Research Reports on Mathematical and Computing Sciences

User's Manual for SparseCoLO:
Conversion Methods for **SPARSE CO**nic-form
Linear **O**ptimization Problems

K. Fujisawa, S. Kim, M. Kojima,
Y. Okamoto, and M. Yamashita

February 2009, B–453

**Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology**

SERIES B: Operations Research

B-453 User's Manual for **SparseCoLO**: Conversion Methods for **SPARSE CO**nic-form **L**inear **O**ptimization Problems

Katsuki Fujisawa[★], Sunyoung Kim[†], Masakazu Kojima[‡], Yoshio Okamoto[‡], and Makoto Yamashita[‡]

February 2009

Abstract.

SparseCoLO is a Matlab package for implementing the four conversion methods, proposed by Kim, Kojima, Mevissen, and Yamashita, via positive semidefinite matrix completion for an optimization problem with matrix inequalities satisfying a sparse chordal graph structure. It is based on quite a general description of optimization problem including both primal and dual form of linear, semidefinite, second-order cone programs with equality/inequality constraints. Among the four conversion methods, two methods utilize the domain-space sparsity of a semidefinite matrix variable and the other two methods the range-space sparsity of a linear matrix inequality (LMI) constraint of the given problem. **SparseCoLO** can be used as a preprocessor to reduce the size of the given problem before applying semidefinite programming solvers. The website for this package is

http://www.is.titech.ac.jp/∼kojima/SparseCoLO

where the package **SparseCoLO** and this manual can be downloaded.

Key words.
Semidefinite program, Sparsity exploitation, Positive semidefinite matrix completion, Conversion methods, Matlab software package.

★ Department of Industrial and Systems Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan. *fujisawa@indsys.chuo-u.ac.jp*

† Department of Mathematics, Ewha W. University, 11-1 Dahyun-dong, Sudaemoon-gu, Seoul 120-750 Korea. S. Kim's research was supported by KRF 2008-531-C00013. *skim@ewha.ac.kr*

‡ Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. This work was supported by Global COE Program "Computationism as a Foundation for the Sciences" and Grant-in-Aid for Scientific Research from Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science. *kojima@is.titech.ac.jp*, *okamoto@is.titech.ac.jp*, *Makoto.Yamashita@is.titech.ac.jp*

We are collecting instances of sparse linear optimization problems (LOPs) to evaluate and improve performances of SparseCoLO. Any instances of sparse LOPs that you could send us to kojima.m.aa-sparsecolo@m.titech.ac.jp would be very much appreciated.

1 Introduction

We introduce a Matlab package SparseCoLO for solving a conic-form linear optimization problem (LOP) of the form:

$$\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \text{ subject to } \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \in \text{J}, \ \boldsymbol{x} \in \text{K}, \tag{1}$$

where

$$
\begin{aligned}
\text{K} &= \text{K.f} \times \text{K.}\ell \times \text{K.q} \times \text{K.s}, \\
\text{K.f} &= \text{a column vector space to denote free variables,} \\
\text{K.}\ell &= \text{an LP cone,} \\
\text{K.q} &= \text{an SOCP cone or a product of SOCP cones,} \\
\text{K.s} &= \text{a column vector space corresponding to an SDP cone or} \\
&\qquad \text{a product of SDP cones,} \\
\text{J} &= \text{J.f} \times \text{J.}\ell \times \text{J.q} \times \text{J.s}, \\
\text{J.f} &= \{\boldsymbol{0}\} \subset \text{a column vector space to denote equality constraints,} \\
\text{J.}\ell &= \text{an LP cone,} \\
\text{J.q} &= \text{an SOCP cone or a product of SOCP cones,} \\
\text{J.s} &= \text{a column vector space corresponding to an SDP cone or} \\
&\qquad \text{a product of SDP cones,} \\
\boldsymbol{c}^T &= (\boldsymbol{c}_f^T, \boldsymbol{c}_\ell^T, \boldsymbol{c}_q^T, \boldsymbol{c}_s^T), \\
\boldsymbol{A} &= \begin{pmatrix} \boldsymbol{A}_{ff} & \boldsymbol{A}_{f\ell} & \boldsymbol{A}_{fq} & \boldsymbol{A}_{fs} \\ \boldsymbol{A}_{\ell f} & \boldsymbol{A}_{\ell\ell} & \boldsymbol{A}_{\ell q} & \boldsymbol{A}_{\ell s} \\ \boldsymbol{A}_{qf} & \boldsymbol{A}_{q\ell} & \boldsymbol{A}_{qq} & \boldsymbol{A}_{qs} \\ \boldsymbol{A}_{sf} & \boldsymbol{A}_{s\ell} & \boldsymbol{A}_{sq} & \boldsymbol{A}_{ss} \end{pmatrix}, \ \boldsymbol{b} = \begin{pmatrix} \boldsymbol{b}_f \\ \boldsymbol{b}_\ell \\ \boldsymbol{b}_q \\ \boldsymbol{b}_s \end{pmatrix}.
\end{aligned}
$$

The main feature of SparseCoLO is that it detects and utilizes two types of sparsities, characterized in terms of a sparse chordal graph structure, in the variable $\boldsymbol{x} \in \text{K}$ and in the constraint $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \in \text{J}$ to reduce the size of the problem. It can be used as a preprocessor to reduce the size of (1) before applying semidefinite programming solvers such as CSDP [1], SDPA [3], SDPT3 [7], SeDuMi [6]. Currently, SparseCoLO calls SeDuMi to solve the resulting problem after the conversion methods are applied to the problem (1).

The problem (1) indeed represents various problems. For instance, if $\text{J.}\ell = \emptyset$, $\text{J.q} = \emptyset$ and $\text{J.s} = \emptyset$, (1) includes a primal form LOP (an equality standard form LOP) in the SeDuMi format [6] as a special case:

$$\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \text{ subject to } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \ \boldsymbol{x} \in \text{K}.$$

In the case that $\text{K.}\ell = \emptyset$, $\text{K.q} = \emptyset$ and $\text{K.s} = \emptyset$, the problem (1) becomes a dual form LOP (an LMI standard form LOP) in the SeDuMi format:

$$\text{maximize} \quad -\boldsymbol{c}^T \boldsymbol{x} \text{ subject to } \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \in \text{J},$$

where $\boldsymbol{x}$ denotes a free variable vector. Thus we may regard the conic-form LOP (1) as a unified extension of primal and dual standard form LOPs. This form of LOP will provide not only flexibility in modeling practical problems in terms of LOPs, but also a convenient framework for the domain- and range-space conversion methods proposed in the paper [4].
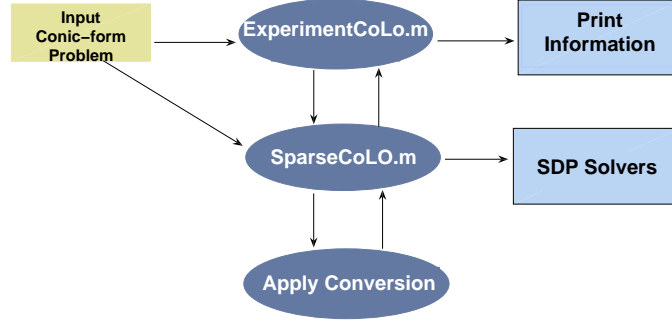
Figure 1: The structure of SparseCoLO

We can rewrite the conic-form LOP (1) as

$$
\begin{array}{ll}
\text{minimize} & \boldsymbol{c}_f^T \boldsymbol{x}_f + \boldsymbol{c}_\ell^T \boldsymbol{x}_\ell + \boldsymbol{c}_q^T \boldsymbol{x}_q + \boldsymbol{c}_s^T \boldsymbol{x}_s \\
\text{subject to} & \boldsymbol{A}_{ff}\boldsymbol{x}_f + \boldsymbol{A}_{f\ell}\boldsymbol{x}_\ell + \boldsymbol{A}_{fq}\boldsymbol{x}_q + \boldsymbol{A}_{fs}\boldsymbol{x}_s - \boldsymbol{b}_f \in \text{J.f}, \\
& \boldsymbol{A}_{\ell f}\boldsymbol{x}_f + \boldsymbol{A}_{\ell\ell}\boldsymbol{x}_\ell + \boldsymbol{A}_{\ell q}\boldsymbol{x}_q + \boldsymbol{A}_{\ell s}\boldsymbol{x}_s - \boldsymbol{b}_\ell \in \text{J}.\ell \\
& \boldsymbol{A}_{qf}\boldsymbol{x}_f + \boldsymbol{A}_{q\ell}\boldsymbol{x}_\ell + \boldsymbol{A}_{qq}\boldsymbol{x}_q + \boldsymbol{A}_{qs}\boldsymbol{x}_s - \boldsymbol{b}_q \in \text{J.q} \\
& \boldsymbol{A}_{sf}\boldsymbol{x}_f + \boldsymbol{A}_{s\ell}\boldsymbol{x}_\ell + \boldsymbol{A}_{sq}\boldsymbol{x}_q + \boldsymbol{A}_{ss}\boldsymbol{x}_s - \boldsymbol{b}_s \in \text{J.s}, \\
& \boldsymbol{x}_f \in \text{K.f}, \ \boldsymbol{x}_\ell \in \text{K}.\ell, \ \boldsymbol{x}_q \in \text{K.q}, \ \boldsymbol{x}_s \in \text{K.s}.
\end{array}
\right\} \tag{2}
$$

We use the quintuplet $(\boldsymbol{A}, \ \boldsymbol{b}, \ \boldsymbol{c}, \ \text{K}, \ \text{J})$ to denote the conic-form linear optimization problem (2). The dual of (2) above is of the form

$$
\begin{array}{ll}
\text{minimize} & -\boldsymbol{b}_f^T \boldsymbol{y}_f - \boldsymbol{b}_\ell^T \boldsymbol{y}_\ell - \boldsymbol{b}_q^T \boldsymbol{y}_q - \boldsymbol{b}_s^T \boldsymbol{y}_s \\
& \boldsymbol{c}_f - \boldsymbol{A}_{ff}^T \boldsymbol{y}_f - \boldsymbol{A}_{\ell f}^T \boldsymbol{y}_\ell - \boldsymbol{A}_{qf}^T \boldsymbol{y}_q - \boldsymbol{A}_{sf}^T \boldsymbol{y}_s \in \text{K.f}^*, \\
& \boldsymbol{c}_\ell - \boldsymbol{A}_{f\ell}^T \boldsymbol{y}_f - \boldsymbol{A}_{\ell\ell}^T \boldsymbol{y}_\ell - \boldsymbol{A}_{q\ell}^T \boldsymbol{y}_q - \boldsymbol{A}_{s\ell}^T \boldsymbol{y}_s \in \text{K}.\ell, \\
& \boldsymbol{c}_q - \boldsymbol{A}_{fq}^T \boldsymbol{y}_f - \boldsymbol{A}_{\ell q}^T \boldsymbol{y}_\ell - \boldsymbol{A}_{qq}^T \boldsymbol{y}_q - \boldsymbol{A}_{sq}^T \boldsymbol{y}_s \in \text{K.q}, \\
& \boldsymbol{c}_s - \boldsymbol{A}_{fs}^T \boldsymbol{y}_f - \boldsymbol{A}_{\ell s}^T \boldsymbol{y}_\ell - \boldsymbol{A}_{qs}^T \boldsymbol{y}_q - \boldsymbol{A}_{ss}^T \boldsymbol{y}_s \in \text{K.s}, \\
& \boldsymbol{y}_f \in \text{J.f}^*, \ \boldsymbol{y}_\ell \in \text{J}.\ell, \ \boldsymbol{y}_q \in \text{J.q}, \ \boldsymbol{y}_s \in \text{J.s},
\end{array}
$$

where K.f$^*(= \{\boldsymbol{0}\})$ is the dual cone of K.f, that is, the space of column vectors for equality constraints, and similarly, J.f$^*(=$ the Euclidean space with the same dimension) is the dual cone of J.f, the space of column vectors for free variables.

The package SparseCoLO contains sparseCoLO.m as the main function, experimentCoLO.m and evaluateCoLO.m for numerical experiments, a set of test problems in the directory examples, and the functions for conversion methods in the directory conversion. The structure of SparseCoLo is illustrated in Figure 1.

In the program sparseCoLO.m, we denote the dual LOP as the quintuplet $(-\boldsymbol{A}', \ -\boldsymbol{c}, \ -\boldsymbol{b}, \ \text{J}, \ \text{K})$, where $\boldsymbol{A}'$ denotes the transpose of the matrix $\boldsymbol{A}$. Under the assumption that the conic-form LOP (2) satisfies a sparsity characterized by a chordal graph structure, sparseCoLO.m

1. transforms the conic-form LOP (2) into another LOP of smaller size by applying the domain- and range-space conversion methods developed in [4], and then

2. calls SeDuMi [6] to solve the resulting LOP efficiently.

In Section 2, we describe how to select the conversion methods using parameters. The usage of SparseCoLO is illustrated with commands in Section 3. Section 4 includes the description of the parameters and output. Section 5 summarizes the manual. In Appendix, we demonstrate the effectiveness of SparseCoLO using the max-cut and the norm minimization problem.

2 The domain- and range-space conversion methods

SparseCoLO includes implementation of the four conversion methods developed in [4]:

(a) The d-space conversion method using clique trees. See Section 3.1 of [4].

(b) The d-space conversion method using basis representation. See Section 3.2 of [4].

(c) The r-space conversion method using clique trees. See Section 5.1 of [4].

(d) The r-space conversion method using matrix decomposition. See Section 5.2 of [4].

We can choose either (a) or (b) by specifying the parameter parCoLO.domain = 1 or parCoLO.domain = 2, respectively, while either (c) or (d) can be chosen by specifying the parameter parCoLO.range = 1 or parCoLO.range = 2, respectively. If no conversion is used, both of the parameters should be set to 0.

After applying the conversion methods specified by the parameters parCoLO.domain and parCoLO.range, the resulting LOP is further converted into an equality standard form or a linear matrix inequality (LMI) standard form for SeDuMi to solve the problem by setting parCoLO.EQorLMI = 1 or parCoLO.EQorLMI = 2.

3 Examples and Sample execution

Let us consider a norm minimization problem

$$\text{minimize } \left\| \boldsymbol{F}_0 + \sum_{k=1}^{r} \boldsymbol{F}_k z_k \right\|_2 \text{ subject to } \|\boldsymbol{z}\| \leq 1.$$

Here $\boldsymbol{z} = (z_1, z_2, \ldots, z_r) \in \mathbb{R}^r$ denotes a variable vector, $\|\boldsymbol{z}\|$ the Euclidean norm of $\boldsymbol{z}$, $\boldsymbol{F}_k$ a $p \times q$ matrix and $\|\boldsymbol{F}\|_2$ the operator norm of $\boldsymbol{F}$. We can formulate this problem as an LOP:

$$
\begin{aligned}
\text{minimize} \quad & \zeta \\
\text{subject to} \quad & z_0 = 1, \\
& \sum_{k=1}^{r} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_k \\ \boldsymbol{F}_k^T & \boldsymbol{O} \end{pmatrix} z_k + \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} \zeta + \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_0 \\ \boldsymbol{F}_0^T & \boldsymbol{O} \end{pmatrix} \\
& \qquad \in \text{ an SDP cone with dimension } (p + q), \\
& \zeta \in \mathbb{R}, \ (z_0, z_1, z_2, \ldots, z_r) \in \text{ an SOCP cone with dimension } r.
\end{aligned}
$$

Thus, letting K.f = 1, K.ℓ = [ ], K.q = [1 + r], K.s = [ ], J.f = 1, J.ℓ = [ ], J.q = [ ], J.s = [q + r], we can describe the conic-form LOP above as in (2). Not that the coefficient matrices of the conic-form LOP become sparse when q is small.

The SparseCoLO package contains a MATLAB program to generate the norm minimization problem. For example, type

```
>> [A,b,c,K,J] = normMinSDP(200,5,10,2009);
```

to load the data $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, K, J for the conic-form LOP formulation of the norm minimization problem with $p = 200$, $q = 5$, and $r = 10$. We note that the last input, 2009 in the example, specifies a seed for the random number generator. Then, type

```
>> parCoLO.domain = 0; parCoLO.range = 1; parCoLO.EQorLMI = 1;
```

to specify the parameters for no conversion in the domain space, the r-space conversion using clique trees and the conversion into an equality standard form. (The LOP (A, b, c, K, J) formulated does not involve any variable matrix, so that the domain-space conversion is not relevant). Finally, issue a command to solve the norm minimization problem as follows:

```
>> [x,y,infoCoLO,cliqueDomain,cliqueRange,LOP] = sparseCoLO(A,b,c,K,J,parCoLO);
parCoLO.domain = 0; parCoLO.range = 1; parCoLO.EQorLMI = 1
Apply the r-space conversion method using clique trees.
Conversion into an equality standard form
SeDuMi 1.1R3 by AdvOL, 2006 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
Split 241 free variables
eqs m = 2540, order n = 770, dim = 5287, blocks = 19
nnz(A) = 13910 + 0, nnz(ADA) = 1365658, nnz(L) = 868957
 it :      b*y          gap    delta  rate    t/tP*  t/tD*   feas cg cg  prec
  0 :             2.70E-02 0.000
  1 :    3.12E+01 5.47E-03 0.000 0.2027 0.9000 0.9000  -0.95  1  1  4.7E+00
  2 :    1.00E+02 1.03E-03 0.000 0.1878 0.9000 0.9000  -0.72  1  1  2.7E+00
  3 :    5.33E+01 2.61E-04 0.000 0.2539 0.9000 0.9000   0.66  1  1  6.5E-01
  4 :    2.81E+01 1.05E-04 0.000 0.4035 0.9000 0.9000   2.31  1  1  1.4E-01
  5 :    1.26E+01 2.83E-05 0.000 0.2696 0.9000 0.9000   2.83  1  1  1.7E-02
  6 :    9.55E+00 8.49E-06 0.000 0.2996 0.9000 0.9000   1.75  1  1  4.0E-03
  7 :    9.17E+00 2.46E-06 0.000 0.2897 0.9000 0.9000   1.14  1  1  1.1E-03
  8 :    9.09E+00 5.70E-07 0.000 0.2317 0.9000 0.9000   1.02  1  1  2.6E-04
  9 :    9.08E+00 3.54E-08 0.000 0.0620 0.9000 0.5415   1.01  1  1  9.7E-05
 10 :    9.07E+00 5.41E-09 0.000 0.1531 0.9105 0.9000   1.00  1  1  1.7E-05
 11 :    9.07E+00 1.84E-10 0.340 0.0339 0.9903 0.9900   1.00  1  1  6.1E-07
 12 :    9.07E+00 7.61E-12 0.103 0.0414 0.9675 0.9683   1.00  2  2  2.5E-08
 13 :    9.07E+00 4.97E-13 0.476 0.0653 0.9900 0.9900   1.00  2  2  1.6E-09
 14 :    9.07E+00 9.58E-14 0.000 0.1929 0.9011 0.9000   1.00  3  3  3.1E-10

iter seconds digits      c*x                b*y
 14      8.1   Inf  9.0699974057e+00   9.0699974538e+00
|Ax-b| =   4.1e-10, [Ay-c]_+ =   1.5E-10, |x|= 2.1e+02, |y|=  1.6e+00

Detailed timing (sec)
   Pre          IPM          Post
6.300E-01    8.120E+00    8.000E-02
Max-norms: ||b||=1.997582e+00, ||c|| = 1,
Cholesky |add|=0, |skip| = 0, ||L.L|| = 46.1161.
```

The last input argument parCoLO can be omitted. Then sparseCoLO.m calls defaultParCoLO to select suitable conversion methods for the given LOP (A, b, c, K, J). For example, type

```
>>  [A,b,c,K,J] = normMinSDP(200,5,10,2009);
>> [x,y,infoCoLO,cliqueDomain,cliqueRange,LOP] = sparseCoLO(A,b,c,K,J);
```

Then,

```
parCoLO.domain = 0; parCoLO.range = 2; parCoLO.EQorLMI = 1
```

```
Also try:
    parCoLO.domain = 0; parCoLO.range = 1; parCoLO.EQorLMI = 2


Apply the r-space conversion method using matrix decomposition.
LOP to be converted into equality standard form is already equality standard form.
SeDuMi 1.1R3 by AdvOL, 2006 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
Split 1 free variables
eqs m = 1216, order n = 1205, dim = 7214, blocks = 202
nnz(A) = 4610 + 10001, nnz(ADA) = 83225, nnz(L) = 42221
Handling 11 + 1 dense columns.
 it :      b*y          gap    delta   rate   t/tP*  t/tD*   feas cg cg  prec
  0 :            2.32E+01 0.000
  1 :    1.46E+02 7.85E-01 0.000 0.0339 0.9900 0.9900  -0.95  1  1  2.3E+02
  2 :    8.48E+01 5.28E-01 0.000 0.6729 0.9000 0.9000   1.41  1  1  9.7E+01
  3 :    5.02E+01 3.21E-01 0.000 0.6071 0.9000 0.9000   3.79  1  1  2.4E+01
  4 :    2.32E+01 1.12E-01 0.000 0.3500 0.9000 0.9000   3.60  1  1  3.0E+00
  5 :    1.13E+01 3.79E-02 0.000 0.3380 0.9000 0.9000   3.07  1  1  4.7E-01
  6 :    9.32E+00 1.09E-02 0.000 0.2880 0.9000 0.9000   1.55  1  1  1.1E-01
  7 :    9.16E+00 3.82E-03 0.000 0.3495 0.9000 0.9000   1.09  1  1  3.8E-02
  8 :    9.12E+00 1.82E-03 0.000 0.4759 0.9000 0.9000   1.03  1  1  1.8E-02
  9 :    9.09E+00 5.11E-04 0.000 0.2811 0.9031 0.9000   1.01  1  1  5.4E-03
 10 :    9.08E+00 7.79E-05 0.000 0.1525 0.9243 0.9000   1.00  1  1  1.3E-03
 11 :    9.07E+00 1.63E-06 0.000 0.0210 0.9266 0.9000   1.00  1  1  1.0E-04
 12 :    9.07E+00 4.49E-09 0.000 0.0027 0.9906 0.9900   1.00  1  1  1.3E-06
 13 :    9.07E+00 8.72E-10 0.123 0.1942 0.9000 0.6601   1.00  1  2  3.5E-07
 14 :    9.07E+00 1.94E-11 0.000 0.0222 0.9901 0.9900   1.00  1  1  9.4E-09
 15 :    9.07E+00 1.59E-12 0.412 0.0820 0.9900 0.9900   1.00  3  3  7.8E-10


iter seconds digits       c*x                b*y
 15      2.5   Inf  9.0699974751e+00   9.0699976838e+00
|Ax-b| =   9.1e-10, [Ay-c]_+ =   3.1E-10, |x|=  1.3e+02, |y|=  5.9e-01


Detailed timing (sec)
   Pre           IPM          Post
2.800E-01     2.500E+00     2.000E-02
Max-norms: ||b||=1.997582e+00, ||c|| = 1,
Cholesky |add|=0, |skip| = 2, ||L.L|| = 765.641.
```

If one wants to know the objective values and the feasibility errors for the solution obtained, type

```
>> [primalObjValue, dualObjValue, primalfeasibility, dualfeasibility] = ...
           evaluateCoLO(x,y,A,b,c,K,J,cliqueDomain,cliqueRange)
primalObjValue =  9.0700
dualObjValue =  9.0700
primalfeasibility =   2.1253e-11
dualfeasibility =   6.5280e-09
```

The package SparseCoLO contains a MATLAB program experimentCoLO.m for numerical experiments using sparseCoLO.m. To use this program, "SelectedParaSet" needs to be specified

in the program. It is a $k \times 3$ matrix with each row of values for parCoLO.domain, parCoLO.range and parCoLO.EQorLMI. For example, let

```
SelectedParaSet = [0,0,2; 0,2,1; 0,1,2];
```

Then,

```
>> experimentCoLO(A,b,c,K,J);
parCoLO.domain = 0; parCoLO.range = 0; parCoLO.EQorLMI = 2
Conversion into an LMI standard form
SeDuMi 1.1R3 by AdvOL, 2006 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, Adaptive Step-Differentiation, theta = 0.250, beta = 0.500
Split 1 free variables
eqs m = 12, order n = 210, dim = 42039, blocks = 3
nnz(A) = 10218 + 0, nnz(ADA) = 142, nnz(L) = 77
 it :      b*y        gap    delta  rate   t/tP*  t/tD*   feas cg cg  prec
  0 :            1.39E+02 0.000
  1 :  -2.48E+01 9.73E+00 0.000 0.0702 0.9900 0.9900  -0.95  1  1  2.6E+02
. . . . . . . . . .
. . . . . . . . . .

parCoLO.domain = 0; parCoLO.range = 2; parCoLO.EQorLMI = 1
. . . . . . . . . .
. . . . . . . . . .

parCoLO.domain = 0; parCoLO.range = 1; parCoLO.EQorLMI = 2
. . . . . . . . . .
. . . . . . . . . .


%    parCoLO|      cpu time| optimality|         feasibility|    ...
%d   r EQ/LMI  cpuC    cpuS         gap     p.feas    d.feas      ...
0 & 0 & 2 &    0.0 &   16.0  & 3.6e-11 & 9.3e-11 & 8.5e-10      ... \\
0 & 2 & 1 &    0.6 &    1.9  & 2.1e-07  & 1.5e-12 & 5.7e-08     ... \\
0 & 1 & 2 &    0.6 &    2.0  & 1.3e-10 & 2.1e-12 & 8.6e-10      ... \\
% max primal objective value over the  3 cases = +9.069997475115329e+00
% min primal objective value over the  3 cases = +9.069997363367211e+00
% max primal obj. value- min primal obj. value = +1.12e-07
```

For the meaning of each number for the table in the LaTeX format above, see the comments at the end of the program experimentCoLO.m.

As an alternative way of solving the norm minimization problem considered previously, we can call normMinSDP(50,5,10,2009) from experimentCoLO.m directly as

```
>> experimentCoLO('normMinSDP(200,5,10,2009)');
```

Then we will obtain the same result as shown above.

The program experimentCoLO.m can directly process input data in mat format, consisting of $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \mathrm{K})$ or $(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \mathrm{K}, \mathrm{J})$, which represents an equality standard form LOP or a conic-form LOP, respectively. For the equality standard form, experimentCoLO.m sets J.f = the row size of $\boldsymbol{A}$, J.ℓ = [ ], J.q = [ ] and J.s = [ ]. For example, to solve maxG11.mat stored in the directory examples/matFormat/, specify

```
SelectedParaSet = [0,0,1; 1,0,1; 2,0,2];
```

in experimentCoLO.m, and issue

```
>> experimentCoLO('maxG11.mat');
```

Then, the output is:

```
parCoLO.domain = 0; parCoLO.range = 0; parCoLO.EQorLMI = 1
. . . . . . . . . . .
. . . . . . . . . .

parCoLO.domain = 1; parCoLO.range = 0; parCoLO.EQorLMI = 1
. . . . . . . . . . .
. . . . . . . . . .

parCoLO.domain = 2; parCoLO.range = 0; parCoLO.EQorLMI = 2
 . . . . . . . . . . .
. . . . . . . . . .
```

```
% maxG11.mat
%     parCoLO|        cpu time| optimality|        feasibility|      ...
%d   r EQ/LMI  cpuC      cpuS          gap      p.feas    d.feas       ...
0 & 0 & 1 &    0.0 &   313.6  &  3.4e-10 & 9.6e-11 & 2.7e-10   ... \\
1 & 0 & 1 &    2.2 &    33.1  &  5.4e-09 & 1.6e-09 & 0.0e+00   ... \\
2 & 0 & 2 &    1.1 &    17.7  &  2.4e-12 & 6.4e-12 & 0.0e+00   ... \\
% max primal objective value over the  3 cases = -6.291647793563401e+02
% min primal objective value over the  3 cases = -6.291647829972889e+02
% max primal obj. value- min primal obj. value = +3.64e-06
```

To apply sparseCoLO.m directly to the LOP provided by maxG11.mat, type

```
>> parCoLO.domain = 2; parCoLO.range = 0; parCoLO.EQorLMI = 2;
```

to specify the parameters for the d-space conversion method using basis representation, no conversion in the range space and the conversion into an LMI standard form, and

```
>> load 'maxG11.mat';
>> clear J; J.f = size(A,1);
>> [x,y,infoCoLO,cliqueDomain,cliqueRange,LOP] = sparseCoLO(A,b,c,K,J,parCoLO);
```

Then we have a primal approximate optimal solution x (or a dual approximate optimal solution y) of the LOP. It should be noted that the x (or y) involves only partial elements of an (approximate) optimal solution vector because the d-space conversion method using basis representation is applied. In fact, we have

```
>> size(x), nnz(x)
ans = 640000            1
ans = 73600
```

Here all elements with the value 0 are not determined yet because they are not relevant to the objective and constraint functions. To retrieve all the elements of the (approximate) optimal solution vector, type

```
>> [x] = psdCompletion(x,K,cliqueDomain);
the minimum eigenvalue of a completed SDP variable matrix = +2.9e-09
```

Then,

```
>> size(x), nnz(x)
ans = 640000          1
ans = 640000
```

Thus all unspecified elements are completed, thus, the variable matrices become (approximately) positive semidefinite;

```
>> XMat = reshape(x,800,800); min(eig(XMat))
ans =  2.8664e-09
```

See the paper [2, 4, 5] for more details.

4   Parameters and output

Tables 1 and 2 show the parameters and output of sparseCoLO.m.

Parameter	Value	Description
	0	No conversion in the domain space. If K.s = $\emptyset$, take this Value because the domain space conversion is irrelevant.
parCoLO.domain	1	The domain space conversion method using clique trees
	2	The domain space conversion method using basis representation
	0	No conversion in the range space If J.s = $\emptyset$, take this value because the range space conversion is irrelevant.
parCoLO.range	1	The range space conversion method using clique trees
	2	The range space conversion method using matrix decomposition
parCoLO.EQorLMI	1	Conversion into an equality form
	2	Conversion into an LMI form
parCoLO.method	0	Output information on the conversion without solving a given LOP
	2	Solve the resulting LOP by SeDuMi
parCoLO.parSeDuMi		Parameter for SeDuMi, *e.g*, parCoLO.parSeDuMi.fid, parCoLO.parSeDuMi.free, parCoLO.parSeDuMi.eps

Table 1: Input parameters for sparseCoLO.m

5   Concluding remarks

The structure and usage of the Matlab package SparseCoLO have been described using the conic-form LOP, which generalizes the representation of linear, second-order, semidefinite equality/ inequality-constraint problems.

With SparseCoLO, users can experiment any combination of the four conversion methods proposed in [4] by setting the appropriate parameters, depending on the problem. This increases chances to solve large-sized problems that are otherwise hard to solve. In addition, SparseCoLO is versatile in that it can be implemented with any of SDP solvers to reduce the size of the problems.

Output	Description
$\boldsymbol{x}$	An approximate primal optimal solution when parCoLO.method = 2
	[ ] when parCoLO.method = 0
$\boldsymbol{y}$	An approximate dual optimal solution when parCoLO.method = 2
	seqLOPs, which includes information of sequence of LOPs
	from the original LOP through the final LOP to be solved
	when parCoLO.method = 0
infoCoLO.infoSeDuMi	The information from sedumi execution
infoCoLO.CPUdomain	CPU time in seconds for the d-space conversion used
infoCoLO.CPUrange	CPU time in seconds for the r-space conversion used
infoCoLO.CPUEQorLMI	CPU time in seconds for the conversion
	into an equality form or an LMI form
infoCoLO.CPUtotal	The total CPU time in seconds
cliqueDomain	Clique information for the d-space conversion used; if this set is
	nonempty, only the components of sdp variables in $\boldsymbol{x}$ with indices
	contained in some cliques are computed, others are set to zero.
	See the last paragraph of Section 3.
cliqueRange	Clique information for the r-space conversion used; the same
	is applied with replacing $\boldsymbol{x}$ by $\boldsymbol{y}$ as in cliqueDomain
LOP	LOP.A, LOP.b, LOP.c, LOP.K, LOP.J to described
	the final conic form linear optimization problem

Table 2: Output from sparseCoLO.m

Numerical results in Appendix and [4] demonstrate the computational advantage of using SparseCoLO for solving the problems.

Acknowledgments

References

[1] B. Borcher (1999), "CSDP, a C library for semidefinite programming", *Optimization Methods and Software*, **11** 613-623.

[2] M. Fukuda, M. Kojima, K. Murota and K. Nakata (2001) "Exploiting sparsity in semidefinite programming via matrix completion I: General framework," *SIAM Journal on Optimization*, **11** 647-674.

[3] K. Fujisawa, M. Fukuda, K. Kobayashi, M. Kojima, K. Nakata, M. Nakata and M. Yamashita (2008) SDPA (SemiDefinite Programming Algorithm) user's manual — Version 7.05, Research Report B-448, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[4] S. Kim, M. Kojima, M. Mevissen and M. Yamashita (2009) "Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion," Research Report

B-452, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan, January 2009.

[5] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima and K. Murota (2003) "Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results," *Mathematical Programming*, **95** 303-327.

[6] J. F. Sturm (1999) "SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, **11 & 12** 625-653.

[7] K. Toh, M. J. Todd, R. H. Tütüntü (1998) SDPT3 — a MATLAB software package for semidefinite programming, Dept. of Mathematics, National University of Singapore, Singapore.

Appendix: Numerical results

We solve a standard SDP relaxation of the max-cut problem and the norm minimization problems to demonstrate and compare the effectiveness of the conversion methods. Numerical experiments were performed on 2.66 GHz Dual Core Intel Xeon with 12GB memory.

In the description of numerical results, the three numbers in the column parCoLO denote par-CoLO.domain, praCoLO.range and parCoLO.EQorLMI, respectively. "cpuC" and "cpuS" mean CPU time for conversion and executing SeDuMi, respectively, and "sizeA," "nnzA," "nnzS," and "nnzL" denote the size of A, the number of nonzero elements in A, the number of nonzero elements in the Schur complement matrix, and the number of nonzero elements of the Cholesky factor of the Schur complement matrix. And, "noBl" indicates the number of SDP blocks and "maxBl" the maximum size of SDP blocks.

The max-cut problem

Table 3 displays the numerical results for a standard SDP relaxation of the max-cut problem. The problem is formulated as a conic-form LOP (2) with J.s = [ ], so that the r-space conversion is not relevant; hence we fixed parCoLO.range = 0 throughout the numerical experiment. For larger sized-problems, parCoLO= [2 0 2] resulted in the shortest CPU time to solve the problem.

The norm minimization problem

The problem is described in Section 3. In Table 5, we observe that the problems could be solved most efficiently with parCoLO =[0 1 2].

maxCutSDP(1,250,4,2009); a sparse graph, 250 nodes , deg. parameter 4 on [0,1] × [0,1]								
parCoLO	cpuC	cpuS	sizeA	nnzA	nnzS	nnzL	noBl	maxBl
0 0 1	0.0	23.3	250 x 62500	250	62500	31375	1	250
1 0 1	0.5	1.9	686 x 9194	1786	96272	53704	17	46
1 0 2	0.9	12.8	4774 x 9880	10316	2567418	1398610	17	46
2 0 1	0.6	9.5	5924 x 11919	15974	288390	495476	186	17
2 0 2	0.2	2.7	1869 x 10300	10300	151341	77172	186	17
maxCutSDP(1,500,4,2009); a sparse graph, 500 nodes, deg. parameter 4 on [0,1] × [0,1]								
parCoLO	cpuC	cpuS	sizeA	nnzA	nnzS	nnzL	noBl	maxBl
0 0 1	0.0	104.6	500 x 250000	500	250000	125250	1	500
1 0 1	1.1	14.1	2337 x 22329	7206	708245	420848	38	54
1 0 2	1.5	61.2	11575 x 24666	26503	6980331	3915669	38	54
2 0 1	2.1	34.1	14456 x 29502	39561	1142308	2206558	369	23
2 0 2	0.4	7.1	4397 x 25605	25605	624929	319955	369	23
maxCutSDP(1,500,4,2009); a sparse graph, 1000 nodes, deg. parameter 4 on [0,1] × [0,1]								
parCoLO	cpuC	cpuS	sizeA	nnzA	nnzS	nnzL	noBl	maxBl
1 0 1	2.5	46.8	5910 x 67157	19346	3829900	2852489	63	81
2 0 2	1.4	26.5	10506 x 63489	63489	2888172	1520601	724	36

Table 3: Numerical results on the max-cut problem.

normMinSDP(250,10,10,2009)								
parCoLO	cpuC	cpuS	sizeA	nnzA	nnzS	nnzL	noBl	maxBl
0 0 1			out of memory					
0 0 2	0.0	28.2	12 x 67612	50272	142	77	1	260
0 1 2	1.3	5.8	617 x 12024	52472	107227	53922	12	34
0 2 1	0.8	5.9	2806 x 30262	55511	6628026	3626666	250	11
normMinSDP(500,10,10,2009)								
parCoLO	cpuC	cpuS	sizeA	nnzA	nnzS	nnzL	noBl	maxBl
0 0 1			out of memory					
0 0 2	0.0	115.0	12 x 260112	100522	142	77	1	510
0 1 2	4.0	6.4	1112 x 24024	104522	199792	100452	21	34
0 2 1	1.0	12.1	5556 x 60512	111011	25878026	14189291	500	11

Table 4: Numerical results on the norm minimization problem.