ISSN 1342-2804

Research Reports on Mathematical and Computing Sciences

SDPA-C
(SemiDefinite Programming Algorithm – Completion method)
User's Manual — Version 6.10

Katsuki Fujisawa, Mituhiro Fukuda, Masakazu Kojima, Kazuhide Nakata and Makoto Yamashita

August 2004, B–409

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology

SERIES B: Operations Research

B-409 SDPA-C (SemiDefinite Programming Algorithm – Completion method)
User's Manual — Version 6.10
Katsuki Fujisawa[†], Mituhiro Fukuda[♮], Masakazu Kojima[⋆], Kazuhide Nakata[♯] and Makoto Yamashita[‡]
August 2004

Abstract. The SDPA-C (SemiDefinite Programming Algorithm – Completion method) is a software package designed for solving large scale sparse SDPs (semidefinite programming problems). In particular, the SDPA-C solves an SDP quite efficiently in computational time and memory if the aggregated sparsity pattern of the data matrices induces a sparse chordal extension of the aggregate sparsity pattern matrix; if not, the standard version of the SemiDefinite Programming Algorithm, SDPA solves the SDP faster. Using the positive definite matrix completion theory, the SDPA-C stores only sparse matrices and perform matrix computations that take advantages of their sparsity. For theoretical and technical details of the SDPA-C, see the papers Fukuda-Kojima-Murota-Nakata (2000) and Nakata-Fujisawa-Fukuda-Kojima-Murota (2003) listed in the references of this manual.

This manual, the SDPA-C, and the instructions for its installation can be found in the WWW site

http://www.is.titech.ac.jp/~kojima/sdpa/index.html.

Key words Semidefinite Programming, Interior-Point Method, Matrix Completion, Computer Software

† Department of Mathematical Sciences, Tokyo Denki University, Ishizuka, Hatoyama, Saitama 350-0394 Japan. *fujisawa@is-mj.archi.kyoto-u.ac.jp.* Research supported by Grant-in-Aid for Scientific Research on Priority Areas 16016234.

♮ Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 100120-1185. *mituhiro@cims.nyu.edu.*

⋆ Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. *kojima@is.titech.ac.jp* Research supported by Grant-in-Aid for Scientific Research on Priority Areas 16016234.

♯ Department of Industrial Engineering and Management, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. *knakata@me.titech.ac.jp.* Research supported by Grant-in-Aid for Scientific Research on Priority Areas 16016234.

‡ Department of Industrial Engineering and Management, Kanagawa University, 3-27-1 Rokkakubashi, Kanagawa-ku, Yokohama-shi, Kanagawa-ken, 221-8686, Japan. *Makoto.Yamashita@ie.kanagawa-u.ac.jp.* Research supported by Grant-in-Aid for Scientific Research on Priority Areas 16016234.

Preface

Solving large scale SDPs (semidefinite programming problems) is still a challenging research issue in optimization, while many SDPs arising from various fields get larger and larger. Particularly a serious weakness of the primal-dual interior-point method lies in the fact the dual positive semidefinite variable matrix becomes fully dense even when it is applied to a (standard equality form) SDP with sparse data matrices. To overcome this weakness, the SDPA-C incorporates positive definite matrix completion techniques into the Semidefinite Programming Algorithm, the SDPA [7]. To measure the sparsity of an SDP to be solved, the SDPA-C employs the aggregated sparsity pattern of the data matrices. When this aggregated sparsity pattern induces a sparse chordal extension of the aggregated sparsity pattern matrix, the SDPA-C can solve the SDP quite efficiently both in computational time and in memory. See Section 7.3 for the definitions of the aggregated sparsity pattern and its chordal extension (we will call the latter as an extended sparsity pattern). It should be noted that if the aggregated sparsity pattern itself is not sparse then the standard version SDPA works more effectively on the SDP. See [4, 6] for technical details of the SDPA-C and SDP examples which are suitable for the SDPA-C.

The usage and the user interface of the SDPA-C is similar to the SDPA [3]. The input file of the SDPA-C is the same format as the SDPA except that the SDPA-C does not support dense data files. To solve SDP problems described in the example1.dat-s file, type as follows.

```
./sdpa-c example1.dat-s example1.out
```

The output file of the SDPA-C is almost the same as the output file of the SDPA. Many parts of this manual are copies of corresponding parts from the manual of the SDPA [3]. Some differences between the SDPA-C and the SDPA are summarized in Section 7. Also see Section 9 of the SDPA manual [3] for transformation from SDPs to the standard form that the SDPA-C accepts as its input.

We hope that the SDPA-C supports many researches in various fields. We also welcome any suggestions and comments that you may have. When you want to contact us, please send an e-mail to `kojima-sdpa@is.titech.ac.jp` .

Contents

1 Installation

The SDPA-C package is available at the following WWW site:

http://www.is.titech.ac.jp/~kojima/sdpa/index.html

From there, you can download the source code files and the installation manual of the latest version of the SDPA-C.

We assume that the SDPA-C is installed in the subdirectory **sdpa-c** where we can find the following files:

sdpa-c_doc.ps	user's manual.
sdpa-c	executable binary, which solves SDPs.
param.sdpa	parameter file, which contains 10 parameters to control the SDPA-C.
example1.dat-s	sample input file in sparse data format.
example2.dat-s	sample input file in sparse data format.
Makefile	makefile to compile the source files.

Before using the SDPA-C, type **sdpa-c** and make sure that the following message will be displayed.

```
$ ./sdpa-c
SDPA-C start at    Tue Jul 27 19:51:56 2004

*** Please assign data file and output file.***

---- option type 1 ------------
./sdpa-c DataFile OutputFile  [-pt parameters]
parameters = 0 default, 1 aggressive, 2 stable
example1-1: ./sdpa-c example1.dat-s example1.result
example1-2: ./sdpa-c example1.dat-s example1.result -pt 2

---- option type 2 ------------
./sdpa-c [option filename]+
  -ds : data sparse     :: -o  : output     :: -p  : parameter
  -pt : parameters , 0 default, 1 aggressive
                     2 stable
example2-1: ./sdpa-c -o example1.result -ds example1.dat-s
example2-2: ./sdpa-c -ds example1.dat-s -o example2.result -p param.sdpa
example2-3: ./sdpa-c -ds example1.dat-s -o example3.result -pt 2
```

2 Semidefinite Program

2.1 Standard Form SDP and Its Dual

The SDPA-C (Semidefinite Programming Algorithm – Completion method) solves the following standard form semidefinite program and its dual. Here

$$
\text{SDP} \left\{ \begin{array}{lll}
\mathcal{P}: & \text{minimize} & \displaystyle\sum_{i=1}^{m} c_i x_i \\
& \text{subject to} & \boldsymbol{X} = \displaystyle\sum_{i=1}^{m} \boldsymbol{F}_i x_i - \boldsymbol{F}_0, \quad \mathcal{S} \ni \boldsymbol{X} \succeq \boldsymbol{O}. \\
\mathcal{D}: & \text{maximize} & \boldsymbol{F}_0 \bullet \boldsymbol{Y} \\
& \text{subject to} & \boldsymbol{F}_i \bullet \boldsymbol{Y} = c_i \ (i = 1, 2, \ldots, m), \quad \mathcal{S} \ni \boldsymbol{Y} \succeq \boldsymbol{O}.
\end{array} \right.
$$

$\mathcal{S}$: the set of $n \times n$ real symmetric matrices.

$\boldsymbol{F}_i \in \mathcal{S}$ $(i = 0, 1, 2, \ldots, m)$: constraint matrices.

$\boldsymbol{O} \in \mathcal{S}$: the zero matrix.

$\boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ \cdot \\ c_m \end{pmatrix} \in R^m$: a cost vector, $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ x_m \end{pmatrix} \in R^m$: a variable vector,

$\boldsymbol{X} \in \mathcal{S}$, $\boldsymbol{Y} \in \mathcal{S}$: variable matrices,

$\boldsymbol{U} \bullet \boldsymbol{V}$: the inner product of $\boldsymbol{U}$, $\boldsymbol{V} \in \mathcal{S}$, *i.e.*, $\displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} U_{ij} V_{ij}$

$\boldsymbol{U} \succeq \boldsymbol{O}$, $\Longleftrightarrow$ $\boldsymbol{U}$ is a positive semidefinite symmetric matrix.

Throughout this manual, we denote the primal-dual pair of $\mathcal{P}$ and $\mathcal{D}$ by SDP. The SDP is determined by m, n, $\boldsymbol{c} \in R^m$, and $\boldsymbol{F}_i \in \mathcal{S}$ $(i = 0, 1, 2, \ldots, m)$. When $(\boldsymbol{x}, \boldsymbol{X})$ is a feasible solution (or a minimum solution, resp.) of the primal problem $\mathcal{P}$ and $\boldsymbol{Y}$ is a feasible solution (or a maximum solution, resp.), we call $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ a feasible solution (or an optimal solution, resp.) of the SDP.

We assume:

Condition 1.1. $\{\boldsymbol{F}_i \ : \ i = 1, 2, \ldots, m\} \subset \mathcal{S}$ is linearly independent.

If the SDP does not satisfy this assumption, it might cause some trouble (numerical instability) that would abnormally stop the execution of the SDPA-C.

If we deal with a different primal-dual pair of $\mathcal{P}$ and $\mathcal{D}$ of the form

$$
\text{SDP2} \left\{ \begin{array}{lll}
\mathcal{P}: & \text{minimize} & \boldsymbol{A}_0 \bullet \boldsymbol{X} \\
& \text{subject to} & \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \ (i = 1, 2, \ldots, m), \quad \mathcal{S} \ni \boldsymbol{X} \succeq \boldsymbol{O}. \\
\mathcal{D}: & \text{maximize} & \displaystyle\sum_{i=1}^{m} b_i y_i \\
& \text{subject to} & \displaystyle\sum_{i=1}^{m} \boldsymbol{A}_i y_i + \boldsymbol{Z} = \boldsymbol{A}_0, \quad \mathcal{S} \ni \boldsymbol{Z} \succeq \boldsymbol{O}.
\end{array} \right.
$$

we can easily transform from the SDP2 into the SDP as follows:

$$\begin{array}{rcl} -\boldsymbol{A}_i \ (i=0,\ldots,m) & \longrightarrow & \boldsymbol{F}_i \ (i=0,\ldots,m) \\ -b_i \ (i=1,\ldots,m) & \longrightarrow & c_i \ (i=1,\ldots,m) \\ \boldsymbol{X} & \longrightarrow & \boldsymbol{Y} \\ \boldsymbol{y} & \longrightarrow & \boldsymbol{x} \\ \boldsymbol{Z} & \longrightarrow & \boldsymbol{X} \end{array}$$

2.2 Example 1

$$\left.\begin{array}{lll} \mathcal{P}: & \text{minimize} & 48y_1 - 8y_2 + 20y_3 \\ & \text{subject to} & \boldsymbol{X} = \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix} y_1 + \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix} y_2 + \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix} y_3 - \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix} \\ & & \boldsymbol{X} \succeq \boldsymbol{O}. \\ \mathcal{D}: & \text{maximize} & \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix} \bullet \boldsymbol{Y} \\ & \text{subject to} & \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix} \bullet \boldsymbol{Y} = 48, \ \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix} \bullet \boldsymbol{Y} = -8 \\ & & \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix} \bullet \boldsymbol{Y} = 20, \ \boldsymbol{Y} \succeq \boldsymbol{O}. \end{array}\right\}$$

Here

$$\begin{aligned} m &= 3, \ n = 2, \ \boldsymbol{c} = \begin{pmatrix} 48 \\ -8 \\ 20 \end{pmatrix}, \ \boldsymbol{F}_0 = \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix}, \\ \boldsymbol{F}_1 &= \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix}, \ \boldsymbol{F}_2 = \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix}, \ \boldsymbol{F}_3 = \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix}. \end{aligned}$$

The data of this problem is contained in the file "example1.dat-s".

2.3 Example 2

$$\begin{aligned} m &= 5, \ n = 7, \ \boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{pmatrix} = \begin{pmatrix} 1.1 \\ -10 \\ 6.6 \\ 19 \\ 4.1 \end{pmatrix}, \\ \boldsymbol{F}_0 &= \begin{pmatrix} -1.4 & -3.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -3.2 & -28 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 15 & -12 & 2.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & -12 & 16 & -3.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 2.1 & -3.8 & 15 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.8 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4.0 \end{pmatrix}, \end{aligned}$$

$$
\boldsymbol{F}_1 = \begin{pmatrix} 0.5 & 5.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 5.2 & -5.3 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 7.8 & -2.4 & 6.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -2.4 & 4.2 & 6.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 6.0 & 6.5 & 2.1 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -4.5 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -3.5 \end{pmatrix}
$$

$$
\vdots
$$

$$
\boldsymbol{F}_5 = \begin{pmatrix} -6.5 & -5.4 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ -5.4 & -6.6 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 6.7 & -7.2 & -3.6 & 0.0 & 0.0 \\ 0.0 & 0.0 & -7.2 & 7.3 & -3.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -3.6 & -3.0 & -1.4 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 6.1 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & -1.5 \end{pmatrix}.
$$

As shown in this example, the SDPA-C handles block diagonal matrices. The data of this example is contained in the file "example2.dat-s".

3 Files Necessary to Execute the SDPA-C

We need the following files to execute the SDPA-C

- "**sdpa-c**" — An executable binary for solving an SDP.
- "input data file" — Any file name with the postfix "**.dat-s**" is possible; for example, "problem.dat-s" and "example.dat-s" are legitimate names for input files. The SDPA-C supports ONLY a sparse input data file with the postfix "**.dat-s**". See Section 4 for details.
- "**param.sdpa**" — A file describing the parameters used in the "sdpa-c". See Section 5 for details. "**param.sdpa**" of the SDPA-C and the SDPA are identical.
- "output file" — Any file name except "**sdpa-c**" and "**param.sdpa**". For example, "problem.1" and "example.out" are legitimate names for output files. See Section 6 for more details.

The files "example1.dat-s" (see Section 4.1) and "example2.dat-s" (see Section 4.2) contain the input date of Example 1 and Example 2, respectively, which we have stated in the previous section. To solve Example 1, type

$ ./sdpa-c example1.dat-s example1.out

Here "example1.out" denotes an "output file" in which the SDPA-C stores computational results such as an approximate optimal solution, and an approximate optimal value of Example 1. Similarly, we can solve Example 2 by using the "sdpa-c".

4 Input Date File

The SDPA-C supports ONLY the sparse data format which gives us a compact description of the constraint matrices.

A sparse input data file must have a name with the postfix ".dat-s"; for example, "problem.dat-s" and "example.dat-s" are legitimate names for sparse input data files.

4.1 "example1.dat-s" — Input Data File of Example 1

```
"Example 1: mDim = 3, nBLOCK = 1, {2}"
   3  =  mDIM
   1  =  nBLOCK
   2  = bLOCKsTRUCT
{48, -8, 20}
0 1 1 1 -11
0 1 2 2 23
1 1 1 1 10
1 1 1 2 4
2 1 2 2 -8
3 1 1 2 -8
3 1 2 2 -2
```

4.2 "example2.dat-s" — Input Data File of Example 2

```
*Example 2:
*mDim = 5, nBLOCK = 3, {2,3,-2}
   5  =  mDIM
   3  =  nBLOCK
   2    3   -2   = bLOCKsTRUCT
1.1 -10 6.6 19  4.1
0 1 1 1  -1.4
0 1 1 2  -3.2
0 1 2 2 -28.0
0 2 1 1  15.0
0 2 1 2 -12.0
0 2 1 3   2.1
0 2 2 2  16.0
0 2 2 3  -3.8
0 2 3 3  15.0
0 3 1 1   1.8
0 3 2 2  -4.0
1 1 1 1   0.5
1 1 1 2   5.2
1 1 2 2  -5.3
1 2 1 1   7.8
```

•

$\vdots$

```
5 2 1 2  -7.2
5 2 1 3  -3.6
5 2 2 2   7.3
5 2 2 3  -3.0
5 2 3 3  -1.4
5 3 1 1   6.1
5 3 2 2  -1.5
```

4.3 Format of the Input Data File

In general, the structure of an input data file is as follows:

Title and Comments
m — the number of the primal variables x_i's
nBLOCK — the number of blocks
bLOCKsTRUCT — the block structure vector
$\boldsymbol{c}$
$\boldsymbol{F}_0$
$\boldsymbol{F}_1$
.
.
$\boldsymbol{F}_m$

In Sections 4.4 through 4.8 , we explain each item of the input data file in details.

4.4 Title and Comments

On the top of the input data file, we can write a single or multiple lines of Title and Comments. Each line of Title and Comments must begin with `"` or `*` and consist of no more than 75 letters; for example

```
"Example 1: mDim = 3, nBLOCK = 1, {2}"
```

in the file "example1.dat-s", and

```
*Example 2:
*mDim = 5, nBLOCK = 3, {2,3,-2}
```

in the file "example2.dat-s". The SDPA-C displays Title and Comments when it starts. Title and Comments can be omitted.

4.5 The Number of Primal Variables

We write the number m of the primal variables in a line following the line(s) of Title and Comments in the input data file. All the letters after m through the end of the line are neglected. We have

```
3 = mDIM
```

in the file "example1.dat-s", and

```
5 = mDIM
```

in the file "example2.dat-s". In either case, the letters "= mDIM" are neglected.

4.6 The Number of Blocks and the Block Structure Vector

The SDPA-C handles block diagonal matrices as we have seen in Section 2.3. We can express a common matrix data structure for the constraint matrices $\boldsymbol{F}_0,\ \boldsymbol{F}_1,\ \ldots,\ \boldsymbol{F}_m$ in terms of the number of blocks, denoted by nBLOCK. For example, if we deal with a block diagonal matrix $\boldsymbol{F}$ of the form

$$\left.\begin{array}{rcl} \boldsymbol{F} & = & \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{B}_2 & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \cdot & \cdot & \cdot & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{B}_\ell \end{pmatrix}, \\ \boldsymbol{B}_i & : & \text{a } p_i \times p_i \text{ symmetric matrix } \ (i = 1, 2, \ldots, \ell), \end{array}\right\} \tag{1}$$

we define the number nBLOCK of blocks and the block structure vector bLOCKsTRUCTURE as follows:

$$\begin{array}{rcl} \text{nBLOCK} & = & \ell, \\ \text{bLOCKsTRUCT} & = & (\beta_1,\ \beta_2,\ \ldots,\ \beta_\ell), \\ \beta_i & = & \left\{ \begin{array}{rl} p_i & \text{if } \boldsymbol{B}_i \text{ is a symmetric matrix,} \\ -p_i & \text{if } \boldsymbol{B}_i \text{ is a diagonal matrix.} \end{array} \right. \end{array}$$

For example, if $\boldsymbol{F}$ is of the form

$$\begin{pmatrix} 1 & 2 & 3 & 0 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 & 0 \\ 3 & 5 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{pmatrix}, \tag{2}$$

we have

$$\text{nBLOCK} = 3 \quad \text{and} \quad \text{bLOCKsTRUCT} = \ \ (3,\ 2,\ -2)$$

If

$$\boldsymbol{F} = \begin{pmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{pmatrix}, \quad \text{where } \star \text{ denotes a real number,}$$

is a usual symmetric matrix with no block diagonal structure, we define

$$\text{nBLOCK} = 1 \quad \text{and} \quad \text{bLOCKsTRUCT} = 3$$

We separately write each of nBLOCK and bLOCKsTRUCT in one line. Any letter after either of nBLOCK and bLOCKsTRUCT through the end of the line is neglected. In addition to blank letter(s), and the tab code(s), we can use the letters

, ( ) { }

to separate elements of the block structure vector bLOCKsTRUCT. We have

```
  1  =  nBLOCK
  2  = bLOCKsTRUCT
```

in Example 1 (see the file "example1.dat-s" in Section 4.1), and

```
  3  =  nBLOCK
  2    3   -2   = bLOCKsTRUCT
```

in Example 2 (see the file "example2.dat-s" in Section 4.2). In either case, the letters "= nBLOCK" and "= bLOCKsTRUCT" are neglected.

4.7 Constant Vector

We write all the elements c_1, c_2, …, c_m of the cost vector $\boldsymbol{c}$. In addition to blank letter(s) and tab code(s), we can use the letters

, ( ) { }

to separate elements of the vector $\boldsymbol{c}$. We have

```
{48, -8, 20}
```

in Example 1 (see the file "example1.dat-s" in Section 4.1), and

```
{1.1, -10, 6.6, 19, 4.1}
```

in Example 2 (see the file "example2.dat-s" in Section 4.2).

4.8 Constraint Matrices

We describe the constraint matrices $\boldsymbol{F}_0$, $\boldsymbol{F}_1$, …, $\boldsymbol{F}_m$ according to the format we defined by nBLOCK and bLOCKsTRUCT stated in Section 4.6. In Example 1 with nBLOCK = 1 and bLOCKsTRUCT = 2, we have

```
0 1 1 1 -11
0 1 2 2 23
1 1 1 1 10
1 1 1 2 4
2 1 2 2 -8
3 1 1 2 -8
3 1 2 2 -2
```

See the file “example1.dat-s” in Section 4.1.

Each line describes a single element of the constraint matrix $\boldsymbol{F}_i$; the 1st line “0 1 1 1 -11” means that the $(1, 1)$th element of the 1st block of the matrix $\boldsymbol{F}_0$ is -11, and the 7th line “3 1 1 2 -8” means that the $(1, 2)$th element of the 1st block of the matrix $\boldsymbol{F}_3$ is -8.

In general, the structure of a sparse input data file is as follows:

Title and Comments
m — the number of the primal variables x_i’s
nBLOCK — the number of blocks
bLOCKsTRUCT — the block structure vector
$\boldsymbol{c}$
$k_1\ b_1\ i_1\ j_1\ v_1$
$k_2\ b_2\ i_2\ j_2\ v_2$
$\ldots$
$k_p\ b_p\ i_p\ j_p\ v_p$
$\ldots$
$k_q\ b_q\ i_q\ j_q\ v_q$

Here $k_p \in \{0, 1, \ldots, m\}$, $b_p \in \{1, 2, \ldots, \text{nBLOCK}\}$, $1 \leq i_p \leq j_p$ and $v_p \in R$. Each line “k_p, b_p, i_p, j_p, v_p” means that the value of the (i_p, j_p)th element of the b_pth block of the constant matrix $\boldsymbol{F}_{k_p}$ is v_p. If the b_pth block is an $\ell \times \ell$ symmetric (non-diagonal) matrix then (i_p, j_p) must satisfy $1 \leq i_p \leq j_p \leq \ell$; hence only nonzero elements in the **upper triangular** part of the b_pth block are described in the file. If the b_pth block is an $\ell \times \ell$ diagonal matrix then (i_p, j_p) must satisfy $1 \leq i_p = j_p \leq \ell$.

5 Parameter File

First we show the default parameter file “param.sdpa” below.

```
40      int maxIteration;
1.0E-7  double 0.0 < epsilonStar;
1.0E2   double 0.0 < lambdaStar;
2.0     double 1.0 < omegaStar;
-1.0E5  double lowerBound;
1.0E5   double upperBound;
0.1     double 0.0 <= betaStar <  1.0;
0.2     double 0.0 <= betaBar  <  1.0, betaStar <= betaBar;
0.9     double 0.0 < gammaStar  <  1.0;
1.0E-7  double 0.0 < epsilonDash;
```

The file “param.sdpa” needs to have these 10 lines which respectively presents 10 parameters. Each line of the file “param.sdpa” contains one of the 10 parameters followed by any comment. When the SDPA-C reads the file “param.sdpa”, it neglects the comments.

- maxIteration — The maximum number of iterations. The SDPA-C stops when the iteration exceeds the maxIteration.

- epsilonStar, epsilonDash — The accuracy of an approximate optimal solution of the SDP. When the current iterate $(\boldsymbol{x}^k, \boldsymbol{X}^k, \boldsymbol{Y}^k)$ satisfies the inequalities

$$
\begin{array}{rcl}
\text{epsilonDash} & \geq & \max\left\{\left|[X^k - \sum_{i=1}^{m} \boldsymbol{F}_i x_i^k + \boldsymbol{F}_0]_{pq}\right| \ : \ p, q = 1, 2, \ldots, n\right\}, \\
\text{epsilonDash} & \geq & \max\left\{\left|\boldsymbol{F}_i \bullet \boldsymbol{Y}^k - c_i\right| \ : i = 1, 2, \ldots, m\right\}, \\
\text{epsilonStar} & \geq & \dfrac{|\sum_{i=1}^m c_i x_i^k - \boldsymbol{F}_0 \bullet \boldsymbol{Y}^k|}{\max\left\{(|\sum_{i=1}^m c_i x_i^k| + |\boldsymbol{F}_0 \bullet \boldsymbol{Y}^k|)/2.0, \ 1.0\right\}} \\
 & = & \dfrac{|\text{the primal objective value} - \text{the dual objective value}|}{\max\{(|\text{the primal objective value}| + |\text{the dual objective value}|)/2.0, \ 1.0\}},
\end{array}
$$

  the SDPA-C stops. Too small epsilonStar and epsilonDash may cause a numerical instability. A reasonable choice is epsilonStar $\geq 1.0E-7$.

- lambdaStar — This parameter determines an initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$ such that

$$\boldsymbol{x}^0 = \boldsymbol{0}, \ \boldsymbol{X}^0 = \text{lambdaStar} \ \times \ \boldsymbol{I}, \ \boldsymbol{Y}^0 = \text{lambdaStar} \ \times \ \boldsymbol{I}.$$

  Here $\boldsymbol{I}$ denotes the identity matrix. It is desirable to choose an initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$ having the same order of magnitude as an optimal solution $(\boldsymbol{x}^*, \boldsymbol{X}^*, \boldsymbol{Y}^*)$ of the SDP. In general, however, choosing such a lambdaStar is difficult. If there is no information on the magnitude of an optimal solution $(\boldsymbol{x}^*, \boldsymbol{X}^*, \boldsymbol{Y}^*)$ of the SDP, we strongly recommend to take a sufficiently large lambdaStar such that

$$\boldsymbol{X}^* \preceq \text{lambdaStar} \ \times \ \boldsymbol{I} \quad \text{and} \quad \boldsymbol{Y}^* \preceq \text{lambdaStar} \ \times \ \boldsymbol{I}.$$

- omegaStar — This parameter determines the region in which the SDPA-C searches an optimal solution. For the primal problem $\mathcal{P}$, the SDPA-C searches a minimum solution $(\boldsymbol{x}, \boldsymbol{X})$ within the region

$$\boldsymbol{O} \preceq \boldsymbol{X} \preceq \text{omegaStar} \ \times \ \boldsymbol{X}^0 = \text{omegaStar} \times \text{lambdaStar} \ \times \ \boldsymbol{I},$$

  and stops the iteration if it detects that the primal problem $\mathcal{P}$ has no minimum solution in this region. For the dual problem $\mathcal{D}$, the SDPA-C searches a maximum solution $\boldsymbol{Y}$ within the region

$$\boldsymbol{O} \preceq \boldsymbol{Y} \preceq \text{omegaStar} \ \times \ \boldsymbol{Y}^0 = \text{omegaStar} \times \text{lambdaStar} \ \times \ \boldsymbol{I},$$

  and stops the iteration if it detects that the dual problem $\mathcal{D}$ has no maximum solution in this region. Again we recommend to take a larger lambdaStar and a smaller omegaStar > 1.

- lowerBound — Lower bound of the minimum objective value of the primal problem $\mathcal{P}$. When the SDPA-C generates a primal feasible solution $(\boldsymbol{x}^k, \boldsymbol{X}^k)$ whose objective value $\sum_{i=1}^{m} c_i x_i^k$ gets smaller than the lowerBound, the SDPA-C stops the iteration; the primal problem $\mathcal{P}$ is likely to be unbounded and the dual problem $\mathcal{D}$ is likely to be infeasible if the lowerBound is sufficiently small.

- upperBound — Upper bound of the maximum objective value of the dual problem $\mathcal{D}$. When the SDPA-C generates a dual feasible solution $\boldsymbol{Y}^k$ whose objective value $\boldsymbol{F}_0 \bullet \boldsymbol{Y}^k$ gets larger than the upperBound, the SDPA-C stops the iteration; the dual problem $\mathcal{D}$ is likely to be unbounded and the primal problem $\mathcal{P}$ is likely to be infeasible if the upperBound is sufficiently large.

- betaStar — A parameter controlling the search direction when $(\boldsymbol{x}^k, \boldsymbol{X}^k, \boldsymbol{Y}^k)$ is feasible. As we take a smaller betaStar > 0.0, the search direction can get closer to the affine scaling direction without centering.

- betaBar — A parameter controlling the search direction when $(\boldsymbol{x}^k, \boldsymbol{X}^k, \boldsymbol{Y}^k)$ is infeasible. As we take a smaller betaBar > 0.0, the search direction can get closer to the affine scaling direction without centering. The value of betaBar must be not less than the value of betaStar; $0 \leq$ betaStar $\leq$ betaBar.

- gammaStar — A reduction factor for the primal and dual step lengths; $0.0 <$ gammaStar $<$ 1.0.

We may encounter some numerical difficulty during the execution of the SDPA-C with the default parameter file "param.sdpa", and/or we may want to solve several easy SDPs with similar data more quickly. In such a case, we need to adjust some of the default parameters, betaStar, betaBar, and gammaStar. We present below two sets of those parameters. The one is the set "Stable_but_Slow" for difficult SDPs, and the other is the set "Unstable_but_Fast" for easy SDPs.

Stable_but_Slow

```
0.10 double 0.0 <= betaStar <  1.0;
0.20 double 0.0 <= betaBar  <  1.0, betaStar <= betaBar;
0.90 double 0.0 < gammaStar  <  1.0;
```

Unstable_but_Fast

```
0.01 double 0.0 <= betaStar <  1.0;
0.02 double 0.0 <= betaBar  <  1.0, betaStar <= betaBar;
0.98 double 0.0 < gammaStar  <  1.0;
```

Besides these parameters, the value of the parameter lambdaStar, which determines an initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$, affects the computational efficiency and the numerical stability. Usually a larger lambdaStar is safe although the SDPA-C may consume a few more iterations.

6 Output

6.1 Execution of the SDPA-C

To execute the SDPA-C, we specify and type the names of three files, "sdpa-c", an "input data file" and an "output file" as follows.

% sdpa-c "input data file" "output file"

To solve Example 1, type:

$ ./sdpa-c example1.dat-s example1.out

6.2 Output on the Display

The SDPA-C shows some information on the display. In the case of Example 1, we have

```
SDPA-C start at    Tue Jul 27 19:59:39 2004
data      is example1.dat-s : sparse
parameter is ./param.sdpa
out       is out

aggregate sparsity pattern :                                4 elements
extended sparsity pattern  :
        METIS4.0.1 (multilevel nested dissection)           4 elements
        Spooles2.2 (minimum degree)                         4 elements
        Spooles2.2 (generalized nested dissection)          4 elements
        Spooles2.2 (multisection)                           4 elements
        Spooles2.2 (best of ND and MS)                      4 elements
        Selecting ..... METIS4.0.1 (multilevel nested dissection)
dense matrix               :                                4 elements
-----------------------------------------------------------------------
   mu      thetaP  thetaD  objP      objD      alphaP  alphaD  beta
 0 1.0e+04 1.0e+00 1.0e+00 -0.00e+00 +1.20e+03 1.0e+00 9.1e-01 3.00e-01
 1 1.6e+03 2.9e-17 9.4e-02 +8.16e+02 +7.51e+01 5.6e+00 9.6e-01 3.00e-01
 2 2.1e+02 1.7e-16 3.6e-03 +2.59e+02 -3.74e+01 2.0e+00 1.0e+00 3.00e-01
 3 3.8e+01 1.4e-16 1.5e-17 +3.35e+01 -4.19e+01 9.7e-01 9.7e-01 1.00e-01
 4 4.8e+00 1.7e-16 1.5e-17 -3.22e+01 -4.19e+01 9.9e-01 9.0e+01 1.00e-01
 5 5.3e-01 1.6e-16 6.6e-16 -4.08e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 6 5.3e-02 1.8e-16 7.5e-18 -4.18e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 7 5.3e-03 1.6e-16 1.5e-17 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
 8 5.3e-04 1.5e-16 1.5e-17 -4.19e+01 -4.19e+01 1.0e+00 9.0e+01 1.00e-01
 9 5.3e-05 1.4e-16 9.9e-16 -4.19e+01 -4.19e+01 1.0e-00 1.0e-00 1.00e-01
10 5.3e-06 1.3e-16 7.5e-18 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01
11 5.3e-07 1.4e-16 1.2e-15 -4.19e+01 -4.19e+01 1.0e-00 9.0e+01 1.00e-01

phase.value = pdOPT
   Iteration = 11
          mu = 5.3127653470865201e-07
relative gap = 2.5359240205895713e-08
         gap = 1.0625530694173040e-06
      digits = 7.5958637625935399e+00
objValPrimal = -4.1899998937446959e+01
objValDual   = -4.1899999999999110e+01
p.feas.error = 1.6653345369377348e-14
d.feas.error = 1.1226575225009583e-12
total time   = 0.000
  main loop time = 0.000000
      total time = 0.000000
file   read time = 0.000000
```

- aggregate sparsity pattern — The number of nonzero elements in the aggregated sparsity pattern over all data matrices of a given SDP. See 7.3 for details.

- extended sparsity pattern — The number of nonzero elements in a chordal extension of the aggregated sparsity pattern. To get more sparsity in the extension, the SDPA-C utilizes five heuristic ordering and chooses the best ordering. See 7.3 for details.

- dense matrix — The number of (virtual) nonzero elements of matrix variables $\boldsymbol{X}$ or $\boldsymbol{Y}$.

- mu — The average complementarity $\boldsymbol{X}^k \bullet \boldsymbol{Y}^k/n$ (an optimality measure). When both $\mathcal{P}$ and $\mathcal{D}$ get feasible, the relation

$$\begin{aligned} \text{mu} &= \left(\sum_{i=1}^{m} c_i x_i^k - \boldsymbol{F}_0 \bullet \boldsymbol{Y}^k \right) /n \\ &= \frac{\text{the primal objective function - the dual objective function}}{n} \end{aligned}$$

  holds.

- thetaP — The SDPA-C starts with thetaP = 0.0 if the initial point $(\boldsymbol{x}^0, \boldsymbol{X}^0)$ of the primal problem $\mathcal{P}$ is feasible, and thetaP = 1.0 otherwise; hence it usually starts with thetaP = 1.0. In the latter case, the thetaP at the kth iteration is given by

$$\text{thetaP} = \frac{\max\left\{ \left| [\sum_{i=1}^{m} \boldsymbol{F}_i x_i^k + \boldsymbol{X}^k - \boldsymbol{F}_0]_{p,q} \right| \ : \ p,q = 1,2,\ldots,n \right\}}{\max\left\{ \left| [\sum_{i=1}^{m} \boldsymbol{F}_i x_i^0 + \boldsymbol{X}^0 - \boldsymbol{F}_0]_{p,q} \right| \ : \ p,q = 1,2,\ldots,n \right\}};$$

  The thetaP is theoretically monotone non-increasing, and when it gets 0.0, we obtain a primal feasible solution $(\boldsymbol{x}^k, \boldsymbol{X}^k)$. In the example above, we obtained a primal feasible solution in the 1st iteration.

- thetaD — The SDPA-C starts with thetaD = 0.0 if the initial point $\boldsymbol{Y}^0$ of the dual problem $\mathcal{D}$ is feasible, and thetaD = 1.0 otherwise; hence it usually starts with thetaD = 1.0. In the latter case, the thetaD at the kth iteration is given by

$$\text{thetaD} = \frac{\max\left\{ \left| \boldsymbol{F}_i \bullet \boldsymbol{Y}^k - c_i \right| \ : i = 1,2,\ldots,m \right\}}{\max\left\{ \left| \boldsymbol{F}_i \bullet \boldsymbol{Y}^0 - c_i \right| \ : i = 1,2,\ldots,m \right\}};$$

  The thetaD is theoretically monotone non-increasing, and when it gets 0.0, we obtain a dual feasible solution $\boldsymbol{Y}^k$. In the example above, we obtained a dual feasible solution in the 3rd iteration.

- objP — The primal objective function value.

- objD — The dual objective function value.

- alphaP — The primal step length.

- alphaD — The dual step length.

- beta — The search direction parameter.

- phase.value — The status when the iteration stops, taking one of the values pdOPT, noINFO, pFEAS, dFEAS, pdFEAS, pdINF, pFEAS_dINF, pINF_dFEAS, pUNBD and dUNBD.

  pdOPT : The normal termination yielding both primal and dual approximate optimal solutions.

noINFO : The iteration has exceeded the maxIteration and stopped with no information on the primal feasibility and the dual feasibility.

pFEAS : The primal problem $\mathcal{P}$ got feasible but the iteration has exceeded the maxIteration and stopped.

dFEAS : The dual problem $\mathcal{D}$ got feasible but the iteration has exceeded the maxIteration and stopped.

pdFEAS : Both primal problem $\mathcal{P}$ and the dual problem $\mathcal{D}$ got feasible, but the iteration has exceeded the maxIteration and stopped.

pdINF : At least one of the primal problem $\mathcal{P}$ or the dual problem $\mathcal{D}$ is expected to be infeasible. More precisely, there is no optimal solution $(\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y})$ of the SDP such that

$$\begin{aligned}
&\boldsymbol{O} \preceq \boldsymbol{X} \preceq \text{omegaStar} \ \times \ \boldsymbol{X}^0, \\
&\boldsymbol{O} \preceq \boldsymbol{Y} \preceq \text{omegaStar} \ \times \ \boldsymbol{Y}^0, \\
&\sum_{i=1}^{m} c_i x_i = \boldsymbol{F}_0 \bullet \boldsymbol{Y}.
\end{aligned}$$

pFEAS_dINF : The primal problem $\mathcal{P}$ has become feasible but the dual problem is expected to be infeasible. More precisely, there is no dual feasible solution $\boldsymbol{Y}$ such that

$$\boldsymbol{O} \preceq \boldsymbol{Y} \preceq \text{omegaStar} \ \times \ \boldsymbol{Y}^0 = \text{lambdaStar} \times \text{omegaStar} \ \times \ \boldsymbol{I}.$$

pINF_dFEAS : The dual problem $\mathcal{D}$ has become feasible but the primal problem is expected to be infeasible. More precisely, there is no feasible solution $(\boldsymbol{x}, \boldsymbol{X})$ such that

$$\boldsymbol{O} \preceq \boldsymbol{X} \preceq \text{omegaStar} \ \times \ \boldsymbol{X}^0 = \text{lambdaStar} \times \text{omegaStar} \ \times \ \boldsymbol{I}.$$

pUNBD : The primal problem is expected to be unbounded. More precisely, the SDPA-C has stopped generating a primal feasible solution $(\boldsymbol{x}^k, \boldsymbol{X}^k)$ such that

$$\text{objP} = \sum_{i=1}^{m} c_i x_i^k < \ \text{lowerBound}.$$

dUNBD : The dual problem is expected to be unbounded. More precisely, the SDPA-C has stopped generating a dual feasible solution $\boldsymbol{Y}^k$ such that

$$\text{objD} = \boldsymbol{F}_0 \bullet \boldsymbol{Y}^k > \ \text{upperBound}.$$

- Iteration — The iteration number which the SDPA-C needs to terminate.
- relative gap — The relative gap is defined as

$$\frac{|\text{objP} - \text{objD}|}{\max\left\{1.0, \ (|\text{objP}| + |\text{objD}|)/2\right\}}.$$

  This value is compared with **epsilonStar** (see Section 5).

- gap — The gap is defined as mu $\times\, n$.
- digits — This value indicates how objP and objD resemble by the following definition.

$$\begin{aligned}
\text{digits} &= -\log_{10} \frac{|\text{objP} - \text{objD}|}{(|\text{objP}| + |\text{objD}|)/2.0} \\
&= -\log_{10} \frac{|\sum_{i=1}^{m} c_i {x_i}^* - \boldsymbol{F}_0 \bullet \boldsymbol{Y}|}{(|\sum_{i=1}^{m} c_i {x_i}^*| + |\boldsymbol{F}_0 \bullet \boldsymbol{Y}|)/2.0}
\end{aligned}$$

- objValPrimal — The primal objective function value.

$$\text{objValPrimal} = \sum_{i=1}^{m} c_i x_i.$$

- objValDual — The dual objective function value.

$$\text{objValD} = \boldsymbol{F}_0 \bullet \boldsymbol{Y}.$$

- p.feas.error — This value is the primal infeasibily in the last iteration,

$$\text{p.feas.error} = \max \left\{ \left| [\sum_{i=1}^{m} \boldsymbol{F}_i x_i + \boldsymbol{X} - \boldsymbol{F}_0]_{p,q} \right| \ : \ p, q = 1, 2, \ldots, n \right\}$$

  This value is compared with **epsilonDash** (see Section 5). Even if primal is feasible, this value may not be 0 due to numerical errors.

- d.feas.error — This value is the dual infeasibily in the last iteration,

$$\text{d.feas.error} = \max \left\{ |\boldsymbol{F}_i \bullet \boldsymbol{Y} - c_i| \ : i = 1, 2, \ldots, m \right\}.$$

  This value is compared with **epsilonDash** (see Section 5). Even if dual is feasible, this value may not be 0 due to numerical errors.

- total time — Indicates the time the SDPA-C needs to execute all subroutines.
- main loop time — Indicates the time the SDPA-C needs between the first iteration and the last iteration.
- file read time — Indicates the time the SDPA-C needs to read from the input file and store the data in memory.

6.3 Output to a File

We show the content of the file "example2.out" on which the SDPA-C has written the computational results of Example 2.

```
*Example 2:
*mDim = 5, nBLOCK = 3, {2,3,-2}
data      is example2.dat-s
parameter is ./param.sdpa
out       is out
aggregate sparsity pattern :                                15 elements
extended sparsity pattern  :
        METIS4.0.1 (multilevel nested dissection)           15 elements
        Spooles2.2 (minimum degree)                         15 elements
        Spooles2.2 (generalized nested dissection)          15 elements
        Spooles2.2 (multisection)                           15 elements
        Spooles2.2 (best of ND and MS)                      15 elements
        Selecting ..... METIS4.0.1 (multilevel nested dissection)
dense matrix               :                                49 elements
```

```
----------------------------------------------------------------------
   mu      thetaP  thetaD  objP      objD      alphaP  alphaD  beta
 0 1.0e+04 1.0e+00 1.0e+00 -0.00e+00 +1.44e+03 7.5e-01 5.7e-01 3.00e-01
 1 4.1e+03 2.5e-01 4.3e-01 +3.42e+02 +4.60e+02 1.0e+00 6.3e-01 3.00e-01
 2 1.7e+03 2.5e-16 1.6e-01 +6.87e+02 +3.15e+01 8.7e-01 8.7e-01 3.00e-01
 3 3.6e+02 4.9e-16 2.0e-02 +8.32e+02 +9.88e-02 6.6e+00 1.0e+00 3.00e-01
 4 9.5e+01 2.2e-16 1.1e-17 +6.65e+02 +6.91e-01 9.2e-01 3.5e+00 1.00e-01
 5 1.4e+01 4.2e-16 1.4e-16 +1.22e+02 +2.07e+01 7.7e-01 1.2e+00 1.00e-01
 6 4.2e+00 3.8e-16 1.1e-16 +5.56e+01 +2.62e+01 7.0e-01 9.0e-01 1.00e-01
 7 1.5e+00 3.6e-16 1.0e-16 +3.86e+01 +2.82e+01 7.0e-01 6.3e-01 1.00e-01
 8 5.8e-01 3.6e-16 1.2e-16 +3.41e+01 +3.01e+01 8.2e-01 2.6e-01 1.00e-01
 9 3.0e-01 3.8e-16 1.0e-16 +3.26e+01 +3.05e+01 7.7e-01 1.0e+00 1.00e+00
10 3.0e-01 3.9e-16 2.0e-16 +3.25e+01 +3.04e+01 9.3e-01 8.6e-01 1.00e-01
11 6.5e-02 3.9e-16 1.6e-16 +3.22e+01 +3.18e+01 1.0e+00 8.8e-01 1.00e-01
12 1.0e-02 3.7e-16 8.2e-17 +3.21e+01 +3.20e+01 1.0e-00 8.9e-01 1.00e-01
13 1.8e-03 3.7e-16 2.8e-17 +3.21e+01 +3.21e+01 1.1e+00 8.3e-01 1.00e-01
14 3.2e-04 3.8e-16 6.8e-17 +3.21e+01 +3.21e+01 1.2e+00 8.0e-01 1.00e-01
15 6.9e-05 3.6e-16 2.1e-16 +3.21e+01 +3.21e+01 1.4e+00 8.6e-01 1.00e-01
16 1.2e-05 3.6e-16 1.1e-15 +3.21e+01 +3.21e+01 1.3e+00 9.4e-01 1.00e-01
17 1.4e-06 3.8e-16 2.2e-15 +3.21e+01 +3.21e+01 1.2e+00 9.5e-01 1.00e-01
18 1.7e-07 3.7e-16 1.7e-15 +3.21e+01 +3.21e+01 1.2e+00 9.5e-01 1.00e-01

phase.value = pdOPT
   Iteration = 18
          mu = 1.7129088891721184e-07
relative gap = 3.7396624349410911e-08
         gap = 1.1990362224204828e-06
      digits = 7.4271675981360969e+00
objValPrimal = 3.2062693134e+01
objValDual   = 3.2062691935e+01
p.feas.error = 4.3170653684e-14
d.feas.error = 3.2862601529e-12
total time   = 0.000


Parameters are
maxIteration =    100
epsilonStar  = 1.000e-07
lambdaStar   = 1.000e+02
omegaStar    = 2.000e+00
lowerBound   = -1.000e+05
upperBound   = 1.000e+05
betaStar     = 1.000e-01
betaBar      = 3.000e-01
gammaStar    = 9.000e-01
epsilonDash  = 1.000e-07

                         Time(sec)  Ratio(% : MainLoop)
 Predictor time  =       0.000000,  nan
```

```
        ... abbreviation ...
 Total           =       0.000000,  nan

xVec =
{+1.552e+00,+6.710e-01,+9.815e-01,+1.407e+00,+9.422e-01}
xMat =
{
}
yMat =
{
}
    main loop time = 0.000000
        total time = 0.000000
  file   read time = 0.000000
```

Now we explain the items that appeared above in the file "example2.out".

- Lines with start '*' — These lines are comments in "example2.dat".
- Data, parameter, output — These are the file names we assigned for data, parameter and output, respectively.
- Lines between 'Predictor time' to 'Total time' — These lines display the profile data. These information may help us to tune up the parameters, but the details are rather complicate, because the profile data seriously depends on the internal algorithms.
- xVec — The primal variable vector $\boldsymbol{x}$.
- xMat — The primal variable matrix $\boldsymbol{X}$. But the SDPA-C does not output the primal variable matrix $\boldsymbol{X}$ because its size could be huge for large scale SDPs.
- yMat — The dual variable matrix $\boldsymbol{Y}$. But the SDPA-C does not output the dual variable matrix $\boldsymbol{Y}$ because its size could be huge for large scale SDPs.

7 Differences between the SDPA-C and the SDPA

7.1 Interface

The interfaces of the SDPA-C is almost the same as the SDPA [7], however the SDPA-C does not support the following features.

- dense input data file
- initial points from an external file
- output of the primal and dual matrix variables $\boldsymbol{X}, \boldsymbol{Y}$
- callable library

7.2 Algorithm and some properties

The SDPA-C incorporates a positive definite matrix completion technique proposed for the primal-dual interior-point method in [4, 6] into the SDPA [3]. The main features of SDPA-C are as follows.

- The SDPA-C employs a path-following primal-dual interior-point method, while the SDPA employs a Mehrotra-type predictor-corrector path-following primal-dual interior-point method.
- Using positive definite matrix completion techniques, the SDPA-C performs a sparse factorization of the dual matrix variable $\boldsymbol{Y}$ which is dense in general.
- The SDPA-C does not store any $n \times n$ dense matrix variables.
- The SDPA-C does not do any computation in which any $n \times n$ dense matrix is handled directly.

As a result, the SDPA-C has the following advantages/disadvantages:

- When a given SDP has large but sparse data matrices, the SDPA-C needs less computing time and less memory than the SDPA.
- When a given SDP does not have sparse data matrices, the SDPA-C needs more computing time and more memory than the SDPA; hence the use of the SDPA-C is not recommended for such SDPs.

The efficiency of the SDPA-C depends essentially on the "extended sparsity pattern" of the constraint matrices $\boldsymbol{F}_i \in \mathcal{S}$ $(i = 0, 1, \dots, m)$ of a given SDP. In the next subsection, we mention this "extended sparsity pattern" in detail.

7.3 Sparsity on SDPA-C

First, we define the aggregate sparsity pattern matrix of the constraint matrices $\boldsymbol{F}_i$ $(i = 0, 1, \dots, m)$ of an SDP. The aggregate sparsity pattern matrix of the constraint matrices $\boldsymbol{F}_i$ $(i = 0, 1, \dots, m)$ is an $n \times n$ symmetric symbolic matrix $\boldsymbol{A}$ defined as

$$\boldsymbol{A}_{pq} = \begin{cases} \star & \text{if } p = q \;\text{ or } [\boldsymbol{F}_i]_{pq} \neq 0 \;\text{ for } \exists i \in \{0, 1, 2, \dots, m\}\}, \\ 0 & \text{otherwise} . \end{cases}$$

where $[\boldsymbol{F}_i]_{pq}$ denotes the (p, q)th element of matrix $\boldsymbol{F}_i$.

The definition of the extended sparsity pattern matrix $\boldsymbol{E}$ is described in [4, 6]. It is obtained by performing a symbolic Cholesky factorization to the aggregate sparsity pattern matrix $\boldsymbol{A}$ with rows and columns symmetric reordered. The SDPA-C uses the extended sparsity pattern matrix to increase the computational efficiency. Accordingly, the extended sparsity pattern matrix $\boldsymbol{E}$ should ideally be as sparse as possible. Unfortunately, the problem of finding such a row and column symmetric ordering that minimizes the fill-in is $\mathcal{NP}$ complete. Hence, we employ heuristic methods to obtain a row and column symmetric ordering which possibly produces less fill-in. The SDPA-C utilizes two libraries for this purpose, METIS [5] and SPOOLES [1], and chooses the best ordering between the following five heuristic methods.

- METIS 4.0.1 multilevel nested dissection
- Spooles 2.2 minimum degree
- Spooles 2.2 generalized nested dissection
- Spooles 2.2 multisection
- Spooles 2.2 best of the generalized nested dissection and the multisection

If you want to obtain only the sparsity of the extended sparsity pattern of a given SDP without actually solving the SDP, assign a negative integer value for maxIteration in parameter file "param.sdpa" and try to solve the SDP by the SDPA-C. "mcp250-1.dat-s" from SDPLIB [2] is given below.

```
SDPA-C start at    Sat Jul 31 19:58:15 2004
data      is mcp250-1.dat-s : sparse
parameter is ./param.sdpa
out       is out2

aggregate sparsity pattern :                                 912 elements
extended sparsity pattern  :
        METIS4.0.1 (multilevel nested dissection)          2450 elements
        Spooles2.2 (minimum degree)                        2282 elements
        Spooles2.2 (generalized nested dissection)         2282 elements
        Spooles2.2 (multisection)                          2282 elements
        Spooles2.2 (best of ND and MS)                     2282 elements
        Selecting ..... Spooles2.2 (minimum degree)
dense matrix               :                              62500 elements
```

The number of elements of the extended sparsity pattern (2282) is much less than the number of elements of the dense matrix (62500). Hence the SDPA-C works efficiently for this SDP problem. In fact, the SDPA-C spent 0.90 seconds to solve it while the SDPA spent 4.43 seconds.

References

[1] C. Ashcraft, D. Pierce, D. K. Wah, and J. Wu, The reference manual for SPOOLES, release 2.2: An object oriented software library for solving sparse linear systems of equations, Boeing Shared Services Group, P. O. Box 24346, Mail Stop 7L-22, Seattle, WA 98124, January 1999; Available at http://netlib.bell-labs.com/netlib/linalg/spooles/.

[2] B. Borchers, "SDPLIB 1.2, a library of semidefinite programming test problems," *Optimization Methods and Software* **11 & 12** (1999) 683-690.

[3] K. Fujisawa, M. Kojima, K. Nakata, and M. Yamashita, "SDPA (Semidefinite Programming Algorithm) User's Manual — Version 6.0, " Research Report B-308, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, December 1995, Revised July 2002.

[4] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion I: General framework," *SIAM Journal on Optimization* **11** (2000) 647–674.

[5] G. Karypis and V. Kumar, METIS — A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 4.0 —, Department of Computer Science/Army HPC Research Center, University of Minnesota, Minneapolis, MN 55455, September 1998; Available at http://www-users.cs.umn.edu/~karypis/metis/metis.

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

[7] M. Yamashita, K. Fujisawa, and M. Kojima, "Implementation and Evaluation of SDPA6.0 (SemiDefinite Programming Algorithm 6.0)," *Optimization Methods and Software* **18** (2003) 491–505.