0^T x + r_0 \\ \text{subject to} & f_i(x) = x^T P_i x + q_i^T x + r_i \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{1}$$

where $x \in \mathbf{R}^n$ is the optimization variable, and $P_i \in \mathbf{R}^{n \times n}$, $q_i \in \mathbf{R}^n$, $r_i \in \mathbf{R}$ are given problem data, for $i = 0, 1, \ldots, m$. Throughout the paper, we will use $f^\star$ to denote the optimal value of (1), and $x^\star$ to denote an optimal point, *i.e.*, that attains the objective value of $f^\star$, while satisfying all the constraints. For simplicity, we assume that all P_i matrices are symmetric, but we do not assume any other conditions such as definiteness. This means that (1), in general, is a nonconvex optimization problem.

The constraint $f_i(x) \leq 0$ is affine if $P_i = 0$. If we need to handle affine constraints differently from quadratic ones, we collect all affine constraints and explicitly write them as a single inequality $Ax \leq b$, where $A \in \mathbf{R}^{p \times n}$ and $b \in \mathbf{R}^p$ are of appropriate dimensions, and the inequality $\leq$ is elementwise. Then, (1) is expressed as:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, \tilde{m} \\ & Ax \leq b, \end{array} \tag{2}$$

where $\tilde{m} = m - p$ is the number of nonaffine constraints. When all constraints are affine (*i.e.*, $\tilde{m} = 0$), the problem is a (nonconvex) quadratic program (QP).

Even though we set up (1) in terms of inequality constraints only, it also allows quadratic equality constraints of the form $h_i(x) = 0$ to be added, as they can be expressed as two quadratic inequality constraints:

$$h_i(x) \leq 0, \quad -h_i(x) \leq 0.$$

An important example of quadratic equality constraint is $x_i^2 = 1$, which forces x_i to be either $+1$ or -1, and thus encoding a Boolean variable.

There are alternative formulations of (1) that are all equivalent. We start with the *epigraph form* of (1), which makes the objective function linear (hence convex) without loss

4

of generality, by introducing an additional scalar variable $t \in \mathbf{R}$:

$$
\begin{array}{ll}
\text{minimize} & t \\
\text{subject to} & f_0(x) \leq t \\
& f_i(x) \leq 0, \quad i = 1, \ldots, m.
\end{array}
\tag{3}
$$

It is also possible to write a problem equivalent to (1) that only has quadratic equality constraints, by introducing an additional variable $s \in \mathbf{R}^m$:

$$
\begin{array}{ll}
\text{minimize} & x^T P_0 x + q_0^T x + r_0 \\
\text{subject to} & x^T P_i x + q_i^T x + r_i + s_i^2 = 0, \quad i = 1, \ldots, m.
\end{array}
$$

The *homogeneous form* of (1) is a QCQP with $n+1$ variables and $m+1$ constraints, where the objective function and lefthand side of every constraint is a quadratic form in the variable, *i.e.*, there are no linear terms in the variable [LMS$^+$10]. For $i = 0, \ldots, m$, define

$$
\tilde{P}_i = \begin{bmatrix} P_i & (1/2)q_i \\ (1/2)q_i^T & r_i \end{bmatrix}.
$$

The homogeneous form of (1) is given by:

$$
\begin{array}{ll}
\text{minimize} & z^T \tilde{P}_0 z \\
\text{subject to} & z^T \tilde{P}_i z \leq 0, \quad i = 1, \ldots, m \\
& z_{n+1}^2 = 1,
\end{array}
\tag{4}
$$

with variable $z \in \mathbf{R}^{n+1}$. Note that the variable dimension and the number of constraints of (4) is one larger than that of (1). This problem is homogeneous in the sense that scaling z by a factor of $t \in \mathbf{R}$ scales both the objective and lefthand sides of the constraints by a factor of t^2. It is easy to check that if $z^\star$ is a solution of (4), then the vector $(z_1^\star / z_{n+1}^\star, \ldots, z_n^\star / z_{n+1}^\star)$ is a solution of (1).

1.2   Tractable cases

The class of problems that can be written in the form of (1) is very broad, and as we will see in §2, QCQPs are NP-hard in general. However, there are a number of special cases which we can solve efficiently.

Convex QCQP.   When all P_i matrices are positive semidefinite, problem (1) is convex and thus easily solvable in polynomial time [BV04, §4.4].

QCQP with one variable.   If the problem has only one variable, *i.e.*, $n = 1$, then the feasible set is explicitly computable using only elementary algebra. The feasible set in this case is a collection of at most $m + 1$ disjoint, closed intervals on $\mathbf{R}$. It is possible to compute

these intervals in $O(m \log m)$ time (using a binary search tree, for example). Then, minimizing a quadratic function in this feasible set can be done by evaluating the objective at the endpoints of the intervals, as well as checking the unconstrained minimizer (if there is one).

While one-variable problems are rarely interesting by themselves, we will take advantage of their solvability in §4.2 to develop a greedy heuristic for (1). For more details on the solution method and time complexity analysis, see Appendix A.

QCQP with one constraint. Consider QCQPs with a single constraint (*i.e.*, $m = 1$):

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & f_1(x) \le 0.
\end{array}
\tag{5}
$$

Even when f_0 and f_1 are both nonconvex, (5) is solvable in polynomial time [BV04, Fer00, YZ03, LMS$^+$10]. This result, also known as the S-procedure in control theory [BEGFB94, PT07], states that even though (5) is not convex, strong duality holds and the Lagrangian relaxation produces the optimal value $f^\star$. A variant of (5) with an equality constraint $f_1(x) = 0$ is also efficiently solvable.

In Appendix B, we derive a solution method for the special case of (5), where the objective function is given by $f_0(x) = \|x - z\|_2^2$. This particular form will be used extensively in §4.4. Refer to [FLSX12, Mor93, PT07] for the solution methods for the general case.

QCQP with one interval constraint. Consider a variant of (5), with an interval constraint:

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & l \le f_1(x) \le u.
\end{array}
\tag{6}
$$

Solving this variant reduces to solving (5) twice, once with the upper bound constraint $f_1(x) \le u$ only, and once with the lower bound constraint $f_1(x) \ge l$ only [BTT96, HS16]. One of the solutions is guaranteed to be an optimal point of (6).

QCQP with homogeneous constraints with one negative eigenvalue. Consider a homogeneous constraint of the form $x^T P x \le 0$, where P has exactly one negative eigenvalue. This constraint can be rewritten as a disjunction of two second-order cone (SOC) constraints [LVBL98]. Let $P = Q \Lambda Q^T$ be the eigenvalue decomposition of P, with $\lambda_1 < 0$. Then, $x^T P x \le 0$ if and only if

$$
\sum_{i=2}^{n} \lambda_i (q_i^T x)^2 \le -\lambda_1 (q_1^T x)^2,
$$

or equivalently,

$$
\left\| (\sqrt{\lambda_2} q_2^T x, \ldots, \sqrt{\lambda_n} q_n^T x) \right\|_2 \le \sqrt{|\lambda_1|} |q_1^T x|,
$$

where $q_1, \ldots, q_n$ are the columns of Q. Depending on the sign of $q_1^T x$, one of the following SOC inequalities is true if and only if $x^T P x \leq 0$:

$$\left\| \left(\sqrt{\lambda_2} q_2^T x, \ldots, \sqrt{\lambda_n} q_n^T x \right) \right\|_2 \leq \sqrt{|\lambda_1|} q_1^T x, \tag{7}$$

$$\left\| \left(\sqrt{\lambda_2} q_2^T x, \ldots, \sqrt{\lambda_n} q_n^T x \right) \right\|_2 \leq -\sqrt{|\lambda_1|} q_1^T x. \tag{8}$$

Suppose that the constraint $x^T P x \leq 0$ was the only nonconvex constraint of (1). Then, we can solve two convex problems, one where the nonconvex constraint is replaced with (7), and the other where the same constraint is replaced with (8). The one that attains a better solution is optimal. Note that if the sign of $q_1^T x$ of any solution is known, then that information can be used to avoid solving both (7) and (8). For example, if it is known a priori that some solution x satisfies $q_1^T x \geq 0$, then only (7) needs to be solved.

This approach generalizes to the case where there are multiple constraints in the form of $x^T P x \leq 0$, where P has exactly one negative eigenvalue. With k such constraints, one needs to solve 2^k convex problems.

1.3 Algorithm

We introduce the *Suggest-and-Improve* framework, which is a simple but flexible idea that encapsulates all of our heuristics. The overall algorithm can be summarized in two high-level steps, as shown in Algorithm 1.1.

Algorithm 1.1 *Suggest-and-Improve algorithm.*

1. *Suggest.* Find a *candidate point* $x \in \mathbf{R}^n$.
2. *Improve.* Run a local method from x to find a point $z \in \mathbf{R}^n$ that is no worse than x.
3. **return** z.

For Algorithm 1.1 to be well-defined, we need the notion of better points used in the *Improve* step. While there are other reasonable ways to define this, we use the following definition throughout the paper. Let $p_+ = \max\{p, 0\}$ denote the positive part of $p \in \mathbf{R}$, and

$$v(x) = \max\{f_1(x)_+, \ldots, f_m(x)_+\}$$

denote the maximum constraint violation of $x \in \mathbf{R}^n$. We say $z \in \mathbf{R}^n$ is *better* than $x \in \mathbf{R}^n$ if one of the following conditions is met:

- Maximum constraint violation of z is smaller than that of x, *i.e.*, $v(z) < v(x)$.

- Maximum constraint violation of z and x are the same, and the objective function attains smaller value at z than at x, *i.e.*, $v(z) = v(x)$ and $f_0(z) < f_0(x)$.

In other words, we are defining better points in terms of the lexicographic order of the pair $(v(x), f_0(x))$. This definition easily extends to the notion of a best point in the set of points (note that there can be multiple best points in a given set).

The candidate points returned from the *Suggest* method serve as starting points of local methods in the *Improve* step, and we do not require any condition on them in terms of feasibility. *Suggest* methods can be randomized, and parallelized to produce multiple candidate points. *Improve* methods attempt to produce better points (as defined above) than the candidate points. *Improve* methods can be applied in a flexible manner. For example, note that composition of any number of *Improve* methods is also an *Improve* method. One can also apply different *Improve* methods to a single candidate point in parallel. Similarly, given multiple candidate points, one can apply multiple *Improve* methods on each candidate point and take a best one.

There are many different ways to implement the *Suggest* and *Improve* methods, and how they are implemented determines the running time, suboptimality, and various other properties of the overall algorithm. Essentially, they can be considered as modules that one can choose from a collection of alternatives. Throughout the paper, we will explore different options to implement them.

To motivate the discussions, we start by recognizing a wide variety of well-known problem classes and examples that can be formulated as QCQPs, in §2. In §3, we explore ways to implement the *Suggest* step. Our focus is on the *relaxation* technique, which is typically used to approximate the solution to computationally intractable problems by replacing constraints with some other constraints that are easier to handle. We will discuss various relaxations of (1) for finding reasonable candidate points, as well as obtaining a lower bound on the optimal value $f^\star$. Then, in §4, we discuss local optimization methods for improving a given candidate point. We will start with specialized methods for some subclasses of QCQP, and give three methods that can be applied to general QCQPs. Finally, we introduce an open-source implementation of these methods in §5, and show several numerical examples in §6.

1.4   Previous work

Quadratic programming.   Research on QP began in the 1950s (see, *e.g.*, [FW56, Hil57]). Several hardness results on QP were published once the concept of NP-completeness and NP-hardness was established in the early 1970s [Coo71, Kar72]. In particular, [Sah74] showed that QP with a negative definite quadratic term is NP-hard. On the other hand, QP with convex objective function was shown to be polynomial-time solvable [KTK80].

Quadratically constrained quadratic programming.   Van de Panne [VdP66] studied a special class of QCQPs, which is to optimize an affine function subject to a single quadratic constraint over a polyhedron. While problems with a quadratic objective function and multiple quadratic constraints were introduced as early as 1951 in [KT51], duality results and cutting plane algorithms for solving them were developed later [Bar72].

QCQP with one constraint. Problem (5) arises from many optimization algorithms, and most notably, in trust region methods [HLS13, Gay81, MS83, SW95, NW99]. Eigenvalue or singular value problems are also formulated in this form [LSY16, SW95, HLS13]. The strong duality result is known under various names in different disciplines. The term $\mathcal{S}$-procedure is from control theory [BEGFB94, PT07]. Variations of the $\mathcal{S}$-procedure are known in linear algebra in the context of simultaneous diagonalization of symmetric matrices [Cal64, Uhl79]. Many related results and additional references can be found in [HJ91, §1.8], and [BTN01, §4.10.5].

Semidefinite programming. Semidefinite programming (SDP) and semidefinite programming relaxation (SDR) are closely related to QCQPs. The study of SDPs started since the early 1990s [Ali91, NN94], and subsequent research during the 1990s was driven by various applications, including combinatorial problems [GW95], control [BEGFB94, SGC97, DP00], communications and signal processing [Luo03, DLW00, MDW⁺02], and many other areas of engineering. For more extensive overviews and bibliographies of SDPs, refer to [WSV00, Tod01, LO96, VB96]. The idea of SDR for QCQPs was suggested as early as 1979 in [Lov79], but the work that started a rapid development of the technique was [GW95], which applied SDR to the maximum cut problem and derived a data-independent approximation factor of 0.87856. Since then, SDR has also been applied outside the domain of combinatorial optimization problems [VB96, LMS⁺10].

Global methods. Global methods for nonconvex problems always find an optimal point and certify it, but are often slow; the worst-case running time grows exponentially with problem size (unless P is NP). Many known algorithms for globally solving (1) are based on the *branch-and-bound* framework. Branch-and-bound generally works by recursively splitting the feasible set into multiple parts and solving the problem restricted in each of the subdivision, typically via relaxation techniques. For more details on the branch-and-bound scheme, see [LW66, Bal68, Moo91, BM03]. A popular variant of the branch-and-bound scheme is the *branch-and-cut* method, which incorporates cutting planes [Kel60] to tighten the subproblems generated from branching [PR91, Mit02]. See, for example, [AHJS00] for a branch-and-cut method for solving (1). Linderoth [Lin05] proposes an algorithm that partitions the feasible region into the Cartesian product of two-dimensional triangles and rectangles. Burer and Vandenbussche [BV08] shows an algorithm that uses SDR as a subroutine. A variant of the method tailored for nonconvex QPs also exists [CB12].

Existing solvers. There are a number of off-the-shelf software packages that can handle various subclasses of QCQP. We mention some of the solvers here: GloMIQO [MF13], BARON [Sah14], Ipopt [WLMK09], Couenne [IBM13], and others provide global methods for (mixed-integer) QCQPs. Gurobi [GO15], CPLEX [IBM13], MOSEK [ApS17], and SCIP [Ach09] provide global methods for mixed-integer nonlinear programs, with limited support for nonconvex constraints. Packages such as ANTIGONE [MF14a], KNITRO [BNW06], and NLopt [Joh14] provide global and local optimization methods for nonlinear optimization

9

problems.

2 Examples and applications

In this section, we show various subclasses of QCQP, as well as several applications that are more specific.

2.1 Examples

Polynomial problems. A polynomial optimization problem seeks to minimize a polynomial over a set defined by polynomial inequalities:

$$
\begin{aligned}
\text{minimize} \quad & p_0(x) \\
\text{subject to} \quad & p_i(x) \leq 0, \quad i = 1, \ldots, m.
\end{aligned}
\tag{9}
$$

Here, each $p_i : \mathbf{R}^n \to \mathbf{R}$ is a polynomial in x. All polynomial optimization problems can be converted to QCQPs by introducing additional variables that represent the product of two terms, and appropriate equality constraints that describe these relations. For example, in order to represent a term $x_1^2 x_2$, one can introduce additional variables, say, u and v, and add constraints $u = x_1^2$ and $v = ux_2$. Then we can simply write v in place of $x_1^2 x_2$. In general, at most $d - 1$ new variables and constraints are sufficient to describe any term of order d. By applying these transformations iteratively, we can transform the original polynomial problem into a QCQP with additional variables. As a concrete example, suppose that we want to solve the following polynomial problem:

$$
\begin{aligned}
\text{minimize} \quad & x^3 - 2xyz + y + 2 \\
\text{subject to} \quad & x^2 + y^2 + z^2 - 1 = 0,
\end{aligned}
$$

in the variables $x, y, z \in \mathbf{R}$. We introduce two new variables $u, v \in \mathbf{R}$ along with two equality constraints:

$$
u = x^2, \quad v = yz.
$$

The problem then becomes:

$$
\begin{aligned}
\text{minimize} \quad & xu - 2xv + y + 2 \\
\text{subject to} \quad & x^2 + y^2 + z^2 - 1 = 0 \\
& u - x^2 = 0 \\
& v - yz = 0,
\end{aligned}
$$

which is now a QCQP in the variables $x, y, z, u, v \in \mathbf{R}$.

Box-constrained mixed-integer quadratic programming. Mixed-integer quadratic programming (MIQP) is the problem of optimizing a quadratic function over a polyhedron, where some variables are constrained to be integer-valued. Typically, MIQP comes with

a box constraint that specifies lower and upper bounds on x, in the form of $l \le x \le u$. Formally, it can be written as the following:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & Ax \le b \\ & l \le x \le u \\ & x_1, \dots, x_p \in \mathbf{Z}. \end{array} \tag{10}$$

We can write the integer constraints as a set of nonconvex quadratic inequalities. For example, $x_1 \in \{l_1, l_1 + 1, \dots, u_1\}$ if and only if $l_1 \le x_1 \le u_1$ and

$$(x_1 - k)(x_1 - (k+1)) \ge 0,$$

for all $k = l_1, l_1 + 1, \dots, u_1 - 1$. By replacing the integer constraints in this way, we can write (10) in the form of (1).

Rank-constrained problems. Let $X \in \mathbf{R}^{p \times q}$ be a matrix-valued variable. The rank constraint $\mathbf{Rank}(X) \le k$ can be written as a quadratic constraint by introducing auxiliary matrix variables $U \in \mathbf{R}^{p \times k}$ and $V \in \mathbf{R}^{k \times q}$, and adding an equality constraint $X = UV$. Note that this is a set of pq equality constraints that are quadratic in the elements of X, U, and V:

$$x_{ij} = \sum_{r=1}^{k} u_{ir} v_{rj}, \qquad i = 1, \dots, p, \quad j = 1, \dots, q.$$

2.2 Applications

Boolean least squares. The Boolean least squares problem has the following form:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2^2 \\ \text{subject to} & x_i \in \{-1, 1\}, \quad i = 1, \dots, n, \end{array} \tag{11}$$

in the variable $x \in \mathbf{R}^n$, where $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$. This is a basic problem in digital communications (maximum likelihood estimation for digital signals). By writing the Boolean constraint $x_i \in \{-1, 1\}$ as $x_i^2 = 1$, we get a QCQP equivalent to (11):

$$\begin{array}{ll} \text{minimize} & x^T A^T A x - 2b^T A x + b^T b \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \dots, n. \end{array}$$

Two-way partitioning problems. The two-way partitioning problem can be written as the following [BV04, §5.1.5]:

$$\begin{array}{ll} \text{maximize} & x^T W x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \dots, n, \end{array} \tag{12}$$

with variable $x \in \mathbf{R}^n$, where $W \in \mathbf{R}^{n \times n}$ is symmetric. This problem is directly a nonconvex QCQP of the form (1). (Notice, however, that this is a maximization problem, equivalent to

minimizing the negative of the objective.) Since the constraints restrict the possible values of each x_i to $+1$ or -1, each feasible x naturally corresponds to the partition

$$\{1, \ldots, n\} = \{i \mid x_i = -1\} \cup \{i \mid x_i = +1\}.$$

The matrix coefficient W_{ij} can be interpreted as the utility of having the elements i and j in the same cluster, with $-W_{ij}$ the utility of having i and j in different clusters. Then, problem (12) can be interpreted as finding the partition that maximizes the total utility over all pairs of elements.

It is possible to generalize this formulation to k-way partitioning problems. In this variant, we would like to partition the set $\{1, \ldots, n\}$ into k clusters, where there is utility associated between every pair of elements not belonging to the same cluster:

$$
\begin{array}{ll}
\text{maximize} & \sum_{u=1}^{k} \sum_{v=1}^{k} x^{(u)T} W^{(u,v)} x^{(v)} \\
\text{subject to} & x_i^{(u)} (x_i^{(u)} - 1) = 0, \quad i = 1, \ldots, n, \quad u = 1, \ldots, k \\
& \sum_{u=1}^{k} x_i^{(u)} = 1, \quad i = 1, \ldots, n.
\end{array}
$$

Here, the matrices $W^{(u,v)}$ describe utility between clusters u and v. The variables $x^{(u)}$ can be considered as indicator variables that represent which elements belong to cluster u. The first set of equality constraints limit each $x_i^{(u)}$ to be either 0 or 1. The second set of equality constraints states that element i belongs to exactly one of the k clusters.

Maximum cut. The maximum cut problem is a classic problem in graph theory and network optimization, and is an instance of two-way partitioning problem (12). On an n-node graph $G = (V, E)$, where the nodes are numbered from 1 to n, we define weights W_{ij} associated with each edge $(i, j) \in E$. If no edge connects i and j, we define $W_{ij} = 0$. The maximum cut problem seeks to find a cut of the graph with the largest possible weight, *i.e.*, a partition of the set of nodes V in two clusters V_1 and V_2 such that the total weight of all edges linking these clusters is maximized. Given an assignment $x \in \{-1, +1\}^n$ of nodes to the clusters, the value of the cut is defined by

$$\frac{1}{2} \sum_{i,j : x_i x_j = -1} W_{ij},$$

which is also equal to

$$\frac{1}{4} \sum_{i,j} W_{ij}(1 - x_i x_j) = -(1/4)x^T W x + (1/4)\mathbf{1}^T W \mathbf{1}.$$

Here, $\mathbf{1}$ represents a vector with all components equal to one. The maximum cut problem, then, can be written as:

$$
\begin{array}{ll}
\text{maximize} & -(1/4)x^T W x + (1/4)\mathbf{1}^T W \mathbf{1} \\
\text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n.
\end{array}
\tag{13}
$$

12

We can further rewrite (13) so that the objective function is homogeneous in x. Let L be the Laplacian matrix of the underlying graph G, which is given by

$$L_{ij} = \begin{cases} -W_{ij} & \text{if } i \neq j \\ \sum_{j \neq i} W_{ij} & \text{otherwise.} \end{cases}$$

If the edge weights W_{ij} are all nonnegative, the Laplacian L is positive semidefinite (see, e.g., [GR13, §13.1]). Using the Laplacian (and ignoring the constant factor), the maximum cut problem can be written in the same form as (12):

$$\begin{array}{ll} \text{maximize} & x^T L x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n. \end{array} \tag{14}$$

The *maximum graph bisection problem* is a variant of the maximum cut problem, which has an additional constraint that the two clusters V_1 and V_2 must have the same size, *i.e.*, $\mathbf{1}^T x = 0$. (In this variant, we assume that n is even.)

Maximum clique. The maximum clique problem is to find the complete subgraph of the maximum cardinality in a given graph. The problem can be formulated as a QCQP:

$$\begin{array}{ll} \text{maximize} & \mathbf{1}^T x \\ \text{subject to} & x_i x_j (1 - A_{ij}) = 0, \quad i, j = 1, \ldots, n \\ & x_i(x_i - 1) = 0, \quad i = 1, \ldots, n. \end{array} \tag{15}$$

Here, $A \in \{0, 1\}^{n \times n}$ is the adjacency matrix of the given graph, where A_{ii} is defined as 1 for $i = 1, \ldots, n$, for convenience. The first set of constraints can be interpreted as: "if nodes i and j are both in the clique, *i.e.*, $x_i = x_j = 1$, then they must be connected by an edge, *i.e.*, $A_{ij} = 1$."

3-satisfiability. The 3-satisfiability (3-SAT) problem is an NP-complete problem, which is to find an assignment to a set of Boolean variables $x_1, \ldots, x_n$ that makes a given logical expression true. The given expression is a conjunction of r logical expressions called *clauses*, each of which is a disjunction of three variables, with optional negations. This can be formulated as a quadratically constrained feasibility problem as the following:

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & Ax + b \geq \mathbf{1} \\ & x_i(x_i - 1) = 0, \quad i = 1, \ldots, n. \end{array} \tag{16}$$

Note that this is a feasibility problem, and thus an arbitrary objective function can be optimized (such as $\mathbf{1}^T x$). Here, the matrix $A \in \mathbf{R}^{r \times n}$ and vector $b \in \mathbf{R}^r$ encode the r clauses in the following way:

$$A_{ij} = \begin{cases} 1 & \text{if the } i\text{th clause includes } x_i \\ -1 & \text{if the } i\text{th clause includes the negation of } x_i \\ 0 & \text{otherwise.} \end{cases}$$

Without loss of generality, we assume that no clause contains a variable and its negation at the same time, because such a clause can be left out without changing the problem. Each b_i is set as the number of negated variables in the ith clause. For example, the inequality corresponding to a clause $(x_1 \vee \neg x_4 \vee x_6)$ would be

$$x_1 + (1 - x_4) + x_6 \geq 1.$$

Phase retrieval. The phase retrieval problem is to recover a general signal such as an image from the magnitude of its Fourier transform. There are many application areas of the phase retrieval problem, including, but not limited to: X-ray crystallography, diffraction imaging, optics, astronomical imaging, and microscopy [Els03, WdM15, CLS15, CESV15, NJS13, QSH+16, HES16b, HES16a, SEC+15]. While there are different ways to formulate the problem, we give one in the form of a feasibility problem:

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & (a_i^T x)^2 = b_i, \quad i = 1, \ldots, m. \end{array}$$

Here, the optimization variable is $x \in \mathbf{R}^n$, and the problem data are $a_1, \ldots, a_m \in \mathbf{R}^n$ and $b_1, \ldots, b_m \in \mathbf{R}$.

Multicast downlink transmit beamforming. In the context of communications, the downlink beamforming problem seeks to design a multiple-input multiple-output wireless communication system that minimizes the total power consumption, while guaranteeing that the users receive certain signal-to-interference noise ratio. (For more details, refer to [GSS+10, MKB07, SDL06, YLS11, LMS+10, PVST09, GS15, THJ14, SDL06].) This problem can be formulated as a QCQP:

$$\begin{array}{ll} \text{minimize} & \|x\|_2^2 \\ \text{subject to} & x^T P_i x \geq 1, \quad i = 1, \ldots, m, \end{array} \tag{17}$$

in the variable $x \in \mathbf{R}^n$, where each $P_i \in \mathbf{R}^{n \times n}$ is positive semidefinite.

3 Relaxations and bounds

A *relaxation* of an optimization problem is obtained by taking a set of constraints and replacing it with a different set of constraints, such that the resulting feasible set contains the feasible set of the original problem. Relaxations have a property that the optimal value f^{rl} gives a lower bound on the optimal value $f^\star$ of the original problem. Tractable relaxations are of particular interest, since we can solve them to compute a lower bound on $f^\star$ of intractable optimization problems. While a solution x^{rl} of a relaxation is generally infeasible in the original problem, it can still serve as a reasonable starting point of various local methods (which we discuss in §4). But if x^{rl} is feasible in the original problem, then it is also a solution of the original problem.

Our main goal of the section is to implement the *Suggest* method of Algorithm 1.1 via various tractable relaxations of (1). As a byproduct, we also get a lower bound on $f^\star$.

3.1 Spectral relaxation

First, we explore a relaxation of (1) that is a generalized eigenvalue problem, hence the name *spectral relaxation.* This method generalizes eigenvalue bounds studied in [DP93a, DP93b, PR95]. Let $\lambda \in \mathbf{R}_+^m$ be an arbitrary vector in the nonnegative orthant, and consider the following optimization problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & \sum_{i=1}^m \lambda_i f_i(x) \leq 0. \end{array} \tag{18}$$

Since λ is elementwise nonnegative, every feasible point x of (1) is also feasible in (18). Thus, (18) is a relaxation of (1), and its optimal value f^{rl} is a lower bound on $f^\star$. Since (18) is a QCQP with one constraint (5), it is a tractable problem that can be solved using methods via matrix pencil, for example [FLSX12, Mor93, PT07]. It is easy to see that the same idea extends to problems with equality constraints. Below, we derive and show an explicit bound on $f^\star$ for some examples.

Two-way partitioning problem. Let us derive a spectral relaxation of (12). Note that via relaxation, we are seeking an upper bound on $f^\star$ as opposed to a lower bound, since (12) is a maximization problem. With $\lambda = \mathbf{1}$, the relaxation is:

$$\begin{array}{ll} \text{maximize} & x^T W x \\ \text{subject to} & \sum_{i=1}^n x_i^2 = \|x\|_2^2 = n. \end{array}$$

It is easy to see that a solution x^{rl} of this relaxation is given by

$$x^{\mathrm{rl}} = \sqrt{n} v,$$

where v is the eigenvector of unit length corresponding to the maximum eigenvalue $\lambda_{\max}$ of W. With this value of x^{rl}, we have $f^{\mathrm{rl}} = n\lambda_{\max}$, which is an upper bound on $f^\star$.

We also note that by taking the sign of each entry of x^{rl}, we get a feasible point z of (12). Then, evaluating the objective $z^T W z$ gives a lower bound on $f^\star$. Thus, we get both lower and upper bounds on $f^\star$, simply by finding the largest eigenvalue and the corresponding eigenvector of W. In §4, we revisit this idea, and discuss various other methods for finding feasible points of general QCQPs in greater detail.

Multicast downlink transmit beamforming. The spectral relaxation of (17) with $\lambda = \mathbf{1}$ is:

$$\begin{array}{ll} \text{minimize} & \|x\|_2^2 \\ \text{subject to} & x^T \bar{P} x \geq 1. \end{array}$$

Here, $\bar{P} = (1/m)(\sum_{i=1}^m P_i)$. Let $\lambda_{\max}$ be the largest eigenvalue of $\bar{P}$, and v be the corresponding eigenvector. Except in the pathological case where $\bar{P} = 0$, we have $\lambda_{\max} > 0$. By observing that the shortest vector x^{rl} satisfying the constraint must be a multiple of v, we get

$$x^{\mathrm{rl}} = (1/\sqrt{\lambda_{\max}})v, \quad f^{\mathrm{rl}} = 1/\lambda_{\max}.$$

It is also possible to find a feasible point $z \in \mathbf{R}^n$ from x^{rl}. Let

$$t = \min_i \, (x^{\mathrm{rl}})^T P_i x^{\mathrm{rl}},$$

and set

$$z = (1/t)x^{\mathrm{rl}}.$$

It is easy to check that $\hat{x}$ is a feasible point of (17). By evaluating the objective at this point, we get an upper bound on $f^\star$:

$$\|z\|_2^2 = (1/t^2)\|x^{\mathrm{rl}}\|_2^2 = \frac{1}{t^2 \lambda_{\max}}.$$

3.2   Lagrangian relaxation

The Lagrangian relaxation provides another way to find a lower bound on $f^\star$, and it can be considered as a generalization of the spectral relaxation (18). See [BV04, §5] or [Ber99, §5] for more background on Lagrangian duality results. In this section, we derive the Lagrangian relaxation of (1), which has been studied since the 1980s by Shor and others [Sho87]. To simplify notation, we first define, for $\lambda \in \mathbf{R}_+^m$,

$$\tilde{P}(\lambda) = P_0 + \sum_{i=1}^m \lambda_i P_i, \qquad \tilde{q}(\lambda) = q_0 + \sum_{i=1}^m \lambda_i q_i, \qquad \tilde{r}(\lambda) = r_0 + \sum_{i=1}^m \lambda_i r_i.$$

The *Lagrangian* of (1) is given by

$$\mathcal{L}(x, \lambda) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) = x^T \tilde{P}(\lambda)x + \tilde{q}(\lambda)^T x + \tilde{r}(\lambda).$$

The *Lagrangian dual function* is then

$$g(\lambda) = \inf_x \mathcal{L}(x, \lambda) = \begin{cases} \tilde{r}(\lambda) - (1/4)\tilde{q}(\lambda)^T \tilde{P}(\lambda)^\dagger \tilde{q}(\lambda) & \text{if } \tilde{P}(\lambda) \succeq 0, \ \tilde{q}(\lambda) \in \mathcal{R}(\tilde{P}(\lambda)) \\ -\infty & \text{otherwise,} \end{cases}$$

where $\tilde{P}(\lambda)^\dagger$ and $\mathcal{R}(\tilde{P}(\lambda))$ denote the Moore–Penrose pseudoinverse and range of $\tilde{P}(\lambda)$, respectively [BV04, §A.5]. Using the Schur complement, we can write the Lagrangian dual problem as a semidefinite program (SDP):

$$
\begin{array}{ll}
\text{maximize} & \alpha \\
\text{subject to} & \lambda_i \geq 0, \quad i = 1, \ldots, n \\
& \begin{bmatrix} \tilde{P}(\lambda) & (1/2)\tilde{q}(\lambda) \\ (1/2)\tilde{q}(\lambda)^T & \tilde{r}(\lambda) - \alpha \end{bmatrix} \succeq 0,
\end{array}
\tag{19}
$$

with variables $\lambda \in \mathbf{R}^n$, $\alpha \in \mathbf{R}$.

Lagrangian relaxation (19) and spectral relaxation (18) are closely related techniques. We can show that solving the Lagrangian relaxation (19) is equivalent to finding the value

of λ that achieves the best spectral bound. This property also implies a natural way of obtaining a candidate point x^{rl}; we first solve (19) to obtain a solution $\lambda^\star$. Then, using the value of $\lambda^\star$, we solve the spectral relaxation (18). Its solution can be taken as a candidate point x^{rl}.

Theorem 1. *Let d_λ be the optimal value of* (18) *for a given $\lambda \in \mathbf{R}_+^m$, and $d^\star$ be the optimal value of* (19). *Then,*

$$d^\star = \sup_{\lambda \in \mathbf{R}_+^m} d_\lambda.$$

Proof. We first use the fact that strong duality holds for (18). The dual problem of (18) can be derived in the same way we derived (19):

$$
\begin{aligned}
\text{maximize} \quad & \gamma \\
\text{subject to} \quad & \eta \geq 0 \\
& \begin{bmatrix} \tilde{P}(\eta\lambda) & (1/2)\tilde{q}(\eta\lambda) \\ (1/2)\tilde{q}(\eta\lambda)^T & \tilde{r}(\eta\lambda) - \gamma \end{bmatrix} \succeq 0,
\end{aligned}
\tag{20}
$$

with variables $\eta, \gamma \in \mathbf{R}$. Due to strong duality, (20) has the same optimal value d_λ as (18). Note that taking the supremum over λ of d_λ for all possible $\lambda \in \mathbf{R}_+^m$ is equivalent to making λ in (20) an additional variable in $\mathbf{R}_+^m$, and solving the resulting problem. In other words, $\sup_{\lambda \in \mathbf{R}_+^m} d_\lambda$ is the optimal value of the following optimization problem:

$$
\begin{aligned}
\text{maximize} \quad & \gamma \\
\text{subject to} \quad & \eta \geq 0 \\
& \lambda_i \geq 0, \quad i = 1, \ldots, n \\
& \begin{bmatrix} \tilde{P}(\eta\lambda) & (1/2)\tilde{q}(\eta\lambda) \\ (1/2)\tilde{q}(\eta\lambda)^T & \tilde{r}(\eta\lambda) - \gamma \end{bmatrix} \succeq 0,
\end{aligned}
$$

with variables $\eta, \gamma \in \mathbf{R}$, $\lambda \in \mathbf{R}^m$. Since both η and λ are constrained to be nonnegative, we can eliminate η by replacing $\eta\lambda$ with λ and keeping the nonnegativity constraint on λ. The resulting problem is then equivalent to (19), which proves the identity. $\qquad\square$

3.3 Semidefinite relaxation

In this section, we derive a semidefinite relaxation (SDR) using a technique called *lifting*. This SDR and the Lagrangian dual problem (19) are Lagrangian duals of each other. To derive the SDR, we start by introducing a new variable $X = xx^T$ and rewriting the problem as:

$$
\begin{aligned}
\text{minimize} \quad & F_0(X, x) = \mathbf{Tr}(P_0 X) + q_0^T x + r_0 \\
\text{subject to} \quad & F_i(X, x) = \mathbf{Tr}(P_i X) + q_i^T x + r_i \leq 0, \quad i = 1, \ldots, m \\
& X = xx^T,
\end{aligned}
\tag{21}
$$

in the variables $X \in \mathbf{R}^{n \times n}$ and $x \in \mathbf{R}^n$. This rewriting is called a *lifting*, as we have embedded the original problem with n variables to a much larger space (of $n^2 + n$ dimensions). By

lifting, we obtain an additional property that the objective and constraints are affine in X and x, except the last constraint $X = xx^T$ which is nonconvex. When we replace this intractable constraint with $X \succeq xx^T$, we get a convex relaxation. By rewriting it using the Schur complement, we obtain an SDR:

$$
\begin{aligned}
\text{minimize} \quad & \mathbf{Tr}(P_0 X) + q_0^T x + r_0 \\
\text{subject to} \quad & \mathbf{Tr}(P_i X) + q_i^T x + r_i \leq 0, \quad i = 1, \ldots, m \\
& Z(X, x) = \begin{bmatrix} X & x \\ x^T & 1 \end{bmatrix} \succeq 0.
\end{aligned}
\tag{22}
$$

The optimal value f^{sdr} of problem (22) is then a lower bound on the optimal value $f^\star$ of (1). Under mild assumptions (*e.g.*, feasibility of the original problem), (22) has the same optimal value as (19). If an optimal point $(X^\star, x^\star)$ of (22) satisfies $X^\star = x^\star x^{\star T}$ (or equivalently, the rank of $Z(X^\star, x^\star)$ is one), then $x^\star$ is a solution of the original problem (1).

It may look like we have created too much slack by lifting the problem to a higher-dimensional space, followed by relaxing the equality constraint $X = xx^T$ to a semidefinite cone constraint. However, we can show that the SDR is tight when the original problem is convex.

Lemma 1. *Let (X, x) be any feasible point of* (22). *Then, x satisfies every convex constraint of* (1).

Proof. It is enough to verify the claim for any arbitrary convex constraint $x^T P x + q^T x + r \leq 0$ with $P \succeq 0$. Since (X, x) is a feasible point of (22), we have $X \succeq xx^T$. Then,

$$
\mathbf{Tr}(PX) \geq \mathbf{Tr}(Pxx^T) = x^T P x.
$$

Therefore,

$$
x^T P x + q^T x + r \leq \mathbf{Tr}(PX) + q^T x + r.
$$

Thus, if the righthand side is nonpositive, so is the lefthand side. The claim follows. $\qquad\square$

Lemma 2. *If problem* (1) *is convex, then its SDR is tight, i.e., $f^\star = f^{\mathrm{sdr}}$.*

Proof. Lemma 1 implies that if (1) is convex, then for any feasible point (X, x) of (22), replacing X with xx^T gives another feasible point with lower or equal objective value, *i.e.*, there exists an optimal point of (22) that satisfies $X = xx^T$. Thus, adding an additional constraint $X = xx^T$ to (22), which makes the problem equivalent to (21), does not change the set of solutions, nor the optimal value. $\qquad\square$

While the SDR bound is not tight in general, in some cases, it is possible to give a lower bound on f^{sdr} in terms of $f^\star$, which effectively gives both lower and upper bounds on $f^\star$. The famous result by [GW95] guarantees a data-independent approximation factor of 0.87856 to the maximum cut problem. Their analysis is based on the fact that the SDR (22) can be interpreted as a stochastic version of (1) [BY99, LMS$^+$10]. This interpretation gives a natural probability distribution to sample points from, which can be used to implement a

randomized *Suggest* method in step 1 of Algorithm 1.1. There are other problem instances where similar approximation factors can be given. These approximation factors, however, greatly depend on the underlying problems. We refer the readers to [LMS$^+$10] for a summary of some major results on SDR approximation factors.

Let $(X^\star, x^\star)$ be any optimal solution of (22). Suppose $x \in \mathbf{R}^n$ is a Gaussian random variable with mean μ and covariance Σ. Then, $\mu = x^\star$ and $\Sigma = X^\star - x^\star x^{\star T}$ solve the following problem of minimizing the expected value of a quadratic form, subject to quadratic inequalities:

$$\begin{array}{ll} \text{minimize} & \mathbf{E}\, f_0(x) = \mathbf{Tr}(P_0\Sigma) + f_0(\mu) \\ \text{subject to} & \mathbf{E}\, f_i(x) = \mathbf{Tr}(P_i\Sigma) + f_i(\mu) \leq 0, \quad i = 1, \ldots, m \\ & \Sigma \succeq 0, \end{array}$$

in variables $\mu \in \mathbf{R}^n$ and $\Sigma \in \mathbf{R}^{n \times n}$. Intuitively, minimizing the expected objective value promotes μ to move closer to the minimizer of f_0 and Σ to be "small," the constraints counteract this and promote Σ to be "large" enough that the constraints hold in expectation. Since the constraints are satisfied only in expectation, there is no guarantee that sampling points directly from $\mathcal{N}(\mu, \Sigma)$ gives feasible points of (1) at all. In particular, if (1) includes an equality constraint, then it will almost certainly fail. However, recall that the *Suggest* methods do not require feasibility; sampling candidate points from $\mathcal{N}(\mu, \Sigma)$ is a reasonable choice for the *Suggest* method. These candidate points then serve as good starting points for the *Improve* methods, which we discuss in detail in §4.

In certain cases, the probabilistic interpretation also allows us to get a hard bound on the gap between the optimal values $f^\star$ of the original problem (1) and f^{sdr} of its SDR (22). A classic example is that of the maximum cut bound described in [GW95], which states that for an undirected graph with nonnegative edge weights, the SDR of the maximum cut problem attains the following bound:

$$\alpha f^{\mathrm{sdr}} \leq f^\star \leq f^{\mathrm{sdr}},$$

for $\alpha \approx 0.87856$. (Note that this is a maximization problem.)

Here, we derive a similar bound of $\alpha = 2/\pi \approx 0.63661$. This result, also known as Nesterov's $\pi/2$ theorem, extends the result of [GW95] to any two-way partitioning problem with $W \succeq 0$. Note that we do not require the off-diagonal entries of W to be nonpositive, which is necessary to get the stronger bound of $\alpha \approx 0.87856$ in [GW95].

Theorem 2 (Nesterov [Nes98a, Nes98b]). *Let f^{sdr} be the optimal value of the SDR of* (12), *where $W \succeq 0$. Then,*

$$\frac{2}{\pi} f^{\mathrm{sdr}} \leq f^\star \leq f^{\mathrm{sdr}}.$$

Proof. The SDR of (12) is:

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(WX) \\ \text{subject to} & X \succeq 0 \\ & X_{ii} = 1, \quad i = 1, \ldots, n. \end{array}$$

Let $X^\star$ be any solution of the SDR, and $f^{\text{sdr}} = \textbf{Tr}(W X^\star)$ be the optimal value of the SDR. Consider drawing x from the Gaussian distribution $\mathcal{N}(0, X^\star)$, and setting $z = \textbf{sign}(x)$, where $\textbf{sign}(\cdot)$ denotes the elementwise sign function. Note that z is always a feasible point of (12). The special form of the objective function allows us to find the expected value of the objective $\mathbf{E}(z^T W z)$ analytically:

$$\mathbf{E}(z^T W z) = \sum_{i,j} W_{ij}\, \mathbf{E}(z_i z_j) = \frac{2}{\pi} \sum_{i,j} W_{ij} \arcsin(X^\star_{ij}) = \frac{2}{\pi} \textbf{Tr}(W \arcsin X^\star),$$

where $\arcsin X^\star$ is a matrix obtained by taking elementwise arcsin of the entries of $X^\star$. Since $\arcsin(X^\star) \succeq X^\star$ (see, *e.g.*, [BTN01, §3.4.1.6]) and W is positive semidefinite, we get

$$\mathbf{E}(z^T W z) = \frac{2}{\pi} \textbf{Tr}(W \arcsin X^\star) \geq \frac{2}{\pi} \textbf{Tr}(W X^\star) = \frac{2}{\pi} f^{\text{sdr}}.$$

On the other hand, since z is always feasible, we have

$$\mathbf{E}(z^T W z) \leq f^\star.$$

Together with the fact that the relaxation attains the optimal value f^{sdr} no worse than $f^\star$, *i.e.*, $f^{\text{sdr}} \geq f^\star$, we have

$$\frac{2}{\pi} f^{\text{sdr}} \leq \mathbf{E}(z^T W z) \leq f^\star \leq f^{\text{sdr}}.$$

$\square$

Note that this result not only bounds f^{sdr} in terms of $f^\star$, but also shows an explicit procedure for generating feasible points that have the expected objective value of at least $(2/\pi) f^{\text{sdr}}$. In §4.1, we revisit this procedure in the context of *Improve* methods. In practice, sampling just a few points is enough to obtain a feasible solution that exceeds this theoretical lower bound.

3.4   Tightening relaxations

Lower bounds obtained from relaxations can be improved by adding additional quadratic inequalities to (1) that are satisfied by any solution of the original problem. In particular, redundant inequalities that hold for all feasible points of (1) can still tighten the relaxation. We note, however, that in order for these inequalities to be useful in practice, they must be computationally efficient to derive. For example, the inequality $f_0(x) \leq f^\star$ holds for every optimal point of the problem, but it cannot be added to the problem without knowing the value of $f^\star$. All the valid inequalities we discuss below, therefore, will be restricted to the ones that can be derived efficiently.

Consider the set of affine inequalities $Ax \leq b$ in (2). For every vector x satisfying $Ax \leq b$, we have

$$(Ax - b)(Ax - b)^T = Axx^T A^T - Axb^T - bx^T A + bb^T \geq 0, \tag{23}$$

where the inequality is elementwise. Each entry of the lefthand side has the form

$$(a_i^T x - b_i)(a_j^T x - b_j) = x^T a_i a_j^T x - (b_i a_j + b_j a_i)^T x + b_i b_j,$$

where $a_i \in \mathbf{R}^n$ is the ith row of A (considered as a column vector). These indefinite quadratic inequalities then can be added to the original QCQP (2) without changing the set of solutions.

In certain special cases, there are other valid inequalities that can be derived directly from the structure of the feasible set. For example, take any Boolean problem where the feasible set is given by $\{-1, +1\}^n$. Since the entries of any feasible x are integer-valued, for any $a \in \mathbf{Z}^n$ and $b \in \mathbf{Z}$, we have

$$(a^T x - b)(a^T x - (b + 1)) \geq 0.$$

While these are redundant inequalities, adding them to the problem can tighten its relaxations.

This technique can be generalized further; any exclusive-disjunction of two affine inequalities can be encoded as a quadratic inequality. Let $a^T x \leq b$ and $c^T x \leq d$ be two affine inequalities such that for every feasible point of (1), exactly one of them holds. Then,

$$(a^T x - b)(c^T x - d) \leq 0$$

is a redundant quadratic inequality that holds for every feasible point x.

3.5   Relaxation of relaxations

The Lagrangian and semidefinite relaxations (19) and (22) are polynomial-time solvable, but in practice, can be expensive to solve as the dimension of the problem gets larger. In this section, we explore several ways to further relax the relaxation methods discussed above to obtain lower bounds on $f^\star$ more efficiently.

We first discuss how to further relax the Lagrangian relaxation (19). Weak duality implies that it is not necessary to solve (19) optimally in order to obtain a lower bound on $f^\star$; any feasible point (λ, α) of (19) induces a lower bound on $f^\star$. We note that α is easy to optimize given λ. In fact, when λ is fixed, optimizing over α is equivalent to solving the spectral relaxation (18) with the same value of λ.

Now, we discuss relaxation methods for the SDR (22). Note that the semidefinite constraint $Z(X, x) \succeq 0$ can be written as an infinite collection of affine constraint $a^T Z(X, x) a \geq 0$ for all $a \in \mathbf{R}^{n+1}$ of unit length, $i.e.$, $\|a\|_2 = 1$. For example, if a is the ith unit vector, the resulting inequality states that X_{ii} must be nonnegative. To approximate the optimal value f^{sdr} of (22), one can generate affine inequalities to replace the semidefinite constraint and solve the resulting linear program (LP). While these affine inequalities can come directly from the valid inequalities we discussed in §3.4, it is also possible to adopt a cutting-plane method to generate them incrementally.

Algorithm 3.1 *Cutting-plane method for solving (22) via LP relaxation.*

---

given an optimization problem $\mathcal{P}$ of the form (22).

1. *Initialize.* Add, for each $i = 1, \ldots, m$, constraint $F_i(X, x) \leq 0$ to $\mathcal{T}$, the list of constraints.

repeat

 2. Solve (22) with constraints $\mathcal{T}$ and get an optimal point $(X^\star, x^\star)$.

 3. **if** $X^\star \succeq x^\star x^{\star T}$, terminate with solution $(X^\star, x^\star)$.

 4. Otherwise, find $a \in \mathbf{R}^{n+1}$ such that $a^T Z(X^\star, x^\star) a < 0$.

 5. Add constraint $a^T Z(X, x) a \geq 0$ to $\mathcal{T}$.

In step 4, vector $a \in \mathbf{R}^{n+1}$ satisfying the condition always exists, because $Z(X^\star, x^\star) \succeq 0$ is equivalent to $X^\star \succeq x^\star x^{\star T}$. For example, one can always take a equal to the eigenvector corresponding to any negative eigenvalue of $Z(X^\star, x^\star)$. It is also possible to adapt the LDL factorization algorithm to verify whether $Z(X^\star, x^\star)$ is positive semidefinite, and terminate with a suitable vector a in case it is not. We note that Algorithm 3.1, in general, need not converge, unless additional constraints are met, *e.g.*, the vector a in step 4 is the eigenvector corresponding to the minimum eigenvalue of $Z(X^\star, x^\star)$. However, at every iteration of step 2, we get a lower bound on $f^\star$ that is no worse than the value from the previous iteration. In practice, this means that the algorithm can terminate any time when a good enough lower bound is obtained.

It is also possible to write a second-order cone programming (SOCP) relaxation of (22), which can be thought of as a middle-point between LP and SDP relaxations [KK01]. Second-order cone (SOC) constraints are more general than affine constraints, and can encode more sophisticated relations that hold for positive semidefinite matrices. For example, in order for $Z(X, x)$ to be positive semidefinite, every 2-by-2 principal submatrices of it must be positive semidefinite, *i.e.*, for every pair of indices i and j,

$$\begin{bmatrix} Z_{ii} & Z_{ij} \\ Z_{ji} & Z_{jj} \end{bmatrix} \succeq 0,$$

or equivalently,

$$Z_{ij}^2 \leq Z_{ii} Z_{jj}, \quad Z_{ii} \geq 0, \quad Z_{jj} \geq 0.$$

These three inequalities are also equivalent to the following SOC inequalities [BV04, §4]:

$$\left\| \begin{bmatrix} 2Z_{ij} \\ Z_{ii} - Z_{jj} \end{bmatrix} \right\|_2 \leq Z_{ii} + Z_{jj}, \quad Z_{ii} \geq 0, \quad Z_{jj} \geq 0. \tag{24}$$

In general, in order for $Z(X, x)$ to be positive semidefinite, we must have, for every $(n+1)$-by-2 matrix A,

$$A^T Z(X, x) A \succeq 0,$$

which can be rewritten as SOC inequalities, just like (24). From this observation, it is simple to adapt Algorithm 3.1 to solve (22) via SOCP relaxation.

4 Local methods

In this section, we implement the *improve* method of Algorithm 1.1 using various local methods. These methods take an arbitrary point $x \in \mathbf{R}^n$ that is not necessarily feasible in (1), and attempts to find a better point $z \in \mathbf{R}^n$. Recall that in §1, we defined better points in terms of maximum constraint violation and objective value. In general, finding a feasible point is an NP-hard problem, for otherwise we can perform bisection on the optimal value of the epigraph form (3) and find a solution to arbitrary precision in polynomial time. For this reason, we do not guarantee convergence of the methods or feasibility of the resulting points, except in some special cases discussed in §4.1.

4.1 Special cases

We start by investigating some special cases, where we can directly exploit the problem structure and find a feasible point. Since we are guaranteed to find a feasible point with these methods, we also get an upper bound on $f^\star$ by evaluating the objective function at the resulting point. As it can be seen from the examples, these heuristics are highly problem dependent.

Partitioning problems. The feasible set of the partitioning problem (12) is

$$\mathcal{S} = \{x \mid x_1^2 = \cdots = x_n^2 = 1\}.$$

For any given $x \in \mathbf{R}^n$, the point $z = \mathbf{sign}(x)$ is always feasible. It is easy to check that z is a projection of x onto $\mathcal{S}$, *i.e.*, z is the closest feasible point to x.

For the maximum graph bisection problem described in page 13, we can employ a slightly different method by [BY99] to satisfy the additional constraint $\mathbf{1}^T x = 0$. Given $x \in \mathbf{R}^n$, we find a feasible z by setting $z_i = 1$ for indices i corresponding to the $n/2$ largest entries in x, and $z_i = -1$ for the other $n/2$ indices. (Ties are broken arbitrarily when some entries of x have equal values.) It can be shown that z is also a projection onto the set of feasible points.

Multicast downlink transmit beamforming. The feasible set of (17) is given by

$$\mathcal{S} = \{x \mid x^T P_1 x \geq 1, \ldots, x^T P_m x \geq 1\}.$$

Let $x \in \mathbf{R}^n$ be an arbitrary point. By setting

$$z = \frac{1}{\min_i x^T P_i x} x,$$

we get a feasible point z [LMS+10]. In other words, z is the smallest multiple of x that makes it feasible. (Recall that every P_i is positive semidefinite.) Unlike in the case of partitioning problems, z is not a projection of x onto $\mathcal{S}$, even when $m = 1$.

Maximum clique. Here, we adapt the main idea from the heuristic for the maximum graph bisection problem and generate a feasible point of the maximum clique problem (15) from a given vector $x \in \mathbf{R}^n$.

Algorithm 4.1 *Heuristic for finding a clique from a given vector.*

given $x \in \mathbf{R}^n$.
1. Find a permutation π of $\{1, \ldots, n\}$ such that $x_{\pi_1} \geq \cdots \geq x_{\pi_n}$.
2. *Initialize empty clique.* $C := \emptyset$ and $z := (0, \ldots, 0) \in \mathbf{R}^n$.
for $k = 1, 2, \ldots, n$
      3. **if** $C \cup \{\pi_k\}$ forms a clique, $C := C \cup \{\pi_k\}$ and $z_{\pi_k} := 1$.
4. **return** z.

The intuition behind this heuristic is that x_i with larger value should be more likely to be included in a clique. In particular, this heuristic always includes the node i with the highest value of x_i. It is clear that at the end of Algorithm 4.1, C is a clique. In addition, we can show that C is a maximal clique, *i.e.*, there is no node $k \notin C$ such that $C \cup \{k\}$ is a clique.

Lemma 3. *The clique generated by Algorithm 4.1 is maximal.*

Proof. Assume on the contrary that the clique C_1 generated by Algorithm 4.1 is a proper subset of some other clique C_2. Choose an arbitrary $k \in C_2 \setminus C_1$. When k is considered in Algorithm 4.1, the partial clique C maintained by the algorithm is a subset of C_1. Since C_2 is a clique and $C_1 \subset C_2$, $C \cup \{k\}$ is a clique and thus k must have been included in C, which leads to a contradiction. $\qquad\square$

4.2 Coordinate descent

From this section on, we consider general QCQPs that have no obvious structures that can be exploited as in §4.1. First, we show a *coordinate descent* heuristic for improving a given point. Coordinate descent is a simple and intuitive method for finding a local minimum of a function. For more results and references on general coordinate descent algorithms, see [Wri15].

Our greedy descent method is based on the fact that one-variable QCQPs are tractable. The algorithm consists of two phases:

Phase I. The goal of the first phase is to reduce the maximum constraint violation and, if possible, reach a feasible point. Let $x \in \mathbf{R}^n$ be a given candidate point. We repeatedly cycle over each coordinate x_j of x, and update it to the value that minimizes the maximum constraint violation. In other words, at each step, we solve:

$$
\begin{aligned}
\text{minimize} \quad & s \\
\text{subject to} \quad & f_i(x) \leq s, \quad i = 1, \ldots, m,
\end{aligned}
\tag{25}
$$

with variables $x_j, s \in \mathbf{R}$. For a fixed value of s, it is easy to adapt the method of Appendix A to check if (25) is feasible in x_j. Therefore, by performing bisection on s, we can find the minimum possible value of s to arbitrary precision, as well as a value of x_j that attains the maximum violation of s. We note that in order to apply the method of Appendix A, we need to extract the quadratic, linear, and constant coefficients of each f_i in x_j. If the P_i matrices are not sparse, then evaluating these coefficients can dominate the running time.

When the optimal value of s is zero or smaller, $i.e.$, a feasible point is found, then phase I ends and phase II begins. On the other hand, if the maximum constraint violation cannot be improved for any x_j, then phase I terminates unsuccessfully (and needs a new candidate point).

Phase II.   Phase II starts once a feasible point is found. In this phase, we restrict ourselves to feasible points only, and look for another feasible point with strictly better objective value. Again, we cycle over each coordinate x_j of x and optimize the objective function while maintaining feasibility. In other words, we solve (1) with all variables fixed but x_j. The implementation of phase II is a direct application of the method in Appendix A. While it is possible to run phase II until no improving direction is remaining, it can terminate at any point and will still yield a feasible point, as well as an upper bound on $f^\star$.

Phase II of the coordinate descent method generalizes local search methods for many combinatorial problems, such as the 1-opt local search heuristic, which have been studied since the 1950s [Cro58].

4.3   Convex-concave procedure

The convex-concave procedure (CCP) is a powerful heuristic method for finding a local optimum of difference-of-convex (DC) programming problems, which have the following form:

$$
\begin{array}{ll}
\text{minimize} & f_0(x) - g_0(x) \\
\text{subject to} & f_i(x) - g_i(x) \leq 0, \quad i = 1, \ldots, m,
\end{array}
\tag{26}
$$

where $f_i : \mathbf{R}^n \to \mathbf{R}^n$ and $g_i : \mathbf{R}^n \to \mathbf{R}^n$ for $i = 0, \ldots, m$ are convex. We refer the readers to [LB16] for extensive review and bibliography of CCP.

The main motivation for considering CCP to solve QCQPs is that any quadratic function can be easily rewritten as a DC expression. Consider, for example, an indefinite quadratic expression: $x^T P x + q^T x + r$. We decompose the matrix P into the difference of two positive semidefinite matrices:

$$
P = P_+ - P_-, \qquad P_+, \ P_- \succeq 0.
\tag{27}
$$

Such decomposition is always possible, by taking, for example, $P_+ = P + tI$ and $P_- = tI$ for large enough $t > 0$. Then, we can explicitly rewrite the expression as the difference of two convex quadratic expressions:

$$
(x^T P_+ x + q_0^T x + r_0) - x^T P_- x.
$$

Once (1) is rewritten as a DC problem, any CCP method for locally solving DC problems can be applied. Here, we consider the penalty CCP method in [LB16], which does not require the initial point to be feasible. The penalty CCP method seeks to optimize the convexified version of the objective, with an additional penalty on the convexified constraint violations. The method gradually increases the penalty on constraint violation over iterations. Convexification of the functions is done by linearizing each g_i around the current iterate x^k:

$$\hat{g}_i(x; x^k) = g_i(x^k) + \nabla g_i(x^k)^T (x - x^k). \tag{28}$$

Formally, the penalty CCP method can be written as below.

Algorithm 4.2 *Penalty CCP.*

given initial point x^0, penalty parameters $\tau_0 > 0$, $\tau_{\max} > 0$, and $\mu > 1$.
for $k = 0, 1, \ldots$
    1. *Convexify.* According to (28), form $\hat{g}_i(x; x^k)$ for $i = 0, \ldots, m$.
    2. *Solve.* Set the value of x^{k+1} to a solution of
        minimize $f_0(x) - \hat{g}_0(x; x^k) + \tau_k \sum_{i=1}^m s_i$
        subject to $f_i(x) - \hat{g}_i(x; x^k) \leq s_i, \quad i = 1, \ldots, m$
                  $s_i \geq 0, \quad i = 1, \ldots, m$.
    3. *Increase constraint violation penalty.* $\tau_{k+1} := \min(\mu \tau_k, \tau_{\max})$.
until stopping criterion is satisfied.

There are a number of reasonable stopping criteria [LB16]. For example, Algorithm 4.2 can terminate when a feasible point is found, or the maximum penalty parameter is reached, *i.e.*, $\tau_k = \tau_{\max}$. When implementing Algorithm 4.2, there are other factors to consider than the initial point or the penalty parameters; the performance of the algorithm can vary depending on how indefinite quadratic functions are split into the difference of two convex quadratic expressions. In Appendix C, we discuss various ways of splitting quadratic functions into convex parts.

4.4   Alternating directions method of multipliers

The alternating directions method of multipliers (ADMM) is an operator splitting algorithm that is originally devised to solve convex optimization problems [BPC+11]. However, due to the flexibility of the ADMM framework, it has been explored as a heuristic to solve nonconvex problems. (See, *e.g.*, [BPC+11, §9], or [DBEY13, HS16].) Here, we consider an adaptation of the algorithm to the QCQP, as considered in [HS16]. Note that due to nonconvexity of QCQPs, typical convergence results on ADMM do not apply.

Let $\mathcal{C} \subseteq \mathbf{R}^n$ be a given set, and consider the following variant of (1):

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & x \in \mathcal{C}. \end{array} \tag{29}$$

With $\mathcal{C} = \mathbf{R}^n$, problem (29) is the same as (1). In this section, we use (29) as our primary problem formulation rather than (1), and handle the last constraint $x \in \mathcal{C}$ differently from the other constraints.

To apply ADMM, we take (29) and form an equivalent problem with a consensus constraint:

$$\begin{array}{ll} \text{minimize} & f_0(z) + \mathcal{I}_{\mathcal{C}}(z) + \sum_{i=1}^m \mathcal{I}_i(x_i) \\ \text{subject to} & z = x_1 = \cdots = x_m, \end{array}$$

with variables $z, x_1, x_2, \ldots, x_m \in \mathbf{R}^n$. It is important to note that x_i does not represent the ith component of x; rather, it represents the ith copy of the variable x, all of which should be equal to each other. The function $\mathcal{I}_{\mathcal{C}}$ is the 0–∞ indicator of the set $\mathcal{C}$:

$$\mathcal{I}_{\mathcal{C}}(z) = \left\{ \begin{array}{ll} 0 & z \in \mathcal{C} \\ \infty & z \notin \mathcal{C}. \end{array} \right.$$

Similarly, $\mathcal{I}_i$ is the 0–∞ indicator function of the constraint $f_i(x) \leq 0$:

$$\mathcal{I}_i(x) = \left\{ \begin{array}{ll} 0 & f_i(x) \leq 0 \\ \infty & f_i(x) > 0. \end{array} \right.$$

The augmented Lagrangian of the problem is:

$$\mathcal{L}_\rho(z, x_1, \ldots, x_m, u_1, \ldots, u_m) = f_0(z) + \mathcal{I}_{\mathcal{C}}(z) + \sum_{i=1}^m \left(\mathcal{I}_i(x_i) + \rho \left(\|z - x_i + u_i\|_2^2 - \|u_i\|_2^2 \right) \right),$$

where each $u_i \in \mathbf{R}^n$ are *scaled dual variables* [BPC$^+$11, §3.1.1]. The *penalty parameter $\rho > 0$* controls the convergence behavior or ADMM. We postpone the discussion on how to choose the parameter, and describe the ADMM iteration first:

$$\begin{array}{rll} z^{k+1} &=& \underset{z}{\text{argmin}} \, \mathcal{L}_\rho(z, x_1^k, \ldots, x_m^k, u_1^k, \ldots, u_m^k), \\ x_i^{k+1} &=& \underset{x_i}{\text{argmin}} \, \mathcal{L}_\rho(z^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_m^k, u_1^k, \ldots, u_m^k), \quad i = 1, \ldots, m, \\ u_i^{k+1} &=& u_i^k + z^{k+1} - x_i^{k+1}, \quad i = 1, \ldots, m. \end{array}$$

The update rules for z and x each involves solving an optimization problem, but some observations allow us to simplify them. To simplify the z-update, we ignore the terms in the augmented Lagrangian that do not depend on z. Then, z^{k+1} is given by the solution of the following QCQP:

$$\begin{array}{ll} \text{minimize} & f_0(z) + \rho \sum_{i=1}^m \|z - (x_i^k - u_i^k)\|_2^2 \\ \text{subject to} & z \in \mathcal{C}. \end{array} \tag{30}$$

in the variable $z \in \mathbf{R}^n$. There are a number of cases where (30) is tractable.

- When the penalty parameter $\rho > 0$ satisfies

$$\lambda_{\min} + m\rho \geq 0,$$

  where $\lambda_{\min}$ is the smallest eigenvalue of P_0, then (30) is a convex problem and thus is tractable.

- If the constraint $z \in \mathcal{C}$ can be written as a single quadratic inequality constraint, then (30) is a QCQP with a single constraint (5), which is tractable regardless of convexity of $\mathcal{C}$ or the value of ρ.

- When $\mathcal{C} = \mathbf{R}^n$ and the penalty parameter ρ is chosen such that the objective function is strictly convex, *i.e.*,

$$\lambda_{\min} + m\rho > 0,$$

  then we can perform the z-update by solving a single linear system. To see this, rewrite the objective function of (30) as

$$z^T \tilde{P} z + 2\tilde{q}^T z + \tilde{r},$$

  with appropriate $\tilde{P} \in \mathbf{R}^{n \times n}$, $\tilde{q} \in \mathbf{R}^n$, and $\tilde{r} \in \mathbf{R}$. Since $\tilde{P} \succ 0$, the minimizer of the expression above can be found by simply setting the gradient with respect to z equal to zero:

$$2\tilde{P}z + 2\tilde{q} = 0,$$

  or equivalently,

$$\tilde{P}z = -\tilde{q}.$$

  Assuming that the value of ρ is fixed for every iteration, the coefficient matrix $\tilde{P}$ on the lefthand side stays the same for every z-update. Using this observation, we can perform the z-update efficiently as follows: we compute the factorization of the coefficient matrix $\tilde{P}$ once, and for every subsequent z-update, use the cached factorization to carry out the back-solve. The improvement in running time is the most significant when $\tilde{P}$ is dense, as the cost of factorization is $O(n^3)$, and each back-solve only takes $O(n^2)$ time.

The x-updates can be simplified as well by ignoring terms that do not depend on x_i. Then, x_i^{k+1} is given by the solution of the following QCQP:

$$
\begin{array}{ll}
\text{minimize} & \|x_i - (z^{k+1} + u_i^k)\|_2^2 \\
\text{subject to} & f_i(x_i) \leq 0,
\end{array}
\tag{31}
$$

with variable $x_i \in \mathbf{R}^n$. Note that (31) is a special form of (5), which is tractable. We cover the solution methods for solving (31) in Appendix B. Since (31) only depends on z^{k+1} and u_i^k, all x-updates trivially parallelizes over $i = 1, \ldots, m$.

Below, we discuss several extensions of the ADMM-based *Improve* method.

Equality constraints. When some constraints are equality constraints, then only the corresponding x-updates need to be modified accordingly. For example, if the ith constraint is an equality constraint $f_i(x) = 0$, then in order to perform the x-update, we would solve

$$
\begin{array}{ll}
\text{minimize} & \|x_i - (z^{k+1} + u_i^k)\|_2^2 \\
\text{subject to} & f_i(x_i) = 0,
\end{array}
$$

instead of (31).

Convex constraints. Note that any number of convex constraints of (1) can be encoded in the constraint $x \in \mathcal{C}$. In general, this will change the behavior of the algorithm, including how quickly the z-update (30) can be performed.

Two-phase ADMM. While the ADMM update rules take a simple form and are easy to implement, it is still a heuristic applied to a nonconvex problem, and thus in practice, it may be more important to set the initial point and the penalty parameter ρ carefully. Here, we show an adaptation of the two-phase ADMM method by Huang and Sidiropoulos [HS16], that can attain faster convergence in practice. In phase I, much like the two-phase coordinate descent algorithm introduced in §4.2, the algorithm first focuses on finding a feasible point by ignoring the objective function. Once a feasible point is found, phase II begins. In phase II, the objective function is brought back into consideration and ADMM iterations are performed until convergence.

The only difference of the two phases is that in phase I, the objective function f_0 is completely ignored. This simplifies the z-update of phase I; in order to perform the z-update, we solve the following optimization problem:

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{m} \| z - (x_i^k - u_i^k) \|_2^2 \\
\text{subject to} & z \in \mathcal{C},
\end{array}
\tag{32}
$$

with variable $z \in \mathbf{R}^n$. Define

$$
\bar{z} = \frac{1}{m} \sum_{i=1}^{m} (x_i^k - u_i^k).
$$

The solution of (32) is simply given by the projection of $\bar{z}$ onto $\mathcal{C}$. As long as $\mathcal{C}$ is convex, the projection can be found efficiently. The x- and u-update rules stay the same. Notice that the new z-update rule is independent of ρ. In other words, the ADMM iterates in phase I are completely determined by the initial point.

We note that depending on the initialization, the iterates in phase I can be stuck in an infinite loop of period greater than 1, even for very small problems. Consider a 3-dimensional two-way partitioning problem (12), which has three equality constraints

$$
x_i^2 = 1, \quad i = 1, 2, 3.
$$

Consider the following initial points:

$$
\begin{array}{rcl}
z^0 & = & (1/3)\mathbf{1}, \\
x_1^0 & = & (-1, 1/3, 1/3), \\
x_2^0 & = & (1/3, -1, 1/3), \\
x_3^0 & = & (1/3, 1/3, -1), \\
u_1^0 & = & (2/3, 0, 0), \\
u_2^0 & = & (0, 2/3, 0), \\
u_3^0 & = & (0, 0, 2/3).
\end{array}
$$

For this particular initialization, we get

$$z^{k+1} = -z^k, \quad x_i^{k+1} = -x_i^k, \quad u_i^{k+1} = -u_i^k,$$

and thus the iterates repeat themselves with period 2. It can be verified that any initialization close to this will also fail to converge to a feasible point.

5 Implementation

We introduce an open source Python package `QCQP` that accepts high-level description QC-QPs as input and implements Algorithm 1.1. As our main platform, we used `CVXPY`, a domain-specific language for convex optimization [DB16]. The source code repository for `QCQP` is available at `https://github.com/cvxgrp/qcqp`.

5.1 Quadratic expressions

The `CVXPY` parser determines curvature, sign, and monotonicity of expressions according to the *disciplined convex programming* (DCP) rules [GBY06]. We have extended the parser so that it can also determine quadraticity of an expression, based on the rules that are similar to the DCP rules. The following list of expressions is directly recognized as quadratic by `CVXPY`:

- any constant or affine expression

- any affine transformation of a quadratic expression, *e.g.*, the sum of quadratic expressions

- product of two affine expressions

- elementwise square `power(X, 2)` or `square(X)`, with affine `X`

- `sum_squares(X)` with affine `X`, representing $\sum_{ij} X_{ij}^2$.

- `quad_over_lin(X, c)` with affine `X` and positive constant `c`, representing $(1/c) \sum_{ij} X_{ij}^2$

- `matrix_frac(x, P)` with affine `x` and symmetric constant `P`, representing $x^T P^{-1} x$

- `quad_form(x, P)` with affine `x` and symmetric constant `P`, representing $x^T P x$

5.2 Constructing problem and applying heuristics

Our implementation can handle QCQPs constructed using the standard `CVXPY` syntax. As long as the objective function and both sides of the constraints are quadratic, the problem is accepted even when it is nonconvex. In order to apply the *Suggest-and-Improve* framework, a `CVXPY` problem object must be passed to the `QCQP` constructor first. For example, if `problem` is a `CVXPY` problem object, then the following code checks whether `problem` describes a QCQP, and if so, prepares the *Suggest* and *Improve* methods:

```
qcqp = QCQP(problem)
```

Once the `qcqp` object is constructed, a number of different *Suggest* and *Improve* methods can be invoked on it. Currently, three *Suggest* methods are available:

- `qcqp.suggest()` or `qcqp.suggest(RANDOM)` fills the values of the variables using independent and identically distributed Gaussian random variables.

- `qcqp.suggest(SPECTRAL)` fills the values of the variables with a solution of the spectral relaxation (18). The spectral lower bound (or upper bound, in the case of a maximization problem) on the optimal value $f^\star$ is accessible via `qcqp.spectral_bound`.

- `qcqp.suggest(SDR)` solves the SDR (22) and fills the values of the variables according to the probabilistic interpretation discussed in page 19. The SDR bound on the optimal value is accessible via `qcqp.sdr_bound`.

Below is a list of available *Improve* methods:

- `qcqp.improve(COORD_DESCENT)` performs the two-stage coordinate descent method, as described in §4.2.

- `qcqp.improve(DCCP)` rewrites the problem in the DC form (26) then runs the penalty CCP method in §4.3 using the open source Python package `DCCP`.

- `qcqp.improve(ADMM)` runs the two-phase ADMM, as described in §4.4.

As mentioned in §1.3, composition of any number of *Improve* methods is also an *Improve* method. It is easy to apply a sequence of *Improve* methods by passing a list of methods to `improve()`:

```
qcqp.improve(method_sequence)
```

This is equivalent to:

```
for method in method_sequence:
    qcqp.improve(method)
```

Various parameters can be supplied to the *Suggest* and *Improve* method, such as the penalty parameter ρ of the two-phase ADMM, penalty parameter τ of the penalty CCP, maximum number of iterations, and tolerance value for determining near-zero quantities. All *Suggest* and *Improve* methods return a pair $(f_0(x), v(x))$, *i.e.*, the objective value and maximum constraint violation at the current point x.

5.3   Sample usage

In this section, we show a sample usage of the `QCQP` package with a small two-way partitioning problem (12) with $n = 10$. We start by importing the necessary packages, and constructing a symmetric matrix $W \in \mathbf{R}^{n \times n}$:

```
import numpy as np, cvxpy as cvx
from qcqp import *

n = 10
W0 = np.random.randn(n, n)
W = 0.5*(W0 + W0.T)
```

For clarity, we imported `NumPy` and `CVXPY` with explicit namespaces. Next, we construct a `CVXPY` problem instance describing (12).

```
x = cvx.Variable(n)
prob = cvx.Problem(
    cvx.Maximize(cvx.quad_form(x, W)),
    [cvx.square(x) == 1]
)
```

While `CVXPY` allows defining `prob`, it is not possible to invoke the `solve()` method on it because the objective function is not concave, and the constraints are nonconvex. However, we can pass it as an argument to the `QCQP` constructor to indicate that `prob` is a QCQP:

```
qcqp = QCQP(prob)
```

Now we can call `suggest()` and `improve()` methods on `qcqp`. Here, we solve the spectral relaxation (18) with $\lambda = \mathbf{1}$, for which both the optimal value and solution are known. Although it is not necessary, we use the MOSEK solver [ApS17] for robustness.

```
qcqp.suggest(SPECTRAL, solver=cvx.MOSEK)
```

This fills `x.value`, the numerical value of `x`, with the solution of the spectral relaxation. The spectral bound can be accessible via `qcqp.spectral_bound`. We print out this value and compare it with $f^{\mathrm{rl}} = n\lambda_{\max}$ shown in §3.1.

```
print ("Spectral bound: %.4f" % qcqp.spectral_bound)

(w, v) = np.linalg.eig(W)
print ("n*lambda_max:   %.4f" % (max(w)*n))
```

Indeed, we see that both values coincide:

```
Spectral bound: 31.2954
n*lambda_max:   31.2954
```

We can also verify that the value `x.value` is correctly populated with the (scaled) eigenvector of W corresponding to its maximum eigenvalue. Next, we apply the two-phase coordinate descent heuristic on `x.value`, and print out the objective value and maximum constraint violation at the resulting point.

```
f_cd, v_cd = qcqp.improve(COORD_DESCENT)
print ("Objective: %.4f" % f_cd)
print ("Maximum violation: %.4f" % v_cd)
```

In this example, we get a feasible point:

```
Objective: 23.1687
Maximum violation: 0.0000
```

We can print out `x.value` and check that every coordinate is indeed ± 1. Since n is small, we can enumerate every one of $2^n = 1024$ feasible points and verify that `x.value` is a global solution of the problem. See Appendix D for the full version of the script.

6 Numerical examples

In this section, we consider two numerical examples and perform the heuristics implemented in the `QCQP` package. In Appendix D, we give sample Python script for writing these example problems as QCQPs and applying the *Suggest-and-Improve* framework using our package. More examples can be found in the package repository.

Specialized methods. We note that any kind of method tailored to particular problems, *e.g.*, maximum cut (13), multicast beamforming (17), or other well-known problems described in §2, will almost certainly perform better than `QCQP`, and our objective is not to compete with these specialized methods. The primary goal of the package, instead, is to provide an easily accessible interface to various heuristics for NP-hard QCQPs that do not have specialized methods.

Running time. Currently, `QCQP` supports minimal parallelism and introduces computational overhead by explicitly representing quadratic expressions by their coefficient matrices. Further optimizing the performance of the heuristics is left as future work. The reported running times are CPU times (measured using the Python `time.clock()` function) based on experiments performed on a 3.40 GHz Intel Xeon machine, running Ubuntu 16.04.

6.1 Boolean least squares

Here, we consider the Boolean least squares problem (11) from §2.2.

Problem instance. We generated a random problem instance with $n = 50$ and $m = 80$, where the entries of A and b are drawn IID from $\mathcal{N}(0, 1)$. The feasible set has $2^n \approx 10^{15}$ points.

Results. We tested various combinations of *Suggest* and *Improve* methods. For each combination, we sampled 20 candidate points and improved them, and took the best point. In this example, we considered all three different *Suggest* methods in §5.2: random, spectral, and SDR. For the *Improve* methods, we considered three methods as follows:

- Round: rounding the candidate point to the nearest feasible point, as discussed in §4.1.

- CD: two-phase coordinate descent, as discussed in §4.2.

- CCP: penalty CCP, as discussed in §4.3.

While it is possible to apply multiple *Improve* methods sequentially, we did not consider them in this experiment. We excluded the two-phase ADMM because it easily fails to generate a feasible point when applied to Boolean problems, as noted in page 29. For the penalty CCP heuristic, we used the penalty parameter of $\tau = 1$.

All three *Improve* methods yielded a feasible point for every candidate point generated by *Suggest* methods. In Table 1, we show, for every combination of the *Suggest* and *Improve* methods, the objective value of the best point found and the total running time (which includes the solve time of the relaxations).

Spectral and semidefinite relaxations yielded lower bounds of 228 and 518, respectively. The best feasible point found had the objective value of 988, and it was obtained by performing coordinate descent on SDR-based candidate points. Since the problem instance was small enough to find the global solution, we used Gurobi [GO15] to find the optimal value $f^\star$, which was 920. As pointed out in [PB17, dB03], this optimal value is closer to the best upper bound of 988, rather than to the SDR bound of 518.

Rounding a random point to the nearest feasible point is equivalent to choosing a random feasible point. As expected, this method performed the worst: the best objective value attained was 2719. Significantly better points were obtained by changing the *Suggest* method: rounding the solution of the spectral relaxation gave a point with objective 1605, and rounding the SDR-based candidate points gave the best objective of 1098. The performance was further improved by performing the two-phase coordinate descent *Improve* method. With two-phase coordinate descent, even random candidate points yielded a better point than rounding SDR-based candidate points. This result is quite intuitive, because coordinate descent is a natural heuristic choice for Boolean problems. In fact, applying the two-phase coordinate descent in this setting reduces to a heuristic that performs well in practice [Cro58], which is to round the candidate point to the nearest point (phase I), and perform a 1-opt local search (phase II). In this example, the penalty CCP found the same feasible point for every given candidate point, regardless of the *Suggest* method.

In terms of the running time, we found, for this problem, that there is no benefit in running the penalty CCP or two-phase ADMM; the two-phase coordinate descent yielded

Best point	Random	Spectral	SDR	Runtime	Random	Spectral	SDR
Round	2719	1605	1098	Round	0.0	26.2	41.3
CD	1043	1017	988	CD	67.3	93.8	199.9
CCP	1063	1063	1063	CCP	872.7	877.9	968.5

Table 1: Objective value of the best point found (left), and total running time (right) of the *Suggest-and-Improve* heuristic on the Boolean least squares problem.

the best point faster than them. We note that the running time of Gurobi for finding the optimal value was 1793 seconds (CPU time).

6.2   Secondary user multicast beamforming

We consider the secondary user multicase beamforming problem, which is a variant of (17) studied in [PVST09, HS16]. The problem can be formulated as the following:

$$
\begin{array}{ll}
\text{minimize} & \|w\|_2^2 \\
\text{subject to} & |h_i^H w|^2 \geq \tau, \quad i = 1, \ldots, m \\
& |g_j^H w|^2 \leq \eta, \quad j = 1, \ldots, l.
\end{array} \tag{33}
$$

Here, $w \in \mathbf{C}^n$ is the variable, and $h_i, g_j \in \mathbf{C}^n$ are given problem data, for $i = 1, \ldots, m$ and $j = 1, \ldots, l$. Since CVXPY and QCQP do not directly support complex-valued variables and data, we form an equivalent problem with real numbers only. For $i = 1, \ldots, m$, let

$$
a_i = (\Re h_i, \Im h_i), \quad b_i = (-\Im h_i, \Re h_i),
$$

be real-valued vectors in $\mathbf{R}^{2n}$. (Here, $\Re z \in \mathbf{R}^n$ and $\Im z \in \mathbf{R}^n$ denote the real and imaginary parts of a complex-valued vector $z \in \mathbf{C}^n$.) Similarly, for $j = 1, \ldots, l$, define

$$
c_j = (\Re g_j, \Im g_j), \quad d_j = (-\Im g_j, \Re g_j).
$$

By introducing another variable $x \in \mathbf{R}^{2n}$ to represent the real and imaginary parts of w, we can rewrite (33) using real-valued variables and data only:

$$
\begin{array}{ll}
\text{minimize} & \|x\|_2^2 \\
\text{subject to} & (a_i^T x)^2 + (b_i^T x)^2 \geq \tau, \quad i = 1, \ldots, m \\
& (c_j^T x)^2 + (d_j^T x)^2 \leq \eta, \quad j = 1, \ldots, l.
\end{array}
$$

Problem instance.   We considered a problem instance with $n = 50$, $m = 20$, and $l = 5$. The entries of the a_i, b_i, c_j, and d_j were drawn IID from $\mathcal{N}(0, 1)$. The other parameters were set as $\tau = 20$ and $\eta = 2$.

Results. As in §6.1, we used a combination of *Suggest* and *Improve* methods. The list of *Improve* method we used for (33) is the following:

- Scale: scaling the candidate point so that it satisfies the first set of constraints,

$$(a_i^T x)^2 + (b_i^T x)^2 \geq \tau, \quad i = 1, \dots, m,$$

  as discussed in §4.1. Unlike (17), this problem has a second set of constraints, and therefore, this method does not necessarily yield a feasible point.

- CD

- ADMM

- ADMM/CD

- CD/ADMM

- CCP

For the two-phase ADMM, we chose the penalty parameter $\rho = \sqrt{m+l}$, as in [HS16]. For the other heuristics, we used the default parameters. For each choice of *Suggest* and *Improve* methods, we sampled 10 candidate points and improved them, and took the best point.

In Table 2, we show, for every combination of the *Suggest* and *Improve* methods, the objective value of the best feasible point found and the total running time (which includes the solve time of the relaxations). The Scale *Improve* method never produced a feasible point for all three *Suggest* methods, and therefore we left it out from the table.

Spectral and semidefinite relaxations yielded lower bounds of 1.11 and 1.27, respectively. The best feasible point found had the objective value of 1.30, obtained by performing the penalty CCP on SDR-based candidate points.

Unlike in §6.1, the two-phase coordinate descent did not produce a good feasible point. On random candidate points, the best feasible point it found had the objective value of 9.62. Even with the SDR-based candidate points, the best objective value was 2.51. The two-phase ADMM, on the other hand, produced a point with objective value 1.86 when combined with the SDR *Suggest* method. The penalty CCP method showed a more consistent performance for both random and SDR *Suggest* methods, and it found the best point with objective 1.30. However, it was unable to produce a feasible point from the solution of the spectral relaxation.

We note that running ADMM followed by coordinate descent performed better than applying only one of the heuristics. However, running the two heuristics in the other order did not find a good feasible point. This implies that depending on the problem, the individual *Improve* methods can be considered as building blocks for more sophisticated *Improve* sequences that can produce a better point.

The running time of the ADMM and penalty CCP *Improve* methods was heavily affected when they were unable to find feasible points. This issue can be addressed by specifying the maximum number of iterations performed by the methods, or prematurely terminating the methods when they reach a preset time limit.

Best point	Random	Spectral	SDR
CD	9.62	9.90	2.51
ADMM	7.84	1.98	1.86
ADMM/CD	4.64	1.73	1.42
CD/ADMM	11.18	8.70	3.07
CCP	1.31	N/A	1.30

Runtime	Random	Spectral	SDR
CD	195.3	572.1	790.7
ADMM	1843	1237	8355
ADMM/CD	1636	1286	6241
CD/ADMM	1518	1317	2173
CCP	5626	4770	2216

Table 2: Objective value of the best point found (left), and total running time (right) of the *Suggest-and-Improve* heuristic on the secondary user multicast beamforming problem.

Acknowledgement

A Solving QCQP with one variable

In this section, we consider the following one-variable QCQP:

$$\begin{array}{ll} \text{minimize} & p_0 x^2 + q_0 x + r_0 \\ \text{subject to} & p_i x^2 + q_i x + r_i \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{34}$$

with variable $x \in \mathbf{R}$. We show an $O(m \log m)$ time solution method for solving (34) using only elementary algebra and data structures. Consider the ith constraint

$$p_i x^2 + q_i x + r_i \leq 0,$$

and the set S_i of x satisfying the constraint. Assuming that S_i is nontrivial, *i.e.*, S_i is not equal to $\mathbf{R}$ or $\emptyset$, it can take three different forms, depending on the sign of p_i:

- If $p_i = 0$, then the constraint is affine, and thus S_i is a left- or right-unbounded interval. That is, S_i has the form $(-\infty, a]$ or $[a, +\infty)$.

- If $p_i > 0$, then S_i is a closed interval $[a, b]$.

- If $p_i < 0$, then S_i is the union of a left-unbounded and a right-unbounded intervals, *i.e.*, $S_i = (-\infty, a] \cup [b, +\infty)$, for some a and b (with $a < b$).

We inductively argue that the feasible set $S = S_1 \cap \cdots \cap S_m$ is a collection of at most $m+1$ disjoint, potentially left- or right-unbounded intervals on $\mathbf{R}$. The base case where $m = 0$ gives $S = \mathbf{R}$, which satisfies the claim. Now, assume that S is a collection of at most $m+1$ disjoint intervals on $\mathbf{R}$. We claim that $S \cap S_{m+1}$ is a collection of at most $m+2$ disjoint intervals on $\mathbf{R}$. This holds trivially if S_{m+1} is a convex interval. The only case where additional intervals can be introduced is when $p_{m+1} < 0$. Let $S_{m+1} = (-\infty, a] \cup [b, +\infty)$, or equivalently, $S_{m+1} = \mathbf{R} \setminus (a, b)$. Taking the intersection of S and S_{m+1} is the same as "subtracting" an open interval $U = (a, b)$ from every interval C in S. Subtracting an open interval U from a closed interval C results in another closed interval C', unless $U \subset C$, in which case $C \setminus U$ is two disjoint closed intervals. Since the intervals in S are disjoint, there can be at most one C such that $U \subset C$ holds. This completes the inductive step, *i.e.*, $S \cap S_{m+1}$ consists of at most $m+2$ disjoint intervals.

To compute these disjoint intervals, we use a balanced binary search tree, where each node corresponds to an interval, ordered by the left endpoint. When taking the intersection of the current intervals with S_i, at most two intervals can change their endpoints, and some intervals can be removed from the data structure. The case where $p_i < 0$ is the only case we potentially insert an additional interval.

Every insertion, deletion, or modification (which can be thought of as deletion followed by insertion) takes $O(\log N)$ time, where N is the maximum number of nodes in the binary search tree at any given time. To analyze the overall time complexity, we can simply count the number of each operation types. As argued above, the total number of insertions is bounded by m. The number of deletions is naturally bounded by the number of insertions,

which is also m. The number of modifications is bounded by $2m$. It follows that the time complexity of computing the feasible set is $O(m \log m)$.

Minimizing a quadratic function in this feasible set $\mathcal{S}$ can be done by evaluating the objective at the endpoints of the intervals, as well as checking the unconstrained minimizer (only when $p_0 > 0$).

B Solving QCQP with one constraint

In this section, we describe the solution method for the following special case of (5):

$$
\begin{array}{ll}
\text{minimize} & \|x - z\|_2^2 \\
\text{subject to} & x^T P x + q^T x + r \le 0,
\end{array}
\tag{35}
$$

with variable $x \in \mathbf{R}^n$. For simplicity, we assume that the constraint is satisfiable for some $x \in \mathbf{R}^n$. Satisfiability is easily verified from the eigenvalue decomposition of P.

If z satisfies the constraint, *i.e.*, $z^T P z + q^T z + r \le 0$, then it is clear that $x^\star = z$ is the solution and there is nothing else to do. Otherwise, due to complementary slackness [BV04, §5.5.2], any optimal point $x^\star$ must satisfy the constraint with equality:

$$
x^{\star T} P x^\star + q^T x^\star + r = 0.
$$

Then, without loss of generality, we can work with the equality constrained version of (35):

$$
\begin{array}{ll}
\text{minimize} & \|x - z\|_2^2 \\
\text{subject to} & x^T P x + q^T x + r = 0.
\end{array}
\tag{36}
$$

Let $P = Q \Lambda Q^T$ be the eigenvalue decomposition of P, with $\Lambda = \mathbf{diag}(\lambda_1, \ldots, \lambda_n)$. Consider the following problem, equivalent to (36):

$$
\begin{array}{ll}
\text{minimize} & \|\hat{x} - \hat{z}\|_2^2 \\
\text{subject to} & \hat{x}^T \Lambda \hat{x} + \hat{q}^T \hat{x} + r = 0,
\end{array}
\tag{37}
$$

where $\hat{z} = Q^T z$, $\hat{q} = Q^T q$, and $\hat{x} = Q^T x$.

The Lagrangian of (37) is given by:

$$
\mathcal{L}(\hat{x}, \nu) = \hat{x}^T (I + \nu \Lambda) \hat{x} + (\nu \hat{q} - 2\hat{z})^T \hat{x} + \nu c + \|\hat{z}\|_2^2.
$$

Since we assumed the feasibility of (37), there must exist ν with $I + \nu \Lambda \succeq 0$, such that the value of $\hat{x}$ minimizing the Lagrangian also satisfies the equality constraint of (37).

There are two cases to consider:

- Case (i): $I + \nu \Lambda \succ 0$.

  First, consider the range of ν so that $I + \nu \Lambda \succ 0$, or equivalently, $1 + \nu \lambda_i > 0$ for all i. In this range, we can find the value of $\hat{x}$ minimizing the Lagrangian by taking the derivative with respect to $\hat{x}$ and setting it to zero:

$$
\hat{x} = -(1/2)(I + \nu \Lambda)^{-1}(\nu \hat{q} - 2\hat{z}).
\tag{38}
$$

By substituting $\hat{x}$ into the equality constraint and expanding out, we get a nonlinear equation in ν:

$$\sum_{i=1}^{n} \left(\frac{\lambda_i(\nu\hat{q}_i - 2\hat{z}_i)^2}{4(1+\nu\lambda_i)^2} - \frac{\hat{q}_i(\nu\hat{q}_i - 2\hat{z}_i)}{2(1+\nu\lambda_i)} \right) + r = 0. \tag{39}$$

If this equation attains a solution in the range where $1 + \nu\lambda_i > 0$ for all i, then the corresponding $\hat{x}$ given by (38) is the solution to (37). In order to solve (39) numerically, observe that the derivative of the lefthand side is

$$-\sum_{i=1}^{n} \frac{(2\lambda_i\hat{z}_i + \hat{q}_i)^2}{2(1+\lambda_i\nu)^3} < 0,$$

and thus the lefthand side monotonically decreases in ν. Then, one can solve (39) by checking where the sign of the lefthand side changes, using either bisection or Newton's method.

- Case (ii): $I + \nu\Lambda \succeq 0$ and $I + \nu\Lambda$ is singular.
  There are at most two values of ν where this is possible. If $\lambda_{\min} = \min_i \lambda_i < 0$, then $\nu = -1/\lambda_{\min}$ is one such value. If $\lambda_{\max} = \max_i \lambda_i > 0$, then $\nu = -1/\lambda_{\max}$ is another such value. For these values of ν, we can simply check if there exists $\hat{x}$ where the Karush-Kuhn-Tucker (KKT) conditions [BV04, §5.5.3] are held:

$$2(I + \nu\Lambda)\hat{x} = 2\hat{z} - \nu\hat{q}, \qquad \hat{x}^T\Lambda\hat{x} + \hat{q}^T\hat{x} + r = 0.$$

Once $\hat{x}$ is found, then the solution $x^\star$ of (35) is simply given by $x^\star = Q\hat{x}$.

C   Splitting quadratic forms

In this appendix, we explore various ways of splitting an indefinite matrix P as a difference $P_+ - P_-$ of two positive semidefinite matrices, as in (27). The motivation for this discussion is that the performance of convex-concave procedure can vary drastically depending on how the quadratic form is split into convex and concave parts [LB16, SDGB16].

C.1   Desired properties

In addition to the running time, there are several other properties of the algorithm that are important in typical applications.

Time complexity.   Clearly, we want the algorithm for computing the representation to be fast. Asymptotic performance is important, but in practice, even with the same asymptotic complexity, one method may outperform another depending on how sparsity is exploited.

Memory usage. In the worst case where both P_+ and P_- are dense, $2n^2$ floating point numbers must be stored. If P has a special sparsity pattern that can be exploited, it is often desirable to keep the pattern as much as possible, and represent P_+ and P_- without introducing too many nonzeros. Ideally, representing P_+ and P_- should not cost much more than representing P itself.

Numerical stability. Due to the round-off errors of floating point numbers, the resulting representation may be numerically inaccurate. This can be particularly problematic when P_+ and P_- are represented implicitly (*e.g.*, via Cholesky factorization), rather than explicitly. Singular or badly conditioned matrices can often introduce big round-off errors, and we want the algorithm to be robust in such settings.

Additional curvature. We want P_+ and P_- to introduce as little "distortion" as possible, *i.e.*, we want the additional curvature measured by the nuclear norm

$$\|P_+\|_* + \|P_-\|_* - \|P\|_*$$

to be small. Since P_+ and P_- are required to be positive semidefinite, this quantity is equal to

$$\mathbf{Tr}\, P_+ + \mathbf{Tr}\, P_- - \|P\|_*.$$

C.2 Simple representations

Any indefinite matrix can be made positive semidefinite by adding a large enough multiple of the identity. Let $\lambda_{\min} < 0$ be the smallest eigenvalue of P. Then, for any $t \geq |\lambda_{\min}|$,

$$P_+ = P + tI, \qquad P_- = tI,$$

is a pair of positive semidefinite matrices whose difference is P. If the magnitude of the maximum eigenvalue $\lambda_{\max} > 0$ is smaller than that of $\lambda_{\min}$, then an alternative representation is also possible:

$$P_+ = tI, \qquad P_- = tI - P,$$

where $t \geq \lambda_{\max}$. This representation is relatively easy to compute as it only requires a lower bound on $|\lambda_{\min}|$ or $|\lambda_{\max}|$. It also has a property that the sparsity of P is preserved as much as possible, in that no new off-diagonal nonzero entries are introduced in P_+ or P_-. Its disadvantage is the additional curvature it introduces when t is large:

$$\|P_+\|_* + \|P_-\|_* - \|P\|_* = 2t.$$

Another simple representation is based on the full eigenvalue decomposition of P. This representation preserves the norm of P and thus introduces no additional curvature. Let $P = Q\Lambda Q^T$ be the eigenvalue decomposition of P, where $\Lambda = \mathbf{diag}(\lambda_1, \ldots, \lambda_n)$ is the eigenvalue matrix with

$$\lambda_1 \geq \cdots \geq \lambda_k \geq 0 > \lambda_{k+1} \geq \cdots \geq \lambda_n.$$

Then, Λ can be written as $\Lambda = \Lambda_+ - \Lambda_-$, where

$$\Lambda_+ = \mathbf{diag}(\lambda_1, \dots, \lambda_k, 0, \dots, 0), \qquad \Lambda_- = \mathbf{diag}(0, \dots, 0, -\lambda_{k+1}, \dots, -\lambda_n).$$

Setting

$$P_+ = Q\Lambda_+ Q^T, \qquad P_+ = Q\Lambda_- Q^T$$

then gives a difference of positive semidefinite matrix representation of P. This approach has a high computational cost from the full eigenvalue decomposition (which takes $O(n^3)$ time). Also, in general, even when P is sparse, the resulting matrices P_+ and P_- are dense.

C.3 Cholesky-like representations

When P is positive definite, there exists a unique lower triangular matrix $L \in \mathbf{R}^{n \times n}$, called the *Cholesky factor* of P, that satisfies $P = LL^T$. This representation is known as the *Cholesky factorization* of P (see, *e.g.*, [BV04, §C.3]). In this subsection, we explore a representation based on the Cholesky factorization. We start by describing the Cholesky algorithm, which, in the simplest form, is a recursive algorithm. If P is 1-by-1, then L is simply given by $\sqrt{P_{11}}$. If P has two or more rows, let

$$P = \begin{bmatrix} a & v^T \\ v & M \end{bmatrix},$$

with $a > 0$, $v \in \mathbf{R}^{n-1}$, $M \in \mathbf{S}^{n-1}$. Then, the Cholesky factor L of P is given by:

$$L = \begin{bmatrix} \sqrt{a} & \\ v/\sqrt{a} & L' \end{bmatrix},$$

where L' is the Cholesky factor of $M - (1/a)vv^T$. The cost of computing a dense Cholesky factorization is $(1/3)n^3$ flops. In case P is sparse, various pivoting heuristics can be used to exploit the sparsity structure and speed up the computation [NW99, §6.3].

Cholesky factorization does not exist when P is indefinite. However, the LDL decomposition, which is a close variant of the Cholesky factorization, exists for all symmetric matrices [BP71]. It also has an additional computational advantage since there is no need to take square roots. The idea of the LDL decomposition is to write P as LDL^T, where $L \in \mathbf{R}^{n \times n}$ is lower triangular with ones on the main diagonal, and $D \in \mathbf{R}^{n \times n}$ is block diagonal consisting of 1-by-1 or 2-by-2 blocks. When D is diagonal (*i.e.*, no 2-by-2 blocks), then one can easily separate out the negative entries of D and the corresponding columns in L to write P as

$$P = L_1 D_1 L_1^T - L_2 D_2 L_2^T,$$

where D_1 and D_2 are nonnegative diagonal matrices. When D has 2-by-2 blocks, however, it is not possible to directly transform the LDL decomposition into the desired form.

C.4 Difference-of-Cholesky representation

In this subsection, we develop an adaptation of the Cholesky decomposition that returns a "difference-of-Cholesky" representation. There are several advantages of this approach over the method discussed in §C.3. First, P can be any indefinite matrix, $i.e.$, its LDL decomposition can contain 2-by-2 blocks. Additionally, the method has a parameter $\delta > 0$ that controls the numerical robustness against division by small numbers; the algorithm will never perform division by numbers whose magnitude is smaller than $\sqrt{\delta}$, at the expense of additional curvature. Finally, various heuristics for Cholesky decomposition that promote sparsity can be applied directly to this modification.

Let $P \in \mathbf{S}^n$ be a symmetric matrix of two or more rows:

$$P = \begin{bmatrix} a & v^T \\ v & M \end{bmatrix},$$

where $a \in \mathbf{R}$, $v \in \mathbf{R}^{n-1}$, $M \in \mathbf{S}^{n-1}$. Our goal is to find a pair of lower triangular matrices L_1 and L_2 such that

$$P = L_1 L_1^T - L_2 L_2^T.$$

As in §C.3, we show a recursion for L_1 and L_2. Without loss of generality, we assume that $a \geq 0$; if $a < 0$, we can simply find the difference-of-Cholesky representation for $-P$ and swap L_1 and L_2. Then, there are only two cases to consider, depending on the magnitude of a.

- Case (i): $a > \delta$.
  In this case, the recursion is given by:

  $$L_1 = \begin{bmatrix} \sqrt{a} & \\ v/\sqrt{a} & L_1' \end{bmatrix}, \quad L_2 = \begin{bmatrix} 0 & \\ 0 & L_2' \end{bmatrix}.$$

  Here, L_1' and L_2' are lower triangular matrices satisfying

  $$L_1' L_1'^T - L_2' L_2'^T = M - (1/a)vv^T,$$

  which can be obtained by recursively applying the algorithm.

- Case (ii): $0 \leq a \leq \delta$.
  The recursion in this case has the following form:

  $$L_1 = \begin{bmatrix} \sqrt{\delta + a} & \\ v_1 & L_1' \end{bmatrix}, \quad L_2 = \begin{bmatrix} \sqrt{\delta} & \\ v_2 & L_2' \end{bmatrix}.$$

  Here, v_1 and v_2 are arbitrary vectors satisfying

  $$\sqrt{\delta + a}\, v_1 - \sqrt{\delta}\, v_2 = v,$$

  and L_1' and L_2' are lower triangular matrices with

  $$L_1' L_1'^T - L_2' L_2'^T = M - v_1 v_1^T + v_2 v_2^T.$$

Note that we have an additional degree of freedom, as we can freely choose v_1 or v_2. For example, by letting $v_1 = 0$ or $v_2 = 0$, it is possible to trade off the number of nonzero elements between L_1 and L_2, which is a property that was not readily available in the other representations we discussed. When we choose $v_1 = 0$, the recursion becomes

$$L_1 = \begin{bmatrix} \sqrt{\delta + a} & \\ 0 & L_1' \end{bmatrix}, \quad L_2 = \begin{bmatrix} \sqrt{\delta} & \\ -v/\sqrt{\delta} & L_2' \end{bmatrix},$$

with

$$L_1' L_1'^T - L_2' L_2'^T = M + (1/\delta)vv^T.$$

Similarly, if we choose $v_2 = 0$, we get

$$L_1 = \begin{bmatrix} \sqrt{\delta + a} & \\ v/\sqrt{\delta + a} & L_1' \end{bmatrix}, \quad L_2 = \begin{bmatrix} \sqrt{\delta} & \\ 0 & L_2' \end{bmatrix},$$

with

$$L_1' L_1'^T - L_2' L_2'^T = M - \frac{1}{\delta + a} vv^T.$$

This method is very close to the original Cholesky algorithm, and most extensions and techniques applicable to Cholesky factorization naturally apply to the difference-of-Cholesky method. For example, it is simple to modify this algorithm to return a pair of LDL factorizations to avoid computing square roots. Other examples include pivoting heuristics for avoiding round-off errors or ill-conditioned matrices.

D Source examples

In this section, we give the full version of the script used in §5.3, as well as sample implementations of the *Suggest-and-Improve* framework for the numerical examples considered in §6. The codes are written using our open source package QCQP. We put special emphasis on the similarities between the description of Algorithm 1.1 and the actual codes, as well as the short length of the codes.

D.1 Sample usage

```
import numpy as np, cvxpy as cvx
from qcqp import *

np.random.seed(1)

# Problem data
n = 10
W0 = np.random.randn(n, n)
```

```
W = 0.5*(W0 + W0.T)

# Construct a nonconvex QCQP
x = cvx.Variable(n)
prob = cvx.Problem(
    cvx.Maximize(cvx.quad_form(x, W)),
    [cvx.square(x) == 1]
)
qcqp = QCQP(prob)

# Solve the spectral relaxation
qcqp.suggest(SPECTRAL, solver=cvx.MOSEK)

# Print bounds
print ("Spectral bound: %.4f" % qcqp.spectral_bound)
(w, v) = np.linalg.eig(W)
print ("n*lambda_max:   %.4f" % (max(w)*n))

# Print solution of the relaxation
print(x.value)
x_rl = np.sqrt(n) * v[:, np.argmax(w)]
print(x_rl)

# Use coordinate descent heuristic
f_cd, v_cd = qcqp.improve(COORD_DESCENT)
print ("Objective: %.4f" % f_cd)
print ("Maximum violation: %.4f" % v_cd)
print(x.value)

# Find the optimal solution by enumeration
f_star = -np.inf
mask_star = None
for mask in range(2**n):
    x = np.array([2*((mask>>i)&1)-1 for i in range(n)])
    f = np.dot(W.dot(x), x)
    if f_star < f:
        f_star = f
        mask_star = mask
print(f_star)

x_star = [2 * ((mask_star>>i)&1) - 1 for i in range(n)]
print(x_star)
```

D.2 Boolean least squares

```
import numpy as np, cvxpy as cvx
from qcqp import *

# Problem data
n, m = 10, 15
A = np.random.randn(m, n)
b = np.random.randn(m, 1)

# Construct a nonconvex QCQP
x = cvx.Variable(n)
prob = cvx.Problem(
    cvx.Minimize(cvx.sum_squares(A*x - b)),
    [cvx.square(x) == 1]
)
qcqp = QCQP(prob)

# Solve the SDR and get a starting point to a local method
qcqp.suggest(SDR, solver=cvx.MOSEK)
print (qcqp.sdr_bound)

# Attempt to improve the starting point using coordinate descent
f, v = qcqp.improve(COORD_DESCENT)
print (f, v) # objective value, maximum violation
print (x.value) # value of the variable
```

D.3 Secondary user multicast beamforming

```
import numpy as np, cvxpy as cvx
from cvxpy import *
from qcqp import *

# Problem data
n, m, l = 30, 10, 3
tau, eta = 20, 2
HR, HI = np.random.randn(m, n), np.random.randn(m, n)
GR, GI = np.random.randn(l, n), np.random.randn(l, n)
A, B = np.hstack((HR, HI)), np.hstack((-HI, HR))
C, D = np.hstack((GR, GI)), np.hstack((-GI, GR))

# Construct a nonconvex QCQP
x = cvx.Variable(2*n)
```

```
prob = cvx.Problem(
    cvx.Minimize(cvx.sum_squares(x)),
    [
        cvx.square(A*x) + cvx.square(B*x) >= tau,
        cvx.square(C*x) + cvx.square(D*x) <= eta
    ]
)
qcqp = QCQP(prob)

# Solve the SDR and get a starting point to a local method
qcqp.suggest(SDR, solver=cvx.MOSEK)
print (qcqp.sdr_bound)

# Attempt to improve the starting point using ADMM
f, v = qcqp.improve(ADMM, rho=np.sqrt(m+l))
print (f, v) # objective value, maximum violation
print (x.value) # value of the variable
```

References

[ABKW08]   T. Achterberg, T. Berthold, T. Koch, and K. Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In *International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming*, pages 6–20. Springer, 2008.

[Ach09]   T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[ACWY99]   K. Anstreicher, X. Chen, H. Wolkowicz, and Y.-X. Yuan. Strong duality for a trust-region type relaxation of the quadratic assignment problem. *Linear Algebra and its Applications*, 301(1):121–136, 1999.

[AG03]   F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming Series B*, 95:3–51, 2003.

[AHJS00]   C. Audet, P. Hansen, B. Jaumard, and G. Savard. A branch and cut algorithm for nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, 87(1):131–152, 2000.

[AHZ11]   W. Ai, Y. Huang, and S. Zhang. New results on Hermitian matrix rank-one decomposition. *Mathematical Programming*, 128(1-2):253–283, 2011.

[AK92]   F. A. Al-Khayyal. Generalized bilinear programming: Part I. models, applications and linear programming relaxation. *European Journal of Operational Research*, 60(3):306–314, 1992.

[AKF83]   F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286, 1983.

[AKLVV95]   F. A. Al-Khayyal, C. Larsen, and T. Van Voorhis. A relaxation method for nonconvex quadratically constrained quadratic programs. *Journal of Global Optimization*, 6(3):215–230, 1995.

[Ali91]   F. Alizadeh. *Combinatorial Optimization with Interior-Point Methods and Semi-Definite Matrices*. PhD thesis, University of Minnesota, 1991.

[Ans09]   K. M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43(2-3):471–484, 2009.

[ApS17]   MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 8.0*, 2017.

[Bal68]   E. Balas. A note on the branch-and-bound principle. *Operations Research*, 16:442–445, 1968.

[Bar72]   D. P. Baron. Quadratic programming with quadratic constraints. *Naval Research Logistics Quarterly*, 19(2):253–260, 1972.

[BBB91]   V. Balakrishnan, S. Boyd, and S. Balemi. Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems. *International Journal of Robust and Nonlinear Control*, 1(4):295–317, 1991.

[BBC$^+$08] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, and N. Sawaya. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204, 2008.

[BBL14]   C. Bliek, P. Bonami, and A. Lodi. Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report. In *Research Association of Mathematical Programming Symposium*, pages 171–180. Research Association of Mathematical Programming, 2014.

[BBP95]   L. Baratchart, M. Berthod, and L. Pottier. Optimization of positive generalized polynomials under l^p constraints. Technical Report 2750, INRIA, 1995.

[BE06]   A. Beck and Y. C. Eldar. Strong duality in nonconvex quadratic optimization with two quadratic constraints. *SIAM Journal on Optimization*, 17(3):844–860, 2006.

[Bec80]   R. I. Becker. Necessary and sufficient conditions for the simultaneous diagonability of two quadratic forms. *Linear Algebra and its Applications*, 30:129–139, 1980.

[BEGFB94]   S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15. Society for Industrial and Applied Mathematics, 1994.

[Ber99]   D. P. Bertsekas. *Nonlinear Programming*. Athena scientific Belmont, 1999.

[Ber14]   D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic press, 2014.

[BGCL15]   S. Bose, D. F. Gayme, K. M. Chandy, and S. H. Low. Quadratically constrained quadratic programs on acyclic graphs with application to power flow. *IEEE Transactions on Control of Network Systems*, 2(3):278–287, 2015.

[BM03]   S. Boyd and J. Mattingley. Branch and bound methods. *EE364b Class Notes*, 2003.

[BNW06]   R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large-scale Nonlinear Optimization*, pages 35–59. Springer, 2006.

[BP71]   J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 8(4):639–655, 1971.

[BPC+11]   S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[Bri61]   L. Brickman. On the field of values of a matrix. *Proceedings of the American Mathematical Society*, 12(1):61–66, 1961.

[BTN01]   A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, volume 2. Society for Industrial and Applied Mathematics, 2001.

[BTT96]   A. Ben-Tal and M. Teboulle. Hidden convexity in some nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, 72(1):51–63, 1996.

[BV97]   S. Boyd and L. Vandenberghe. Semidefinite programming relaxations of nonconvex problems in control and combinatorial optimization. In *Communications, Computation, Control, and Signal Processing*, pages 279–287. Springer, 1997.

[BV04]   S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[BV08]   S. Burer and D. Vandenbussche. A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Mathematical Programming*, 113(2):259–282, 2008.

[BV10]   M. R. Bussieck and S. Vigerske. MINLP solver software. *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[BY99]   D. Bertsimas and Y. Ye. Semidefinite relaxations, multivariate normal distributions, and order statistics. In *Handbook of Combinatorial Optimization*, pages 1473–1491. Springer, 1999.

[Cal64]   E. Calabi. Linear systems of real quadratic forms. *Proceedings of the American Mathematical Society*, 15(5):844–846, 1964.

[CB12]   J. Chen and S. Burer. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 4(1):33–52, 2012.

[CESV15]   E. J. Candes, Y. C. Eldar, T. Strohmer, and V. Voroninski. Phase retrieval via matrix completion. *SIAM Review*, 57(2):225–251, 2015.

[CH98]   S. H. Cheng and N. J. Higham. A modified Cholesky algorithm based on a symmetric indefinite factorization. *SIAM Journal on Matrix Analysis and Applications*, 19(4):1097–1110, 1998.

[CLS15]   E. J. Candes, X. Li, and M. Soltanolkotabi. Phase retrieval via Wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

[Coo71]   S. A. Cook. The complexity of theorem-proving procedures. In *Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[CP11]   P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011.

[Cro58]   G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958.

[CS07]   T. Cour and J. Shi. Solving Markov random fields with spectral relaxation. In *Artificial Intelligence and Statistics Conference*, volume 2, page 15, 2007.

[CS13]   J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*, volume 290. Springer Science & Business Media, 2013.

[dB03]   A. d'Aspremont and S. Boyd. Relaxations and randomized methods for nonconvex QCQPs. *EE392o Class Notes*, 2003.

[DB16]   S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[DBEY13]   N. Derbinsky, J. Bento, V. Elser, and J. S. Yedidia. An improved three-weight message-passing algorithm. *arXiv preprint:1305.1961*, 2013.

[DJS96]   J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, volume 16. Society for Industrial and Applied Mathematics, 1996.

[DLW00]   T. N. Davidson, Z.-Q. Luo, and K. M. Wong. Design of orthogonal pulse shapes for communications via semidefinite programming. *IEEE Transactions on Signal Processing*, 48(5):1433–1445, 2000.

[DP93a]   C. Delorme and S. Poljak. Laplacian eigenvalues and the maximum cut problem. *Mathematical Programming*, 62(1-3):557–574, 1993.

[DP93b]   C. Delorme and S. Poljak. The performance of an eigenvalue bound on the max-cut problem in some classes of graphs. *Discrete Mathematics*, 111(1-3):145–156, 1993.

[DP00]   G. E. Dullerud and F. Paganini. *A Course in Robust Control Theory: A Convex Approach.* Springer, 2000.

[EB92]   J. Eckstein and D. P. Bertsekas. On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[Els03]   V. Elser. Phase retrieval by iterated projections. *JOSA A*, 20(1):40–55, 2003.

[EN75]   J. G. Ecker and R. D. Niemi. A dual method for quadratic programs with quadratic constraints. *SIAM Journal on Applied Mathematics*, 28(3):568–576, 1975.

[Fer00]   E. Feron. Nonconvex quadratic programming, semidefinite relaxations and randomization algorithms in information and decision systems. In *System Theory*, pages 255–274. Springer, 2000.

[FHB01]   M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference*, volume 6, pages 4734–4739. IEEE, 2001.

[FKS98]   G. C. Fehmers, L. P. J. Kamp, and F. W. Sluijter. An algorithm for quadratic optimization with one quadratic constraint and bounds on the variables. *Inverse Problems*, 14(4):893, 1998.

[Fle76]   R. Fletcher. Factorizing symmetric indefinite matrices. *Linear Algebra and its Applications*, 14(3):257–272, 1976.

[FLSX12]   J.-M. Feng, G.-X. Lin, R.-L. Sheu, and Y. Xia. Duality and solutions for quadratic programming over single non-homogeneous quadratic constraint. *Journal of Global Optimization*, 54(2):275–293, 2012.

[FPRR02]   P. Festa, P. M. Pardalos, M. G. C. Resende, and C. C. Ribeiro. Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, 17(6):1033–1058, 2002.

[FW56]   M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[Gay81]   D. M. Gay. Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, 2(2):186–197, 1981.

[GB14]   M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[GBY06]   M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. In *Global Optimization*, pages 155–210. Springer, 2006.

[GFG$^+$16] G. Gamrath, T. Fischer, T. Gally, A. M. Gleixner, G. Hendel, T. Koch, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, S. Vigerske, D. Weninger, M. Winkler, J. T. Witt, and J. Witzig. The SCIP optimization suite 3.2. Technical Report 15–60, ZIB, Takustr.7, 14195 Berlin, 2016.

[GLRT99]   N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint. Solving the trust-region subproblem using the lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999.

[GM98]   S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19(3):701–719, 1998.

[GO15]   Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.

[GQT66]   S. M. Goldfeld, R. E. Quandt, and H. F. Trotter. Maximization by quadratic hill-climbing. *Econometrica: Journal of the Econometric Society*, 34(3):541–551, 1966.

[GR13]   C. Godsil and G. F. Royle. *Algebraic Graph Theory*, volume 207. Springer Science & Business Media, 2013.

[GS15]   B. Gopalakrishnan and N. D. Sidiropoulos. High performance adaptive algorithms for single-group multicast beamforming. *IEEE Transactions on Signal Processing*, 63(16):4373–4384, 2015.

[GSS$^+$10] A. B. Gershman, N. D. Sidiropoulos, S. Shahbazpanahi, M. Bengtsson, and B. Ottersten. Convex optimization-based beamforming: From receive to transmit and network designs. *IEEE Signal Processing Magazine*, 27(3):62–75, 2010.

[GT92]   C. C. Gonzaga and M. J. Todd. An $o(\sqrt{n}l)$-iteration large-step primal-dual affine algorithm for linear programming. *SIAM Journal on Optimization*, 2(3):349–359, 1992.

[Gül91]   O. Güler. On the convergence of the proximal point algorithm for convex minimization. *SIAM Journal on Control and Optimization*, 29(2):403–419, 1991.

[Gül92a]   O. Güler. Augmented lagrangian algorithms for linear programming. *Journal of Optimization Theory and Applications*, 75(3):445–470, 1992.

[Gül92b]   O. Güler. New proximal point algorithms for convex minimization. *SIAM Journal on Optimization*, 2(4):649–664, 1992.

[GVM91]   G. H. Golub and U. Von Matt. Quadratically constrained least squares and quadratic problems. *Numerische Mathematik*, 59(1):561–580, 1991.

[GW95]   M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, 1995.

[GW15]   P. E. Gill and E. Wong. Methods for convex and general quadratic programming. *Mathematical Programming Computation*, 7(1):71–112, 2015.

[Hel00]   C. Helmberg. *Semidefinite Programming for Combinatorial Optimization*. Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000.

[Hel02]   C. Helmberg. Semidefinite programming. *European Journal of Operational Research*, 137(3):461–482, 2002.

[Hes68]   M. R. Hestenes. Pairs of quadratic forms. *Linear Algebra and its Applications*, 1(3):397–407, 1968.

[HES16a]   K. Huang, Y. C. Eldar, and N. D. Sidiropoulos. On convexity and identifiability in 1-D Fourier phase retrieval. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3941–3945. IEEE, 2016.

[HES16b]   K. Huang, Y. C. Eldar, and N. D. Sidiropoulos. Phase retrieval from 1D Fourier measurements: Convexity, uniqueness, and algorithms. *IEEE Transactions on Signal Processing*, 64(23):6105–6117, 2016.

[Hil57]   C. Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4:79–85, 1957.

[HJ91]   R. A. Horn and C. R. Johnson. Topics in matrix analysis. *Cambridge University Press*, 37:39, 1991.

[HLR16]   M. Hong, Z.-Q. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

[HLS13]   Y. Hsia, G.-X. Lin, and R.-L. Sheu. A revisit to quadratic programming with one inequality quadratic constraint via matrix pencil. *arXiv preprint:1312.1400*, 2013.

[Hor76]   R. Horst. An algorithm for nonconvex programming problems. *Mathematical Programming*, 10(1):312–321, 1976.

[HP10]   Y. Huang and D. P. Palomar. Rank-constrained separable semidefinite programming with applications to optimal beamforming. *IEEE Transactions on Signal Processing*, 58(2):664–678, 2010.

[HS16]   K. Huang and N. D. Sidiropoulos. Consensus-ADMM for general quadratically constrained quadratic programming. *IEEE Transactions on Signal Processing*, 64(20):5297–5310, 2016.

[IBM13]   IBM ILOG CPLEX Optimization Studio. *CPLEX User's Manual Version 12 Release 6*, 2013.

[JMZ14]   B. Jiang, S. Ma, and S. Zhang. Alternating direction method of multipliers for real and complex polynomial optimization models. *Optimization*, 63(6):883–898, 2014.

[JNS13]   P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Annual ACM Symposium on Theory of Computing*, pages 665–674. ACM, 2013.

[Joh14]   S. G. Johnson. The NLopt nonlinear-optimization package. `http://ab-initio.mit.edu/nlopt`, 2014.

[JOP16]   E. Jones, T. Oliphant, and P. Peterson. SciPy: open source scientific tools for Python. `http://www.scipy.org`, 2016.

[Kar72]   R. M. Karp. *Reducibility Among Combinatorial Problems*. Springer, 1972.

[KB57]   T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica: Journal of the Econometric Society*, 25(1):53–76, 1957.

[Kel60]   J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[KK01]   S. Kim and M. Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15(3-4):201–224, 2001.

[KK03]   S. Kim and M. Kojima. Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. *Computational Optimization and Applications*, 26(2):143–154, 2003.

[KK16]   S. Kim and M. Kojima. Binary quadratic optimization problems that are difficult to solve by conic relaxations. *Discrete Optimization*, 2016.

[KS15]   A. Konar and N. D. Sidiropoulos. Hidden convexity in QCQP with Toeplitz-Hermitian quadratics. *IEEE Signal Processing Letters*, 22(10):1623–1627, 2015.

[KT51]   H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, Berkeley, California, 1951.

[KTK80]   M. K. Kozlov, S. P. Tarasov, and L. G. Khachiyan. The polynomial solvability of convex quadratic programming. *USSR Computational Mathematics and Mathematical Physics*, 20(5):223–228, 1980.

[KVV04]   R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the Association for Computing Machinery*, 51(3):497–515, 2004.

[Lan05]   K. Lang. Fixing two weaknesses of the spectral method. In *Advances in Neural Information Processing Systems*, pages 715–722, 2005.

[LB16]   T. Lipp and S. Boyd. Variations and extension of the convex-concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

[LH06]   M. Leordeanu and M. Hebert. Efficient MAP approximation for dense energy functions. In *International Conference on Machine Learning*, pages 545–552. ACM, 2006.

[Lin05]   J. Linderoth. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Mathematical Programming*, 103(2):251–282, 2005.

[LMS+10]   Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, and S. Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, 2010.

[LO96]   A. S. Lewis and M. L. Overton. Eigenvalue optimization. *Acta Numerica*, 5:149–190, 1996.

[LO+99]   C. Lemaréchal, F. Oustry, et al. Semidefinite relaxations and Lagrangian duality with application to combinatorial optimization. Technical Report 3710, INRIA, 1999.

[Lov79]   L. Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information Theory*, 25(1):1–7, 1979.

[LS09]   F. Lauer and C. Schnörr. Spectral clustering of linear subspaces for motion segmentation. In *International Conference on Computer Vision*, pages 678–685. IEEE, 2009.

[LS15]   A. P. Liavas and N. D. Sidiropoulos. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(20):5450–5463, 2015.

[LSTZ07]   Z.-Q. Luo, N. D. Sidiropoulos, P. Tseng, and S. Zhang. Approximation bounds for quadratic optimization with homogeneous quadratic constraints. *SIAM Journal on Optimization*, 18(1):1–28, 2007.

[LSY16]   A. Lemon, A. M.-C. So, and Y. Ye. Low-rank semidefinite programming: Theory and applications. *Foundations and Trends in Optimization*, 2(1-2):1–156, 2016.

[Luo03]   Z.-Q. Luo. Applications of convex optimization in signal processing and digital communication. *Mathematical Programming Series B*, 97:177–207, 2003.

[LVBL98]   M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1):193–228, 1998.

[LW66]   E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[MDW+02]   W.-K. Ma, T. N. Davidson, K. M. Wong, Z.-Q. Luo, and P.-C. Ching. Quasi-maximum-likelihood multiuser detection using semi-definite relaxation with application to synchronous CDMA. *IEEE Transactions on Signal Processing*, 50:912–922, 2002.

[MF12]   R. Misener and C. A. Floudas. Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. *Mathematical Programming*, 136(1):155–182, 2012.

[MF13]   R. Misener and C. A. Floudas. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, 57:3–30, 2013.

[MF14a]   R. Misener and C. A. Floudas. ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014.

[MF14b]   R. Misener and C. A. Floudas. A framework for globally optimizing mixed-integer signomial programs. *Journal of Optimization Theory and Applications*, 161(3):905–932, 2014.

[MHG+15]   O. Mehanna, K. Huang, B. Gopalakrishnan, A. Konar, and N. D. Sidiropoulos. Feasible point pursuit and successive approximation of non-convex QCQPs. *IEEE Signal Processing Letters*, 22(7):804–808, 2015.

[Mit02]   J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*, pages 65–77. Oxford University Press, 2002.

[MKB07]   A. Mutapcic, S.-J. Kim, and S. Boyd. A tractable method for robust downlink beamforming in wireless communications. In *Asilomar Conference on Signals, Systems and Computers*, pages 1224–1228. IEEE, 2007.

[Moo91]   R. E. Moore. Global optimization to prescribed accuracy. *Computers & Mathematics with Applications*, 21(6):25–39, 1991.

[Mor78]   J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical Analysis*, pages 105–116. Springer, 1978.

[Mor93]   J. J. Moré. Generalizations of the trust region problem. *Optimization Methods and Software*, 2(3-4):189–209, 1993.

[MS83]   J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[MSF15]   R. Misener, J. B. Smadbeck, and C. A. Floudas. Dynamically generated cutting planes for mixed-integer quadratically constrained quadratic programs and their incorporation into GloMIQO 2. *Optimization Methods and Software*, 30(1):215–249, 2015.

[MTF11]   R. Misener, J. P. Thompson, and C. A. Floudas. APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Computers & Chemical Engineering*, 35(5):876–892, 2011.

[MWA06]   B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *International Conference on Machine Learning*, pages 641–648. ACM, 2006.

[Nes98a]   Y. Nesterov. Global quadratic optimization via conic relaxation. Technical report, UCL, 1998.

[Nes98b]   Y. Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9(1-3):141–160, 1998.

[NJS13]   P. Netrapalli, P. Jain, and S. Sanghavi. Phase retrieval using alternating minimization. In *Advances in Neural Information Processing Systems*, pages 2796–2804, 2013.

[NN94]   Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Methods in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[NW99]   J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[PB17]   J. Park and S. Boyd. A semidefinite programming method for integer convex quadratic minimization. *Optimization Letters*, 2017.

[PR91]   M. Padberg and G. Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100, 1991.

[PR95]   S. Poljak and F. Rendl. Solving the max-cut problem using eigenvalues. *Discrete Applied Mathematics*, 62(1):249–278, 1995.

[PRW95]   S. Poljak, F. Rendl, and H. Wolkowicz. A recipe for semidefinite relaxation for (0, 1)-quadratic programming. *Journal of Global Optimization*, 7(1):51–73, 1995.

[PS75]   C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[PT07]   I. Pólik and T. Terlaky. A survey of the $\mathcal{S}$-lemma. *SIAM Review*, 49(3):371–418, 2007.

[PV91]   P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1(1):15–22, 1991.

[PVST09]   K. T. Phan, S. A. Vorobyov, N. D. Sidiropoulos, and C. Tellambura. Spectrum sharing in wireless networks via QoS-aware secondary multicast beamforming. *IEEE Transactions on Signal Processing*, 57(6):2323–2335, 2009.

[QBM12]   A. Qualizza, P. Belotti, and F. Margot. Linear programming relaxations of quadratically constrained quadratic programs. In *Mixed Integer Nonlinear Programming*, pages 407–426. Springer, 2012.

[QSH$^+$16]   C. Qian, N. D. Sidiropoulos, K. Huang, L. Huang, and H. C. So. Phase retrieval using feasible point pursuit: Algorithms and Cramér-Rao bound. *IEEE Transactions on Signal Processing*, 64(20):5282–5296, 2016.

[Rab98]   U. Raber. A simplicial branch-and-bound method for solving nonconvex all-quadratic programs. *Journal of Global Optimization*, 13(4):417–432, 1998.

[Rab99]   U. Raber. *Nonconvex all-quadratic global optimization problems: solution methods, application and related topics*. PhD thesis, University of Trier, 1999.

[RW97]   F. Rendl and H. Wolkowicz. A semidefinite framework for trust region subproblems with applications to large scale minimization. *Mathematical Programming*, 77(1):273–299, 1997.

[Sah74]   S. Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4):262–279, 1974.

[Sah14]   N. V. Sahinidis. *BARON 14.3.1: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual*, 2014.

[SDGB16]   X. Shen, S. Diamond, Y. Gu, and S. Boyd. Disciplined convex-concave programming. In *Conference on Decision and Control*, pages 1009–1014. IEEE, 2016.

[SDL06]   N. D. Sidiropoulos, T. N. Davidson, and Z.-Q. Luo. Transmit beamforming for physical-layer multicasting. *IEEE Transactions on Signal Processing*, 54(6):2239–2251, 2006.

[SEC+15]   Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev. Phase retrieval with application to optical imaging: a contemporary overview. *IEEE Signal Processing Magazine*, 32(3):87–109, 2015.

[SGC97]   C. Scherer, P. Gahinet, and M. Chilali. Multiobjective output-feedback control via LMI optimization. *IEEE Transactions on Automatic Control*, 42(7):896–906, 1997.

[Sho87]   N. Z. Shor. Quadratic optimization problems. *Soviet Journal of Computer and Systems Sciences*, 25(6):1–11, 1987.

[SM00]   J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[Sor82]   D. C. Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, 1982.

[ST95]   H. D. Sherali and C. H. Tuncbilek. A reformulation-convexification approach for solving nonconvex quadratic programming problems. *Journal of Global Optimization*, 7(1):1–31, 1995.

[SW95]   R. J. Stern and H. Wolkowicz. Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. *SIAM Journal on Optimization*, 5(2):286–313, 1995.

[SZ03]   J. F. Sturm and S. Zhang. On cones of nonnegative quadratic functions. *Mathematics of Operations Research*, 28(2):246–267, 2003.

[THJ14]   L.-N. Tran, M. F. Hanif, and M. Juntti. A conic quadratic programming approach to physical layer multicasting for large-scale antenna arrays. *IEEE Signal Processing Letters*, 21(1):114–117, 2014.

[TM06]   D. A. Tolliver and G. L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1053–1060. IEEE, 2006.

[Tod01]   M. J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[TS05]   M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.

[Uhl79]   F. Uhlig. A recurring theorem about pairs of quadratic forms and extensions: A survey. *Linear Algebra and its Applications*, 25:219–237, 1979.

[VB96]   L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[VdP66]   C. Van de Panne. Programming with a quadratic constraint. *Management Science*, 12(11):798–815, 1966.

[vdWCV11] S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[VSMRR15] D. M. Van Sy Mai, B. Ramasubramanian, and M. C. Rotkowitz. Convex methods for rank-constrained optimization problems. In *Conference on Control and its Applications*. Society for Industrial and Applied Mathematics, 2015.

[WdM15]   I. Waldspurger, A. d'Aspremont, and S. Mallat. Phase recovery, MaxCut and complex semidefinite programming. *Mathematical Programming*, 149(1-2):47–81, 2015.

[WLMK09]   A. Waechter, C. Lairdl, F. Margot, and Y. Kawajir. Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT, 2009.

[Wri15]   S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[WSV00]   H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming*. Kluwer Academic Publishers, 2000.

[WSVDH13] P. Wang, C. Shen, and A. Van Den Hengel. A fast semidefinite approach to solving binary quadratic problems. In *Conference on Computer Vision and Pattern Recognition*, pages 1312–1319. IEEE, 2013.

[WYZ15]   Y. Wang, W. Yin, and J. Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *arXiv preprint:1511.06324*, 2015.

[XL15]   M. Xu and Z. Luo. An efficient algorithm for linear semi-infinite programming over positive polynomials. In *International Symposium on Operations Research and its Applications in Engineering, Technology and Management*, pages 1–4. IET, 2015.

[XYWZ12]   Y. Xu, W. Yin, Z. Wen, and Y. Zhang. An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China*, 7(2):365–384, 2012.

[YLS11]   Y.-H. Yang, S.-C. Lin, and H.-J. Su. Multiuser MIMO downlink beamforming design based on group maximum SINR filtering. *IEEE Transactions on Signal Processing*, 59(4):1746–1758, 2011.

[YR03]   A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[YS04]   S. X. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):173–183, 2004.

[Yua91]   Y.-X. Yuan. A dual algorithm for minimizing a quadratic function with two quadratic constraints. *Journal of Computational Mathematica*, 9(4):348–359, 1991.

[YZ03]   Y. Ye and S. Zhang. New results on quadratic minimization. *SIAM Journal on Optimization*, 14(1):245–267, 2003.

[ZSL11]   X. J. Zheng, X. L. Sun, and D. Li. Convex relaxations for nonconvex quadratically constrained quadratic programming: matrix cone decomposition and polyhedral approximation. *Mathematical Programming*, 129(2):301–329, 2011.