Noname manuscript No.
(will be inserted by the editor)

Globally Solving Concave Quadratic Program via Doubly Nonnegative Relaxation

Zheng Qu · **Tianyou Zeng** · **Yuchen Lou**

Abstract We consider the problem of maximizing a convex quadratic function over a bounded polyhedral set. We design a new framework based on SDP relaxation and cutting plane method for solving the associated reference value problem. The major novelty is a new way to generate valid cut through the doubly nonnegative (DNN) relaxation. We establish various theoretical properties of the DNN relaxation. This includes its equivalence with the Shor relaxation of the equivalent quadratically constrained problem, the strong duality and generation of valid cut from an approximate solution of the DNN relaxation returned by an arbitrary SDP solver. Computational results on both real and synthetic data demonstrate the efficiency of the proposed new method and its ability to solve high dimensional problems with dense data. In particular, our new algorithm successfully solved in 3 days the reference value problem arising from computational biology for a dataset containing more than 300,000 instances of dimension 100. In contrast, CPLEX or Gurobi is estimated to need years of computational time for the same dataset on the same computing platform.

Keywords Concave quadratic programming · Doubly nonnegative relaxation · Cutting plane method · Strong duality · Valid bound · Large-scale nonconvex programming

Mathematics Subject Classification (2020) 90C20 · 90C22 · 90C26 · 90C59

Z. Qu
Department of Mathematics, the University of Hong Kong.
E-mail: zhengqu@hku.hk

T. Zeng
Department of Mathematics, the University of Hong Kong.
E-mail: logic@connect.hku.hk

Y. Lou
Department of Industrial Engineering and Management Sciences, Northwestern University.
E-mail: yuchenlou2026@u.northwestern.edu

1 Introduction

1.1 Background

We consider the following problem of maximizing a convex quadratic function over a polyhedral set:

$$\begin{aligned} \Phi^*(\mathscr{F}) := \max_{x\in\mathbb{R}^n} \quad & \Phi(x) \\ \text{s.t.} \quad & x \in \mathscr{F}. \end{aligned} \tag{1}$$

Here,

$$\Phi(x) \equiv x^\top Hx + 2p^\top x$$

is a convex quadratic function where $H \in \mathbb{R}^{n\times n}$ is a positive semidefinite matrix. The feasible region $\mathscr{F}$ is a polyhedral set written in standard form:

$$\mathscr{F} := \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}, \tag{2}$$

where $A \in \mathbb{R}^{m\times n}$ is a matrix of rank m. We further assume that $\mathscr{F}$ is nonempty and bounded. Note that problem (1) is equivalent to minimizing a concave quadratic function over a polyhedral set. For this reason, we shall refer to problem (1) as a concave quadratic program (QP).

Concave QP has important applications in many areas, including engineering [9, 22], industry [1,4] and medical diagnosis [7,20]. More recently, concave QP finds new applications in computational biology [14,36]. The biology problem concerned is the detection of undiscovered protein/genome sequence and a key step in the approach proposed in [14,36] consists in solving the following problem:

$$\text{determine whether } \; v_R > \Phi^*(\mathscr{F}) \;\; \text{ or } \;\; \Phi^*(\mathscr{F}) \geq v_R \;\; ? \tag{3}$$

Here, v_R is a certain prescribed reference value. For convenience, we shall refer to (3) as the *reference value problem* associated with the concave QP problem (1). In this real application, the dimension n is the number of known protein/genome sequences of a certain species, and the number of constraints is fixed to $m = 22$. Depending on the species considered, the dimension n or the number of reference value problems may be very large. For example, we have the access to a dataset with 317,584 instances of dimension $n = 100$, and a dataset with 3 instances of dimension $n = 841$. The new protein detection problem requires to solve the reference value problem for **all** the instances in the dataset.

We randomly selected 100 instances from the 317,584 instances of dimension $n = 100$ and tried to solve them with off-the-shelf commercial solvers `CPLEX` and `Gurobi` within 600 seconds. `CPLEX` failed to give an answer to (3) on 92 instances and `Gurobi` failed on 48 instances. The performance of both `CPLEX` and `Gurobi` clearly does not make the grade as the computational time to solve all the 317,584 instances using the best of these two solvers is **at least 3 years**[1]. Moreover, the 3 instances of dimension $n = 841$ can not be handled by `CPLEX` nor by `Gurobi` due to the large dimension.

[1] If half of the instances require at least 600 seconds to solve on average, in total we need at least $600 \times 317584/2 = 95275200$ seconds, which is roughly 3 years.

Motivated by the challenge raised above, we develop in this paper a new method for solving the reference value problem (3) associated with the concave QP problem (1). This new method also naturally extends to a global solver of (1). We focus on the efficiency of the algorithm for large dimensional problems. In particular, the algorithm should be highly efficient so that the 317,584 instances of dimension 100 can be solved in a reasonable amount of time. Moreover, the algorithm should be able to deal with high dimensional problems so that instances of dimension up to 841 can be handled as well.

1.2 Related work

Problem (1) falls into the class of general nonconvex QP problems. There are three major techniques to find a global solution of general QP problems: reformulation, branching and relaxation [18,24]. QP problems can be reformulated in different ways, including bilinear reformulation [16,17], KKT reformulation [3,6], completely positive reformulation [2] and mixed-integer LP reformulation [8,34]. After certain reformulation of the QP according to the problem structures, branch-and-bound (B&B) algorithms are employed in nearly all the state-of-the-art global QP solvers. There exist various branching strategies based on different reformulations [6,34]. Importantly, the efficiency of the bounding step crucially depends on the quality of the relaxations utilized. Well-known relaxations include, for example, McCormick relaxation [21], semidefinite programming (SDP) relaxation [19,23], convex quadratic relaxation by separable programming or DC (difference of convex functions) programming [12], and doubly nonnegative relaxation [2].

Regarding the concave QP problem (1), it has also been richly studied in the literature and is known to be an NP-hard problem [26]. Traditional methods mainly include enumerative methods [5], cutting plane methods [17,31], successive approximation methods (see Chapter 6 in [13]), and branch-and-bound methods [3,6]. Details and references for early work on concave QP can be found in the survey paper of Pardalos and Rosen [25] and in the book of Horst and Tuy [13]. More recently, Zamani [35] proposed a method which combines the cutting plane method and the branch-and-bound method for concave QP. Telli et al. [29] proposed to approximate the global optimum by solving a sequence of linear programs constructed from a local approximation set. Hladik et al. [10,11] proposed new bounds for concave QP based on factorization.

1.3 Doubly nonnegative relaxation

We recall the doubly nonnegative relaxation of (1). A square matrix is completely positive (CP) if it can be written as $BB^\top$ where B is a matrix with nonnegative elements. Denote by $\mathscr{C}$ the cone of n-by-n completely positive matrices:

$$\mathscr{C} = \left\{ BB^\top : B \in \mathbb{R}^{n\times k}_+, k \in \mathbb{N} \right\}.$$

It is known that problem (1) can be reformulated as a linear program over the cone $\mathscr{C}$.

Theorem 1 ([2]) *The quadratic program* (1) *is equivalent to*

$$\begin{aligned} \max \quad & \langle H, X\rangle + 2p^\top x \\ \text{s.t.} \quad & Ax = b \\ & \operatorname{diag}(AXA^\top) = b \circ b \\ & \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \in \mathscr{C}. \end{aligned} \tag{4}$$

The CP cone $\mathscr{C}$ can be approximated from the outside by a hierarchy of cones of linear- and semidefinite-representable cones $\mathscr{D}_0 \supset \mathscr{D}_1 \supset \cdots \supset \mathscr{C}$, with $\mathscr{D}_0$ corresponding to the cone of doubly nonnegative (DNN) matrices, i.e., matrices that are positive semidefintie and nonnegative. Replacing $\mathscr{C}$ by $\mathscr{D}_0$, we obtain the following *doubly nonnegative relaxation* of (1):

$$\begin{aligned} \bar{\Phi}(\mathscr{F}) := \max \quad & \langle H, X\rangle + 2p^\top x \\ \text{s.t.} \quad & Ax = b \\ & \operatorname{diag}(AXA^\top) = b \circ b \\ & \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \geq 0, \ \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \succeq 0. \end{aligned} \tag{5}$$

It is easy to see that:

$$\bar{\Phi}(\mathscr{F}) \geq \Phi^*(\mathscr{F}). \tag{6}$$

In the following we call (5) the DNN relaxation of (1) and call $\bar{\Phi}(\mathscr{F})$ the DNN bound of $\Phi^*(\mathscr{F})$. Note that (5) is a semidefinite program (SDP) and can be solved numerically by many existing SDP solvers, e.g. [28,30,33]. There are also efficient solvers specially dedicated to approximate the DNN bound $\bar{\Phi}(\mathscr{F})$, e.g. [15].

1.4 Approach and contribution

We start by searching for a Karush–Kuhn–Tucker (KKT) point $\bar{x}$ of (1). If $\Phi(\bar{x}) \geq v_R$, we conclude that $\Phi^*(\mathscr{F}) \geq v_R$ and the reference value problem is solved. If $\Phi(\bar{x}) < v_R$, we solve the SDP problem (5) to get the DNN bound $\bar{\Phi}(\mathscr{F})$. If

$$\bar{\Phi}(\mathscr{F}) < v_R, \tag{7}$$

we conclude that $\Phi^*(\mathscr{F}) < v_R$ and the reference value problem is solved. If instead $\bar{\Phi}(\mathscr{F}) \geq v_R$, we propose to rely on the classical cutting plane method to proceed.

We will add valid cut to the original problem (1) so that the feasible region is reduced successively until we find an answer to (3). The major novelty of our approach

is the computation of valid cuts from DNN relaxation. Let $\bar{x}$ be a KKT point of the QP problem (1) such that $\Phi(\bar{x}) < v_R$. If there is a simplex $\Delta \ni \bar{x}$ such that

$$v_R > \Phi^*(\mathscr{F} \cap \Delta) := \max\{\Phi(x) : x \in \mathscr{F} \cap \Delta\}, \tag{8}$$

then we only need to consider the restricted region $\mathscr{F}\backslash\Delta$ instead of $\mathscr{F}$ to find an answer to (3). Valid cut can be added to describe the restricted region $\mathscr{F}\backslash\Delta$ if $\bar{x}$ is a vertex of Δ. This cut will exclude the region $\mathscr{F} \cap \Delta$ which contains the KKT point $\bar{x}$. We then deal with a concave QP problem with a strictly smaller feasible region $\mathscr{F}\backslash\Delta$ and we can repeat the same process until an answer to (3) is obtained.

It is interesting to note that (8) is in fact a reference value problem. Directly verifying (8) being impossible, it is common to replace $\Phi^*(\mathscr{F} \cap \Delta)$ in (8) by a computable upper bound of it. In [31], Tuy proposed to generate valid cut using the following computable upper bound of $\Phi^*(\mathscr{F} \cap \Delta)$:

$$\max\{\Phi(x) : x \in \Delta\}.$$

In [17], Konno proposed another computable upper bound of $\Phi^*(\mathscr{F} \cap \Delta)$:

$$\bar{\Phi}^K(\mathscr{F} \cap \Delta) := \max\{\Psi(x,\tilde{x}) : x \in \Delta, \tilde{x} \in \mathscr{F}\}, \tag{9}$$

where $\Psi(x,\tilde{x}) := x^\top H\tilde{x} + p^\top x + p^\top \tilde{x}$. In this paper, we propose to employ the DNN bound and validate (8) if the following condition holds:

$$\bar{\Phi}(\mathscr{F} \cap \Delta) < v_R. \tag{10}$$

We show in Theorem 2 that

$$\bar{\Phi}(\mathscr{F} \cap \Delta) \le \bar{\Phi}^K(\mathscr{F} \cap \Delta). \tag{11}$$

Therefore, the cut generated based on (10) is always deeper than Konno's cut.

The computation of DNN bound, which is needed in (7) and (10), clearly plays the decisive role in the overall performance of the procedure described above. With the aim of developing efficient and robust concave QP solver in the large-scale setting, we also tackled the following two problems:

1. It is known that the SDP problem in the form of (5) does not have an interior point. However, the theoretical convergence of the existing popular SDP solvers (e.g. [28,30,33]) is established by assuming Slater's condition, i.e., the existence of an interior point.
2. Any numerical solver using finite floating-point arithmetic is incapable of returning the exact value of $\bar{\Phi}(\mathscr{F})$. Let ν be the value returned by a numerical SDP solver (e.g. `Mosek` and `SDPNAL+`) for solving (5). Then ν is only an approximation (whatever the precision is) of $\bar{\Phi}(\mathscr{F})$. In particular, $\nu < v_R$ does not directly imply (7). Similar problem occurs for computing $\bar{\Phi}(\mathscr{F} \cap \Delta)$ and verifying (10).

To fix the first issue, we prove in Lemma 1 that the DNN relaxation (5) has an equivalent SDP formulation (19), which corresponds to Shor's relaxation applied to an equivalent quadratically constrained quadratic program (QCQP) of (1). We then show in Proposition 1 that under some mild assumptions, Slater's condition holds

for (19) so that all the above mentioned SDP solvers are guaranteed to converge when applied to solve (19). For the second issue, we establish an explicit formula in Proposition 2 for computing a valid upper bound of $\Phi^*(\mathscr{F})$ from any inexact primal dual solution of the SDP problem (19). This result is crucial to allow the use of arbitrary SDP solver for the computation of the DNN bound, including in particular those solvers with medium accuracy specially designed for large-scale SDP.

We demonstrate the effectiveness of our approach through extensive computational experiments on both real and synthetic instances. We compare the performance of our algorithm with two of the most powerful commercial solvers `CPLEX` and `Gurobi` and an academic open-source software `quadprogIP` [34]. Our algorithm successfully solved the earlier mentioned 317,584 instances of dimension 100 within **3 days** using 32 processors in parallel. For the three instances of dimension 841, our algorithm is able to solve them in **a few minutes** on a standard laptop. On a variety of randomly generated instances of dimension 100 to 500, our algorithm also exhibits superior performance with remarkable differences.

The paper is organized as follows. In Section 2 we study properties of the DNN relaxation (5), show its connection with Shor's relaxation, and discuss the computation of valid bound from inexact SDP solutions. In Section 3 we review the cutting plane method for the reference value problem, propose new cuts based on DNN relaxation, and show its connection with Konno's cut. In Section 4 we describe in detail our algorithm. In Section 5 we report numerical results. In Section 6 we conclude. For clarity of presentation, some details are moved to the Appendix.

1.5 Notations

Throughout the paper we let $[n] := \{1,\dots,n\}$. We use $\mathbf{0}$ to denote a zero vector or matrix of appropriate dimension, $\mathbf{1}$ to denote an all-one vector, and $\mathbf{I}$ to denote the identity matrix of appropriate dimension. For any vector $x \in \mathbb{R}^n$ and matrix $X \in \mathbb{R}^{m\times n}$, we use $x_i \in \mathbb{R}$ to denote the ith entry of x, $X_{i,j} \in \mathbb{R}$ to denote the (i,j)th entry of X and $X_i \in \mathbb{R}^m$ to denote the ith column vector of X. We adopt a MATLAB-like notation to represent the submatrix of $X \in \mathbb{R}^{m\times n}$, for example $X_{1:k,n}$ is the submatrix of X formed by the elements at the intersection of the first k rows and the nth column of X.

For a matrix X, $X \geq 0$ means that X is nonnegative, i.e. all the elements in X are nonnegative, and $X \succeq 0$ means that X is a positive semidefinite matrix. We denote by $\mathscr{S}^n$ the set of n-by-n symmetric matrices and by $\mathscr{S}^n_+$ the cone of positive semidefinite matrices. For two matrices $X, Y \in \mathscr{S}^n$, $X \succeq Y$ means that $X - Y \in \mathscr{S}^n_+$, and $X \geq Y$ means that $X - Y \in \mathbb{R}^{n\times n}_+$. We use $\mathrm{diag}(h)$ to represent the diagonal matrix with diagonal vector being equal to h. We consider the Frobenius inner product $\langle\cdot,\cdot\rangle$ in the space of matrices, i.e., $\langle A, B\rangle := \mathrm{tr}(A^\top B)$.

2 Doubly Nonnegative Relaxation

The DNN relaxation (5) provides an upper bound for the optimal value of (1). However, it is known from [2, Prop. 8.3] that the feasible region of the SDP problem (5)

has no interior. The lack of interior point can be a serious defect from both theoretical and computational aspects. In particular, strong duality may not hold for problems with no interior feasible point and convergence of existing SDP solvers is not guaranteed without the strong duality property, see [27] for more discussion.

To overcome this issue, we propose to consider the standard Shor relaxation of the equivalent QCQP problem (see (19)). We prove the strong duality for the SDP problem obtained from Shor's relaxation and its equivalence with the SDP problem (5) obtained from the DNN relaxation. We also establish formula for computing a valid upper bound of $\Phi^*(\mathscr{F})$ from an approximate solution returned by any arbitrary SDP solver, which is crucial to make the algorithm adapted to a wide range of inexact SDP solvers.

2.1 Reduction at vertex

To start with, we consider an equivalent form of (1) by a change of coordinates. Let $\bar{x}$ be a vertex of $\mathscr{F}$. Let $\{i_1,\ldots,i_m\}$ be the indices of basic variables of $\bar{x}$ and $\{j_1,\ldots,j_{n-m}\}$ be the indices of nonbasic variables. Note that the choice of basic variables is not unique if $\bar{x}$ is a degenerate vertex. Let $B := \begin{pmatrix} A_{i_1} & \cdots & A_{i_m} \end{pmatrix}$ be the basis matrix and $N := \begin{pmatrix} A_{j_1} & \cdots & A_{j_{n-m}} \end{pmatrix}$ be the nonbasis matrix. Following the standard notations in linear programming, for any $x \in \mathbb{R}^n$ we denote by x_B the subvector $(x_{i_1},\ldots,x_{i_m}) \in \mathbb{R}^m$ and by x_N the subvector $(x_{j_1},\ldots,x_{j_{n-m}}) \in \mathbb{R}^{n-m}$. Further, we partition accordingly the matrix H and vector p into blocks:

$$\begin{pmatrix} H_{BB} & H_{BN} \\ H_{BN}^\top & H_{NN} \end{pmatrix}, \quad \begin{pmatrix} p_B \\ p_N \end{pmatrix}. \tag{12}$$

Hereinafter, with a given vertex $\bar{x}$ and a choice of basis matrix B and nonbasis matrix N, we denote:

$$\begin{aligned} &F = \left(B^{-1}N\right)^\top, \;\; w = B^{-1}b, \;\; v = w^\top H_{BB} w + 2p_B^\top w, \\ &Q = H_{NN} + F H_{BB} F^\top - F H_{BN} - H_{BN}^\top F^\top, \\ &d = p_N + H_{BN}^\top w - F p_B - F^\top H_{BB} w. \end{aligned} \tag{13}$$

Then we have

$$\mathscr{F} \simeq \left\{ \begin{pmatrix} y \\ s \end{pmatrix} \in \mathbb{R}^n : F^\top y + s = w, y \geq 0, s \geq 0 \right\}. \tag{14}$$

Here the vector $y \in \mathbb{R}^{n-m}$ corresponds to the nonbasis vector x_N and the vector $s \in \mathbb{R}^m$ corresponds to the basis vector x_B. By replacing the basic variables in the objective function of (1) using the expression $x_B = B^{-1}b - B^{-1}Nx_N$, we arrive at the following concave QP problem:

$$\begin{aligned} \max \quad & y^T Q y + 2d^\top y + v \\ \text{s.t.} \quad & F^\top y + s = w \\ & y \geq 0, s \geq 0. \end{aligned} \tag{15}$$

The DNN relaxation of (15) yields the following SDP:

$$\begin{aligned}
\max_{X\in\mathscr{S}^{n+1}} \quad & \langle \hat{H}, X\rangle \\
\text{s.t.} \quad & \begin{pmatrix} F^\top & \mathbf{I} & 0\end{pmatrix} X_{n+1} = w, \\
& \operatorname{diag}\left(\begin{pmatrix} F^\top & \mathbf{I} & 0\end{pmatrix} X \begin{pmatrix} F \\ \mathbf{I} \\ 0\end{pmatrix}\right) = w\circ w \\
& \langle C, X\rangle = 1 \\
& X \geq 0,\ X \succeq 0
\end{aligned} \tag{16}$$

where

$$\hat{H} := \begin{pmatrix} Q & \mathbf{0} & d \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ d^\top & \mathbf{0}^\top & v \end{pmatrix},\ C := \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^\top & \mathbf{0}^\top & 1 \end{pmatrix}.$$

It can be verified that (15) is equivalent to (1) and (16) is equivalent to (5). In particular, the optimal value of (15) is equal to $\Phi^*(\mathscr{F})$ and the optimal value of (16) is equal to $\bar{\Phi}(\mathscr{F})$.

2.2 Equivalence with the Shor relaxation

The program (15) can be further simplified to the following form:

$$\begin{aligned}
\max_{y\in\mathbb{R}^{n-m}} \quad & y^T Q y + 2d^\top y + v \\
\text{s.t.} \quad & F^\top y \leq w \\
& y \geq 0.
\end{aligned} \tag{17}$$

Adding redundant constraints to (17), we obtain the following QCQP problem:

$$\begin{aligned}
\max_{y\in\mathbb{R}^{n-m}} \quad & y^\top Q y + 2d^\top y + v \\
\text{s.t.} \quad & F^\top y \leq w, y \geq 0 \\
& (F^\top y - w)(F^\top y - w)^\top \geq 0 \\
& y y^\top \geq 0 \\
& (F^\top y - w) y^\top \leq 0.
\end{aligned} \tag{18}$$

It is easy to see that the optimal value of (18) is equal to $\Phi^*(\mathscr{F})$. The standard Shor relaxation of the QCQP problem (18) yields the following SDP problem:

$$\begin{array}{ll}\max\limits_{\substack{Y\in\mathscr{S}^{n-m}\\ y\in\mathbb{R}^{n-m}}} & \langle Q,Y\rangle+2d^\top y+v\\ \text{s.t.} & F^\top y\le w, y\ge 0\\ & F^\top YF-wy^\top F-F^\top yw^\top+ww^\top\ge 0\\ & Y\ge 0\\ & wy^\top-F^\top Y\ge 0\\ & \begin{pmatrix} Y & y\\ y^\top & 1\end{pmatrix}\succeq 0.\end{array}\tag{19}$$

Lemma 1 *The DNN relaxation* (16) *is equivalent to the Shor relaxation of the QCQP problem* (18)*, i.e., the optimal value of* (19) *is equal to* $\bar{\Phi}(\mathscr{F})$.

Proof Let (Y,y) be a feasible solution to (19). Let

$$X=\begin{pmatrix} Y & yw^\top-YF & y\\ wy^\top-F^\top Y & F^\top YF-F^\top yw^\top-wy^\top F+ww^\top & w-F^\top y\\ y^\top & w^\top-y^\top F & 1\end{pmatrix}.$$

Clearly we have $X\ge 0$. It can be checked easily that we have

$$\begin{pmatrix} F & \mathbf{I} & 0\end{pmatrix}X_{n+1}=w,$$

and

$$\begin{pmatrix} F^\top & \mathbf{I} & 0\end{pmatrix}X\begin{pmatrix} F\\ \mathbf{I}\\ 0\end{pmatrix}=ww^\top.$$

Besides,

$$X=\begin{pmatrix} y\\ w-F^\top y\\ 1\end{pmatrix}\begin{pmatrix} y\\ w-F^\top y\\ 1\end{pmatrix}^\top+\begin{pmatrix} \mathbf{I}\\ -F^\top\\ 0\end{pmatrix}\left(Y-yy^\top\right)\begin{pmatrix} \mathbf{I}\\ -F^\top\\ 0\end{pmatrix}^\top\succeq 0.$$

Therefore, X is a feasible solution to (16) and

$$\langle\hat{H},X\rangle=\langle Q,Y\rangle+2d^\top y+v.\tag{20}$$

Now let X be any feasible solution of (16) such that

$$X=\sum_{k=1}^{r}\alpha_k\begin{pmatrix}\xi^k\\ s^k\\ 1\end{pmatrix}\begin{pmatrix}\xi^k\\ s^k\\ 1\end{pmatrix}^\top+\sum_{k=r+1}^{t}\begin{pmatrix}\xi^k\\ s^k\\ 0\end{pmatrix}\begin{pmatrix}\xi^k\\ s^k\\ 0\end{pmatrix}^\top,$$

where $\alpha_1, \dots \alpha_r \geq 0$ and $\sum_{k=1}^r \alpha_k = 1$. By [2, Prop. 8.3], we know that

$$s^k = \begin{cases} w - F^\top \xi^k, & \forall k \in [r], \\ -F^\top \xi^k, & \forall k \in \{r+1, \dots, t\}. \end{cases}$$

Let

$$Y = \sum_{k=1}^r \alpha_k \xi^k (\xi^k)^\top + \sum_{k=r+1}^t \xi^k (\xi^k)^\top, \;\; y = \sum_{k=1}^r \alpha_k \xi^k.$$

Then

$$\begin{aligned} X &= \sum_{k=1}^r \alpha_k \begin{pmatrix} \xi^k \\ w - F^\top \xi^k \\ 1 \end{pmatrix} \begin{pmatrix} \xi^k \\ w - F^\top \xi^k \\ 1 \end{pmatrix}^\top + \sum_{k=r+1}^t \begin{pmatrix} \xi^k \\ -F^\top \xi^k \\ 0 \end{pmatrix} \begin{pmatrix} \xi^k \\ -F^\top \xi^k \\ 0 \end{pmatrix}^\top \\ &= \begin{pmatrix} Y & yw^\top - YF & y \\ wy^\top - F^\top Y & F^\top YF - F^\top yw^\top - wy^\top F + ww^\top & w - F^\top y \\ y^\top & w^\top - y^\top F & 1 \end{pmatrix}. \end{aligned}$$

And so (Y, y) is a feasible solution to (19) satisfying (20). □

Remark 1 The proof of Lemma 1 shows how to construct an optimal solution of (19) from an optimal solution of (16), and vice versa.

Remark 2 The size of the largest SDP matrix in (16) is $n+1$, while the size of the largest SDP matrix in (19) is $n-m+1$. This already suggests that (19) could be computationally more interesting than (16). Moreover, we will show in the next subsection that strong duality holds for (19).

2.3 Strong duality

We consider the SDP relaxation in the form of (19). Let us express (19) in the following abstract way:

$$\begin{aligned} \max_{\hat{Y} \in \mathcal{S}^{n-m+1}} \quad & \langle \hat{Q}, \hat{Y} \rangle \\ \text{s.t.} \quad & \langle W^{(i,j)}, \hat{Y} \rangle \leq 0, \;\; 0 \leq i < j \leq n \\ & \langle W^0, \hat{Y} \rangle = 1 \\ & \hat{Y} \succeq 0. \end{aligned} \tag{21}$$

Here,

$$\hat{Q} := \begin{pmatrix} Q & d \\ d^\top & v \end{pmatrix}, \quad W^0 := \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{pmatrix},$$

and $\{W^{(i,j)} : 0 \le i < j \le n\}$ are matrices for representing the polyhedral constraints in (19):

$$W^{(i,j)} := \begin{cases} \begin{pmatrix} \mathbf{0} & -e_j \\ -e_j^\top & 0 \end{pmatrix} & i = 0,\ \ 1 \le j \le n-m \\ \begin{pmatrix} \mathbf{0} & \tilde{F}_j \\ \tilde{F}_j^\top & -2\tilde{w}_j \end{pmatrix} & i = 0,\ \ n-m+1 \le j \le n \\ \begin{pmatrix} -e_i e_j^\top - e_j e_i^\top & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{pmatrix} & 1 \le i < j \le n-m \\ \begin{pmatrix} \tilde{F}_j e_i^\top + e_i \tilde{F}_j^\top & -\tilde{w}_j e_i \\ -\tilde{w}_j e_i^\top & 0 \end{pmatrix} & 1 \le i \le n-m < j \le n \\ \begin{pmatrix} -\tilde{F}_j \tilde{F}_i^\top - \tilde{F}_i \tilde{F}_j^\top & \tilde{w}_i \tilde{F}_j + \tilde{w}_j \tilde{F}_i \\ \tilde{w}_i \tilde{F}_j^\top + \tilde{w}_j \tilde{F}_i^\top & -2\tilde{w}_i \tilde{w}_j \end{pmatrix} & n-m+1 \le i < j \le n. \end{cases}$$

Here, $e_1, \ldots, e_{n-m}$ are the standard basis vectors of $\mathbb{R}^{n-m}$ and $\tilde{F}_j := F_{j-n+m}$, $\tilde{w}_j := w_{j-n+m}$ for each $n-m+1 \le j \le n$. The dual of (19) can thus be written as:

$$\begin{aligned} \min\ & \nu \\ \text{s.t.}\ & \hat{Q} + \sum_{i<j} \lambda_{i,j} W^{(i,j)} - \nu W^0 \preceq 0 \\ & \lambda_{i,j} \le 0,\ \ 0 \le i < j \le n. \end{aligned} \tag{22}$$

We shall need the following assumptions.

Assumption 1 *There is $t_* \ge 0$ such that*

$$\begin{aligned} t_* = \max_{y \in \mathbb{R}^{n-m}}\ & \mathbf{1}^\top y \\ \text{s.t.}\ & F^\top y \le w, y \ge 0. \end{aligned} \tag{23}$$

Assumption 2 *There is $0 < \rho_* \le 1$ such that*

$$\{y \in \mathbb{R}^{n-m} : F^\top y \le w - \rho_*, y \ge \rho_*\} \ne \emptyset.$$

Remark 3 Assumption 1 holds if $\mathscr{F}$ is bounded.

Remark 4 Assumption 2 is equivalent to requiring that the set $\{y \in \mathbb{R}^{n-m} : F^\top y \le w, y \ge 0\}$ has a nonempty interior. If this is not true, then there is some $i \in [n]$ such that $x_i = 0$ for any $x \in \mathscr{F}$ and so that the variable x_i can be replaced by 0. In other words, Assumption 2 can be always made true by removing superfluous variables from (1) if necessary.

Now we establish the strong duality between (21) and its dual (22). For this, we shall prove that (21) and (22) have a nonempty interior under the above two assumptions.

Lemma 2 *Under Assumption 2, there exists a solution (Y, y) strictly feasible to* (19).

Proof Let $y^0 \in \mathbb{R}^{n-m}$ such that $F^\top y^0 < w$ and $y^0 > 0$. Let $\varepsilon > 0$ and $y^i = y^0 + \varepsilon e_i$ for each $i \in [n-m]$. Choose $\varepsilon > 0$ sufficiently small so that $\{y^1, \ldots, y^{n-m}\}$ are all in the set $\{y \in \mathbb{R}^{n-m} : F^\top y \le w, y \ge 0\}$. Let

$$Y^i = y^i (y^i)^\top, \ \ \forall i \in \{0\} \cup [n-m].$$

Then for each i, (Y^i, y^i) is a feasible solution to (19). Let

$$Y = \frac{1}{n-m+1} \sum_{i=0}^{n-m} Y^i, \ \ y = \frac{1}{n-m+1} \sum_{i=0}^{n-m} y^i.$$

Clearly (Y, y) is feasible to (19). Since the vectors $\left\{ \begin{pmatrix} y^0 \\ 1 \end{pmatrix}, \ldots, \begin{pmatrix} y^{n-m} \\ 1 \end{pmatrix} \right\}$ are linearly independent, the rank of the matrix $\begin{pmatrix} Y & y \\ y^\top & 1 \end{pmatrix}$ is $n-m+1$. It follows that (Y, y) is strictly feasible to (19). □

Lemma 3 *Under Assumption 1, the dual SDP problem* (22) *is strictly feasible.*

Proof Under Assumption 1 and by strong duality, there is $a \in \mathbb{R}^m_+$ such that

$$\xi := Fa \ge \mathbf{1}, \ \ a^\top w = t_*.$$

We have

$$\begin{aligned}
&\sum_{i=1}^{n-m} \sum_{j=n-m+1}^{n} a_{j-n+m} W^{(i,j)} \\
&= \sum_{i=1}^{n-m} \sum_{j=1}^{m} a_j \begin{pmatrix} F_j e_i^\top + e_i F_j^\top & -w_j e_i \\ -w_j e_i^\top & 0 \end{pmatrix} \\
&= \sum_{i=1}^{n-m} \begin{pmatrix} \xi e_i^\top + e_i \xi^\top & -t_* e_i \\ -t_* e_i^\top & 0 \end{pmatrix} \\
&= \begin{pmatrix} \xi \mathbf{1}^\top + \mathbf{1} \xi^\top & -t_* \mathbf{1} \\ -t_* \mathbf{1}^\top & 0 \end{pmatrix}.
\end{aligned}$$

Thus

$$\begin{aligned}
&\sum_{i=1}^{n-m} \sum_{j=n-m+1}^{n} a_{j-n+m} W^{(i,j)} + \sum_{1 \le i < j \le n-m} (\xi_i + \xi_j) W^{(i,j)} \\
&= \begin{pmatrix} \xi \mathbf{1}^\top + \mathbf{1} \xi^\top & -t_* \mathbf{1} \\ -t_* \mathbf{1}^\top & 0 \end{pmatrix} + \sum_{1 \le i < j \le n-m} (\xi_i + \xi_j) \begin{pmatrix} -e_i e_j^\top - e_j e_i^\top & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{pmatrix} \\
&= \begin{pmatrix} 2\,\mathrm{diag}(\xi) & -t_* \mathbf{1} \\ -t_* \mathbf{1}^\top & 0 \end{pmatrix}.
\end{aligned}$$

It follows that

$$\begin{aligned}
&\sum_{i=1}^{n-m} \sum_{j=n-m+1}^{n} a_{j-n+m} W^{(i,j)} + \sum_{1 \le i < j \le n-m} (\xi_i + \xi_j) W^{(i,j)} + 2n t_*^2 W^0 \\
&= \begin{pmatrix} 2\,\mathrm{diag}(\xi) & -t_* \mathbf{1} \\ -t_* \mathbf{1}^\top & 2n t_*^2 \end{pmatrix}.
\end{aligned}$$

Since

$$(-t_*\mathbf{1})^\top(2\,\mathrm{diag}(\xi))^{-1}(-t_*\mathbf{1}) = \frac{1}{2}t_*^2\left(\sum_{i=1}^{n-m}\xi_i^{-1}\right) \le \frac{1}{2}(n-m)t_*^2 < nt_*^2,$$

we know that

$$\sum_{i=1}^{n-m}\sum_{j=n-m+1}^{n} a_{j-n+m}W^{(i,j)} + \sum_{1\le i<j\le n-m}(\xi_i+\xi_j)W^{(i,j)} + nt_*^2W^0 \succ 0.$$

Perturb the left hand side with sufficiently small $\varepsilon > 0$, we still have

$$\sum_{i=1}^{n-m}\sum_{j=n-m+1}^{n} (a_{j-n+m}+\varepsilon)W^{(i,j)} + \sum_{1\le i<j\le n-m}(\xi_i+\xi_j)W^{(i,j)} + nt_*^2W^0 \succ 0.$$

Note that the coefficients above can be scaled by arbitrary positive constant without changing the sign of semidefinite inequality. Therefore, if we take

$$\begin{aligned}
&\lambda_{i,j} = -\alpha(a_{j-n+m}+\varepsilon), \quad 1 \le i \le n-m < j \le n,\\
&\lambda_{i,j} = -\alpha(\xi_i+\xi_j), \quad 1 \le i < j \le n-m,\\
&\nu = nt_*^2,
\end{aligned}$$

for sufficiently large α, and take other $\lambda_{i,j}$ (where $i = 0$ or $i > n-m$) to be negative and sufficiently close to 0, we see that there exist $\{\lambda_{i,j} : 0 \le i < j \le n\} \subset \mathbb{R}_-$ and $\nu \in \mathbb{R}$ such that

$$\hat{Q} + \sum_{i<j}\lambda_{i,j}W^{(i,j)} - \nu W^0 \prec 0.$$

□

We then deduce the strong duality result for the SDP relaxation (19).

Proposition 1 *Under Assumption 1 and 2, both the SDP problem* (19) *and its dual SDP problem* (22) *have an optimal solution and the strong duality holds.*

2.4 Valid DNN bound from inexact SDP solution

Hereinafter, we will assume both Assumption 1 and 2. By Lemma 1, in order to compute the DNN bound $\bar{\Phi}(\mathscr{F})$, we can solve the SDP problem (19).

With the strong duality property of (19), we rest assured for the convergence of a wide range of SDP solvers. We mention in particular the interior point type methods and the augmented Lagrangian type methods. There are many powerful solvers that implement these methods: `Mosek`, `Gurobi`, `SDPT3`, `SDPNAL+`, etc.

However, it is important to note that any numerical solver using finite floating-point arithmetic is incapable of returning the exact value of $\bar{\Phi}(\mathscr{F})$. Indeed, such numerical solvers terminate when a solution with sufficiently small infeasibility gap and primal dual optimality gap is found. The value returned by the numerical SDP

solver is only an approximation of the true value $\bar{\Phi}(\mathscr{F})$. Additional care needs to be taken in order to get a valid upper bound of $\Phi^*(\mathscr{F})$.

To be more precise, consider applying any suitable SDP numerical solver to the primal SDP problem (21) and its dual SDP problem (22). Let $\varepsilon > 0$ and assume that the pair of primal dual solution $(\hat{Y}; \lambda, \nu)$ returned by the SDP solver satisfies the following infeasibility and duality gap condition

$$\begin{cases} \langle W^{(i,j)}, \hat{Y} \rangle \le \varepsilon, \ \ 0 \le i < j \le n \\ \langle W^0, \hat{Y} \rangle = 1 \\ \hat{Y} \succeq 0 \\ \hat{Q} + \sum_{i<j} \lambda_{i,j} W^{(i,j)} - \nu W^0 \preceq \varepsilon \mathbf{I} \\ \lambda \le 0 \\ |\langle \hat{Q}, \hat{Y} \rangle - \nu| \le \varepsilon. \end{cases} \tag{24}$$

As long as $\varepsilon > 0$, there is no guarantee that the approximate primal optimal value $\langle \hat{Q}, \hat{Y} \rangle$ or the approximate dual optimal value ν is larger than $\Phi^*(\mathscr{F})$. We next show how to obtain a valid upper bound of $\Phi^*(\mathscr{F})$ from $(\hat{Y}; \lambda, \nu)$ satisfying (24).

Lemma 4 *Let (λ^*, ν^*) be an optimal solution of the dual problem* (22)*, then*

$$\sum_{i<j} \lambda^*_{i,j} \ge \frac{\Phi_*(\mathscr{F}) - \bar{\Phi}(\mathscr{F})}{2\rho_*^2}, \tag{25}$$

where

$$\begin{aligned} \Phi_*(\mathscr{F}) &:= \min\{\Phi(x) : x \in \mathscr{F}\} \\ &= \min\{y^\top Q y + 2d^\top y + v : F^\top y \le w, y \ge 0\}, \end{aligned}$$

and $0 < \rho_* < 1$ is the constant in Assumption 2.

Proof Let $y \in \mathbb{R}^{n-m}$ such that $F^\top y - w \le -\rho_*$ and $y \ge \rho_*$. Let

$$\hat{Y} = \begin{pmatrix} yy^\top & y \\ y^\top & 1 \end{pmatrix}.$$

Then

$$\langle W^{(i,j)}, \hat{Y} \rangle \le \max(-2\rho_*, -2\rho_*^2) = -2\rho_*^2, \ \ 0 \le i < j \le n.$$

In view of Proposition 1, we have $\nu^* = \bar{\Phi}(\mathscr{F})$. Hence,

$$\begin{aligned} &\Phi_*(\mathscr{F}) - 2\sum_{i<j} \lambda^*_{i,j} \rho_*^2 - \bar{\Phi}(\mathscr{F}) \\ &= \Phi_*(\mathscr{F}) - 2\sum_{i<j} \lambda^*_{i,j} \rho_*^2 - \nu^* \\ &\le y^\top Q y + 2d^\top y + v - 2\sum_{i<j} \lambda^*_{i,j} \rho_*^2 - \nu^* \\ &\le \langle \hat{Q}, \hat{Y} \rangle + \sum_{i<j} \lambda^*_{i,j} \langle W^{(i,j)}, \hat{Y} \rangle - \nu^* \langle W^0, \hat{Y} \rangle \\ &= \langle \hat{Q} + \sum_{i<j} \lambda^*_{i,j} W^{(i,j)} - \nu^* W^0, \hat{Y} \rangle \le 0. \end{aligned}$$

□

Lemma 5 *For any feasible solution $\hat{Y}$ of* (21)*, we have*

$$\operatorname{trace}(\hat{Y}) \leq 1 + t_*^2, \tag{26}$$

where $t_* \geq 0$ is the constant in Assumption 1.

Proof Let

$$\hat{Y} = \begin{pmatrix} Y & y \\ y^\top & 1 \end{pmatrix}$$

be a feasible solution of (21). In the proof of Lemma 3, we have shown that

$$\sum_{i=1}^{n-m} \sum_{j=n-m+1}^{n} a_{j-n+m} W^{(i,j)} = \begin{pmatrix} \xi \mathbf{1}^\top + \mathbf{1}\xi^\top & -t_* \mathbf{1} \\ -t_* \mathbf{1}^\top & 0 \end{pmatrix},$$

where $a_{j-n+m} \geq 0$ for all $j \in \{n-m+1, \ldots, n\}$ and $\xi \geq \mathbf{1}$. Therefore, since $\langle W^{(i,j)}, \hat{Y} \rangle \leq 0$, we have

$$\xi^\top Y \mathbf{1} \leq t_* \mathbf{1}^\top y \leq t_*^2.$$

Since $Y \geq 0$ and $\xi \geq \mathbf{1}$, it follows that $\operatorname{trace}(Y) \leq t_*^2$. □

Proposition 2 *Let $\varepsilon > 0$ and let $(\hat{Y}; \lambda, \nu)$ be a pair of approximate primal and dual solutions satisfying the gap conditions in* (24)*. Then,*

$$\langle \hat{Q}, \hat{Y} \rangle + \frac{\left(\Phi_*(\mathscr{F}) - \bar{\Phi}(\mathscr{F})\right)\varepsilon}{2\rho_*^2} \leq \bar{\Phi}(\mathscr{F}) \leq \varepsilon\left(t_*^2 + 1\right) + \nu.$$

Proof Let $(\hat{Y}^*, \lambda^*, \nu^*)$ be an optimal primal dual solution pair. Then,

$$\begin{aligned}
\bar{\Phi}(\mathscr{F}) = \nu^* &= \nu^* \langle W^0, \hat{Y} \rangle \\
&= \langle \nu^* W^0 - \hat{Q} - \sum_{i<j} \lambda_{i,j}^* W^{(i,j)}, \hat{Y} \rangle + \langle \hat{Q}, \hat{Y} \rangle + \sum_{i<j} \lambda_{i,j}^* \langle W^{(i,j)}, \hat{Y} \rangle \\
&\overset{(24)}{\geq} \langle \hat{Q}, \hat{Y} \rangle + \varepsilon \sum_{i<j} \lambda_{i,j}^* \\
&\overset{(25)}{\geq} \langle \hat{Q}, \hat{Y} \rangle + \frac{\left(\Phi_*(\mathscr{F}) - \bar{\Phi}(\mathscr{F})\right)\varepsilon}{2\rho_*^2},
\end{aligned}$$

and

$$\begin{aligned}
\bar{\Phi}(\mathscr{F}) &= \langle \hat{Q}, \hat{Y}^* \rangle \\
&= \langle \hat{Q} + \sum_{i<j} \lambda_{i,j} W^{(i,j)} - \nu W^0, \hat{Y}^* \rangle - \sum_{i<j} \lambda_{i,j} \langle W^{(i,j)}, \hat{Y}^* \rangle + \nu \langle W^0, \hat{Y}^* \rangle \\
&\overset{(24)}{\leq} \varepsilon \operatorname{trace}(\hat{Y}^*) + \nu \\
&\overset{(26)}{\leq} \varepsilon\left(1 + t_*^2\right) + \nu.
\end{aligned}$$

□

Proposition 2 enables to obtain a valid upper bound of $\Phi^*(\mathscr{F})$ from any approximate primal dual solution $(\hat{Y}; \lambda, \nu)$ returned by any SDP solver. The correction term to be added is $\varepsilon(t_*^2 + 1)$, where ε depends on the feasibility gap and the duality gap and t_* is defined in (23).

2.5 Upper bound of the DNN bound

In this subsection, we establish an upper bound for the DNN bound $\bar{\Phi}(\mathscr{F})$. This result will be useful in the later section when we discuss the generation of cuts.

Proposition 3 *Let $\Lambda \in \mathbb{R}_+^{(n-m)\times m}$ and $P \in \mathbb{R}_+^{(n-m)\times(n-m)} \cap \mathscr{S}^{n-m}$ be nonnegative matrices such that*

$$-Q - P + \Lambda F^\top + F\Lambda^\top \succeq 0. \tag{27}$$

Then,

$$\bar{\Phi}(\mathscr{F}) \le v + \max\left\{2(d+\Lambda w)^\top y : F^\top y \le w, y \ge 0\right\}.$$

Proof Let $e_1, \ldots, e_m$ be the standard basis vectors of $\mathbb{R}^m$. Let $X \in \mathscr{S}^{n+1}$ be a feasible solution of (16). Then X satisfies

$$\begin{cases} \langle A^i, X\rangle = w_i^2, \ \ \forall i \in [m] \\ \langle B^i, X\rangle = w_i, \ \ \forall i \in [m] \\ \langle C, X\rangle = 1 \ \ , \end{cases} \tag{28}$$

where

$$A^i := \begin{pmatrix} F_iF_i^\top & F_ie_i^\top & \mathbf{0} \\ e_iF_i^\top & e_ie_i^\top & \mathbf{0} \\ \mathbf{0}^\top & \mathbf{0}^\top & 0 \end{pmatrix}, \ \ B^i := \begin{pmatrix} \mathbf{0} & \mathbf{0} & \frac{1}{2}F_i \\ \mathbf{0} & \mathbf{0} & \frac{1}{2}e_i \\ \frac{1}{2}F_i^\top & \frac{1}{2}e_i^\top & 0 \end{pmatrix}, \ \ \forall i \in [m]. \tag{29}$$

Let any $\lambda_1, \ldots, \lambda_m \in \mathbb{R}$. We have

$$\begin{aligned} \langle \hat{H}, X\rangle &= \langle \hat{H}, X\rangle + \sum_{i=1}^m \lambda_i \left(w_i^2 - \langle A^i, X\rangle\right) \\ &\quad - \sum_{i=1}^m 2\lambda_i w_i \left(w_i - \langle B^i, X\rangle\right) + \sum_{i=1}^m \lambda_i w_i^2 \left(1 - \langle C, X\rangle\right) \\ &= \langle \hat{H} - \sum_{i=1}^m \lambda_i A^i + 2\sum_{i=1}^m \lambda_i w_i B^i - \sum_{i=1}^m \lambda_i w_i^2 C,\ X\rangle, \end{aligned} \tag{30}$$

and

$$\begin{aligned} &\hat{H} - \sum_{i=1}^m \lambda_i A^i + 2\sum_{i=1}^m \lambda_i w_i B^i - \sum_{i=1}^m \lambda_i w_i^2 C \\ &= \begin{pmatrix} Q - \sum_{i=1}^m \lambda_i F_i F_i^\top & -\lambda_1 F_1 & \cdots & -\lambda_m F_m & d + \sum_{i=1}^m \lambda_i w_i F_i \\ -\lambda_1 F_1^\top & -\lambda_1 & \cdots & 0 & \lambda_1 w_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\lambda_m F_m^\top & 0 & \cdots & -\lambda_m & \lambda_m w_m \\ d^\top + \sum_{i=1}^m \lambda_i w_i F_i^\top & \lambda_1 w_1 & \cdots & \lambda_m w_m & v - \sum_{i=1}^m \lambda_i w_i^2 \end{pmatrix}. \end{aligned}$$

Let any $\varepsilon > 0$. Denote $\tilde{Q} = Q - \varepsilon \mathbf{I} + P$ and

$$S = \begin{pmatrix} -\tilde{Q} + \sum_{i=1}^{m} \lambda_i F_i F_i^\top & \lambda_1 F_1 - \Lambda_1 & \cdots & \lambda_m F_m - \Lambda_m & -\sum_{i=1}^{m} \lambda_i w_i F_i + \Lambda w \\ \lambda_1 F_1^\top - \Lambda_1^\top & \lambda_1 & \cdots & 0 & -\lambda_1 w_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \lambda_m F_m^\top - \Lambda_m^\top & 0 & \cdots & \lambda_m & -\lambda_m w_m \\ \sum_{i=1}^{m} \lambda_i w_i F_i^\top + w^\top \Lambda^\top & -\lambda_1 w_1 & \cdots & -\lambda_m w_m & \sum_{i=1}^{m} \lambda_i w_i^2 \end{pmatrix}. \quad (31)$$

Further, let

$$T = \begin{pmatrix} P & \Lambda_1 & \cdots & \Lambda_m & \mathbf{0} \\ \Lambda_1^\top & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Lambda_m^\top & 0 & \cdots & 0 & 0 \\ \mathbf{0}^\top & 0 & \cdots & 0 & 0 \end{pmatrix}, \quad \Delta_0 = \begin{pmatrix} d + \Lambda w \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Then

$$\begin{aligned} &\hat{H} - \sum_{i=1}^{m} \lambda_i A^i + 2 \sum_{i=1}^{m} \lambda_i w_i B^i - \sum_{i=1}^{m} \lambda_i w_i^2 C \\ &= -S - T + \begin{pmatrix} \mathbf{0} & \Delta_0 \\ \Delta_0^\top & 0 \end{pmatrix} + \begin{pmatrix} \varepsilon \mathbf{I} & \mathbf{0} \\ \mathbf{0}^\top & v \end{pmatrix}. \end{aligned} \quad (32)$$

Plugging (32) into (30) we obtain

$$\langle \hat{H}, X \rangle = \left\langle -S - T + \begin{pmatrix} \mathbf{0} & \Delta_0 \\ \Delta_0^\top & 0 \end{pmatrix} + \begin{pmatrix} \varepsilon \mathbf{I} & \mathbf{0} \\ \mathbf{0}^\top & v \end{pmatrix}, X \right\rangle. \quad (33)$$

Since $\begin{pmatrix} F^\top & \mathbf{I} & 0 \end{pmatrix} X_{n+1} = w$ and $X_{n+1} \geq 0$, we have

$$\begin{aligned} \left\langle \begin{pmatrix} \mathbf{0} & \Delta_0 \\ \Delta_0^\top & 0 \end{pmatrix}, X \right\rangle &= 2 (d + \Lambda w)^\top X_{1:n-m,n+1} \\ &\leq \max \left\{ 2 (d + \Lambda w)^\top y : F^\top y \leq w, y \geq 0 \right\}. \end{aligned}$$

Since $X \geq 0$ and $T \geq 0$, we have $\langle -T, X \rangle \leq 0$. By (27), there exists $\lambda(\varepsilon) \in \mathbb{R}^m_{++}$ such that

$$-\tilde{Q} + \Lambda F^\top + F \Lambda^\top \succeq \Lambda \operatorname{diag}(\lambda^{-1}(\varepsilon)) \Lambda^\top. \quad (34)$$

In view of Lemma 6 (which will be formally stated later), if $\lambda = \lambda(\varepsilon)$, then $S \succeq 0$ and

$$\langle \hat{H}, X \rangle \leq \left\langle \begin{pmatrix} \varepsilon \mathbf{I} & \mathbf{0} \\ \mathbf{0}^\top & v \end{pmatrix}, X \right\rangle + \max \left\{ 2 (d + \Lambda w)^\top y : F^\top y \leq w, y \geq 0 \right\}.$$

Since the later condition holds for any $\varepsilon > 0$, we deduce that

$$\langle \hat{H}, X \rangle \leq v + \max \left\{ 2 (d + \Lambda w)^\top y : F^\top y \leq w, y \geq 0 \right\}.$$

□

Now we verify the auxiliary result used in the proof of Proposition 3.

Lemma 6 *The matrix S defined in* (31) *is positive semidefinite if* $\lambda_1, \ldots, \lambda_m$ *are all positive and*

$$-\tilde{Q} + \sum_{i=1}^{m} \left(\Lambda_i F_i^\top + F_i \Lambda_i^\top \right) \succeq \sum_{i=1}^{m} \lambda_i^{-1} \Lambda_i \Lambda_i^\top. \tag{35}$$

Proof Let any $y \in \mathbb{R}^{n-m}$, $z \in \mathbb{R}^m$ and $\rho \in \mathbb{R}$. Then we have

$$\begin{aligned}
&\begin{pmatrix} y^\top & z^\top & \rho \end{pmatrix} S \begin{pmatrix} y \\ z \\ \rho \end{pmatrix} \\
&= y^\top \left(-\tilde{Q} + \sum_{i=1}^{m} \lambda_i F_i F_i^\top \right) y + 2 \sum_{i=1}^{m} z_i \left(\lambda_i F_i^\top y - \Lambda_i^\top y \right) \\
&\quad + 2 y^\top \left(\sum_{i=1}^{m} w_i \Lambda_i - \sum_{i=1}^{m} w_i \lambda_i F_i \right) \rho + \sum_{i=1}^{m} \lambda_i (\rho w_i - z_i)^2 \\
&= y^\top \left(-\hat{Q} + \sum_{i=1}^{m} \lambda_i F_i F_i^\top \right) y + 2 y^\top \left(\sum_{i=1}^{m} w_i \Lambda_i - \sum_{i=1}^{m} w_i \lambda_i F_i \right) \rho \\
&\qquad + \sum_{i=1}^{m} \lambda_i \left[z_i^2 + 2 z_i \left(F_i^\top y - \lambda_i^{-1} \Lambda_i^\top y - \rho w_i \right) \right] + \sum_{i=1}^{m} \lambda_i w_i^2 \rho^2 \\
&\geq y^\top \left(-\hat{Q} + \sum_{i=1}^{m} \lambda_i F_i F_i^\top \right) y + 2 y^\top \left(\sum_{i=1}^{m} w_i \Lambda_i - \sum_{i=1}^{m} w_i \lambda_i F_i \right) \rho \\
&\qquad - \sum_{i=1}^{m} \lambda_i \left(F_i^\top y - \lambda_i^{-1} \Lambda_i^\top y - \rho w_i \right)^2 + \sum_{i=1}^{m} \lambda_i w_i^2 \rho^2 \\
&= y^\top \left(-\hat{Q} + \sum_{i=1}^{m} \lambda_i F_i F_i^\top \right) y - \sum_{i=1}^{m} \lambda_i \left(F_i^\top y - \lambda_i^{-1} \Lambda_i^\top y \right)^2 \\
&= y^\top \left(-\tilde{Q} + \sum_{i=1}^{m} \left(F_i \Lambda_i^\top + \Lambda_i F_i^\top \right) \right) y - y^\top \left(\sum_{i=1}^{m} \lambda_i^{-1} \Lambda_i \Lambda_i^\top \right) y \\
&\geq 0.
\end{aligned}$$

□

3 Generation of Valid Cuts

In this section we review the idea of cutting plane algorithms and present classical as well as novel methods for the generation of valid cuts at any given KKT point. Recall that we have, at our disposal, a reference value v_R and the task is to determine whether

$$v_R > \Phi^*(\mathscr{F}) \quad \text{or} \quad \Phi^*(\mathscr{F}) \geq v_R \ ? \tag{36}$$

3.1 Cutting plane method

Let $\bar{x}$ be a vertex of $\mathscr{F}$ which is a KKT point of (1). If $\Phi(\bar{x}) \geq v_R$, then we have $\Phi^*(\mathscr{F}) \geq v_R$ and the question (36) is solved. So let us assume that $\Phi(\bar{x}) < v_R$.

Let (B,N) be a pair of basis and nonbasis matrices associated with $\bar{x}$. Compute (Q,d,F,w,v) as in (13). Note that

$$v = \Phi(\bar{x}) < v_R.$$

Rewrite the original problem (1) into its reduced form (17). Since $\bar{x}$ is a KKT point, we can assume that the basis matrix B is such that $d \leq 0$ (see Appendix B for the selection of such B).

Let

$$\mathscr{R} := \{y \in \mathbb{R}^{n-m} : F^\top y \leq w, y \geq 0\},$$

and

$$\phi(y) := y^\top Q y + 2d^\top y + v, \quad \forall y \in \mathbb{R}^{n-m}.$$

To generate a valid cutting plane, we search for a nonnegative vector $\theta \in \mathbb{R}^{n-m}_+$ such that

$$\max\{\phi(y) : y \in \mathscr{R} \cap \Delta(\theta)\} < v_R. \tag{37}$$

Here

$$\Delta(\theta) := \left\{ y \in \mathbb{R}^{n-m} : \sum_{i=1}^{n-m} y_i \theta_i \leq 1, y \geq 0 \right\}$$

is a simplex.

The nonnegative vector θ satisfying (37) yields a valid cut $\{x \in \mathbb{R}^n : \theta^\top x_N \geq 1\}$. Indeed, (37) is equivalent to that

$$\begin{array}{rll} v_R > \max & x^\top H x + 2p^\top x & \\ \text{s.t.} & Ax = b & \\ & x \geq 0 & \\ & \theta^\top x_N \leq 1. & \end{array} \tag{38}$$

Therefore, we have

$$\Phi^*(\mathscr{F}_{N,\theta}) \leq \Phi^*(\mathscr{F}) \leq \max\left(v_R, \Phi^*(\mathscr{F}_{N,\theta})\right),$$

where $\mathscr{F}_{N,\theta}$ is the intersection of the original feasible region $\mathscr{F}$ and the cut $\{x \in \mathbb{R}^n : \theta^\top x_N \geq 1\}$:

$$\mathscr{F}_{N,\theta} := \mathscr{F} \cap \{x \in \mathbb{R}^n : \theta^\top x_N \geq 1\}.$$

It is important to note that $\bar{x} \notin \mathscr{F}_{N,\theta}$ and hence $\mathscr{F}_{N,\theta}$ is strictly included in $\mathscr{F}$. In order to determine (36), it suffices to determine whether

$$v_R > \Phi^*(\mathscr{F}_{N,\theta}) \quad \text{or} \quad \Phi^*(\mathscr{F}_{N,\theta}) \geq v_R \ ? \tag{39}$$

We then need to deal with a problem with a strictly smaller feasible region $\mathscr{F}_{N,\theta}$. Repeating this procedure of adding valid cuts, the feasible region is reduced at each iteration until we find an answer to (39).

In the following, we present different methods to find a positive vector θ satisfying (37).

3.2 Tuy's and Konno's cut revisited

We first give a brief review of Tuy and Konno's work [16,17,31]. Tuy [31] proposed to use the following value

$$\max\{\phi(y) : y \in \Delta(\theta)\} = \max\left(\phi\left(\theta_1^{-1}e_1\right), \ldots, \phi\left(\theta_{n-m}^{-1}e_{n-m}\right)\right) \tag{40}$$

as an upper bound of $\max\{\phi(y) : y \in \mathscr{R} \cap \Delta(\theta)\}$. Let $\delta > 0$ and

$$\tau_i := \frac{Q_{ii}}{-d_i + \sqrt{d_i^2 + Q_{ii}(v_R - \delta - v)}}, \quad i = 1, \ldots, n-m. \tag{41}$$

Then we have

$$\max\{\phi(y) : y \in \mathscr{R} \cap \Delta(\tau)\} \leq \max\{\phi(y) : y \in \Delta(\tau)\} = v_R - \delta < v_R. \tag{42}$$

This cut is known as Tuy's cut. The perturbation parameter $\delta > 0$ ensures that v_R is a strict upper bound of the maximum value of the region $\mathscr{R} \cap \Delta(\tau)$ so that $\{y \in \mathbb{R}^{n-m} : \tau^\top y \geq 1\}$ is a valid cut. Note that one can choose $\delta > 0$ to be arbitrarily small.

Next we recall the method proposed by Konno in [16], for improving any valid cut including in particular Tuy's cut. Denote by $\mathscr{R}_\tau \subset \mathbb{R}^{n-m}$ the region obtained after adding Tuy's cut:

$$\mathscr{R}_\tau := \{y \in \mathbb{R}^{n-m} : F^\top y \leq w, y \geq 0, \tau^\top y \geq 1\}.$$

It is clear that we can assume $\mathscr{R}_\tau \neq \emptyset$. Let any $\theta > 0$ and define $\phi^*_{\tau,\theta}$ to be:

$$\begin{aligned} \phi^*_{\tau,\theta} := \max \quad & \phi(y) \\ \text{s.t.} \quad & y \in \mathscr{R}_\tau \cap \Delta(\theta). \end{aligned} \tag{43}$$

Define the function $\psi : \mathbb{R}^{n-m} \times \mathbb{R}^{n-m} \to \mathbb{R}$ as follows:

$$\psi(y, \tilde{y}) := \tilde{y}^T Q y + d^\top \tilde{y} + d^\top y + v, \quad \forall y, \tilde{y} \in \mathbb{R}^{n-m}.$$

This is a relaxed bilinear form of ϕ. Since $Q \succeq 0$, it can be checked that:

$$\psi(y, \tilde{y}) \leq \frac{\phi(\tilde{y}) + \phi(y)}{2} \leq \max\left(\phi(\tilde{y}), \phi(y)\right), \quad \forall y, \tilde{y} \in \mathbb{R}^{n-m}, \tag{44}$$

and

$$\begin{aligned} \phi^*_{\tau,\theta} = \max \quad & \psi(\tilde{y}, y) \\ \text{s.t.} \quad & \tilde{y}, y \in \mathscr{R}_\tau \cap \Delta(\theta). \end{aligned} \tag{45}$$

For a proof of (44) and (45), see Appendix A. Konno [16] proposed to relax (45) as follows:

$$\begin{aligned} \bar{\phi}^K_{\tau,\theta} := \max \quad & \psi(\tilde{y}, y) \\ \text{s.t.} \quad & \tilde{y} \in \mathscr{R}_\tau, y \in \Delta(\theta). \end{aligned} \tag{46}$$

It is clear that $\bar{\phi}^K_{\tau,\theta} \geq \phi^*_{\tau,\theta}$. Define the function $g : \mathbb{R}^{n-m} \to \mathbb{R}$ as follows:

$$g(y) := \max_{\tilde{y} \in \mathscr{R}_\tau} \tilde{y}^\top Q y + d^\top \tilde{y} + d^\top y + v, \quad \forall y \in \mathbb{R}^{n-m}.$$

It is easy to see that g is a convex function and

$$\bar{\phi}^K_{\tau,\theta} = \max_{y\in\Delta(\theta)} g(y) = \max\left(g(\theta_1^{-1}e_1),\ldots,g(\theta_{n-m}^{-1}e_{n-m})\right). \tag{47}$$

In order to obtain $\bar{\phi}^K_{\tau,\theta}$, it suffices to find the values of $\{g(\theta_i^{-1}e_i) : i \in [n-m]\}$ and each of these $n-m$ values can be computed by solving an LP program.

In the following discussion we assume that

$$\bar{\phi}^K_{\tau,\tau} < v_R. \tag{48}$$

In fact, if (48) does not hold, then by (47), $g(\tau_i^{-1}e_i) \geq v_R$ for some $i \in [n-m]$ and there is $y \in \mathscr{R}_\tau$ such that

$$\psi(y, \tau_i^{-1}e_i) \geq v_R,$$

which together with (44) implies that

$$\max(\phi(y), v_R) \geq \max(\phi(y), \phi(\tau_i^{-1}e_i)) \geq v_R.$$

Hence, we conclude that $\Phi^*(\mathscr{F}) \geq v_R$ and the reference value problem is solved.

Given (48), we let $\delta > 0$ be a perturbation parameter[2] satisfying

$$\delta \leq v_R - \bar{\phi}^K_{\tau,\tau}, \tag{49}$$

and θ_K be the smallest $\theta \geq 0$ such that

$$\bar{\phi}^K_{\tau,\theta} = v_R - \delta < v_R. \tag{50}$$

Then $\{y \in \mathbb{R}^{n-m} : \theta_K^\top y \geq 1\}$ is a valid cut, which will be referred to as Konno's cut.

In view of (49) and (50), Konno's cut is deeper than Tuy's cut, i.e.,

$$\theta_K \leq \tau.$$

In addition, Konno [17, Thm 3.3] showed that the search of θ_K can be done by solving exactly $n-m$ LP programs, see Appendix C for details.

In the remaining of this section, we will present a new method to further improve Konno's cut. Recall that the key idea in generating Konno's cut is to give a computable upper bound on $\phi^*_{\tau,\theta} = \max\{\phi(y) : y \in \mathscr{R}_\tau \cap \Delta(\theta)\}$. For finding deeper cuts, we next propose two computable upper bounds of $\phi^*_{\tau,\theta}$ which are tighter than $\bar{\phi}^K_{\tau,\theta}$.

[2] By abuse of notation, the same symbol δ is used in several places to denote possibly different perturbation parameters.

3.3 Tighter bound from LP relaxation

We start with an estimate of $\bar{\phi}^K_{\tau,\theta}$.

Lemma 7 *If $\mathscr{R}_\tau$ is nonempty, then for any $\theta > 0$ we have*

$$\begin{aligned} \bar{\phi}^K_{\tau,\theta} \geq \min \quad & \beta + v \\ \text{s.t.} \quad & Q \leq -\Lambda_0 \tau^\top - \tau \Lambda_0^\top + \Lambda F^\top + F \Lambda^\top - \frac{1}{2}\left(d\theta^\top + \theta d^\top \right) \\ & d \leq \beta \theta + 2\Lambda_0 - 2\Lambda w \\ & \Lambda_0 \geq 0, \Lambda \geq 0. \end{aligned} \tag{51}$$

Proof Let $\beta = \bar{\phi}^K_{\tau,\theta} - v$. In view of (47),

$$g(\theta_i^{-1} e_i) - v = \max_{y \in \mathscr{R}_\tau} \left(\theta_i^{-1} Q_i + d \right)^\top y + \theta_i^{-1} d_i \leq \beta, \ \ \forall i \in [n-m].$$

Consequently, for each $i \in [n-m]$, the following system (on $y \in \mathbb{R}^{n-m}$) is infeasible:

$$\begin{cases} \left(\theta_i^{-1} Q_i + d \right)^\top y > \beta - \theta_i^{-1} d_i \\ \tau^\top y \geq 1 \\ F^\top y \leq w \\ y \geq 0 \end{cases} \tag{52}$$

By Farkas lemma and $\mathscr{R}_\tau \neq \emptyset$, for each $i \in [n-m]$, the infeasibility of (52) implies the feasibility of the following system (on $\lambda_0^i \in \mathbb{R}$ and $\lambda^i \in \mathbb{R}^m$):

$$\begin{cases} \theta_i^{-1} Q_i + d \leq -\lambda_0^i \tau + F \lambda^i \\ \beta - \theta_i^{-1} d_i \geq -\lambda_0^i + (\lambda^i)^\top w \\ \lambda_0^i \geq 0, \lambda^i \geq 0 \end{cases} \tag{53}$$

Multiplying both sides of the inequalities by the positive number θ_i, the system (53) can be equivalently written as

$$\begin{cases} Q_i + \theta_i d \leq -\lambda_0^i \tau + F \lambda^i \\ \beta \theta_i - d_i \geq -\lambda_0^i + (\lambda^i)^\top w \\ \lambda_0^i \geq 0, \lambda^i \geq 0 \end{cases} \tag{54}$$

The fact that the system (54) is solvable for each $i \in [n-m]$ implies that the following system on $(\Lambda_0 \in \mathbb{R}^{n-m}, \Lambda \in \mathbb{R}^{(n-m)\times m})$ is solvable:

$$\begin{cases} Q \leq -\Lambda_0 \tau^\top - \tau \Lambda_0^\top + \Lambda F^\top + F \Lambda^\top - \frac{1}{2}\left(d\theta^\top + \theta d^\top \right) \\ d \leq \beta \theta + 2\Lambda_0 - 2\Lambda w \\ \Lambda_0 \geq 0, \Lambda \geq 0 \end{cases} \tag{55}$$

To see this, just let $\Lambda_0 = \frac{1}{2}\left(\lambda_0^1 \ \cdots \ \lambda_0^{n-m} \right)^\top$ and $\Lambda = \frac{1}{2}\left(\lambda^1 \ \cdots \ \lambda^{n-m} \right)^\top$ and use the fact that $Q = Q^\top$. □

Remark 5 In general the inequality in (51) is strict since (55) cannot imply (54).

We next show that the lower bound of $\bar{\phi}^K_{\tau,\theta}$ provided by Lemma 7 is also an upper bound of $\phi^*_{\tau,\theta}$. For this, consider the following extended LP program:

$$\begin{aligned}\bar{\phi}^L_{\tau,\theta} := \min \quad & -\alpha_0 + \alpha^\top w + \beta + v \\ \text{s.t.} \quad & Q \le -\Lambda_0\tau^\top - \tau\Lambda_0^\top + \Lambda F^\top + F\Lambda^\top + \left(\Lambda_{m+1}\theta^\top + \theta\Lambda_{m+1}^\top\right) \\ & 2d - 2\Lambda_0 + 2\Lambda w + 2\Lambda_{m+1} \le -\alpha_0\tau + F\alpha + \beta\theta \\ & \Lambda_0 \ge 0, \Lambda \ge 0, \Lambda_{m+1} \ge 0, \alpha_0 \ge 0, \alpha \ge 0, \beta \ge 0.\end{aligned} \tag{56}$$

Proposition 4 *For any positive vector $\theta \in \mathbb{R}^{n-m}$, we have*

$$\bar{\phi}^K_{\tau,\theta} \ge \bar{\phi}^L_{\tau,\theta} \ge \phi^*_{\tau,\theta}. \tag{57}$$

Proof Any feasible solution to the optimization problem in (51) can be extended to a feasible solution of (56) by letting

$$\alpha_0 = 0, \;\; \alpha = 0, \;\; \Lambda_{m+1} = -d/2.$$

(Recall that we have chosen a basis so that $d \le 0$.) Then we apply Lemma 7 to obtain the first inequality.

Let $\Lambda_0, \Lambda, \Lambda_{m+1}, \alpha, \beta$ be feasible to (56). Then for any $y \in \mathscr{R}_\tau \cap \Delta(\theta)$, we have

$$\begin{aligned} y^\top Q y &\le -2y^\top\tau\Lambda_0^\top y + 2y^\top\Lambda F^\top y + 2y^\top\Lambda_{m+1}\theta^\top y \\ &\le -2\Lambda_0^\top y + 2y^\top\Lambda w + 2y^\top\Lambda_{m+1}. \end{aligned}$$

Here, the first inequality follows from the first constraint of (56) and $y \ge 0$. The second inequality follows from the nonnegativity of $\Lambda_0^\top y$, $\Lambda^\top y$ and $\Lambda_{m+1}^\top y$ and the fact that $\theta^\top y \le 1$, $\tau^\top y \ge 1$ and $F^\top y \le w$. Now we use the second constraint of (56) to obtain

$$\begin{aligned} y^\top Q y &\le -2d^\top y - \alpha_0\tau^\top y + \alpha^\top F^\top y + \beta\theta^\top y \\ &\le -2d^\top y - \alpha_0 + \alpha^\top w + \beta. \end{aligned}$$

Hence

$$\phi(y) \le -\alpha_0 + \alpha^\top w + \beta + v, \;\; \forall y \in \mathscr{R}_\tau \cap \Delta(\theta),$$

which implies $\phi^*_{\tau,\theta} \le \bar{\phi}^L_{\tau,\theta}$. □

Based on Proposition 4, the vector θ such that $\bar{\phi}^L_{\tau,\theta} = v_R - \delta$ leads to a cut deeper than Konno's cut. Unfortunately, the search of such θ cannot be done by solving $n-m$ LP programs as for the search of θ satisfying (50). Nevertheless, we can try to improve Konno's cut through the standard bisection trick. Specifically, let θ_K be the vector generating Konno's cut and choose some $\theta < \theta_K$. If $\bar{\phi}^L_{\tau,\theta} \le v_R - \delta$, then we get an improved cut. Otherwise, we increase θ, for example to $(\theta + \theta_K)/2$, and repeat until we get an improved cut.

3.4 Deeper cut by doubly nonnegative relaxation

Let $\hat{F} = \begin{pmatrix} F & \theta & -\tau \end{pmatrix}$ and $\hat{w} = \begin{pmatrix} w \\ 1 \\ -1 \end{pmatrix}$ so that (43) can be written as

$$\begin{aligned} \phi^*_{\tau,\theta} = \max \quad & y^\top Q y + 2d^\top y + v \\ \text{s.t.} \quad & \hat{F}^\top y \le \hat{w} \\ & y \ge 0. \end{aligned} \tag{58}$$

Note that (58) takes the same form as (17). So all the results on the DNN relaxation of (17) presented in Section 2 can be directly adapted to the DNN relaxation of (58). Denote by $\bar{\phi}^D_{\tau,\theta}$ the DNN relaxation value of (58), i.e.,

$$\begin{aligned} \bar{\phi}^D_{\tau,\theta} := \max_{\substack{Y \in \mathscr{S}^{n-m} \\ y \in \mathbb{R}^{n-m}}} \quad & \langle Q, Y \rangle + 2d^\top y + v \\ \text{s.t.} \quad & \hat{F}^\top y \le \hat{w}, y \ge 0 \\ & \hat{F}^\top Y \hat{F} - \hat{w} y^\top \hat{F} - \hat{F}^\top y \hat{w}^\top + \hat{w}\hat{w}^\top \ge 0 \\ & Y \ge 0 \\ & \hat{w} y^\top - \hat{F}^\top Y \ge 0 \\ & \begin{pmatrix} Y & y \\ y^\top & 1 \end{pmatrix} \succeq 0. \end{aligned} \tag{59}$$

Using directly Proposition 3 we get the following upper bound of $\bar{\phi}^D_{\tau,\theta}$.

Corollary 1 *If there exists $\Lambda_0, \Lambda_{m+1} \in \mathbb{R}^{n-m}_+$, $\Lambda \in \mathbb{R}^{(n-m)\times m}_+$ and $P \in \mathbb{R}^{(n-m)\times(n-m)}_+ \cap S^{n-m}$ such that*

$$-Q - P - \Lambda_0 \tau^\top - \tau \Lambda_0^\top + \Lambda F^\top + F \Lambda^\top + \Lambda_{m+1} \theta^\top + \theta \Lambda_{m+1}^\top \succeq 0. \tag{60}$$

Then

$$\bar{\phi}^D_{\tau,\theta} \le v + 2 \max_{y \in \mathscr{R}_\tau \cap \Delta(\theta)} (d - \Lambda_0 + \Lambda w + \Lambda_{m+1})^\top y.$$

Proof Obvious from Proposition 3 since $\Lambda_0, \Lambda_{m+1}, \tau, \theta$ are all nonnegative. □

Theorem 2 *For any positive vector $\theta \in \mathbb{R}^{n-m}_{++}$, we have*

$$\bar{\phi}^K_{\tau,\theta} \ge \bar{\phi}^L_{\tau,\theta} \ge \bar{\phi}^D_{\tau,\theta} \ge \phi^*_{\tau,\theta}. \tag{61}$$

Proof The first inequality follows from Proposition 4. Let $\Lambda_0, \Lambda, \Lambda_{m+1}, \alpha_0, \alpha, \beta$ be feasible to (56). Then (60) is satisfied by taking

$$P = -Q - \Lambda_0 \tau^\top - \tau \Lambda_0^\top + \Lambda F^\top + F \Lambda^\top + \Lambda_{m+1} \theta^\top + \theta \Lambda_{m+1}^\top \ge 0.$$

Corollary 1 allows to deduce:

$$\bar{\phi}^D_{\tau,\theta} \le v + 2 \max_{y \in \mathscr{R}_\tau \cap \Delta(\theta)} (d - \Lambda_0 + \Lambda w + \Lambda_{m+1})^\top y.$$

Now we use the second constraint of (56) to obtain

$$\begin{aligned}
&2 \max_{y \in \mathscr{R}_\tau \cap \Delta(\theta)} (d - \Lambda_0 + \Lambda w + \Lambda_{m+1})^\top y \\
&\leq \max_{y \in \mathscr{R}_\tau \cap \Delta(\theta)} (-\alpha_0 \tau + F\alpha + \beta\theta)^\top y \\
&\leq -\alpha_0 + \alpha^\top w + \beta.
\end{aligned}$$

It follows that

$$\bar{\phi}^L_{\tau,\theta} \geq \bar{\phi}^D_{\tau,\theta}.$$

□

Theorem 2 provides a foundation of the new cutting plane method that we are about to propose. In order to search for a vector θ such that (37) holds, we choose a perturbation parameter $\delta > 0$ and start by computing a vector θ_K such that

$$\bar{\phi}^K_{\tau,\theta_K} = v_R - \delta.$$

Then we choose a factor $\eta \in (0,1)$ and test whether $\theta = \eta\theta_K$ satisfies

$$\bar{\phi}^L_{\tau,\theta} \leq v_R - \delta. \tag{62}$$

If (62) holds, then θ yields a valid cut, which is deeper than Konno's cut. Otherwise we further check if

$$\bar{\phi}^D_{\tau,\theta} \leq v_R - \delta. \tag{63}$$

If (63) holds, then θ yields a valid cut, which is deeper than Konno's cut. The pseudocode of this process is given in Algorithm 6. One can also use the bisection trick mentioned in the end of Section 3.3. In any case, the core theory behind is Theorem 2, which guarantees that

$$\bar{\phi}^D_{\tau,\theta_K} \leq \bar{\phi}^L_{\tau,\theta_K} \leq \bar{\phi}^K_{\tau,\theta_K} = v_R - \delta < v_R,$$

and provides a way to improve Konno's cut.

4 Algorithm

In this section we summarize our method in Algorithm 1 and Algorithm 2. For the future reference, we shall name our algorithm as `QuadProgCD`, where the letter "C" refers to the cutting plane method and the letter "D" refers to the doubly nonnegative relaxation.

The algorithm has two variants: 1. `QuadProgCD-R` (Algorithm 1) is designed for solving the reference value problem (36), where we use the suffix `-R` to refer to the reference value problem; 2. `QuadProgCD-G` (Algorithm 2) is designed for solving the global optimization problem (1), where the suffix `-G` refers to the global optimization mode.

4.1 Algorithm for the reference value problem

Algorithm 1 takes as input the feasible region $\mathscr{F}$, the quadratic objective function Φ, the reference value v_R, an initial lower bound $\underline{v}$ of the optimal value $\Phi^*(\mathscr{F})$ of (1), an initial upper bound $\bar{v}$ of $\Phi^*(\mathscr{F})$, a factor $\eta \in (0,1)$, and a perturbation parameter $\delta > 0$. The algorithm solves the reference value problem (36) by returning a lower bound $\underline{v}$ of $\Phi^*(\mathscr{F})$ and an upper bound $\bar{v}$ of $\Phi^*(\mathscr{F})$ such that $v_R \notin (\underline{v}, \bar{v}]$.

Algorithm 1 starts by searching a KKT point $\bar{x}$ of (1) (line 3). There are many ways to achieve a KKT point and we adopt the mountain climbing algorithm proposed by Konno in [17]. The details are recalled in Appendix A.

If $\Phi(\bar{x})$ is larger than or equal to v_R, then we obtain a lower bound $\underline{v}$ such that $v_R \le \underline{v} \le \Phi^*(\mathscr{F})$. Otherwise, we compute the DNN bound $\bar{\Phi}(\mathscr{F})$. If the latter is strictly smaller than v_R, then we obtain an upper bound $\bar{v}$ such that $v_R > \bar{v}$ (note that if $\mathscr{F} = \varnothing$, we would also reach this case because $\bar{\Phi}(\mathscr{F}) = -\infty$). Otherwise, we proceed to the cutting plane step (from line 13 to line 28).

In line 13 we choose suitable basic and nonbasic variables associated with $\bar{x}$ and compute the parameters (Q, d, v, F, w, N) defining the minimal program (17). For details of this step please check Appendix B. In line 14 we compute the vector τ which defines Tuy's cut. From line 15 to line 21 we check whether $\bar{\phi}^K_{\tau,\tau} < v_R$. If not, then we obtain a lower bound $\underline{v}$ such that $v_R \le \underline{v} \le \Phi^*(\mathscr{F})$, see the discussion around (48). Otherwise, we proceed to generate a deeper cut (from line 22 to line 28).

We first reset δ to ensure (49). Here, line 23 ensures (49) because of the fact that $\bar{\phi}^K_{\tau,\tau} \le (\underline{v} + v_R)/2$, which can be deduced from (44). We provide two options for the generation of a cut deeper than Tuy's cut. The first option is to use Konno's cut as recalled in Section 3.2. The second option is to generate an even deeper cut using the theory that we developed in Section 3.4. The two algorithms for option I and II are described in Appendix C and Appendix D.

In the while loop, the lower bound $\underline{v}$ is nondecreasing and the upper bound $\bar{v}$ is nonincreasing. When we solve the reference value problem (36), the while loop breaks when either $v_R \le \underline{v}$ or $v_R > \bar{v}$ so that the reference value problem is solved.

4.2 Algorithm for global solution of the concave QP problem

Now we present `QuadProgCD-G` as in Algorithm 2 for globally solving the concave QP problem (1). `QuadProgCD-G` follows a similar structure as Algorithm 1, with some differences that we detail below.

The input of Algorithm 2 includes a gap tolerance $\varepsilon > 0$. The output is a lower bound $\underline{v}$ and an upper bound $\bar{v}$ of $\Phi^*(\mathscr{F})$ such that

$$\frac{\bar{v} - \underline{v}}{\max\{\varepsilon, |\underline{v}|\}} \le \varepsilon. \tag{64}$$

Here, we take the maximum of ε and $|\underline{v}|$ in the denominator to handle the pathological case $|\underline{v}| = 0$.

There is no longer a prescribed reference value v_R. Instead, Algorithm 2 uses a dynamic reference value, which is set to be the lower bound $\underline{v}$ and updated throughout

Algorithm 1 `QuadProgCD-R`

Input: Feasible region $\mathscr{F}$, quadratic objective function Φ, reference value v_R, factor $\eta \in (0,1)$, perturbation $\delta > 0$.

1: $\underline{v} \leftarrow -\infty, \bar{v} \leftarrow \infty$;
2: **while** $\underline{v} < v_R \leq \bar{v}$ **do**
3: $\bar{x} \leftarrow$ `Search_of_KKT_Point`($\mathscr{F}$, Φ); ▷ See Algorithm 3.
4: **if** $\Phi(\bar{x}) \geq v_R$ **then**
5: $\underline{v} \leftarrow \Phi(\bar{x})$ and break; ▷ Terminate with $\underline{v} \geq v_R$.
6: **end if**
7: $\underline{v} \leftarrow \max(\underline{v}, \Phi(\bar{x}))$; ▷ Keep the maximum lower bound value ever reached.
8: $t \leftarrow \bar{\Phi}(\mathscr{F})$; ▷ Compute the DNN bound.
9: **if** $t < v_R$ **then**
10: $\bar{v} \leftarrow \max(v_R - \delta, t)$ and break; ▷ Terminate with $\bar{v} < v_R$.
11: **end if**
12: $\bar{v} \leftarrow \min(\bar{v}, t)$;
13: $(Q, d, v, F, w, N) \leftarrow$ `Minimal_Program`($\bar{x}$, $\mathscr{F}$, Φ); ▷ See Algorithm 4.
14: Compute τ according to (41); ▷ Tuy's cut.
15: **for** $i = 1, \ldots, n-m$ **do**
16: $y^i \leftarrow \operatorname{argmax}\{(\tau_i^{-1} Q_i + d)^\top y : F^\top y \leq w, \tau^\top y \geq 1, y \geq 0\}$;
17: $\underline{v} \leftarrow \max\left(\underline{v}, (y^i)^\top Q y^i + 2d^\top y^i + v\right)$;
18: **if** $\underline{v} \geq v_R$ **then**
19: break; ▷ Terminate with $\underline{v} \geq v_R$.
20: **end if**
21: **end for**
22: **if** $\underline{v} < v_R$ **then**
23: $\delta \leftarrow \min(\delta, (v_R - \underline{v})/2)$; ▷ Reset δ to ensure (49).
24: $\theta_K \leftarrow$ `Konno_Cut`($Q, d, v, F, w, v_R, \tau, \delta$); ▷ See Algorithm 5.
25: **Option I**: $\theta \leftarrow \theta_K$; ▷ Use Konno's cut.
26: **Option II**: $\theta \leftarrow$ `DNN_Cut`($Q, d, v, F, w, v_R, \tau, \theta_K, \eta, \delta$); ▷ See Algorithm 6.
27: $\mathscr{F} \leftarrow \mathscr{F} \cap \left\{x \in \mathbb{R}^n : \theta^\top x_N \geq 1\right\}$; ▷ Add a valid cut.
28: **end if**
29: **end while**

Output: upper bound $\bar{v}$, lower bound $\underline{v}$ such that $v_R \notin (\underline{v}, \bar{v}]$.

the iterations (lines 4 and 14). The termination criteria are also modified to ensure global convergence (lines 2 and 6).

Another difference with Algorithm 1 is that in Algorithm 2 the perturbation parameter δ is set to be zero. This is because Algorithm 2 uses the dynamic reference value $v_R = \underline{v}$. As a result, there is no harm to cut off any solution with objective value equal to v_R, since we must have recorded some points with the same objective value $\underline{v}$ beforehand.

Finally, the lower bound $\underline{v}$ returned by Algorithm 2 corresponds to the objective value of a certain feasible solution of (1). The corresponding solution can be added to the output if needed.

4.3 Computation of the DNN bound

The DNN bound in line 8 of Algorithm 1 and in line 5 of Algorithm 2 can be replaced by any valid upper bound of $\Phi^*(\mathscr{F})$. In fact, as discussed in Section 2.4, it is impossible to compute the exact value of $\bar{\Phi}(\mathscr{F})$, which corresponds to the optimal value of

Algorithm 2 `QuadProgCD-G`

Input: Feasible region $\mathscr{F}$, quadratic objective function Φ, relative gap tolerance $\varepsilon > 0$, factor $\eta \in (0,1)$.

$\underline{v} \leftarrow -\infty; \bar{v} \leftarrow \infty$;
while $\bar{v} - \underline{v} > \varepsilon \max(|\underline{v}|, \varepsilon)$ **do**
  $\bar{x} \leftarrow$ `Search_of_KKT_Point`$(\mathscr{F}, \Phi)$; ▷ See Algorithm 3.
  $\underline{v} \leftarrow \max(\underline{v}, \Phi(\bar{x}))$; ▷ Keep the maximum lower bound value ever reached.
  $t \leftarrow \bar{\Phi}(\mathscr{F})$; ▷ Compute the DNN bound.
  if $t \le \underline{v}$ **then**
    $\bar{v} \leftarrow \underline{v}$ and break; ▷ Terminate and conclude $\underline{v}$ being optimal.
  end if
  $\bar{v} \leftarrow \min(\bar{v}, t)$; ▷ Keep the minimum upper bound value ever reached.
  $(Q, d, v, F, w, B, N) \leftarrow$ `Minimal_Program`$(\bar{x}, \mathscr{F}, \Phi)$; ▷ See Algorithm 4.
  Compute τ according to (41); ▷ Tuy's cut.
  for $i = 1, \ldots, n-m$ **do**
    $y^i \leftarrow \operatorname{argmax}\{(\tau_i^{-1} Q_i + d)^\top y : F^\top y \le w, \tau^\top y \ge 1, y \ge 0\}$;
    $\underline{v} \leftarrow \max\left(\underline{v}, (y^i)^\top Q y^i + 2d^\top y^i + v\right)$;
  end for
  $\theta_K \leftarrow$ `Konno_Cut`$(Q, d, v, F, w, \underline{v}, \tau, 0)$; ▷ See Algorithm 5.
  Option I: $\theta \leftarrow \theta_K$; ▷ Use Konno's cut.
  Option II: $\theta \leftarrow$ `DNN_Cut`$(Q, d, v, F, w, \underline{v}, \tau, \theta_K, \eta, 0)$; ▷ See Algorithm 6.
  $\mathscr{F} \leftarrow \mathscr{F} \cap \{x \in \mathbb{R}^n : \theta^\top x_N \ge 1\}$; ▷ Add a valid cut.
end while

Output: Upper bound $\bar{v}$, lower bound $\underline{v}$ such that $0 \le \bar{v} - \underline{v} \le \varepsilon \max(|\underline{v}|, \varepsilon)$.

the SDP problem (19). For high-dimensional problems, even an approximation with high accuracy of $\bar{\Phi}(\mathscr{F})$ is not accessible due to memory issue or time limit. We rely on Proposition 2 for obtaining a valid upper bound of $\Phi^*(\mathscr{F})$ from any approximate SDP solution. Similarly, when we compute the DNN cut in line 26 of Algorithm 1 and in line 18 of Algorithm 2, it suffices to find a valid upper bound of the optimal value of the SDP problem (59) and we again rely on Proposition 2. In this way, Algorithm 1 and Algorithm 2 are open to a wide range of SDP solvers, including in particular those that are able to provide medium accuracy solutions for high dimensional problems. This flexibility is crucial to make `QuadProgCD` competitive when the problem size increases. The price to pay is the degradation of the quality of the upper bound. Nevertheless, as we will show in the next section through extensive numerical tests, the gain appears to outweigh the loss and `QuadProgCD` shows superior performance compared with existing solvers especially for problems with large size. Finally let us mention that the convergence properties of Algorithm 1 and Algorithm 2, even assuming the exact knowledge of $\bar{\Phi}(\mathscr{F})$, appears to be a challenging problem and is left for future research.

5 Numerical Experiments

In this section, we evaluate the numerical performance of the proposed algorithms `QuadProgCD-R` (Algorithm 1) and `QuadProgCD-G` (Algorithm 2), one for solving the reference value problem (36) and the other for solving globally (1). For the sake of simplicity, when the problem type is clear, we use `QuadProgCD` to refer to both of the two variants. We compare `QuadProgCD` with two commercial solvers `CPLEX`

(version 22.1.0) [3] and `Gurobi` (version 9.5.0) [4], and also with an open-source solver `quadprogIP`[5] which is a nonconvex QP solver proposed very recently in [34].

Our algorithm `QuadProgCD` is implemented and run with `MATLAB R2021a` (version 9.10.0.1851785). In numerical experiments, the interfaces of `CPLEX` and `Gurobi` are called through a Python script. Unless otherwise specified, all the tests are performed on a Windows laptop with Intel(R) Core(TM) i7-8750H CPU 2.20 GHz, 6 cores and 16GB memory. The code of `QuadProgCD` is available at `https://github.com/tianyouzeng/QuadProgCD`.

5.1 Problem instances

We use both real and synthetic data for experiments. All the instances used for the comparison are uploaded at the above address. We describe them in detail below.

5.1.1 Real data

This set of data has a real application background in computational biology. In [36], the authors proposed an approach to detect new genome or protein sequence based on some known sequence data. A key step in their approach is to solve a set of reference value problems which can be described by (36). Due to this very special application background, the matrices and vectors H, p, A, b for describing the concave QP problem (1) are dense. Moreover, all the entries in p, A, b are nonnegative and H is a completely positive matrix.

The data is divided into two groups: one group of dimension $n = 100$ and the other group of dimension $n = 841$. For both groups, the number of rows of A is $m = 22$. The reference value is fixed to be $n(n+1)(2n+1)/6$ for all the instances. For convenience, we call the two groups BioData100 and BioData841 respectively. For BioData100, there are 317,458 instances, while for BioData841 we have only 3 instances.

5.1.2 Synthetic data

For $k_1, k_2 \in \mathbb{N}$ and $a, b \in \mathbb{R}$, we denote by $\mathscr{U}(k_1, k_2, a, b)$ a random matrix with k_1 rows and k_2 columns such that each entry is a uniform random variable in $[a, b]$ and entries are mutually independent. The synthetic data is generated randomly as follows. We first fix the data dimension n, then we generate the feasible region $\mathscr{F} := \{x : Ax = b, x \geq 0\}$ by the following steps:

1. Choose m uniformly randomly from the integers in $[0.1n, 0.5n]$;
2. Generate $A \sim \mathscr{U}(n, m, -20, 20)$;
3. Generate $x_0 \sim \mathscr{U}(n, 1, 0, 1)$;

[3] https://www.ibm.com/hk-en/analytics/cplex-optimizer

[4] https://www.gurobi.com

[5] https://github.com/xiawei918/quadprogIP

4. Let $b = Ax_0/\|x_0\|$. If $\mathscr{F}$ is bounded, remove redundant rows of A if necessary and terminate. Otherwise go to step 1.

Such a procedure guarantees that $\mathscr{F}$ is feasible and bounded so that Assumption 1 is satisfied. The objective function Φ is generated as follows:

5. Generate $U \sim \mathscr{U}(n,n,-1,1)$, $p \sim \mathscr{U}(n,1,-10,10)$;
6. Generate $h \sim \mathscr{U}(n,1,0,1)$. Let $D_0 = \mathrm{diag}(h)$ and $H_0 = UD_0U^\top$;
7. Generate $\alpha \sim \mathscr{U}(1,1,10,11)$. Set $H = n\alpha H_0/\|H_0\|$ and $D = n\alpha D_0/\|H_0\|$.

Note that the matrices and vectors generated in this way are dense.

The synthetic data is divided into seven groups: CQMAX20, CQMAX50, CQMAX100, PCQMAX20, PCQMAX50, PCQMAX100 and PCQMAX500. The integer number in the suffix refers to the dimension n of the instance. Instances with prefix CQMAX are generated following the above procedure. Instances with prefix PCQMAX are generated with a small difference in step 2 and 5: $A \sim \mathscr{U}(n,m,0,20)$, $U \sim \mathscr{U}(n,n,0,1)$ and $p \sim \mathscr{U}(n,1,0,10)$.

For problems of dimension $n \leq 100$, we solve the concave QP problem (1) by `CPLEX` or `Gurobi` to get the value $\Phi^*(\mathscr{F})$ and generate v_R in a neighborhood of $\Phi^*(\mathscr{F})$. For problem of dimension 500, as we have no access to $\Phi^*(\mathscr{F})$ using `CPLEX` or `Gurobi`, we set v_R to be a random number larger than some known lower bound of $\Phi^*(\mathscr{F})$.

5.2 Numerical results

This subsection is divided into three parts. In the first part (Section 5.2.1 and Section 5.2.2), we present numerical results for solving the reference problem with Algorithm 1. In the second part (Section 5.2.3), we show the numerical results for the global solution of (1) with Algorithm 2. In the third part (Section 5.2.4), we demonstrate the difference between option I (Konno's cut) and option II (DNN cut) in Algorithm 2.

Unless specified otherwise, we use $\eta = 1/2$ and $\delta = 10^{-6}$. The displayed computational time are all measured in seconds. The SDP problems are solved with either `MOSEK` or `SDPNAL+`. A valid upper bound based on the solution returned by the SDP solver is then computed using the formula established in Proposition 2.

5.2.1 Reference value problem with real data

In this section, we report the computational results of `QuadProgCD-R` (Algorithm 1) for solving the reference value problem arising from the new genome/protein detection problem. Recall that we have two groups of instances: BioData100 with 317,458 instances and BioData841 with 3 instances.

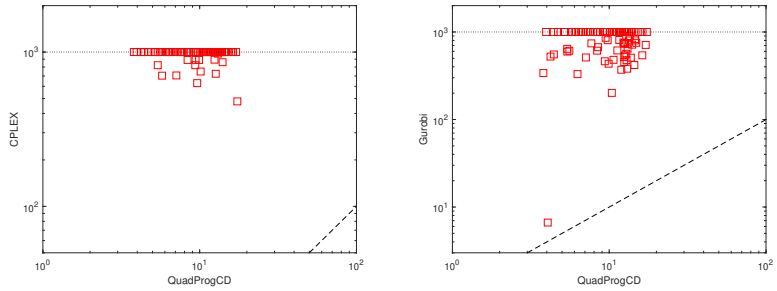
For BioData841, the test result is summarized in Table 1. `CPLEX` and `Gurobi` both failed to solve these three instances within **1000** seconds, while our algorithm `QuadProgCD` managed to solve them in less than **40** seconds each. Due to the large dimension $n = 841$, `MOSEK` fails to terminate in reasonable time limit for the inner SDP problems and we had to adaopt `SDPNAL+` for solving the inner SDP problems.

Table 1 Comparison of three algorithms on the dataset BioData841. Note that the instance with time equal to 1000 means that the corresponding solver failed to solve it within 1000 seconds.

Instance	`QuadProgCD` Time	`CPLEX` Time	`Gurobi` Time
1	**36.87**	1000	1000
2	**0.32**	1000	1000
3	**36.8**	1000	1000

For BioData100, we randomly selected 100 instances out of the 317,458 instances and tested the three algorithms for the reference value problem. The results are displayed in Figure 1. Here we follow the format used in [6] to display the comparison. Each square represents one instance, and its xy-coordinates represent the wall-clock times of the two methods. The diagonal $y = x$ line is plotted in dashed line for reference. If a square is located above the diagonal, `QuadProgCD` solves the instance faster. The far the square from the diagonal, the larger the ratio between the two wall-clock times. We set the maximum time limit to be 1000 seconds and the dotted horizontal line represents this time limit. If a square sits on the dotted horizontal line in the left (resp. right) plot, it means that `CPLEX` (resp. `Gurobi`) is not able to determine whether $v_R > \Phi^*(\mathscr{F})$ or $v_R \leq \Phi^*(\mathscr{F})$ within 1000 seconds.

From Figure 1, we observe that `CPLEX` failed to solve most of the instances within **1000** seconds. `Gurobi` seems to perform better than `CPLEX` but it also failed to solve about half of the instances within the time limit. In contrast, our algorithm `QuadProgCD` is able to solve all the 100 instances with an average wall-clock time around **10** seconds.



(a) `QuadProgCD` vs `CPLEX` on BioData100. (b) `QuadProgCD` vs `Gurobi` on BioData100.

Fig. 1 Wall-clock time comparison for solving the reference value problem on 100 instances randomly selected from BioData100, plotted in log-log scale.

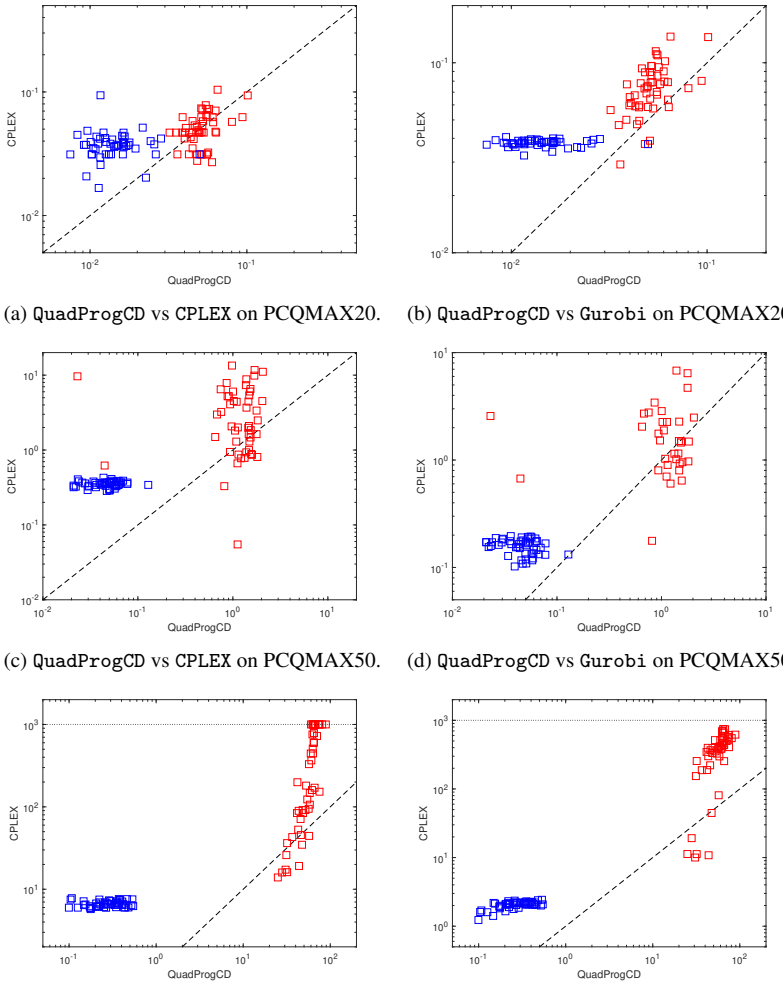
We also run our algorithm `QuadProgCD` to solve the reference value problem for all the 317,458 instances of dimension 100 on the HKU High Performance Computing cluster[6] using 32 processors at the same time. The algorithm successfully solved all the instances within **3 days** (about 5% of them are infeasible instances). However, with `CPLEX` and `Gurobi`, the computational time for one single instance may take up

[6] https://hpc.hku.hk/hpc/hpc2021/

to several hours and it is estimated to take **years** of computational time for solving all the 317,458 instances with these two solvers in the same computational environment.

5.2.2 Reference value problem with synthetic data

In this section we compare the performance of `QuadProgCD-R` (Algorithm 1), `CPLEX` and `Gurobi` for the reference problem on synthetic data PCQMAX. Each group PCQMAX20, PCQMAX50, PCQMAX100 has 100 instances. The group PCQMAX500 has 10 instances.



(a) `QuadProgCD` vs `CPLEX` on PCQMAX20. (b) `QuadProgCD` vs `Gurobi` on PCQMAX20.

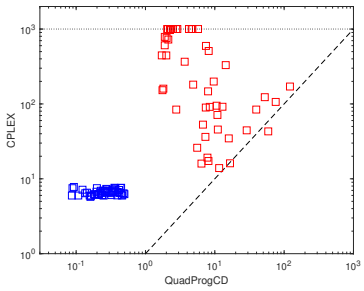
(c) `QuadProgCD` vs `CPLEX` on PCQMAX50. (d) `QuadProgCD` vs `Gurobi` on PCQMAX50.

(e) `QuadProgCD` vs `CPLEX` on PCQMAX100. (f) `QuadProgCD` vs `Gurobi` on PCQMAX100.

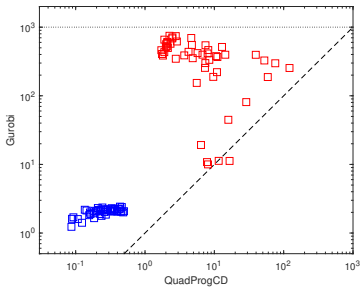
Fig. 2 Wall-clock time comparison for solving the reference value problem on PCQMAX, plotted in log-log scale, where the SDP problem in `QuadProgCD` is solved by `MOSEK`.

For instances of relatively low dimension from the dataset PCQMAX20, PCQMAX50 and PCQMAX100, it is possible to solve the doubly nonnegative relaxation using an interior point method. The results with `MOSEK` as the SDP solver are summarized in Figure 2. The red squares stand for instances for which $v_R \leq \Phi^*(\mathscr{F})$, while the blue squares stand for instances for which $v_R > \Phi^*(\mathscr{F})$. It can be observed that `QuadProgCD` outperforms significantly `CPLEX` and `Gurobi` in most of the cases. Moreover, the superior performance of `QuadProgCD` is more and more remarkable when the dimension increases.

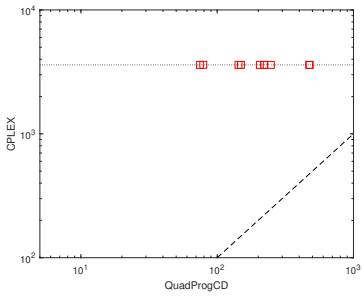
For instances of relatively large dimension from the dataset PCQMAX100 and PCQMAX500, we adopt `SDPNAL+` as the SDP solver. The results are displayed in Figure 3. By comparing Figure 2e and Figure 2f with Figure 3a and Figure 3b, we conclude that `QuadProgCD` with medium-accuracy SDP solver `SDPNAL+` can be faster than `QuadProgCD` with high-accuracy SDP solver `Mosek` on the dataset PCQMAX100. For the dataset PCQMAX500, we set the maximum time limit to be **3600** seconds. It can be seen from Figure 3c and Figure 3d that `CPLEX` and `Gurobi` failed on all the 10 instances within the time limit **3600** seconds. In contrast, `QuadProgCD` solved all the 10 instances of dimension 500 all within **1000** seconds.



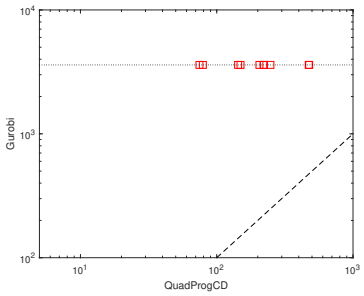
(a) `QuadProgCD` vs CPLEX on PCQMAX100.



(b) `QuadProgCD` vs Gurobi on PCQMAX100.



(c) `QuadProgCD` vs CPLEX on PCQMAX500.



(d) `QuadProgCD` vs Gurobi on PCQMAX500.

Fig. 3 Wall-clock time comparison for solving the reference value problem on PCQMAX, plotted in log-log scale, where the SDP problem in `QuadProgCD` is solved inexactly by `SDPNAL+`.

5.2.3 Global solution

In this section, we compare `QuadProgCD-G` (Algorithm 2) with three other global solvers `CPLEX`, `Gurobi` and `quadprogIP` for globally solving (1). The relative gap tolerance ε in the termination criteria (64) is set to be $\varepsilon = 10^{-6}$. The test is performed on samples from the real dataset BioData100 and from the synthetic dataset PCQMAX20, PCQMAX50, PCQMAX100, CQMAX20, CQMAX50 and CQMAX100.

Let us examine the performance of the four algorithms on real dataset displayed in Table 2. We randomly selected 20 instances from the 317,458 instances of the dataset BioData100. `QuadProgCD` reached the gap tolerance $\varepsilon = 10^{-6}$ in **hundreds** of seconds for 19 instances. `CPLEX` reached the gap tolerance $\varepsilon = 10^{-6}$ for only 4 instances within the time limit which is set to be **3600** seconds. `Gurobi` reached the gap tolerance $\varepsilon = 10^{-6}$ for only 6 instances within the time limit. `quadprogIP` reached the gap tolerance $\varepsilon = 10^{-6}$ for only 6 instances within the time limit.

Table 2 Comparison of four algorithms on 20 instances from the real dataset BioData100 for reaching the relative gap tolerance $\varepsilon = 10^{-6}$. The shortest wall-clock time in each row is in bold font. A hyphen (-) indicates that a gap less than 10^{-6} is reached, and N/A represents that the algorithm fails to obtain an upper or lower bound before the time limit. If a gap less than 10^{-6} is not reached before the time limit, then the achieved gap is displayed in the column RelGap.

	`QuadProgCD`		`CPLEX`		`Gurobi`		`quadprogIP`	
Instance	RelGap	Time	RelGap	Time	RelGap	Time	RelGap	Time
257424	-	**258.88**	-	2269.93	-	2233.72	N/A	3600
84896	-	**130.18**	-	705.71	-	3045.90	N/A	3600
141783	-	**203.66**	6.94×10^{-2}	3600	1.93×10^{-4}	3600	N/A	3600
123044	-	**124.27**	1×10^{-1}	3600	2.31×10^{-1}	3600	N/A	3600
303256	-	**47.86**	2.46×10^{-2}	3600	1.86×10^{-3}	3600	-	151.68
312988	5.85×10^{-5}	3600	-	**1486.21**	9.54×10^{-5}	3600	N/A	3600
229644	-	**57.92**	8.78×10^{-2}	3600	2.55×10^{-1}	3600	N/A	3600
259490	-	**47.21**	1.25×10^{-1}	3600	8.64×10^{-4}	3600	N/A	3600
143339	-	**124.83**	8.62×10^{-2}	3600	-	1759.16	N/A	3600
95618	-	**334.64**	2.58×10^{-2}	3600	1.71×10^{-4}	3600	N/A	3600
23659	-	**47.33**	1.45×10^{-1}	3600	2.97×10^{-1}	3600	-	901.61
284341	-	**434.11**	9.83×10^{-2}	3600	6.51×10^{-3}	3600	N/A	3600
79834	-	113.41	1.39×10^{-1}	3600	1.06×10^{-3}	3600	-	**36.23**
88552	-	**55.56**	3.13×10^{-2}	3600	1.69×10^{-1}	3600	-	143.80
305293	-	772.30	2.82×10^{-3}	3600	8.69×10^{-4}	3600	-	**124.47**
149864	-	**54.02**	3.55×10^{-2}	3600	6.69×10^{-2}	3600	N/A	3600
81426	-	**673**	5.47×10^{-2}	3600	7.20×10^{-5}	3600	N/A	3600
274863	-	**125.34**	-	1093.11	-	3220.62	N/A	3600
82857	-	**124.21**	3.83×10^{-2}	3600	-	2335.98	-	1056.03
211213	-	**204.01**	2.65×10^{-2}	3600	-	2076.43	N/A	3600

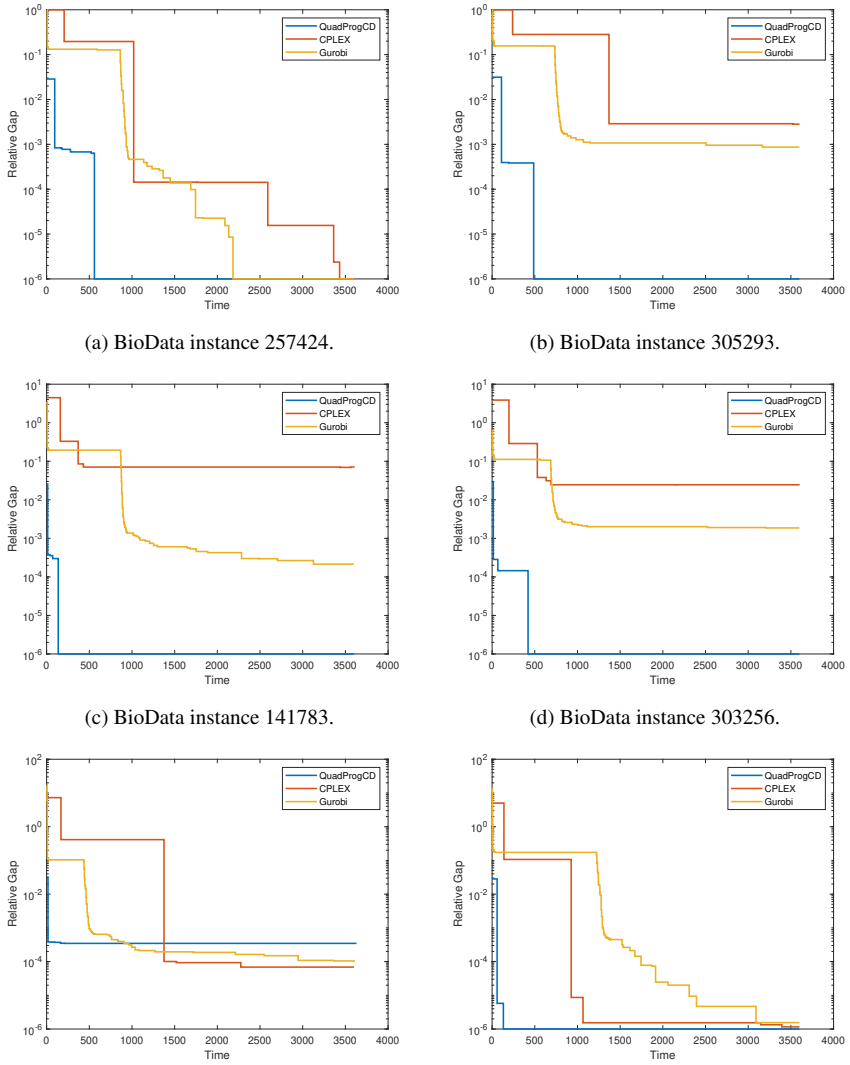
If we examine the performance of the four algorithms on the synthetic dataset in Table 3, we observe that `QuadProgCD` has comparable short running time with `CPLEX` for low dimensional instances with $n = 20$. However, when the dimension increases to $n = 100$, `QuadProgCD` outperforms the other three algorithms by a significant margin in most of the cases.

Finally we demonstrate the long time behavior of `QuadProgCD`, `CPLEX` and `Gurobi` on six selected instances from the real dataset BioData100. The plot of time versus

Table 3 Comparison of four algorithms on synthetic data PCQMAX and CQMAX for reaching the relative gap tolerance $\varepsilon = 10^{-6}$. The shortest wall-clock time in each row is in bold font. A hyphen (-) indicates that a gap less than 10^{-6} is reached, and N/A represents that the algorithm fails to obtain an upper or lower bound before the time limit. If a gap less than 10^{-6} is not reached before the time limit, then the achieved gap is displayed in the column RelGap.

	`QuadProgCD`		`CPLEX`		`Gurobi`		`quadprogIP`	
Instance	RelGap	Time	RelGap	Time	RelGap	Time	RelGap	Time
pcqmax20-1	-	0.08	-	**0.08**	-	0.20	-	0.98
pcqmax20-2	-	0.10	-	**0.05**	-	0.07	-	0.13
pcqmax20-3	-	0.17	-	**0.07**	-	0.10	-	0.98
pcqmax20-4	-	0.15	-	**0.06**	-	0.11	-	1.07
pcqmax20-5	-	0.03	-	**0.03**	-	0.06	-	0.13
pcqmax20-6	-	0.07	-	**0.03**	-	0.09	-	0.42
pcqmax20-7	-	0.31	-	**0.05**	-	0.07	-	0.13
pcqmax20-8	-	0.09	-	**0.08**	-	0.12	-	0.44
pcqmax20-9	-	0.10	-	**0.05**	-	0.08	-	1.19
pcqmax20-10	-	0.08	-	**0.06**	-	0.08	-	1.08
pcqmax50-1	-	**1.38**	-	4.27	-	41.47	N/A	3600
pcqmax50-2	-	**1.06**	-	13.11	-	34.78	N/A	3600
pcqmax50-3	-	**2.62**	-	9.63	-	42.38	N/A	3600
pcqmax50-4	-	**2.53**	-	11.77	-	35.97	N/A	3600
pcqmax50-5	-	**2.45**	-	4.79	-	40.08	N/A	3600
pcqmax50-6	-	**2.57**	-	6.05	-	60.76	N/A	3600
pcqmax50-7	-	**0.50**	-	0.55	-	0.80	-	31.83
pcqmax50-8	-	**0.94**	-	4.40	-	52.33	N/A	3600
pcqmax50-9	-	**0.63**	-	1.15	-	1.85	-	73.36
pcqmax50-10	-	2.26	-	**1.53**	-	7.84	N/A	3600
pcqmax100-1	-	**53.79**	-	76.38	-	337.77	N/A	3600
pcqmax100-2	-	**96.99**	-	3380.71	-	3427.90	N/A	3600
pcqmax100-3	-	122.83	-	**94.42**	-	550.03	N/A	3600
pcqmax100-4	-	48.37	-	**45.88**	-	298.55	N/A	3600
pcqmax100-5	-	**65.96**	-	100.63	-	358.33	N/A	3600
pcqmax100-6	-	**178.47**	1.87×10^{-2}	3600	-	2576.43	N/A	3600
pcqmax100-7	-	**148.27**	-	856.33	-	802.37	N/A	3600
pcqmax100-8	-	**77.36**	-	86.75	-	479.76	N/A	3600
pcqmax100-9	-	**106.64**	-	2738.93	-	2261.27	N/A	3600
pcqmax100-10	-	92.37	-	**48.94**	-	367.20	N/A	3600
Instance	RelGap	Time	RelGap	Time	RelGap	Time	RelGap	Time
cqmax20-1	-	0.09	-	**0.07**	-	134.77	-	1.52
cqmax20-2	-	0.08	-	**0.03**	-	445.84	-	0.38
cqmax20-3	-	0.08	-	**0.07**	-	73.46	-	0.94
cqmax20-4	-	0.11	-	**0.04**	-	10.38	-	0.53
cqmax20-5	-	0.07	-	**0.03**	-	231.45	-	1.33
cqmax20-6	-	0.08	-	**0.06**	-	11.38	-	0.68
cqmax20-7	-	0.16	-	**0.08**	-	90.88	-	1.20
cqmax20-8	-	0.07	-	**0.04**	-	1932.81	-	0.57
cqmax20-9	-	0.08	-	**0.05**	-	10.62	-	1.38
cqmax20-10	-	0.07	-	**0.06**	-	26.91	-	1.45
cqmax50-1	-	**1.14**	-	1.28	N/A	3600	N/A	3600
cqmax50-2	-	2.40	-	**1.29**	N/A	3600	N/A	3600
cqmax50-3	-	2.60	-	**1.86**	N/A	3600	N/A	3600
cqmax50-4	-	**1.27**	-	1.61	N/A	3600	N/A	3600
cqmax50-5	-	**3.02**	-	3.44	N/A	3600	N/A	3600
cqmax50-6	-	**2.80**	-	5.58	N/A	3600	N/A	3600
cqmax50-7	-	**2.38**	-	3.16	N/A	3600	N/A	3600
cqmax50-8	-	**3.29**	-	5.99	N/A	3600	N/A	3600
cqmax50-9	-	**1**	-	1.97	N/A	3600	N/A	3600
cqmax50-10	-	**0.84**	-	2.24	N/A	3600	N/A	3600
cqmax100-1	-	**109.85**	-	322.51	N/A	3600	N/A	3600
cqmax100-2	-	**39.83**	-	108.96	N/A	3600	N/A	3600
cqmax100-3	-	**107.91**	-	253.22	N/A	3600	N/A	3600
cqmax100-4	-	**116.20**	-	514.25	N/A	3600	N/A	3600
cqmax100-5	-	**30.90**	-	328.38	N/A	3600	N/A	3600
cqmax100-6	-	**187.58**	-	922.90	N/A	3600	N/A	3600
cqmax100-7	-	**101.41**	-	540.52	N/A	3600	N/A	3600
cqmax100-8	-	**309.48**	7.18×10^{-5}	3600	N/A	3600	N/A	3600
cqmax100-9	-	**196.82**	2.38×10^{-3}	3600	N/A	3600	N/A	3600
cqmax100-10	-	**115.48**	-	696.97	N/A	3600	N/A	3600

relative gap is shown in Figure 4. It is easy to conclude that `QuadProgCD` outperforms notably `CPLEX` and `Gurobi` both in terms of the convergence speed and in terms of the accuracy that can ever be reached.



(a) BioData instance 257424.

(b) BioData instance 305293.

(c) BioData instance 141783.

(d) BioData instance 303256.

(e) BioData instance 312988.

(f) BioData instance 84896.

Fig. 4 Convergence behaviour of three algorithms, with relative gap $(\bar{v}-\underline{v})/|\underline{v}|$ plotted in log scale.

5.2.4 Comparison between Konno's cut and DNN cut

Recall that Theorem 2 states:

$$\bar{\phi}^K_{\tau,\theta} \geq \bar{\phi}^L_{\tau,\theta} \geq \bar{\phi}^D_{\tau,\theta} \geq \phi^*_{\tau,\theta}, \ \ \forall \theta \geq 0. \tag{65}$$

We calculate the relative improvement of the LP relaxation bound and the DNN relaxation bound, which are defined by

$$\mathrm{RI}_L(\theta_K) := \frac{\bar{\phi}^K_{\tau,\theta_K} - \bar{\phi}^L_{\tau,\theta_K}}{\bar{\phi}^K_{\tau,\theta_K}}, \quad \mathrm{RI}(\theta_K) := \frac{\bar{\phi}^K_{\tau,\theta_K} - \bar{\phi}^D_{\tau,\theta_K}}{\bar{\phi}^K_{\tau,\theta_K}}.$$

Note that (65) implies that

$$0 \leq \mathrm{RI}_L(\theta_K) \leq \mathrm{RI}(\theta_K) \leq 1. \tag{66}$$

Let us verify (66) with numerical experiments. We randomly selected 20 instances from the real dataset BioData100 and run one iteration of `QuadProgCD`. We display the values of $(\mathrm{RI}_L(\theta_K), \mathrm{RI}(\theta_K))$ in Table 4 for those instances with nonempty feasible region after adding Konno's cut.

Table 4 Comparison of different relaxation bounds.

Instance Index	$\mathrm{RI}_L(\theta_K)$	$\mathrm{RI}(\theta_K)$
465	0.0207	0.0208
1582	0.0143	0.0152
4455	0.0163	0.0164
4571	0.0163	0.0169
6664	0.0096	0.0156
8946	0.0172	0.0172
9217	0.0112	0.0142
10824	0.0182	0.0198
11114	0.0127	0.0142
11669	0.0142	0.0153

To demonstrate the impact on generation of deeper cuts, we show in Table 5 the acceptance/rejection rate of the deepened cut $\theta = \theta_K/2$ under the criteria $\bar{\phi}^L_{\tau,\theta} \leq v_R - \delta$ and $\bar{\phi}^D_{\tau,\theta} \leq v_R - \delta$ over the dataset PCQMAX and CQMAX. The second column displays the number of instances with nonempty feasible region after adding Konno's cut in the first iteration of `QuadProgCD`. The third column displays the percentage of the instances which satisfy $\bar{\phi}^L_{\tau,\theta} \leq v_R - \delta$ and the fourth column displays the percentage of the instances which satisfy $\bar{\phi}^D_{\tau,\theta} \leq v_R - \delta$.

Finally, we compare the performance of Option I (Konno's cut) with Option II (DNN cut). We randomly selected 20 instances from the real dataset Biodata100 and set $\eta = 1/10$ in this experiment. The termination criterion is the reach of a relative gap $\varepsilon = 10^{-6}$. The time limit is set to be 1000 seconds. The result is summarized

Table 5 Acceptance/rejection rates of deepened cut $\theta := \theta_K/2$ using different criteria. Here, a valid instance refers to an instance which is feasible after adding a Konno's cut.

Dataset	Valid instances	Valid LR Cut	Valid DNNR Cut
PCQMAX20	11	100.00%	100.00%
PCQMAX50	62	98.39%	100.00%
PCQMAX100	97	89.69%	85.57%
CQMAX20	19	89.47%	100.00%
CQMAX50	57	82.46%	91.23%
CQMAX100	75	64.00%	82.67%

in Table 6. It can be seen that Option II is superior than Option I in most of the cases, which confirms the effectiveness of the proposed strategy of deepening Konno's cut through the LP and DNN relaxation.

Table 6 Comparison of Option I and with Option II in `QuadProgCD-R`. The time limit is set to 1000 seconds. The shortest wall-clock time in each row is in bold font.

Instance	Option I	Option II
252565	32.99	**18.54**
280796	103.45	**38.21**
283147	474.75	**40.65**
196032	288.48	**189.81**
30238	35.75	**21.87**
86335	37.40	**17.80**
169534	107.67	**21.17**
299116	53.08	**18.46**
48861	**1000.00**	**1000.00**
150467	40.62	**25.72**
248087	87.00	**26.14**
43985	105.71	**24.30**
130746	1000.00	**296.92**
283879	1000.00	**347.46**
245585	465.54	**25.97**
203280	241.20	**67.04**
11071	58.82	**21.99**
263231	**1000.00**	**1000.00**
289538	39.22	**24.45**
210408	**34.07**	54.26

6 Conclusion

In this paper, we develop an efficient global solver for the concave QP problem (1), which is known to be an NP-hard problem. The concave QP problem finds a recent application in computational biology, where the associated reference value problem needs to be solved for a huge number (317,584) of instances of dimension 100. The existing QP solvers such as `CPLEX` or `Gurobi` seem to require years of computational time for this particular task. By revisiting the classical cutting plane method proposed by Tuy and Konno, we propose to construct deeper cuts through the DNN relaxation. We prove that the DNN relaxation is equivalent to the Shor relaxation of

an equivalent QCQP problem. This allows to write down an SDP formulation of the DNN relaxation which satisfies Slater's condition, which is crucial for the robustness of applying the existing SDP solvers to compute the DNN bound. We also provide the explicit formula for obtaining a valid upper bound from any approximate primal and dual solution. The proposed algorithm is tested on a variety of real and synthetic instances and outperforms `CPLEX`, `Gurobi` and `quadprogIP` in most of the cases. In particular, our method successfully solved all the 317,584 instances within 3 days on the HKU HPC cluster using 32 processors. Moreover, our algorithm demonstrates superior performance on large-scale instances of dimension up to 841.

Acknowledgements The authors would like to thank Prof. Stephen S.-T. Yau for introducing the application of concave QP in computational biology and providing the dataset. We thank Xinyuan Zhang for the proofreading work. Parts of computations were performed using research computing facilities offered by Information Technology Services, the University of Hong Kong.

Statements and Declarations The work of the authors was supported by NSFC Young Scientist Fund grant 12001458 and Hong Kong Research Grants Council grant 17317122. The authors have no relevant financial or non-financial interests to disclose.

All authors contributed to the study conception and design. Implementation, data generation and numerical experiments were performed by Tianyou Zeng and Yuchen Lou. The first draft of the manuscript was written by Zheng Qu and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

The datasets generated during and/or analysed during the current study are available in the repository https://github.com/tianyouzeng/QuadProgCD.

References

1. Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., Vanthienen, J.: Benchmarking state-of-the-art classification algorithms for credit scoring. Journal of the operational research society **54**(6), 627–635 (2003)
2. Burer, S.: Copositive Programming, pp. 201–218. Springer US, Boston, MA (2012). DOI 10.1007/978-1-4614-0769-0_8. URL `https://doi.org/10.1007/978-1-4614-0769-0_8`
3. Burer, S., Vandenbussche, D.: A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. Mathematical Programming **113**(2), 259–282 (2008)
4. Burkard, R.E., Cela, E., Pardalos, P.M., Pitsoulis, L.S.: The quadratic assignment problem. In: Handbook of combinatorial optimization, pp. 1713–1809. Springer (1998)
5. Cabot, A.V., Francis, R.L.: Solving certain nonconvex quadratic minimization problems by ranking the extreme points. Operations Research **18**(1), 82–86 (1970)
6. Chen, J., Burer, S.: Globally solving nonconvex quadratic programming problems via completely positive programming. Mathematical Programming Computation **4**(1), 33–52 (2012)
7. Fung, G.: The disputed federalist papers: Svm feature selection via concave minimization. In: Proceedings of the 2003 Conference on Diversity in Computing, pp. 42–46 (2003)
8. Gondzio, J., Yıldırım, E.A.: Global solutions of nonconvex standard quadratic programs via mixed integer linear programming reformulations. Journal of Global Optimization **81**(2), 293–321 (2021)
9. Guisewite, G.M., Pardalos, P.M.: Minimum concave-cost network flow problems: Applications, complexity, and algorithms. Annals of Operations Research **25**(1), 75–99 (1990)
10. Hladík, M., Hartman, D.: Maximization of a convex quadratic form on a polytope: Factorization and the chebyshev norm bounds. In: H.A. Le Thi, H.M. Le, T. Pham Dinh (eds.) Optimization of Complex Systems: Theory, Models, Algorithms and Applications, pp. 119–127. Springer International Publishing, Cham (2020)
11. Hladik, M., Hartman, D., Zamani, M.: Maximization of a psd quadratic form and factorization. Optimization Letters **15**(7), 2515–2528 (2021)
12. Horst, R., Pardalos, P.M.: Handbook of global optimization, vol. 2. Springer Science & Business Media (2013)

13. Horst, R., Tuy, H.: Global optimization: Deterministic approaches. Springer Science & Business Media (2013)
14. Jiao, X., Pei, S., Sun, Z., Kang, J., Yau, S.S.T.: Determination of the nucleotide or amino acid composition of genome or protein sequences by using natural vector method and convex hull principle. Fundamental Research **1**(5), 559–564 (2021). DOI https://doi.org/10.1016/j.fmre.2021.08.010. URL https://www.sciencedirect.com/science/article/pii/S2667325821001564
15. Kim, S., Kojima, M., Toh, K.C.: A lagrangian–dnn relaxation: a fast method for computing tight lower bounds for a class of quadratic optimization problems. Mathematical Programming **156**(1), 161–187 (2016). DOI 10.1007/s10107-015-0874-5. URL https://doi.org/10.1007/s10107-015-0874-5
16. Konno, H.: A cutting plane algorithm for solving bilinear programs. Mathematical Programming **11**(1), 14–27 (1976)
17. Konno, H.: Maximization of a convex quadratic function under linear constraints. Mathematical programming **11**(1), 117–127 (1976)
18. Liuzzi, G., Locatelli, M., Piccialli, V.: A computational study on qp problems with general linear constraints. Optimization Letters **16**(6), 1633–1647 (2022)
19. Luo, Z.Q., Ma, W.K., So, A.M.C., Ye, Y., Zhang, S.: Semidefinite relaxation of quadratic optimization problems. IEEE Signal Processing Magazine **27**(3), 20–34 (2010)
20. Mangasarian, O.L., Street, W.N., Wolberg, W.H.: Breast cancer diagnosis and prognosis via linear programming. Operations Research **43**(4), 570–577 (1995)
21. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. Mathematical programming **10**(1), 147–175 (1976)
22. Momoh, J., Dias, L., Guo, S., Adapa, R.: Economic operation and planning of multi-area interconnected power systems. IEEE transactions on power systems **10**(2), 1044–1053 (1995)
23. Nesterov, Y.: Semidefinite relaxation and nonconvex quadratic optimization. Optimization methods and software **9**(1-3), 141–160 (1998)
24. Nohra, C.J., Raghunathan, A.U., Sahinidis, N.: Spectral relaxations and branching strategies for global optimization of mixed-integer quadratic programs. SIAM Journal on Optimization **31**(1), 142–171 (2021)
25. Pardalos, P.M., Rosen, J.B.: Methods for global concave minimization: A bibliographic survey. Siam Review **28**(3), 367–379 (1986)
26. Pardalos, P.M., Vavasis, S.A.: Quadratic programming with one negative eigenvalue is np-hard. Journal of Global optimization **1**(1), 15–22 (1991)
27. Pataki, G.: Characterizing bad semidefinite programs: Normal forms and short proofs. SIAM Review **61**(4), 839–859 (2019). DOI 10.1137/17M1140844. URL https://doi.org/10.1137/17M1140844
28. Sun, D., Toh, K.C., Yuan, Y., Zhao, X.Y.: Sdpnal+: A matlab software for semidefinite programming with bound constraints (version 1.0). Optimization Methods and Software **35**(1), 87–115 (2020)
29. Telli, M., Bentobache, M., Mokhtari, A.: A successive linear approximation algorithm for the global minimization of a concave quadratic program. Computational and Applied Mathematics **39**(4), 1–28 (2020)
30. Toh, K.C., Todd, M.J., Tütüncü, R.H.: Sdpt3 — a matlab software package for semidefinite programming, version 1.3. Optimization Methods and Software **11**(1-4), 545–581 (1999). DOI 10.1080/10556789908805762. URL https://doi.org/10.1080/10556789908805762
31. Tuy, H.: Concave programming under linear constraints. Soviet Math. **5**, 1437–1440 (1964)
32. Tuy, H.: Nonconvex quadratic programming. In: Convex Analysis and Global Optimization, pp. 337–390. Springer (2016)
33. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented lagrangian methods for semidefinite programming. Mathematical Programming Computation **2**(3), 203–230 (2010). DOI 10.1007/s12532-010-0017-1. URL https://doi.org/10.1007/s12532-010-0017-1
34. Xia, W., Vera, J.C., Zuluaga, L.F.: Globally solving nonconvex quadratic programs via linear integer programming techniques. INFORMS Journal on Computing **32**(1), 40–56 (2020)
35. Zamani, M.: A new algorithm for concave quadratic programming. Journal of Global Optimization **75**(3), 655–681 (2019)
36. Zhao, R., Pei, S., Yau, S.S.T.: New genome sequence detection via natural vector convex hull method. IEEE/ACM Transactions on Computational Biology and Bioinformatics (2020)

A Search of KKT Vertex

Consider problem (1). Define the bilinear function $\Psi : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as follows:

$$\Psi(x,\tilde{x}) := x^\top H\tilde{x} + p^\top x + p^\top \tilde{x}, \ \ \forall x,\tilde{x} \in \mathbb{R}^n.$$

It can be checked that

$$\Psi(x,\tilde{x}) = \Phi(x) + \frac{1}{2}\langle \nabla\Phi(x), \tilde{x}-x\rangle = \Phi(\tilde{x}) + \frac{1}{2}\langle \nabla\Phi(\tilde{x}), x-\tilde{x}\rangle, \ \ \forall x,\tilde{x} \in \mathbb{R}^n. \tag{67}$$

It follows from (67) that

$$\Psi(x,\tilde{x}) = \frac{\Phi(x)+\Phi(\tilde{x})}{2} - \frac{1}{4}\langle \nabla\Phi(x) - \nabla\Phi(\tilde{x}), x-\tilde{x}\rangle \le \frac{\Phi(x)+\Phi(\tilde{x})}{2}, \ \ \forall x,\tilde{x} \in \mathbb{R}^n, \tag{68}$$

where the inequality follows from the convexity of the function Φ. Based on (68), we deduce immediately the following result, see [17, Thm 2.2].

Theorem 3 ([17]) *For any subset $\mathscr{X} \subset \mathbb{R}^n$, we have*

$$\max_{x\in\mathscr{X}} \Phi(x) = \max_{x\in\mathscr{X},\tilde{x}\in\mathscr{X}} \Psi(x,\tilde{x}).$$

Theorem 3 implies that (1) is equivalent to the following bilinear program.

$$\begin{array}{ll} \max & \Psi(x,\tilde{x}) \\ \text{s.t.} & x \in \mathscr{F}, \tilde{x} \in \mathscr{F} \end{array} \tag{69}$$

In [16], Konno proposed a *mountain climbing* algorithm for (69), which corresponds to alternatively maximizing over x and $\tilde{x}$.

Algorithm 3 Mountain Climbing Algorithm [16] (`Search_of_KKT_Point`)

Input: $x^0 \in \mathbb{R}^n$, feasible region $\mathscr{F}$

1: **while** true **do**
2: $x^{k+1} \in V(\mathscr{F}) \cap \operatorname{argmax}\{\Psi(x^k,x) : x \in \mathscr{F}\}$
3: **if** $\Psi\left(x^{k+1},x^{k+1}\right) = \Psi\left(x^k,x^k\right)$ **then**
4: $\bar{x} \leftarrow x^k$;
5: break;
6: **end if**
7: $k \leftarrow k+1$
8: **end while**

Output: $\bar{x} \in V(\mathscr{F})$ such that

$$\bar{x} \in V(\mathscr{F}) \cap \operatorname{argmax}\{\Psi(\bar{x},x) : x \in \mathscr{F}\}. \tag{70}$$

Since $V(\mathscr{F})$ is a finite set, Algorithm 3 terminates in finitely many iterations. The main computational step in Algorithm 3 is at line 2, where one needs to solve an LP problem. Note also that the starting point x^0 does not need to be a feasible solution. In view of (67), (70) is equivalent to

$$\begin{array}{rll} 0 = \max & \langle \nabla\Phi(\bar{x}), x-\bar{x}\rangle \\ \text{s.t.} & x \in \mathscr{F}. \end{array} \tag{71}$$

Therefore, the output $\bar{x}$ of Algorithm 3 is a KKT point of problem (1).

It is interesting to note that step 2 in Algorithm 3 can be written equivalently as:

$$x^{k+1} \in V(\mathscr{F}) \cap \operatorname{argmax}\{\langle \nabla\Phi(x^k), x\rangle : x \in \mathscr{F}\}.$$

B Minimal Program

Let

$$\mathscr{F} = \{x \in \mathbb{R}^n : Ax = b; x \geq 0\}, \tag{72}$$

and

$$\Phi(x) = x^\top Hx + 2p^\top x. \tag{73}$$

Suppose that $\bar{x}$ is an output of Algorithm 3. Let $c = H\bar{x} + p$. Then $\bar{x}$ is a basic feasible solution and also an optimal solution of

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0. \end{aligned} \tag{74}$$

Hence, there exists a basis matrix B associated with $\bar{x}$ such that

$$c_B^\top B^{-1} N - c_N^\top \geq 0,$$

where

$$c_B = (H\bar{x} + p)_B = H_{BB}B^{-1}b + p_B, \;\; c_N = H_{BN}^\top B^{-1}b + p_N.$$

In other words, there exists a basis matrix B associated with $\bar{x}$ such that

$$(B^{-1}N)^\top \left(H_{BB}B^{-1}b + p_B\right) - H_{BN}^\top B^{-1}b - p_N \geq 0. \tag{75}$$

Algorithm 4 `Minimal_Program`

Input: Feasible region $\mathscr{F}$, objective function Φ, KKT point $\bar{x}$.
1: Write $\mathscr{F}$ and Φ as in (72) and (73);
2: Choose basis matrix B and nonbasis matrix N associated with $\bar{x}$ such that (75) holds.
3: Compute (Q, d, F, w, v) according to (13).
Output: (Q, d, F, w, v, B, N).

C Konno's Cut

Given Tuy's cut $\tau \in \mathbb{R}^{n-m}$, for each $i \in [n-m]$ and for any λ we have

$$\begin{aligned} g(\lambda e_i) = \max \quad & \lambda (Q_i)^\top y + d^\top y + d_i + v \\ \text{s.t.} \quad & \tau^\top y \geq 1 \\ & F^\top y \leq w \\ & y \geq 0. \end{aligned} \tag{76}$$

Konno's cut searches for the vector $\theta = (\theta_1, \ldots, \theta_{n-m})$ such that

$$g(\theta_1^{-1} e_1) = \cdots = g(\theta_{n-m}^{-1} e_{n-m}) = v_R - \delta. \tag{77}$$

By strong duality of LP, once the feasible region after adding Tuy's cut is nonempty, we have

$$\begin{aligned} \theta_i = \min \quad & -d^\top z + (v_R - \delta - v) z_0 \\ \text{s.t.} \quad & -Fz + wz_0 \geq 0 \\ & \tau^\top z - z_0 \geq 0 \\ & (Q_i)^\top z + d_i z_0 = 1 \\ & z \geq 0, z_0 \geq 0, \end{aligned} \tag{78}$$

for each $i \in [n-m]$. Thus Konno's cut can be computed by solving $n-m$ linear programming problems. We refer interested readers to [17,32] for more details.

Below in Algorithm 5 is the pseudo code for the generation of Konno's cut.

Algorithm 5 `Konno_Cut`

Input: $Q, d, v, F, w, v_R, \tau, \delta$.

for $i = 1, \ldots, n-m$ **do**
    Compute θ_i by solving the LP problem (78).
end for

Output: θ such that $\bar{\phi}^K_{\tau,\theta} = v_R - \delta$.

D Generation of Deeper Cut

Recall in Section 3 it is mentioned that deeper cuts can be generated using bisection tricks. Below in Algorithm 6 we attach the pseudo code for generation of those deeper cuts.

Algorithm 6 `DNN_Cut`

Input: $Q, d, v, F, w, v_R, \tau, \theta_0, \eta, \delta$.

$\theta \leftarrow \theta_0$;
Compute $\bar{\phi}^L_{\tau,\eta\theta_0}$ based on (56);
if $\bar{\phi}^L_{\tau,\eta\theta_0} \le v_R - \delta$ **then**
    $\theta \leftarrow \eta\theta_0$;
else
    Compute $\bar{\phi}^D_{\tau,\eta\theta_0}$ based on (59).
    if $\bar{\phi}^D_{\tau,\eta\theta_0} \le v_R - \delta$ **then**
        $\theta \leftarrow \eta\theta_0$;
    end if
end if

Output: $\theta \le \theta_0$ such that $\phi^*_{\tau,\theta} \le v_R - \delta$.