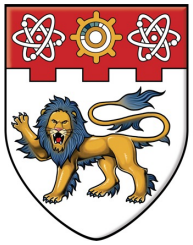


NANYANG
TECHNOLOGICAL
UNIVERSITY

ENHANCING CONVEXIFICATION TECHNIQUES FOR SOLVING NONCONVEX QUADRATIC PROGRAMMING PROBLEMS

QI XIAOFEI

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

2017

ENHANCING CONVEXIFICATION TECHNIQUES FOR SOLVING NONCONVEX QUADRATIC PROGRAMMING PROBLEMS

QI XIAOFEI

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University
in partical fulfillment of the requirement for the degree of
Doctoral of Philosophy

2017

Acknowledgements

I would first love to express my deepest appreciation to my supervisor, Professor Jitamitra Desai, who brought me into this interesting area of nonconvex optimization. Throughout my whole Ph.D. life, he gave me invaluable suggestions and provided gentle guidance and encouragement. Whenever losing the confidence on the research, I can pick up my faith quickly and walk further after talking with him. I cannot thank you enough for all your help and support on my way of research. I am also very grateful to my supervisor Professor Tai Kang and co-superivisor Professor Duan Fei. Deeply thankful for your help with my complement of Ph.D. Programme. Professor Duan, many thanks for your kindness and invaluable suggestions for my research, career and life.

I also wish to express my deepest thanks to my Thesis Advisory Committee members, Professor Nie Xiaofeng and Professor Chua Chek Beng.

Many thanks to my dearest friends in Singapore. I will always keep in my heart the wonderful time we spent here together. Thank all you guys from the bottom of my heart.

Last but not least, I owe a great debt of gratitude to my loving parents and sister, whose unconditional support and love have been giving me great strength and confidence. Your faith in me never been faltered.

Contents

Abstract

At the intersection of combinatorial and nonlinear optimization, *quadratic programming* **(QP)** plays an important role in practice and optimization theory. With wide-spread real-world applications including, but not limited to, engineering design, game theory, and economics, quadratic programming has many valuable contributions and has attracted considerable interest over the past few decades. As a more general formulation, *quadratically constrained quadratic programming* **(QCQP)** problems arise naturally from a lot of applications such as facility location, production planning, economic equilibria and Euclidean distance geometry, etc.

Most global optimization approaches for nonconvex QCQPs are based on constructing a *convex* or *linear* relaxation of the problem, and embedding it within an exhaustive search mechanism, where a lower bound is calculated as the value of the objective function of this relaxation (for a minimization problem) at each iteration of the search tree. The two principle elements affecting the overall effectiveness of such exhaustive search methods are the tightness of the underlying relaxation and the efficiency of calculating the lower bound. Besides, the performance of a branch-and-bound algorithm also relies on the procedure used to select a branching node and branching variable, which leads to good quality feasible solutions or equivalently an upper bound.

We begin by enhancing existing linearization techniques (notably RLT-based methods) for solving 0-1 QCQPs with the new class of valid cuts, which we refer to as *Minimum Triangle* **(MINT)** inequalities. These MINT cuts are derived based on the relationship that $y_{ij} = \min\{x_i, x_j\}$ at any feasible solution (and hence, at optimality), and we prove that these *super-linear* cuts are tighter than existing triangle inequalities, thereby leading to improved lower bounds at each node of the branch-and-bound tree. Furthermore, we extend the logic used in defining these minimum triangle inequalities to construct a novel variable fixing and affiliated branching strategy (in a branch-and-bound procedure) that identifies a feasible solution relatively early on in the search process while expending a very minimal computational cost.

The derived MINT inequalities in concert with the newly designed variable fixing and branching strategy gives rise to a specialized branch-and-bound algorithm that we refer to as the *Minimum Triangle Algorithm* **(MINT Algorithm)**, and we demonstrate the efficacy of this algorithm in finding a feasible solution (and subsequently, the optimal solution) for various classes of QP test instances obtained from the literature. Our computational results show that the number of nodes required to obtain the (global) optimum is greatly reduced, and overall, the MINT algorithm performs exceedingly well in comparison with existing methods.

Besides adding MINT inequalities to RLT relaxations, we also introduce a polynomial-time scheme to generate *higher rank-order semidefinite cutting planes* that serve to tighten convex relaxations of (discrete and continuous) nonconvex quadratically constrained quadratic

programs (QCQPs) and significantly improve lower bounds. Suitably defined row-and-column based operations are used to speed up the process of generating these cuts, and computational comparisons across different types of relaxations show the efficacy of these new cutting plane strategies.

Finally, noting that 0-1 QCQPs can be transformed into an equivalent 0-1 *mixed-integer program* (**MIP**), in our final contribution, we focus on constructing a *second-order cone programming* (**SOCP**) procedure to get tight lower bounds on the underlying 0-1 MIP. This SOCP relaxation method combines the RLT relaxation technique with the p-norm cone programming procedure from Burer and Chen [24], and in the illustrative example, we show that the lower bound reaches the optimal value, which in turn dramatically reduces the computational time taken to determine an optimal solution. In our computations, we test the level-1 SOCP relaxation on several 0-1 QP instances, and the results indicate that even the SOCP level-1 relaxation is a viable alternative to generating tight lower bounds for the difficult class of 0-1 QCQP problems.

List of Figures

List of Tables

Nomenclature

$[\cdot]_L$	the linearized version of $[\cdot]$
$[n]$	set $\{1, \cdots, n\}$
$\bullet$	Frobenius inner product
$\|J\|$	the cardinality of a finite set J
$\|x\|$	$\equiv \sqrt{x^\top x}$, the Euclidean norm of vector x
$\|x\|_p$	$\equiv \sqrt[p]{\sum_i \|x_i\|^p}$, the p-norm of vector x
$\mathcal{K}$	$\equiv \{1, 2, ..., K\}$, the set of quadratic constraints in QCQP
$\mathcal{L}$	$\equiv \{1, 2, ..., L\}$, the set of linear constraints in QCQP
$\mathbb{R}^n$	the n-dimensional real vector space
$\mathbb{R}^{n\times n}$	the space of $(n \times n)$ real matrices
$\mathbb{S}^n$	the set of all $n \times n$ symmetric real matrices
$\mathbb{S}^n_+$	the set of symmetric positive semidefinite matrices
$\mathbb{Z}_+$	nonnegative integer set
$\nu(\cdot)$	the optimal objective value of Problem $(\cdot)$
$\mathrm{Tr}(\cdot)$	trace of $(\cdot)$
$\varnothing$	empty set
e_n	n dimensional vector of ones

$conv(\cdot)$	convex hull of a set $(\cdot)$
BQP	boolean quadric polytope
LB	lower bound
MILP	mixed-integer linear program
MIQCQP	mixed-integer quadratically constrained quadratic program
QAP	quadratic assignment problem
QCQP	quadratically constrained quadratic program
QCR	quadratic convex reformulation
QP	quadratic program
RLT	reformulation-linearization technique
SDP	semidefinite program
SILP	semi-infinite linear program
SOCP	second order cone program
UB	upper bound

1 Introduction

Let $\mathbb{R}^{n\times n}$ represent the set of all $n\times n$ real matrices and let $\mathbb{S}^n$ denote the set of all $n\times n$ symmetric real matrices. The Frobenius inner product of X and Y, denoted $X \bullet Y$, in the linear space $\mathbb{R}^{n\times n}$ is $\text{Tr}(X^\top Y)$, *i.e.*, trace of $(X^\top Y)$. We write $X \succ 0$ if $X \in \mathbb{R}^{n\times n}$ is positive definite, *i.e.*, $\alpha^\top X\alpha > 0, \forall\ \alpha \in \mathbb{R}^n, \alpha \not\equiv 0$, and $X \succeq 0$ if X is positive semidefinite, *i.e.*, $\alpha^\top X\alpha \geq 0, \forall\ \alpha \in \mathbb{R}^n, \alpha \not\equiv 0$. Let $\mathbb{S}^n_+ = \{X \in \mathbb{S}^n : X \succeq 0\}$ denote the set of symmetric positive semidefinite matrices.

Now, consider the following *quadratically constrained quadratic program* **(QCQP)**:

$$\begin{aligned}
\text{QCQP:}\quad \min\quad & f_0(x) = c_0^\top x + x^\top Q_0 x && \text{(1.1a)}\\
s.t.\quad & f_k(x) = c_k^\top x + x^\top Q_k x + d_k \leq 0, \quad k \in \mathcal{K} && \text{(1.1b)}\\
& g_l(x) = a_l^\top x + b_l \leq 0, \quad l \in \mathcal{L}, && \text{(1.1c)}
\end{aligned}$$

where $x \in \mathbb{R}^n$, $Q_k \in \mathbb{S}^n$, $k \in \mathcal{K} \equiv \{1, ..., K\}$ indexes the set of quadratic constraints, and $l \in \mathcal{L} \equiv \{1, ..., L\}$ indexes the set of linear constraints. As introduced above, a quadratically constrained quadratic program (QCQP) is the problem of optimizing a quadratic objective function of some variables subject to linear and quadratic constraints on these variables.

Let $\mathcal{F}$ denote the feasible set of Problem (1.1), *i.e.*,

$$\mathcal{F} \equiv \{x \in \mathbb{R}^n : f_k(x) \leq 0,\ \forall\ k \in \mathcal{K}, g_l(x) \leq 0,\ \forall\ l \in \mathcal{L}\}.$$

If $Q_k \succeq 0$, for each $k \in \{0\} \cup \mathcal{K}$, then f_0 is a convex function, $\mathcal{F}$ is a convex set, and thus QCQP is a *convex* quadratic program. But in general QCQP does not necessarily possess the property of convexity by virtue of the fact that there exist some

indices $k \in \{0\} \cup \mathcal{K}$ such that $Q_k \notin \mathbb{S}^n_+$, thereby rendering QCQP as a *nonconvex* program. Unless explicitly specified, the remainder of this thesis will focus on the more general case of nonconvex QCQPs.

A further generalization to QCQP arises when we impose *integer* restrictions on a subset of the decision variables, which leads to the class of *mixed-integer quadratically constrained quadratic programming* **(MIQCQP)** problems. A special case of MIQCQP is when all the decision variables are *binary*, giving rise to the following pure 0-1 QCQP:

$$\begin{aligned}
\text{0-1 QCQP:} \quad \min \quad & f_0(x) = c_0^\top x + x^\top Q_0 x && \text{(1.2a)}\\
s.t. \quad & f_k(x) = c_k^\top x + x^\top Q_k x + d_k \leq 0, \quad k \in \mathcal{K} && \text{(1.2b)}\\
& g_l(x) = a_l^\top x + b_l \leq 0, \quad l \in \mathcal{L} && \text{(1.2c)}\\
& x \in \{0,1\}^n. && \text{(1.2d)}
\end{aligned}$$

In the absence of quadratic constraints of the type (1.1b), the problem of optimizing (minimizing or maximizing) a quadratic objective function subject to only *linear* restrictions is simply referred to as a *quadratic program* **(QP)**. At the intersection of combinatorial and nonlinear optimization, quadratic programming plays a major role in both practice and optimization theory. In the past few decades, with wide-spread real-world applications including, but not limited to, engineering design, machine scheduling [3], combinatorial optimization, game theory, and economics and capital budgeting [43], quadratic programming has attracted significant interest over the past few decades. As a more general formulation, quadratically constrained quadratic programming (QCQP) problems arise naturally from a lot of practical and theoretical applications such as economic equilibria, Euclidean distance geometry, multiperiod tankage quality problems in refinery processes, facility location and

production planning, max-cut problem and circle packing problems, amongst others. For a more exhaustive list of QCQP examples, we would like to refer the reader to Adjiman *et al.* [2].

Clearly, QCQP is a specially structured global optimization problem, and in addition to direct applications of QCQP, it has been demonstrated in the literature that mixed-integer 0-1 programs, fractional programs, bilinear programs, polynomial programs, chance-constrained programs, bilevel programs, generalized linear complementarity problems, and other classes of optimization problems can be reformulated as special cases of QCQP through simple transformations. This generality however comes at a price, with several theoretical and practical difficulties involved in the process of solving such problems. The complexity of nonconvex quadratically constrained quadratic programming exists at two levels:

(i) The problem of finding even a single *feasible* solution is NP-hard as it generalizes the linear complementarity problem. (Chung [27] analyzes the complexity of the latter problem.)

(ii) The nonlinear constraints define a feasible region, which is in general neither convex nor connected.

Moreover, even if the feasible region is a polyhedron (as in the case of QPs), optimizing the quadratic objective function is strongly NP-hard as the resulting problem subsumes the disjoint bilinear programming problem. (Hansen *et al.* [34] show that an equivalent problem, the linear maximin problem, is strongly NP-hard.) It follows that finding a finite and exact algorithm that solves large QCQPs is a computationally challenging task.

There are several powerful numerical methods that can (theoretically) solve convex QCQPs in polynomial time (see Alizadeh and Goldfarb [4], Buchheim *et al.*

[20], Goldfarb *et al.* [31], Jarre [36], Mehrotra and Sun [49]). However, unlike the convex case, the nonconvex quadratically constrained quadratic program is an NP-hard problem and there exist no (rigorous) necessary and sufficient global optimality conditions. In fact, Pardalos and Vavasis [53] show that the case of quadratic programming (QP) is also NP-hard even when the Q_0 matrix in the objective function has only *one* associated negative eigenvalue (for minimization problems).

Most global optimization methods for nonconvex QCQPs are based on constructing a *convex* or *linear relaxation* of the problem, and embedding it within an exhaustive search mechanism, in which a lower bound is computed as the value of the objective function of this relaxation at each iteration of the search tree. The tightness of the underlying relaxation and the efficiency in computing the lower bound are the major factors affecting the overall effectiveness of such exhaustive search methods. Hence, a significant amount of research has been dedicated towards the development of suitable relaxations, as required by exact solution algorithms, for solving nonconvex QCQPs.

Such methods invariably add appropriately defined (implied) constraints to QCQP in a higher dimensional space that tightens the underlying convex or linear programming relaxation. Generic approaches, alluded to above, include those based on the Reformulation-Linearization Technique (RLT) (Adams and Sherali [1], Audet *et al.* [9], Sherali and Tuncbilek [61, 62]) and efforts to integrate semidefinite programming (SDP) relaxations (or linear projections of SDP relaxations) into the underestimation scheme (Anstreicher [6], Bao *et al.* [14], Burer and Vandenbussche [21], Saxena *et al.* [56, 57]).

Rather than adding implied constraints to the relaxation of QCQP, an alternative set of techniques prescribes adding cuts to strengthen the relaxation based on eigenvector projections (Cambini and Sodini [26], Pardalos [52], Rosen and Pardalos

[55], Saxena *et al.* [57]); or polyhedral facets (Anstreicher and Burer [7], Bao *et al.* [13], Burer and Letchford [22]); or the necessary KKT optimality conditions (Vandenbussche and Nemhauser [68, 69]).

1.1 RLT relaxations

As aforementioned, a frequently used and powerful idea for dealing with hard optimization problems is to develop tractable convex relaxations and subsequently embed these relaxations within a branch-and-bound framework, thereby leading to the global optimum. One of the most noteworthy amongst these methods is the *Reformulation-Linearization Technique* **(RLT)**. Broadly stated, the RLT reduces a given nonconvex program to a corresponding equivalent or sometimes lower bounding (for minimization problems) linear/convex program, and solves this resultant problem.

In their pioneering work, Adams and Sherali [1] present an RLT strategy to generate tight linear programming relaxations for 0-1 QCQPs and develop a decomposition-based approach for solving this problem to global optimality. This RLT methodology transforms the 0-1 QCQP problem (as defined in (1.2)) to an equivalent 0-1 *mixed-integer linear program* **(MILP)**. To illustrate this methodology, consider the following (explicitly written form of the) pure 0-1 QP, where $q_{ij} \equiv [Q_0]_{ij}, \forall\, (i,j)$. (An extension of this RLT procedure to generic QCQPs is fairly straightforward.)

$$\text{0-1 QP:} \quad \min \quad \sum_{i=1}^{n} c_i x_i + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j \tag{1.3a}$$

$$s.t. \quad \sum_{i=1}^{n} a_{li} x_i + b_l \leq 0, \quad \forall\, l \in \mathcal{L} \tag{1.3b}$$

$$x \in \{0,1\}^n. \tag{1.3c}$$

Substituting

$$y_{ij} = x_i x_j, \ \forall \ (i,j), \ i < j, \tag{1.4}$$

and recognizing that $x_i^2 \equiv x_i, \forall \ i = 1, ..., n$, we obtain the following equivalent 0-1 MILP formulation:

$$\text{0-1 QP}_{\text{RLT}}: \quad \min \quad \sum_{i=1}^{n} c_i x_i + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} y_{ij} \tag{1.5a}$$

$$s.t. \quad \sum_{i=1}^{n} a_{li} x_i + b_l \leq 0, \quad \forall \ l \in \mathcal{L} \tag{1.5b}$$

$$x_i \geq y_{ij}, \quad \forall \ (i,j), \ i < j \tag{1.5c}$$

$$x_j \geq y_{ij}, \quad \forall \ (i,j), \ i < j \tag{1.5d}$$

$$1 + y_{ij} - x_i - x_j \geq 0, \quad \forall \ (i,j), \ i < j \tag{1.5e}$$

$$y_{ij} \geq 0, \quad \forall \ (i,j), \ i < j \tag{1.5f}$$

$$x \in \{0,1\}^n, \tag{1.5g}$$

where the constraints (1.5c)-(1.5f) enforce $y_{ij} = 1$, if $x_i = x_j = 1, \ \forall \ (i,j)$, and $y_{ij} = 0$, if either $x_i = 0$ or $x_j = 0, \ \forall \ (i,j)$.

Henceforth, for the remainder of this thesis, we will refer to constraints (1.5c)-(1.5f) as "*RLT constraints*", and the linear programming relaxation to 0-1 QP_{RLT} (obtained by setting the x-variables to be continuous, *i.e.*, $0 \leq x \leq e_n$, in constraint (1.5g)) as the "*RLT relaxation*". This LP relaxation can then be solved by utilizing a large-scale commercial LP solver (for example, Cplex [28] or Gurobi [33]) to obtain a valid *lower bound* to the original 0-1 QP. However, the lower bounds obtained from this relaxation are sometimes found to be weak in practice and can be improved by adding suitably defined valid inequalities.

Remark 1. In essence, the constraints (1.5c)-(1.5f) form the *convex and concave outer-envelopes* to the function $y_{ij} = x_i x_j$, over the domain $0 \leq x \leq e_n$, and in

the literature they are also referred to as *McCormick envelopes*. A further tightening of the RLT relaxation to 0-1 QP_{RLT} can be obtained by multiplying constraint (1.5b) by x_j and $(1 - x_j), \forall\, j = 1, ..., n$. This yields the constraints:

$$\sum_{i=1}^{n} a_{li}x_i + b_l \leq \sum_{i<j} a_{li}y_{ij} + \sum_{i>j} a_{li}y_{ji} + (a_{lj} + b_l)x_j \leq 0 \quad \forall\ (l, j), \tag{1.6}$$

and Adams and Sherali [1] proved that (1.6) implies (1.5b), leading to a tighter relaxation.

Theorem 1. *ν(0-1 QP) $\equiv$ ν(0-1 QP_{RLT}).*

Proof. Obvious from construction.

□

Furthermore, recognizing that

$$x^\top Q x = 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij}x_i x_j \equiv Q \bullet [xx^\top] = Q \bullet X, \tag{1.7}$$

where $X = xx^\top$ and the $[\bullet]$ operator represents the Frobenius inner product, we can represent 0-1 QP_{RLT} as:

$$\begin{aligned}
\text{0-1 QP}_{\text{RLT}}: \quad \min \quad & c_0^\top x + Q_0 \bullet X_L && \text{(1.8a)}\\
& a_l^\top x + b_l \leq 0, \quad \forall\, l \in \mathcal{L} && \text{(1.8b)}\\
& \text{RLT constraints,} && \text{(1.8c)}\\
& x \in \{0, 1\}^n, && \text{(1.8d)}
\end{aligned}$$

where,

$$X_L = \begin{pmatrix} x_1 & y_{12} & \cdots & y_{1n} \\ y_{12} & x_2 & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} & y_{2n} & \cdots & x_n \end{pmatrix}. \tag{1.9}$$

Here, $X_L = [xx^\top]_L$, where $[\cdot]_L$ denotes the linearization of $[\cdot]$ under the substitution (1.4), and furthermore, as $X \equiv X_L$ at optimality, we have explicitly introduced X_L into the above formulation.

1.2 SDP and SOCP relaxations

SDP relaxations

It is well-known that *Semidefinite Programming* **(SDP)** relaxations have been an attractive approach for finding good bounds and approximate solutions for a variety of NP-hard optimization problems due to the development of efficient primal-dual interior-point methods for solving SDP problems. SDP relaxations belong to the class of convex relaxations, and often yield the tightest lower bounds (achievable in polynomial time), and are computationally attractive particularly for small-scale problem instances.

An SDP representation of Problem (1.1) is given by:

$$\begin{aligned}
\min \quad & f_0(x) = c_0^\top x + Q_0 \bullet X && \text{(1.10a)} \\
s.t. \quad & f_k(x) = c_k^\top x + Q_k \bullet X + d_k \leq 0, \quad k \in \mathcal{K} && \text{(1.10b)} \\
& g_l(x) = a_l^\top x + b_l \leq 0, \quad l \in \mathcal{L} && \text{(1.10c)} \\
& X \succeq 0, && \text{(1.10d)} \\
& rank(X) = 1, && \text{(1.10e)}
\end{aligned}$$

where, following (1.7), we have introduced a new variable $X = xx^\top$(similar to the case of the previously encountered RLT relaxation); replaced the terms $x^\top Q_k x$ with $Q_k \bullet X$; and recognizing the fact that $X = xx^\top$ indicates that X is a *rank-one* positive semidefinite matrix, we have explicitly introduced this nonconvex rank constraint into the model. Dropping the nonconvex rank constraint (1.10e) from the formulation leads to an *SDP relaxation* to QCQP.

A further strengthening of (1.10) can be obtained by replacing the constraint $X \succeq 0$ with $X \succeq xx^\top$, which can be expressed via the augmented matrix $\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0$. This yields the following SDP relaxation:

$$\text{QCQP}_{\text{SDP}}: \quad \min \quad \begin{pmatrix} 0 & \frac{1}{2}c_0^\top \\ \frac{1}{2}c_0 & Q_0 \end{pmatrix} \bullet \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \tag{1.11a}$$

$$s.t. \quad \begin{pmatrix} d_k & \frac{1}{2}c_k^\top \\ \frac{1}{2}c_k & Q_k \end{pmatrix} \bullet \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \leq 0, \quad k \in \mathcal{K} \tag{1.11b}$$

$$\begin{pmatrix} b_l & \frac{1}{2}a_l^\top \\ \frac{1}{2}a_l & O \end{pmatrix} \bullet \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \leq 0, \quad l \in \mathcal{L} \tag{1.11c}$$

$$\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0, \tag{1.11d}$$

where the terms in the objective function and constraints have been intelligently reorganized to reflect the new lifted constraint (1.11d).

SOCP relaxations

As aforementioned, SDP relaxations provide tight lower bounds but at the expense of significant computational cost. Kim and Kojima [37] were the first to point out

that *second order cone programming* **(SOCP)** provides a viable alternative to relaxing nonconvex quadratic programs, while achieving a reasonable compromise between the effectiveness of SDP relaxations and the low computational cost of lift-and-project based ideas, such as an RLT-based LP relaxation.

A *convex* QCQP of the form (1.1) can be expressed as an SOCP with the use of auxiliary variables $u_k \in \mathbb{R}^n, k = 0, ..., K$, as:

$$\begin{aligned} \text{QCQP}_{\text{SOCP}}: \quad \min \quad & c_0^\top x + t && \text{(1.12a)} \\ & u_0^\top u_0 \leq t && \text{(1.12b)} \\ & u_0 = Q_0^{\frac{1}{2}} x && \text{(1.12c)} \\ & u_k^\top u_k \leq -c_k^\top x - d_k, \quad \forall\, k \in \mathcal{K} && \text{(1.12d)} \\ & u_k = Q_k^{\frac{1}{2}} x, \quad \forall\, k \in \mathcal{K}, && \text{(1.12e)} \end{aligned}$$

where, for brevity, we have ignored the linear constraints (1.1c) in constructing the SOCP relaxation.

An SOCP relaxation to the 0-1 QCQP strengthens the lift-and-project linear programming relaxation method by adding valid *convex quadratic* inequalities in lieu of the positive semidefinite cone involved in the SDP relaxation. The positive semidefinite condition (1.11d) holds true if and only if $C \bullet (X - xx^\top) \geq 0, \forall\, C \in \mathbb{S}^n_+$, which yields the following valid inequality:

$$-C \bullet X + x^\top C x \leq 0. \tag{1.13}$$

Note that the left hand side of the inequality with a fixed $C \in \mathbb{S}^n_+$ is linear in X and convex quadratic in x. This allows us to add inequality (1.13) to the lift-and-project LP relaxation of QP. Noting that there are infinitely many SOCP constraints of the

type (1.13), Kim and Kojima [37] proposed an efficient methodology to select a reduced number of constraints and derive an effective SOCP relaxation to the original nonconvex QP.

Burer and Chen [24] introduced a lift-and-project procedure for 0-1 linear programming problems based on a hierarchy of p-order cone programming problems. This p-order cone procedure also generalizes and unifies several existing lift-and-project mechanisms.

1.3 Extensions and combinations of relaxations

RLT enhanced with Triangle Inequalities

In his seminal work, Padberg [51] defined the *Boolean Quadric Polytope* **(BQP)**, denoted by BQP^n, as:

$$BQP^n \equiv conv\{(x,y) \in \{0,1\}^{n(n+1)/2} : \ (x,y) \text{ satisfies } (1.5c) - (1.5f)\} \tag{1.14}$$

and the *linear relaxation* of BQP^n, denoted by BQP^n_{LP}, as:

$$BQP^n_{\text{LP}} \equiv \{(x,y) \in \mathbb{R}^{n(n+1)/2} : \ (x,y) \text{ satisfies } (1.5c) - (1.5f)\}. \tag{1.15}$$

BQP^n and BQP^n_{LP} are bounded polyhedra, *i.e.*, they are polytopes by definition, and clearly, BQP^n has exactly 2^n vertices while BQP^n_{LP} has in general many more vertices. For $n = 2$, we have $BQP^2 \equiv BQP^2_{\text{LP}}$. When $n = 3$, letting i, j, k index the three coordinate dimensions, the polytope BQP^3 is defined by the RLT constraints and following inequalities:

$$x_i + x_j + x_k \leq y_{ij} + y_{jk} + y_{ik} + 1 \tag{1.16a}$$

$$y_{ij} - y_{jk} + y_{ik} \leq x_i \tag{1.16b}$$

$$y_{ij} + y_{jk} - y_{ik} \leq x_j \tag{1.16c}$$

$$y_{ik} + y_{jk} - y_{ij} \leq x_k. \tag{1.16d}$$

We refer to the system (1.16) as the *triangle inequalities*, and all of them define *facets* of BQP^3. For $n \geq 4$ the structure of BQP^n is far more complicated, and an explicit representation is not readily available. However, Padberg [51] has defined certain types of facet-inducing inequalities for the general BQP^n case as well.

Grötschel and Wakabayashi [32] used triangular inequalities within a branch-and-cut algorithm in a graph theoretic context and showed that an adaptation of the triangle inequalities to general cliques of odd order leads to the well-known *clique inequalities*. Many other inequalities for the BQP^n have been discovered thereafter, and for more detailed information, we refer the reader to Deza and Laurent [29].

In the context of our research, we explore the use of adding triangle inequalities of the type (1.16) to 0-1 QP_{RLT} to get an enhanced formulation, and furthermore, we extend the theory of triangle inequalities by using *super-linear* functions as seen subsequently in the thesis.

RLT+SDP

Many researchers have proposed using a combined RLT+SDP relaxation method, which has proven to be more effective as compared to using either the RLT or SDP relaxations as a standalone approach (see Anstreicher [6, 8], Sherali [65]). After merging the RLT and SDP relaxations given by (1.5) and (1.11) respectively, the enhanced QCQP formulation is given by:

$$\text{QCQP}_{\text{RLT+SDP}}: \quad \min \quad f_0(x) = c_0^\top x + Q_0 \bullet X_L \tag{1.17a}$$

$$s.t. \quad f_k(x) = c_k^\top x + Q_k \bullet X_L + d_k \leq 0,\ k \in \mathcal{K} \tag{1.17b}$$

$$g_l(x) = a_l^\top x + b_l \leq 0,\ l \in \mathcal{L} \tag{1.17c}$$

$$\text{RLT constraints,} \tag{1.17d}$$

$$\begin{pmatrix} 1 & x^\top \\ x & X_L \end{pmatrix} \succeq 0, \tag{1.17e}$$

where, X_L is as defined in (1.9), and in this case, noting that $X_L \equiv X$ at optimality, we have explicitly introduced this linearized augmented matrix into the PSD constraint (1.17e).

Anstreicher [6] also proposed an RLT+SDP relaxation for the nonconvex QP with box constraints, in which the RLT relaxation is further strengthened by adding the SDP constraint. While such RLT+SDP relaxations tend to provide very tight lower bounds, the disadvantage is the prohibitively expensive computational cost required for solving such problems. To circumvent this computational difficulty, Sherali and Fraticelli [63] presented a novel technique to generate additional inequalities (while preserving the dimensionality of the problem), which further enhances the relaxation generated by the RLT constraints, by employing concepts obtained from semidefinite programming. These additional constraints are called *linear semidefinite cuts* (or *SDP cuts*).

In lieu of directly solving the semidefinite program $\text{QCQP}_{\text{RLT+SDP}}$, in our research, we also explore the use of generating *higher rank-ordered cutting planes* based on SDP constructs to enhance the lift-and-project relaxation, particularly for large-scale instances.

1.4 Research scope and objectives

The primary goal of our research is to improve the solvability of QCQPs (as defined in (1.1)) by developing theoretical, algorithmic, and computational advancements, that enhance and complement existing relaxation techniques and algorithms.

As aforementioned, most global optimization methods for nonconvex QCQPs are based on constructing a convex or linear relaxation of the problem, and embedding it within an exhaustive search mechanism, in which a *lower bound* **(LB)** is computed as the value of the objective function of this relaxation at each iteration of the search tree. The tightness of the underlying relaxation and the efficiency in computing the lower bound are the major factors affecting the overall effectiveness of such exhaustive search methods. The effectiveness of a branch-and-bound algorithm depends fundamentally on the procedure used to obtain lower bounds and the method used to select a branching node and branching variable, which leads to a good quality feasible solution or equivalently an *upper bound* **(UB)**. Moreover, as noted earlier, the problem of finding even a single feasible solution is also NP-hard.

To improve the quality of the obtained lower bounds, there are two research directions that become evident. The first is to add appropriately defined valid inequalities, such as the triangle inequalities (1.16), or other types of feasibility and optimality conditions that are based on the underlying properties of the convex hull of feasible 0-1 integer points. The other direction is to design cut generation schemes based on extended concepts such as semidefinite programming, akin to the one presented by Sherali and Fraticelli [63], which utilize the information gleaned from the current (infeasible) lower bounding solution to generate cutting planes and embed this mechanism into an iterative scheme for improving lower bounds.

While both these approaches have been successful with varying degrees in

practice, they serve only to improve the lower bound, and do not address the issue of obtaining feasible upper bounding solutions relatively quickly in the search process. In our research, recognizing that a tightening of the LB-UB gap rapidly improves the convergence of the search tree, we design mechanisms that improve both lower and upper bounds simultaneously, thereby improving the efficiency of branch-and-bound algorithms significantly. In light of these motivating aspects, our research objectives can be delineated as follows.

Research Objective 1: Minimum Triangle Inequalities and Algorithms

For 0-1 QCQPs, branch-and-bound algorithms are typically used to find exact solutions. The effectiveness of a branch-and-bound algorithm depends fundamentally on the procedure used to obtain lower bounds and the method used to select a branching node (Anstreicher *et al.* [5] and Brixius [19]). Moreover, as noted earlier, even the problem of detecting a feasible solution of nonconvex QP is NP-hard (Chung [27]).

To address the first limitation, in the spirit of the triangle inequalities (1.16), we begin by enhancing existing linearization techniques (notably RLT-based methods) for solving 0-1 QCQPs with the new class of valid cuts, which we refer to as *Minimum Triangle* **(MINT)** Inequalities. The MINT cuts are derived based on the relationship that $y_{ij} = \min\{x_i, x_j\}$ at any feasible solution (and hence, at optimality), and we prove that these *super-linear* cuts are tighter than the existing triangle inequalities, thereby leading to improved lower bounds at each node of the branch-and-bound search tree.

Furthermore, we extend the logic used in defining these minimum triangle inequalities to construct a novel variable fixing and affiliated branching strategy (within a branch-and-bound procedure) that identifies a feasible solution relatively early on in the search process while expending a very minimal computational cost. As existing large-scale commercial LP solvers, such as Gurobi [33], can determine the optimal

solution to an MILP very efficiently if a feasible solution is provided as an input, this branching strategy significantly enhances the solvability of large-scale 0-1 QCQPs.

The derived MINT cuts in concert with the newly designed variable fixing and branching strategy gives rise to a branch-and-bound algorithm that we refer to as the *Minimum Triangle Algorithm* **(MINT Algorithm)**, and we demonstrate the efficacy of this algorithm in finding a feasible solution (and subsequently the optimal solution) for standard QP test instances obtained from the literature. Our computational results show that the number of nodes required in determining the optimal solution as well as the computation time to obtain the (global) optimum is greatly reduced, and overall, the MINT algorithm performs exceedingly well in comparison with existing methods.

Research Objective 2: Higher-rank Ordered Semidefinite Cutting Planes

For smaller 0-1 QP problems ($n \leq 50$), cutting plane approaches (such as MINT cuts, SDP cuts, etc.) applied at the root node of the branching tree are sufficient to solve the problem. For larger instances, cutting plane strategies are used in concert with branch-and-bound approaches, leading to branch-and-cut methodologies. Even for cases where the number of nodes in the branching tree is rather small, the computational time expended each node can be quite large.

In the second thrust of our research, we will derive valid cutting planes based on exploiting the SDP constraint (1.10d), or equivalently (1.11d) for solving (continuous or binary) QCQP problems. Note that such an approach has already been advocated by Sherali and Fraticelli [63], but in their case, the cut generation strategy outputs an α-vector (of the correct dimension) such that the semidefinite cut $\alpha^\top X \alpha \geq 0$ eliminates the current (infeasible) solution of an underlying relaxation, if this solution violates the PSD constraint. Clearly, this cut is a *rank-one cut*.

In our research, we extend the SDP cutting plane generation routine and we improve the Sherali and Fraticelli [63] approach in two ways: the first, by directly

generating a symmetric positive semidefinite matrix (instead of a vector), say H, at any violated solution such that the semidefinite cut $H \bullet X \geq 0$ eliminates the current infeasible solution; and second, by not restricting the matrix H to be of rank-one, we generate cuts of higher-rank orders. Moreover, we still retain the worst case complexity of $O(n^3)$, and hence the cutting plane strategy employed in our research also belongs to the class of inductive polynomial time algorithms.

Research Objective 3: Second Order Cone Programming (SOCP)-based hierarchy for 0-1 QCQPs

As aforementioned, QCQPs can be transformed into either equivalent 0-1 MIPs or SDPs, and such relaxations have been hierarchically used to converge to the convex hull of the original QCQP (see Lasserre [41], Sherali and Adams [60]). However, to the best of our knowledge, a convergent second order cone programming hierarchy to solve 0-1 QCQPs has not yet been studied. In our last contribution, we begin by constructing a valid SOCP relaxation based on the p-cone programming relaxation [24], and demonstrate that this gives a much better lower bound as compared to other methods. Then, we lay the framework for extending this relaxation to obtain a hierarchy of SOCP relaxations leading to the convex hull of the feasible region.

1.5 Organization of the thesis

The remainder of the thesis is organized as follows. In Chapter 2, we review prior studies related to linear and convex relaxations of QCQP, with the main focus being on RLT and SDP relaxations. In Chapter 3, we derive a new set of valid super-linear triangle inequalities for 0-1 QCQPs, and in addition, we prescribe a new variable fixing and branching strategy, geared towards finding feasible solutions very efficiently. Detailed computational results that showcase the efficacy of the derived cutting planes

and branching strategy, when employed within a branch-and-cut framework, are presented, along with comparisons to other methods. In Chapter 4, we propose a new mechanism to generate a class of polynomial-time higher rank-order semidefinite cutting planes for nonconvex QCQPs that serve to tighten the underlying convex relaxations and improve the lower bound. In Chapter 5, we propose an SOCP relaxation technique to achieve a tight lower bound for 0-1 QCQPs, and finally, conclusions and our future research directions are stated in Chapter 6.

2 Literature Review

In the context of exact algorithms for solving nonconvex optimization problems, most of the attention in the optimization literature has been focused on pure and mixed-integer 0-1 programming problems because of the wide variety of real-world applications that can be modeled using these techniques. Despite limiting the scope to describing exact algorithms for solving nonconvex quadratic and polynomial programs, it is still a daunting task to provide a comprehensive literature review on this topic. Hence, this chapter only reviews specific methodologies for solving (nonconvex) QCQP problems that are relevant to this research effort, and lays the foundation for developing effective solution techniques based on the RLT methodology.

2.1 The basic RLT procedure

The importance of having tight linear programming representations to enhance the effectiveness of an algorithm for solving integer programming problems has been extensively dealt with in the optimization literature. While several strategies exist for constructing good underlying representations of the problem, the RLT in particular has demonstrated the capability of generating very tight relaxations. A recent overview of RLT can be found in Sherali [65]. Sherali and Alameddine [59] use the Reformulation-Linearization Technique to solve bilinear programming problems, and Audet *et al.* [9] extend the use of RLT in solving QCQP by including different classes of linearizations. A brief survey of the RLT methodology and related applications can also be found in Sherali and Desai [64].

The basic RLT construct operates in two phases. In the *Reformulation Phase*, certain additional types of implied (polynomial) constraints are appended to the

problem. In the subsequent *Linearization/Convexification Phase*, the resulting program is linearized by performing auxiliary variable substitutions that involve defining new variables to replace each variable-product term. The obtained higher dimensional representation yields a linear/convex programming relaxation. To illustrate, consider the feasible region F of a 0-1 linear programming problem with n variables as defined below:

$$F = \{x \in \mathbb{R}^n | \sum_{j=1}^{n} a_{lj}x_j \leq b_l, l = 1, ..., L, \ x \text{ binary}\}. \tag{2.1}$$

Sherali and Adams [58] determine a hierarchy of relaxations of F between the continuous and convex hull representations for 0-1 programming problems. In order to construct the linear programming bounding problem using RLT, generate implied constraints using distinct products of the *bound factors*, namely $(x_j - 0) \geq 0$ and $(1 - x_j) \geq 0$. A series of relaxations are defined by using various d-degree polynomial factors to operate on the constraints of F. For $d = 0$, assume that these factors are *unity* so that we obtain the linear programming relaxation of F. For $d = 1, ..., n$, define the polynomial factor $F_d(J_1, J_2)$ as:

$$F_d(J_1, J_2) = \prod_{j \in J_1} (x_j - 0) \prod_{j \in J_2} (1 - x_j), \tag{2.2}$$

where,

$$J_1, J_2 \subseteq \{1, ..., n\}, J_1 \cap J_2 = \varnothing, |J_1 \cup J_2| = d. \tag{2.3}$$

Clearly, for any $d = 1, ..., n$, and any J_1, J_2 satisfying the above restrictions, $F_d(J_1, J_2) \geq 0$, for all binary x.

As an abbreviating terminology, we will refer to any (J_1, J_2) satisfying the conditions in (2.3) as being of *order* d. Note that for each $d = 1, ..., n$, there are $\begin{pmatrix} n \\ d \end{pmatrix} 2^d$ such factors, and these factors are comprised using a total of $\sum_{i=1}^{d} \begin{pmatrix} n \\ i \end{pmatrix}$

distinct variable terms. Now, for any $d \in \{1, ..., n\}$, construct relaxation F_d of F via the following two steps.

Step 1: Multiply each of the L linear constraints in F_0 by each of the (non-negative) factors defined in (2.2), using the relationship $x_j^2 = x_j$ for $j = 1, ..., n$.

Step 2: Linearize the resulting polynomial constraint expressions by substituting w_J for the nonlinear term $\prod_{j \in J} x_j$ for each set J with $|J| \geq 2$.

Let F_0 denote the linear programming relaxation of F, and $conv(F)$ represent the convex hull of F. Define $F_{Pd} = \{x : \ (x, w) \in F_d\}$ to be the projection of F_d onto the space of the x-variables for $d = 1, ..., n$. Then we have,

$$F_0 \supseteq F_{P1} \supseteq \cdots \supseteq F_{Pn} \equiv conv(F), \tag{2.4}$$

and it is worth noting that the final relaxation F_{Pn} is indeed the convex hull of F.

As evident from the foregoing discussion, higher order polynomial constraints can be used to generate a hierarchy of relaxations beyond the first level ($d = 1$) relaxation. However, with the increasing of the level-d, the number of variables increases exponentially. From a computational viewpoint, it has been found that the first order relaxation works sufficiently well in practice. The RLT methodology provides us with a mechanism to convert quadratic or polynomial programming problems (which are nonlinear in nature) to an equivalent linear form, albeit in a higher dimensional space. Linear formulations can be solved far more efficiently as compared to their nonlinear counterparts. However, this comes at price of higher dimensionality and possibly weak lower bounds. As an extension, incorporating other constraints, such as triangle inequalities or SDP cuts, into the RLT framework can further tighten this first order relaxation. Linderoth [47] extended the results in Sherali and Alameddine [59] to give second-order cone constraints that are valid on certain

types of triangular regions, and showed that these inequalities give the best convex or concave envelopes of $x_i x_j$ in certain cases; he then used these inequalities in a branch-and-bound approach. Mitchell *et al.* [50] proposed to tighten RLT-based QCQP relaxations by using second-order cone constraints that are constructed based on linear combinations of pairs of variables resulting in a convex quadratic relaxation. In the presence of good (tight) bounds on these linear combinations, the resulting constraints were shown to strengthen the McCormick (or equivalently RLT) relaxation bounds.

In similar vein, global optimization techniques for general nonconvex QCQPs follow a recursive partitioning of the convex feasible set and thus could generate an infinite number of branch-and-bound nodes. Burer and Vandenbussche [21] proposed a finite branch-and-bound algorithm for nonconvex quadratic programming via SDP relaxations. Some other branch-and-bound methods are presented by Audet *et al.* [9], Linderoth [47], Vandenbussche and Nemhauser [68], and Cambini and Sodini [26].

Burer [23] modeled nonconvex quadratic programming with a mix of binary and continuous variables as a linear program over the dual of the cone of copositive matrices. The author showed in a seminal paper that the copositive programming approach is in fact a reformulation rather than a relaxation, of any mixed 0-1 QP, under mild conditions. This also brodens the application of copositive and completely positive programming. Triggered by Burer [23]'s seminal work, many copositive reformulations of mixed-binary QPs have been discussed by now. Bomze *et al.* [18] related the reformulations of Burer [23] with more recent reformulations, and complemented these with some new reformulations.

In Li and Zhang [45], an algorithm for solving large-size quadratic programs is proposed, wherein the authors decompose a large-scale quadratic program into a series of smaller-sized problems and then approximate the solution of the original quadratic program via the solutions of these decomposed subproblems. They prove

that the accumulation point of the iterates generated by the algorithm converges to a global minimum of the quadratic program.

To reduce computational difficulties in solving large-scale integer quadratic programming problems, another approach is to reduce the number of integer variables by using certain feasibility and optimality conditions (Babayev and Mardanov [11], Zhou and Chen [73]). Efforts towards reformulating MIQCQP have also taken the form of reducing the number of nonconvex bilinear terms (Audet *et al.* [10], Ben-Tal *et al.* [15], Liberti and Pantelides [46]). Anstreicher *et al.* [5] and Helmberg and Rendl [35] presented methods to solve large-scale *Quadratic Assignment Problems* **(QAPs)**, which is one of the most difficult of combinatorial optimization problems.

2.2 Semidefinite representations and cutting planes

A *semidefinite program* **(SDP)** can be stated as follows:

$$\min\{C \bullet X : \ A_i \bullet X = b_i, \ \forall\, i = 1, ..., m, \ \text{and } X \succeq 0\}, \tag{2.5}$$

where $C, A_1, A_2, ..., A_m$, and $X \in \mathbb{S}^n$, $b_i \in \mathbb{R}$, $\forall\ i = 1, ..., m$. Given the structure of Problem (2.5), it is evident that semidefinite programming can be regarded as an extension to linear programming where the non-negativities on the variables are replaced by positive semidefinite restrictions on matrices. This is equivalently a replacement of the first orthant (or non-negativity constraints) by a cone of PSD matrices. Alternatively, we can also view problem (2.5) as a *semi-infinite linear program* **(SILP)** since the matrix inequality $X \succeq 0$ is equivalent to a set of infinite linear constraints of the type $\alpha^\top X \alpha \geq 0, \forall\ \alpha \in \mathbb{R}^n$. Usually, many convex optimization problems such as linear programs or (convex) quadratic programs can be recast as semidefinite programs.

To showcase a simple illustration of reformulating an optimization problem as an SDP, consider a (convex) quadratically constrained quadratic program (QCQP) of the type (1.1), where, for notational convenience, the convex quadratic functions appearing in the problem are expressed as $f_k(x) = c_k^\top x + (Q_k^{\frac{1}{2}} x)^\top (Q_k^{\frac{1}{2}} x) + d_k$. It follows that this convex QCQP can be rewritten as below:

$$\min \quad t \tag{2.6a}$$

$$s.t. \quad \begin{pmatrix} I & Q_0^{\frac{1}{2}} x \\ (Q_0^{\frac{1}{2}} x)^\top & -c_0^\top x - d_0 + t \end{pmatrix} \succeq 0 \tag{2.6b}$$

$$\begin{pmatrix} I & Q_k^{\frac{1}{2}} x \\ (Q_k^{\frac{1}{2}} x)^\top & -c_k^\top x - d_k \end{pmatrix} \succeq 0, \ k = 1, ..., K, \tag{2.6c}$$

which is a semidefinite program of appropriate dimension.

Recently, Billionnet *et al.* [16, 17] proposed a method called the *Quadratic Convex Reformulation* (QCR), wherein they use SDPs to exactly reformulate 0-1 QPs, and not to merely obtain a relaxation. The idea is to perturb the objective function in such a way that it is rendered convex, while simultaneously ensuring that the continuous relaxation of the 0-1 QP is tight. Galli and Letchford [30] explored the possibility of extending the reformulation approach to the more general case of mixed-integer QCQPs based on the QCR technique. Their research indicated that the QCR method could be adapted easily to deal with 0-1 QCQP instances in which all quadratic constraints are equations, but there exist some difficulties when quadratic inequalities are present in the problem formulation.

Zheng *et al.* [72] proposed a general D.C. (difference of convex functions) decomposition scheme for constructing SDP relaxations for a class of nonconvex quadratic programs with a nonconvex quadratic objective function and convex

quadratic constraints. As a further research, Zheng *et al.* [71] developed a decomposition-approximation approach for constructing convex relaxations for general nonconvex QCQP problems which might include nonconvex quadratic constraints. The key idea is inspired by exploring the "best" decomposition of a nonconvex objective over certain specifically selected cones and then constructing a corresponding polyhedral underestimation of the nonconvex quadratic term.

When QCQP (1.1) is a convex program, *i.e.*, $Q_k \succeq 0, \forall\ k \in \{0\} \cup \mathcal{K}$, Problem (2.6) or Problem (1.11) provide an exact representation. Bao *et al.* [14] compare the tightness and computational tractability of a collection of SDP-based relaxations for QCQP problems, and investigate the theoretical strength and computational efficiency of these relaxations. Moreover, if QCQP (1.1) is a nonconvex program, the corresponding SDP maybe unbounded, even if all of the original variables have finite lower and upper bounds. Anstreicher [6] suggested a remedy by adding upper bounds to the diagonal components of X. Note that for the case of the 0-1 QCQP, as $x_i^2 = x_i \leq 1$, this condition is automatically satisfied.

While it is interesting to note that QCQPs can be cast as semidefinite programming problems, it may not be a good idea from an algorithmic point of view. Hence Sherali and Fraticelli [63] proposed a mechanism to tighten RLT-based relaxations for solving nonconvex programming problems by importing concepts from semidefinite programming, leading to a new class of *semidefinite cuts*. Instead of relying on specific SDP solvers, the positive semidefinite stipulation is first re-written to develop an SILP representation of the problem. An equivalent SILP restatement of Problem (1.17) is constructed as follows:

$$\text{QCQP}_{\text{RLT+SILP}}: \quad \min \quad f_0(x) = c_0^\top x + Q_0 \bullet X_L \tag{2.7a}$$

$$s.t. \quad f_k(x) = c_k^\top x + Q_k \bullet X_L + d_k \leq 0, \quad i \in \mathcal{K} \tag{2.7b}$$

$$g_l(x) = a_l^\top x + b_l \leq 0, \quad l \in \mathcal{L} \tag{2.7c}$$

RLT constraints, (2.7d)

$$\alpha^\top X_L \alpha \geq 0, \forall\, \alpha \in \mathbb{R} \text{ and } \|\alpha\| = 1. \tag{2.7e}$$

Rather than solving the $\text{QCQP}_{\text{RLT+SILP}}$ directly, a cutting plane generation strategy is employed. Specifically, the infinite set of constraints (2.7e) is relaxed, and members of this set are generated as needed via a separation routine in polynomial time. As the work in this thesis is strongly related to this cutting plane strategy, we explain the cut generation routine in detail below.

Semidefinite Cut Generation Strategy

Denote $\alpha^j \equiv \alpha^j(X_L)$, $j = 1, ..., n$, as the set of linearly independent normalized eigenvectors of X_L. Then $X_L \succeq 0$ is equivalent to the condition that $(\alpha^j)^\top X_L \alpha^j \geq 0$ for $j = 1, ..., n$. Hence, focusing on the α-vectors corresponding to such eigenvectors of X_L, one can generate violated members of these constraints (in a relaxation) framework based on detected negative eigenvalues. While the Lanczos Algorithm could be used for efficiently finding such extremal eigenvalues, Sherali and Fraticelli [63] derive a (polynomial-time) separation mechanism for generating suitable members of (2.7e) in a sequential fashion. This operation is based on the LU factorization of X_L and the cuts can be generated in polynomial time.

To begin with, solve $\text{QCQP}_{\text{RLT+SILP}}$ with the constraints (2.7e) omitted. Let the resulting solution to this problem be denoted as $(\bar{x}, \overline{X}_L)$. If $\overline{X}_L \succeq 0$, then $\overline{X}_L$ solves Problem (2.7) (or equivalently (1.17)) as well. Otherwise, the solution $\overline{X}_L$ violates at least one of the constraints in (2.7e), and the following algorithm generates an α-vector based on the *superdiagonalization algorithm*.

Key Step:

Solve the first-level RLT relaxation along with any generated constraints of the type (2.7e). Let $(\overline{x}, \overline{X}_L)$ be the solution obtained. If $\overline{X}_L$ is PSD, stop; this is an optimal solution to (2.7). If a prescribed limit on the number of cuts to be generated has been attained, stop. The resulting enhanced model is recommended for solution to an MIP solver. Else, delete previously added inactive SDP cuts, put $i = 1$, $G^i = \overline{X}_L$, and continue until either of the two previous conditions are satisfied.

Step 1: Given G^i as $G^i \equiv \begin{pmatrix} G^i_{11} & (g^i)^\top \\ g^i & G \end{pmatrix}$, if $G^i_{11} < 0$, then set $\alpha^i = (\alpha_i, ..., \alpha_n)^\top = (1, 0, 0, ..., 0)^\top \in \mathbb{R}^{n-i+1}$, and go to Step 8. Else, go to Step 2.

Step 2: Check whether $i = n$. If yes, then $[xx^\top]_L$ evaluated at $(\overline{x}, \overline{X}_L)$ is a PSD matrix; stop and solve the resulting model to optimal solution via an MIP solver. Otherwise, proceed to Step 3.

Step 3: If $G^i_{11} = 0$, go to Step 4. Else, go to Step 6.

Step 4: ($G^i_{11} = 0$) If $G^i_{1j} = 0, \ \forall\, j$, go to Step 5.
Else, select j with the smallest resulting value of $\lambda = \dfrac{\phi - \sqrt{\phi^2 + 4\theta^2}}{2}$ from $j \in \{2, ..., n - i + 1\}$ and j satisfying $G^i_{1j} = G^i_{j1} \neq 0$, where $\theta = G^i_{1j} = G^i_{j1}$ and $\phi \equiv G^i_{jj}$. Accordingly, then, set $\alpha^i = (\alpha_i, 0, 0, ..., \alpha_{i+j-1}, 0, ..., 0)^\top \in \mathbb{R}^{n-i+1}$, where $\alpha_i = \dfrac{1}{1 + \dfrac{\lambda^2}{\theta^2}}$, $\alpha_{i+j-1} = \dfrac{\alpha_i \lambda}{\theta}$, and where $\lambda = \dfrac{\phi - \sqrt{\phi^2 + 4\theta^2}}{2}$.
Go to Step 8.

Step 5: Store $G^i_{11} = 0$, and delete the first row and column of G^i to get G^{i+1}. Go to Step 7.

Step 6: Store $\begin{pmatrix} G^i_{11} \\ g^i \end{pmatrix}$ and calculate $G^{i+1} = G - \dfrac{g^i (g^i)^\top}{G^i_{11}}$. go to Step 7.

Step 7: Let $i = i + 1$ and return to Step 1.

Step 8: If $i > 1$, go to Step 9. Otherwise, go to Step 11.

Step 9: From $\gamma = i - 1$, recursively calculate the value of α_γ and α^γ as below:

If $G_{11}^\gamma = 0$, then put $\alpha_\gamma = 0$; otherwise, compute $\alpha_\gamma = \dfrac{-(\alpha^{\gamma+1})^\top g^\gamma}{G_{11}^\gamma}$, where $\alpha^{\gamma+1} \equiv (\alpha_{\gamma+1}, ..., \alpha_n)^\top$. Set $\alpha^\gamma \equiv \begin{pmatrix} \alpha_\gamma \\ \alpha^{\gamma+1} \end{pmatrix}$, and go to Step 10.

Step 10: Normalize vector α, and then go to Step 11.

Step 11: Use $\alpha \equiv \alpha^1$ to generate a new SDP cut $\alpha^\top X_L \alpha \geq 0$, just like the type of (2.7e), append this new cut to the current relaxation, and return to the Key Step.

The foregoing approach establishes an inductive polynomial-time process for generating valid inequalities for the first RLT relaxation. Note that different variants of the termination criterion, for the semidefinite cut generation scheme, can be employed. Moreover, when used within a branch-and-bound framework, the cutting plane algorithm provides a significant tightening of the lower bound, obtained by using the RLT as a stand-alone approach, thereby reducing the effort required to obtain globally optimal solutions. Although this cut generation process is quite efficient in practice, its worst-case complexity of $O(n^3)$ could be an obstacle for large-scale problems, particularly if the matrix of second-order variables turns out to be dense. More importantly, all of the cuts generated are *rank-one* cuts, and this cut generation procedure is incapable of generating higher-rand order cuts.

Helmberg and Rendl [35] presented an approach of combining a semidefinite relaxation with a cutting plane technique, which used the triangle inequalities for triggering small adds, *i.e.*, they check after each iteration for a violated triangle inequality by enumeration. However, although the optimal value obtained from the RLT+SDP relaxation to QCQP gives a tight solution, the performance of RLT+SDP

has so far not been remarkable for large-scale 0-1 QCQPs. Konno *et al.* [38] have also proposed using a similar cut of the type $\tilde{\alpha}^\top X\tilde{\alpha} \geq 0$, where $\tilde{\alpha} \in \mathbb{R}^n$ is the *normalized eigenvector* corresponding to the *minimum eigenvalue* of the matrix X (evaluated at the solution $(\overline{x}, \overline{X})$, given that $\overline{X}$ is not positive semidefinite). However, computing $\tilde{\alpha}$ can be difficult in practice, and moreover, the round of cuts (2.7e) predicated on the augmented matrix (1.11d) closely mirrors the eigenvector-based cuts, while expending a significantly lower computational cost.

As a precursor to this approach, Shor [67] had recommended examining the minimum eigenvalue of X as a function $\lambda_1(X)$, where $\lambda_1(X) \equiv \min\{\alpha^\top X\alpha : \ \|\alpha\| = 1\}$ by the Raleigh-Ritz formula. Noting that $\lambda_1(X)$ is a concave, but nondifferentiable, function of X that is characterized by the minimum of a family of linear functions, Shor developed a nondifferentiable optimization approach for quadratic programs in which $\lambda_1(X) \geq 0$ is incorporated within the model formulation in lieu of $X \succeq 0$.

Using a similar thrust, Vanderbei and Benson [70] proposed an alternative, smooth, convex, finite linear programming representation for the relationship $X \succeq 0$, noting that $X \succeq 0$ if and only if it can be factored as $X = LDL^\top$, where L is a lower triangular matrix having unit diagonal elements and D is a nonnegative diagonal matrix. Denoting $d_j(X),\ j = 1, ..., n$, as the diagonal elements of D for a given $n \times n$ symmetric matrix $X \succeq 0$, Vanderbei and Benson [70] showed that $d_j(X)$ is a concave, twice differentiable function on the set of positive semidefinite matrices. Accordingly, they replaced $X \succeq 0$ by the nonlinear, smooth (but only implicitly defined) constraints $d_j(X) \geq 0,\ j = 1, ..., n$, and developed a specialized interior point algorithm to solve the resulting problem.

Lasserre [42] has also discussed the pros and cons of using semidefinite programming versus LP relaxations in solving polynomial programs. He concludes that although semidefinite representations are more theoretically appealing, LP

relaxation-based approaches have more practical significance due to the availability of effective software capable of handling large-scale problem instances. Further discussions on semidefinite programming relaxations versus LP relaxations are provided in Laurent and Rendl [44].

Sherali *et al.* [66] extended their semidefinite cutting plane mechanism for polynomial programming problems. Given an RLT relaxation, instead of imposing simply nonnegativity constraints on the RLT product variables, they imposed positive semidefiniteness on suitable dyadic variable-product matrices, and correspondingly derived implied semidefinite cuts. In the case of polynomial programs, there are several possible variants for selecting such particular variable-product matrices for imposing the positive semidefiniteness restrictions in order to derive implied valid inequalities, which they called v-*semidefinite cuts*. Various strategies for generating such cuts were presented, and they exhibited relative effectiveness of these cuts towards tightening the RLT relaxations and solving the underlying polynomial programming problems in conjunction with an RLT-based branch-and-bound scheme.

2.3 SOCP-based relaxations

In Burer and Chen [24], the authors introduce a lift-and-project procedure for 0-1 linear programming on the basis of p-order cone programming. This procedure generalizes and unifies several existing lift-and-project hierarchies.

Now, we introduce the p-order cone procedure in details. Considering the following feasible region,

$$F = \{x \in \{0,1\}^n : a_i^\top x \geq b_i \; \forall i \in \mathcal{L}\}. \tag{2.8}$$

The continuous convex relaxation could be given as follows immediately,

$$P = \{x \in \mathbb{R}^n : a_i^\top x \geq b_i \; \forall i \in \mathcal{L}\}. \tag{2.9}$$

The first step of p-order cone procedure is to lift the feasible region F into a higher dimensional space.

Proposition 1. *[24] Define* $r = \sqrt[p]{|\mathcal{J}|}/2$ *and* $c = e/2 \in \mathbb{R}^{|\mathcal{J}|}$. *Then* $x_J \in \{0,1\}^{|\mathcal{J}|}$ *implies* $\|x_\mathcal{J} - c\|_p \leq r$.

Note that the value of r relies on order p and $|\mathcal{J}|$ and that the value of c depends on $|\mathcal{J}|$. Using Proposition 1, we can rewrite F redundantly as the following formulation,

$$F = \{x \in \mathbb{R} : x = x^2, a_i^\top x \geq b_i \; \forall i \in \mathcal{L}, \|x_\mathcal{J} - c\| \leq r\}. \tag{2.10}$$

Considering that

$$\left.\begin{array}{l} a_i^\top x \geq b_i \\ \|x_\mathcal{J} - c\| \leq r \end{array}\right\} \Rightarrow \|(a_i^\top x - b_i)(x_\mathcal{J} - c)\|_p \leq r(a_i^\top x - b_i) \tag{2.11}$$

and $a_i^\top x - b_i$ is kept its non-negativity, F can be represented as

$$F = \{x \in \mathbb{R} : x = x^2, \|x_\mathcal{J} a^\top x_i - b_i x_\mathcal{J} - (a_i^\top x - b_i)c\|_p \leq r(a_i^\top x - b_i) \; \forall i \in \mathcal{L}\}.$$

Next, introducing a matrix variable X as same as the one in (1.7), where $X \in \mathbb{R}^{n \times n}$

and $X = xx^\top$ and defining

$$\hat{F} = \left\{ (x, X) \in \mathbb{R}^n \times \mathbb{R}^{n \times n} : \begin{array}{l} X = xx^\top, \ \text{diag}(X) = x \\ \|X_{\mathcal{J}} a_i - b_i x_{\mathcal{J}} - (a_i^\top x - b_i)c\|_p \leq r(a_i^\top x - b_i) \ \forall i \in \mathcal{L} \end{array} \right\}. \tag{2.12}$$

It can be seen that $F = \text{proj}_x(\hat{F})$. In another word, $\hat{F}$ is the lifted version of F. Additionally, a convex relaxation of $\hat{F}$ are obtained by dropping the non-convex constraint $X = xx^\top$ from $\hat{F}$, which I described as follows:

$$\hat{P} = \left\{ (x, X) \in \mathbb{R}^n \times \mathbb{R}^{n \times n} : \begin{array}{l} \text{diag}(X) = x \\ \|X_{\mathcal{J}} a_i - b_i x_{\mathcal{J}} - (a_i^\top x - b_i)c\|_p \leq r(a_i^\top x - b_i) \forall i \in \mathcal{L} \end{array} \right\}. \tag{2.13}$$

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

$$N(P) = \text{proj}_x(\hat{P}). \tag{2.14}$$

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

Proposition 2. *[24] $P^{01} \subseteq N(P) \subseteq P$, where $P^{01} := conv(F)$.*

For a fixed set $\mathcal{J}$, a single implementation of the p-cone lift-and-project procedure gives rise to a family of relaxations of P^{01} parameterized by p. According to the monotonicity property that the larger p is, the tighter the corresponding relaxation. Hence $p = \infty$ enjoys the strongest relaxation among $p \in [1, \infty]$. However, when $p = 2$, the procedure enjoys a better theoretical iteration complexity than existing lift-and-project technique when applying one iteration.

SOCP falls between LP and SDP. We want to construct a hierarchy based on SOCP relaxation, which could blend RLT and SDP procedures' advantage together and moderate their disadvantages in the same time.

3 Minimum Triangle Inequalities and Algorithm

In this chapter, we begin by enhancing existing linearization techniques (notably RLT-based methods) for solving 0-1 QCQPs with a new class of valid cuts, which we refer to as *Minimum Triangle* **(MINT)** inequalities. These MINT cuts are derived based on the relationship that $y_{ij} = \min\{x_i, x_j\} = \max\{0, x_i + x_j - 1\}$ at any feasible solution (and hence, at optimality), and we prove that these *super-linear* cuts are tighter than the existing triangle inequalities (1.16), thereby leading to improved lower bounds at each node of the branch-and-bound tree. Then a new variable fixing strategy is proposed by according to the logic used in MINT cuts. The MINT cuts in concert with the newly designed variable fixing and branching strategy give rise to a branch-and-bound algorithm that we refer to as the *Minimum Triangle Algorithm* **(MINT Algorithm)**.

3.1 Minimum Triangle Inequalities

Firstly, we introduce MINT cuts. The scheme of deriving MINT cuts is according to the *order* d. The order mentioned here means the number of indices appearing in a cut. We come up with this cut starting from order 1.

Order $d = 1$:

When there is only one index, we say i, the implied constraint is $y_{ii} = x_i$. In our relaxation formulation we have already replaced y_{ii} with x_i. The order 1 is redundant.

Order $d = 2$:

Order 2 is simple. from $y_{ij} = \min\{x_i, x_j\}$, we derive the following approximations:

- $y_{ij} \geq 0$.

  Since x_i and $x_j \geq 0$, this is obvious.

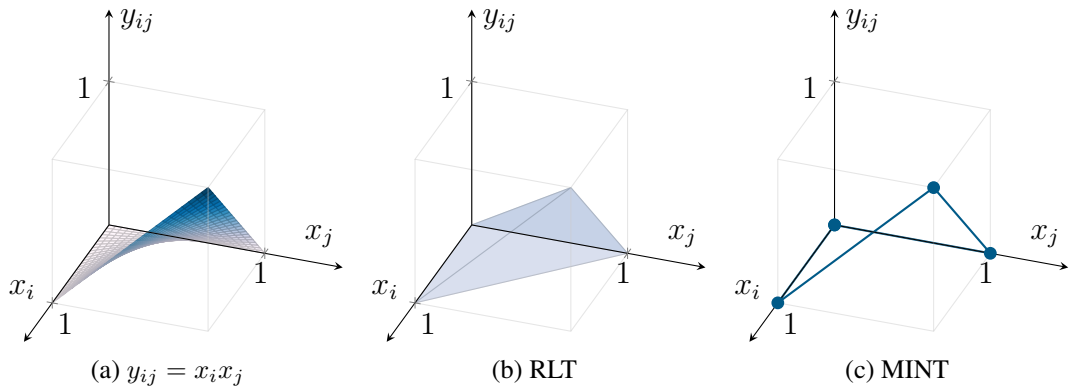
- $y_{ij} \leq x_i$ and $y_{ij} \leq x_j$.

  $y_{ij} = \min\{x_i, x_j\} \Rightarrow y_{ij} \leq \min\{x_i, x_j\} \Rightarrow y_{ij} \leq x_i, y_{ij} \leq x_j$.

- $y_{ij} + 1 \geq x_i + x_j$.

  $y_{ij} = \min\{x_i, x_j\} \Rightarrow y_{ij} + \max\{x_i, x_j\} = \min\{x_i, x_j\} + \max\{x_i, x_j\}$. Combined with $\max\{x_i, x_j\} \leq 1$ and $\min\{x_i, x_j\} + \max\{x_i, x_j\} = x_i + x_j$, it is easy to get that $y_{ij} + 1 \geq x_i + x_j$.

We see that for order 2, the cuts we get are exactly the RLT constraints. From Figure 3.1, it is easy to see that when relaxing the binary variables to continuous ones, we get Figure 3.1a, which is nonconvex; the notable RLT relaxation is shown as in Figure 3.1b; The MINT cuts for 2 variables' situation are constructed on basis of the four segments of 4 feasible points, as shown in Figure 3.1c, whose convex hull is RLT relaxation. After relaxing $y_{ij} = \min\{x_i, x_j\} = y_{ij} = \max\{0, x_i + x_j - 1\}$, it leads to 3.1b, which implies that the tightness of MINT is at least as good as RLT.



(a) $y_{ij} = x_i x_j$ (b) RLT (c) MINT

Figure 3.1: Illustration of different relaxations in the situation of two variables x_i and x_j

Order $d = 3$:

Consider the following minimum triangle inequality to the 0-1 QCQP (1.2):

$$\textbf{MINT-1}: \; y_{ij} + y_{ik} - y_{jk} \leq \min\{x_i, x_j\} + \min\{x_i, x_k\} - \min\{x_j, x_k\}. \tag{3.1}$$

As $x_i, x_j, x_k \in \{0, 1\}$, it can be easily verified that (3.1) is a valid inequality for the 0-1 QCQP, and in fact, it is *exact* at all feasible points of the corresponding 0-1 hypercube. Comparing (3.1) with its corresponding triangle inequality (1.16b), the following theorem is evident.

Theorem 2. *A minimum triangle inequality is tighter than its corresponding triangle inequality.*

Proof. It is sufficient to prove that the RHS of (3.1) is less than x_i. Recognizing that,

$$\min\{x_i, x_j\} \equiv \frac{1}{2}\{x_i + x_j - |x_i - x_j|\}, \tag{3.2}$$

and using the well-known triangle inequality that

$$|x_i - x_j| = |x_i - x_k + x_k - x_j| \leq |x_i - x_k| + |x_j - x_k|, \tag{3.3}$$

we get:

$$\begin{aligned}
y_{ij} + y_{ik} - y_{jk} &\leq \min\{x_i, x_j\} + \min\{x_i, x_k\} - \min\{x_j, x_k\} && \text{(3.4a)}\\
&= x_i + \frac{1}{2}\{|x_j - x_k| - |x_i - x_j| - |x_i - x_k|\} && \text{(3.4b)}\\
&\leq x_i + \frac{1}{2}\{|x_j - x_i| + |x_i - x_k| - |x_i - x_j| - |x_i - x_k|\} && \text{(3.4c)}\\
&= x_i, && \text{(3.4d)}
\end{aligned}$$

which completes the proof. □

Remark 2. The inequality (3.1) is predicated on the equation $y_{ij} = \min\{x_i, x_j\}$, which involves the minimum function, and moreover, as the 3-tuple (x_i, x_j, y_{ij}) forms a right-angled triangle in the X_L matrix (1.9), this lends the term "*minimum triangle inequalities*" to this particular class of cutting planes. Furthermore, as seen in the proof of Theorem 2, the minimum function terms are amenable to algebraic manipulation and approximation via the use of absolute value triangle inequalities, further augmenting the prescribed nomenclature of these cutting planes.

From an algorithmic standpoint, recognizing that the MINT cuts contain the minimum function terms, which are both super-linear and nondifferentiable, a mechanism to handle such cuts is via the use of auxiliary *indicator binary variables*. Let,

$$\delta_{ij} = \begin{cases} 1, & \text{if } x_i \leq x_j \\ 0, & \text{otherwise} \end{cases}. \tag{3.5}$$

Then, $y_{ij} = \min\{x_i, x_j\}$ can be equivalently expressed as:

$$y_{ij} \leq x_i \tag{3.6a}$$

$$y_{ij} \leq x_j \tag{3.6b}$$

$$y_{ij} \geq x_i - M(1 - \delta_{ij}) \tag{3.6c}$$

$$y_{ij} \geq x_j - M\delta_{ij}. \tag{3.6d}$$

To demonstrate the efficacy of the MINT cut (3.1), consider the following illustrative example. Here, the objective function and the quadratic constraint are both nonconvex, and the optimal solution is given by $x^* = (0, 0, 0, 1, 0)^\top$ with $f(x^*) = -2$.

Table 3.1: Lower bounds obtained at root node for Example 1

f_{opt}	RLT	RLT+TI	RLT+MINT-1	RLT+SDP
-2	-36.9375	-35.5625	-27.5	-36.2925

Example 1.

$$
\begin{aligned}
\min \quad f(x) = \; & 163x_1^2 - 92x_1x_2 + 565x_1x_3 + 77x_1x_4 - 6x_1x_5 + 55x_2^2 + 5984x_2x_3 \\
& -22x_2x_4 + 31x_2x_5 + 55x_3^2 + 22x_3x_4 + 2543x_3x_5 - 2x_4^2 - 76x_4x_5 + 10x_5^2 \\
s.t. \quad & -2x_1^2 - 4x_1x_2 - 2x_2^2 + 8x_1 + 6x_2 + x_3 - 4x_4 \leq -2.5 \\
& x_1 - 2x_2 + x_3 + x_4 + x_5 \leq 1 \\
& x \in \{0,1\}^5.
\end{aligned}
$$

Table 3.1 lists the lower bounds obtained by using four different relaxations, which are RLT relaxation, RLT relaxation + Triangle inqualities, RLT relaxation + MINT-1 cut and RLT relaxation + SDP constraints separately. As clearly seen, the RLT+MINT-1 relaxation generates a significantly tighter lower bound than that of RLT, RLT+TI, and even RLT+SDP relaxations in this example.

Following the same vein as ineuqality (3.1), an entire class of such minimum triangle inequalities can be derived. A few sample cuts belonging to this overarching class of cuts is presented below, and moreover, a generic automatic mechanism that can be used to derive corresponding MINT cuts of various orders is expounded subsequently in the following section. Furthermore, using certain inherent characteristics of min/max functions and absolute value inequalities, we derive *approximations* of these MINT cuts, which are computationally more viable while yet retaining the inherent strength of the original cuts.

There are several MINT cuts and their approximations that can be derived:

MINT-2: $$y_{ij} - y_{ik} - y_{jk} \geq \min\{x_i, x_j\} - \min\{x_i, x_k\} - \min\{x_j, x_k\}. \tag{3.8}$$

Approximation : Using the reverse triangle inequality

$$|x_i - x_j| \geq |x_i| - |x_j| = x_i - x_j, \tag{3.9}$$

an approximation to MINT-2 can be derived as follows:

$$\begin{aligned} y_{ij} - y_{ik} - y_{jk} &\geq \frac{1}{2}\{x_i - x_k + x_j - x_k - |x_i - x_j|\} - x_k \\ &= \frac{1}{2}\{x_i + x_j - |x_i - x_j|\} - 2x_k \\ &= x_i + x_j - 2x_k - \max\{x_i, x_j\}. \end{aligned} \tag{3.10}$$

MINT-3: $$\begin{aligned} y_{ij} - y_{ik} - y_{jk} &\leq \min\{x_i, x_j\} - \min\{x_i, x_k\} - \min\{x_j, x_k\} \\ &= \frac{1}{2}\{|x_i - x_k| + |x_j - x_k| - |x_i - x_j|\} - x_k. \end{aligned} \tag{3.11}$$

Approximation 1: Using

$$\min\{x_i, x_j\} + \max\{x_i, x_j\} = x_i + x_j, \tag{3.12}$$

MINT-3 can be rewritten as

$$y_{ij} - y_{ik} - y_{jk} \leq -2x_k + \max\{x_i, x_k\} + \max\{x_j, x_k\} - \max\{x_i, x_j\}. \tag{3.13}$$

Approximation 2: Using $|x_i - x_k| \leq |x_i - x_j| + |x_j - x_k|$, the second approximation of MINT-3 is

$$\begin{aligned} y_{ij} - y_{ik} - y_{jk} &\leq \frac{1}{2}\{|x_i - x_k| + |x_j - x_k| - |x_i - x_j|\} - x_k \\ &\leq |x_j - x_k| - x_k. \end{aligned} \tag{3.14}$$

Approximation 3: Using $|x_i - x_j| \geq |x_i| - |x_j| = x_i - x_j$, the third approximation of MINT-3 is

$$\begin{aligned} y_{ij} - y_{ik} - y_{jk} &\leq \frac{1}{2}\{|x_i - x_k| + |x_j - x_k| - |x_i - x_j|\} - x_k \\ &\leq \frac{1}{2}\{|x_i - x_k| + |x_j - x_k| - x_i + x_j\} - x_k. \end{aligned} \tag{3.15}$$

MINT-4: $$\begin{aligned} y_{ij} + y_{ik} - y_{jk} &\geq \min\{x_i, x_j\} - \min\{x_j, x_k\} \\ &= \frac{1}{2}\{x_i - x_k - |x_i - x_j| + |x_j - x_k|\}. \end{aligned} \tag{3.16}$$

Approximation 1: Once again using the subadditivity property (3.3), an approximation to MINT-4 can be obtained as:

$$\begin{aligned} y_{ij} + y_{ik} - y_{jk} &\geq \frac{1}{2}\{x_i - x_k - |x_i - x_j| + |x_j - x_k|\} \\ &\geq \frac{1}{2}\{x_i - x_k - |x_i - x_k|\} \\ &= x_i - \max\{x_i, x_k\}. \end{aligned} \tag{3.17}$$

Approximation 2: Using (3.9), we get:

$$y_{ij} + y_{ik} - y_{jk} \geq \frac{1}{2}\{x_i - x_k - |x_i - x_j| + x_j - x_k\}$$

$$= \frac{1}{2}\{x_i + x_j - 2x_k - |x_i - x_j|\}$$
$$= x_i + x_j - x_k - \max\{x_i, x_j\}. \tag{3.18}$$

MINT-5: $y_{ij} + y_{ik} + y_{jk} \geq \min\{x_i, x_j\} + \min\{x_i, x_k\} + \min\{x_j, x_k\}$ (3.19)

Approximation : Since $\min\{x_i, x_j\} = \frac{1}{2}\{x_i + x_j - |x_i - x_j|\}$, we can get the following inequality:

$$y_{ij} + y_{ik} + y_{jk} \geq \min\{x_i, x_j\} + \min\{x_i, x_k\} + \min\{x_j, x_k\}$$
$$= x_i + x_j + x_k - \frac{1}{2}\{|x_i - x_j| + |x_i - x_k| + |x_j - x_k|\}. \tag{3.20}$$

Linearizing Order 3 MINT cuts

While the methodology utilized in implementing MINT cuts, via the addition of indicator binary variables that enforce appropriately chosen min/max/absolute value constraints, can be very effective for small-sized problems ($n \leq 50$), such an approach would be prohibitively expensive from a computational standpoint for large-scale instances. However, apart from demonstrating the tightness of the MINT cut, Theorem 2 also provides us with an insight into effectively linearizing these MINT cuts by using simple variable relationships and inequality bounds.

As seen in the proof of Theorem 2, using (3.2) and (3.3), the MINT-1 cut was approximated by its linearized version, given by

$$y_{ij} + y_{ik} - y_{jk} \leq x_i. \tag{3.21a}$$

Note that in this case, the linearized MINT-1 cut coincides with its corresponding triangle inequality (1.16b). Similarly, all of the MINT cuts can be linearized via their approximations, and the set of these linearized cuts is collected below (See details in Appendix):

$$y_{ij} - y_{jk} - y_{ik} \geq -x_i - x_j \tag{3.21b}$$

$$1 + y_{ij} - y_{jk} - y_{ik} \geq x_i + x_j - 2x_k \tag{3.21c}$$

$$y_{ij} - y_{jk} - y_{ik} \leq 2(1 - x_k) \tag{3.21d}$$

$$y_{ij} - y_{jk} - y_{ik} \leq x_j \tag{3.21e}$$

$$y_{ij} - y_{jk} + y_{ik} \geq -1 \tag{3.21f}$$

$$1 + y_{ij} - y_{jk} + y_{ik} \geq x_i \tag{3.21g}$$

$$1 + y_{ij} - y_{jk} + y_{ik} \geq x_i + x_j - x_k \tag{3.21h}$$

$$1 + y_{ij} + y_{jk} + y_{ik} \geq x_i + x_j + x_k. \tag{3.21i}$$

Theorem 3. *The class of linearized MINT cuts subsumes the combined class of RLT constraints and triangle inequalities.*

Proof. It is sufficient to prove that both RLT constraints and triangle inequalities can be derived from the class of minimum triangle inequalities. We begin by demonstrating the derivation of RLT constraints using linearized MINT cuts.

Consider,

$$y_{ij} = \min\{x_i, x_j\} \Rightarrow \begin{cases} y_{ij} \leq x_i \\ y_{ij} \leq x_j \end{cases}. \tag{3.22}$$

Now, without loss of generality, suppose that $x_i \leq x_j$. Then $y_{ij} = \min\{x_i, x_j\} \Rightarrow$

$y_{ij} = x_i$. Combined with $1 \geq x_j$, we get

$$x_i + x_j - y_{ij} \leq 1. \tag{3.23}$$

As $y_{ij} \geq 0$, it is evident from (3.22) and (3.23) that the RLT inequalities are a subset of the MINT cuts.

To show that the MINT cuts subsume the triangle inequalities (1.16), from (3.4), we have $y_{ij}+y_{ik}-y_{jk} \leq x_i$. An appropriate reordering of indices yields $y_{ij}+y_{jk}-y_{ik} \leq x_j$ and $y_{ik} + y_{jk} - y_{ij} \leq x_k$.

Finally, $y_{ij} + y_{ik} + y_{jk} = \min\{x_i, x_j\} + \min\{x_i, x_k\} + \min\{x_j, x_k\}$

$$\begin{aligned} &= x_i + x_j + x_k - \frac{1}{2}\{|x_i - x_j| + |x_i - x_k| + |x_j - x_k|\} \\ &\geq x_i + x_j + x_k - 1 \end{aligned}$$

$\Rightarrow 1 + y_{ij} + y_{ik} + y_{jk} \geq x_i + x_j + x_k.$

This completes the proof. □

When the order $d \geq 4$, we also can demonstrate valid cuts using the same logic. However, the corresponding MINT cuts are more complicated and with huge quantity.

3.2 Minimum Triangle Algorithm

Having established the expansive nature of the MINT class of inequalities, we next present a novel *variable fixing strategy* based on the inherent properties of 0-1 QCQP problems, which can be gainfully employed at the solution of any lower bounding relaxation. The process of deriving this variable fixing mechanism also motivates the choice of a *branching variable* that forms the basis of creating the child nodes in the context of the prescribed branch-and-bound algorithm. In our numerical experiments,

we demonstrate the use of these two complementary strategies towards significantly improving the convergence of the search tree.

Variable Fixing Strategy:

Consider the following pair of valid equalities for a 0-1 QCQP:

$$y_{ij} = \min\{x_i, x_j\} \equiv \max\{0, x_i + x_j - 1\}. \tag{3.24}$$

Without loss of generality, assume that $x_i \leq x_j$. Then,

$$y_{ij} = x_i = \max\{0, x_i + x_j - 1\}, \tag{3.25}$$

which leads to two possible cases.

Case 1: $x_i + x_j - 1 \leq 0$

In this case, $y_{ij} = x_i = 0;\ x_j \in \{0, 1\}$, *i.e.*, the value of x_i can be fixed to 0, and the value of the x_j-variable cannot be determined in this case.

Case 2: $x_i + x_j - 1 > 0$

In this case, $y_{ij} = x_i = x_i + x_j - 1 \Rightarrow x_j = 1;\ x_i \in \{0, 1\}$, *i.e.*, the value of x_j can be fixed to 1, and the value of the x_i-variable cannot be determined in this case.

Note that in both cases, at least one variable can be fixed to either 0 or 1. Hence, in the context of a branch-and-bound scheme, the foregoing analysis extends into a variable fixing and associated branching strategy, which is presented in the theorem and its following corollary below.

Theorem 4. *Let $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ be a solution to a relaxation of the 0-1 QCQP. Without loss of generality, assume that $\bar{x}_1 \leq \bar{x}_2 \leq \cdots \leq \bar{x}_n$. Then, there exists a $\hat{p} \in \{1, ..., n\}$ such that $x_i = 0,\ \forall\, i < \hat{p}$, $x_i = 1,\ \forall\, i > \hat{p}$ and $x_{\hat{p}} \in \{0, 1\}$.*

Proof. Select $\hat{p} \in \{1, ..., n\}$ such that $\bar{x}_{\hat{p}-1} + \bar{x}_{\hat{p}} - 1 \leq 0$ and $\bar{x}_{\hat{p}} + \bar{x}_{\hat{p}+1} - 1 > 0$. Then, beginning with $i = 1$, in increments of one, fix the variables to either 0 or 1 as follows:

$$x_i = \begin{cases} 0, & \text{if } \bar{x}_i + \bar{x}_{i+1} - 1 \leq 0 \\ 1, & \text{if } \bar{x}_{i-1} + \bar{x}_i - 1 > 0 \end{cases} \Rightarrow x_i = \begin{cases} 0, & \text{if } i < \hat{p} \\ 1, & \text{if } i > \hat{p} \end{cases}. \tag{3.26}$$

Then, all variables are fixed at either 0 or 1, and $x_{\hat{p}} \in \{0, 1\}$, which completes the proof. □

Corollary 1. *Every node of the branch-and-bound tree either generates a feasible solution and/or determines a branching variable to produce two children nodes.*

Proof. Using Theorem 4, fix all variables to either 0 or 1, except the $x_{\hat{p}}$-variable. Then, evaluate the following two solutions for feasibility, and select $x_{\hat{p}}$ as the branching variable to create two child nodes.

$$x^{(1)} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_{\hat{p}-1}, \bar{x}_{\hat{p}}, \bar{x}_{\hat{p}+1}, ..., \bar{x}_n) = (0, 0, ..., 0, 0, 1, ..., 1) \tag{3.27a}$$

$$x^{(2)} = (\bar{x}_1, \bar{x}_2, ..., \bar{x}_{\hat{p}-1}, \bar{x}_{\hat{p}}, \bar{x}_{\hat{p}+1}, ..., \bar{x}_n) = (0, 0, ..., 0, 1, 1, ..., 1). \tag{3.27b}$$

□

We are now ready to unveil the *Minimum Triangle Algorithm* (**MINT Algorithm**), which is a specialized branch-and-bound scheme that utilizes the RLT-based relaxation enhanced with an appropriately chosen subset of MINT cuts for the 0-1 QCQP, along with the variable fixing strategy presented in Theorem 4.

A branch-and-bound algorithm is one among the most widely used mechanisms for solving various optimization problems to globally optimal solutions, especially for those arising in discrete and combinatorial optimization. The branch-and-bound search process involves in enumerating all candidate solutions systematically, where large subsets of fruitless candidates are cut off, by utilizing estimated lower and upper bounds of the objective function. The branch-and-bound mechanism generates a searching *tree*, and the very first node of this searching tree is called the *root* node. When a node is processed and needs further refinement, the branching step generates two new subnodes, namely the *children* nodes. The *incumbent solution* refers to the best solution currently known, and corresponding objective value is the *incumbent value* (typically set to $+\infty$ at the start of execution of the algorithm).

At each stage t of the branch-and-bound framework, $t = 0, 1, ..., T$, there will be a set of non-fathomed nodes or *active nodes* and we use A_t to represent it, where each node $a \in A_t$ is indexed by a set of *fixed variables*. To initialize, at $t = 0$, the set $A_0 = \{0\}$, with none of the variables being fixed. For each node $a \in A_t$, a lower bound LB_a will be given by $\nu[\text{0-1 QCQP}]^a_{\text{LP}}$, where, in our notation, $\nu[\text{0-1 QCQP}]^a_{\text{LP}}$ represents the objective function value corresponding to the solution of an LP relaxation to the 0-1 QCQP at node a. (In our case, this LP relaxation is constructed with the RLT constraints augmented by suitably chosen MINT cuts.) As a result, the (global) *lower bound* at stage t for 0-1 QCQP (and equivalently, 0-1 QCQP_{RLT}) is given by

$$\text{LB} \equiv \min_{a \in A_t} \{\text{LB}_a\} \tag{3.28}$$

Whenever any lower bounding node subproblem, corresponding to some node $a \in A_t$, is solved, we can apply the variable fixing strategy for purpose of (potentially) deriving an upper bound and possibly updating the incumbent solution on the overall

problem, where the corresponding lower bound value is given by LB_a. Accordingly, let (x^*, y^*) denote the best such incumbent solution found, and the corresponding objective value is represented to ν^*. Naturally, whenever $\text{LB} \geq \nu^*$, we can fathom node a. (Practically, we can cut off node a whenever $\text{LB}_a \geq \nu^*(1-\varepsilon)$, for some specified optimality tolerance $\varepsilon > 0$.) It follows that the active nodes at any stage t would satisfy $\text{LB}_a < \nu^*,\ \forall a \in A_t$. Now we select a node $a(t)$ that yields the *smallest* lower bound from the set of active nodes, *i.e.*, for which $\text{LB}_{a(t)} \equiv \text{LB}$ as given by (3.28). It should be noted that for the corresponding solution $(x^{a(t)}, y^{a(t)})$ to 0-1 $\text{QCQP}^{a(t)}_{\text{LP}}$, we could not possibly have $\bar{y}_{ij} = \bar{x}_i\bar{x}_j,\ \forall (i,j), i < j$, because then the current solution would be feasible to 0-1 $\text{QCQP}^{a(t)}_{\text{LP}}$, thereby implying that $\text{LB}_{a(t)} \geq \nu^*$, which is a contradiction. Therefore, we can detect a branching variable and thus split the node subproblem according to Corollary 1, and then go to the next stage.

A more formal statement and flow chart (Figure 3.3) of the aforementioned procedure is given below.

Step 0: (*Initialization*) Set $t = 0$ and $t_{\text{max}} = 0$. Define $A_t = \{t\}$, and the current incumbent solution (if any) as $\nu^* = UB$. Go to Step 1.

Step 1: (*Node Selection Step*) If $A_t = \varnothing$, stop; return ν^* as optimum. Else, select an active node from A_t, say node $a(t)$, and solve 0-1 $\text{QCQP}^{a(t)}_{\text{LP}}$. Let $\text{LB}_a \equiv \nu[\text{0-1 QCQP}^{a(t)}_{\text{LP}}]$ be the optimum objective function value at node a. Go to Step 2.

Step 2: (*Fathoming Step*) If $\text{LB}_a \geq \nu^*$, then fathom node a, and update $A_t \leftarrow A_t \backslash \{a(t)\}$, and go to Step 1. If $\text{LB}_a < \nu^*$ and feasible to 0-1 QCQP, then set $\nu^* = \text{LB}_a$, fathom node a, update $A_t \leftarrow A_t \backslash \{a(t)\}$, and go to Step 1. Else, go to Step 3.

Step 3: (*Variable Fixing Step*) Using Theorem 4 and its corollary, fix all variables to

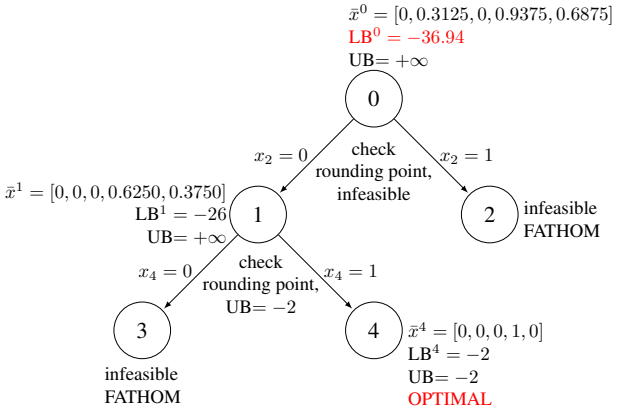
either 0 or 1, and evaluate the two candidate solutions $x^{(1)}$ and $x^{(2)}$, given by (3.27a)-(3.27b), for feasibility and incumbency. Update incumbent if possible, and proceed to Step 4.

Step 4: (*Branching Step*) Choose a branching variable based on Corollary 1, create two children nodes $t_{\max}+1$ and $t_{\max}+2$, and update $t_{\max} \leftarrow t_{\max}+2$. Set $t \leftarrow t+1$, $A_t \leftarrow A_t \cup \{t_{\max}+1, t_{\max}+2\} \backslash \{a(t)\}$. Go to Step 1.

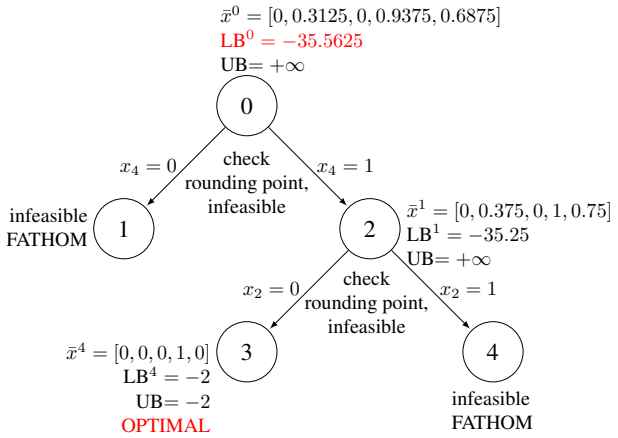
The foregoing MINT algorithm is used for solving Example 1. Figures 3.2 showcases the performance of different branch-and-bound strategies: the first uses the RLT relaxation to determine the lower bounds and we use simple rounding (to the nearest integer) to obtain upper bounds; the second combines MINT relaxation with rounding strategy; the last strategy uses the MINT algorithm. As clearly seen, the MINT algorithm dominates the RLT procedure in every performance metric including quality of the lower bounding solution, number of nodes required to determine the global optimum, and even the number of nodes required to find the first feasible solution.

3.3 Computational results

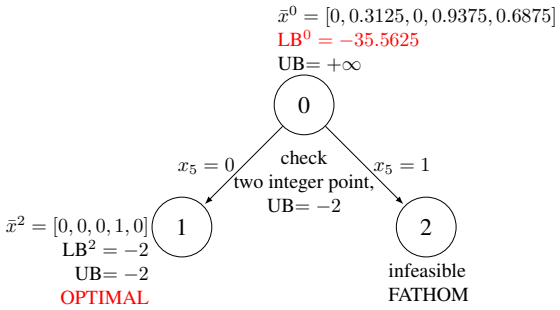
In this section, we present some preliminary numerical experiments, by solving standard test instances from the literature, which demonstrate the efficiency of the developed theoretical and algorithmic advancements towards solving 0-1 QCQPs. All of the numerical experiments were performed on a computer equipped with CPU 3.70GHz and RAM 32.0GB. The underlying linear programming relaxations and 0-1 MILP problems were solved using Gurobi [33], and the SDP relaxations are modeled using CVX version 2.0, utilizing SeDuMi as the backend solver. In all cases, the default



(a) BBRLT_rounding



(b) BBMINT_rounding



(c) BBMINT_minta

Figure 3.2: Different branch-and-bound strategies for Example 1

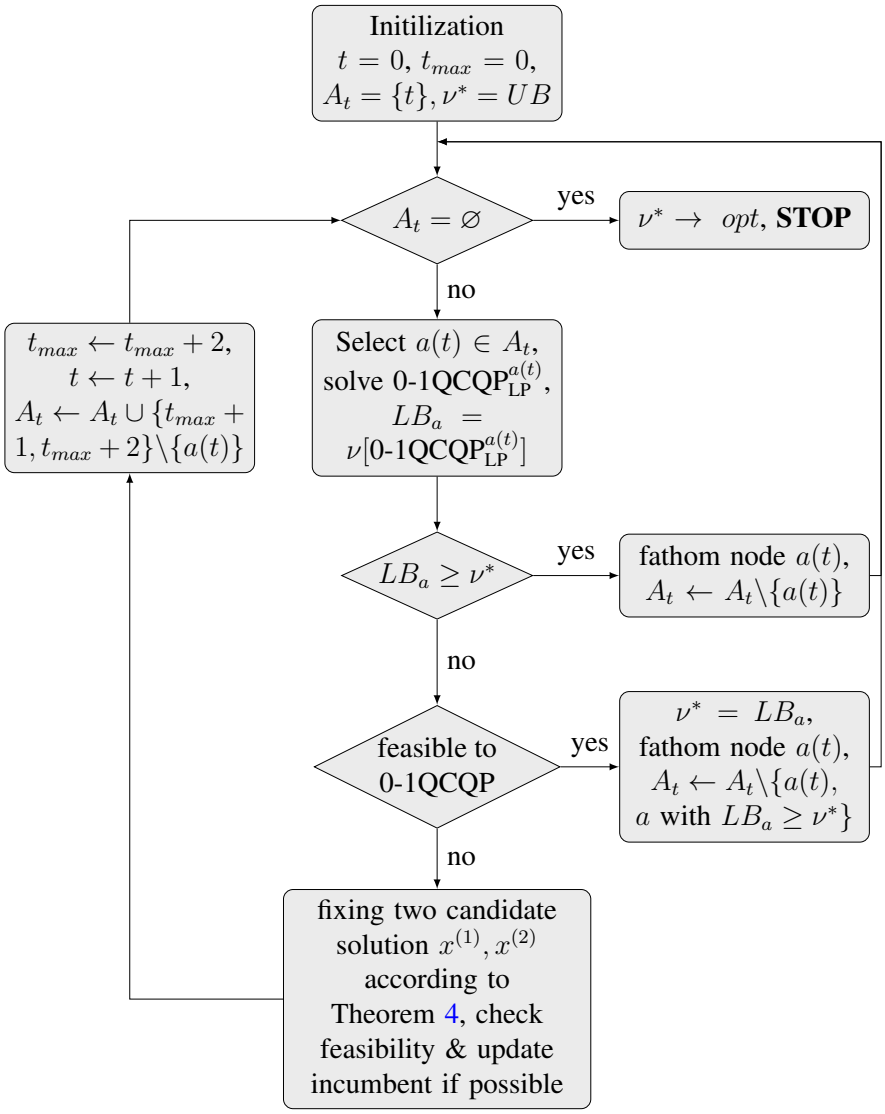


Figure 3.3: Flow chart of MINT algorithm

numerical tolerance specified by the software is used as a termination criterion.

3.3.1 Quasi Clique Problems

Firstly, we show the performance of our proposed relaxation with MINT cuts on the maximum quasi-clique problem. The testing instances are from Pattillo *et al.* [54].

Let $G = (V, E)$ denote a simple *undirected* graph with the set V of n vertices

and the set E of m edges. Graph G is called *complete* if all its vertices are pairwise adjacent. If C is a subset of V such that the subgraph $G[C]$ induced by C on G is complete, we call C a *clique*. Furthermore, a clique is called a *maximal* clique if there is no larger clique including it, and it is called a *maximum* clique if we cannot find any clique with greater number of vertices in the graph. The classical *maximum clique problem* is to explore a clique with maximum cardinality in G, which is called the *clique number* and denoted by $\omega(G)$. Given $G = (V, E)$ and a fixed real γ satisfying $0 < \gamma < 1$, a subset of vertices Q is called γ*-quasi-clique* or, simply, a γ*-clique* if the edge density of the indued subgraph $G[Q]$, which is given by the ratio of the number of edges in $G[Q]$ to $\binom{|Q|}{2}$, is at least γ. The *maximum* γ*-clique problem* looks for a γ-clique with the largest number of vertices in graph G. Specifically, for $\gamma = 0$, the maximum γ-clique problem would be trivial; and for $\gamma = 1$, it would become the classical maximum clique problem.

MIP formulations of the maximum γ-clique problem are as follows:

$$\omega_\gamma(G) = \max \sum_{i=1}^{n} x_i \tag{3.29a}$$

$$s.t. \quad \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij} x_i x_j \geq \gamma \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i x_j \tag{3.29b}$$

$$x_i \in \{0, 1\},\ i = 1, ..., n, \tag{3.29c}$$

where $a_{ij} = 1$ if there is an edge between node i and j, *i.e.*, $(i, j) \in E$, and $a_{ij} = 0$ otherwise.

As for test instances, we use some examples from Trick's graph coloring page (the corresponding page can be found at http://mat.gsia.cmu.edu/COLOR04/#XXSGB). Those graph with vertices $n \leq 100$ are picked. Since the problem is maximizing problem, the smaller the bound (upper bound) is, the better.

For $\gamma = 0.75, 0.85, 0.95$ and 1, we compute the upper bound for maximum γ-clique problem with three different methods:

- RLT relaxations [59],
- MINT cuts: we choose a few cuts from the pool of MINT cuts and removing the redundant ones,
- Proposition 3 in [54]: The γ-clique number $\omega_\gamma(G)$ of a graph G with n vertices and m edges satisfied the following inequality:

$$\omega_\gamma(G) \leq \frac{\gamma + \sqrt{\gamma^2 + 8\gamma m}}{2\gamma}.$$

Moreover, if G is connected then

$$\omega_\gamma(G) \leq \frac{\gamma + 2 + \sqrt{(\gamma+2)^2 + 8(m-n)\gamma}}{2\gamma}.$$

Tables 3.2-3.5 show the performance of MINT cuts. The last two rows give the average bound and the number of istances where the bound has improved seperately. From the results, we can see that, MINT inequalities gives the best bound on average (except for $\gamma = 1$). And MINT relaxation can improve much more instances' bound than RLT. On the other side, on average, the computational run time of the MINT is gerater than the run time required by the RLT formulation. This is a natural consequence of the fact that there are more constraints in the MINT formulation, and therefore, it is reasonable to expect that the run time of MINT is more than RLT in general. But, this increase in run time occurs at the expense of getting a significantly improved lower bound. To show the results more intuitively, box-plots of %improvement for 0.75 and 0.85-quasi Clique problems are given in Figure 3.4.

Table 3.2: Maximum 0.75-quasi Clique problems

Graph	n	γ	Analytical bound	MINT		RLT	
				Bound	Time(s)	Bound	Time(s)
myciel3	11	0.75	7.0641	4.2381	0.0090	6.1471	0.0035
myciel4	23	0.75	13.2946	8.5588	0.2224	12.5244	0.0127
queen5_5	25	0.75	20.8954	11.7826	0.3424	15.5660	0.0183
1-FullIns_3	30	0.75	15.6184	10.7829	0.7415	15.9434	0.0193
queen6_6	36	0.75	27.9235	14.9009	0.6883	21.0796	0.0579
2-Insertions_3	37	0.75	11.6667	12.6667	0.7965	18.9444	0.0237
myciel5	47	0.75	24.3580	17.0360	1.9769	25.0955	0.0692
queen7_7	49	0.75	35.6272	19.2143	2.2176	27.7114	0.1147
2-FullIns_3	52	0.75	21.8507	18.0573	2.8328	26.9437	0.0947
3-Insertions_3	56	0.75	13.9726	18.9697	4.3625	28.4167	0.0919
1-Insertions_4	67	0.75	22.8895	22.9747	9.5704	34.3205	0.1560
huck	74	0.75	28.8358	26.1618	15.8180	38.3131	0.2876
4-Insertions_3	79	0.75	16.2796	26.6190	26.0509	39.9000	0.2046
3-FullIns_3	80	0.75	28.5297	27.3851	23.0725	40.9719	0.3118
jean	80	0.75	26.5304	27.5241	23.4976	40.8676	0.2806
david	87	0.75	31.0571	30.9057	26.0520	45.2394	0.5304
mug88_1	88	0.75	14.4042	29.5524	81.8920	44.2943	0.2272
mug88_25	88	0.75	14.4042	29.5537	78.0469	44.2943	0.2954
1-FullIns_4	93	0.75	38.3942	32.2820	46.0900	48.1359	0.3815
myciel6	95	0.75	43.8257	33.7352	49.8578	49.9270	0.4262
AVERAGE			22.8711	21.1451	19.7069	31.2318	0.1804
#improved				13/20		5/20	

Table 3.3: Maximum 0.85-quasi Clique problems

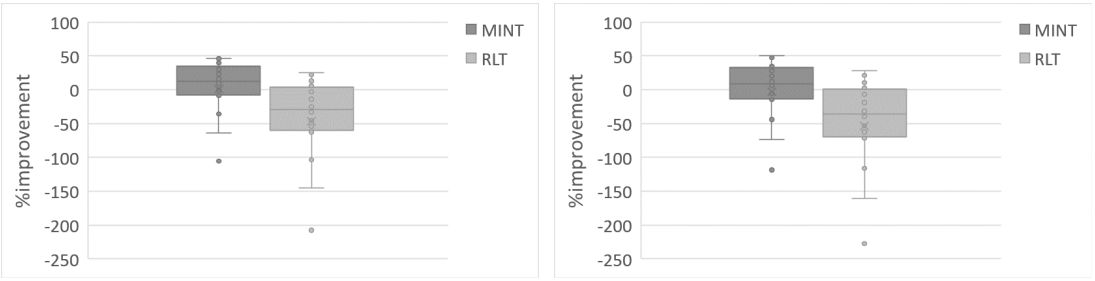
Graph	n	γ	Analytical bound	MINT		RLT	
				Bound	Time(s)	Bound	Time(s)
myciel3	11	0.85	6.5741	3.9524	0.0110	5.8529	0.0024
myciel4	23	0.85	12.4353	8.1130	0.1908	12.0637	0.0097
queen5_5	25	0.85	19.5778	9.7019	0.3518	14.0784	0.0160
1-FullIns_3	30	0.85	14.6193	10.3700	0.7340	15.4951	0.0179
queen6_6	36	0.85	26.1807	13.3466	0.8388	19.6262	0.0465
2-Insertions_3	37	0.85	10.9049	12.5000	0.8243	18.7353	0.0322
myciel5	47	0.85	22.8311	16.3717	1.7220	24.3885	0.1175
queen7_7	49	0.85	33.4179	17.6348	2.3199	26.1819	0.1027
2-FullIns_3	52	0.85	20.4754	17.6896	2.8767	26.4963	0.0985
3-Insertions_3	56	0.85	13.0725	18.8205	4.6459	28.2206	0.0953
1-Insertions_4	67	0.85	21.4514	22.6729	9.8642	33.9652	0.3005
huck	74	0.85	27.1174	25.6251	13.3873	37.8462	0.3015
4-Insertions_3	79	0.85	15.2407	26.4797	26.7070	39.7118	0.3329
3-FullIns_3	80	0.85	26.7502	27.0270	24.7422	40.5118	0.2853
jean	80	0.85	24.9519	27.1284	17.4220	40.5160	0.3128
david	87	0.85	29.1246	30.2582	25.1374	44.6539	0.7781
mug88_1	88	0.85	13.4782	29.4484	86.3345	44.1555	0.2291
mug88_25	88	0.85	13.4782	29.4491	82.2556	44.1555	0.2307
1-FullIns_4	93	0.85	36.0171	31.6429	37.1803	47.3746	0.4070
myciel6	95	0.85	41.1195	32.7548	33.6418	48.8824	0.3252
AVERAGE			21.4409	20.5494	18.5594	30.6456	0.2021
#improved				11/20		5/20	

Table 3.4: Maximum 0.95-quasi Clique problems

Graph	n	γ	Analytical bound	MINT		RLT	
				Bound	Time(s)	Bound	Time(s)
myciel3	11	0.95	6.1741	3.7407	0.0108	5.6053	0.0027
myciel4	23	0.95	11.7243	7.7862	0.2237	11.6699	0.0083
queen5_5	25	0.95	18.4825	8.7193	0.4303	13.0214	0.0168
1-FullIns_3	30	0.95	13.7911	10.1017	0.7447	15.1466	0.0183
queen6_6	36	0.95	24.7291	12.3392	0.6382	18.4793	0.0416
2-Insertions_3	37	0.95	10.2759	12.3810	0.8487	18.5702	0.0276
myciel5	47	0.95	21.5603	15.8567	1.4598	23.7700	0.1117
queen7_7	49	0.95	31.5753	16.6988	2.2095	25.0204	0.1483
2-FullIns_3	52	0.95	19.3317	17.4340	3.0864	26.1473	0.0919
3-Insertions_3	56	0.95	12.3274	18.7111	4.6405	28.0658	0.0851
1-Insertions_4	67	0.95	20.2550	22.4283	8.1067	33.6388	0.2233
huck	74	0.95	25.6781	25.0514	8.7804	37.3961	0.5917
4-Insertions_3	79	0.95	14.3790	26.3759	26.3486	39.5632	0.2461
3-FullIns_3	80	0.95	25.2678	26.7698	24.1253	40.1520	0.2893
jean	80	0.95	23.6298	26.7947	15.2333	40.1554	0.5559
david	87	0.95	27.5139	29.7964	24.7341	44.1705	0.6155
mug88_1	88	0.95	12.7113	29.3675	88.3731	44.0463	0.2572
mug88_25	88	0.95	12.7113	29.3677	82.0478	44.0463	0.3002
1-FullIns_4	93	0.95	34.0340	31.1787	37.7215	46.7603	0.3814
myciel6	95	0.95	38.8606	31.9641	27.1849	47.9227	0.3363
AVERAGE			20.2506	20.1432	17.8474	30.1674	0.2175
#improved				11/20		5/20	

Table 3.5: Maximum (1-quasi) Clique problems

Graph	n	γ	Analytical bound	MINT		RLT	
				Bound	Time(s)	Bound	Time(s)
myciel3	11	1	6.0000	3.6667	0.0047	5.5000	0.0019
myciel4	23	1	11.4121	7.6667	0.1177	11.5000	0.0026
queen5_5	25	1	18.0000	8.3333	0.0388	12.5000	0.0035
1-FullIns_3	30	1	13.4269	10.0000	0.0477	15.0000	0.0055
queen6_6	36	1	24.0887	12.0000	0.1885	18.0000	0.0051
2-Insertions_3	37	1	10.0000	12.3333	0.1257	18.5000	0.0054
myciel5	47	1	21.0000	15.6667	0.2924	23.5000	0.0091
queen7_7	49	1	30.7618	16.3333	0.3958	24.5000	0.0089
2-FullIns_3	52	1	18.8277	17.3333	0.4032	26.0000	0.0097
3-Insertions_3	56	1	12.0000	18.6667	0.4183	28.0000	0.0095
1-Insertions_4	67	1	19.7277	22.3333	0.9077	33.5000	0.0167
huck	74	1	25.0408	24.6667	1.8643	37.0000	0.0233
4-Insertions_3	79	1	14.0000	26.3333	1.5125	39.5000	0.0259
3-FullIns_3	80	1	24.6138	26.6667	1.6048	40.0000	0.0566
jean	80	1	23.0444	26.6667	2.1751	40.0000	0.0180
david	87	1	26.8032	29.3333	2.5237	43.5000	0.0468
mug88_1	88	1	12.3743	29.3333	2.2572	44.0000	0.0661
mug88_25	88	1	12.3743	29.3333	2.2735	44.0000	0.0680
1-FullIns_4	93	1	33.1583	31.0000	3.0420	46.5000	0.1012
myciel6	95	1	37.8628	31.6667	3.2122	47.5000	0.0645
AVERAGE			19.7258	19.9667	1.1703	29.9250	0.0274
#improved				11/20		4/20	



(a) box-plot on 0.75-quasi Clique problems (b) box-plot on 0.85-quasi Clique problems

Figure 3.4: Box-plots of %improvement on Quasi Clique problems

The %improvement is calculated based on Analytical bound, and we can see from Figure 3.4 that for most of instances, MINT improves much more than RLT relaxations.

3.3.2 0-1 QCQPs

We continue our computational experience by solving a class of more complicated test problems obtained from the Zheng *et al.* [71, 72] papers, 0-1 QCQP. These 0-1 QCQPs have a number of applications and have been well documented in the global optimization literature. The nomenclature for these set of problems is in the format of "set-n-s", where n is the number of variables and s indicates the serial number. There are 10 instances of problems with $n = 30$. Table 3.6 records the lower bounds obtained at the root node using the RLT relaxation, RLT + TI, and MINT cuts. Note that the MINT cuts are modeled using the indicator variables (3.5), and in the results presented in Table 3.6, only these variables are set as binary to accurately represent the minimum function relationship.

The relative improvement of the MINT cuts over existing methods is measured in Table 3.7, where the *improved ratio percentage* and *relative gap percentage* are defined as

$$\textit{improved ratio\%} = \frac{\text{MINT}_{opt} - g_{opt}}{f_{opt} - g_{opt}} \times 100 \tag{3.30}$$

Table 3.6: Root node computational results for 0-1 QCQP from Zheng *et al.* [71, 72]

Instance	f_{opt}	LB			CPU time(s)	
		RLT	RLT+TI	MINT	RLT+TI	MINT
v30-01	−37.0446	−51.7843	−50.1294	−43.9678	25.96	30.54
v30-02	−16.4241	−23.3159	−22.3901	−19.2945	25.23	27.42
v30-03	28.5732	−9.5165	−9.0301	15.0959	25.45	27.28
v30-04	10.2513	0.9300	1.5344	5.0422	25.98	28.22
v30-05	−66.6521	−76.9488	−76.0788	−71.9208	26.48	27.18
v30-06	9.2026	−28.7472	−25.7707	3.8302	26.13	27.87
v30-07	−33.2597	−52.6762	−52.6752	−39.1070	25.91	40.51
v30-08	8.6860	−5.5867	−2.6743	6.1088	25.92	33.50
v30-09	−19.5714	−43.9139	−39.2457	−20.2259	25.68	32.10
v30-10	−25.2909	−37.7290	−37.3085	−27.5455	26.19	27.64

Table 3.7: Relative gap and improved ratio percentages

Instance	relative gap(%)		improved ratio(%) over RLT+TI
	RLT+TI	MINT	MINT
v30-01	35.32	18.69	47.09
v30-02	36.32	17.48	51.89
v30-03	131.60	47.17	64.16
v30-04	85.03	50.81	40.24
v30-05	14.14	7.90	44.11
v30-06	380.04	58.38	84.64
v30-07	58.38	17.58	69.88
v30-08	130.79	29.67	77.31
v30-09	100.53	3.34	96.67
v30-10	47.52	8.91	81.24

$$\textit{relative gap\%} = \left| \frac{\mathrm{M}_{opt} - f_{opt}}{f_{opt}} \right| \times 100 \tag{3.31}$$

where, M_{opt} denotes the objective value of corresponding method, g_{opt} is the optimal value to the RLT + TI relaxation, and f_{opt} is the global optimum to the problem.

It is easy to see from Table 3.6, MINT gives the best lower bound among all these methods, which is also reflected by the *relative gap* and *improved ratio* of Table 3.7.

3.3.3 Box QPs

Having demonstrated the efficiency of the MINT cuts towards generating tight lower bounds, we next consider the effect of the proposed variable fixing and branching

Table 3.8: Strategies of branch and bound methods

	(1)RLT_rounding	(2)RLT_minta	(3)MINT_minta
LB at root node	RLT	RLT	MINT
branching node	first-best strategy is used (choose the node with smallest lower bound)		
branching variable	max element in $\lvert Y - xx^\top \rvert$	MINT algorithm fixing strategy	MINT algorithm fixing strategy
UB checking strategy	rounding	MINT algorithm fixing strategy	MINT algorithm fixing strategy

strategies on tightening the LB-UB gap at the root node, and subsequently, the number of nodes required to find the global optimum using the specialized MINT algorithm. For these computational runs, we use the set of *0-1 box QP instances* [25], which is also a standard test bed of instances available in the literature. These problem instances are nonconvex 0-1 quadratic programs subject to only binary restrictions on the decision variables; there are no structural constraints in the problem. Recognizing that any 0-1 solution is feasible to the box QP, we test three different strategies at the root node: (1) Utilize RLT relaxation to obtain a lower bound, and the lower bounding solution is *rounded* to the nearest integer to obtain a feasible solution (upper bound); (2) Utilize RLT relaxation to calculate a lower bound, and the prescribed variable fixing strategy to obtain a feasible solution (upper bound); (3) Utilize MINT relaxation to calculate a lower bound, and the prescribed variable fixing strategy to obtain a feasible solution (upper bound). We describle these three strategies more clearly in Table 3.8.

Table 3.9 lists the results obtained using these three strategies. For most of instances, MINT_minta find the optimal value at root node since the lower bound is equal to the upper bound. To compare the results, we compute the average of 'ub-lb' and 'ub-opt', which is the last row of Table 3.9. Obviously, MINT_minta $\ll$ RLT_minta $<$ RLT_rounding.

To further showcase the performance of the MINT algorithm, Table 3.10 records the number of nodes required to detect the global optimum for the 0-1 box QP problems, where the strategies listed above are utilized at each node of the branch-and-bound tree. There are five instances (colored by grey) whose optimal

Table 3.9: Computational results at the root node for 0-1 box QP instances

Instance	f_{opt}	RLT_rounding				MINT_minta				RLT_minta			
		LowerBound	UpperBound	UB-LB	UB-opt	LowerBound	UpperBound	UB-LB	ub-opt	LowerBound	UpperBound	UB-LB	UP-opt
spar020-100-1	-1,500.00	-2085	938	3023	2438	-1500	-1500	0	0	-2085	0	2085	1500
spar020-100-2	-1,729.00	-2535.5	-741	1794.5	988	-1729	-1729	0	0	-2535.5	-30	2505.5	1699
spar020-100-3	-1,609.00	-2323	-732	1591	877	-1609	-1609	0	0	-2323	-18	2305	1591
spar030-060-1	-1,369.00	-2793.5	1153	3946.5	2522	-1369	-1369	0	0	-2793.5	0	2793.5	1369
spar030-060-2	-2,663.00	-3230	-520	2710	2143	-2663	-2663	0	0	-3230	0	3230	2663
spar030-060-3	-2,512.00	-3910	-1182	2728	1330	-2512	-2512	0	0	-3910	0	3910	2512
spar030-070-1	-1,282.00	-3029.5	2033	5062.5	3315	-1316.964732	-975	341.964732	307	-3029.5	-7	3022.5	1275
spar030-070-2	-2,774.00	-3776.5	-941	2835.5	1833	-2774	-2774	0	0	-3776.5	0	3776.5	2774
spar030-070-3	-3,246.00	-4337.5	-1783	2554.5	1463	-3246	-3246	0	0	-4337.5	-53	4284.5	3193
spar030-080-1	-1,845.00	-4069.5	931	5000.5	2776	-1845	-1845	0	0	-4069.5	0	4069.5	1845
spar030-080-2	-3,351.00	-4414	-442	3972	2909	-3351	-3351	0	0	-4414	0	4414	3351
spar030-080-3	-3,400.00	-4645.5	-1001	3644.5	2399	-3400	-3400	0	0	-4645.5	0	4645.5	3400
spar030-090-1	-2,442.00	-4669	748	5417	3190	-2442	-2442	0	0	-4669	-55	4614	2387
spar030-090-2	-2,915.00	-5235	-772	4463	2143	-2915	-2915	0	0	-5235	-3	5232	2912
spar030-090-3	-2,788.00	-4904	-124	4780	2664	-2788	-2788	0	0	-4904	0	4904	2788
spar030-100-1	-2,384.00	-4959.5	631	5590.5	3015	-2384	-2384	0	0	-4959.5	-31	4928.5	2353
spar030-100-2	-2,799.00	-5492.5	-449	5043.5	2350	-2799	-2799	0	0	-5492.5	-55	5437.5	2744
spar030-100-3	-2,993.00	-5357.5	39	5396.5	3032	-2993	-2993	0	0	-5357.5	-17	5340.5	2976
spar040-030-1	-1,665.00	-2068	2610	4678	4275	-1665	-1665	0	0	-2068	-32	2036	1633
spar040-030-2	-2,879.00	-3268.5	-1949	1319.5	930	-2879	-2879	0	0	-3268.5	-452	2816.5	2427
spar040-030-3	-2,227.00	-2658.5	609	3267.5	2836	-2227	-2227	0	0	-2658.5	0	2658.5	2227
spar040-040-1	-1,642.00	-3084	2616	5700	4258	-1644.5	-1640	4.5	2	-3084	-15	3069	1627
spar040-040-2	-2,975.00	-3882	-890	2992	2085	-2975	-2975	0	0	-3882	0	3882	2975
spar040-040-3	-2,461.00	-4025.5	-237	3788.5	2224	-2461	-2461	0	0	-4025.5	0	4025.5	2461
spar040-050-1	-2,147.00	-4078	2998	7076	5145	-2147	-2147	0	0	-4078	-66	4012	2081
spar040-050-2	-2,803.00	-4688	714	5402	3517	-2803	-2803	0	0	-4688	0	4688	2803
spar040-050-3	-3,264.00	-5155.5	-549	4606.5	2715	-3264	-3264	0	0	-5155.5	-42	5113.5	3222
spar040-060-1	-2,550.00	-5479.5	1189	6668.5	3739	-2580.11549	-2305	275.11549	245	-5479.5	0	5479.5	2550
spar040-060-2	-3,863.00	-5666.5	-1043	4623.5	2820	-3863	-3863	0	0	-5666.5	-33	5633.5	3830
spar040-060-3	-4,750.00	-6538.5	-1517	5021.5	3233	-4750	-4750	0	0	-6538.5	-83	6455.5	4667
spar040-070-1	-3,173.00	-6028	1868	7896	5041	-3173	-3173	0	0	-6028	0	6028	3173
spar040-070-2	-3,568.00	-6699.5	103	6802.5	3671	-3568	-3568	0	0	-6699.5	-17	6682.5	3551
spar040-070-3	-4,625.00	-7278.5	-2259	5019.5	2366	-4625	-4625	0	0	-7278.5	0	7278.5	4625
spar040-080-1	-3,607.00	-7493.5	259	7752.5	3866	-3607	-3607	0	0	-7493.5	0	7493.5	3607
spar040-080-2	-3,951.00	-7636	-242	7394	3709	-3951	-3951	0	0	-7636	0	7636	3951
spar040-080-3	-4,939.00	-8414	-2032	6382	2907	-4939	-4939	0	0	-8414	-22	8392	4917
spar040-090-1	-4,204.00	-8600.5	37	8637.5	4241	-4204	-4204	0	0	-8600.5	-6	8594.5	4198
spar040-090-2	-4,365.00	-8577.5	763	9340.5	5128	-4365	-4365	0	0	-8577.5	-53	8524.5	4312
spar040-090-3	-4,882.00	-8754	-1012	7742	3870	-4882	-4882	0	0	-8754	-49	8705	4833
spar040-100-1	-4,969.00	-9733.5	-1311	8422.5	3658	-4969	-4969	0	0	-9733.5	0	9733.5	4969
spar040-100-2	-4,188.00	-9727.5	273	10000.5	4461	-4255.666667	-2990	1265.666667	1198	-9727.5	-31	9696.5	4157
spar040-100-3	-3,527.00	-9914	-748	9166	2779	-4171.666667	-2696	1475.666667	831	-9914	0	9914	3527
spar050-030-1	-2,600.00	-3625.5	3857	7482.5	6457	-2600	-2600	0	0	-3625.5	-35	3590.5	2565
spar050-030-2	-3,370.00	-4702	-406	4296	2964	-3370	-3370	0	0	-4702	0	4702	3370
spar050-030-3	-3,043.00	-4247	200	4447	3243	-3043	-3043	0	0	-4247	0	4247	3043
spar050-040-1	-2,766.00	-5028.5	3441	8469.5	6207	-2766	-2766	0	0	-5028.5	0	5028.5	2766
spar050-040-2	-3,651.00	-5882.5	-7	5875.5	3644	-3651	-3651	0	0	-5882.5	0	5882.5	3651
spar050-040-3	-4,164.00	-6420	-1110	5310	3054	-4164	-4164	0	0	-6420	-46	6374	4118
AVERAGE				5315.26	3313.96			70.06	53.81			5205.64	3024.33

Table 3.10: Computational performance of three strategies to find the global optimum for 0-1 box QPs

Instance	# nodes			CPU time(s)		
	BBRLT_rounding	BBRLT_minta	BBMINT_minta	BBRLT_rounding	BBRLT_minta	BBMINT_minta
spar020-100-1	19	21	1	0.32	0.72	0.05
spar020-100-2	43	45	1	0.53	1.43	0.10
spar020-100-3	41	31	1	0.51	0.98	0.06
spar030-060-1	971	1153	1	15.70	29.02	0.98
spar030-060-2	21	21	1	0.55	0.64	0.77
spar030-060-3	333	227	1	5.82	6.82	0.96
spar030-070-1	1565	4055	5	23.84	91.16	4.99
spar030-070-2	85	85	1	1.99	2.43	0.50
spar030-070-3	79	83	1	1.81	1.96	0.60
spar030-080-1	2321	2011	1	35.75	72.09	1.17
spar030-080-2	63	47	1	1.49	2.63	0.53
spar030-080-3	103	119	1	2.06	4.99	0.56
spar030-090-1	731	865	1	12.69	37.17	0.64
spar030-090-2	999	649	1	17.65	17.94	0.69
spar030-090-3	593	711	1	10.39	18.11	0.62
spar030-100-1	1745	1629	1	27.94	40.15	1.11
spar030-100-2	1361	1681	1	22.96	42.20	1.06
spar030-100-3	1005	553	1	16.74	14.96	0.76
spar040-030-1	21	23	1	0.82	1.30	1.69
spar040-030-2	43	39	1	1.72	2.35	3.18
spar040-030-3	51	77	1	1.65	3.94	2.94
spar040-040-1	1929	1843	3	50.26	65.55	6.79
spar040-040-2	213	211	1	10.05	10.35	3.38
spar040-040-3	873	1371	1	30.68	54.37	3.15
spar040-050-1	2897	2175	1	73.13	114.47	2.81
spar040-050-2	4799	775	1	127.01	71.62	3.23
spar040-050-3	1037	1209	1	30.56	66.68	3.61
spar040-060-1	35561	11285	3	870.32	527.25	6.93
spar040-060-2	831	599	1	25.94	43.63	3.57
spar040-060-3	167	517	1	6.23	45.62	3.64
spar040-070-1	7005	4825	1	187.31	246.40	3.34
spar040-070-2	9353	3259	1	247.86	178.55	3.09
spar040-070-3	1463	3015	1	46.98	169.16	4.24
spar040-080-1	16531	13931	1	466.50	689.09	3.18
spar040-080-2	10171	7117	1	309.36	387.19	3.25
spar040-080-3	3929	3963	1	126.88	203.36	3.41
spar040-090-1	17603	12727	1	497.68	589.73	2.98
spar040-090-2	6041	10141	1	181.44	482.40	3.07
spar040-090-3	5745	3673	1	172.29	188.20	3.66
spar040-100-1	16947	8401	1	550.26	314.69	3.32
spar040-100-2	50951	48245	5	1451.51	926.15	11.90
spar040-100-3	517145	307217	83	13602.03	5736.15	141.97
spar050-030-1	191	301	1	13.52	9.83	6.28
spar050-030-2	1211	2751	1	64.78	89.36	7.52
spar050-030-3	465	1203	1	22.35	39.83	8.18
spar050-040-1	18495	5217	1	775.01	151.00	6.64
spar050-040-2	15267	8593	1	652.78	296.94	9.10
spar050-040-3	2015	3863	1	99.31	136.73	9.88

solutions are not got at root node by BBMINT_minta. We can see except the first one (spar030-070-1), RLT_minta reduce a lot number of nodes comparing with RLT_rounding. Furthermore, BBMINT_minta dramatically reduce the number of

nodes, which leads to much less CPU time to get the optimal solution. For those instances whose optimal solutions are got at root node by BBMINT_minta, the MINT cuts and our variable fixing strategy leads to these great results. Even comparing RLT_rounding and RLT_minta, we could conclude that the average number of nodes of the latter is less than the former.

3.4 Conclusions

In this chapter, we enhance linearization techniques for solving 0-1 QCQPs, thereby leading to improved lower bound and a novel variable fixing and affiliated branching strategy in a branch-and-bound procedure. Specifically, we present a novel and simple relaxation scheme to generate proper lower bound for 0-1 QCQP. In this scheme, we develop a new class class of valid inequalities (referred to as MINT inequalities). Theoretically, we demonstrate the tightness of MINT inequalities. Numerical experiences show that our method is superior to quite a few existing methods. With using the logic in defining MINT inequalities, the newly designed variable fixing and branching strategy gives rise to a branch-and-bound algorithm called MINT algorithm. We demonstrate the efficacy of this algorithm in finding a feasible solution (and subsequently, the optimal solution) for standard QP test instances obtained from the literature. Experimental results are given to illustrate that the number of nodes required to obtain the (global) optimum is greatly reduced, and overall, the MINT algorithm performs exceedingly well in comparison with existing methods.

Besides using the methods on pure 0-1 QCQPs, we also can extend them on an more complicated class of problems, 0-1 mixed integer QCQPs. Assume all variables are boundary, and without loss of generality, we suppose all continuous variables are in region [0, 1]. Noting that for $x_i \in \{0,1\}, x_j \in [0,1]$, $y_{ij} = x_i x_j = \min\{x_i, x_j\}$ stands true. For order $d = 3$, when there are at least two binary variables in x_i, x_j,

and x_k, $y_{pq} = \min\{x_p, x_q\}$ is satisfied for any pair $\{p, q\} \in \{i, j, k\}$. then all MINT inequalities we discussed still stand true. Specifically, for MINT-4, it holds true if only x_j is binary. Following this fashion, the MINT inequalities could be utilized in 0-1 mixed integer QCQPs. Furthermore, the method we propose also can be extended towards 0-1 multilinear programs and the more general class of polynomial programs. As polynomial programs of higher degree have proven to be very difficult to solve in practice, a carefully designed research framework must be employed when extending such techniques to a more general class of problems.

4 Enhancing RLT with A Class of Higher-Order Semidefinite Cutting Planes

Besides considering to add MINT cuts to the RLT relaxations, we think about another direction-SDP cuts. In this chapter, we mainly focus on adding SDP cuts to the RLT relaxations to reduce more gap.

In Chapter 2, we have introduced a notable relaxation technique, RLT, proposed by Adams and Sherali [1], which drive a higher-dimensional lower bounding Linear Programming. The lowest-level RLT relaxation is most frequently implemented due to the problem size. After applying this relaxation technique on QCQP (1.1), we get its RLT relaxation's formulation:

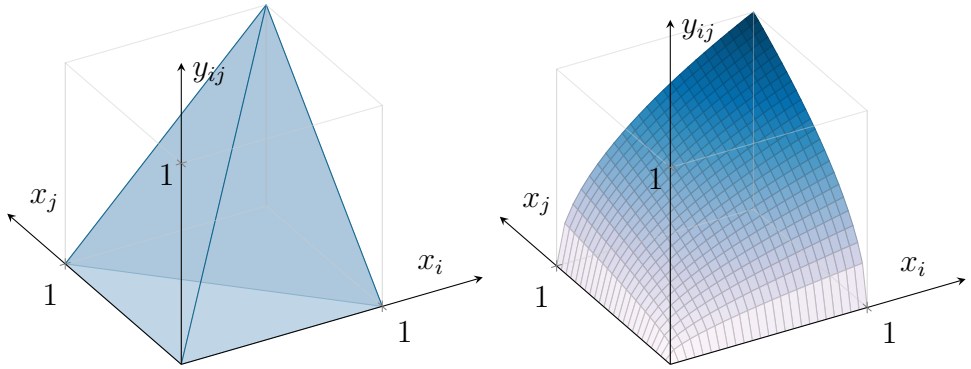
$$
\begin{aligned}
\text{QCQP}_{\text{RLT}}: \quad \min \quad & f_0(x) = c_0^\top x + Q_0 \bullet X && \text{(4.1a)}\\
s.t. \quad & f_k(x) = c_k^\top x + Q_k \bullet X + d_k \le 0, \quad k \in \mathcal{K} && \text{(4.1b)}\\
& g_l(x) = a_l^\top x + b_l \le 0, \quad l \in \mathcal{L} && \text{(4.1c)}\\
& \text{RLT constraints}, && \text{(4.1d)}
\end{aligned}
$$

where,

$$
X = [xx^\top]_L = \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{12} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} & y_{2n} & \cdots & y_{nn} \end{pmatrix}. \tag{4.2}
$$

Noting that without loss of generality, here we suppose $0 \le x \le 1$. Variables substituaiton $\dfrac{x - l_x}{u_x - l_x}$ can be used if $l_x \le x \le u_x$.

Starting from [48], the SDP relaxation for combinatorial problems has become a



(a) RLT (b) SDP

Figure 4.1: Illustration of RLT and SDP relaxations for two variables x_i and x_j

focus of interest for many researchers. This technique sometimes leads to much tighter bounds than the linear relaxation. Let X denote $xx^\top$, then X should be a positive semidefinite (PSD) matrix. By substituting $X = xx^\top$, a SDP relaxation to the QCQP problem is given by:

$$\text{QCQP}_{\text{SDP}} \quad \min \quad f_0(x) = c_0^\top x + Q_0 \bullet X \tag{4.3a}$$

$$s.t. \quad f_k(x) = c_k^\top x + Q_k \bullet X + d_k \leq 0, \quad k \in \mathcal{K} \tag{4.3b}$$

$$g_l(x) = a_l^\top x + b_l \leq 0, \quad l \in \mathcal{L} \tag{4.3c}$$

$$X \succeq 0. \tag{4.3d}$$

Figure 4.1 gives an intuitive image of RLT and SDP relaxations in the situation of 2 variables x_i and x_j.

A further tighten way is to combine SDP relaxation with the RLT relaxation. In this case, an RLT+SDP relaxation of QCQP could be expressed as follows:

$$\text{QCQP}_{\text{RLT+SDP}}: \quad \min \quad f_0(x) = c_0^\top x + Q_0 \bullet X \tag{4.4a}$$

$$s.t. \quad f_k(x) = c_k^\top x + Q_k \bullet X + d_k \leq 0, \quad k \in \mathcal{K} \tag{4.4b}$$

$$g_l(x) = a_l^\top x + b_l \leq 0, \quad l \in \mathcal{L} \tag{4.4c}$$

$$\text{RLT constraints}, \tag{4.4d}$$

$$X \succeq 0. \tag{4.4e}$$

Rather than solving this SDP problem directly, we try to figure some other way to deal with it.

4.1 An alternative semi-infinite representation for SDP relaxation of QCQP

First, we introduce an alternative equivalent SILP restatement of the Problem $\text{QCQP}_{\text{RLT+SDP}}$ (4.4). Before introducing the equivalent formulation, consider the following result.

Property 1. *The positive semidefinite condition $X \succeq 0$ holds if and only if $H \bullet X \geq 0$ for $\forall H \in \mathbb{S}_+^n$.[37]*

Now we have the following conclusion.

Property 2. *The Problem $QCQP_{RLT+SDP}$ given by (4.4) is equvalent to the simi-infinite linear program stated in (4.5) below.*

$$\text{QCQP}_{\text{RLT+SILP-H}}: \quad \min \quad f_0(x) = c_0^\top x + Q_0 \bullet X \tag{4.5a}$$

$$s.t. \quad f_k(x) = c_k^\top x + Q_k \bullet X + d_k \leq 0, \quad i \in \mathcal{K} \tag{4.5b}$$

$$g_l(x) = a_l^\top x + b_l \leq 0, \quad l \in \mathcal{L} \tag{4.5c}$$

$$\text{RLT constraints}, \tag{4.5d}$$

$$H \bullet X \geq 0, \quad \forall\, H \in \mathbb{S}_n^+ \,. \tag{4.5e}$$

Proof. Obviously from Property 1. □

In the following section, we will afford an efficient process of generating valid cutting planes for level-1 RLT relaxation with polynomial time complexity. Note that different variants of the termination criterion, for the semidefinite cut generation scheme, can be employed. Furthermore, embedded into branch-and-bound algorithm, this cut generation procedure produces a significant tightening of the lower bound, obtained by using the RLT as a stand-alone approach, thereby reducing the effort required to obtain globally optimal solutions.

4.2 A new way of generating SDP cuts

Rather than solve $\text{QCQP}_{\text{RLT+SILP-H}}$ (4.5) directly, we adopt a relaxation approach which leads to a cutting plane generation scheme.

To begin with, we solve Problem (4.5) without constraints (4.5e). We denote the solution of current relaxation problem as $\overline{X}$. If $\overline{X}$ is PSD, then it solves 0-1 $\text{QCQP}_{\text{RLT+SDP}}$; otherwise, there exists $H \in \mathbb{S}_+^n$ such that $H \bullet \overline{X} < 0$ according to Property 1. In that case, we are thinking about generating such kind of PSD matrix H such that $H \bullet \overline{X} < 0$ and appending cutting plane $H \bullet X \geq 0$ into current relaxation.

Starting with row 1, we perform elementary row and column operations to zero out all the corresponding non-diagonal elements on $\overline{X}$. Assume the row operations are $P_1, \cdots, P_r$. Denote the current matrix as Y. We have the following property:

Property 3. *Given an symmetric matrix $D \in \mathbb{R}^{n\times n}$, there exists*

$$H = P_1^\top P_2^\top \cdots P_r^\top D P_r P_{r-1} \cdots P_1,$$

such that Tr(DY) = Tr$(H\overline{X})$.

Proof. The row operations are $P_1, \cdots, P_r$, to zero out the corresponding non-diagonal

elements the corresponding column operation becomes $P_1^\top, \cdots, P_r^\top$,

$$\begin{aligned}
&\text{Tr}(DY) \\
=&\text{Tr}(DP_rP_{r-1}\cdots P_1\,\overline{X}P_1^\top P_2^\top\cdots P_r^\top) \\
=&\text{Tr}(\overline{X}P_1^\top P_2^\top\cdots P_r^\top DP_rP_{r-1}\cdots P_1) \\
=&\text{Tr}(\overline{X}H) \\
=&\text{Tr}(H\overline{X})\ (\overline{X} \text{ and } H \text{ are symmetric})
\end{aligned}$$

□

Then the problem reduce to an easier one – to construct a matrix D to satisfy $\text{Tr}(DY) < 0$. To design new classes of cutting planes, we try to develop a polynomial time semidefinite cutting plane approach for improving the resulting relaxations. This idea comes from [63], which proposed a class of SDP cut by generating an α vector. In this section, the focus is on generating a symmetric H-matrix (of appropriate size) directly to construct SDP cuts.

In the process of doing operations on the matrix X, we consider four cases as below:

Case 1: a diagonal element is positive;

Case 2: a diagonal element is negative;

Case 3: a diagonal element is zero, and all elements along that row are zero;

Case 4: a diagonal element is zero, and there exists a nonzero element along the row.

The following is the details of this algorithm:

For notational simplicity, we use $r_i(j,k)$ to denote the elementary row transformation that is adding k multiple of ith row to jth row. Also, $c_i(j,k)$ is written

for the elementary column transformation that is adding k multiple of ith column to jth column.

Step 1: Let $r = 1$. Find a minimum of the diagonal elements and we denote by $\bar{k}$ the corresponding row number.

(a) If $X_{\bar{k}\bar{k}} < 0$, set $H = \text{Diag}(0, \ldots, 0, 1, 0, \ldots, 0)$ and stop the algorithm.

(b) If $X_{\bar{k}\bar{k}} > 0$, go to step 2.

(c) Otherwise ($X_{\bar{k}\bar{k}} = 0$), move to step 3.

Step 2: Let $i = 1$ and $j = i + 1$.

While $X_{ii} > 0$

(1) Do $r_i(j, -\dfrac{X_{ji}}{X_{ii}})$, get a new matrix Y and mark the operation by an elementary matrix P_r. Set $r = r + 1$.

(2) If $Y_{jj} < 0$, set $D = \text{diag}(0, \ldots, 0, 1, \ldots, 0)$. Exit the loop and go to step 4.

(3) If $Y_{jj} = 0$, find the index k such that $|Y_{jk}| = \max |Y_{j\bullet}|$.

a) If $Y_{jk} \neq 0$, set$D_{jj} = 1, \quad D_{kj} = D_{jk} = -\dfrac{Y_{jk}}{|Y_{kk}|}, \quad D_{kk} = D_{kj}^2$ and 0 for other entries. Then exit the loop and go to step 4.

b) If $Y_{jk} = 0$, delete the j-th row and j-th column of Y. Let P_r to denote the matrix associated with row deletion operation. Set $r = r + 1, j = j - 1, n = n - 1$ and set X as the new obtained matrix.

(4) Do $c_i(j, -\dfrac{X_{ij}}{X_{ii}})$ and set X as the new obtained matrix.

(5) If $j < n$, set $j = j + 1$, else set $i = i + 1, j = i + 1$.

Step 3: Find the index set $I = \{i \,|\, X_{ii} = 0\}$ and find k, j such that $|X_{kj}| = \max_{i \in I, l > i} |X_{il}|$

(1) If $X_{kj} \neq 0, j \in I$, set $H_{kk} = H_{jj} = 1, H_{kj} = H_{jk} = -\text{sgn}(X_{kj})$ and 0 for other elements. Stop the algorithm.

(2) If $X_{kj} \neq 0, j \notin I$, set $H_{kk} = 1, \quad H_{kj} = H_{jk} = -\dfrac{X_{kj}}{X_{jj}}, \quad H_{jj} = H_{kj}^2$.

(3) If $X_{kj} = 0$, delete all the rows and columns whose numbers are in I and use P_r to denote the row deletion operation. Set $r = r + 1$ and proceed to step 2.

Step 4: Select H as $H = P_1^\top \dots P_r^\top D P_r \dots P_1$.

In our implementation, H is computed by using operation associated with the matrix $P_1, \cdots, P_r$ instead of doing matrix multiplication.

More intuitively, the flowchart of SDP cut generation scheme is described in Figure 4.2, where for different cases, matrix D is set as follows:

Case 1. $\begin{pmatrix} \diagdown & \mathbf{0} \\ \hline & a \\ \mathbf{0} & \end{pmatrix}$, $D_1 = \begin{pmatrix} 0 & \mathbf{0} \\ \hline & 1 \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$, where $a = \overline{X}_{jj}$.

In this case, we can get that $H \bullet \overline{X} = D \bullet Y = a < 0$, which show the correctness of our construction;

Case 2. $\begin{pmatrix} \diagdown & \mathbf{0} \\ \hline & a \quad b \\ \mathbf{0} & b \quad c \neq 0 \end{pmatrix}$, $D_2 = \begin{pmatrix} 0 & \mathbf{0} \\ \hline & 1 \quad \text{-}b/\lvert c\rvert \\ \mathbf{0} & -b/\lvert c\rvert \quad (b/c)^2 \end{pmatrix}$, where $b = \overline{X}_{jk}$ and $c = \overline{X}_{kk}$.

Hence, $H \bullet \overline{X} = D \bullet Y = -\dfrac{b^2}{|c|} < 0$, so the correctness of construction is verified;

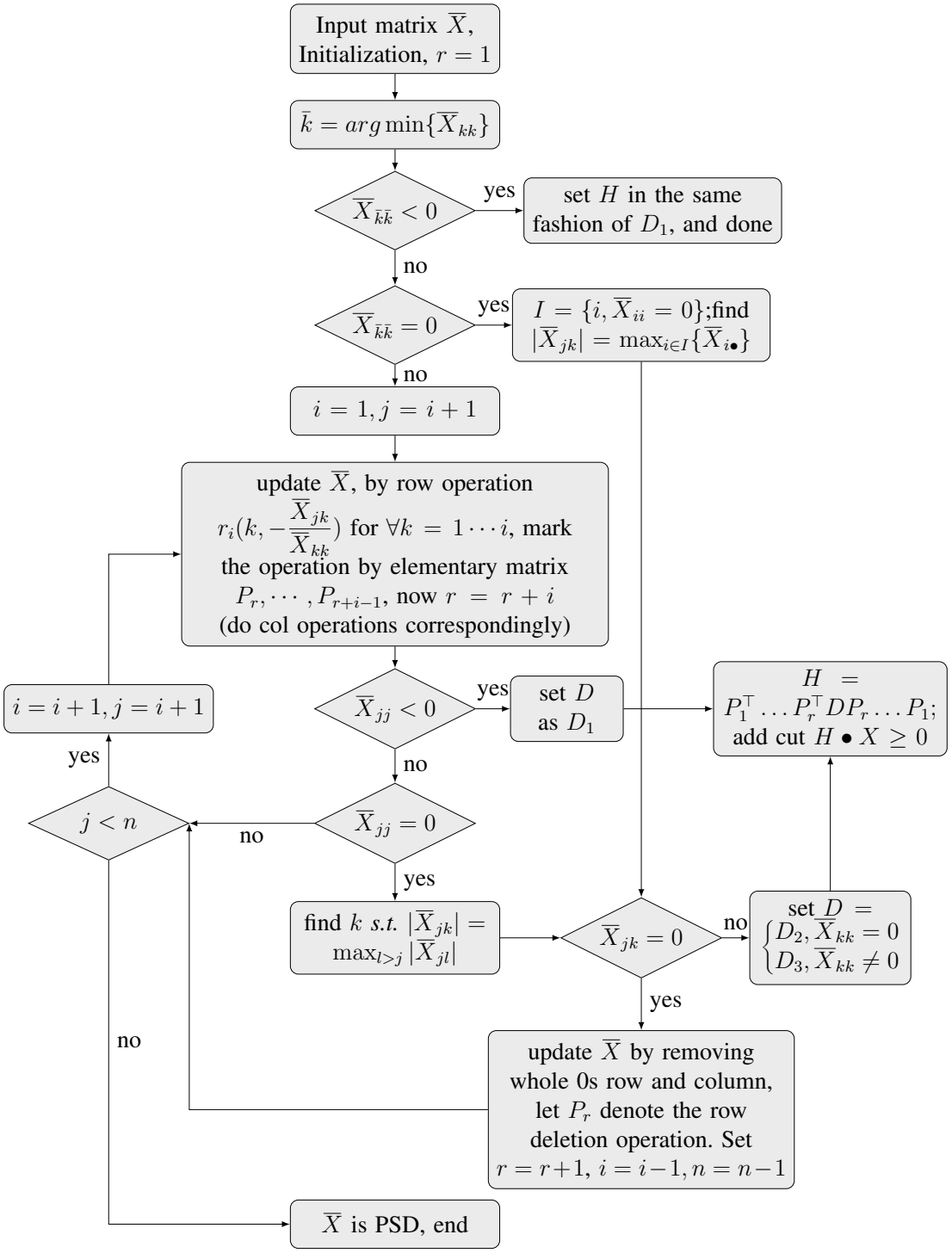


Figure 4.2: Flow chart of generating SDP cuts scheme

Case 3. $\left(\begin{array}{c|cc} \diagdown & \multicolumn{2}{c}{\mathbf{0}} \\ \hline & a & b \\ \mathbf{0} & b & 0 \end{array}\right)$, $D_3 = \left(\begin{array}{c|cc} 0 & \multicolumn{2}{c}{\mathbf{0}} \\ \hline & 1 & \text{-sgn}(b) \\ \mathbf{0} & \text{-sgn}(b) & 1 \end{array}\right)$.

So we have $H \bullet \overline{X} = D \bullet Y = -2|b| < 0$ and the positive semidefinite matrix H is found.

In the previous construction, we have show the correctness of the construction. Here gives a very simple example to illustrate this procedure:

Example 2. $A = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.1 \\ 0.2 & 0.3 & 0.2 & 0.3 \\ 0.3 & 0.2 & 0.4 & 0.2 \\ 0.1 & 0.3 & 0.2 & 0.5 \end{bmatrix}$. *Find an matrix* H *such that trace*$(HA) < 0$ *if* A *is not PSD.*

Following the algorithm before, we have $A = \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.1 \\ 0.2 & 0.3 & 0.2 & 0.3 \\ 0.3 & 0.2 & 0.4 & 0.2 \\ 0.1 & 0.3 & 0.2 & 0.5 \end{bmatrix}$

$$\xrightarrow{\text{do } r_1(2,-2),\, P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.1 \\ 0 & \textcircled{-0.1} & -0.4 & 0.1 \\ 0.3 & 0.2 & 0.4 & 0.2 \\ 0.1 & 0.3 & 0.2 & 0.5 \end{bmatrix}.$$

$$\text{Let } D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{ then } H = P_1^\top D P_1 = \begin{bmatrix} 4 & -2 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

H is a PSD matrix we want, and $H \bullet A = -0.1 < 0$.

Remark 3. There are some variations for the proposed scheme. For example, in Step 1, all negative diagonal elements are considered; In Step 2 and 3, we consider all the k such that $Y_{jk} \neq 0$ rather than the k satisfying $|Y_{jk}| = \max |Y_{j\bullet}|$. In this situation, muti-matrix SDP-cuts can be added, which include the one added in the proposed scheme.This variation gives tighter lower bound theoretically. However, there might increase the computation time since many cuts might be added. Hence, we also consider the variation of summating the multiple cuts obtained to get single cut a single cut, which is related to a higher order coefficient matrix. We also test this variations' computational performance.

4.3 Generating SDP cuts through an augmented matrix

The SDP cuts proposed in Section 4.2 is derived from the SDP constraint $X \succeq 0$. Another more strictly way is to recognize $X \succeq xx^\top$, which can be equivalently expressed to $\begin{pmatrix} X & x^\top \\ x & 1 \end{pmatrix} \succeq 0$ or $\begin{pmatrix} 1 & x \\ x^\top & X \end{pmatrix} \succeq 0$ by using Schurs complement. Hence, a further strengthening of Problem (4.3) can be obtained by replacing the constraint $X \succeq 0$ with $\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0$. This yields the following SDP relaxation:

$$\text{QCQP}_{\text{SDP2}}: \quad \min \quad \begin{pmatrix} 0 & \frac{1}{2}c_0^\top \\ \frac{1}{2}c_0 & Q_0 \end{pmatrix} \bullet \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \tag{4.7a}$$

$$s.t. \quad \begin{pmatrix} d_k & \frac{1}{2}c_k^\top \\ \frac{1}{2}c_k & Q_k \end{pmatrix} \bullet \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \leq 0, \quad k \in \mathcal{K} \tag{4.7b}$$

$$\begin{pmatrix} b_l & \frac{1}{2}a_l^\top \\ \frac{1}{2}a_l & O \end{pmatrix} \bullet \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \leq 0, \quad l \in \mathcal{L} \tag{4.7c}$$

$$\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0, \tag{4.7d}$$

where the terms in the objective function and constraints have been intelligently reorganized to reflect the new lifted constraint (4.7d).

We could generate the SDP cuts based on the augmented matrix $\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0$. One point we need to mention is that $\exists\ H$ is PSD, s.t $H \bullet X < 0 \Rightarrow \exists H' = \begin{pmatrix} 0 & 0 \\ 0 & H \end{pmatrix}$ s.t. $H' \bullet \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} < 0$. Even we can find such an H' for the augmented matrix. But in this case, it shows the same tightness with generating SDP cuts based on X, and the latter one consumes less computational time. Hence, when considering augmented matrix, we are trying to find some new cuts based on the augmented matrix rather than construct it based on the results from X.

4.4 Computational results

In this section, we give results of some computational experiments to show the performance of the proposed SDP cut generation method. All of the numerical experiments were performed on a computer equipped with CPU 3.70GHz and RAM 32.0GB. The underlying linear programming relaxations were solved using Gurobi [33].

4.4.1 Illustration example

To illustrate the performance of the new SDP cut generation, firstly, we observe the following pair of examples [42]:

Example 3. ***P1****:* $\min x - x^2$ *s.t.* $0 \leq x \leq 1$ *and* ***P2****:* $\min x^2 - x$ *s.t.* $0 \leq x \leq 1$. *These two examples are continuous and one-dimentional problems.*

Substitute $y = x^2$, and add the RLT constraints for P1 and P2 respectively. The RLT relaxations are stated below respectively:

$$\textbf{P1-RLT:} \quad \begin{array}{ll} \min & x - y \\ s.t. & 0 \leq x \leq 1 \\ & y - 2x + 1 \geq 0 \\ & y \leq x \\ & y \geq 0, \end{array} \qquad \textbf{P2-RLT:} \quad \begin{array}{ll} \min & y - x \\ s.t. & 0 \leq x \leq 1 \\ & y - 2x + 1 \geq 0 \\ & y \leq x \\ & y \geq 0. \end{array}$$

P1-RLT's objective value is 1 with optimal solution $(x, y) = (0, 0)$, which therefore is an optimum solution of P1. P2-RLT's solution $(x, y) = (\frac{1}{2}, 0)$ with objective value $-\frac{1}{2}$. However, P2's real optimum solution is $x = \frac{1}{2}$ with objective value $-\frac{1}{4}$.

Imposing SDP constraint to P2, we obtain the corresponding RLT+SDP relaxation, as given below:

$$\begin{array}{ll} \min & y - x \\ s.t. & 0 \leq x \leq 1 \\ & y - 2x + 1 \geq 0 \\ & y \leq x \\ & y \geq 0 \\ & \begin{pmatrix} y & x \\ x & 1 \end{pmatrix} \succeq 0. \end{array}$$

According to P2-RLT's solution, $\begin{pmatrix} y & x \\ x & 1 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{pmatrix}$, we construct one SDP cut using different methods respectively based on this matrix.

The SDP cut generated by Sherali and Fraticelli [63]'s is as follows (For convenience, we denote this method as SDP-α in the following paper): $0.8536y - 0.7071x + 0.1464 \geq 0$.

And SDP cut generated by our scheme, named as SDP-H, is given below:

$$y - x + \frac{1}{4} \geq 0.$$

After adding these SDP cuts to P2-RLT and then solving them separately, we get the result showing by Table 4.1. We can see that the objective value reaches the real optimum value with single-cut added in the case of implementing SDP-H.

We also gives a few variations of our proposed generation scheme. One is multi-H, in which we add multiple cuts simultaneously in one iteration step. Another is higher-H, in which the rank(H) ≥ 1. Hence, the cut appended will effect more x variables in this case. We illustrate the details of different strategies by implementing on the following example:

Example 4.

$$\begin{aligned} \min \quad & f(x) = 163x_1^2 - 92x_1x_2 + 565x_1x_3 + 77x_1x_4 - 6x_1x_5 + 55x_2^2 + 5984x_2x_3 - 22x_2x_4 \\ & \qquad +31x_2x_5 + 55x_3^2 + 22x_3x_4 + 2543x_3x_5 - 2x_4^2 - 76x_4x_5 + 10x_5^2 \\ s.t. \quad & -2x_1^2 - 4x_1x_2 - 2x_2^2 + 8x_1 + 6x_2 + x_3 - 4x_4 \leq -2.5 \\ & x_1 - 2x_2 + x_3 + x_4 + x_5 \leq 1 \\ & x \in [0,1]. \end{aligned}$$

Table 4.1: Objective value with different relaxations

f_{opt}	RLT	SDP-α	SDP-H
-0.25	0.5	0.2929	-0.25

Table 4.2: Comparison of different cut generation strategies

iteration	SDP-α	SDP-H	multi-H	higher-H
0	-45.5	-45.5	-45.5	-45.5
1	-44.735923	-43	-42.725	-43.347222
2	-44.717548	-42.725	-42.722222	-39.591967
3	-44.717091	-42.722222	-42.722194	-38.628009
4	-44.717091	-42.722194	-42.3724	-38.349405
5	-43.825628	-42.3724	-40.846336	-38.347588
6	-41.637739	-40.846336	-38.605577	-38.313089
7	-41.625425	-38.605577	-38.389473	-38.268846
8	-41.619332	-38.389473	-38.285906	-38.267849
9	-41.617788	-38.285906	-38.274485	-38.267384
10	-41.617291	-38.274485	-38.267895	-38.267152
11	-41.617288	-38.267895	-38.267352	-38.267021
12	-41.617286	-38.267352	-38.267088	-38.266978
13	-41.617286	-38.267088	-38.266963	-38.266963
14	-41.617285	-38.266963	-38.266962	-38.266961
15	-41.617285	-38.266962	-38.266961	-38.266961
16	-41.617285	-38.266961	-38.266961	-38.26696
17	-41.617285	-38.266961	-38.26696	X
18	-41.617285	-38.26696	-38.26696	X
19	-41.617285	-38.26696	X	X
20	-41.617285	X	X	X

Table 4.2 gives the results of different strategies for Example 4. The first column shows the number of iteration times or the number of cut added times, and 0 means the lower bound without any cut added. We can see that $\overline{X}$ satisfies semidefiniteness condition after 16 iterations for higher-H, 18 iterations for multi-H, and 19 iterations for SDP-H. However, it doesn't give a solution with semidefiniteness satisfied for the SDP-α generation method, even after 50 iterations.

4.4.2 Comparison of CPU time

In our implementation, we perform row and column operations, instead of matrix multiplications to speed up the computational time. To show the performance of the

cut generation scheme, we compare the CPU time of following different approaches. Sherali and Fraticelli [63]'s cut generation method, SDP-α; the proposed method in Section 4.2, SDP-H; SDP-λ is the eigen vector method Shor [67].

The input matrices are non-SDP and generated randomly. There are 25 matrices for each matrix size n, which is from $n = 20$ to $n = 250$ with step 10.

In Table 4.3, Column 2-4 give the computation time of different methods generating SDP cuts. The numbers are the summation of all 25 instances' time with same size. And the last column shows the percentage improvement that our proposed method over Sherali's. We can see that with matrix size-n's increasing, our method could generate much faster than Sherali's. Figure 4.3 illustrates the comparison of CPU time intuitively. x-axis is the matrix size and y-axis is the sum of all 25 instances computational time. Even the eigenvector method gives the best cut without considering the CPU time. However, with the increasing of problem size, eigenvector method's CPU time increase dramatically, which leads us to not consider this method especially when solving large-size problem. When we remove the eigenvector method and compare the remained two (Figure 4.3b), it clearly shows that the proposed method SDP-H generates SDP cuts more efficiently, which performs much better than SDP-α.

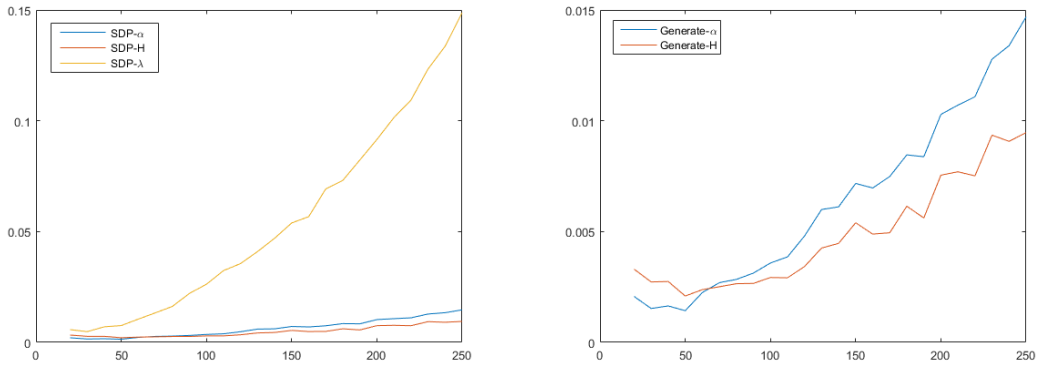
4.4.3 QCQP problems

We test 20 QCQP examples (from [71] and assume these decision variables continuous) to show the performance of our method. The gap and CPU time are the factors we consider. All linear programming are solved by groubi with MATLAB interface.

Table 4.4 gives the lower bound of different methods. Columns 2-5 give the lower bounds (rounded up to the forth decimal point) generated by linear relaxations SDP-α, SDP-H, and its variations mentioned at the end of Section 4.2 multi-H,

Table 4.3: CPU time of generating SDP cuts

Matrix size $n \times n$	SDP-α	SDP-H	SDP-λ	%improvement($\frac{\alpha - H}{H}$)
20	0.002072	0.003303	0.005760	-37.26
30	0.001528	0.002727	0.004838	-43.97
40	0.001645	0.002747	0.007022	-40.11
50	0.001429	0.002092	0.007582	-31.69
60	0.002246	0.002379	0.010497	-5.56
70	0.002694	0.002511	0.013315	7.28
80	0.002842	0.002647	0.016221	7.38
90	0.003127	0.002661	0.022090	17.48
100	0.003582	0.002927	0.026232	22.4
110	0.003858	0.002914	0.032383	32.39
120	0.004801	0.003423	0.035564	40.26
130	0.005995	0.004256	0.040992	40.84
140	0.006117	0.004469	0.046993	36.89
150	0.007172	0.005400	0.053876	32.82
160	0.006967	0.004885	0.056727	42.62
170	0.007486	0.004947	0.069276	51.33
180	0.008461	0.006143	0.073088	37.75
190	0.008377	0.005614	0.082294	49.21
200	0.010286	0.007544	0.091544	36.34
210	0.010708	0.007697	0.101450	39.1
220	0.011083	0.007513	0.109330	47.52
230	0.012780	0.009352	0.123362	36.66
240	0.013389	0.009070	0.133576	47.61
250	0.014686	0.009465	0.148745	55.17



(a) (b)

Figure 4.3: Comparison of CPU time

higher-H. Columns 6-9 is the corresponding lower bound with adding cuts based on augmented matrix. The last two rows give the average lower bound and the number

Table 4.4: Comparison of the lower bounds on QCQP examples

Instance	Normal Matrix				Augmented Matrix			
	SDP-α	SDP-H	multi-H	higher-H	SDP-α	SDP-H	multi-H	higher-H
set2-30-1	−51.7545	−51.7413	−51.7147	−51.7427	−51.7545	−51.7413	−51.7147	−51.7432
set2-30-2	−23.3151	−23.1563	−23.1281	−23.2270	−23.3151	−23.1563	−23.1281	−23.2309
set2-30-3	−9.4396	−9.4053	−9.3422	−9.3690	−9.4396	−9.4053	−9.3422	−9.3897
set2-30-4	0.9328	1.0154	1.0025	0.9668	0.9328	1.0154	1.0025	0.9760
set2-30-5	−76.9486	−76.9486	−76.9486	−76.9486	−76.9486	−76.9486	−76.9486	−76.9486
set2-30-6	−28.6994	−28.3733	−28.3737	−28.3835	−28.6994	−28.3956	−28.3737	−28.3816
set2-30-7	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762
set2-30-8	−5.4130	−5.3495	−5.1755	−5.3121	−5.4130	−5.3495	−5.1755	−5.3855
set2-30-9	−43.6991	−42.9795	−42.9961	−42.9924	−43.6991	−43.0169	−42.9961	−42.9940
set2-30-10	−37.7290	−37.7290	−37.7290	−37.7290	−37.5249	−37.6246	−37.6246	−37.6246
set2-50-1	−20.4542	−20.3175	−20.3175	−20.3175	−20.4542	−20.2456	−20.2456	−20.2496
set2-50-2	−6.1578	−6.0755	−6.1578	−6.0385	−6.1578	−6.0755	−6.1578	−6.0385
set2-50-3	−125.1570	−125.1391	−125.0919	−125.1273	−125.1570	−125.1391	−125.0919	−125.1371
set2-50-4	−104.1538	−104.0282	−103.9448	−104.0099	−104.1538	−104.0282	−103.9475	−104.0159
set2-50-5	−119.3448	−118.5532	−118.4960	−118.5649	−119.3448	−118.7302	−118.4960	−119.2143
set2-50-6	−129.8462	−128.3583	−125.8354	−127.4061	−129.8462	−128.3583	−125.8354	−127.7323
set2-50-7	−128.3688	−125.6790	−124.8283	−125.4745	−128.3688	−125.6790	−124.8283	−125.4822
set2-50-8	−65.8753	−65.8257	−65.8138	−65.8157	−65.8753	−65.8257	−65.8138	−65.8160
set2-50-9	−13.1067	−13.1026	−13.1026	−13.1043	−13.1067	−13.1026	−13.1026	−13.1050
set2-50-10	7.2697	7.2699	7.2699	7.2699	7.2697	7.2699	7.2699	7.2699
AVERAGE	−51.6968	−51.3576	−51.1700	−51.3001	−51.6866	−51.3607	−51.1613	−51.3460
#improved		17/20	16/20	17/20		17/20	16/20	17/20

of improved instances comparing with SDP-α. The ideal situation of those SDP cuts related methods is that their optimal value is exactly equal to RLT+SDP's. However, most optimal value are not reach to it, since we set the stopping criteria as either X is SDP or iteration times of generating SDP cuts $k = 50$. Furthermore, we plot box-plots of percentage improvement over SDP-α for computational results of QCQP examples. Figure 4.4 shows the %improvement for our proposed strategy and its variations on normal matrix. It shows that our proposed approach and its variations could improve much more than SDP-α for most of instances. Table 4.5 illustrates the computational time for difference methods. From these two table, we can see that, our proposed method SDP-H and its variations (Multi-H and Higher-H) are comparable methods with SDP-α. Especially Multi-H and Higher-H, they gives better lower bound averagely, without consuming more CPU time.

Table 4.5: Comparison of CPU time on QCQP examples

Instance	Normal Matrix				Augmented Matrix			
	SDP-α	SDP-H	multi-H	higher-H	SDP-α	SDP-H	multi-H	higher-H
set2-30-1	2.4410	2.6190	2.6069	2.4752	2.1179	2.3577	2.7619	2.1383
set2-30-2	2.5200	2.5652	2.4418	2.2507	2.4418	2.6620	2.7865	2.7476
set2-30-3	2.2966	2.0065	2.4023	2.1922	2.1210	2.2064	2.2317	2.4395
set2-30-4	1.9434	2.1884	1.9177	2.2837	2.0644	1.9066	2.4154	2.3733
set2-30-5	1.5083	1.9821	1.5839	2.0341	1.7842	1.7658	2.1613	2.1715
set2-30-6	1.9604	2.0734	1.8184	2.0629	2.1623	2.2280	2.1425	2.0925
set2-30-7	0.0556	0.0496	0.0558	0.0342	0.0403	0.0608	0.0371	0.0716
set2-30-8	1.8649	2.0689	2.0173	1.9417	2.2363	2.2847	2.1750	2.1088
set2-30-9	1.8806	2.0441	2.4183	1.9549	2.1034	2.5075	2.4979	2.3879
set2-30-10	0.0348	0.0558	0.0629	0.0569	2.6260	2.7637	2.6232	2.6431
set2-50-1	14.1685	12.7106	12.5093	13.6340	13.9209	13.1583	13.1787	12.9600
set2-50-2	14.8055	14.6297	14.2741	14.2899	15.3513	15.0812	15.6435	15.3294
set2-50-3	10.8693	11.3509	10.1239	11.3515	11.4120	12.1167	10.4472	10.8140
set2-50-4	12.5086	15.4024	15.5446	14.7579	12.9744	15.7292	15.8936	15.1609
set2-50-5	12.5248	12.3735	11.8229	12.1913	13.3100	12.4268	12.3149	12.3833
set2-50-6	15.2830	12.8470	14.4483	14.0517	15.3048	13.5230	14.9822	14.1139
set2-50-7	15.0628	15.2750	14.9293	15.2111	15.1800	15.9198	15.6995	15.4709
set2-50-8	15.4936	14.8429	15.7888	15.2269	15.5804	15.1548	15.6256	15.0824
set2-50-9	11.4621	10.8141	9.9370	11.9177	10.9677	10.1523	9.4723	11.3443
set2-50-10	11.9878	6.8238	6.9612	7.4371	12.0401	6.4051	6.8338	6.9813
AVERAGE	7.5336	7.2361	7.1832	7.3678	7.7870	7.5205	7.5962	7.5407

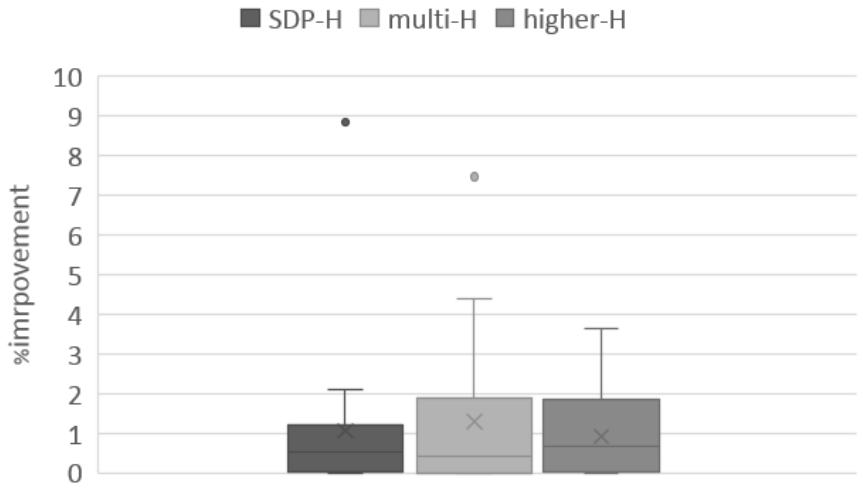


Figure 4.4: Box-plot of %improvement over SDP-α for QCQP examples on the situation of Normal matrix

Table 4.6: Comparison of the lower bounds on 0-1 QCQP examples

Instance	Normal Matrix				Augmented Matrix			
	SDP-α	SDP-H	multi-H	higher-H	SDP-α	SDP-H	multi-H	higher-H
set2-30-1	−51.7545	−51.7413	−51.7147	−51.7426	−51.7526	−51.7376	−51.7109	−51.7337
set2-30-2	−23.2166	−23.1563	−23.1281	−23.2270	−23.3087	−23.1515	−23.1505	−23.2680
set2-30-3	−9.4994	−9.4053	−9.3422	−9.3690	−9.4392	−9.3533	−9.3520	−9.3750
set2-30-4	0.9328	1.0154	1.0025	0.9668	1.0327	1.0329	1.0528	0.9999
set2-30-5	−76.9486	−76.9486	−76.9486	−76.9486	−76.9486	−76.9485	−76.9485	−76.9485
set2-30-6	−28.4024	−28.3733	−28.3737	−28.3835	−28.3824	−28.3696	−28.3585	−28.3696
set2-30-7	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762	−52.6762
set2-30-8	−5.4130	−5.3743	−5.1755	−5.3121	−5.4369	−5.4237	−5.1225	−5.2748
set2-30-9	−43.6991	−42.9795	−42.9961	−42.9924	−43.6623	−43.1396	−43.1484	−43.4110
set2-30-10	−37.7290	−37.7290	−37.7290	−37.7290	−37.5925	−37.5254	−37.5253	−37.5258
set2-50-1	−12.5841	−12.6038	−12.5716	−12.6504	−12.6567	−12.5996	−12.5690	−12.6044
set2-50-2	−5.4590	−5.4590	−5.4590	−5.4590	−5.4590	−5.4590	−5.4590	−5.4590
set2-50-3	−38.4992	−38.5067	−38.4873	−38.5019	−38.4987	−38.5065	−38.4822	−38.5018
set2-50-4	−40.8956	−40.8449	−40.8115	−40.8773	−40.9012	−40.8386	−40.8460	−40.8726
set2-50-5	−44.6065	−44.6065	−44.6065	−44.6494	−44.6277	−44.6168	−44.6135	−44.6424
set2-50-6	−72.4809	−72.4711	−72.4116	−72.4708	−72.4417	−72.4326	−72.3118	−72.4417
set2-50-7	−68.9721	−68.8956	−68.8073	−68.9138	−68.8736	−68.8634	−68.7880	−68.8139
set2-50-8	−53.9375	−53.9375	−53.9375	−53.9375	−53.9972	−53.9774	−53.9774	−53.9774
set2-50-9	−8.7438	−8.7176	−8.7176	−8.7241	−8.7438	−8.7164	−8.7013	−8.7283
set2-50-10	9.0832	9.1109	9.1109	9.1291	9.1439	9.0917	9.0917	9.1082
AVERAGE	−33.2751	−33.2150	−33.1890	−33.2234	−33.2611	−33.2106	−33.1798	−33.2258
#improved		17/20	16/20	17/20		17/20	16/20	17/20

4.4.4 0-1 QCQP examples

The algorithm we proposed could be applied on binary QCQP problem. What needs to be careful is that $x = \text{diag}(X)$ in the binary situation, which means

$$X = \begin{pmatrix} x_1 & y_{12} & \cdots & y_{1n} \\ y_{12} & x_2 & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1n} & y_{2n} & \cdots & x_n \end{pmatrix}.$$

We test 20 pure 0-1 QCQP examples [71] to show the performance of our method on 0-1 QCQP.

Table 4.7: Comparison of CPU time on 0-1 QCQP examples

Instance	Normal Matrix				Augmented Matrix			
	SDP-α	SDP-H	multi-H	higher-H	SDP-α	SDP-H	multi-H	higher-H
set2-30-1	2.4474	2.1141	2.4114	2.2550	2.0144	1.8624	2.0888	2.0795
set2-30-2	2.0439	1.9354	1.8735	2.1727	1.9378	2.0775	2.2039	1.9656
set2-30-3	1.7219	1.4919	1.6793	1.6072	1.7477	1.7218	1.6277	1.8262
set2-30-4	1.5040	1.7705	1.6978	1.8073	1.7835	1.7498	1.8544	1.6676
set2-30-5	1.4394	1.7227	1.7800	1.5916	1.7625	1.6221	1.6813	1.8396
set2-30-6	1.8142	1.5043	1.5067	1.5685	1.8960	1.6486	1.6011	1.6632
set2-30-7	0.0514	0.0452	0.0455	0.0555	0.0372	0.0364	0.0780	0.0578
set2-30-8	1.7802	2.1058	1.8832	2.0715	1.9843	1.8558	2.0578	1.6216
set2-30-9	1.6997	1.6627	1.6874	2.0966	1.8359	1.8603	1.9479	1.6912
set2-30-10	0.0540	0.0388	0.0382	0.0485	1.9724	2.1610	1.8620	1.7446
set2-50-1	6.2773	6.4183	6.7638	6.1813	5.6565	5.9118	6.5887	5.8856
set2-50-2	5.7346	5.7396	5.8585	5.9058	5.6076	5.5390	5.5712	5.9185
set2-50-3	6.7867	6.9683	6.7366	6.6954	6.4261	6.3840	6.7153	6.5248
set2-50-4	6.1330	6.4277	6.2077	6.1502	5.6131	6.0201	6.2657	5.7594
set2-50-5	5.4231	6.0355	5.9844	5.5032	5.4208	5.6949	5.9902	5.5827
set2-50-6	5.5661	5.8126	6.1595	6.0372	5.6828	6.0208	6.4281	6.3071
set2-50-7	5.9780	6.0542	6.0746	5.9121	5.8503	6.5090	6.7849	6.8943
set2-50-8	6.9586	7.2041	7.6091	7.1756	6.4063	7.3059	7.5290	7.6415
set2-50-9	5.7478	5.5276	6.0195	5.7845	5.6301	5.9540	6.0959	5.8691
set2-50-10	6.1994	6.5403	6.3590	6.4670	5.9461	0.5719	0.5235	6.4110
AVERAGE	3.7680	3.8560	3.9188	3.8543	3.7606	3.6254	3.7748	3.9475

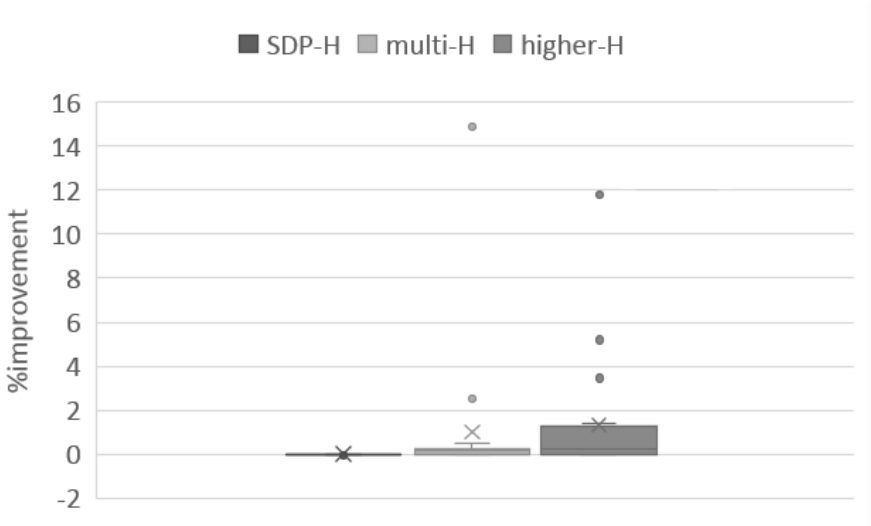


Figure 4.5: Box-plot of %improvement over SDP-α for 0-1 QCQP examples on the situation of Normal matrix

Table 4.6 gives the lower bound of different methods using the same fashion with Table 4.4. Columns 2-5 shows the lower bounds and columns 6-9 are the corresponding lower bound with adding cuts based on augmented matrix. The last

two rows are average lower bound and the number of improved instances over corresponding SDP-α. Similarly, Table 4.7 illustrates the computational time for difference approaches. From these two table, we can see that, SDP-H is a comparable method. Both Multi-H and Higher-H are better than them, especially Multi-H, which could be explained by that more cuts are added in this approaches. We also add box-plots for percentage improvement over corresponding SDP-α, see Figure 4.5 for generating cuts on normal matrix. Our proposed SDP cuts generating scheme not only performs well on QCQP problems, but also on 0-1 QCQP problems.

4.5 Conclusions and extensions

We proposed an improved SDP cuts generation scheme, which could generate a matrix H directly such that $H \bullet X < 0$ if X is not PSD, and its worse case complexity is $O(n^3)$. Some variational approaches are proposed. When adding multiple cuts (Multi-H) or certain more complicated cut (Higher-H) , it improves the lower bound with little time consuming. The relaxation problem and valid inequality added in our scheme are linear, which is easy to solve with linear solver. The SDP cuts strategy we proposed can be extended towards 0-1 multilinear programs and the more general class of polynomial programs. A crude way is to do preparation by transforming them to equivalent QCQP problems and then follow the process regarding solving QCQP problem. However, as polynomial programs of higher degree have proven to be very difficult to solve in practice, a carefully designed research framework must be employed when extending such techniques to a more general class of problems. Furthermore, this process could be embedded with branch and bound/cut algorithm to get an global optimum solution in future research. For each branching node, SDP cuts could be generated accordingly. There is one disadvantage that the generated cut may only be suit for the corresponding node, and this could lead computational burden.

5 A Second Order Cone Procedure for Solving 0-1 QPs

In Chapter 3 and 4, we mainly focus on solving 0-1 QCQPs. As we mentioned before, a 0-1 QCQP could be transformed to an equivalently 0-1 MIP by using RLT. In this chapter, we focus on solving 0-1 QCQPs through solving 0-1 MIPs.

In paper [24], the authors introduce a lift-and-project procedure for 0-1 linear programming on basis of p order cone. This procedure generalizes and unifies several existing methods and it makes a theoretical contribution to the literature in regard to lift-and-project hierarchy. In the particular case of $p = 2$, it has been proven that the procedure enjoys a better theoretical iteration complexity in contrast with existing lift-and-project technique when applying one iteration. And $p = \infty$ enjoys the strongest relaxation among $p \in [1, \infty]$.

This paper leads us to think to utilize this second order cone procedure and embed it with the RLT techniques, which may give us a procedure with fast convergence rate. We are trying to construct a procedure (second order cone procedure) between the RLT and SDP hierarchy, which enjoys the benefits of both of them.

The tightness of the different relaxations' relationship is as follows:

$$\text{RLT [58]} \leq \text{SOCP} \leq \text{SDP [39, 40]}.$$

RLT relaxations: $conv(F) \equiv P_n \subseteq \cdots \subseteq P_2 \subseteq P_1 \subseteq P_0 \equiv \text{LP}(F)$, where $P_i = \text{proj}(\text{RLT-}i)$. This procedure is convergent within n-steps.

p-cone procedure [24]: let $N^k(P_0) := N(N^{k-1}(P_0))$, $conv(F) \subseteq \cdots \subseteq N^k(P_0) \subseteq N^{k-1}(P_0) \subseteq \cdots \subseteq P_0 \equiv \text{LP}(F)$, which processes infinite convergence.

Our purpose is to construct an SOCP procedure by applying SOCP(RLT-i)for step i,where the second SOCP means the operation $N(2, J)$ in [24].

5.1 Methodology

First, we discuss some properties related with index set J.

5.1.1 Preliminary analysis

We start from considering the relaxation from the aspect of how does $\mathcal{J}$ influence the tightness of SOCP constraint? When $\mathcal{J}_1 \subseteq \mathcal{J}_2$, we want to compare $\|x_{\mathcal{J}_1} - c\| \leq r$ with $\|x_{\mathcal{J}_2} - c\| \leq r$. Is SOCP($\mathcal{J}_2$) tighter than SOCP($\mathcal{J}_1$)? Instinctively, it seems true. But the answer is NO. We have following property.

Property 4. *$J_1 \subseteq J_2$ can not imply the relationship between $N(2, J_1)$ and $N(2, J_2)$.*

Proof. Some counterexample could be found. Considering the following example:

Example 5. $F = \{-3.5x_1 - x_2 + 2.5 \geq 0, x \in \{0, 1\}\}$.

Its linear relaxation is

$$\begin{aligned} P_0: \quad & -3.5x_1 - x_2 + 2.5 \geq 0 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & 1 - x_2 \geq 0 \\ & (\text{noting that } 1 - x_1 \geq 0 \text{ is redundant, and we remove it}) \end{aligned}$$

- **SOCP** with $\mathcal{J} = \{1\}$, $\|(x_1 - c)\| \leq r$ (in this case, $c = \frac{1}{2}$, $r = \frac{1}{2}$)
  - **SOCP-0** (SOCP of LP)

$$\begin{aligned} & \|(x_1 - \frac{1}{2})(-3.5x_1 - x_2 + 2.5)\| \leq \frac{1}{2}(-3.5x_1 - x_2 + 2.5) \\ & \|(x_1 - \frac{1}{2})x_1\| \leq \frac{1}{2}x_1 \\ & \|(x_1 - \frac{1}{2})x_2\| \leq \frac{1}{2}x_2 \\ & \|(x_1 - \frac{1}{2})(1 - x_2)\| \leq \frac{1}{2}(1 - x_2) \end{aligned}$$

$\rightarrow$

$$
\begin{aligned}
&-x_1 - x_{12} \geq 0 && \text{(5.1a)}\\
&-2.5x_1 + x_{12} - x_2 + 2.5 \geq 0 && \text{(5.1b)}\\
&\\
&x_1 \geq 0 && \text{(5.1c)}\\
&0 \geq 0 && \text{(5.1d)}\\
&\\
&x_{12} \geq 0 && \text{(5.1e)}\\
&x_2 - x_{12} \geq 0 && \text{(5.1f)}\\
&\\
&x_1 - x_{12} \geq 0 && \text{(5.1g)}\\
&-x_1 - x_2 + x_{12} + 1 \geq 0 && \text{(5.1h)}
\end{aligned}
$$

After removing the redundant constraint, it becomes

$$
\begin{aligned}
&x_1 = 0\\
&0 \leq x_2 \leq 1\\
&x_{12} = 0,
\end{aligned}
$$

which is conv(F).

- **SOCP** with $\mathcal{J} = \{1, 2\}$, $\|(x - c)\| \leq r$ (in this case, $c = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$, $r = \frac{\sqrt{2}}{2}$)

  - **SOCP-0** (SOCP of LP)

$$
\begin{aligned}
&\left\| \begin{pmatrix} x_1 - \frac{1}{2} \\ x_2 - \frac{1}{2} \end{pmatrix} (-3.5x_1 - x_2 + 2.5) \right\| \leq \frac{\sqrt{2}}{2}(-3.5x_1 - x_2 + 2.5)\\
&\left\| \begin{pmatrix} x_1 - \frac{1}{2} \\ x_2 - \frac{1}{2} \end{pmatrix} x_1 \right\| \leq \frac{\sqrt{2}}{2} x_1\\
&\left\| \begin{pmatrix} x_1 - \frac{1}{2} \\ x_2 - \frac{1}{2} \end{pmatrix} x_2 \right\| \leq \frac{\sqrt{2}}{2} x_2
\end{aligned}
$$

$$\left\| \begin{pmatrix} x_1 - \frac{1}{2} \\ x_2 - \frac{1}{2} \end{pmatrix} (1 - x_2) \right\| \leq \frac{\sqrt{2}}{2}(1 - x_2)$$

After substitution $\rightarrow$

$$\left\| \begin{pmatrix} 0.75x_1 - x_{12} + 0.5x_2 - 1.25 \\ 1.75x_1 - 3.5x_{12} - 1.25 + 2x_2 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(-3.5x_1 - x_2 + 2.5) \tag{5.2a}$$

$$\left\| \begin{pmatrix} 0.5x_1 \\ -0.5x_1 + x_{12} \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}x_1 \tag{5.2b}$$

$$\left\| \begin{pmatrix} x_{12} - 0.5x_2 \\ 0.5x_2 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}x_2 \tag{5.2c}$$

$$\left\| \begin{pmatrix} x_1 - x_{12} - 0.5 + 0.5x_2 \\ -0.5 + 0.5x_2 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(1 - x_2) \tag{5.2d}$$

Plot and get the projection by plotting all the points satisfied all the four constraints. We get the projection as the shadow part in Figure 5.1. We can see that SOCP($J = \{1, 2\}$) $\not\subseteq$ SOCP($J = \{1\}$).

□

In most relaxations, some new variables are added due to substitution of products and we say those new variables as a vector y. We start to think of the relationship among the following three situations: (1) $\|x - c\| \leq r$; (2) $\left\| \begin{pmatrix} x \\ y \end{pmatrix} - c \right\| \leq r$; and (3) $\|x - c\| \leq r$ & $\|y - c\| \leq r$.

For (2) and (3), we have the following property.

Property 5. $N(2, J_1) + N(2, J_2) \subseteq N(2, J)$, *if* $J_1 \cup J_2 = J$ *and* $J_1 \cap J_2 = \varnothing$

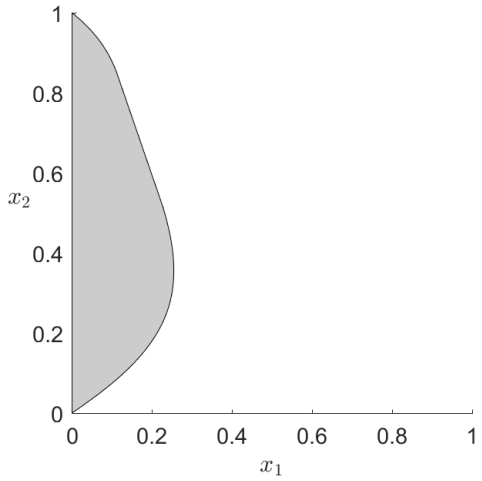


Figure 5.1: Projection of SOCP$_{J=\{1,2\}}$.

Proof.

$$\|(x_{J_1} - c)(a^\top x - b)\| \leq \frac{\sqrt{|J_1|}}{2}(a^\top x - b) \tag{5.3a}$$

$$\|(x_{J_2} - c)(a^\top x - b)\| \leq \frac{\sqrt{|J_2|}}{2}(a^\top x - b) \tag{5.3b}$$

$$\|(x_J - c)(a^\top x - b)\| \leq \frac{\sqrt{|J|}}{2}(a^\top x - b) \tag{5.3c}$$

It is easy to see that (5.3a)^2 + (5.3b)^2 $\Rightarrow$ (5.3c)^2. Hence, when we construct SOCP relaxation, using J_1 and J_2 together is tighter than J. □

Similarly, we assert that this conclusion is suitable for general p-order cone relaxations.

According to Property 5, we can see that to split set J into two different index set J_1 and J_2 with $p = 2$ in the process of constructing SOCP relaxation will tight the relaxation. Simultaneously, it will increase the number of constraints, then the computational time.

From an intuitive point of view, we consider a very simple example.

Example 6. $F = \{x \in \{0,1\}^2, x_1 + 2x_2 \leq 2.5\}$.

- **RLT-0(LP)**

$$
\begin{aligned}
-x_1 - 2x_2 + 2.5 &\geq 0 && \text{(5.4a)}\\
x_1 &\geq 0 && \text{(5.4b)}\\
1 - x_1 &\geq 0 && \text{(5.4c)}\\
x_2 &\geq 0 && \text{(5.4d)}\\
1 - x_2 &\geq 0 && \text{(5.4e)}
\end{aligned}
$$

- **RLT-1**

$$
\begin{aligned}
1.5x_1 - 2x_{12} &\geq 0 && \text{(5.5a)}\\
-2.5x_1 - 2x_2 + 2x_{12} + 2.5 &\geq 0\ (*) && \text{(5.5b)}\\
0.5x_2 - x_{12} &\geq 0 && \text{(5.5c)}\\
-x_1 + x_{12} + 2.5 - 2.5x_2 &\geq 0\ (*) && \text{(5.5d)}\\
x_{12} &\geq 0 && \text{(5.5e)}\\
x_1 - x_{12} &\geq 0\ (*) && \text{(5.5f)}\\
x_2 - x_{12} &\geq 0\ (*) && \text{(5.5g)}\\
-x_1 - x_2 + x_{12} + 1 &\geq 0 && \text{(5.5h)}
\end{aligned}
$$

where the constraints marked $(*)$ are redundant. Remove the $(*)$ constraints, RLT-1 is

$$
\begin{aligned}
1.5x_1 - 2x_{12} &\geq 0 && \text{(5.6a)}\\
0.5x_2 - x_{12} &\geq 0 && \text{(5.6b)}\\
x_{12} &\geq 0 && \text{(5.6c)}\\
-x_1 - x_2 + x_{12} + 1 &\geq 0. && \text{(5.6d)}
\end{aligned}
$$

To get the projection of RLT-1, using Fourier-Motzkin Elimination, we have

$$
\left.\begin{matrix} 0 \\ x_1 + x_2 - 1 \end{matrix}\right\} \leq x_{12} \leq \left\{\begin{matrix} 0.5x_2 \\ \\ 0.75x_1 \end{matrix}\right. ,
$$

then we obtain the following projection P_1 of RLT-1 on x:

$$-x_1 - 0.5x_2 + 1 \geq 0 \tag{5.7a}$$
$$-0.25x_1 - x_2 + 1 \geq 0 \tag{5.7b}$$
$$x_1 \geq 0 \tag{5.7c}$$
$$x_2 \geq 0. \tag{5.7d}$$

- **SOCP-1** to construct based on (5.7)

  – with $\left\| \begin{pmatrix} x_1 - 0.5 \\ x_2 - 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}$, and we denote this situation as *case* x since we construct SOCP constraints with J including all index in x.

$$\left\| \begin{pmatrix} -0.5x_1 - 0.25x_2 + 0.5x_{12} + 0.5 \\ -0.5x_1 - 0.75x_2 + x_{12} + 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(-x_1 - 0.5x_2 + 1) \tag{5.8a}$$

$$\left\| \begin{pmatrix} -0.875x_1 - 0.5x_2 + x_{12} + 0.5 \\ -0.125x_1 - 0.5x_2 + 0.25x_{12} + 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(-0.25x_1 - x_2 + 1) \tag{5.8b}$$

$$\left\| \begin{pmatrix} 0.5x_1 \\ x_{12} - 0.5x_1 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}x_1 \tag{5.8c}$$

$$\left\| \begin{pmatrix} x_{12} - 0.5x_2 \\ 0.5x_2 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}x_2 \tag{5.8d}$$

  – with $\left\| \begin{pmatrix} x_1 - 0.5 \\ x_2 - 0.5 \\ x_{12} - 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}$ and we denote this situation as *case* xy since we construct SOCP constraints with J including all index in x and all new

variables.

$$\left\| \begin{pmatrix} -0.5x_1 - 0.25x_2 + 0.5x_{12} + 0.5 \\ -0.5x_1 - 0.75x_2 + x_{12} + 0.5 \\ -0.5x_1 - 0.25x_2 + 0.5x_{12} + 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}(-x_1 - 0.5x_2 + 1) \tag{5.9a}$$

$$\left\| \begin{pmatrix} -0.875x_1 - 0.5x_2 + x_{12} + 0.5 \\ -0.125x_1 - 0.5x_2 + 0.25x_{12} + 0.5 \\ -0.125x_1 - 0.5x_2 + 0.25x_{12} + 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}(-0.25x_1 - x2 + 1) \tag{5.9b}$$

– with $\left\| \begin{pmatrix} x_1 - 0.5 \\ x_2 - 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}$ and $\|x_{12} - 0.5\| \leq \frac{1}{2}$, and we denote this situation as case $x + y$ since we construct SOCP constraints with J including all index in x and also construct SOCP constraints with J including all new defined variables.

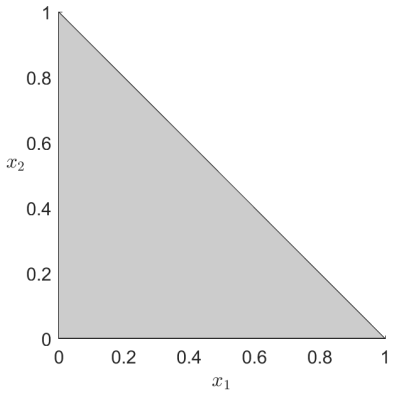
Equations (5.8) and

$$|-0.5x_1 - 0.25x_2 + 0.5x_{12} + 0.5| \leq \frac{1}{2}(-x_1 - 0.5x_2 + 1) \tag{5.10a}$$

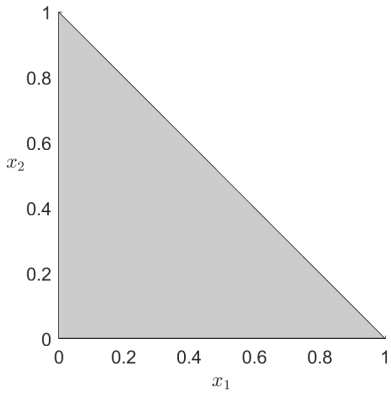
$$|-0.125x_1 - 0.5x_2 + 0.25x_{12} + 0.5| \leq \frac{\sqrt{1}}{2}(-0.25x_1 - x2 + 1) \tag{5.10b}$$

Noting that Figure 5.2d shows the LP relaxation and the projection of RLT-1 (5.7). From Figure 5.2a-5.2c, we can see that all these three different situations gives convex hull of F in this case.

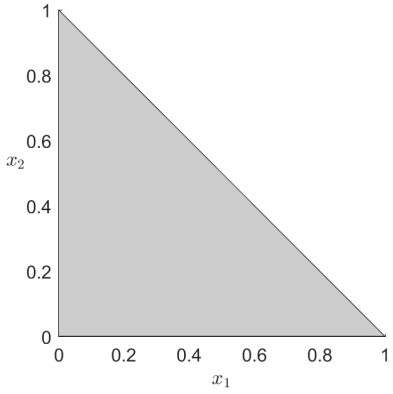
It is coincident with Property 5's conclusion that (5.10) should be at least as good as (5.9). Comparing with Figure 5.2d, all the SOCP relaxations tight the region quite a lot and equal to the convex hull in this simple example, which shows the efficiency of the SOCP relaxation.



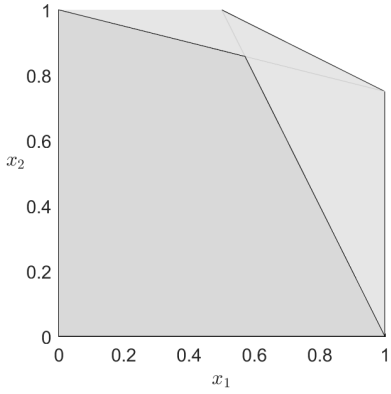
(a) Projection for case x



(b) Projection for case xy



(c) Projection for case $x + y$



(d) LP relaxation and Projection of RLT-1

- case x: Projection of 5.8
- case xy: Projection of 5.9
- case $x + y$: Projection of 5.10

Figure 5.2: Projections for different cases of Example 6

- **SOCP-1** to construct based on (5.6)
  - with $\left\| \begin{pmatrix} x_1 - 0.5 \\ x_2 - 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}$

$$\left\| \begin{pmatrix} 0.75x_1 - x_{12} \\ -.75x_1 + 0.5x_{12} \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(1.5x_1 - 2x_{12}) \tag{5.11a}$$

$$\left\| \begin{pmatrix} -0.25x_2 \\ 0.25x_2 - 0.5x_{12} \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(0.5x_2 - x_{12}) \tag{5.11b}$$

$$\left\| \begin{pmatrix} 0.5x_{12} \\ 0.5x_{12} \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}x_{12} \qquad (**)$$

(5.11c)

$$\left\| \begin{pmatrix} 0.5x_1 + 0.5x_2 - 0.5x_{12} - 0.5 \\ 0.5x_1 + 0.5x_2 - 0.5x_{12} - 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(-x_1 - x_2 + x_{12} + 1) \quad (**)$$

(5.11d)

(**) part could be replaced by $x_{12} \geq 0$ and $-x_1 - x_2 + x_{12} + 1 \geq 0$

– with $\left\| \begin{pmatrix} x_1 - 0.5 \\ x_2 - 0.5 \\ x_{12} - 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}$

$$\left\| \begin{pmatrix} 0.75x_1 - x_{12} \\ -.75x_1 + 0.5x_{12} \\ -.75x_1 + 0.5x_{12} \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}(1.5x_1 - 2x_{12}) \tag{5.12a}$$

$$\left\| \begin{pmatrix} -0.25x_2 \\ 0.25x_2 - 0.5x_{12} \\ -0.25x_2 \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}(0.5x_2 - x_{12}) \tag{5.12b}$$

$$\left\| \begin{pmatrix} 0.5x_{12} \\ 0.5x_{12} \\ 0.5x_{12} \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}x_{12} \qquad (**)$$

(5.12c)

$$\left\| \begin{pmatrix} 0.5x_1 + 0.5x_2 - 0.5x_{12} - 0.5 \\ 0.5x_1 + 0.5x_2 - 0.5x_{12} - 0.5 \\ 0.5x_1 + 0.5x_2 - 0.5x_{12} - 0.5 \end{pmatrix} \right\| \leq \frac{\sqrt{3}}{2}(-x_1 - x_2 + x_{12} + 1) \quad (**)$$

(5.12d)

(**) part could be replaced by $x_{12} \geq 0$ and $-x_1 - x_2 + x_{12} + 1 \geq 0$

Both of them give the convex hull (which is a triangle in figure). Hence, we can not tell which one is tighter from this case.

For now, we can make the following conclusion from the previous example:

- For these three cases: (1) $\|x - c\| \leq r$; (2) $\left\| \begin{pmatrix} x \\ y \end{pmatrix} - c \right\| \leq r$; (3)$\|x - c\| \leq r$ & $\|y - c\| \leq r$, using (3) is tighter than (2), and (3) is also tighter than (1). But we can not compare (2) and (1). In this section's case, Example 6, we can see (2) (5.9) is tighter than (1) (5.8), but in Example 5 (1) is tighter than (2).

- In this case, to construct SOCP relaxation based on RLT-1 is tighter than based on P_1(projection of (RLT-1)) (This is kind of similar with Sherali's RLT procedure is tighter than Balas-Ceria-Cornuejols construction [12]). Is this a general conclusion? Yes. And we have the following conclusion.

Property 6. *SOCP(RLT-1)($J = [n]$) $\subseteq$ SOCP(P_1)($J = [n]$).*

Proof. Lift P_1 to $n + C_n^2$ dimension and the C_n^2 dimension could be arbitrary value. Obviously SOCP(RLT-1) $\subseteq$ SOCP(lifted P_1). After projecting to x, SOCP(lifted P_1) = SOCP(P_1) since the C_n^2 variables are never be used in SOCP(lifted P_1) ($J = [n]$). Intuitively, P_1 lose some information after projecting RLT-1 to x, if we lift it back to higher dimension, the result $\supseteq$ RLT-1. □

5.1.2 Construct SOCP procedure (from RLT to SOCP)

Consider 0-1 linear programming with feasible region as defined below (the same with (2.1)):

$$F = \{x \in \mathbb{R}^n, \sum_j a_{lj}x_j \geq b_l, l = 1, ..., L, x \text{ is binary}\}, \tag{5.13}$$

and its linear relaxation is

$$P_0 = \{x \in \mathbb{R}^n, \sum_j a_{lj}x_j \geq b_l, l = 1, ..., L, 0 \leq x \leq e\}, \tag{5.14}$$

where e is a n-vector with all elements are ones.

Firstly, apply RLT rechnique. Multiply every L linear restrictions in P_0 (*i.e.*, excluding the bounding restrictions $0 \leq x \leq e$) by all possible bound factors (the bound factors in RLT relaxation is defined as (2.2)), using the relationships $x_j^2 = x_j, \forall j = 1, \cdots, n$. Append the resulting collection of polynomial relationships the constraints representing nonnegativities on all bound factors of degree $D = \min\{d + 1, n\}$; that is, $F_D(J_1, J_2) \geq 0$ for each (J_1, J_2) of order D.

Then, linearize the resulting polynomial constraint formulations by replacing the nonlinear term $\prod_{j \in J} x_j$ with associated y_J for each set J with $|J| \geq 2$, where the cardinality of set J are sequenced in order of increment. Denoting in generating the linear expression for any factor $F_d(J_1, J_2)$ obtained upon making such a replacement regarding the variables (x, y) as $f_d(J_1, J_2)$, we obtain the resulting polyhedral set P_d given in (5.15) as follows:

$$\begin{aligned} P_d = &\{(x, y) : \\ &(\sum_{j \in J_1} a_{lj} - b_l) f_d(J_1, J_2) + \sum_{j \in N-(J_1 \cup J_2)} a_{lj} f_{d+1}(J_1 + j, J_2) \geq 0, \qquad \text{(5.15a)} \\ &\text{for } l = 1, ..., L, \forall (J_1, J_2) \text{ of order } d, \end{aligned}$$

$$f_D(J_1, J_2) \geq 0, \forall (J_1, J_2) \text{ of order } d \tag{5.15b}$$

where $D = \min\{d+1, n\}$ }.

Lemma 1. *[58], For any $1 \leq d < n$, the constraints $f_d(J_1, J_2)$ for all (J_1, J_2) of order d are implied by the constraints $f_{d+1}(J_1, J_2) \geq 0$ for all (J_1, J_2) of order $(d+1)$.*

Theorem 5. *[58], $P_0 \supseteq P_1 \supseteq \cdots \supseteq P_n \equiv conv(F)$.*

Now we start to construct our SOCP relaxation based on RLT relaxation:

To reduce the computational complexity, we keep the linear constraints unchanged and do not multiply bound factor F_d with them in the relaxation part, and use the following RLT relaxations,

$$\begin{aligned} \text{RLT}_d = &\{(x, y) : \\ &a^\top x - b \geq 0, \qquad (5.16a) \\ &f_D(J_1, J_2) \geq 0, \forall (J_1, J_2) \text{ of order } d \}, \qquad (5.16b) \end{aligned}$$

where $D = \min\{d+1, n\}$. For convenience of representation, we consider the situation that there only is one constraint in the formulation of RLT_d. (For multiple constraints, we can use $a_l^\top x - b_l \geq 0, l = 1, \cdots, L.$)

Now we multiply $\|x_J - c\| \leq r$ with left hand side of (5.16a).

In RLT_d, there are $\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{d+1}$ variables. We split these variables as two part:

Part 1, $x_J = x, J = [n], c = \frac{1}{2}e, r = \frac{\sqrt{n}}{2}$

- $\|(a^\top x - b)(x - \frac{1}{2}e)\| \leq \frac{\sqrt{n}}{2}(a^\top x - b)$

Linearise quadratic terms by introducing new variables $\rightarrow$

$$\left\|[xx^\top]_{\mathrm{L}}a - bx - \tfrac{1}{2}(a^\top x - b)e\right\| \leq \frac{\sqrt{n}}{2}(a^\top x - b)$$

- $\left\|f_D(J_1, J_2)(x - \frac{1}{2}e)\right\| \leq \frac{\sqrt{n}}{2} f_D(J_1, J_2)$, where (J_1, J_2) with order D.

  Linearise quadratic terms by introducing new variables $\rightarrow$

  $$\text{LHS} \quad = \quad \left\| \begin{pmatrix} \vdots \\ f_D(J_1, J_2)x_k - \dfrac{1}{2} f_D(J_1, J_2) \\ \vdots \end{pmatrix} \right\|, \text{ where the } k_{th} \text{ element } =$$

  $$\begin{cases} \dfrac{1}{2} f_D(J_1, J_2) & k \in J_1 \\ -\dfrac{1}{2} f_D(J_1, J_2) & k \in J_2 \\ f_D(J_1 + k, J_2) - \dfrac{1}{2} f_D(J_1, J_2) & k \in N \backslash \{J_1 \cup J_2\} \end{cases} .$$

Hence, the LHS could be represented as $\|u\|$, where

$$u_k = \begin{cases} \frac{1}{2} f_D(J_1, J_2) & k \in J_1 \cup J_2 \\ f_D(J_1 + k, J_2) - \dfrac{1}{2} f_D(J_1, J_2) & k \in N \backslash \{J_1 \cup J_2\} \end{cases} ,$$

$$\begin{aligned} \|u\| &= \sqrt{\sum_{k \in \{J_1 \cup J_2\}} (\frac{1}{2} f_D(J_1, J_2))^2 + \sum_{k \notin \{J_1 \cup J_2\}} (f_D^2(J_1 + k, J_2) - f_D(J_1, J_2 + k) f_D(J_1, J_2) + (\frac{1}{2} f_D(J_1, J_2))^2)} \\ &= \sqrt{(\frac{1}{2} f_D(J_1, J_2))^2 \cdot n + \sum_{k \notin \{J_1 \cup J_2\}} (f_D^2(J_1 + k, J_2) - f_D(J_1 + k, J_2) f_D(J_1, J_2))} \end{aligned}$$

$$\begin{aligned} [f_D(J_1 + k, J_2) f_D(J_1, J_2)]_{\mathrm{L}} &= \left[\left[\prod_{j \in \{J_1 + k\}} x_j \prod_{j \in J_2} (1 - x_j) \right]_{\mathrm{L}} \cdot \left[\prod_{j \in J_1} x_j \prod_{j \in J_2} (1 - x_j) \right]_{\mathrm{L}} \right]_{\mathrm{L}} \\ &= \left[x_k \prod_{j \in J_1} x_j \prod_{j \in J_2} (1 - x_j) \cdot \prod_{j \in J_1} x_j \prod_{j \in J_2} (1 - x_j) \right]_{\mathrm{L}} \\ &\quad \text{(I claim that} [[\bullet]_{\mathrm{L}} [\bullet]_{\mathrm{L}}]_{\mathrm{L}} = [\bullet\bullet]_{\mathrm{L}} \text{.Please see the proof in Appendix.} \\ &\quad \text{under that} [x_{\{I_1\}} x_{\{I_2\}}]_{\mathrm{L}} = x_{\{I_1 \cup I_2\}}) \\ &= \left[\prod_{j \in \{J_1 + k\}} x_j \prod_{j \in J_2} (1 - x_j) \right]_{\mathrm{L}} \end{aligned}$$

$$=f_D(J_1+k,J_2)$$

$$\begin{aligned}
[f_D^2(J_1+k,J_2)]_{\mathrm{L}} &= \left[\left[\prod_{j\in\{J_1+k\}} x_j \prod_{j\in J_2}(1-x_j)\right]_{\mathrm{L}} \cdot \left[\prod_{j\in\{J_1+k\}} x_j \prod_{j\in J_2}(1-x_j)\right]_{\mathrm{L}}\right]_{\mathrm{L}} \\
&= \left[\prod_{j\in\{J_1+k\}} x_j \prod_{j\in\{J_2\}}(1-x_j) \prod_{j\in\{J_1+k\}} x_j \prod_{j\in J_2}(1-x_j)\right]_{\mathrm{L}} \\
&= \left[\prod_{j\in\{J_1+k\}} x_j \prod_{j\in J_2}(1-x_j)\right]_{\mathrm{L}} \\
&= f_D(J_1+k,J_2)
\end{aligned}$$

Hence, LHS $\|u\| = \dfrac{\sqrt{n}}{2}|f_D(J_1,J_2)|$.

This inequalities is equivalent with the corresponding constraint $f_D(J_1,J_2)\geq 0$.

Part 2, let x_J = the new variables added in relaxations formulation, $|J| = \binom{n}{2}+\cdots+\binom{n}{D}$, $c=\dfrac{1}{2}e$, $r=\dfrac{\sqrt{|J|}}{2}$, where e is $|J|$-dim vector with all elements 1. Here vector x_J's elements include all $X_{\{I_D\}},\ 2\leq D\leq n$, where the index $\{I_D\}$ represents an ordered sequence with elements number is D, whose components compose a subset of $\{1,\cdots,n\}$ (*i.e.*, $\{I_D\}=i_1i_2\cdots i_D,\ i_1,\cdots,i_D\in\{1,\cdots n\},\ 1\leq i_1<i_2<\cdots<i_k\leq n,\ 2\leq D$). $X_{\{I_D\}}=\{x_s, s\in\{I_D\}\}$ (noting that $X_{\{I_1\}}\equiv x$). We call this vector as Y_D,

$$Y_D = \text{vector with all elements in } \cup_{k=2}^{D} X_{\{I_k\}}.$$

Since the order of elements in Y_D does not affect the result of our relaxation, we do not specify the order here.

Now we apply the SOCP relaxations.

- $\left\|(a^\top x - b)(Y_D - \frac{1}{2}e)\right\| \leq \frac{\sqrt{|J|}}{2}(a^\top x - b)$

  $\rightarrow \left\|[Y_D x^\top]_{\mathrm{L}} a - bY_D - \frac{1}{2}(a^\top x - b)e\right\| \leq \frac{\sqrt{|J|}}{2}(a^\top x - b)$

- $\left\|f_D(J_1, J_2)(Y_D - \frac{1}{2}e)\right\| \leq \frac{\sqrt{|J|}}{2} f_D(J_1, J_2)$, where (J_1, J_2) with order D.

  $$\text{LHS} = \begin{pmatrix} \vdots \\ f_D(J_1, J_2)x_{i_1 \cdots i_k} - \frac{1}{2} f_D(J_1, J_2) \\ \vdots \end{pmatrix}$$

  the elements $f_D(J_1, J_2)x_{i_1 \cdots i_k} - \frac{1}{2} f_D(J_1, J_2)$

  $$= \begin{cases} -\frac{1}{2} f_D(J_1, J_2) & \{i_1, \cdots, i_k\} \cap J_2 \neq \varnothing \\ f_D(J_1 \cup \{i_1, \cdots, i_k\}, J_2) - \frac{1}{2} f_D(J_1, J_2) & \{i_1, \cdots, i_k\} \cap J_2 = \varnothing \end{cases}$$

  Similarly, we also can prove this inequalities is equivalent with $f_D(J_1, J_2) \geq 0$.

Hence we get the following property,

Property 7. *Applying SOCP to $f_D \geq 0$ is equivalent with $f_D \geq 0$.*

Now we define level d SOCP relaxation as follows:

SOCP(RLT-d):

$$\left\|[xx^\top]_{\mathrm{L}} a - bx - \frac{1}{2}(a^\top x - b)e\right\| \leq \frac{\sqrt{n}}{2}(a^\top x - b) \tag{5.17a}$$

(noting that this constraint is not changed for each level d)

$$\left\|[Y_D x^\top]_{\mathrm{L}} a - bY_D - \frac{1}{2}(a^\top x - b)e\right\| \leq \frac{\sqrt{|J|}}{2}(a^\top x - b) \tag{5.17b}$$

$$f_D(J_1, J_2) \geq 0. \tag{5.17c}$$

The following two truths are obvious,

- SOCP-d is tighter than RLT-d. (According to Proposition 3.7 in Burer and Chen [24]);

- SOCP-d is tighter than Burer and Chen [24]'s relaxation when $p = 2$ (for the first iteration).

Then we have the following theorem,

Theorem 6. *SOCP-0* $\supseteq$ *SOCP-1* $\supseteq \cdots \supseteq$ *SOCP-*$n \equiv$ *conv*(F).

Let's see the details of applying SOCP on RLT-0 and RLT-1.

SOCP(RLT-0(LP))

RLT-0(LP) relaxation of 0-1 LP (5.13):

$$a_l^\top x - b_l \geq 0,\ l = 1, .., L.$$

WLOG, we suppose that $0 \leq x \leq 1$ is implied by the linear constraints, and for convenient we discuss one constraint $a^\top x - b \geq 0$ firstly,

The following gives the detail of SOCP(RLT-0) and SOCP(RLT-1). Introduce an $n \times n$ symmetric matrix X, $diag(X) = x$.

SOCP(RLT-0)

$(c = \frac{1}{2}e \in R^{|J|\times 1}, r = \frac{\sqrt{|J|}}{2}$, and we let $J = [n]$, where$[n] := 1, ..., n)$

$$||(a^\top x - b)(x_J - c)|| \leq r(a^\top x - b)$$

$$\rightarrow ||(a^\top x - b)(x_J - c)|| \leq r(a^\top x - b)$$

$$\rightarrow ||X_J \cdot a - bx_J - (a^\top x - b)c|| \leq r(a^\top x - b)$$

$\rightarrow$project to x

SOCP(RLT-1)

RLT-1

$$
\begin{aligned}
& a^\top x - b \geq 0 && \\
& x_{ij} \geq 0 && \forall (i,j) \\
& x_i - x_{ij} \geq 0 && \forall (i,j),\ i \neq j \\
& 1 - x_i - x_j + x_{ij} \geq 0 && \forall (i,j),\ i \neq j
\end{aligned}
$$

SOCP(RLT-1)

Part 1: $J = [n], c = \frac{1}{2}e, r = \frac{\sqrt{n}}{2}$

$$||(a^\top x - b)(x - c)|| \leq r(a^\top x - b) \tag{5.20a}$$

$$||x_{ij}(x - c)|| \leq r x_{ij} \tag{5.20b}$$

$$||(x_i - x_{ij})(x - c)|| \leq r(x_i - x_{ij}) \tag{5.20c}$$

$$||1 - x_i - x_j + x_{ij}(x - c)|| \leq r(1 - x_i - x_j + x_{ij}) \tag{5.20d}$$

For part 1: (5.20a) is the same with SOCP(RLT-0), (5.20b)-(5.20d) are equivalent with the original corresponding RLT constrains.

Part 2: $x_{J_2} = all(x_{ij}), c = \frac{1}{2}e, r = \frac{\sqrt{|J_2|}}{2}$

$$||(a^\top x - b)(col(tril(X, -1)) - c)|| \leq r(a^\top x - b) \tag{5.21a}$$

$$||x_{ij}(col(tril(X, -1)) - c)|| \leq r x_{ij} \tag{5.21b}$$

$$||(x_i - x_{ij})((col(tril(X, -1)) - c)|| \leq r(x_i - x_{ij}) \tag{5.21c}$$

$$||(1 - x_i - x_j + x_{ij})(col(tril(X, -1)) - c)|| \leq r(1 - x_i - x_j + x_{ij}) \quad (5.21d)$$

where $col(tril(X, -1))$ means take the lower triangle (not including diag) variables of X. For convenient, I also can directly use X.

For part 2, $|J_2| = \begin{pmatrix} n \\ 2 \end{pmatrix}$, $c = \frac{1}{2}e$, $r = \frac{\sqrt{n(n-1)/2}}{2}$ and $x_{J_2} =$

$$col \begin{pmatrix} & & & \\ x_{12} & & & \\ x_{13} & x_{23} & & \\ \vdots & & \ddots & \\ x_{1n} & x_{2n} & \cdots & x_{(n-1)n} \end{pmatrix} = (x_{12}, x_{13}, ..., x_{1n}, x_{23}, ..., x_{2n}, ..., x_{(n-1)n})^\top$$

LHS of (5.21a)

$$= \left\| (a^\top x - b) \cdot col \begin{pmatrix} & & & \\ x_{12} - \frac{1}{2} & & & \\ x_{13} - \frac{1}{2} & x_{23} - \frac{1}{2} & & \\ \vdots & & \ddots & \\ x_{1n} - \frac{1}{2} & x_{2n} - \frac{1}{2} & \cdots & x_{(n-1)n} - \frac{1}{2} \end{pmatrix} \right\|$$

$$= \left\| col \begin{pmatrix} \vdots & & & \ddots & \\ \cdot & (a_i + a_j - b)x_{ij} + \sum_{k \neq i,j} a_k x_{ijk} - \frac{1}{2}(a^\top x - b) & & \ddots & \\ \vdots & & & & \ddots \end{pmatrix} \right\|$$

$$\leq \frac{\sqrt{n(n-1)/2}}{2}(a^\top x - b)$$

(5.21b)-(5.21d) are the corresponding constraint $f_2 \geq 0$.

5.2 Computational results

In this section, we show the performance of the proposed SOCP-1 relaxation by illustrative examples and 0-1 QP problems. All of the numerical experiments were performed on a computer equipped with CPU 3.70GHz and RAM 32.0GB. MOSEK is the solver we utilized to solve second order cone programming problem.

5.2.1 Illustrative example

To see the performance of the SOCP-1 relaxation, firstly, we test our relaxations on a simple example, the Example 1 from Chapter 3. Using RLT, Example 1 could be expressed as the following equivalent 0-1 MILP problem:

$$
\begin{aligned}
\min \quad f(x) = \; & 163x_1 - 92x_{12} + 565x_{13} + 77x_{14} - 6x_{15} + 55x_2 + 5984x_{23} \\
& -22x_{24} + 31x_{25} + 55x_3 + 22x_{34} + 2543x_{35} - 2x_4 - 76x_{45} + 10x_5 \\
s.t. \quad & -2x_1 - 4x_{12} - 2x_2 + 8x_1 + 6x_2 + x_3 - 4x_4 \leq -2.5 \\
& x_1 - 2x_2 + x_3 + x_4 + x_5 \leq 1 \\
& x_{ij} \leq x_i \\
& x_{ij} \leq x_j \\
& x_{ij} \geq 0 \\
& 1 - x_i - x_j + x_{ij} \geq 0 \\
& x \in \{0,1\}^5.
\end{aligned}
$$

Adding redundant constraint y (where y is an vector whose elements are composed of x_{ij}) is binary, the problem is also equivalent to 0-1 LP:

Example 7.

$$\begin{aligned}
\min \quad f(x) = \; & 163x_1 + 55x_2 + 55x_3 - 2x_4 + 10x_5 - 92x_{12} + 565x_{13} + 77x_{14} - 6x_{15} \\
& +5984x_{23} - 22x_{24} + 31x_{25} + 22x_{34} + 2543x_{35} - 76x_{45} && \text{(5.24a)} \\
s.t. \quad & -6x_1 - 4x_2 - x_3 + 4x_4 + 4x_{12} - 2.5 \geq 0 && \text{(5.24b)} \\
& -x_1 + 2x_2 - x_3 - x_4 - x_5 + 1 \geq 0 && \text{(5.24c)} \\
& x_{ij} \leq x_i && \text{(5.24d)} \\
& x_{ij} \leq x_j && \text{(5.24e)} \\
& x_{ij} \geq 0 && \text{(5.24f)} \\
& 1 - x_i - x_j + x_{ij} \geq 0 && \text{(5.24g)} \\
& x \text{ and } y \text{ are binary}. && \text{(5.24h)}
\end{aligned}$$

Then we get the SOCP relaxations formulation by using $\|x - c\| \leq r_x$ and $\|y - c\| \leq r_y$.

$$\begin{aligned}
\min \quad & 163x_1 + 55x_2 + 55x_3 - 2x_4 + 10x_5 - 92x_{12} + 565x_{13} + 77x_{14} - 6x_{15} \\
& +5984x_{23} - 22x_{24} + 31x_{25} + 22x_{34} + 2543x_{35} - 76x_{45} \\
s.t. \quad & \left\| \begin{pmatrix} -5.5x_1 - 2x_{12} + 2x_2 - x_{13} + 0.5x_3 + 4x_{14} - 2x_4 + 1.25 \\ 3x_1 - 4.5x_2 - 4x_{12} - x_{23} + 0.5x_3 + 4x_{24} - 2x_4 + 1.25 \\ 3x_1 - 6x_{13} - 4x_{23} + 2x_2 - 3x_3 + 4x_{34} - 2x_4 + 4x_{123} - 2x_{12} + 1.25 \\ -6x_{14} + 3x_1 - 4x_{24} + 2x_2 - x_{34} + 0.5x_3 - 0.5x_4 + 4x_{124} - 2x_{12} + 1.25 \\ -6x_{15} + 3x_1 - 4x_{25} + 2x_2 - x_{35} + 0.5x_3 + 4x_{45} - 2x_4 + 4x_{125} - 2x_{12} - 2.5x_5 + 1.25 \end{pmatrix} \right\| \leq r_x \text{(5.24b)} \\
& \left\| \begin{pmatrix} 0.5x_1 + 2x_{12} - x_2 - x_{13} + 0.5x_3 - x_{14} + 0.5x_4 - x_{15} + 0.5x_5 - 0.5 \\ -x_{12} + 0.5x_1 + 2x_2 - 0.5 - x_{23} + 0.5x_3 - x_{24} + 0.5x_4 - x_{25} + 0.5x_5 \\ -x_{13} + 0.5x_1 + 2x_{23} - x_2 + 0.5x_3 - x_{34} + 0.5x_4 - x_{35} + 0.5x_5 - 0.5 \\ -x_{14} + 0.5x_1 + 2x_{24} - x_2 - x_{34} + 0.5x_3 + 0.5x_4 - x_{45} + 0.5x_5 - 0.5 \\ -x_{15} + 0.5x_1 + 2x_{25} - x_2 - x_{35} + 0.5x_3 - x_{45} + 0.5x_4 + 0.5x_5 - 0.5 \end{pmatrix} \right\| \leq r_x \text{(5.24c)}
\end{aligned}$$

$$\left\| \begin{pmatrix} -10.5x_{12} + 3x_1 + 2x_2 - x_{123} + 0.5x_3 + 4x_{124} - 2x_4 + 1.25 \\ -9.5x_{13} + 3x_1 + 2x_2 + 0.5x_3 + 4x_{134} - 2x_4 - 2x_{12} + 1.25 \\ -4.5x_{14} + 3x_1 + 2x_2 - x_{134} + 0.5x_3 - 2x_4 - 2x_{12} + 1.25 \\ -8.5x_{15} + 3x_1 + 2x_2 - x_{135} + 0.5x_3 + 4x_{145} - 2x_4 - 2x_{12} + 1.25 \\ -2x_{123} + 3x_1 - 7.5x_{23} + 2x_2 + 0.5x_3 + 4x_{234} - 2x_4 - 2x_{12} + 1.25 \\ -2x_{124} + 3x_1 - 2.5x_{24} + 2x_2 - x_{234} + 0.5x_3 - 2x_4 - 2x_{12} + 1.25 \\ -2x_{125} + 3x_1 - 6.5x_{25} + 2x_2 - x_{235} + 0.5x_3 + 4x_{245} - 2x_4 - 2x_{12} + 1.25 \\ -6x_{134} + 3x_1 - 4x_{234} + 2x_2 + 0.5x_{34} + 0.5x_3 - 2x_4 + 4x_{1234} - 2x_{12} + 1.25 \\ -6x_{135} + 3x_1 - 4x_{235} + 2x_2 - 3.5x_{35} + 0.5x_3 + 4x_{345} - 2x_4 + 4x_{1235} - 2x_{12} + 1.25 \\ -6x_{145} + 3x_1 - 4x_{245} + 2x_2 - x_{345} + 0.5x_3 + 1.5x_{45} - 2x_4 + 4x_{1245} - 2x_{12} + 1.25 \end{pmatrix} \right\| \leq r_y \quad (5.24b)$$

$$\left\| \begin{pmatrix} 2x_{12} + 0.5x_1 - x_2 - x_{123} + 0.5x_3 - x_{124} + 0.5x_4 - x_{125} + 0.5x_5 - 0.5 \\ -x_{13} + 0.5x_1 + 2x_{123} - x_2 + 0.5x_3 - x_{134} + 0.5x_4 - x_{135} + 0.5x_5 - 0.5 \\ -x_{14} + 0.5x_1 + 2x_{124} - x_2 - x_{134} + 0.5x_3 + 0.5x_4 - x_{145} + 0.5x_5 - 0.5 \\ -x_{15} + 0.5x_1 + 2x_{125} - x_2 - x_{135} + 0.5x_3 - x_{145} + 0.5x_4 + 0.5x_5 - 0.5 \\ -x_{123} + 0.5x_1 + 2x_{23} - x_2 + 0.5x_3 - x_{234} + 0.5x_4 - x_{235} + 0.5x_5 - 0.5 \\ -x_{124} + 0.5x_1 + 2x_{24} - x_2 - x_{234} + 0.5x_3 + 0.5x_4 - x_{245} + 0.5x_5 - 0.5 \\ -x_{125} + 0.5x_1 + 2x_{25} - x_2 - x_{235} + 0.5x_3 - x_{245} + 0.5x_4 + 0.5x_5 - 0.5 \\ -x_{134} + 0.5x_1 + 2x_{234} - x_2 - x_{34} + 0.5x_3 + 0.5x_4 - x_{345} + 0.5x_5 - 0.5 \\ -x_{135} + 0.5x_1 + 2x_{235} - x_2 - x_{35} + 0.5x_3 - x_{345} + 0.5x_4 + 0.5x_5 - 0.5 \\ -x_{145} + 0.5x_1 + 2x_{245} - x_2 - x_{345} + 0.5x_3 - x_{45} + 0.5x_4 + 0.5x_5 - 0.5 \end{pmatrix} \right\| \leq r_y \quad (5.24c)$$

RLT constraints,

where $r_x = \dfrac{\sqrt{5}}{2}$, $r_y = \dfrac{\sqrt{10}}{2}$. Solving this SOCP relaxation problem, we get the lower bound is -2 with optimal solution $(x_4, x_{1234}, x_{1235}, x_{1245}) = (1, 0.3471, 0.2992, 0.3672)$ and all the other variables equal to 0. From Table 5.1, we see that the SOCP relaxation gives the best lower bound which even equal to f_{opt}. This will lead us to find the optimal solution more efficiently. Actually in this example, the value of $x_1, \cdots, x_5$ in the solution of SOCP relaxation is already an optimal solution.

Remark 4. Noting that, for 0-1 linear constraints problem, there is no x_{ij} in the linear constraints, so adding RLT constraints $f_d \geq 0$ will not tighter the feasible region. Hence, in this case the feasible region of linear relaxation(LR) is equivalent with proj(RLT$_{\text{weak}}$). For SDP, it would be the same. RLT$_{\text{weak}}$, SDP$_{\text{aug}}$ and RLT$_{\text{weak}}$+SDP$_{\text{aug}}$'s projection will not reduce the feasible region of linear relaxation. But, with quadratic

Table 5.1: Lower bounds obtained at root node for Example 1

f_{opt}	RLT	RLT+TI	RLT+MINT-1	RLT+SDP	SOCP
-2	-36.9375	-35.5625	-27.5	-36.2925	-2

objective function, it does influence the lower bound since there are x_{ij} in objective function which will be affected.

Property 8. *When $n = 2$, the RLT constraints $f_2 \geq$ are tighter than $X \succeq 0$.*

Proof. For $X \succeq 0$,if x_1 or x_2 equals zero, f_2 implies $\begin{pmatrix} x_1 & x_{12} \\ x_{12} & x_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & x_2 \end{pmatrix}$ or $\begin{pmatrix} x_1 & 0 \\ 0 & 0 \end{pmatrix}$. Apparently, eigen values are non-negative, then PSD. if $x_1 \neq 0$, $\begin{pmatrix} x_1 & x_{12} \\ x_{12} & x_2 \end{pmatrix} \longrightarrow \begin{pmatrix} x_1 & x_{12} \\ 0 & x_2 - \frac{x_{12}^2}{x_1} \end{pmatrix} = \begin{pmatrix} x_1 & x_{12} \\ 0 & \frac{x_1 x_2 - x_{12}^2}{x_1} \end{pmatrix}$, eigen values are nonnegative, hence PSD. □

The following Examples 8 and 9 show the tightness of SOCP relaxations.

Example 8. *From Burer and Chen [24]*

$$x_1 + 2x_2 \leq 2.5 \tag{5.26a}$$

$$3x_1 + x_2 \leq 2.5 \tag{5.26b}$$

$$x_1, x_2 \in \{0, 1\}. \tag{5.26c}$$

Actually, the first constraint (5.26a) is redundant in this example, so we simplify it to

$$3x_1 + x_2 \leq 2.5$$

$$x_1, x_2 \in \{0, 1\}.$$

- **RLT$_{\text{weak}}$**

$$\left.\begin{array}{c} 3x_1 + x_2 \leq 2.5 \\ x_{12} \leq x_1 \\ x_{12} \leq x_2 \\ 1 - x_1 - x_2 + x_{12} \geq 0 \\ x_{12} \geq 0 \end{array}\right\} \; f_2 \geq 0$$

- **SOCP**

$$\left\| (2.5 - 3x_1 - x_2) \begin{pmatrix} x_1 - 1/2 \\ x_2 - 1/2 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(2.5 - 3x_1 - x_2)$$

$$\|(2.5 - 3x_1 - x_2)(x_{12} - 1/2)\| \leq \frac{1}{2}(2.5 - 3x_1 - x_2)$$

$$f_2 \geq 0$$

$\longrightarrow$

$$\left\| \begin{pmatrix} -1.25 + x_1 - x_{12} + .5x_2 \\ 2x_2 - 1.25 - 3x_{12} + 1.5x_1 \end{pmatrix} \right\| \leq \frac{\sqrt{2}}{2}(2.5 - 3x_1 - x_2)$$

$$\|-1.5x_{12} - 1.25 + 1.5x_1 + 0.5x_2\| \leq \frac{1}{2}(2.5 - 3x_1 - x_2)$$

$$f_2 \geq 0$$

- **SDP$_{\text{aug}}$+RLT$_{\text{weak}}$**

$$\text{RLT}_{\text{weak}}$$
$$\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0$$

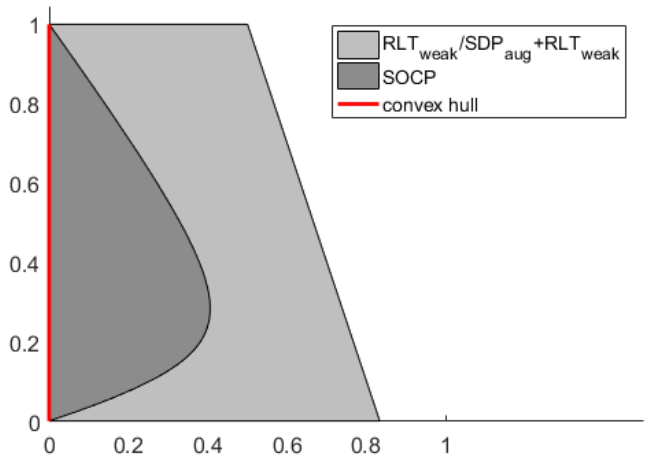


Figure 5.3: Projection of Example 8

Since $\text{RLT}_{\text{strong}}$ gives convex hull(red line) in this example, SDP_{aug}+$\text{RLT}_{\text{strong}}$ also gives convex hull.

Example 9.

$$x_1 + 2x_2 \leq 2.5$$

$$x_1, x_2 \in \{0, 1\}$$

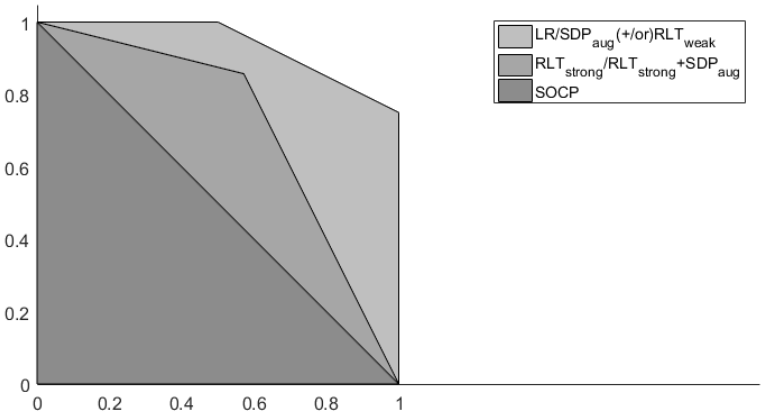


Figure 5.4: Projection of Example 9

5.2.2 0-1 QPs

We test 0-1 QP problems with linear constraints in this section. Table 5.2 lists the three strategies we considered. And we compare these different methods on 0-1 QP problems with $n = 20, 30, 40, 50$. To reduce the computational time, constrains (5.17b) is not considered in SOCP relaxation and we call the simple version as $\text{SOCP}_{\text{weak}}$. Table 5.3-5.6 show the performance. From the results we can see that even we use the simplified version of SOCP relaxation, the lower bound it gives is much better than RLT_{weak}. We note that while the proposed SOCP method works better than SDP_{aug}+RLT_{weak}, especially for the instances of $n = 20, 30$ and 50, the average of lower bounds is indeed weaker when $n = 40$. However, merely looking at the average value might not be a good computational comparison metric as there are more than half of the instances where the SOCP method provides tighter lower bounds as compared with SDP_{aug}+RLT_{weak} even for the case when $n = 40$. In this case, while the $\text{SOCP}_{\text{weak}}$ method does not yield a better average lower bound, it does provide better solutions for several instances, and is in general a computationally superior alternative. Figure 5.5 gives box-plots of SDP_{aug}+RLT_{weak}'s and $\text{SOCP}_{\text{weak}}$'s %improvement over RLT_{weak}, in which Figure 5.5a is for 0-1 QPs with $n = 20$ and Figure 5.5b is for 0-1 QPs with $n = 30$. The box-plots show that our proposed $\text{SOCP}_{\text{weak}}$ could improve the lower bound dramatically. For example, for $n = 30$, 75% of instances applied $\text{SOCP}_{\text{weak}}$ approach improve more than half instances with SDP_{aug}+RLT_{weak} relaxations.

Table 5.2: Description of different strategies

Strategy	Relaxation Technique
RLT_{weak}	linear constraints(structured constraints) + $f_2 \geq 0$
$\text{SOCP}_{\text{weak}}$	LP$\times$$||x - c||$ + $f_2 \geq 0$
SDP_{aug}+RLT_{weak}	RLT_{weak} + $\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0$

Table 5.3: Computational results of 0-1 QPs with $n = 20$

Instances	RLT_{weak}		SDP_{aug}+RLT_{weak}		$SOCP_{weak}$	
	LowerBound	Time(s)	LowerBound	Time(s)	LowerBound	Time(s)
qp20_10_1_1	-70.061245	0.004813	-70.061244	0.027298	-67.392509	0.029769
qp20_10_1_2	-82.495270	0.004540	-82.495269	0.033142	-79.947969	0.053317
qp20_10_1_3	-38.820370	0.004182	-38.490543	0.037261	-37.773701	0.048457
qp20_10_1_4	0.000000	0.002953	0.000000	0.028876	0.000000	0.042909
qp20_10_2_1	-16.460975	0.003398	-16.460975	0.027020	-16.460975	0.028591
qp20_10_2_2	-115.298795	0.004293	-115.298794	0.032547	-111.873093	0.138289
qp20_10_2_3	-59.554698	0.005303	-56.730154	0.069587	-48.962318	0.046997
qp20_10_2_4	0.000000	0.003110	0.000000	0.027048	0.000000	0.041930
qp20_10_3_1	-106.720690	0.006178	-102.667108	0.068561	-101.389908	0.032907
qp20_10_3_2	0.000000	0.003629	0.000000	0.027731	0.000000	0.033248
qp20_10_3_3	-11.663819	0.004507	-11.626702	0.042657	-11.266431	0.047761
qp20_10_3_4	-77.001506	0.005097	-73.538635	0.042110	-71.445649	0.050260
qp20_10_4_1	-3.378062	0.004260	0.000000	0.028927	-0.217845	0.036913
qp20_10_4_2	-46.379898	0.005330	-46.379898	0.050886	-43.750085	0.046023
qp20_10_4_3	0.000000	0.003735	0.000000	0.032394	0.000000	0.046028
qp20_10_4_4	0.000000	0.003568	0.000000	0.031781	0.000000	0.037262
AVERAGE	-39.239708	0.004306	-38.35933263	0.037989125	-36.90503019	0.047541313

Table 5.4: Computational results of 0-1 QPs with $n = 30$

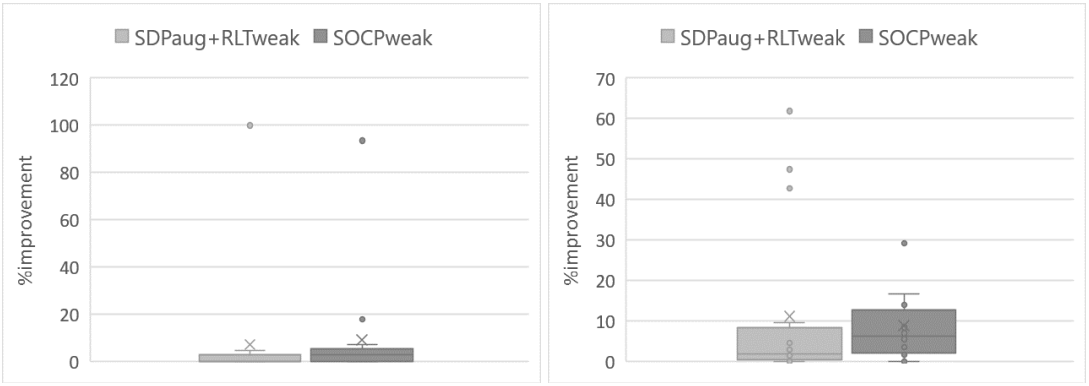
Instances	RLT_{weak}		SDP_{aug}+RLT_{weak}		$SOCP_{weak}$	
	LowerBound	Time(s)	LowerBound	Time(s)	LowerBound	Time(s)
qp30_15_1_1	-160.129948	0.012063	-158.884503	0.190298	-153.754138	0.113243
qp30_15_1_2	0.000000	0.006506	0.000000	0.091764	0.000000	0.110689
qp30_15_1_3	-187.995751	0.009561	-185.981990	0.120218	-179.722934	0.207399
qp30_15_1_4	-97.523185	0.009829	-97.514136	0.109010	-89.004918	0.118074
qp30_15_2_1	-79.659538	0.009776	-77.546526	0.193630	-78.345291	0.111936
qp30_15_2_2	-230.775015	0.013271	-229.409593	0.183760	-222.560816	0.104698
qp30_15_2_3	-32.351872	0.009517	-30.845602	0.120204	-26.975276	0.173357
qp30_15_2_4	-158.442072	0.011245	-157.782326	0.297809	-147.314398	0.193476
qp30_15_3_1	-3.495053	0.006790	-1.332902	0.135661	-3.465412	0.111704
qp30_15_3_2	-237.592408	0.016769	-232.042732	0.192355	-224.433906	0.119200
qp30_15_3_3	0.000000	0.007575	0.000000	0.102097	0.000001	0.117600
qp30_15_3_4	-137.925989	0.011092	-135.964333	0.126718	-126.261218	0.246459
qp30_15_4_1	-149.472244	0.015056	-85.506447	0.214066	-105.638736	0.116903
qp30_15_4_2	-35.504715	0.014729	-18.643841	0.237907	-25.110042	0.136231
qp30_15_4_3	-197.496096	0.021643	-178.573574	0.203460	-169.988455	0.102040
qp30_15_4_4	-197.503879	0.014958	-191.542214	0.252699	-178.762309	0.173860
AVERAGE	-119.1167353	0.01189875	-111.3481699	0.1732285	-108.2086155	0.141054313

Table 5.5: Computational results of 0-1 QPs with $n = 40$

Instances	RLT_{weak}		SDP_{aug}+RLT_{weak}		$SOCP_{weak}$	
	LowerBound	Time(s)	LowerBound	Time(s)	LowerBound	Time(s)
qp40_20_1_1	-43.035736	0.010870	-37.726672	0.434022	-40.978083	0.361471
qp40_20_1_2	-91.104232	0.016677	-87.393718	0.621533	-82.020534	0.414885
qp40_20_1_3	-81.738170	0.019616	-79.976664	0.627400	-67.003087	0.354337
qp40_20_1_4	-162.183293	0.018896	-162.003044	0.437781	-154.822296	0.413619
qp40_20_2_1	-103.922444	0.018949	-88.674035	0.461470	-90.472815	0.691900
qp40_20_2_2	-0.419693	0.010741	-0.419693	0.347877	0.000001	0.456269
qp40_20_2_3	-133.375727	0.022042	-130.205467	0.639962	-121.104991	0.440256
qp40_20_2_4	-120.320724	0.026378	-117.563173	0.579984	-111.967359	0.392664
qp40_20_3_1	-32.004825	0.017693	-8.520082	0.628396	-21.184559	0.415925
qp40_20_3_2	-41.799748	0.024106	-37.180180	0.526649	-33.993087	0.506150
qp40_20_3_3	-34.005474	0.032103	-24.787509	0.588864	-25.393986	0.466161
qp40_20_3_4	-179.102612	0.019341	-175.916893	0.538522	-158.790281	0.653605
qp40_20_4_1	-476.335780	0.030528	-343.759349	0.577302	-425.739670	0.722439
qp40_20_4_2	-85.009010	0.039583	-55.815651	0.808613	-63.159665	0.460852
qp40_20_4_3	-185.240897	0.058666	-125.197745	0.803495	-134.080832	0.442250
qp40_20_4_4	0.000000	0.018477	0.000000	0.351852	0.000001	0.337927
AVERAGE	-110.5998978	0.024041625	-92.19624219	0.560857625	-95.66945269	0.470669375

Table 5.6: Computational results of 0-1 QPs with $n = 50$

Instances	RLT_{weak}		SDP_{aug}+RLT_{weak}		$SOCP_{weak}$	
	LowerBound	Time(s)	LowerBound	Time(s)	LowerBound	Time(s)
qp50_25_1_1	-10.252837	0.016539	-10.252837	0.786501	-10.252833	1.825192
qp50_25_1_2	-192.674914	0.027804	-191.858575	1.259373	-175.942945	1.979557
qp50_25_1_3	-317.602990	0.039455	-312.671312	1.834495	-302.096902	1.507945
qp50_25_1_4	-75.924848	0.026251	-75.913520	1.049033	-68.538413	1.032247
qp50_25_2_1	-229.227794	0.043051	-217.368415	1.472974	-209.899102	1.851333
qp50_25_2_2	0.000000	0.013725	0.000000	0.866043	0.000000	1.887637
qp50_25_2_3	-248.588587	0.037715	-246.493044	0.988725	-219.120790	1.194912
qp50_25_2_4	-23.333405	0.040335	-3.629367	2.120042	-2.608868	1.092729
qp50_25_3_1	-103.151601	0.048384	-73.455247	1.449792	-83.323912	1.881942
qp50_25_3_2	-532.647235	0.053772	-528.989053	1.512796	-506.761957	1.852431
qp50_25_3_3	-235.180322	0.047933	-222.734231	1.758262	-206.459754	0.906167
qp50_25_3_4	-25.030288	0.039199	-9.275723	1.592056	-8.388898	1.334384
qp50_25_4_1	-33.744645	0.035943	-2.257290	0.952927	-13.055601	2.567303
qp50_25_4_2	-89.883704	0.101939	-35.448692	1.909412	-41.662103	1.592323
qp50_25_4_3	-83.177285	0.111645	-34.641192	1.553501	-28.142123	0.808318
qp50_25_4_4	0.000000	0.025468	0.000000	0.902945	0.000001	1.057911
AVERAGE	-137.5262784	0.044322375	-122.8117811	1.375554813	-117.2658875	1.523270688



(a) box-plot for 0-1 QPs with $n = 20$ (b) box-plot for 0-1 QPs with $n = 30$

Figure 5.5: Box-plots of %improvement over RLT_{weak} for 0-1 QPs

5.3 Conclusions

In this chapter, we construct an SOCP procedure to solve 0-1 LPs, then solve 0-1 QCQPs . From the computational results, we can see that the SOCP-based relaxation technique gives a very good bound. With the degree's increasing, the bound will becomes better, and at the same time it also would cost more time in computation. With level-d's reaching to n, this SOCP procedure will give the convex hull. However, it will be a complicated formulation. In the experimental examples, we use a more relaxed relaxation-$\text{SOCP}_{\text{weak}}$ rather that SOCP-1 relaxation to reduce the burden of time consumption. But in this situation, we cannot make sure that the $\text{SOCP}_{\text{weak}}$ procedure is still convergent to convex hull. In the future, we will try to simplify SOCP procedure and without loss of its convergence. Furthermore, this technique also could be extended to polynomial programming by linearizing the problem first and then applying the relaxation.

6 Summary and Conclusions

The goal of this research is to improve the solvability of QCQPs (as defined in (1.1)) by developing theoretical, algorithmic, and computational advancements that enhance and complement existing relaxation techniques and algorithms. As aforementioned, most global optimization approaches for nonconvex QCQPs are based on constructing a convex or linear relaxation of the problem, and embedding it within an exhaustive search mechanism, in which a lower bound (LB) is calculated as the value of the objective function of this relaxation at each iteration of the search tree. The two critical elements affecting the overall effectiveness of such exhaustive search methods are the tightness of the underlying relaxation (which governs the efficiency in computing the lower bound), and the method used to select a branching node and branching variable, which leads to a good quality feasible solution or equivalently an upper bound (UB). Moreover, as noted earlier, the problem of finding even a single feasible solution is also NP-hard, for the general case of a nonconvex QCQP.

To improve the quality of the obtained lower bounds, we pursue two research directions that are based on very different modeling constructs. The first is to add appropriately defined *valid inequalities*, such as the triangle inequalities (1.16), or other types of feasibility and optimality conditions that are based on the underlying properties of the convex hull of feasible 0-1 integer points. The other research direction is to design *cut generation schemes* based on extended concepts such as semidefinite programming, akin to the one presented by Sherali and Fraticelli [63], which utilize the information gleaned from the current (infeasible) lower bounding solution to generate cutting planes and embed this mechanism into an iterative scheme for improving lower bounds.

While both these approaches have been successful with varying degrees in

practice, they serve to only improve the lower bound, and do not address the issue of obtaining feasible upper bounding solutions relatively quickly in the search process. In our research, recognizing that a tightening of the LB-UB gap rapidly improves the convergence of the search tree, we designed mechanisms that improve both lower and upper bounds simultaneously, thereby significantly improving the efficiency of branch-and-bound algorithms.

More specifically, in the spirit of the triangle inequalities (1.16), we enhanced existing linearization techniques (notably RLT-based methods) for solving 0-1 QCQPs with the new class of valid cuts, which we refer to as the Minimum Triangle (MINT) inequalities. These MINT cuts were derived based on the relationship that $y_{ij} = \min\{x_i, x_j\}$ at any feasible solution (and hence, at optimality), and we prove that these super-linear cuts are tighter than the existing triangle inequalities, thereby leading to improved lower bounds at each node of the branch-and-bound tree. Furthermore, we extended the logic used in defining these minimum triangle inequalities to construct a novel variable fixing and affiliated branching strategy (within a branch-and-bound scheme) that identifies a feasible solution relatively early on in the search process while expending a very minimal computational cost. As existing large-scale commercial LP solvers, such as Gurobi [33], can determine the optimal solution to an MILP very efficiently if a feasible solution is provided as an input, this branching strategy significantly enhances the solvability of large-scale 0-1 QCQPs. The derived MINT cuts in concert with the newly designed variable fixing and branching strategy gives rise to a branch-and-bound algorithm that we refer to as the Minimum Triangle Algorithm (MINT Algorithm), and we demonstrate the efficacy of this algorithm in finding a feasible solution (and subsequently the optimal solution) for standard QP test instances obtained from the literature. Our computational results show that the number of nodes required in determining the optimal solution as well as the computation time

to obtain the (global) optimum is greatly reduced, and overall, the MINT algorithm performs exceedingly well in comparison with existing methods.

In our second contribution, using the property that for a PSD matrix X, the Frobenius inner product $H \bullet X \geq 0, \forall H \in \mathbb{S}^n_+$, we design an inductive polynomial time mechanism to construct a PSD matrix H that yields $H \bullet \overline{X}_L < 0$ whenever $\overline{X}_L$ is not PSD, and impose the cut $H \bullet X_L \geq 0$, where X_L is as defined in (1.9). Several variations of this cut generation scheme are presented, and in particular, we note that generating a single higher-rank ordered cut performs efficiently as compared to multiple rank-one cuts.

Finally, noting that a 0-1 QCQP can be transformed into an equivalent 0-1 MIP, we propose an SOCP-based procedure for obtaining a tight lower bound at a relatively low computational cost. Theoretically, we show the convergence of the SOCP-based procedure. In computational experiments, for the large size problems, we only test on a relaxed SOCP level 1 relaxation - $\text{SOCP}_{\text{weak}}$ rather than utilizing SOCP-1 directly to reduce the computational burden. It would be much more complex when $d \geq 2$ for the SOCP procedure. We will dig out more features of the proposed SOCP procedure according to the degree d and trying to construct a simplified SOCP procedure without loss of it convergence. For future research, note that all of the algorithm advancements proposed in this thesis can be readily extended towards 0-1 multilinear programs and the more general class of polynomial programs. To illustrate, consider the case of 0-1 mixed integer QCQPs. WLOG, assuming that all continuous variables are bounded in [0, 1], we have that for $x_i \in \{0, 1\}, x_j \in [0, 1]$, $y_{ij} = x_i x_j = \min\{x_i, x_j\}$ holds true even in this case. For order $d = 3$, when there are at least two binary variables in x_i, x_j, and x_k, $y_{pq} = \min\{x_p, x_q\}$ is satisfied for any pair $\{p, q\} \in \{i, j, k\}$, and consequently, all of the proposed MINT inequalities are valid. However, to apply these cuts towards polynomial programs of higher degree, a carefully designed research framework must

be developed, and these investigations are currently underway. From the perspective of application, our proposed approaches could be utilized on a lot of real-world problems that conform to QCQPs, such as signal processing, production planning and industry design problems. In the process of solving real-life problems, some specific conditions or constraints could be added depending on the particular realistic problem, which should be considered carefully.

A Appendix

A.1 Some useful relations

$$x_i \in \{0,1\} \tag{A.1a}$$

$$y_{ii} = x_i \tag{A.1b}$$

$$y_{ij} = x_i x_j \tag{A.1c}$$

$$y_{ij} = \min\{x_i, x_j\} \tag{A.1d}$$

$$\min\{x_i, x_j\} = \frac{1}{2}\{x_i + x_j - |x_i - x_j|\} \tag{A.1e}$$

$$\min\{x_i, x_j\} + \max\{x_i, x_j\} = x_i + x_j \tag{A.1f}$$

$$0 \leq |x_i - x_j| \leq 1 \tag{A.1g}$$

$$|x_i - x_j| \leq x_i + x_j \tag{A.1h}$$

$$|x_i - x_j| \leq |x_i - x_k| + |x_k - x_j| \tag{A.1i}$$

A.2 MINT cuts with order 3

MINT-2 $y_{ij} - y_{ik} - y_{jk} \geq \min\{x_i, x_j\} - \min\{x_i, x_k\} - \min\{x_j, x_k\} \Rightarrow$

- $$\begin{aligned} y_{ij} - y_{ik} - y_{jk} &\geq \frac{1}{2}\{x_i + x_j - x_i - x_k - x_j - x_k - |x_i - x_j| + |x_i - x_k| + |x_j - x_k|\} \\ &\geq \frac{1}{2}\{x_k - x_i + x_k - x_j - x_i - x_j\} - x_k \\ &= -x_i - x_j \end{aligned}$$

  $\Rightarrow \underline{y_{ij} - y_{ik} - y_{jk} \geq -x_i - x_j}.$

- $y_{ij} - y_{ik} - y_{jk} \geq x_i + x_j - 2x_k - \max\{x_i, x_j\}$ (According to (3.10) from MINT-2)

  $\geq x_i + x_j - 2x_k - 1$

  $\Rightarrow \underline{1 + y_{ij} - y_{ik} - y_{jk} \geq x_i + x_j - 2x_k}.$

MINT-3 $y_{ij} - y_{ik} - y_{jk} \leq \min\{x_i, x_j\} - \min\{x_i, x_k\} - \min\{x_j, x_k\} \Rightarrow$

- $y_{ij} - y_{ik} - y_{jk} \leq -2x_k + \max\{x_i, x_k\} + \max\{x_j, x_k\} - \max\{x_i, x_j\}$

  (According to (3.13) from MINT-3)

  $\leq -2x_k + 2$

  $= 2(1 - x_k)$

  $\Rightarrow \underline{y_{ij} - y_{ik} - y_{jk} \leq 2(1 - x_k)}.$

- $y_{ij} - y_{ik} - y_{jk} \leq |x_j - x_k| - x_k$(According to (3.14) from MINT-3)

  $\leq x_j$

  $\Rightarrow \underline{y_{ij} - y_{ik} - y_{jk} \leq x_j}.$

- $y_{ij} - y_{ik} - y_{jk} \leq \frac{1}{2}\{|x_i - x_k| + |x_j - x_k| - x_i + x_j\} - x_k$

  (According to (3.15) from MINT-3)

  $\leq \frac{1}{2}\{x_i + x_k + x_j + x_k - x_i + x_j\} - x_k$

$$= x_j$$

$\Rightarrow \underline{y_{ij} - y_{ik} - y_{jk} \leq x_j}$. (Same as the one before).

MINT-4 $y_{ij} + y_{ik} - y_{jk} \geq \min\{x_i, x_j\} - \min\{x_j, x_k\} \Rightarrow$

- $$\begin{aligned} y_{ij} + y_{ik} - y_{jk} &\geq \frac{1}{2}\{x_i + x_j - (x_j + x_k) - |x_i - x_j| + |x_j - x_k|\} \\ &\geq \frac{1}{2}\{x_i - x_k - (x_i + x_j) + |x_j - x_k|\} \\ &= -\frac{1}{2}\{x_j + x_k - |x_j - x_k|\} \\ &= -\min\{x_j, x_k\} \\ &\geq -1 \end{aligned}$$

  $\Rightarrow \underline{y_{ij} + y_{ik} - y_{jk} \geq -1}$.

- $$\begin{aligned} y_{ij} + y_{ik} - y_{jk} &\geq x_i - \max\{x_i, x_k\} \quad \text{(According to(3.17) from MINT-4)} \\ &\geq x_i - 1 \end{aligned}$$

  $\Rightarrow \underline{1 + y_{ij} + y_{ik} - y_{jk} \geq x_i}$.

- $$\begin{aligned} y_{ij} + y_{ik} - y_{jk} &\geq x_i + x_j - x_k - \max\{x_i, x_j\} \quad \text{(According to(3.18) from MINT-4)} \\ &\geq x_i + x_j - x_k - 1 \end{aligned}$$

  $\Rightarrow \underline{1 + y_{ij} + y_{ik} - y_{jk} \geq x_i + x_j - x_k}$.

MINT-5 $y_{ij} + y_{ik} + y_{jk} \geq \min\{x_i, x_j\} + \min\{x_i, x_k\} + \min\{x_j, x_k\} \Rightarrow$

- $y_{ij} + y_{ik} + y_{jk} \geq x_i + x_j + x_k - \frac{1}{2}\{|x_i - x_j| + |x_i - x_k| + |x_j - x_k|\}$

  (Using the fact $\frac{1}{2}\{|x_i - x_j| + |x_j - x_k| + |x_i - x_k|\} \leq 1$)

  $\geq x_i + x_j + x_k - 1$

$\Rightarrow \underline{1 + y_{ij} + y_{ik} + y_{jk} \geq x_i + x_j + x_k}$.

A.3 MINT cuts with order 4

MINT-4.1 $y_{ij} + y_{jk} + y_{kl} = \min\{x_i, x_j\} + \min\{x_j, x_k\} + \min\{x_k, x_l\}$

$= x_j + x_k + \frac{1}{2}\{(x_i + x_l) - |x_i - x_j| - |x_j - x_k| - |x_k - x_l|\}$

- RHS $\leq x_j + x_k + \frac{1}{2}\{(x_i + x_l) - |x_i - x_l|\}$

  (using $|x_i - x_j| + |x_j - x_k| + |x_k - x_l| \geq |x_i - x_l|$)

  $= x_j + x_k + y_{il}$

- RHS $\leq x_j + x_k + \frac{1}{2}\{(x_i + x_l - |x_i - x_k| - |x_k - x_l|)\}$

  (using $|x_i - x_j| + |x_j - x_k| \geq |x_i - x_k|$)

  $= x_j + y_{ik} + y_{kl}$

MINT-4.2 $y_{ij} - y_{jk} + y_{kl} = \min\{x_i, x_j\} - \min\{x_j, x_k\} + \min\{x_k, x_l\}$

$= \frac{1}{2}\{(x_i + x_l) - |x_i - x_j| + |x_j - x_k| - |x_k - x_l|\}$

- RHS $\geq \frac{1}{2}\{(x_i + x_l) + |x_i + x_k| - |x_k - x_l|\}$

  (using $|x_j - x_k| - |x_i - x_j| \geq |x_i - x_k|$)

  $= \frac{1}{2}\{-(x_i + x_k) + |x_i + x_k| + (x_k + x_l) - |x_k - x_l|\} + x_i$

$$= -y_{ik} + y_{kl} + x_i.$$

A.4 Proof of Claim

Claim $[[f]_L \cdot [g]_L]_L = [f \cdot g]_L$, *where* f *and* g *are polynomial regarding binary variables.*

Proof. We start from both f and g are monomial. Since coefficients will not influence the linearization operation, we ignore coefficients in proof for convenience. Assume that $f = \prod_{i \in J_1} x_i$ and $g = \prod_{i \in J_2} x_i$. Noting that due to x is binary, in linearization operation $[\bullet]_{\mathrm{L}}$, we can substitute x_p^2 with x_p.

$$\begin{aligned}
[[f]_{\mathrm{L}} \cdot [g]_{\mathrm{L}}]_{\mathrm{L}} =& [[\prod_{i \in J_1} x_i]_{\mathrm{L}} \cdot [\prod_{i \in J_2} x_i]_{\mathrm{L}}]_{\mathrm{L}} \\
=& [x_{\{J_1\}} \cdot x_{\{J_2\}}]_{\mathrm{L}} \\
=& x_{\{J_1 \cup J_2\}}
\end{aligned}
\qquad
\begin{aligned}
[f \cdot g]_{\mathrm{L}} =& [\prod_{i \in J_1} x_i \cdot \prod_{i \in J_2} x_i]_{\mathrm{L}} \\
=& [\prod_{i \in \{J_1 + J_2\}} x_i]_{\mathrm{L}} \\
& (\{J_1 + J_2\} \text{ allows repetitive elements}) \\
=& x_{\{J_1 \cup J_2\}}
\end{aligned}$$

Now we extend to polynomial situation. Assume that $f = \prod_{i \in J_{f1}} x_i + \prod_{i \in J_{f2}} x_i + \cdots + \prod_{i \in J_{fm}} x_i$ and $g = \prod_{i \in J_{g1}} x_i + \prod_{i \in J_{g2}} x_i + \cdots + \prod_{i \in J_{gn}} x_i$.

$$\begin{aligned}
[[f]_{\mathrm{L}} \cdot [g]_{\mathrm{L}}]_{\mathrm{L}} =& [[\prod_{i \in J_{f1}} x_i + \cdots + \prod_{i \in J_{fm}} x_i]_{\mathrm{L}} \cdot [\prod_{i \in J_{g1}} x_i + \cdots + \prod_{i \in J_{gn}} x_i]_{\mathrm{L}}]_{\mathrm{L}} \\
=& [([\prod_{i \in J_{f1}} x_i]_{\mathrm{L}} + \cdots + [\prod_{i \in J_{fm}} x_i]_{\mathrm{L}}) \cdot ([\prod_{i \in J_{g1}} x_i]_{\mathrm{L}} + \cdots + [\prod_{i \in J_{gn}} x_i]_{\mathrm{L}})]_{\mathrm{L}} \\
=& [[\prod_{i \in J_{f1}} x_i]_{\mathrm{L}} \cdot [\prod_{i \in J_{g1}} x_i]_{\mathrm{L}} + \cdots + [\prod_{i \in J_{fm}} x_i]_{\mathrm{L}} \cdot [\prod_{i \in J_{gn}} x_i]_{\mathrm{L}}]_{\mathrm{L}} \\
=& [[\prod_{i \in J_{f1}} x_i]_{\mathrm{L}} \cdot [\prod_{i \in J_{g1}} x_i]_{\mathrm{L}}]_{\mathrm{L}} + \cdots + [[\prod_{i \in J_{fm}} x_i]_{\mathrm{L}} \cdot [\prod_{i \in J_{gn}} x_i]_{\mathrm{L}}]_{\mathrm{L}} \\
=& [\prod_{i \in J_{f1}} x_i \cdot \prod_{i \in J_{g1}} x_i]_{\mathrm{L}} + \cdots + [\prod_{i \in J_{fm}} x_i \cdot \prod_{i \in J_{gn}} x_i]_{\mathrm{L}}
\end{aligned}$$

$$=[\prod_{i\in J_{f1}} x_i \cdot \prod_{i\in J_{g1}} x_i + \cdots + \prod_{i\in J_{fm}} x_i \cdot \prod_{i\in J_{gn}} x_i]_{\mathrm{L}}$$

$$=[f \cdot g]_{\mathrm{L}}.$$

Proof completed. □

References

[1] Adams, W.P. and Sherali, H.D. (1986). A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science* 32, 1274–1290.

[2] Adjiman, C., Androulakis, I. and Floudas, C. (1997). Global optimization of MINLP problems in process synthesis and design. *Computers & Chemical Engineering* 21, S445–S450.

[3] ALIDAEE, B., KOCHENBERGER, G.A. and AHMADIAN, A. (1994). 0-1 quadratic programming approach for optimum solutions of two scheduling problems. *International Journal of Systems Science* 25, 401–408.

[4] Alizadeh, F. and Goldfarb, D. (2003). Second-order cone programming. *Mathematical Programming* 95, 3–51.

[5] Anstreicher, K.M., Brixius, N., Goux, J.P. and Linderoth, J. (2002). Solving large quadratic assignment problems on computational grids. *Mathematical Programming* 91, 563–588.

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

[7] Anstreicher, K.M. and Burer, S. (2010). Computable representations for convex hulls of low-dimensional quadratic forms. *Mathematical Programming* 124, 33–43.

[8] Anstreicher, K.M. (2012). On convex relaxations for quadratically constrained quadratic programming. *Mathematical Programming* 136, 233–251.

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

[10] Audet, C., Brimberg, J., Hansen, P., Le Digabel, S. and Mladenović, N. (2004). Pooling problem: Alternate formulations and solution methods. *Management Science* 50, 761–776.

[11] Babayev, D.A. and Mardanov, S.S. (1994). Reducing the number of variables in integer and linear programming problems. *Computational Optimization and Applications* 3, 99–109.

[12] Balas, E., Ceria, S. and Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming* 58, 295–324.

[13] Bao, X., Sahinidis, N.V. and Tawarmalani, M. (2009). Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. *Optimization Methods & Software* 24, 485–504.

[14] Bao, X., Sahinidis, N.V. and Tawarmalani, M. (2011). Semidefinite relaxations for quadratically constrained quadratic programming: a review and comparisons. *Mathematical Programming* 129, 129–157.

[15] Ben-Tal, A., Eiger, G. and Gershovitz, V. (1994). Global minimization by reducing the duality gap. *Mathematical programming* 63, 193–212.

[16] Billionnet, A., Elloumi, S. and Plateau, M.C. (2009). Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discrete Applied Mathematics* 157, 1185–1197.

[17] Billionnet, A., Elloumi, S. and Lambert, A. (2012). Extending the QCR method to general mixed-integer programs. *Mathematical programming* 131, 381–401.

[18] Bomze, I.M., Cheng, J., Dickinson, P.J. and Lisser, A. (2017). A fresh CP look at mixed-binary QPs: new formulations and relaxations. *Mathematical Programming* 1–26.

[19] Brixius, N.W. (2000). *Solving Large-Scale Quadratic Assignment Problems*. Ph.D. thesis The University of Iowa.

[20] Buchheim, C., Caprara, A. and Lodi, A. (2012). An effective branch-and-bound algorithm for convex quadratic integer programming. *Mathematical Programming* 135, 369–395.

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

[22] Burer, S. and Letchford, A.N. (2009). On nonconvex quadratic programming with box constraints. *SIAM Journal on Optimization* 20, 1073–1089.

[23] Burer, S. (2009). On the copositive representation of binary and continuous nonconvex quadratic programs. *Mathematical Programming* 120, 479–495.

[24] Burer, S. and Chen, J. (2009). A p-cone sequential relaxation procedure for 0-1 integer programs. *Optimization Methods & Software* 24, 523–548.

[25] Burer, S. (2015). http://sburer.github.io/projects.html.

[26] Cambini, R. and Sodini, C. (2005). Decomposition Methods for Solving Nonconvex Quadratic Programs via Branch and Bound. *Journal of Global Optimization* 33, 313–336.

[27] Chung, S.J. (1989). NP-completeness of the linear complementarity problem. *Journal of Optimization Theory and Applications* 60, 393–399.

[28] Cplex, I. (2007). 11.0 User's manual. *ILOG SA, Gentilly, France*.

[29] Deza, M.M. and Laurent, M. (1997). *Geometry of cuts and metrics* vol. 15. Springer.

[30] Galli, L. and Letchford, A.N. (2011). Reformulating mixed-integer quadratically constrained quadratic programs. *Technical report, Department of Management Science, Lancaster University*.

[31] Goldfarb, D., Liu, S. and Wang, S. (1991). A logarithmic barrier function algorithm for quadratically constrained convex quadratic programming. *SIAM Journal on Optimization* 1, 252–267.

[32] Grötschel, M. and Wakabayashi, Y. (1989). A cutting plane algorithm for a clustering problem. *Mathematical Programming* 45, 59–96.

[33] Gurobi Optimization, I. (2014). Gurobi Optimizer Reference Manual.

[34] Hansen, P., Jaumard, B. and Savard, G. (1992). New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing* 13, 1194–1217.

[35] Helmberg, C. and Rendl, F. (1998). Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Mathematical Programming* 82, 291–315.

[36] Jarre, F. (1990). On the convergence of the method of analytic centers when applied to convex quadratic programs. *Mathematical Programming* 49, 341–358.

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

[38] Konno, H., Kawadai, N. and Tuy, H. (2003). Cutting plane algorithms for nonlinear semi-definite programming problems with applications. *Journal of Global Optimization* 25, 141–155.

[39] Lasserre, J.B. (2001). An explicit exact SDP relaxation for nonlinear 0-1 programs. In *Integer Programming and Combinatorial Optimization* 293–303 Springer.

[40] Lasserre, J.B. (2001). Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization* 11, 796–817.

[41] Lasserre, J.B. (2002). An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM Journal on Optimization* 12, 756–769.

[42] Lasserre, J.B. (2002). Semidefinite programming vs. LP relaxations for polynomial programming. *Mathematics of Operations Research* 27, 347–360.

[43] Laughhunn, D. (1970). Quadratic binary programming with application to capital-budgeting problems. *Operations Research* 18, 454–461.

[44] Laurent, M. and Rendl, F. (2005). Semidefinite programming and integer programming. *Handbooks in Operations Research and Management Science* 12, 393–514.

[45] Li, H.M. and Zhang, K.C. (2006). A decomposition algorithm for solving large-scale quadratic programming problems. *Applied Mathematics and Computation* 173, 394–403.

[46] Liberti, L. and Pantelides, C.C. (2006). An exact reformulation algorithm for large nonconvex NLPs involving bilinear terms. *Journal of Global Optimization* 36, 161–189.

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

[48] Lovász, L. and Schrijver, A. (1991). Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization* 1, 166–190.

[49] Mehrotra, S. and Sun, J. (1991). A method of analytic centers for quadratically constrained convex quadratic programs. *SIAM Journal on Numerical Analysis* 28, 529–544.

[50] Mitchell, J.E., Pang, J.S. and Yu, B. (2012). Convex quadratic relaxations of nonconvex quadratically constrained quadratic programs. *Optimization Methods and Software* 1–17.

[51] Padberg, M. (1989). The Boolean quadric polytope: some characteristics, facets and relatives. *Mathematical Programming* 45, 139–172.

[52] Pardalos, P.M. (1991). Global optimization algorithms for linearly constrained indefinite quadratic problems. *Computers & Mathematics with Applications* 21, 87–97.

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

[54] Pattillo, J., Veremyev, A., Butenko, S. and Boginski, V. (2013). On the maximum quasi-clique problem. *Discrete Applied Mathematics* 161, 244–257.

[55] Rosen, J. and Pardalos, P. (1986). Global minimization of large-scale constrained concave quadratic problems by separable programming. *Mathematical Programming* 34, 163–174.

[56] Saxena, A., Bonami, P. and Lee, J. (2010). Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. *Mathematical Programming* 124, 383–411.

[57] Saxena, A., Bonami, P. and Lee, J. (2011). Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Mathematical Programming* 130, 359–413.

[58] Sherali, H.D. and Adams, W.P. (1990). A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics* 3, 411–430.

[59] Sherali, H.D. and Alameddine, A. (1992). A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization* 2, 379–410.

[60] Sherali, H.D. and Adams, W.P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zero—one programming problems. *Discrete Applied Mathematics* 52, 83–106.

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

[62] Sherali, H.D. and Tuncbilek, C.H. (1997). New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operations Research Letters* 21, 1–9.

[63] Sherali, H.D. and Fraticelli, B.M. (2002). Enhancing RLT relaxations via a new class of semidefinite cuts. *Journal of Global Optimization* 22, 233–261.

[64] Sherali, H.D. and Desai, J. (2005). On solving polynomial, factorable, and black-box optimization problems using the RLT methodology. In *Essays and Surveys in Global Optimization* 131–163 Springer.

[65] Sherali, H.D. (2007). RLT: A unified approach for discrete and continuous nonconvex optimization. *Annals of Operations Research* 149, 185–193.

[66] Sherali, H.D., Dalkiran, E. and Desai, J. (2012). Enhancing RLT-based relaxations for polynomial programming problems via a new class of v-semidefinite cuts. *Computational Optimization and Applications* 52, 483–506.

[67] Shor, N.Z. (1998). *Nondifferentiable optimization and polynomial problems* vol. 24. Springer.

[68] Vandenbussche, D. and Nemhauser, G.L. (2005). A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming* 102, 559–575.

[69] Vandenbussche, D. and Nemhauser, G.L. (2005). A polyhedral study of nonconvex quadratic programs with box constraints. *Mathematical Programming* 102, 531–557.

[70] Vanderbei, R.J. and Benson, H.Y. (2000). On formulating semidefinite programming problems as smooth convex nonlinear optimization problems. Tech. rep. ORFE-99-01, Princeton University, NJ.

[71] Zheng, X.J., Sun, X.L. and Li, D. (2011). Convex relaxations for nonconvex

quadratically constrained quadratic programming: matrix cone decomposition and polyhedral approximation. *Mathematical Programming* 129, 301–329.

[72] Zheng, X.J., Sun, X.L. and Li, D. (2011). Nonconvex quadratically constrained quadratic programming: best DC decompositions and their SDP representations. *Journal of Global Optimization* 50, 695–712.

[73] Zhou, M. and Chen, W. (2010). Reducing the number of variables in integer quadratic programming problem. *Applied Mathematical Modelling* 34, 424–436.