User Manual for **BBCPOP**: A Sparse Doubly Nonnegative Relaxation of **P**olynomial **O**ptimization **P**roblems with **B**inary, **B**ox and **C**omplementarity Constraints

Naoki Ito[⋆], Sunyoung Kim[†], Masakazu Kojima[‡], Akiko Takeda[§], Kim-Chuan Toh[¶]

March 2018

Abstract

BBCPOP proposed in [4] is a MATLAB implementation of a hierarchy of sparse doubly nonnegative (DNN) relaxations of a class of polynomial optimization (minimization) problems (POPs) with binary, box and complementarity constraints. Given a POP in the class and a relaxation order (or a hierarchy level), BBCPOP constructs a simple conic optimization problem (COP), which serves as a DNN relaxation of the POP, and then solves the COP by applying the bisection and projection (BP) method [6, 5]. The software package **BBCPOP**, this manual, and a test set of POPs are available at https://sites.google.com/site/bbcpop1/.

Key words. Polynomial optimization problems, Doubly nonnegative relaxation, Bisection and projection method, Large-scale problems, MATLAB software package.
AMS Classification. 90C20, 90C22, 90C25, 90C26.

⋆ Department of Mathematical Informatics, The University of Tokyo, Tokyo 113-8656, Japan. This work was supported by Grant-in-Aid for JSPS Research Fellowship JP17J07365. (`naoki_ito@mist.i.u-tokyo.ac.jp`).

† Department of Mathematics, Ewha W. University, 52 Ewhayeodae-gil, Sudaemoon-gu, Seoul 03760 Korea. The research was supported by NRF 2017-R1A2B2005119. (`skim@ewha.ac.kr`).

‡ Department of Industrial and Systems Engineering, Chuo University, Tokyo 112-8551 Japan. This research was supported by Grant-in-Aid for Scientific Research (A) 26242027. (`kojimamasakazu@mac.com`).

§ Department of Mathematical Analysis and Statistical Inference, The Institute of Statistical Mathematics, 10-3 Midori-cho, Tachikawa, Tokyo 190-8562, Japan The work of this author was supported by Grant-in-Aid for Scientific Research (C), 15K00031. (`atakeda@ism.ac.jp`).

¶ 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`).

1 Introduction

BBCPOP proposed in [4] is a MATLAB software package to compute tight lower bounds for global optimal values of a class of polynomial optimization (minimization) problems (POPs) with binary, box and complementarity constraints. As shown in Figure 1, BBCPOP.m constructs a hierarchy of sparse DNN relaxations of a POP in the class and solves them by applying the bisection and projection method [6, 1, 5] and the accelerated proximal gradient (APG) method [2]. These two methods were described as BP Algorithm and APGR Algorithm (an enhanced version of the APG method) in [4], respectively.

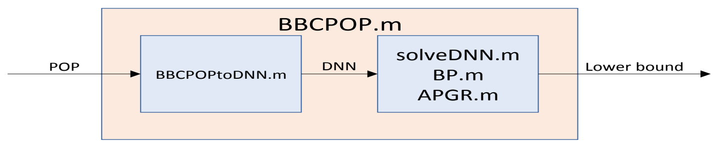

Figure 1: The structure of the main function BBCPOP.m

Let f_0 be a real valued polynomial defined on the n-dimensional Euclidean space $\mathbb{R}^n$, $I_{\rm bin}$ and $I_{\rm box}$ a partition of $N \equiv \{1, 2, \ldots, n\}$, *i.e.*, $I_{\rm bin} \cup I_{\rm box} = N$ and $I_{\rm bin} \cap I_{\rm box} = \emptyset$, and $\mathcal{C}$ a family of subsets of N. Each POP in the class is described as

$$\zeta^* \quad = \quad \min_{\boldsymbol{x}} \left\{ f_0(\boldsymbol{x}) \middle| \begin{array}{l} x_i \in \{0,1\} \ (i \in I_{\rm bin}) \ \text{(box constraint)}, \\ x_j \in [0,1] \ (j \in I_{\rm box}) \ \text{(binary constraint)}, \\ \prod_{j \in C} x_j = 0 \ (C \in \mathcal{C}) \ \text{(complementarity constraint)} \end{array} \right\}. \tag{1}$$

As an illustrative example, we consider the following POP with five variables x_1, x_2, x_3, x_4 and x_5:

$$\left. \begin{array}{ll} \text{minimize} & f_0(\boldsymbol{x}) \equiv 0.5x_1 - 1.8x_3 - 2.2x_5 + 3x_3^2 + x_1x_2x_4 + 1.3x_2x_4x_5 \\ \text{subject to} & x_2x_3 = 0, \ x_3x_4 = 0, \ x_1, x_2 \in \{0,1\}, \ x_3, x_4, x_5 \in [0,1] \end{array} \right\} \tag{2}$$

In this case, $I_{\rm bin} = \{1, 2\}$, $I_{\rm box} = \{3, 4, 5\}$ and $\mathcal{C} = \{\{2, 3\}, \{3, 4\}\}$. We can easily compute the optimal solution $\boldsymbol{x}^* = (0, 0, 0.3, 0, 1)$ and the optimal value $\zeta^* = -2.47$.

The input of BBCPOP.m consists of the data for POP (1), the relaxation order ω which determines the hierarchy level of DNN relaxation to be constructed, and parameters which control the execution of BBCPOP.m. With the input data, BBCPOP.m constructs a simple conic optimization problem (COP):

$$y_0^* \quad = \quad \max_{y_0, \boldsymbol{Y}_1, \boldsymbol{Y}_2} \{ y_0 \mid \boldsymbol{Q}_0 - y_0 \boldsymbol{H}_0 = \boldsymbol{Y}_1 + \boldsymbol{Y}_2, \ \boldsymbol{Y}_1 \in \mathbb{K}_1^*, \ \boldsymbol{Y}_2 \in \mathbb{K}_2^* \}, \tag{3}$$

which serves as a DNN relaxation of POP (1) (hence $y_0^* \leq \zeta^*$). Here $\boldsymbol{H}_0$ denotes a constant vector in a linear space $\mathbb{V}$ (= the Cartesian product of symmetric matrix spaces) endowed with an inner product, $\mathbb{K}_1, \mathbb{K}_2 \subset \mathbb{V}$ closed convex cones, and $y_0 \in \mathbb{R}$, $\boldsymbol{Y}_1 \in \mathbb{V}$, $\boldsymbol{Y}_2 \in \mathbb{V}$ variables. The output of BBCPOP.m is an approximate optimal solution $(y_0, \boldsymbol{Y}_1, \boldsymbol{Y}_2)$ of (3) such that $y_0 \leq y_0^*$; hence $y_0 \leq \zeta^*$ is guaranteed.

The degree of POP (1) is defined as $\max\{\deg f_0,\ |C|\ (C \in \mathcal{C})\}$, where $|C|$ denotes the number of elements in C ($C \in \mathcal{C}$). The relaxation order ω needs to be a positive integer not less than the half of the degree of POP (1). Hence the minimum relaxation order that can be taken is

$$\omega_{\min} = \text{the smallest positive integer not less than the half of the degree of POP (1).}$$

As we take a larger ω, we can expect a tighter lower bound y_0 (= the approximate optimal value of (3)) for ζ^*, but COP (3) to be solved by the BP Algorithm becomes larger, so that longer execution time is required. For many application problems in practice, taking $\omega_{\min}$ for the relaxation order ω is sufficient to obtain a tight lower bound for ζ^*.

In the above example (2), we see that $\deg f_0 = 3$, $|\{2,3\}| = 2$ and $|\{3,4\}| = 2$. Hence $\omega_{\min} = 2 \leq \omega$.

In Section 2, we present how to describe POP (1) for the input of the main function BBCPOP.m. In Section 3, we illustrate the execution of BBCPOP.m and present the details on the output of BBCPOP.m. Section 4 lists the parameters which control the execution of BBCPOP.m, and Section 5 some main functions contained in the BBCPOP software package.

2 Representing polynomial optimization problems

POP (1) is described by

$$\mathsf{objPoly},\ \mathsf{I01}\ \text{ and } \mathsf{Icomp},$$

which are input arguments of BBCPOP.m. Let $\text{term}f_0$ denote the number of terms of f_0. We employ a simplified SparsePOP format to describe the objective function of POP (1):

$$\begin{aligned} \mathsf{objPoly.supports} &= \text{a set of supports of } f_0(\boldsymbol{x}),\ \ \text{term}f_0 \times n \text{ matrix.} \\ \mathsf{objPoly.coef} &= \text{coefficients, column vector of dimension term}f_0. \end{aligned}$$

For the original SparsePOP format, see [7]. The functions simplifyPoly.m, addPoly.m and multiplyPoly.m in the directory polyTools/ can be used when describing an objective function $f_0(\boldsymbol{x})$ in the simplified SparsePOP format.

The partition $I_{\text{bin}} \cup I_{\text{box}}$ of $N \equiv \{1, 2, \ldots, n\}$ is described by the n-dimensional row vector I01 such that

$$\mathsf{I01}_j = \begin{cases} \mathsf{true} & \text{if } j \in I_{\text{bin}}, \\ \mathsf{false} & \text{otherwise, } i.e.,\ j \in I_{\text{box}}. \end{cases}$$

The family $\mathcal{C}$ of subsets of N, which represents the complementarity condition in POP (1), is represented by Icomp. Suppose that $\mathcal{C}$ consists of m nonempty subsets $C_1, \ldots, C_m$ of N. Then we set Icomp to be the $m \times n$ matrix such that

$$\mathsf{Icomp}_{ij} = \begin{cases} \mathsf{true} & \text{if } j \in C_i, \\ \mathsf{false} & \text{otherwise, } i.e.,\ j \notin C_i \end{cases}$$

$(i = 1, \ldots, m,\ j = 1, \ldots, n)$. If $\mathcal{C} = \emptyset$, set $\mathsf{Icomp} = [\ ]$.

The three input arguments objPoly, I01 and Icomp for POP (2) are described as follows:

```
function [objPoly, I01, Icomp] = example1;
       objPoly.supports   = ...
                         [1 0 0 0 0;
                          0 0 1 0 0;
                          0 0 0 0 1;
                          0 0 2 0 0;
                          1 1 0 1 0;
                          0 1 0 1 1];
         objPoly.coef = [0.5; -1.8; -2.2; 3; 1; 1.3];
         Icomp = logical([0 1 1 0 0; 0 0 1 1 0]);
         I01 = logical([1 1 0 0 0]);
end
```

3 Executing BBCPOP

A lower bound for the optimal value of POP (2) can be computed by BBCPOP.m as follows:

```
>>[objPoly, I01, Icomp] = example1;
>>relaxOrder = 2; params = [];
>>[sol, info] = BBCPOP(objPoly, I01, Icomp, relaxOrder, params);
```

The following is shown on the screen as the BBCPOP.m terminates at 17 iterations.

```
Original
iter=17:y0=-2.469903e+00,LBv=-2.470066e+00,[LB,UB]=[-2.470066e+00,-2.469903e+00]
Scaled
iter=17:y0=-9.058401e-01,LBv=-9.058999e-01,[LB,UB]=[-9.058999e-01,-9.058401e-01]

     LBv            y0             UB             UB-LBv   relnormX iter b_yes
 0  -1.000000e+21 +5.800000e+00 +5.800000e+00 9.80e+00
 1  -4.536533e+01 +5.800000e+00 +5.800000e+00 9.80e+00 8.59e-01  101 90
 2  -1.867610e+01 +9.000000e-01 +9.000000e-01 4.90e+00 4.31e-01   56 1
 3  -6.974229e+00 -1.550000e+00 -1.550000e+00 2.45e+00 9.58e-02  218 90
 4  -2.775000e+00 -2.775000e+00 -1.550000e+00 1.22e+00 0.00e+00   17 2
 5  -2.775000e+00 -2.162500e+00 -2.162500e+00 6.12e-01 3.06e-02  301 88
 6  -2.476112e+00 -2.468750e+00 -2.162500e+00 3.14e-01 1.21e-04  350 88
 7  -2.476112e+00 -2.319306e+00 -2.319306e+00 1.57e-01 1.48e-02  274 90
 8  -2.476112e+00 -2.397709e+00 -2.397709e+00 7.84e-02 7.06e-03  180 90
 9  -2.476112e+00 -2.436911e+00 -2.436911e+00 3.92e-02 3.22e-03  301 88
10  -2.476112e+00 -2.456512e+00 -2.456512e+00 1.96e-02 1.31e-03  294 90
11  -2.476112e+00 -2.466312e+00 -2.466312e+00 9.80e-03 3.58e-04  301 88
12  -2.471212e+00 -2.471212e+00 -2.466312e+00 4.90e-03 0.00e+00   10 2
13  -2.471212e+00 -2.468762e+00 -2.468762e+00 2.45e-03 1.20e-04  301 88
14  -2.470066e+00 -2.469987e+00 -2.468762e+00 1.30e-03 1.29e-06  339 97
```

```
15  -2.470066e+00 -2.469414e+00 -2.469414e+00 6.52e-04 5.69e-05  339 97
16  -2.470066e+00 -2.469740e+00 -2.469740e+00 3.26e-04 2.53e-05  339 97
17  -2.470066e+00 -2.469903e+00 -2.469903e+00 1.63e-04 9.45e-06  301 99
 timeBP (Excecution time) = 1.98,  termcodeBP = 2
```

The value $-2.470039e{+}00$ is the approximate optimal value of (2) obtained by BP Algorithm, which is a valid lower bound of the optimal value $\zeta^* = -2.47$ of POP (2).

The output sol is a structure containing the solution information for (3):

```
y0init: 2.1272
   LBv: -2.4700
    LB: -2.4700
    UB: -2.4700
    Y1: [1226 double]
    Y2: [1226 double]
```

Here y0init corresponds to the initial value of y_0^m, and LBv, LB, UB, Y1, Y2 to the terminal values of $y_0^{\ell v}$, y^ℓ, y_0^u, $\hat{\boldsymbol{Y}}_1$ and $\hat{\boldsymbol{Y}}_2$, respectively, in BP Algorithm in [4]. The output info is a structure for some of the execution information of BP.m (BP Algorithm) and APGR.m (APGR Algorithm):

```
    iterBP: 17
    timeBP: 1.5884
termcodeBP: 2
  iterAPGR: 3545
```

Here iterBP denotes the number of iteration of BP.m, timeBP its execution time, termcodeBP its termination code and iterAPGR the total number of iterations of APGR.m. BP.m stops when

(i) the difference of the upper bound y_0^u and the lower bound y_0^ℓ becomes smaller than the prescribed parameter params.delta,

(ii) $(y_0^u - y_0^\ell)/\max\{1.0, |y_0^\ell|, |y_0^u|\}$ is smaller than the prescribed parameter params.delta2

(iii) the reduction of the length of the interval $[y_0^u - y_0^\ell]$ gets smaller than max{params.delta, params.delta2}.

(iv) the iteration of BP.m exceeds the prescribed parameter params.maxiterBP, or

(v) the execution time exceeds the prescribed parameter params.maxtimeBP.

These terminations are associated with the termination code 1,2,3,0,-1, respectively.

4 Parameters

In addition to objPoly, I01, Icomp and relaxOrder for describing a POP, the MATLAB function BBCPOP.m has params as an input argument. It is a structure consisting of many parameters that control the performance of the function. Table 1 shows the list of parameters used in BBCPOP.m. The default values of parameters are given in the MATLAB function defaultParamBP.m. They can be modified if necessary.

Table 1: The fields of params, default values and possible values

field of params	default	usage, possible values
Parameters for DNN relaxation		
sparseSW	1	Set 0 for dense DNN relaxation. Set 1 for sparse DNN relaxation.
Parameters for BP and APGR Algorithms		
maxtimeBP	20000	the maximum execution time
maxiterBP	40	the maximum iteration for BP Algorithm
maxiterAPGR	20000	the maximum iteration for APGR Algorithm
delta1	1e-4	the relative tolerance for BP Algorithm Set 0 if delta is used.
delta	0	the absolute tolerance for BP Algorithm Set delta $\in (0, 1)$ if ζ^* is integer. Set 0 otherwise.
printyes	2	print level, 0,1,2 or 3
UbdObjVal	$f_0(\mathbf{0})$	the upper bound for the optimal value ζ^* of POP (1)
UbdIX		$= \rho$ given in Assumption (A1) of [4]. To specify this parameter, see Sections 2.3 and 4.1 of [4].

5 Some functions

We list some functions contained in the BBCPOP software package. Important subfunctions of BBCPOP.m are listed in Section 5.1, and functions for some test instances are listed in Section 5.2.

5.1 Main subfunctions of BBCPOP.m

relaxation/BBCPOPtoDNN.m constructs COP (3), which serves as a DNN relaxation of (1).

solver/solveDNN.m solves COP (3) by applying BP Algorithm and APGR Algorithm.

solver/BP/BP.m is an implementation of BP Algorithm in [4].

solver/BP/APGR.m is an implementation of APGR Algorithm in [4].

5.2 Functions for POP instances

All the functions below output objPoly, I01, Icomp, relaxOrder $= \omega_{min}$ and params, which can be used as input of BBCPOP.m.

instances/POPrandom/genPOPdense.m generates a dense POP instance with binary, box and complementarity constraints.

```
>> degree=3; nDim=5; isBin=true; addComplement=false;
>> [objPoly,I01,Icomp,relaxOrder,params] = ...
        genPOPdense(degree,nDim,isBin,addComplement);
```

instances/POPrandom/genPOParrow.m generates a sparse POP instance (whose objective function $f_0(\boldsymbol{x})$ has an arrow type sparsity pattern Hessian matrix) with binary, box and complementarity constraints.

```
>> degree=4; a=10; b=2; c=2; l=3; isBin=0; addComplement=true;
>> [objPoly, I01, Icomp, relaxOrder, params] = ...
        genPOParrow(degree,a,b,c,l,isBin,addComplement);
```

instances/POPrandom/genPOPchordal.m generates a sparse POP instance (whose objective function $f_0(\boldsymbol{x})$ has a sparse Hessian matrix characterized by a chordal graph) with binary, box and complementarity constraints.

```
>> degree=3; nDim=100; radiorange=0.1; isBin=1; addComplement=false;
>> [objPoly, I01, Icomp, relaxOrder, params] = ...
        genPOPchordal(degree,nDim,radiorange,isBin,addComplement);
```

instances/QAP/qapreadBP.m generates a QOP with box and complementarity constraints which is induced from a Lagrangian relaxation of a QAP instance from QAPLIB [3].

```
>> instance='chr12a'; lambda=10000;
>> [objPoly, Icomp, I01, relaxOrder, params] = ...
        qapreadBP(instance,lambda);
```

instances/BIQ/biqreadBP.m generates a QOP with box and complementarity constraints which is induced from a Lagrangian relaxation of a binary/box constrained QOP instance from BIQMAC [8].

```
>> instance='bqp100-1'; lambda=10000;
>> [objPoly, I01, Icomp, relaxOrder, params] = ...
        biqreadBP(instance,lambda);
```

References

[1] N. Arima, S. Kim, M. Kojima, and K.C. Toh. A robust Lagrangian-DNN method for a class of quadratic optimization problems. *Comput. Optim. Appl.*, 66(3):453–479, 2017.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2:183–202, 2009.

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

[4] 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. Research Rport B-48?, Tokyo Institute of Technology, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro-ku, Tokyo 152-8552, March 2018.

[5] S. Kim, M. Kojima, and K. C. Toh. Doubly nonnegative relaxations for quadratic and polynomial optimization problems with binary and box constraints. Research Rport B-483, Tokyo Institute of Technology, Department of Mathematical and Computing Sciences, Oh-Okayama, Meguro-ku, Tokyo 152-8552, July 2016.

[6] 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.

[7] H. Waki, S. Kim, M. Kojima, M. Muramatsu, H. Sugimoto, and M. Yamashita. *User Manual for SparsePOP: a Sparse Semidefinite Programming Relaxation of Polynomial Optimization Problems.* https://sourceforge.net/projects/sparsepop/?source=typ_redirect, August 2009.

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