Conic optimization software

Imre Pólik*

Abstract

We give an overview of cone optimization software, with special attention to the differences between the existing packages. We assume the reader is familiar with the theory and algorithms of cone optimization, thus technical details are kept at a minimum. We also outline current research trends and area of potential improvement.

1 Problem description

Conic optimization solvers target problems of the form

$$\begin{aligned} \min\ & c^T x & \qquad \max\ & b^T y \\ Ax &= b & A^T y + s &= c \\ x &\in \mathcal{K} & s &\in \mathcal{K}^*, \end{aligned} \tag{1.1}$$

where $b, y \in \mathbb{R}^m$, $c, x, s \in \mathbb{R}^N$, $A \in \mathbb{R}^{m\times N}$, $\mathcal{K}, \mathcal{K}^* \subset \mathbb{R}^N$, and $\mathcal{K}^*$ is the dual of $\mathcal{K}$. In general solvers exist if $\mathcal{K}$ is one of, or the product of a few copies of the following cones:

nonnegative orthant: the set of nonnegative vectors, $\mathbb{R}^n_+$;

Lorentz cone: the set $\mathbb{L}_{n+1} = \{(u_0, u) \in \mathbb{R}_+ \times \mathbb{R}^n : u_0 \geq \|u\|\}$, also called the quadratic or ice-cream cone;

rotated Lorentz cone: the set $\mathbb{L}^r_{n+1} = \left\{(u_0, u_1, u) \in \mathbb{R}_+ \times \mathbb{R}^n : u_0 u_1 \geq \|u\|^2, u_0 \geq 0\right\}$;

positive semidefinite cone: the cone $\mathbb{PS}^{n\times n}$ of $n \times n$ real symmetric positive semidefinite matrices;

complex Hermitian cone: the cone $n \times n$ complex Hermitian positive semidefinite matrices.

*Lehigh University, imre@polik.net

The dimensions of the cones forming the product can be arbitrary. For semidefinite optimization we need to introduce some special notation, since the variables are matrices:

$$\begin{array}{ll} \min C \bullet X & \max b^T y \\ A^{(i)} \bullet X = b_i, i = 1, \ldots, m & \sum_{i=1}^{m} A^{(i)} y_i + S = C \\ X \in \mathbb{PS}^{n \times n} & S \in \mathbb{PS}^{n \times n}, \end{array} \tag{1.2}$$

where $X, S, C, A^{(i)} \in \mathbb{R}^{n \times n}$, $b, y \in \mathbb{R}^m$. For symmetric matrices U and V the quantity $U \bullet V = \mathrm{Tr}\,(UV)$ is a scalar product defined on symmetric matrices, and is identical to the sum of the componentwise products of the matrix elements.

Very often the matrices $A^{(i)}$ are sparse or have some other special structure, which can be exploited in the algorithms. For general results on the theory and algorithms of conic optimization see [1, 2] and the references therein.

2 Algorithms

Interior point methods are practically the only choice for semidefinite optimization, most of the existing general purpose solvers fall into this category. PENSDP [3], a modified version of PENNON is the only general purpose semidefinite programming solver using a different approach. The implementation of IPMs for conic optimization is more complicated than that for linear optimization, see [4–6] for more details.

2.1 General features and capabilities

For generic, dense, unstructured semidefinite optimization problems the following limits apply on a current PC. The number of linear equalities can be $m \leq 10000$, using a few matrix variables each with dimensions $n_i \leq 5000$. Larger problems can be solved if the problem has some special structure, see §6.2. One of the most serious limitations of IPMs for semidefinite programming is that the Newton system tends to be dense even if the original problem is very sparse. This is due to the symmetric scaling introduced in forming the Newton system, and thus it limits the problem size for all methods that actually form the Newton system.

2.2 Computational cost

The computation cost of one iteration of interior-point methods for the semidefinite programming problem (1.2) is $\mathcal{O}\left(m^3 + mn^3 + m^2n^2\right)$. Depending on the dimensions of the cones, any of the three terms can dominate this expression, since m can be as large as n^2. If the problem is sparse, then the second and third terms can usually be improved. Alternatively, one can decide not to form the Newton system explicitly, and try to use an iterative method to solve the Newton system without forming it. For second-order cone programming the cost per iteration for a problem with K cones each of dimension n and m linear equalities is $\mathcal{O}\left(m^3 + m^2n + Kn^2\right)$.

2.3 Initialization

Interior-point methods require a strictly feasible solution to start the algorithm. However, if the initial solution is far from the central path, then the algorithm will progress very slowly, thus the standard way of initializing is not practical. There are two techniques that are used widely to circumvent this:

self-dual embedding: This technique embeds the original problem in a larger problem, which will have a strictly feasible starting solution on the central path. From the optimal solution of the larger problem one can recover an optimal solution of the original problem, or detect infeasibility [7–12];

infeasible interior-point methods: This method [6, 13, 14] starts with an infeasible solution and works towards improving feasibility and optimality simultaneously.

3 Input formats

3.1 Text-based formats

One of the earliest input formats for general symmetric cone programming is the one introduced in SDPpack. That solver is now obsolete, but there are converters from the SDPpack format to SeDuMi's binary format, for example. The sparse SDPA format is a simplified version of the SDPpack format and has become the standard format for general semidefinite programming problems. It supported by most major solvers and can be easily converted to any other format.

3.2 Binary formats

Solvers that are implemented in Matlab use a standard Matlab data file as their input, but the actual structure of the data is different between solvers. Also, Python-based solvers can store their data in a pickled file. These formats are special to the given solver and are not portable between different solvers.

3.3 Modelling language support

Unfortunately, commercial modelling languages do not support SDP or cone programming in general, thus limit their use in the commercial sector. Second order conic optimization is in a slightly better situation, since it is easily formulated, but there are only very few specialized solvers available. Of course, the best way to formulate these problems would be an indirect one: the user should not even know that the problem is solved using conic programming. Robust optimization fits this scheme very well.

There are two widely used open-source modelling languages that support conic optimization: Yalmip [15], `http://control.ee.ethz.ch/~joloef/yalmip.php` and CVX [16, 17], `http://www.stanford.edu/~boyd/cvx`. Both of these packages are written in Matlab and are interfaced to a variety of conic optimization packages.

3.4 COIN Optimization Services

The Optimization Services interface in COIN-OR has been extended [18] to model general conic optimization problems. Driven by applications and practical problems, the format provides ways to express problems in the most natural way, preserving the important structure of the problem.

There are some key differences between problem instance representation in COIN OS and modelling languages. Modelling languages tend to separate problem data from problem structure, which prevents exploiting the structure lying within the data. In contrast, COIN OS keeps the data and the structure together.

4 Current software packages

In the following we give an overview of existing cone optimization solvers in alphabetical order.

4.1 CPLEX

License: Commercial
Developer: IBM
Capabilities: LP, SOCP
Algorithm: Interior-point method
Special features: Mixed-integer SOCP
Website: `http://www.ibm.com`
Reference: [19]

CPLEX solves second-order conic problems by treating them as special (non-convex) quadratically constrained optimization problems. It is also one of the few solvers that can handle mixed-integer conic problems.

4.2 CSDP

License: open source
Developer: Brian Borchers
Capabilities: LP, SDP
Algorithm: Interior-point method
Special features: Callable library, Interface to R
Website: `http://projects.coin-or.org/Csdp`
Reference: [4, 20]

Parallel version in both shared [21] and distributed [22] memory configurations are available. CSDP is one of the most advanced solvers, it exploits sparsity in the data matrices $A^{(i)}$.

4.3 CVXOPT

License: open source
Developer: Joachim Dahl, Lieven Vandenberghe
Capabilities: LP, SOCP, SDP
Algorithm: Interior-point method
Special features: Exploits triangular, band, and sparse matrices
Website: `http://abel.ee.ucla.edu/cvxopt/`
Reference: [23]

A fairly new package written in Python, with interfaces to BLAS, LAPACK, sparse matrix libraries and NumPy. The interfaces can also be used independent of the solver. The package is related to the modelling interface CVXMOD.

4.4 DSDP

License: free
Developer: Steve Benson, Yinyu Ye, and Xiong Zhang
Capabilities: LP, SDP
Algorithm: Interior-point method
Special features: dual scaling
Website: `http://www.mcs.anl.gov/hs/software/DSDP`
Reference: [24–26]

DSDP is using a dual scaling algorithm and is very efficient when the dual slack variable is sparse. It is the only interior-point implementation that is not using a primal-dual approach.

4.5 LMIlab

License: commercial
Developer: The Mathworks
Capabilities: SDP, LMI
Algorithm: Interior-point method
Special features: exploiting a Kronecker structure
Website: `http://mathworks.com`
Reference: [27, 28]

A Matlab toolbox, later renamed to Robust Control Toolbox, implementing a projective algorithm of Nesterov and Nemirovski. This is the only solver that can exploit a Kronecker structure in the problems, and is thus very efficient for optimal control problems, which is its intended use.

4.6 LOQO

License: commercial
Developer: Robert Vanderbei
Capabilities: LP, SOCP
Algorithm: Interior-point method
Special features: nonlinear programming
Website: `http://www.princeton.edu/~rvdb/loqo`
Reference: [29]

LOQO does solve second-order conic optimization problems but it uses a different approach. It handles the constraint $x_1 - \|x_{2:n}\|_2 \geq 0$ as a general nonlinear constraint, with some extra care taken due to the nondifferentiability of this form. In a similar way, other nonlinear programming solvers can solve SOCO problems at least in principle.

4.7 MOSEK

License: Commercial
Developer: Erling Andersen, Mosek ApS
Capabilities: LP, SOCP
Algorithm: Interior-point method
Special features: Mixed-integer SOCP, Nonlinear programming
Website: `http://www.mosek.com`
Reference: [30]

MOSEK is a commercial solver for second-order cone and nonlinear problems. It also implements methods to solve mixed-integer problems.

4.8 PENSDP

License: Commercial
Developer: Michal Kočvara
Capabilities: SDP
Algorithm: Augmented Lagrangian method
Special features: nonlinear SDP
Website: `http://www.penopt.com/pensdp.html`
Reference: [3]

A modified version of the PENNON nonlinear programming solver. It is very efficient even for large, sparse problems. The only truly general purpose sonic solver not implementing an interior-point method.

4.9 SDPA

License: open source
Developers: Katsuki Fujisawa, Mituhiro Fukuda, Yoshiaki Futakata, Kazuhiro Kobayashi, Masakazu Kojima, Kazuhide Nakata, Maho Nakata, Makoto Yamashita
Capabilities: LP, SDP

Algorithm: Interior-point method
Special features: Callable library, distributed/shared memory parallel version, arbitrary precision, sparsity/decomposition
Website: `http://sdpa.indsys.chuo-u.ac.jp/sdpa/software.html`
Reference: [31–34]

One of the most advanced packages for semidefinite optimization developed by a large research group in Japan. Efficient parallel versions [35, 36] are released for both shared and distributed memory environments. SDPA also implements techniques to decompose sparse problems using matrix completion [37–39] methods. A Matlab interface [40] is also available.

4.10 SDPlr

License: open source
Developer: Samuel Burer, Renato D.C. Monteiro, Changhui Choi
Capabilities: LP, SDP
Algorithm: Interior-point method
Special features: Special handling of low-rank coefficient matrices.
Website: `http://dollar.biz.uiowa.edu/~sburer/`
Reference: [41–43]

SDPlr puts special emphasis on exploiting the special low-rank structure of the coefficient matrices and variables. It is very efficient on some special problems, arising from, e.g., combinatorics.

4.11 SDPT3

License: open source
Developer: Kim Toh, Michael J. Todd, Reha Tütüncü
Capabilities: LP, SOCP, SDP
Algorithm: Interior-point method
Special features: logarithmic objective function, mixed second-order and semidefinite constraints
Website: `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`
Reference: [14, 44, 45]

SDPT3 is written in Matlab and C and can handle problems with mixed nonnegative, second-order and semidefinite variables. It is also one of the few solvers that can handle a logarithmic term ($\log x$, $\log(x_0 - \|x\|)$ or $\log\det(X)$ depending on the cone) in the objective function simply by internally manipulating the central path parameter in the interior-point method.

4.12 SeDuMi

License: GPL
Developer: Jos Sturm (1998-2003), Imre Pólik (since 2005)
Capabilities: LP, SOCP, SDP
Algorithm: Interior-point method
Special features: mixed second-order and semidefinite constraints
Website: `http://sedumi.ie.lehigh.edu`
Reference: [46, 47]

SeDuMi was one of the first widely successful conic optimization packages. It is written in Matlab and C. Its success is due to its simple user interface and its numerical accuracy and robustness. It is one of the few solvers that used a self-dual embedding instead of an infeasible scheme to initialize the problem. It implements most of the techniques discussed in [5].

Following the tragic death of Jos Sturm in 2003, the development has been continued by Imre Pólik.

4.13 SMCP

License: open source
Developers: M. S. Andersen, J. Dahl, and L. Vandenberghe
Capabilities: LP, SDP
Algorithm: Interior-point method
Special features: Special handling of sparse problem data
Website: `http://abel.ee.ucla.edu/smcp`
Reference: [48]

A new research code, currently in experimental stage. It exploits the sparsity of the problem using chordal graphs and decomposition techniques. Written in Python and C.

5 Obsolete packages

Here is a list of some packages that are no longer developed and maintained.

5.1 SBmethod

Written in C++ by Christoph Helmberg, it implements a spectral bundle algorithm [49, 50] for semidefinite optimization. Not very efficient as a general purpose solver. Last updated in 2004.

5.2 SDPHA

A Matlab package for SDP by F. A. Potra, R. Sheng and N. Brixius. No longer available.

5.3 SDPpack

Launched in 1997, SDPpack was the first package to solve mixed SOCP and SDP problems. It was written by Farid Alizadeh, Jean-Pierre A. Haeberly, Madhu V. Nayakkankuppam, Michael L. Overton and Stefan Schmieta. It introduced an input format combining both sparse and dense storage schemes. The SDPA format is a simpler version of this, using only sparse data and SDP constraints. It is the only package that uses the AHO scaling technique, developed by three of the authors. Available from `http://cs.nyu.edu/overton/software/sdppack.html`.

5.4 Sdpsol

Developed by Shao-Po Wu and Stephen Boyd [51, 52], it was solving SDPs and determinant maximization problems. It was last updated in 1996. Available from `http://www.stanford.edu/~boyd/old_software/SDPSOL.html`.

5.5 SOCP.C

A simple SOCP solver [53] written by Miguel Sousa Lobo, Lieven Vandenberge, Stephen Boyd and Herve Lebret, last updated in 1997. Available from `http://www.stanford.edu/~boyd/old_software/SOCP.html`. This is the only open-source SOCP solver written in C.

5.6 SP

SP (`http://www.ee.ucla.edu/~vandenbe/sp.html`), released in 1994 by Lieven Vandenberghe, was the first freely distributed SDP package. It used a Matlab/C implementation of Nesterov and Todd's primal-dual potential reduction method.

6 Current research trends and future directions

It is commonly agreed that the for unstructured or fully dense semidefinite optimization problems interior-point methods have reached their limits. In what follows we give a brief overview of current efforts and research directions.

6.1 Parallelization

Interior-point methods are very good candidates for parallelization, as they take very few iterations, while the cost per iteration is typically very high. There are a number of ways to achieve good parallel performance. As most of the operations in an iteration can be written using standard BLAS/LAPACK calls, parallelization is fairly easy for dense data on a shared-memory architecture. Another approach is to use OpenMP to automagically parallelize loops in the code. This approach is used by CSDP [4].

SDPA and CSDP have also been extended to distributed memory clusters [22, 32]. These approaches distribute the computation of the Newton system.

6.2 Exploiting the structure of the problems

Current research focuses on uncovering and exploiting the special structure of the problem to reduce memory requirements and speed up the computations, see [54] for a survey. The following techniques are known:

6.2.1 Chordal decomposition

These methods [34, 36–39] exploit the fact that the dual slack variable S inherits the sparsity structure of the coefficient matrices. By enforcing positive semidefiniteness only on a small number of carefully chosen submatrices of a large sparse matrix, one can make sure that the original matrix can be completed to be positive semidefinite. This can lead to a dramatic reduction of the problem size, but in general it may take a long time to find a good decomposition. Also, the decomposed problem may take a longer time to

solve, especially if a lot of submatrices are used. On the other hand, the reduced problem can be solved with a standard SDP solver.

SDPA and SMCP are the two solvers that implement this technique in their preprocessing routines. The developers of SDPA have also released a standalone converter useable with any compatible solver.

6.2.2 Permutations and group symmetry

Even if the problem is fully dense, the structure induced by the physical problem can be exploited to reduce the size of the matrices. In particular, invariance under permutations has been shown to have a dramatic impact. See [55, 56] for applications in quadratic assignment and truss-topology design problems.

There is currently no solver that would try to uncover symmetry in the problem. In fact, without knowing anything about the background of a particular problem instance, it is very hard to detect this kind of structure. It is left to the modeler to use the more efficient representation. On the other hand, the reduced problem does not require any special SDP solvers.

There are no known software tools in this area yet.

6.2.3 Special linear operators

These techniques exploit the special structure of the linear operator $\mathcal{A}$. In general $\mathcal{A}$ maps an $n \times n$ matrix to an m-dimensional vector, thus its size is mn^2 and it takes $\mathcal{O}\left(mn^2\right)$ operations to apply it to an $n \times n$ matrix. There are some cases when the structure of the operator allows to save in both the storage and computational cost. A few examples are:

Kronecker products: If $\mathcal{A}(X) = AXB$ in problem (1.2), then $\mathcal{A}$ is mapping an $n \times n$ matrix to another $n \times n$ matrix. In general such an operator is represented by the Kronecker product $B^T \otimes A$, which is an $n^2 \times n^2$ matrix. This only allows problems with n at most 100. Moreover, not only we can save in storage, but this special structure can be exploited inside the IPM to obtain an $\mathcal{O}\left(n^4\right)$ cost per iteration instead of the standard $\mathcal{O}\left(n^6\right)$. This structure is very common in optimal control problems [57].

Low-rank coefficients: If the coefficient matrices A_i are of low rank, then $A_i \bullet X$ can be simplified quite a bit. For example if $A_i = aa^T$, then $A_i \bullet X = a^T A a$, and also this structure can be exploited inside the IPM to speed up the computation of the Newton system.

The solvers LMILab [27], SDPlr [41] and modified versions of CSDP [22] can take advantage of this format.

This technique faces a few challenges. First, it is not immediately clear how structures like this can be expressed in the existing input formats. Some tried to modify the SDPA format to accommodate low-rank coefficient matrices [22]. Modelling systems CVX and Yalmip can express these problems, but they convert them into standard SDP form before passing them on to the solvers. The extension of COIN Optimization Services [18] to cone programming gives both the modeler and the solver access to this structure.

The second problem is that one needs to modify the solvers to be able to exploit these kinds of structures. LMIlab was created with this goal in mind and it remains a very efficient solver for problems in control theory as part of the Robust Control Toolbox in Matlab.

6.2.4 Graph problems

As combinatorial optimization and graph theory is a fruitful area of application for SDP, it is not surprising that problems defined over a graph have a very special structure. In fact, it is possible to rewrite the internal computations of an IPM in terms of the graph data. SDPlr is currently the only solver that can exploit this structure.

References

[1] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications.* Kluwer Academic Publishers, 2000.

[2] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming, Series B*, 95:3–51, 2002.

[3] M. Kočvara and M. Stingl. *PENSDP Users Guide (Version 2.2).* PENOPT GbR, 2006.

[4] B. Borchers. CSDP, A C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

[5] J. F. Sturm. Implementation of interior point methods for mixed semidefinite and second order cone optimization problems. *Optimization Methods and Software*, 17(6):1105–1154, 2002.

[6] K. C. Toh, R. H. Tütüncü, and M. J. Todd. On the implementation of SDPT3 (version 3.1) – a Matlab software package for semidefinite-quadratic-linear programming. In *Proceedings of the IEEE Conference on Computer-Aided Control System Design*, 2004.

[7] B. Jansen, C. Roos, and T. Terlaky. The theory of linear programming: Skew symmetric self-dual problems and the central path. *Optimization*, 29:225–233, 1994.

[8] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization. An Interior Approach.* Springer, New York, USA, 2nd edition, 2006.

[9] Y. Ye, M. J. Todd, and S. Mizuno. An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19:53–67, 1994.

[10] R. M. Freund. On the behavior of the homogeneous self-dual model for conic convex optimization. *Mathematical Programming*, 106(3):527 – 545, 2006.

[11] E. de Klerk, C. Roos, and T. Terlaky. Infeasible-start semidefinite programming algorithms via self-dual embeddings. In P. M. Pardalos and H. Wolkowicz, editors, *Topics in Semidefinite and Interior Point Methods*, volume 18 of *Fields Institute Communications*, pages 215–236. AMS, Providence, RI, 1998.

[12] Z.-Q. Luo, J. F. Sturm, and S. Zhang. Conic linear programming and self-dual embedding. *Optimization Methods and Software*, 14:169–218, 2000.

[13] S. J. Wright. *Primal-Dual Interior-Point Methods.* SIAM, Philadelphia, USA, 1997.

[14] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming Series B*, 95:189–217, 2003.

[15] J. Löfberg. YALMIP: a toolbox for modeling and optimization in MATLAB. In *Proceedings of the 2004 IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, 2004.

[16] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming. Web page and software, `http://stanford.edu/~boyd/cvx`, 2008.

[17] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control (a tribute to M. Vidyasagar)*. Springer, 2008. To appear.

[18] Horand Gassmann, Jun Ma, Kipp Martin, and Imre Pólik. *Extending COIN OS to model conic optimization problems*. In preparation., 2010.

[19] R. E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1):3–15, 2002.

[20] B. Borchers. CSDP 2.3 user's guide. *Optimization Methods and Software*, 11(1):597–611, 1999.

[21] B. Borchers and J. G. Young. Implementation of a primal-dual method for SDP on a shared memory parallel architecture. *Computational Optimization and Applications*, 37(3):355–369, 2007.

[22] I. D. Ivanov and E. de Klerk. Parallel implementation of a semidefinite programming solver based on CSDP in a distributed memory cluster. CentER Discussion Paper 2007-20, Tilburg University, The Netherlands, 2007.

[23] J. Dahl and L. Vandenberghe. CVXOPT: A python package for convex optimization. In *Proc. Eur. Conf. Op. Res*, 2006.

[24] Steven J. Benson and Yinyu Ye. DSDP5: Software for semidefinite programming. Technical Report ANL/MCS-P1289-0905, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, September 2005. Submitted to ACM Transactions on Mathematical Software.

[25] Steven J. Benson, Yinyu Ye, and Xiong Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM Journal on Optimization*, 10(2):443–461, 2000.

[26] Steven J. Benson and Yinyu Ye. DSDP5 user guide — software for semidefinite programming. Technical Report ANL/MCS-TM-277,

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, September 2005. `http://www.mcs.anl.gov/~benson/dsdp`.

[27] A. Nemirovski and P. Gahinet. The projective method for solving linear matrix inequalities. In *Proc. Amer. Contr. Conf.*, pages 840–844, 1994.

[28] Y. E. Nesterov and A. Nemirovski. *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13 of *SIAM Studies in Applied Mathematics*. SIAM Publications, Philadelphia, PA, 1994.

[29] R. J. Vanderbei. *LOQO User's Guide – Version 4.05*. Princeton University, School of Engineering and Applied Science, Department of Operations Research and Financial Engineering, Princeton, New Jersey, 2006.

[30] E. D. Andersen, B. Jensen, R. Sandvik, and U. Worsøe. The improvements in MOSEK version 5. Technical report 1-2007, MOSEK ApS, Fruebjergvej 3 Box 16, 2100 Copenhagen, Denmark, 2007.

[31] Katsuki Fujisawa, Masakazu Kojima, Kazuhide Nakata, and Makoto Yamashita. SDPA (SemiDefinite Programming Algorithm) user's manual — version 6.2.0. Research Report B-308, Dept. Math. & Comp. Sciences, Tokyo Institute of Technology, Japan, 1995. Revised in 2004.

[32] M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of SDPA 6.0 (SemiDefinite Programming Algorithm 6.0). *Optimization Methods and Software*, 18:491–505, 2003.

[33] Katsuki Fujisawa, Mituhiro Fukuda, Masakazu Kojima, and Kazuhide Nakata. Numerical evaluation of the SDPA (SemiDefinite Programming Algorithm). In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 267–301. Kluwer Academic Press, 2000.

[34] Katsuki Fujisawa, Masakazu Kojima, and Kazuhide Nakata. Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. *Mathematical Programming*, 79:235–253, 1997.

[35] Makoto Yamashita, Katsuki Fujisawa, and Masakazu Kojima. SDPARA : SemiDefinite Programming Algorithm paRAllel version. *Parallel Computing*, 29:1053–1067, 2003.

[36] Kazuhide Nakata, Makoto Yamashita, Katsuki Fujisawa, and Masakazu Kojima. A parallel primal-dual interior-point method for semidefinite programs using positive definite matrix completion. *Parallel Computing*, 32:24–43, 2006.

[37] Katsuki Fujisawa, Mituhiro Fukuda, Masakazu Kojima, Kazuhide Nakata, and Makoto Yamashita. SDPA-C (SemiDefinite Programming Algorithm – Completion method) User's Manual — Version 6.10. Research Report B-409, Dept. Math. & Comp. Sciences, Tokyo Institute of Technology, Japan, 2004.

[38] Mituhiro Fukuda, Masakazu Kojima, Kazuo Murota, and Kazuhide Nakata. Exploiting sparsity in semidefinite programming via matrix completion I: General framework. *SIAM Journal on Optimization*, 11:647–674, 2001.

[39] Kazuhide Nakata, Katsuki Fujisawa, Mituhiro Fukuda, Masakazu Kojima, and Kazuo Murota. Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results. *Mathematical Programming*, 95:303–327, 2003.

[40] Katsuki Fujisawaa, Yoshiaki Futakata, Satoshi Nakamura Masakazu Kojima, Satoshi Matsuyama, Kazuhide Nakata, and Makoto Yamashita. SDPA-M (SemiDefinite Programming Algorithm in MATLAB) User's manual — version 6.2.0. Research Report B-359, Dept. Math. & Comp. Sciences, Tokyo Institute of Technology, Japan, 2000. Revised in 2005.

[41] S. Burer and R.D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming (series B)*, 95(2):329–357, 2003.

[42] S. Burer and R.D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, 2005.

[43] S. Burer and C. Choi. Computational enhancements in low-rank semidefinite programming. *Optimization Methods and Software*, 21(3):493–512, 2006.

[44] K.C. Toh, M.J. Todd, and R.H. Tutuncu. SDPT3 – a Matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

[45] R.H Tutuncu, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95:189–217, 2003.

[46] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[47] J. F. Sturm. Primal-dual interior point approach to semidefinite programming. In J. B. G. Frenk, C. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[48] M. S. Andersen, J. Dahl, and L. Vandenberghe. Implementation of non-symmetric interior-point methods for linear optimization over sparse matrix cones. Technical report, 2009. Submitted to Mathematical Programming Computation.

[49] C. Helmberg and K. C. Kiwiel. A spectral bundle method with bounds. *Mathematical Programming*, 93(2):173–194, 2002.

[50] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal of Optimization*, 10(3):673–696, 2000.

[51] S.-P. Wu and S. Boyd. Sdpsol: A parser/solver for semidefinite programs with matrix structure. In L. El Ghaoui and S.-I. Niculescu, editors, *Recent Advances in LMI Methods for Control*, chapter 4, pages 79–91. SIAM, 2000.

[52] S.-P. Wu and S. Boyd. Design and implementation of a parser/solver for SDPs with matrix structure. In *Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design (CACSD)*, pages 240–245, 1996.

[53] M. Lobo, L. Vandenberge, S. Boyd, and H. Lebret. Second-order cone programming. Available as part of the SOCP package, 1997.

[54] E. de Klerk. Exploiting special structure in semidefinite programming: A survey of theory and applications. *European Journal of Operational Research*, 201(1):1–10, 2010.

[55] E. de Klerk and R Sotirov. Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. *Mathematical Programming*, 122(2):225–246, 2010.

[56] Y. Q. Bai, E. de Klerk, D. V. Pasechnik, and R. Sotirov. Exploiting group symmetry in truss topology optimization. *Optimization and Engineering*, 10(3):331–349, 2009.

[57] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear Matrix Inequalities and System and Control Theory*, volume 15 of *SIAM Studies in Applied Mathematics*. SIAM, Philadelphia, USA, 1994.