Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: http://www.tandfonline.com/loi/goms20

SDPLIB 1.2, a library of semidefinite programming test problems

Brian Borchers

To cite this article: Brian Borchers (1999) SDPLIB 1.2, a library of semidefinite programming test problems, Optimization Methods and Software, 11:1-4, 683-690, DOI: 10.1080/10556789908805769

To link to this article: https://doi.org/10.1080/10556789908805769

Published online: 30 Jan 2008.

Optimization Meth. & Soft., 1999, Vol. 11&12, pp. 683–690
Reprints available directly from the publisher
Photocopying permitted by license only

SDPLIB 1.2, A LIBRARY OF SEMIDEFINITE PROGRAMMING TEST PROBLEMS

BRIAN BORCHERS

Department of Mathematics, New Mexico Tech., Socorro, NM 87801, USA.

(Received 17 July 1998; Revised 22 September 1998; In final form 29 June 1999)

SDPLIB is a collection of semidefinite programming (SDP) test problems. The problems are drawn from a variety of applications, including truss topology design, control systems engineering, and relaxations of combinatorial optimization problems. The current version of the library contains a total of 92 SDP problems encoded in a standard format. It is hoped that SDPLIB will stimulate the development of improved software for the solution of SDP problems.

Keywords: Semidefinite Programming

1 INTRODUCTION

Semidefinite programming (SDP) is an important new area in optimization. Applications of semidefinite programming include truss topology design, control systems engineering, and relaxations of combinatorial optimization problems such as graph partitioning problems and quadratic assignment problems. [2,3,14,19]. A number of software packages for solving semidefinite programming problems are available [1,8,9,11,17,18].

Collections of test problems in various areas of optimization have been developed in recent years, including the NETLIB collection of linear programming test problems [13], the MIPLIB collection of mixed integer linear programming test problems [5,6], and the CUTE collection of nonlinear programming problems [7]. These libraries are a ready source of benchmarks for comparing the performance and robustness of software for solving these optimization problems. Such comparisons have led to significant improvements in the speed and robustness of optimization software.

The goal of the SDPLIB project is to gather together a collection of SDP test problems and make them freely available to researchers. The current version of the library contains a total of 92 problems, drawn from many different applications. The library contains problems in a wide range of sizes. Some problems are quite easy to solve, while others require considerable CPU time and memory to solve with current codes. The library also includes some primal and dual infeasible instances.

2 THE SDP PROBLEM

We work with semidefinite programming problems that have been written in the standard form used by the SDPA package [11].

$$\begin{aligned} &\min \quad c^T x \\ &(P) \sum_{i=1}^{m} x_i F_i - F_0 = X. \\ &\quad X \succeq 0 \end{aligned} \tag{1}$$

Here the matrices X and F_i, $i = 0, \dots, m$ are assumed to be of size n by n and symmetric. The constraint $X \succeq 0$ means that X must be positive semidefinite. The dual of the problem is:

$$\begin{aligned} &\max \operatorname{tr} F_0 Y \\ &(D) \operatorname{tr} F_i Y = c_i \quad i = 1, \dots, m. \\ &\quad Y \succeq 0 \end{aligned}$$

Note that several other standard forms for SDP have been used by a number of authors — these can be translated into the SDPA standard form with little effort. However, this translation often has the effect of changing the sign of the optimal objective function value.

3 THE SDPA SPARSE FILE FORMAT

The problems in SDPLIB are currently encoded in the SDPA sparse format [11]. The SDPA sparse format was designed to accommodate SDP problems in which the matrices F_i, $i = 0, \dots, m$, are block diagonal with sparse blocks. An SDPA sparse format file consists of six sections:

1. Comments. The file can begin with any number of lines of comments. Each line of comments must begin with '"' or '*'.

2. The first line after the comments contains `m`, the number of constraint matrices. Additional text on this line after `m` is ignored.
3. The second line after the comments contains `nblocks`, the number of blocks in the block diagonal structure of the matrices. Additional text on this line after `nblocks` is ignored.
4. The third line after the comments contains a vector of numbers that give the sizes of the individual blocks. The special characters ',', '(', ')', '{', and '}' can be used as punctuation and are ignored. Negative numbers may be used to indicate that a block is actually a diagonal submatrix. Thus a block size of "-5" indicates a 5 by 5 block in which only the diagonal elements are nonzero.
5. The fourth line after the comments contains the objective function vector `c`.
6. The remaining lines of the file contain entries in the constraint matrices, with one entry per line. The format for each line is

   `<matno> <blkno> <i> <j> <entry>`

   Here `matno` is the number of the matrix to which this entry belongs, `blkno` specifies the block within this matrix, `i` and `j` specify a location within the block, and `entry` gives the value of the entry in the matrix. Note that since all matrices are assumed to be symmetric, only entries in the upper triangle of a matrix are given.

For example, consider the problem:

$$\begin{aligned} &\min\ 10x_1 + 20x_2 \\ &\quad x_1F_1 + x_2F_2 - F_0 = X \\ &\quad X \succeq 0 \end{aligned}$$

where

$$F_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

$$F_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$F_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 5 & 2 \\ 0 & 0 & 2 & 6 \end{bmatrix}.$$

In SDPA sparse format, this problem can be written as:

```
"A sample problem.
2 =mdim
2 =nblocks
{2, 2}
10.0 20.0
0 1 1 1 1.0
0 1 2 2 2.0
0 2 1 1 3.0
0 2 2 2 4.0
1 1 1 1 1.0
1 1 2 2 1.0
2 1 2 2 1.0
2 2 1 1 5.0
2 2 1 2 2.0
2 2 2 2 6.0
```

4 THE SDPLIB PROBLEMS

1. These truss topology design problems were contributed by Katsuki Fujisawa. They are originally from [16].
2. These problems from control and system theory were contributed by Katsuki Fujisawa [10].
3. These graph equipartition problems were supplied by Steve Benson [4]. The random graphs were originally generated by Christoph Helmberg and Franz Rendl [15].
4. These graph partitioning problems were contributed by Katsuki Fujisawa [12,10].
5. These linear matrix inequalities from control systems engineering are taken from the SDPPACK web site [1]. The problems were originally developed by P. Gahinet.

TABLE 1 The SDPLIB problems

Problem	m	n	Optimal Objective Value	Notes
arch0	174	335	5.66517e−01	1
arch2	174	335	6.71515e−01	1
arch4	174	335	9.726274e−01	1
arch8	174	335	7.05698e+00	1
control1	21	15	1.778463e+01	2
control2	66	30	8.300000e+00	2
control3	136	45	1.363327e+01	2
control4	231	60	1.979423e+01	2
control5	351	75	1.68836e+01	2
control6	496	90	3.73044e+01	2
control7	666	105	2.06251e+01	2
control8	861	120	2.0286e+01	2
control9	1081	135	1.46754e+01	2
control10	1326	150	3.8533e+01	2
control11	1596	165	3.1959e+01	2
eqaulG11	801	801	6.291553e+02	3
equalG51	1001	1001	4.005601e+03	3
gpp100	101	100	−4.49435e+01	4
gpp124-1	125	124	−7.3431e+00	4
gpp124-2	125	124	−4.68623e+01	4
gpp124-3	125	124	−1.53014e+02	4
gpp124-4	125	124	−4.1899e+02	4
gpp250-1	250	250	−1.5445e+01	4
gpp250-2	250	250	−8.1869e+01	4
gpp250-3	250	250	−3.035e+02	4
gpp250-4	250	250	−7.473e+02	4
gpp500-1	501	500	−2.53e+01	4
gpp500-2	501	500	−1.5606e+02	4
gpp500-3	501	500	−5.1302e+02	4
gpp500-4	501	500	−1.56702e+03	5
hinf1	13	14	2.0326e+00	5
hinf2	13	16	1.0967e+01	5
hinf3	13	16	5.69e+01	5
hinf4	13	16	2.74764e+02	5
hinf5	13	16	3.63e+02	5
hinf6	13	16	4.490e+02	5

6. These infeasible problems were generated by a MATLAB procedure provided by Mike Todd.
7. These max cut problems were supplied by Steve Benson [4]. The random graphs G11, G32, and G51 were originally generated by Christoph Helmberg and Franz Rendl [15].
8. These max cut problems were contributed by Katsuki Fujisawa [10].
9. These quadratic assignment problems were contributed by Katsuki Fujisawa [12].

TABLE 2 The SDPLIB problems

Problem	m	n	Optimal Objective Value	Notes
hinf7	13	16	3.91e+02	5
hinf8	13	16	1.16e+02	5
hinf9	13	16	2.3625e+02	5
hinf10	21	18	1.09e+02	5
hinf11	31	22	6.59e+01	5
hinf12	43	24	2.0e−1	5
hinf13	57	30	4.6e+01	5
hinf14	73	34	1.30e+01	5
hinf15	91	37	2.5e+01	5
infd1	10	30	dual infeasible	6
infd2	10	30	dual infeasible	6
infp1	10	30	primal infeasible	6
infp2	10	30	primal infeasible	6
maxG11	800	800	6.291648e+02	7
maxG32	2000	2000	1.567640e+03	7
maxG51	1000	1000	4.003809e+03	7
maxG55	5000	5000	9.999210e+03	7
maxG60	7000	7000	1.522227e+04	7
mcp100	100	100	2.261574e+02	8
mcp124-1	124	124	1.419905e+02	8
mcp124-2	124	124	2.698802e+02	8
mcp124-3	124	124	4.677501e+02	8
mcp124-4	124	124	8.644119e+02	8
mcp250-1	250	250	3.172643e+02	8
mcp250-2	250	250	5.319301e+02	8
mcp250-3	250	250	9.811726e+02	8
mcp250-4	250	250	1.681960e+03	8
mcp500-1	500	500	5.981485e+02	8
mcp500-2	500	500	1.070057e+03	8
mcp500-3	500	500	1.847970e+03	8
mcp500-4	500	500	3.566738e+03	8
qap5	136	26	−4.360e+02	9
qap6	229	37	−3.8144e+02	9
qap7	358	50	−4.25e+02	9
qap8	529	65	−7.57e+02	9
qap9	748	82	−1.410e+03	9
qap10	1021	101	−1.093e+01	9

10. These SDP relaxations of box constrained quadratic programming problems were supplied by Steve Benson [4]. The random graphs which these problems are based on were originally generated by Christoph Helmberg and Franz Rendl [15].
11. These Lovasz ϑ numbers problems are taken from [8].
12. These Lovasz theta problems were contributed by Steve Benson [4]. The random graphs were originally generated by Christoph Helmberg and Franz Rendl [15].

TABLE 3 The SDPLIB problems

Problem	m	n	Optimal Objective Value	Notes
qpG11	800	1600	2.448659e+03	10
qpG51	1000	2000	1.181000e+03	10
ss30	132	426	2.02395e+01	1
theta1	104	50	2.300000e+01	11
theta2	498	100	3.287917e+01	11
theta3	1106	150	4.216698e+01	11
theta4	1949	200	5.032122e+01	11
theta5	3028	250	5.723231e+01	11
theta6	4375	300	6.347709e+01	11
thetaG11	2401	801	4.000000e+02	12
thetaG51	6910	1001	3.49000e+02	12
truss1	6	13	−8.999996e+00	13
truss2	58	133	−1.233804e+02	13
truss3	27	31	−9.109996e+00	13
truss4	12	19	−9.009996e+00	13
truss5	208	331	−1.326357e+02	13
truss6	172	451	−9.01001e+02	13
truss7	86	301	−9.00001e+02	13
truss8	496	628	−1.331146e+02	13

13. These truss topology design problems are taken from the SDPPACK web site [15]. The problems were originally developed by A. Nemirovski.

5 AVAILABILITY

SDPLIB is available as a unix tar file, compressed with either the unix compress utility or the gzip compression program. Note that the complete uncompressed library requires about 65 megabytes of storage. The current version of the library is available at http://www.nmt.edu/~sdplib/.

References

[1] Alizadeh, F., Haberly, J.-P., Nayakkankuppam, M. V., Overton, M. L. and Schmieta, S. (1997). SDPpack user's guide — version 0.9 beta. Technical Report TR1997-737, Courant Institute of Mathematical Sciences, NYU, New York, NY, June.

[2] Farid Alizadeh. (1995). Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, **5**(1), 13–51.

[3] Ben-Tal, A. and Merirovski, A. (1997). Robust truss topology design via semidefinite programming. *SIAM Journal on Optimization*, **7**(4), 991–1016.

[4] Steven J. Benson, Yinyu ye, and Xiong Zhang. (1998). Solving large — scale sparse semidefinite programs for combinatorial optimization. Department of Management Science, University of Iowa, May.

[5] Bixby, R. E., Boyd, E. A. and Indovina, R. R. (1992). MIPLIB: A test set of mixed integer programming problems. *SIAM News*, **25**(2), March.

[6] Bixby, R. E., Ceria, S., McZeal, C. M. and Savelsbergh, M. W. P. (1998). An updated mixed integer programming library: MIPLIB 3.0. *Optima*, **54**, 12–15.

[7] Bongartz, I., Conn, A. R., Gould, N. and Toint, Ph. L. (1995). CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, **21**, 123–160.

[8] Borchers, B. (1998). CSDP, a C library for semidefinite programming. Technical report, New Mexico Tech, June.

[9] Brixius, N., Potra, F. A. and Sheng, R. (1998). SDPHA: a MATLAB implementation of homogeneous interior-point algorithms for semidefinite programming. Technical report, University of Iowa, Iowa City, IA.

[10] Fujisawa, K., Fukuda, M., Kojima, M. and Nakata, K. (1997). Numerical evaluation of SDPA (semidefinite programming algorithm). Technical Report B-330, Tokyo Institute of Technology, September.

[11] Katsuki Fujisawa and Masakazu Kojima. (1995). SDPA (semidefinite programming algorithm) users manual. Technical Report B-308, Tokyo Institute of Technology, December.

[12] Katsuki Fujisawa, Masakazu Kojima, and Kazuhide Nakata. (1997). Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. Technical Report B-324, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology.

[13] Gay, D. M. (1992). Electronic mail distribution of linear programming test problems. *COAL Newsleter*, **13**, 10–12.

[14] Goemans, M. X. (1997). Semidefinite programming in combinatorial optimization. *Mathematical Programming*, **79**, 143–161.

[15] Helmberg, C. and Rendl, F. (1997). A spectral bundle method for semidefinite programming. Technical Report ZIB Preprint SC-97-37, Konrad-Zuse-Zentrum für Informationstechnik, August. To appear in *SIAM Journal on Optimization*.

[16] Nakamura, T. and Ohsaki, M. (1992). A natural generator of optimum topology of plane trusses for specified fundamental frequency. *Computer Methods in Applied Mechanics and Engineering*, **94**, 113–129.

[17] Toh, K. C., Todd, M. J. and Tutuncu, R. H. (1996). SDPT3 — a MATLAB software package for semidefinite programming. Technical Report TR1177, Cornell University, December.

[18] Lieven Vandenberghe and Stephen Boyd. (1994). *SP Software for Semidefinite Programming. User's Guide*. K. U. Leuven and Standford University, October.

[19] Lieven Vandenberghe and Stephen Boyd. (1996). Semidefinite programming. *SIAM Review*, **38**(1), 49–95.