User manual of NewtBracket: "A Newton-Bracketing method for a simple conic optimization problem" [10] with applications to QOPs in binary variables

Sunyoung Kim*, Masakazu Kojima†, Kim-Chuan Toh‡

November 26, 2020

Abstract

We describe the Matlab package NewtBracket for solving a simple conic optimization problem that minimizes a linear objective function subject to a single linear equality constraint and a convex cone constraint. The problem is converted into the problem of finding the largest zero y^* of a continuously differentiable (except at y^*) convex function $g : \mathbb{R} \to \mathbb{R}$ such that $g(y) = 0$ if $y \leq y^*$ and $g(y) > 0$ otherwise. The Newton-Bracketing method [10] generates a sequence of lower and upper bounds of y^* both converging to y^*. For applications, we present the Lagrangian doubly nonnegative relaxation of binary quadratic optimization problems, quadratic multiple knapsack problems, quadratic assignment problems and maximum stable set problems. The MATLAB package NewtBracket can be obtained at

https://sites.google.com/site/masakazukojima1/softwares-developed/newtbracket

1 A simple conic optimization problem and the Newton-bracketing method

Let $\mathbb{R}$ the set of real numbers, $\mathbb{V}$ a finite dimensional linear space endowed with an inner product $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle$, and $\mathbb{K}_i$ a nonempty closed convex cone in $\mathbb{V}$ $(i = 1, 2)$. The precise descriptions of $\mathbb{V}$ and $\mathbb{K}_i$ $(i = 1, 1)$ will be given in the subsequent sections. We let $\boldsymbol{Q}$, $\boldsymbol{H} \in \mathbb{V}$.

The Newton-bracketing method [1, 10] is designed to solve the pair of primal-dual conic optimization problems (COPs):

$$\eta^p = \inf \left\{ \langle \boldsymbol{Q}, \boldsymbol{X} \rangle : \boldsymbol{X} \in \mathbb{K} \equiv \mathbb{K}_1 \cap \mathbb{K}_2, \langle \boldsymbol{H}, \boldsymbol{X} \rangle = 1 \right\}, \tag{1}$$

$$\eta^d = \sup \left\{ y_0 : \boldsymbol{Q} - \boldsymbol{H} y_0 = \boldsymbol{Y}_1 + \boldsymbol{Y}_2, \ y_0 \in \mathbb{R}, \ \boldsymbol{Y}_1 \in \mathbb{K}_1^*, \ \boldsymbol{Y}_2 \in \mathbb{K}_2^* \right\} \tag{2}$$

*Department of Mathematics, Ewha W. University, 52 Ewhayeodae-gil, Sudaemoon-gu, Seoul 120-750, Korea (skim@ewha.ac.kr). The research was supported by NRF 2017-R1A2B2005119.

†Department of Industrial and Systems Engineering, Chuo University, Tokyo 192-0393, Japan (kojima@is.titech.ac.jp). This research was supported by Grant-in-Aid for Scientific Research (A) 19H00808

‡Department of Mathematics, and Institute of Operations Research and Analytics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076 (mattohkc@nus.edu.sg). This research is supported in part by the Ministry of Education, Singapore, Academic Research Fund (Grant number: R-146-000-257-112).

under the following conditions:

(I) $\mathbb{K}^* = \mathbb{K}_1^* + \mathbb{K}_2^*$, where $\mathbb{J}^*$ denotes the dual cone of a cone $\mathbb{J} \subset \mathbb{V}$;

$$\mathbb{J}^* = \{\boldsymbol{Y} \in \mathbb{V} : \langle \boldsymbol{X}, \boldsymbol{Y} \rangle \geq 0 \text{ for every } \boldsymbol{X} \in \mathbb{J}\}.$$

(II) COPs (1) and (2) are feasible.

(III) $\boldsymbol{H} \in \mathbb{K}^* = \mathbb{K}_1^* + \mathbb{K}_2^*$.

(IV) The metric projection of each $\boldsymbol{X} \in \mathbb{V}$ onto $\mathbb{K}_i$ can be easily computed $(i = 1, 2)$.

(V) There exists a positive number ρ and an $\boldsymbol{I}$ in the interior of $\mathbb{K}_1^* + \mathbb{K}_2^*$ such that $\langle \boldsymbol{I}, \boldsymbol{X} \rangle \leq \rho$ for every feasible solution $\boldsymbol{X}$ of (1).

By the duality theorem, $\eta^p = \eta^d$ holds. See [1, Lemma 2.3].

Given $\mathrm{lb}^0 = -\infty$ and $\mathrm{ub}^0 > \eta^d$, the Newton-brackting method generates a sequence of intervals $[\mathrm{lb}^k, \mathrm{ub}^k]$ $(k = 1, 2, \ldots)$ such that

$$\begin{aligned}
&\mathrm{lb}^0 < \mathrm{lb}^1 \leq \mathrm{lb}^2 \leq \cdots \leq \eta^d \leq \cdots \leq \mathrm{ub}^2 \leq \mathrm{ub}^1 \leq \mathrm{ub}^0, \\
&\mathrm{lb}^k \to \eta^d = \eta^p \leftarrow \mathrm{ub}^k \text{ as } k \to \infty, \\
&\mathrm{ub}^k = \langle \boldsymbol{Q}, \boldsymbol{X}^k \rangle \text{ for some feasible solution } \boldsymbol{X}^k \text{ of (1) } (k = 1, 2, \ldots)
\end{aligned}$$

in theory. See [10] for details.

In this manual, we use the following notation and symbols.

$$\begin{aligned}
\mathbb{R}^t &= \text{the linear space of } t\text{-dimensional row vectors } \boldsymbol{x} = (x_1, \ldots, x_t), \\
\mathbb{S}^n &= \text{the linear space of } n \times n \text{ symmetric matrices } \boldsymbol{X} = [X_{ij}] \text{ with the} \\
&\quad \text{inner product } \langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} Y_{ij} \text{ for every } \boldsymbol{X}, \boldsymbol{Y} \in \mathbb{S}^n, \\
\mathbb{S}^n_+ &= \text{the cone of positive semidefinite matrices in } \mathbb{S}^n, \\
\mathbb{N}^n &= \{\boldsymbol{X} \in \mathbb{S}^n : X_{ij} \geq 0 \ (1 \leq i, j \leq n)\} \\
&\quad \text{(the cone of nonnegative symmetric matrices in } \mathbb{S}^n), \\
\mathrm{vec}(\boldsymbol{X}) &= \mathsf{reshape}(\boldsymbol{X}, 1, n * n) \text{ for every } \boldsymbol{X} \in \mathbb{S}^n \\
&\quad (\text{ the } n^2\text{-dimensional row vector obtained by row-wise vectorization of } \boldsymbol{X} \in \mathbb{S}^n).
\end{aligned}$$

2 Simple illustrative examples

Throughout this section, we assume that

$$\mathbb{V} = \mathbb{S}^n, \ \mathbb{K}_1 \ = \ \mathbb{S}^n_+ \text{ and } \ \boldsymbol{I} \ = \ \text{the } n \times n \text{ identity matrix.}$$

When we take $\mathbb{N}^n$ for $\mathbb{K}_2$ as in the first example below, the cone $\mathbb{K}_1 \cap \mathbb{K}_2$ is called the doubly nonnegative (DNN) cone. We note that the metric projection of $\boldsymbol{X} \in \mathbb{S}^n$ onto $\mathbb{S}^n_+$ is computed through the eigenvalue decomposition of $\boldsymbol{X} \in \mathbb{S}^n$, which requires $O(n^3)$ arithmetic operations in dense computation, and the one onto $\mathbb{N}^n$ is obtained from the matrix with elements $\max\{X_{ij}, 0\}$ $(1 \leq i, j \leq n)$. Here, dense computation means computing with dense matrices as opposed to sparse matrices. Hence Condition (IV) is satisfied.

Kim, Kojima and Toh in their paper [9] considered the DNN cone to which a Lagrangian-DNN relaxation of quadratic optimization problem (QOP) with linear and complementarity condition in binary variables was reduced. They proposed the bisection-projection method for solving the dual COP (2), which was released later as a software package BBCPOP [7].

We begin with two examples to show how COPs (1) and (2) can be solved by newtBracket.

Example 2.1. Let $n = 4$, $\mathbb{V} = \mathbb{S}^4$, $\mathbb{K}_1 = \mathbb{S}^4_+$, $\mathbb{K}_2 = \mathbb{N}^4$,

$$\boldsymbol{Q} = \begin{pmatrix} 0 & 3 & -4 & 0 \\ 3 & 1 & 2 & 0 \\ -4 & 2 & -1 & 2 \\ 0 & 0 & 2 & 4 \end{pmatrix} \in \mathbb{S}^4, \ \boldsymbol{H} = \frac{1}{n}\boldsymbol{I} = \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} \in \mathbb{S}^4_+ \subset \mathbb{K}_1^* + \mathbb{K}_2^* \quad (3)$$

in COPs (1) and (2). Then $\langle \boldsymbol{I}, \boldsymbol{X} \rangle = 4$ for every feasible solution $\boldsymbol{X}$ of COP (1), so that Condition (V) holds with $\rho = 4$. We also see that $\boldsymbol{X} = \boldsymbol{I} \in \mathbb{S}^4$ is a feasible solution of COP (1). Hence we can take the initial upper bound $\mathrm{ub}^0 = 1.2 \times \langle \boldsymbol{Q}, \boldsymbol{I} \rangle$ for $\eta^d = \eta^p$. The following is a MATLAB script to run newtBracket for this problem:

```
% Script file Example21
n = 4;
QMat = [ 0, 3, -4, 0; ...
         3, 1,  2, 0; ...
        -4, 2, -1, 2; ...
         0, 0   2, 4];
HMat = eye(n,n)/n;
QVect = reshape(QMat,1,n*n);
HVect = reshape(HMat,1,n*n);
K1.s = n;

% Projection onto the positive semidefinite cone.
projMap.K1     = @(x) projOntoK1(x, K1);
projMap.K1Star = @(x) projOntoK1(x, K1);

% Projection onto the nonnegative matrix cone K2.
K2 = []; % specify [] for the nonnegative matrix cone K2.
projMap.K2     = @(x) projOntoK2(x,K2);
projMap.K2Star = @(x) x + projOntoK2(-x,K2);

X0Mat = eye(n,n); % a feasible solution of COP.

% An initial lower bound for eta^d, which can be -Inf if unavailable.
LB0 = -Inf;

% An initial upper bound for eta^d must be strictly larger than eta^d
% and not so close to eta^d.
UB0 = QVect * X0Mat(:) + 0.2 * abs(QVect * X0Mat(:));
```

```
rho = n; % Since <I,X> = n for every feasible solution of COP (1).

[solNB, infoNB] ...
    = newtBracket(QVect,HVect,K1,projMap,LB0,UB0,rho);
```

Then the following is shown on the screen as newtBracket terminates in two iterations.

```
  k   [LBk,              UBk             ]     (UBk-LBk)     errmax   APGR  eTime
                                              ---------               iter
                                           max(1,|UBk|,|LBk|)
 0   [            -Inf,+4.800000000e+00]         NaN      1.7e-17   510 1.2e-01
 1   [-1.812451550e+01,-1.737846813e+01]     4.1e-02      9.4e-18   510 2.3e-01
 2   [-1.812451550e+01,-1.812404638e+01]     2.6e-05
 execution time = 0.23,  break_yes = 1
```

As a result, we obtain $\eta^p \in [\mathrm{lb}^2, \mathrm{ub}^2] = [\mathrm{LB2}, \mathrm{UB2}] = [-1.812451550\mathrm{e}{+01}, -1.812404638\mathrm{e}{+01}]$. We also obtain an approximate solution to (1):

$$\boldsymbol{X} = \mathsf{reshape(solNB.X,K1.s,K1.s)} = \begin{pmatrix} 1.7519 & 0 & 1.9846 & 0 \\ 0 & 0 & 0 & 0 \\ 1.9846 & 0 & 2.2481 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

We note that the DNN problem solved is a DNN relaxation of the QOP

$$\zeta^p \;=\; \min\left\{\boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \boldsymbol{x} \in \mathbb{R}^n_+,\ \boldsymbol{x}\boldsymbol{H}\boldsymbol{x}^T = 1\right\}.$$

If we take $\mathsf{x} = \mathsf{sqrt(diag(X))'} = (1.3236, 0, 1.4994, 0)$, then $\mathsf{X} = \mathsf{x'} * \mathsf{x}$. Hence x provides an approximate optimal solution to the QOP.

Example 2.2. Let $n = 4$, $\mathbb{V} = \mathbb{S}^4$, $\mathbb{K}_1 = \mathbb{S}^4_+$, $I_{01} = \{2,3,4\}$, $I_{\mathrm{comp}} = \{(2,3)\}$ and $\boldsymbol{Q} \in \mathbb{S}^n$ be the same as in Example 2.1. We consider the following QOP in binary variables x_2, x_3, x_4.

$$\begin{aligned} \zeta &= \min\left\{\boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \begin{array}{l} \boldsymbol{x} \in \mathbb{R}^4_+,\ x_1 = 1,\ x_i(1-x_i)\ (i \in I_{01}), \\ x_i x_j = 0\ ((i,j) \in I_{\mathrm{comp}}) \end{array}\right\} \qquad (4) \\ &= \min\left\{\langle \boldsymbol{Q},\ \boldsymbol{X}\rangle : \begin{array}{r} \boldsymbol{X} = \boldsymbol{x}^T\boldsymbol{x} \in \mathbb{S}^4_+ \cap \mathbb{N}^4,\ \langle \boldsymbol{H},\ \boldsymbol{X}\rangle = 1, \\ X_{1i} = X_{i1} = X_{ii}\ (i \in I_{01}), \\ X_{ij} = X_{ij} = 0\ ((i,j) \in I_{\mathrm{comp}}) \end{array}\right\} \\ &= \min\left\{\langle \boldsymbol{Q},\ \boldsymbol{X}\rangle : \boldsymbol{X} = \boldsymbol{x}^T\boldsymbol{x} \in \mathbb{K}_1 \cap \mathbb{K}_2,\ \langle \boldsymbol{H},\ \boldsymbol{X}\rangle = 1\right\}. \qquad (5) \end{aligned}$$

Here

$$\begin{aligned} \mathbb{K}_2 &= \left\{\boldsymbol{X} \in \mathbb{N}^4 : X_{1i} = X_{i1} = X_{ii}\ (i \in I_{01}),\ X_{ij} = X_{ij} = 0\ ((i,j) \in I_{\mathrm{comp}})\right\}, \\ \boldsymbol{H} &= \text{the } 4 \times 4 \text{ matrix whose elements are zeros except } X_{11} = 1. \end{aligned}$$

We note that the constraint $x_i(1-x_i)$ is equivalent to $x_i \in \{0,1\}$ $(i \in I_{01})$ and that the constraint $\boldsymbol{X} = \boldsymbol{x}^T\boldsymbol{x}$ to $\mathrm{rank}\boldsymbol{X} = 1$. We relax the constraint of QOP (5) by removing the rank-1 constraint $\boldsymbol{X} = \boldsymbol{x}^T\boldsymbol{x}$ to obtain the DNN relaxation problem of the form (1); $\eta^p = \eta^d \le \zeta$. A MATLAB script to execute newtBracket for this problem is:

```
% Script file Example22
% BQOP
% min        x^T QMat x
% subject to x \in \Real^4_+, x_1 = 1,
%            x_i \in {0,1} (i \in {2,3,4}),
%            x_i x_j = 0 ((i,j) \in {(2,3)}.

% Input data --->
n = 4;
QMat = [ 0, 3, -4,  0; ...
         3, 1,  2,  0; ...
        -4, 2, -1,  2; ...
         0, 0   2,  4];
HMat = sparse(1,1,1,n,n,1); % <HMat,x x^T> = x_1.
I01 = [0,1,1,1]; % <==> x_i \in {0,1} (i \in {2,3,4}.
% Fix the first element of I01 to 0 since it is already set to 1
% by <HMat,x x^T> = x_1 = 1!
Icomp = [0,1,1,0]; % <==> x_2x_3 = 0;
% if, additionally, x_3x_4 = 0, then set Icomp = [0,1,1,0; 0,0,1,1];
% <-- Input data

QVect = reshape(QMat,1,n*n);
HVect = reshape(HMat,1,n*n);
K1.s = n;

% Projection onto the positive semidefinite cone.
projMap.K1     = @(x) projOntoK1(x, K1);
projMap.K1Star = @(x) projOntoK1(x, K1);

% Projection onto the matrix cone K2 induced from the 01,
% complementarity and nonnegativity conditions, and its dual K2*.
[K2] = genK2BCQOPeq(K1,I01,Icomp);
projMap.K2     = @(x) projOntoK2(x,K2);
projMap.K2Star = @(x) x + projOntoK2(-x,K2);

x0Vect = [1, 0, 0, 0]; % a feasible solution of BQOP.

% A feasible solution of the DNN relaxation problem.
X0Mat = x0Vect' * x0Vect;

% An initial lower bound for eta^d can be -Inf if unavailable.
LB0 = -Inf;

% An initial upper bound for eta^d must be strictly larger than eta^d
% and not so close to eta^d.
UB0 = QMat(:)' * X0Mat(:) + 0.2*abs(QMat(:)' * X0Mat(:));
```

```
rho = n; % Since \sum_{i=1}^n x_i^2 \leq n.

[solNB, infoNB] ...
    = newtBracket(QVect,HVect,K1,projMap,LB0,UB0,rho);
```

Then the following is displayed on the screen as newtBracket terminates in two iterations.

```
 k  [LBk,              UBk              ]     (UBk-LBk)     errmax  APGR  eTime
                                            ---------              iter
                                        max(1,|UBk|,|LBk|)
 0  [            -Inf,+0.000000000e+00]         NaN      1.6e-18   510 1.7e-01
 1  [-1.800000000e+01,-9.000000000e+00]     5.0e-01      1.0e-10  3690 1.1e+00
 2  [-9.000013022e+00,-9.000006511e+00]     7.2e-07
 execution time = 1.08,  break_yes = 1.
```

As a result, we obtain $\eta^p \in [\mathrm{lb}^2, \mathrm{ub}^2] = [\mathrm{LB2}, \mathrm{UB2}] = [-9.000013022\mathrm{e}{+00}, -9.000006511\mathrm{e}{+00}]$. We also obtain an approximate solution of (1):

$$\begin{aligned} \boldsymbol{X} &= \mathsf{reshape(solNB.X,K1.s,K1.s)} = \begin{pmatrix} 1.0000 & 0 & 1.0011 & 0 \\ 0 & 0 & 0 & 0 \\ 1.0011 & 0 & 1.0011 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\ &\approx \begin{pmatrix} 1.0000 & 0 & 1.0011 & 0 \end{pmatrix}^T \begin{pmatrix} 1.0000 & 0 & 1.0011 & 0 \end{pmatrix}, \end{aligned}$$

which implies that $\boldsymbol{x} = \begin{pmatrix} 1.0000 & 0 & 1.0011 & 0 \end{pmatrix}$ is an approximate optimal solution to QOP (4).

3 General description of $\mathbb{V}$ and $\mathbb{K}_i$ $(i = 1, 2)$

Here we use the following notation:

$$\begin{aligned} \mathbb{V} &= \prod_{p=1}^{m} \mathbb{S}^{n^p}, \ \mathbb{K}_1 = \prod_{p=1}^{m} \mathbb{S}^{n^p}_+, \\ \langle \boldsymbol{X}, \boldsymbol{Y} \rangle &= \sum_{p=1}^{m} \langle \boldsymbol{X}^p, \boldsymbol{Y}^p \rangle \text{ for every } \boldsymbol{X} = (\boldsymbol{X}^1, \ldots, \boldsymbol{X}^m), \ \boldsymbol{Y} = (\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^m) \in \mathbb{V}, \\ \mathbb{K}_2 &= \text{a nonempty polyhedral cone in } \mathbb{V} \text{ satisfying Condition (IV)} \\ &\quad \text{(A detailed description is presented in Section 3.2),} \\ \boldsymbol{I} &= \prod_{p=1}^{m} \boldsymbol{I}^p, \text{ where } \boldsymbol{I}^p \in \mathbb{S}^{n^p} \text{ denotes the identity matrix,} \\ \boldsymbol{Q} &\in \mathbb{V}, \ \boldsymbol{H} \in \mathbb{K}_1^* + \mathbb{K}_2^*. \end{aligned}$$

We assume Conditions (I) through (V). To describe the input and output of newtBracket, we use the notation:

$$\mathrm{vec}(\boldsymbol{Z}) = (\mathrm{vec}(\boldsymbol{Z}^1), \ldots, \mathrm{vec}(\boldsymbol{Z}^1)) \in \mathbb{R}^t \text{ for every } \boldsymbol{Z} = (\boldsymbol{Z}^1, \ldots, \boldsymbol{Z}^m) \in \prod_{p=1}^{m} \mathbb{S}^{n^p},$$

where $t = \sum_{p=1}^{m} n^p \times n^p$.

As the input for newtBracket, we need to specify

$$\mathsf{QVect},\ \mathsf{HVect},\ \mathsf{K1}, \mathsf{projMap}, \mathsf{LB0}, \mathsf{UB0},\ \mathsf{rho},\ \mathsf{paramsNB}.$$

3.1 QVect, HVect and K1

The first three inputs QVect, HVect and K1 correspond to $\boldsymbol{Q} \in \mathbb{V}$, $\boldsymbol{H} \in \mathbb{V}$ and $\mathbb{K}_1$, respectively. The newtBracket package employs the Sedumi format [13] to represent them. Assume that

$$\boldsymbol{Q} \ = \ (\boldsymbol{Q}^1, \ldots, \boldsymbol{Q}^m) \in \mathbb{V} = \prod_{p=1}^{m} \mathbb{S}^{n^p}, \ \boldsymbol{H} \ = \ (\boldsymbol{H}^1, \ldots, \boldsymbol{H}^m) \in \mathbb{K}_1^* + \mathbb{K}_2^*. \tag{6}$$

Then

$$\begin{array}{rcl} \mathsf{K1.s} & = & [n^1, n^2, \ldots, n^m], \\ \mathsf{QVect} & = & \mathrm{vec}(\boldsymbol{Q}) \ = \ [\mathrm{vec}(\boldsymbol{Q}^1), \ldots, \mathrm{vec}(\boldsymbol{Q}^m)] \in \mathbb{R}^t, \\ \mathsf{HVect} & = & \mathrm{vec}(\boldsymbol{H}) \ = \ [\mathrm{vec}(\boldsymbol{H}^1), \ldots, \mathrm{vec}(\boldsymbol{H}^m)] \in \mathbb{R}^t, \end{array}$$

where $t = \sum_{p=1}^{m} n^p \times n^p$. We note that every $\boldsymbol{X} = (\boldsymbol{X}^1, \ldots, \boldsymbol{X}^m) \in \prod_{p=1}^{m} \mathbb{S}^{n^p}$ is also represented as a t dimensional row vector $\mathsf{XVect} = [\mathrm{vec}(\boldsymbol{X}^1), \ldots, \mathrm{vec}(\boldsymbol{X}^m)]$.

For example, if

$$\left.\begin{array}{rcl} \boldsymbol{Q} & = & (\boldsymbol{Q}^1, \boldsymbol{Q}^2), \ \boldsymbol{Q}^1 \ = \ \begin{pmatrix} 0 & 3 & -4 \\ 3 & 1 & 2 \\ -4 & 2 & -1 \end{pmatrix}, \ \boldsymbol{Q}^2 \ = \ \begin{pmatrix} 0 & 2 \\ 2 & 4 \end{pmatrix}, \\ \boldsymbol{H} & = & (\boldsymbol{H}^1, \boldsymbol{H}^2), \ \boldsymbol{H}^1 \ = \ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ \boldsymbol{H}^2 \ = \ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \end{array}\right\} \tag{7}$$

then $t = 3 \times 3 + 2 \times 2 = 13$,

$$\begin{array}{rcl} \mathsf{K1.s} & = & [3,\ 2], \\ \mathsf{QVect} & = & [\mathrm{vec}(\boldsymbol{Q}^1), \mathrm{vec}(\boldsymbol{Q}^1)] \\ & = & [0, 3, -4,\ 3, 1, 2,\ -4, 2, -1,\ \ 0, 2,\ 2, 4] \in \mathbb{R}^{13}, \\ \mathsf{HVect} & = & [\mathrm{vec}(\boldsymbol{H}^1), \mathrm{vec}(\boldsymbol{H}^2)] \ = \ [1, \mathbf{0}] \in \mathbb{R}^{13}. \end{array}$$

3.2 projMap

The metric projections from the space $\mathbb{V}$ onto the cones $\mathbb{K}_1$, $\mathbb{K}_1^*$, $\mathbb{K}_2$ and $\mathbb{K}_2^*$ are carried out through the functions projMap.K1, projMap.K1Star, projMap.K2 and projMap.K2Star, respectively. They are defined as

```
% Projection onto K1
projMap.K1     = @(x) projOntoK1(x, K1);
projMap.K1Star = @(x) projOntoK1(x, K1);
% Projection onto K2
projMap.K2     = @(x) projOntoK2(x,K2);
projMap.K2Star = @(x) XVect + projOntoK2(-x,K2);
```

Here projOntoK1 and projOntoK2 are MATLAB functions stored in the directory projection, K1.s $= [n_1, \ldots, n_m]$, and K2 is described below.

The cone $\mathbb{K}_2$ in the following specific form can be handled with in the newtBracket package.

$$\left.\begin{array}{rcl} \mathbb{K}_2 & = & \left\{ \begin{array}{l} \boldsymbol{X} = (\boldsymbol{X}^1, \ldots, \boldsymbol{X}^m) \in \prod_{p=1}^m \mathbb{S}^{n^p} : \\ \end{array} \begin{array}{l} X_{ij}^p \geq 0 \text{ for all } p, i, j, \\ X_{ij}^p = 0 \text{ for } (p, i, j) \in \mathcal{E}_0, \\ \boldsymbol{X} \in \mathbb{L} \end{array} \right\}, \\ \mathbb{L} & = & \left\{ \boldsymbol{X} = (\boldsymbol{X}^1, \ldots, \boldsymbol{X}^m) \in \prod_{p=1}^m \mathbb{S}^{n^p} : \begin{array}{l} X_{ij}^p - X_{k\ell}^q = 0 \\ \text{for } (p, i, j, q, k, \ell) \in \mathcal{E}_{\rm eq} \end{array} \right\}. \end{array} \right\} \tag{8}$$

Recall that each $\boldsymbol{X} = (\boldsymbol{X}^1, \ldots, \boldsymbol{X}^m) \in \prod_{p=1}^m \mathbb{S}^{n^p}$ is represented as a $t = \sum_{p=1}^m n^p \times n^p$ dimensional vector $\mathsf{x} = (\mathrm{vec}\boldsymbol{X}^1, \ldots, \mathrm{vec}\boldsymbol{X}^m)$. Since

$$L \;=\; \left\{\mathsf{x} = (\mathrm{vec}\boldsymbol{X}^1, \ldots, \mathrm{vec}\boldsymbol{X}^m) : \boldsymbol{X} = (\boldsymbol{X}^1, \ldots, \boldsymbol{X}^m) \in \mathbb{L}\right\}$$

forms a linear subspace of $\mathbb{R}^t$, we can take a $t \times r$ matrix $\boldsymbol{A}$ with **column full rank** such that $L = \{\mathsf{x} \in \mathbb{R}^t : \mathsf{x}\boldsymbol{A} = \boldsymbol{0}\}$ where $r = (t-$the dimension of $L)$. Hence, in the x space, the cone $\mathbb{K}_2$ is represented as $L \cap \{\mathsf{w} \in \mathbb{R}^t : \mathsf{w}(k) = 0 \text{ for every } k \in E_0\} \cap \mathbb{R}^t_+$ for some $E_0 \subset \{1, \ldots, t\}$. Then the metric projection of $\mathsf{x} \in \mathbb{R}^t \to \mathsf{z} \in$ the cone is decomposed as follows

$$\begin{array}{rcl} \mathsf{x} & \to & \mathsf{v} = \mathsf{x}\big(\boldsymbol{I} - \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\big) \in L \\ & \to & \mathsf{w} \text{ such that } \mathsf{w}(k) = 0 \text{ if } k \in E_0 \text{ and } \mathsf{w}(k) = \mathsf{v}(k) \text{ otherwise} \\ & \to & \mathsf{z} = \mathsf{max}(\mathsf{w}, 0) \in L \cap \{\mathsf{v} \in \mathbb{R}^t : \mathsf{w}(k) = 0 \text{ for every } k \in E_0\} \cap \mathbb{R}^t_+. \end{array}$$

(This decomposition depends very much on the special structure of the cone $\mathbb{K}_2$ (8). See [8] for more details.) Here $\boldsymbol{I}$ denotes the $t \times t$ identity matrix. We set

$$\mathsf{K2.A} \;=\; \boldsymbol{A},\; \mathsf{K2.zeroIndexSet} \;=\; E_0,\; \mathsf{K2.projMat} \;=\; \boldsymbol{I} - \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T.$$

Assume that $\boldsymbol{Q}$ and $\boldsymbol{H}$ are given as in (6) and that $\mathbb{K}_2$ is given as in (8). Then their conversion into QVect, HVect and K2 together with K1 is carried out by the MATLAB function [K1,K2,QVect,HVect] = genK2COPeq(Qcell,Hcell,E0,Eeq), where

$$\mathsf{Qcell}\{p\} = \boldsymbol{Q}^p,\; \mathsf{Hcell}\{p\} = \boldsymbol{H}^p \;(p = 1, \ldots, m),\; \mathsf{E0} \leftarrow \mathcal{E}_0,\; \mathsf{Eeq} \leftarrow \mathcal{E}_{\rm eq}.$$

Example 3.1. Suppose that $\boldsymbol{Q}$ and $\boldsymbol{H}$ are given as in (7), and that

$$\begin{array}{rcl} \mathcal{E}_0 & = & \{(1,2,3)\}, \\ \mathcal{E}_{\rm eq} & = & \{(1,3,3,2,1,1), (1,2,2,1,1,2), (1,3,3,1,1,3), (2,2,2,2,1,2)\}. \end{array}$$

The script file is provided to generate K1,K2,QVect and HVect from $\boldsymbol{Q}$, $\boldsymbol{H}$, $\mathcal{E}_0$ and $\mathcal{E}_{\rm eq}$ by genK2COPeq.

```
% Example31
noOfCells = 2;
Qcell = cell(1,noOfCells);
```

```
Hcell = cell(1,noOfCells);
n1 = 3; n2 = 2;
% K1.s = [n1,n2];

Qcell{1} = [ 0, 3, -4; ...
             3, 1,  2; ...
            -4, 2, -1];
Qcell{2} = [ 0,  2; ...
             2,  4];

Hcell{1} = sparse(1,1,1,3,3,1);
Hcell{2} = sparse(2,2);

E0  = [1,2,3]; % <==> X^1_{23} = 0

Eeq = [1,3,3,2,1,1; ...
       1,2,2,1,1,2; ...
       1,3,3,1,1,3; ...
       2,2,2,2,1,2];
% <==> X^1_{33} = X^2_{11};
%      X^1{2,2} = X^1_{12};
%      X^1{3,3} = X^1_{13};
%      X^2{2,2} = X^2_{12};

% Generating K1, K2, QVect,HVect from Qcell,Hcell,E0,Eeq.
[K1,K2,QVect,HVect] = genK2COPeq(Qcell,Hcell,E0,Eeq);

% Projection onto the positive semidefinite cone.
projMap.K1     = @(x) projOntoK1(x, K1);
projMap.K1Star = @(x) projOntoK1(x, K1);

% Projection onto the cone K2 and its dual K2*.
projMap.K2     = @(x) projOntoK2(x,K2);
projMap.K2Star = @(x) x + projOntoK2(-x,K2);

LB0 = -Inf;
UB0 = 0;
rho = sum(K1.s .* K1.s);
paramNB = [];
[solNB, infoNB] ...
    = newtBracket(QVect,HVect,K1,projMap,LB0,UB0,rho,paramNB);

[XMatcell] = vecToMatcell(solNB.X,K1);

fprintf('\nObjVal = %10.9e\n',solNB.LBv);
for p=1:length(XMatcell)
```

```
    fprintf('XMatcell{%d} = \n',p);
    disp(XMatcell{p});
end
```

As a result, we obtain the following result.

```
 k  [LBk,              UBk              ]     (UBk-LBk)     errmax  APGR  eTime
                                              ---------             iter
                                          max(1,|UBk|,|LBk|)
 0  [      -Inf,+0.000000000e+00]     NaN      3.3e-17   510 1.5e-01
 1  [-4.679999998e+01,-9.000000000e+00]     8.1e-01      6.8e-11  4200 1.3e+00
 2  [-9.000030349e+00,-9.000005837e+00]     2.7e-06
 execution time = 1.27,  break_yes = 1

ObjVal = -9.000030349e+00
XMatcell{1} =
    1.0000         0    1.0011
         0         0         0
    1.0011         0    1.0011

XMatcell{2} =
    1.0011         0
         0         0
```

Alternatively, we can represent the linear subspace $L \cap \{\mathsf{w} \in \mathbb{R}^t : w(k) = 0 \text{ for every } k \in E_0\}$ as $\{\boldsymbol{yB} \in \mathbb{R}^t : \boldsymbol{y} \in \mathbb{R}^u\}$ for some $u \times t$ matrix $\boldsymbol{B}$ with full-row rank. Then the metric projection of $\mathsf{x} \in \mathbb{R}^t \to \mathsf{z} \in$ the cone is decomposed as follows

$$\begin{aligned} \mathsf{x} &\to \mathsf{w} = \mathsf{x}\boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{B}^T)^{-1}\boldsymbol{B} \in \{\boldsymbol{yB} : \boldsymbol{y} \in \mathbb{R}^u\} \\ &\to \mathsf{z} = \mathsf{max}(\mathsf{w}, 0) \in \{\boldsymbol{yB} : \boldsymbol{y} \in \mathbb{R}^u\} \cap \mathbb{R}^t_+. \end{aligned}$$

In this case, we set

$$\mathsf{K2.B} \;=\; \boldsymbol{B} \;\text{ and }\; \mathsf{K2.projMat} \;=\; \boldsymbol{B}^T(\boldsymbol{B}\boldsymbol{B}^T)^{-1}\boldsymbol{B}.$$

This method is employed in genK2BCQOPbasis for a sparse DNN relaxation of a QOP with complementarity constraints in binary variables which will be discussed in Section 4.2.

3.3 LB0, UB0 and rho

The input arguments LB0, UB0 and rho correspond to lb^0, ub^0 and ρ, respectively. We need to choose lb^0 and ub^0 such that $-\infty \le \mathrm{lb}^0 \le \eta^d < \mathrm{ub}^0 < \infty$. See also Condition (V) for choosing $\rho > 0$ in newtBracket.

3.4 Output of newtBracket

solNB.LBv = a (valid) lower bound for the optimal value η^d of COP (2).

solNB.UB = an upper bound for the optimal value η^d of COP (2).

solNB.Y1 = $\mathrm{vec}(\boldsymbol{Y}_1)$, where $(\mathsf{solNB.LBv}, \boldsymbol{Y}_1, \boldsymbol{Y}_2)$ is an approximate optimal solution to COP (2).

solNB.Y2 = $\mathrm{vec}(\boldsymbol{Y}_2)$, where $(\mathsf{solNB.LBv}, \boldsymbol{Y}_1, \boldsymbol{Y}_2)$ is an approximate optimal solution to COP (2).

solNB.X = $\mathrm{vec}(\boldsymbol{X})$, where $\boldsymbol{X}$ is an approximate optimal solution to COP (1).

infoNB.iter = the number of Newton (secant) iterations.

infoNB.timeNB = the execution time for newtBracket.

infoNB.APGiter = the total number of iterations of APRR (the accelerated proximal gradient method).

infoNB.break_yes = the terminal code: 1 if the absolute error $|\mathsf{solNB.UB} - \mathsf{solNB.LB}|$ is less than paramNB.delta; 2 if the relative error

$$|\mathsf{solNB.UB} - \mathsf{solNB.LB}| / \max\{1.0\text{e-}15, |\mathsf{solNB}|.|\mathsf{UB}, \mathsf{solNB.LB}|\}$$

is less than paramNB.delta1; 4 if paramNB.intValSW = 1 and ceil(solNB.UB) = ceil(solNB.LB); -1 if newtBracket failed.

3.5 Parameters

paramsNB.problemId = the instance name or the identification number displayed on the screen while newtBracket is running.

paramsNB.intValSW = 1 if the optimal value is integer (in this case newtBracket stops when $\lceil \mathrm{ub}^k \rceil = \lceil \mathrm{lb}^k \rceil$), 0 otherwise.

paramsNB.delta = the absolute terminal accuracy; newtBracket stops when $\mathrm{ub}^k - \mathrm{lb}^k <$ paramsNB.delta; the default value is 0.01.

paramsNB.delta1 = the relative terminal accuracy; newtBracket stops when $|\mathsf{solNB.UB} - \mathsf{solNB.LB}| / \max\{1.0\text{e-}15, |\mathsf{solNB}|.|\mathsf{UB}, \mathsf{solNB.LB}|\} <$ paramsNB.delta1; the default value is 1.0e-6.

paramsNB.maxiterAPGR = the maximum number of iterations of APGR (the accelerated proximal gradient method); the default value is 6000.

paramNB.bestObjVal : if an integer approximate optimal value (or a target objective value) $\hat{\eta}^d$ of COP (2) is known, set paramNB.bestObjVal = $\hat{\eta}^d$; then newtBracket stops when $\lceil \mathrm{lb}^k \rceil$ = paramNB.bestObjVal holds.

4 DNN relaxation of a class of QOPs with linear and complementarity constraints in binary variables

Let $\boldsymbol{Q} \in \mathbb{S}^n$, $\boldsymbol{A} \in \mathbb{R}^{n \times r}$, $I_{01} \subset \{2, \ldots, n\}$ and $I_{\mathrm{comp}} \subset \{(i,j) : 2 \le i < j \le n\}$. We consider the QOP

$$\zeta \;=\; \min \left\{ \boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \begin{array}{l} \boldsymbol{x} \in \mathbb{R}^n_+,\ x_1 = 1,\ \boldsymbol{x}\boldsymbol{A} = 0, \\ x_i \in \{0,1\}\ (i \in I_{01}),\ x_i x_j = 0\ ((i,j) \in I_{\mathrm{comp}}) \end{array} \right\}. \tag{9}$$

(Note that x_1 is fixed to 1 and $1 \notin I_{01}$). We assume the condition

(VI) The feasible region of QOP (9) is nonempty and

$$F_0 \equiv \left\{ \boldsymbol{x} \in \mathbb{R}^n_+ : \ x_1 = 1, \ \boldsymbol{x}\boldsymbol{A} = \boldsymbol{0}, \ x_i \in \{0,1\} \ (i \in I_{01}) \ \right\} \subset [0,1]^n.$$

This condition guarantees that QOP (9) has an optimal solution.

To derive a DNN relaxation of the form (1), we first replace the linear equality constraint $\boldsymbol{x}\boldsymbol{A} = \boldsymbol{0}$ by a single quadratic equality constraint $\boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x}^T = 0$, and then apply the Lagrangian relaxation to the resulting problem.

$$\zeta_\lambda \ = \ \left\{ \boldsymbol{x}\boldsymbol{Q}_\lambda\boldsymbol{x}^T : \begin{array}{l} \boldsymbol{x} \in \mathbb{R}^n_+, \ x_1 = 1, \ x_i \in \{0,1\} \ (i \in I_{01}), \\ x_i x_j = 0 \ ((i,j) \in I_{\rm comp}) \end{array} \right\}, \tag{10}$$

where $\boldsymbol{x}\boldsymbol{Q}_\lambda\boldsymbol{x}^T = \boldsymbol{x}(\boldsymbol{Q} + \lambda\boldsymbol{A}\boldsymbol{A}^T)\boldsymbol{x}^T = \boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T + \lambda\boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x}^T$, and $\lambda \in \mathbb{R}$ is a Lagrangian multiplier for the equality constraint $\boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x} = 0$. Since $\boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x} \geq 0$ for every $\boldsymbol{x} \in \mathbb{R}^n$, the term $\lambda\boldsymbol{x}\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x}^T$ with $\lambda > 0$ added to the objective function $\boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T$ of the original QOP (9) serves as a penalty for violating the equality constraint $\boldsymbol{x}\boldsymbol{A} = \boldsymbol{0}$. Under Condition (VI), we can prove that

- QOP (10) has an optimal solution $\boldsymbol{x}^\lambda$ if $\lambda > 0$ is sufficiently large,
- the optimal value $\boldsymbol{x}^\lambda\boldsymbol{Q}(\boldsymbol{x}^\lambda)^T$ of QOP (10) converges to the optimal value ζ of QOP (9) as $\lambda \to \infty$,
- any accumulation point of $\boldsymbol{x}^\lambda$ ($\lambda > 0$) is an optimal solution of QOP (9).

QAP (10) is a generalization of QOP (4); replace 4 by n and $\boldsymbol{Q}$ by $\boldsymbol{Q}_\lambda$ in (4) to obtain (10). We present the dense DNN relaxation and the sparse DNN relaxation of QOP (10) in Sections 4.1 and 4.2, respectively. In the newtBracket package, we represent I_{01} and $I_{\rm comp}$ of QOP (10) as

$$\left.\begin{array}{lcl} \mathsf{I01} & = & \mathsf{sparse}(1, I_{01}, 1, 1, n, n), \\ \mathsf{Icomp} & = & \mathsf{sparse}([1,1,2,2,\ldots,\mathsf{m},\mathsf{m}], [\mathsf{i}_1,\mathsf{j}_1,\mathsf{i}_2,\mathsf{j}_2,\ldots,\mathsf{i}_\mathsf{m},\mathsf{j}_\mathsf{m}], 1, \mathsf{m}, \mathsf{n}, 2*\mathsf{m}) \end{array}\right\} \tag{11}$$

for both dense and sparse relaxation, where $I_{\rm comp} = \{(i_1,j_1),(i_2,j_2),\ldots,(i_m,j_m)\}$.

4.1 Dense DNN relaxation

In the dense DNN relaxation, we simply set

$\mathsf{QVect} = \mathsf{reshape}(\boldsymbol{Q}_\lambda, 1, \mathsf{K1.s} * \mathsf{K1.s})$; $\mathsf{HVect} = \mathsf{sparse}(1,1,1,1,\mathsf{n},1)$;
$\mathsf{K2} = \mathsf{genK2BCQOPeq}(\mathsf{K1}, \mathsf{I01}, \mathsf{Icomp})$;
$\mathsf{projMap.K1} = @(\mathsf{x})\mathsf{projOntoK1}(\mathsf{x}, \mathsf{K1})$; $\mathsf{projMap.K1Star} = @(\mathsf{x})\mathsf{projOntoK1}(\mathsf{x}, \mathsf{K1})$;
$\mathsf{projMap.K2} = @(\mathsf{x})\mathsf{projOntoK2}(\mathsf{x}, \mathsf{K2})$; $\mathsf{projMap.K2Star} = @(\mathsf{x})\mathsf{XVect} + \mathsf{projOntoK2}(-\mathsf{x}, \mathsf{K2})$;
$\mathsf{LB0} = -\mathsf{Inf}$; $\mathsf{UB0} \geq \eta^p + 0.2|\eta^p|$;
rho = some positive number such that $\sum_{i=1}^2 x_i^2 \leq \mathsf{rho}$ for every feasible solution of QOP (9);
(we can take $\mathsf{rho} = n$ by Condition (VI), but a smaller rho is better).

Then we can issue [solNB, infoNB] = newtBracket(QVect,HVect,K1,projMap,LB0,UB0,rho).

4.2 Sparse DNN relaxation

When QOP (9) satisfies a structured sparsity characterized by a chordal graph, we can apply a sparse DNN relaxation described in this section. The sparse DNN relaxation aims to efficiently compute a lower bound of ζ_λ of a large scale QOP (9) whose sparsity is characterized by a sparse chordal graph as described below.

We first need to construct the $n \times n$ sparsity pattern matrix $\boldsymbol{R}$ and the sparsity pattern graph $G(N, E)$ by taking account of not only the sparsity of $\boldsymbol{Q}$ but also the sparsity of $\boldsymbol{A}$ and $I_{\rm comp}$. Define the $n \times n$ symmetric matrix $\boldsymbol{R} = [R_{ij}]$ by

$$R_{ij} \;=\; \begin{cases} 1 & \text{if } i = j,\ Q_{ij} \neq 0,\ [\boldsymbol{A}\boldsymbol{A}^T]_{ij} \neq 0 \text{ or } (i,j) \in I_{\rm comp}, \\ 0 & \text{otherwise,} \end{cases}$$

and a chordal graph $G(N, E)$ such that

$$N = \{1, \ldots, n\} \text{ and } E \supset \{(i,j) \in N \times N : i \neq j,\ R_{ij} \neq 0\}.$$

Here an undirected graph is called as chordal if every cycle at least 4 edges has a chord (see [2] for fundamental properties of a chordal graph). Let $C_1, \ldots, C_m$ be the maximal cliques of $G(N, E)$. In the newtBracket MATLAB package, each clique C_p is represented as an n-dimensional 0-1 row vector whose ith element takes 1 if $i \in C_p$ and 0 otherwise, and the set of all cliques are represented as an $m \times n$ 0-1 matrix whose pth row corresponds to the clique C_p $(p = 1, \ldots, m)$. The cliques can be obtained by issuing the command

$$[\mathsf{cliques}] = \mathsf{cliquesFromSpMat}(\boldsymbol{R}).$$

For example, suppose that $n = 6$ and

$$\boldsymbol{R} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \in \mathbb{S}^6.$$

Then the maximal cliques are $C_1 = \{1, 2, 3\},\ \{1, 3, 4, 5\}$ and $\{1, 5, 6\}$. In this case, their representation in the newtBracket MATLAB package is

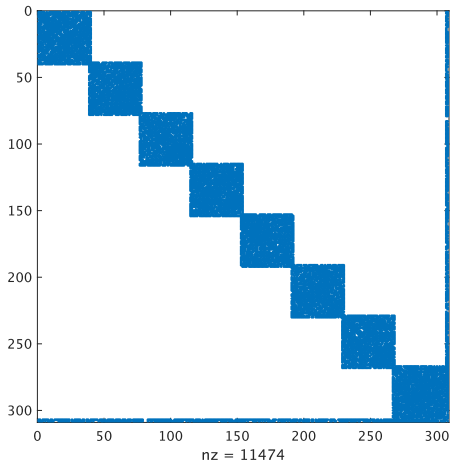
$$\mathsf{cliques} = [1, 1, 1, 0, 0, 0; 1, 0, 1, 1, 1, 0; 1, 0, 0, 0, 1, 1];$$

which is obtained by $[\mathsf{cliques}] = \mathsf{cliquesFromSpMat}(\boldsymbol{R})$.

If we generate I01 and Icomp as in (11), and cliques as above, we set

$$\begin{array}{l} \mathsf{paramNB.sparseSW} = 1;\ \%\ \text{for the sparse DNN relaxation;} \\ [\mathsf{K1}, \mathsf{K2}, \mathsf{QVect}, \mathsf{HVect}] = \mathsf{genK2BCQOPbasis}(\mathsf{paramNB}, \boldsymbol{Q}_\lambda, \mathsf{I01}, \mathsf{Icomp}, \mathsf{cliques}); \\ \mathsf{projMap.K1} = @(\mathsf{x})\mathsf{projOntoK1}(\mathsf{x}, \mathsf{K1});\ \mathsf{projMap.K1Star} = @(\mathsf{x})\mathsf{projOntoK1}(\mathsf{x}, \mathsf{K1}); \\ \mathsf{projMap.K2} = @(\mathsf{x})\mathsf{projOntoK2}(\mathsf{x}, \mathsf{K2});\ \mathsf{projMap.K2Star} = @(\mathsf{x})\mathsf{XVect} + \mathsf{projOntoK2}(-\mathsf{x}, \mathsf{K2}); \\ \mathsf{LB0} = -\mathsf{Inf};\ \mathsf{UB0} \geq \eta^p + 0.2|\eta^p|; \\ \mathsf{rho} = \mathsf{sum}(\mathsf{K1.s});\ \%\ \text{which could be taken smaller by taking account of} \\ \qquad\qquad \%\ \text{the constraint } \boldsymbol{x}\boldsymbol{A} = \boldsymbol{0} \text{ and } I_{\rm comp}. \end{array} \tag{12}$$

Figure 1: Arrow type sparsity pattern matrix $\boldsymbol{R}$. $n = 308$ and the cliques are $\{38r+1, 38r+2, \ldots, 38r+40, 307, 308\}$ $(r = 0, 1, \ldots, 7)$.



Then, we can run newtBracket with the command:
[solNB, infoNB] = newtBracket(QVect,HVect,K1,projMap,LB0,UB0,rho).

Example 4.1. Consider the simple sparse binary QOP:

$$\zeta \;=\; \left\{\boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \boldsymbol{x} \in \mathbb{R}^n_+,\; x_i \in \{1, \ldots, n\}\right\}.$$

Here $\boldsymbol{Q}$ denotes a 195×195 randomly generated symmetric matrix with arrow type sparsity, which is a well-known chordal sparsity, given in Figure 1. We present a MATLAB script below to solve the DNN relaxation of the problem.

```
function Example41(sparseSW)
% Example41.m
%
% Sparse binary QOP
% minimize   x QMat x^T
% subject to x \in {0,1}^n

if nargin == 0
    sparseSW = 1;
end

load('Example41QMat.mat','QMat');
figure(1);
spy(QMat);
n = size(QMat,1);

Icomp = [];
I01 = ones(1,n);
```

```
paramNB.sparseSW = sparseSW;
if paramNB.sparseSW == 0
    cliques = ones(1,n);
    [K1,K2,QVect,HVect] = ...
        genK2BCQOPbasis(paramNB,QMat,I01,Icomp);
else % paramNB.sparseSW == 1
    mergeSW = 1;
    [cliques] = cliquesFromSpMat(QMat,mergeSW);
    %     figure(2);
    %     spy(cliques);
    [K1,K2,QVect,HVect] = ...
        genK2BCQOPbasis(paramNB,QMat,I01,Icomp,cliques);
end

projMap.K1     = @(x) projOntoK1(x, K1);
projMap.K1Star = @(x) projOntoK1(x, K1);

projMap.K2     = @(x) projOntoK2(x,K2);
projMap.K2Star = @(x) x + projOntoK2(-x,K2);

UB0 = 0;
LB0 = -Inf;
rho = sum(K1.s);
paramNB.intValSW = 1;

[solNB, infoNB] ...
    = newtBracket(QVect,HVect,K1,projMap,LB0,UB0,rho,paramNB);
```

To execute the sparse DNN relaxation, run function Example41(sparseSW) with sparseSW = 1 as

```
>> Example41(1);
```

Then we have

```
 k  [LBk,              UBk             ]     (UBk-LBk)     errmax  APGR  eTime
                                           ---------              iter
                                       max(1,|UBk|,|LBk|)
 0  [            -Inf,+0.000000000e+00]         NaN      3.2e-09   510 1.8e+00
 1  [-7.143327765e+03,-4.964845649e+02]     9.3e-01      5.0e-07   540 3.4e+00
 2  [-3.205006527e+03,-7.362640665e+02]     7.7e-01      6.0e-07   640 5.6e+00
 3  [-2.012341794e+03,-9.175708738e+02]     5.4e-01      7.1e-07   830 8.1e+00
 4  [-1.220013371e+03,-9.706617616e+02]     2.0e-01      7.4e-08  1310 1.2e+01
 5  [-1.003608980e+03,-9.770966607e+02]     2.6e-02      2.3e-08   925 1.4e+01
 6  [-9.779994829e+02,-9.772776891e+02]     7.4e-04
 execution time = 14.50,  break_yes = 4
```

To execute the dense DNN relaxation, run function Example41(sparseSW) with sparseSW = 0 as

```
>> Example41(0);
```

Then we have

```
 k  [LBk,              UBk              ]     (UBk-LBk)     errmax  APGR  eTime
                                            ---------             iter
                                        max(1,|UBk|,|LBk|)
 0  [            -Inf,+0.000000000e+00]         NaN      6.7e-10   510 5.0e+00
 1  [-1.538738657e+04,-1.862497134e+02]     9.9e-01      1.3e-08   530 9.6e+00
 2  [-8.176507315e+03,-3.475907701e+02]     9.6e-01      8.6e-08   560 1.4e+01
 3  [-5.374731373e+03,-5.893468119e+02]     8.9e-01      8.5e-07   680 2.0e+01
 4  [-2.925085917e+03,-7.850899314e+02]     7.3e-01      7.2e-07   810 2.7e+01
 5  [-1.746903628e+03,-9.169291055e+02]     4.8e-01      3.0e-07  1310 3.9e+01
 6  [-1.181278946e+03,-9.662825792e+02]     1.8e-01      5.7e-08  2000 5.8e+01
 7  [-1.000351723e+03,-9.735600549e+02]     2.7e-02      2.9e-09  3250 9.1e+01
 8  [-9.748682099e+02,-9.738505002e+02]     1.0e-03      3.0e-08   205 9.3e+01
 9  [-9.739831056e+02,-9.738831679e+02]     1.0e-04
 execution time = 93.18,  break_yes = 4
```

We observe that the sparse DNN relaxation is much faster than the dense DNN relaxation but the lower bound -9.779996062e+02 obtained is slightly worse than the lower bound -9.739831056e+02 by the dense DNN relaxation.

In general, exploiting sparsity is regarded as an essential method for solving large scale COPs of the form (1), but the optimal value may be slightly worsened. We also note that the sparse DNN relaxation associated with a chordal graph $G(N, E)$ with the maximal cliques $C_1, \ldots, C_m$ may not be always faster than the dense DNN relaxation. For example, if the dimension $n = |N|$ of the QOP is small, say, $n \leq 50$, then the dense relaxation with cliques=ones(1,n) is often faster. Or the sizes of many cliques are small, say, less than 30, then it is usually better to reconstruct the sparsity pattern graph by merging some of those cliques for the computational efficiency. We can specify the parameter mergeSW to be 1 as in this example to call [cliques] = cliquesFromSpMat(QMat,mergeSW) to create such modified cliques.

5 Applications

We present 4 types QOPs, binary QOPs, quadratic multiple knapsack problems, quadratic assignment problems, and maximum stable set problems as special cases of QOP (9). MATLAB scripts for solving some instances of those QOPs are given in the directory applications.

5.1 Binary QOPs

The problem is described as

$$\zeta \;=\; \min \left\{ \boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \; \boldsymbol{x} \in \mathbb{R}^n_+, \; x_1 = 1, \; x_i \in \{0,1\} \; (i \in \{2, \ldots, n\}) \; \right\}.$$

Since there is no linear equality constraint, the Lagrangian relaxation is unnecessary. We may apply the dense or sparse DNN relaxation presented in Section 4 to the QOP above. To strengthen the relaxation ([6, Section 6.1]), we introduce slack variables $x_{n-1+i} \in \{0,1\}$ for $x_i \in \{0,1\}$ such that $x_i + x_{n-1+i} = 1$ $(i \in \{2,\ldots,n\})$.

$$\zeta \;=\; \min \left\{ \boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \begin{array}{l} \boldsymbol{x} \in \mathbb{R}^n_+,\ x_1 = 1, x_i + x_{n-1+i} = 1\ (i \in \{2,\ldots,n\}), \\ x_i,\ x_{n-1+i} \in \{0,1\}\ (i \in \{2,\ldots,n\}) \end{array} \right\}.$$

The converted QOP is also a special case of QOP (9) to which the dense or sparse DNN relaxation can be applied. A MATLAB script expBqopNB.m is given in the directory applications/Bqop for solving binary QOP instances from BiqMac library [14]. In those instances, the elements of $\boldsymbol{Q}$ are integers. Hence their optimal values are integers, and we set paramNB.intValSW = 1.

5.2 Quadratic multiple knapsack problems

The problem [3, 11] is described as

$$\zeta \;=\; \min \left\{ \boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \begin{array}{l} \boldsymbol{u} \in \mathbb{R}^\ell_+,\ u_1 = 1,\ (u_2,\ldots,u_\ell)\boldsymbol{C} \le \boldsymbol{d}, \\ u_i \in \{0,1\}\ (i \in \{2,\ldots,\ell\}) \end{array} \right\}.$$

Here $\boldsymbol{C}$ denote an $(\ell-1)\times r$ nonnegative matrix and $\boldsymbol{0} < \boldsymbol{d} \in \mathbb{R}^r$. Defining $n = 2r + \ell - 1$,

$$I_{01} = \{2,\ldots,2\ell-1\},\ I_{\rm comp} = \emptyset,\ \boldsymbol{A} = \begin{pmatrix} -\boldsymbol{d} & -\boldsymbol{e} \\ \boldsymbol{C} & \boldsymbol{I}_{\ell-1} \\ \boldsymbol{O} & \boldsymbol{I}_{\ell-1} \\ \boldsymbol{I}_r & \boldsymbol{O} \end{pmatrix} \in \mathbb{R}^{(2\ell-1+r)\times(\ell-1+r)},$$

where $\boldsymbol{e} \in \mathbb{R}^{\ell-1}$ denotes the row vector of 1's and $\boldsymbol{I}_k$ the identity matrix of size k $(k = \ell-1, r)$, we convert the problem to QOP (9). Thus the dense or sparse Lagrangian DNN relaxation can be applied as presented in Section 4. A MATLAB script expKNqopNB.m is given in the directory applications/KnapsackQop for solving some instances with $\boldsymbol{Q}$ from BiqMac library [14] and randomly generated $\boldsymbol{C}$ and $\boldsymbol{d}$. Their optimal values are integers, and we set paramNB.intValSW = 1.

5.3 Quadratic assignment problems

A quadratic assignment problem (QAP) is formulated as

$$\zeta \;=\; \min \left\{ \boldsymbol{x}\boldsymbol{Q}\boldsymbol{x}^T : \begin{array}{l} \boldsymbol{x} \in \mathbb{R}^{1+n^2}_+,\ x_0 = 1, \\ x_{ij} \in \{0,1\}\ (i = 1,\ldots,n, j = 1,\ldots,n) \\ \sum_{i=1}^n x_{ij} = 1\ (j = 1,\ldots,n), \\ \sum_{j=1}^n x_{ij} = 1\ (i = 1,\ldots,n) \end{array} \right\},$$

where $\boldsymbol{Q}$ denotes a constant matrix in $\mathbb{S}^{1+n^2}$, $\boldsymbol{x} = (x_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{1+n^2}$ a variable vector and x_{ij} the jth element of $\boldsymbol{x}_i \in \mathbb{R}^n$ $(i = 1,\ldots,n, j = 1,\ldots,n)$. We may regard the problem

as a special case of QOP (9), so that we can apply the dense DNN relaxation as presented in Section 4.1. From the condition, we know that $x_0^2+\sum_{i=1}^n\sum_{j=1}^n x_{ij}^2 = 1+n$ for every feasible solution of the problem. As a result, we can take $\rho = 1+n$. A MATLAB script expQapNB.m is given in the directory applications/Qap for solving some instances from QAPLIB [5]. Since their optimal values are known to be integers, we set paramNB.intValSW =1.

5.4 Maximum stable set problems

For an directed graph G with n nodes and edge set $\mathcal{E}$, the stability number $\alpha(G)$ is defined as the cardinality of a maximal stable set of G [4], and

$$-\alpha(G) \;=\; \min\left\{-\boldsymbol{u}\boldsymbol{e}^T : \boldsymbol{u}\in\mathbb{R}^n_+,\ u_i\in\{0,1\}\ (i\in N),\ u_iu_j=0\ ((i,j)\in\mathcal{E})\right\},$$

where $\boldsymbol{e}\in\mathbb{R}^n$ is the row vector of 1's. We could deal with the problem as a special case of QOP (9), so that we could apply the dense or sparse DNN relaxation to it as we discussed in Section 4.

We will present a slightly different and more efficient approach. By setting $\boldsymbol{x} = \boldsymbol{u}\boldsymbol{e}^T/\sqrt{\boldsymbol{u}\boldsymbol{e}^T}$ for $\boldsymbol{0}\neq\boldsymbol{u}\in\{0,1\}^n$, we can derive that

$$\begin{aligned}
-\alpha(G) &= \min\left\{\boldsymbol{x}(-\boldsymbol{e}^T\boldsymbol{e})\boldsymbol{x}^T : \begin{array}{l}\boldsymbol{x}\in\mathbb{R}^n_+,\ \sum_{i=1}^n x_i^2=1,\\ x_ix_j=0\ ((i,j)\in\mathcal{E})\end{array}\right\}\\
&= \min\left\{\langle\boldsymbol{Q},\,\boldsymbol{X}\rangle : \begin{array}{l}\boldsymbol{X}=\boldsymbol{x}^T\boldsymbol{x},\ \boldsymbol{x}\in\mathbb{R}^n_+,\langle\boldsymbol{I},\,\boldsymbol{X}\rangle=1,\\ X_{ij}=0\ ((i,j)\in\mathcal{E})\end{array}\right\}\\
&\geq \min\left\{\langle\boldsymbol{Q},\,\boldsymbol{X}\rangle :\ \boldsymbol{X}\in\mathbb{K}_1\cap\mathbb{K}_2,\ \langle\boldsymbol{I},\,\boldsymbol{X}\rangle=1\ \right\}.
\end{aligned}$$

Here

$$\begin{aligned}
&\boldsymbol{I} = \text{ the } n\times n \text{ identity matrix,}\\
&\boldsymbol{Q} = -\boldsymbol{e}^T\boldsymbol{e},\ \mathbb{K}_1=\mathbb{S}^n_+,\ \mathbb{K}_2=\{\boldsymbol{X}\in\mathbb{N}^n : X_{ij}=0\ ((i,j)\in\mathcal{E}\}.
\end{aligned}$$

Thus we have reduced a DNN relaxation of the max stable set problem to COP of the form (1). The following is a MATLAB script to generate input arguments for newtBracket.

```
K1.s = n;
E0 = [[i_1,j_1];[i_2.j_2]; . . . ;[i_q,j_q]];
% where {\script E} = {(i_1,j_1),(i_2.j_2), . . . ,(i_q,j_q)}
Eeq = [];
[K1,K2,QVect,HVect] = genK2COPeq(Q,I,E0,Eeq,K1);
rho = 1;
LB0 = -Inf;
UB0 = 0;
projMap.K1     = @(x) projOntoK1(x, K1);
projMap.K1Star = @(x) projOntoK1(x, K1);
projMap.K2     = @(x) projOntoK2(x,K2);
projMap.K2Star = @(x) x + projOntoK2(-x,K2);
```

A MATLAB script expMsNB.m is given in the directory applications/MaxStableSet for solving some instances (arising from coding theory) collected by Sloane [12]. Their optimal values are integers, and we set paramNB.intValSW = 1.

References

[1] N. Arima, S. Kim, M. Kojima, and K. C. Toh. Lagrangian-conic relaxations, Part I: A unified framework and its applications to quadratic optimization problems. *Pacific J. Optim.*, 14(1):161–192, 2018.

[2] J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In Liu J.W.H. George A., Gilbert J. R., editor, *Graph Theory and Sparse Matrix Computation*. Springer-Verlag, New York, 1993.

[3] S. Burer. Optimizing a polyhedral-semidefinite relaxation of completely positive programs. *Math. Prog. Comp.*, 2:1–19, 2010.

[4] E. de Klerk and D. V. Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM J. Optim.*, 12:875–892, 2002.

[5] P. Hahn and M. Anjos. QAPLIB – A quadratic assignment problem library. http://www.seas.upenn.edu/qaplib.

[6] N. Ito, S. Kim, M. Kojima, A. Takeda, and K.C. Toh. Equivalences and differences in conic relaxations of combinatorial quadratic optimization problems. *J. Global Optim.*, 72(4):619–653, 2018.

[7] N. Ito, S. Kim, M. Kojima, A. Takeda, and K.C. Toh. BBCPOP: A sparse doubly nonnegative relaxation of polynomial optimization problems with binary, box and complementarity constraints. *ACM Trans. Math. Soft.*, 45((3) 34), 2019.

[8] S. Kim, M. Kojima, and K. C. Toh. Doubly nonnegative relaxations for quadratic and polynomial optimization problems with binary and box constraints. *To appear in Math. Program.*

[9] S. Kim, M. Kojima, and K. C. Toh. A Lagrangian-DNN relaxation: a fast method for computing tight lower bounds for a class of quadratic optimization problems. *Math. Program.*, 156:161–187, 2016.

[10] S. Kim, M. Kojima, and K.C. Toh. A Newton-bracketing method for a simple conic optimization problem. *To appear in Optim. Methods and Softw.*

[11] T. Sarac and A. Sipahioglu. A genetic algorithm for the quadratic multiple knapsack problem. In *Advances in Brain, Vision, and Artificial Intelligence*, volume 4729 of *Lecture Notes in Computer Science*, pages 490–498. Springer, Heidelberg, 2007.

[12] N. Sloane. Challenge problems: Independent sets in graphs.

[13] J. F. Sturm. SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods and Softw.*, 11&12:625–653, 1999.

[14] A. Wiegele. Biq mac library. http://www.biqmac.uni-klu.ac.at/biqmaclib.html, 2007.