The State-of-the-Art in Conic Optimization Software

H. D. Mittelmann

School of Mathematical and Statistical Sciences, Arizona State University,
mittelmann@asu.edu

Abstract This work gives an overview of the major codes available for the solution of linear semidefinite (SDP) and second-order cone (SOCP) programs. Many of these codes also solve linear programs (LP). Some developments since the 7th DIMACS Challenge [10, 18] are pointed out as well as some currently under way. Instead of presenting performance tables, reference is made to the ongoing benchmark webpage [20] as well as other related efforts.

1 Introduction

The following two sections are taken from [18]. In the first section the optimization problems under consideration are defined and the needed notation is introduced. In the second section a comprehensive set of error measures is given which helps evaluate and compare different codes. These measures have become standard since the 7th DIMACS Challenge and are produced my most of the codes listed in the following.

1.1 The Problems Solved

The primal and dual pair of conic optimization problems over a self-dual cone are defined as

$$(P) \qquad \begin{array}{lll} \min & \langle c, x\rangle & \\ \text{s.t.} & x \in K & \\ & \mathcal{A}x = b & \end{array} \qquad\qquad \begin{array}{lll} \max & b^T y & \\ s.t. & z \in K & \\ & \mathcal{A}^* y + z = c & \end{array} \qquad (D)$$

where

- K is a closed, convex cone in a Euclidean space X.
- $\mathcal{A} : X \to \mathbb{R}^m$ is a linear operator, and $\mathcal{A}^*$ is its adjoint.
- $b \in \mathbb{R}^m$, and $c \in X$.

In the case of a semidefinite-quadratic-linear program these are defined as follows:

- **The space X:** $x \in X \Leftrightarrow x = (x^s_1, \ldots, x^s_{n_s}, x^q_1, \ldots, x^q_{n_q}, x^\ell)$, where
  - $x^s_1, \ldots, x^s_{n_s}$ are symmetric matrices (possibly of various sizes).
  - $x^q_1, \ldots, x^q_{n_q}$ are vectors (again, possibly of various sizes).
  - x^ℓ is a vector.
- **The cone K:** $x \in K \Leftrightarrow x^s_j \succeq 0 \; \forall j, \; x^q_k \geq_q 0 \; \forall k, \; x^\ell \geq 0$, where
  - $u \succeq 0$ means that the symmetric matrix u is positive semidefinite.
  - $v \geq_q 0$ means that the vector v is in a quadratic cone (also known as the second-order cone, Lorentz cone or ice cream cone) of appropriate size. That is, if $v \in \mathbb{R}^k$, then $v \geq_q 0 \Leftrightarrow v_1 \geq \|v_{2:k}\|_2$.
  - $w \geq 0$ means that the vector w is componentwise nonnegative.
- **The inner product $\langle ., . \rangle$:** For $x, z \in K$

$$\langle x, z \rangle = \sum_{j=1}^{n_s} x^s_j \bullet z^s_j + \sum_{k=1}^{n_q} x^q_k * z^q_k + x^\ell * z^\ell,$$

  where
  - For matrices a and b, $a \bullet b = \text{trace } a^T b = \sum_{i,j} a_{ij} b_{ij}$.
  - For vectors a and b, $a * b = a^T b = \sum_i a_i b_i$.

In the following a_{ij} and a_i denote the respective matrix and vector parts of the operators $\mathcal{A}$ and $\mathcal{A}^*$. Thus (P) and (D) become

$$\begin{array}{lll} \min & \sum_{j=1}^{n_s} c^s_j \bullet x^s_j & + \sum_{k=1}^{n_q} c^q_k * x^q_k \quad + c^\ell * x^\ell \\ \text{s.t.} & \sum_{j=1}^{n_s} a^s_{ij} \bullet x^s_j & + \sum_{k=1}^{n_q} a^q_{ik} * x^q_k + a^\ell_i * x^\ell = b_i \; \forall i \\ & x^s_j \succeq 0 \; \forall j & x^q_k \geq_q 0 \; \forall k \qquad x^\ell \geq 0 \end{array} \qquad (SQLP - P)$$

and

$$\begin{array}{lllll} \max & \sum_{i=1}^m b_i y_i & & & \\ \text{s.t.} & \sum_{i=1}^m a^s_{ij} y_i + z^s_j & & & = c^s_j \; \forall j \\ & \sum_{i=1}^m a^q_{ij} y_i & + z^q_k & & = c^q_j \; \forall k \\ & \sum_{i=1}^m a^\ell_i y_i & & + z^\ell & = c^\ell \\ & & z^s_j \succeq 0 \quad z^q_k \geq_q 0 & \quad z^\ell \geq 0 & \end{array} \qquad (SQLP - D)$$

Thus the feasible set is a product of semidefinite, quadratic and nonnegative orthant cones, intersected with an affine subspace. It is possible that one or more of the three parts of the problem is absent, i.e., any of n_s, n_q, or the length of x^ℓ may be zero.

The *rotated quadratic cone* is defined as

$$\{ \, v \in \mathbb{R}^k \mid v_1 v_2 \geq \|v_{3:k}\| \, \}.$$

It is simply a rotation of the usual quadratic cone, but for the purpose of modeling quadratic inequalities, it is often more convenient to use, thus several participating codes support this cone.

1.2 Computing Errors

Suppose that we are given *approximate* optimal solutions of $(SQLP-P)$ and $(SQLP-D)$, respectively. To compute how far they are from an exact solution pair, we define a norm on X, and a minimum eigenvalue with respect to the cone K. Precisely, if $x \in X$, then

$$\|x\| = \sum_{j=1}^{n_s} \|x_j^s\|_F \ + \ \sum_{k=1}^{n_q} \|x_k^q\|_2 \ + \ \|x^\ell\|_2,$$

where for a matrix a, $\|a\|_F$ is the Frobenius norm of a. Also,

$$\lambda_{\min,K}(x) = \min\left\{ \min_{j=1,\ldots,n_s} \lambda_{\min}(x_j^s),\ \min_{k=1,\ldots,n_q} \lambda_{\min,q}(x_k^q),\ \min_h x_h^\ell \right\}$$

where

- for a symmetric matrix a, $\lambda_{\min}(a)$ is the usual smallest eigenvalue of a.
- for a vector a, $\lambda_{\min,q}(a) = a_1 - \|a_{2:k}\|_2$.

Also, we denote by $\|a\|_1$ the absolute value of the largest component of a, for an arbitrary $a \in X$.

Then, for approximate optimal solutions x of $(SQLP-P)$ and (y,z) of $(SQLP-D)$, we define

$$\begin{aligned}
\mathrm{err}_1(x,y,z) &= \frac{\|\mathcal{A}x - b\|_2}{1 + \|b\|_1} \\
\mathrm{err}_2(x,y,z) &= \max\left\{0, \frac{-\lambda_{\min,K}(x)}{1 + \|b\|_1}\right\} \\
\mathrm{err}_3(x,y,z) &= \frac{\|\mathcal{A}^* y + z - c\|_2}{1 + \|c\|_1} \\
\mathrm{err}_4(x,y,z) &= \max\left\{0, \frac{-\lambda_{\min,K}(z)}{1 + \|c\|_1}\right\} \\
\mathrm{err}_5(x,y,z) &= \frac{\langle c, x\rangle - b^T y}{1 + |\langle c, x\rangle| + |b^T y|}
\end{aligned}$$

Furthermore, when x and z are both in K, that is $\mathrm{err}_2(x,y,z) = \mathrm{err}_4(x,y,z) = 0$, we also define

$$\mathrm{err}_6(x,y,z) = \frac{\langle x, z\rangle}{1 + |\langle c, x\rangle| + |b^T y|}$$

A few remarks are in order.

- If x and z are both feasible, then in exact arithmetic $\mathrm{err}_5(x,y,z) = \mathrm{err}_6(x,y,z)$.

- As x and z are *approximate* optimal solutions only, we may have $\mathrm{err}_5(x,y,z) < 0$. It is possible that, all other error measures being the same, if $\mathrm{err}_5(x,y,z) = -\delta$ with $\delta > 0$, then (x,y,z) corresponds to a solution that is "worse", than if $\mathrm{err}_5(x,y,z)$ were δ. Thus, we decided to report $\mathrm{err}_5(x,y,z)$, not merely the maximum of $\mathrm{err}_5(x,y,z)$ and 0 (as done in several papers), so as not to suppress any information.
- Several codes do not explicitly maintain z; in this case, one should set

$$z = c - \mathcal{A}^* y$$

  Of course, then $\mathrm{err}_3(x,y,z)$ will be zero (depending on the accuracy achieved by the computer).

The above six error measures were introduced in [18]. They are evaluated by most conic codes and their output, in addition to any solver-specific measures, is routine or may be activated (SDPA).

The evaluation of conic optimization software was continued after [18] in the webpage [20]. This comparison reflects in mid-2010 the status of mid-2008 with a few exceptions. The reason is that the software considered already in [20] has practically not undergone any major algorithmic development. Nevertheless, it is planned to update the conic part of [20] within the next twelve months when some major developments are expected, notably the rewriting of DSDP, the future development of SMCP, and extensions such as that of MOSEK to SDP. One major code improvement that has taken place lately is documented separately, namely the parallelization and extension to multiple precision of SDPA [11].

In the following we devote one paragraph to each of the codes similar to [18]. These have partly been coordinated with the authors. A code not included but part of the benchmarks [20] is PENSDP. It is one of the very few codes able to solve nonlinear SDP problems. A separate article in this volume is devoted to the code PENNON which PENSDP is a part of.

2 The Codes

The submitted codes fall naturally into two major groups and several subgroups:

- The first major group is that of primal-dual interior point methods designed for small to medium sized problems.
  - In the first subgroup are the codes SeDuMi, SDPT3, and SMCP. These codes handle all 3 types of cones.
  - In the second subgroup are SDPA and CSDP, which are limited to SDP.
  - In the third subgroup are codes that had started out as LP codes, CPLEX, LOQO, and MOSEK. They were extended to QP, SOCP, and convex (MOSEK) or nonconvex (LOQO) nonlinear optimization.

- The second group is that of large-scale SDP codes designed to provide approximate solutions for large scale problems: SDPLR, ConicBundle, and DSDP. The first two of these do not make use of second order derivative information, while DSDP is a dual interior point code.

  The major input formats are:

  Matlab: SeDuMi format in Matlab binary form [24].
  QPS: extended MPS format, different for MOSEK and CPLEX
  SDPA: the sparse SDPA format as explained in the SDPA user's guide.
  Language: Codes that have no specific input format (ConicBundle); other codes can be used from one or more languages as indicated.

2.1 SDPA

Authors: K. Fujisawa, M. Fukuda, Y. Futakata, K. Kobayashi, M. Kojima, K. Nakata, M. Nakata, and M. Yamashita
Version 7.3.1, 7/2009
Available: yes; at `http://sdpa.indsys.chuo-u.ac.jp/sdpa/`
Key paper: [11]. See [27] for implementation and numerical results.
Features: primal-dual method
Language; Input format: C++; SDPA, Matlab, SeDuMi
Error computations: yes
Solves: SDP
SDPA (SemiDefinite Programming Algorithm) is based on a Mehrotra-type predictor-corrector infeasible primal-dual interior-point method with the H..K..M direction. It is implemented in C++ language utilizing BLAS/LAPACK libraries for dense matrix computations and MUMPS for sparse Cholesky factorizations. SDPA exploits sparsity of data input matrices and employs multithreading computing to solve semidefinite programs in practice in a short time. SDPA is available as a callable library, too. In addition, it supports a Matlab interface and accepts SDPA or SeDuMi input formats.

SDPA uses a set of parameters which the user can adjust to cope with numerically difficult semidefinite programs. Stopping criteria:

$$- \min\{||\mathcal{A}x - b||_\infty, ||\mathcal{A}^* y + z - c||_\infty\} \leq \epsilon' \quad \text{and}$$
$$- \frac{|\langle c, x\rangle - b^T y|}{\max\{(|\langle c, x\rangle| + |b^T y|)/2.0, 1.0\}} \leq \epsilon^*$$

Typical values of the parameters ϵ' and ϵ^* are from 10^{-8} to 10^{-6}.
Remarks: In addition, SDPA has some variants which all together are named SDPA Family. SDPA-GMP is an implementation of the algorithms with arbitrary accuracy libraries and enables to attain $\epsilon' = \epsilon^* = 10^{-30}$, for example. A parallel version, SDPARA, solves extremely large-scale semidefinite programs on multiple PCs connected by high-speed networks.

The contribution by the SDPA authors in this volume will introduce each software of the SDPA Family.

2.2 SeDuMi

Authors: J. Sturm, I. Polik
Version: 1.3, 4/2010
Available: yes; from `http://sedumi.ie.lehigh.edu/`
Key papers: [24, 25]
Features: self-dual embedding, dense column handling
Language; Input format: Matlab+C; Matlab, SDPA
Error computations: yes
Solves: SDP, SOCP

The primal-dual interior point algorithm implemented in SeDuMi [24] is described in [25]. The algorithm has an $O(\sqrt{n}\log\epsilon)$ worst case bound, and treats initialization issues by means of the self-dual embedding technique of [28]. The iterative solutions are updated in a product form, which makes it possible to provide highly accurate solutions.

The algorithm terminates successfully if the norm of the residuals in the optimality conditions, or the Farkas system with $b^T y = 1$ or $\langle c, x\rangle = -1$, are less than the parameter `pars.eps`. The default value for `pars.eps` is `1E-9`.
Remarks:

- SeDuMi exploits sparsity in solving the normal equations; this results in a benefit for problems with a large number of small order matrix variables, such as the *copositivity*-problems in the DIMACS set.
- However, for problems that involve a huge matrix variable (without a block diagonal structure), the implementation is slow and consumes an excessive amount of memory.

2.3 CSDP

Author: B. Borchers
Version 6.1.1, 4/2010
Available: yes; from `http://projects.coin-or.org/Csdp/`
Key paper: [6]
Features: infeasible predictor-corrector variant of a primal dual method based on the H..K..M direction
Language; Input format: C; SDPA, Interfaces to Matlab/Octave and R
Error computations: yes
Solves: SDP

CSDP consists of a callable library and standalone solver for SDP problems. It is not applicable to problems with second order cone constraints. The code uses a predictor–corrector variant of the primal–dual method of Helmberg, Rendl, Vanderbei, and Wolkowicz [14] and Kojima, Shindoh, and Hara [17]. CSDP is suited to the solution of small and medium size SDPs with general structure. The algorithm supports matrices with block diagonal structure as well as linear programming variables which are expressed as a diagonal block within the SDP problem.

CSDP is written in portable ANSI C with calls to subroutines from either the Linpack or LAPACK libraries. The required Linpack routines are supplied with the code. However, improved performance can be obtained by using BLAS and LAPACK libraries that have been optimized for a particular computer.

The default stopping criteria are:

- $\frac{|\langle c,x\rangle - b^T y|}{1+|b^T y|} < 10^{-7}$ and
- $\frac{\|\mathcal{A}x - b\|_2}{1+\|b\|_2} < 10^{-8}$ and
- $\frac{\|\mathcal{A}^* y - c + z\|_2}{1+\|c\|_2} < 10^{-8}$.

In addition, CSDP maintains positive definite X and Z matrices at all times. CSDP checks this by computing the Cholesky factorizations of X and Z.

CSDP uses OPENMP to parallelize on shared memory multiprocessor systems; see [7]

2.4 DSDP

Authors: S. Benson, Y. Ye
Version: 5.8, 10/2005
Available: yes; from `http://www.mcs.anl.gov/hs/software/DSDP/`
Key paper: [5]
Features: Dual scaling potential reduction method, rank-1 constraints, Matlab interface, MPI parallel
Language; Input format: C; SDPA
Error computations: yes
Solves: SDP

The DSDP software package is an implementation of the dual scaling algorithm for SDP. Unlike primal-dual interior point methods, this algorithm uses only the dual solution to generate a step direction. It can generate feasible primal solutions as well, but it saves time and memory if this feature is not used. Many large problems have well structured constraints in the form of low rank matrices or sparse matrices. DSDP explicity accounts for these features by using sparse data structures for the dual matrix and the eigenvalue/eigenvector decomposition of the data matrices. Theoretical convergence results exist for feasible starting points, and strong computational results have been achieved using feasible and infeasible starting points. DSDP initially solves the Schur complement equations using the conjugate residual method, and then it switches to the Cholesky method when the conditioning of the matrix worsens.
Stopping criteria:

$$|\langle c, x\rangle - b^T y| \,/(|b^T y| + 1) \leq 10^{-3} \quad \text{and}$$
$$\|\mathcal{A}^* y + z - c\|_\infty \leq 10^{-8}$$

or

$$|b^T y| \geq 10^8 \quad \text{(unbounded dual)}$$

or

stepsizes less than 10^{-3} for more than 10 iterations (dual infeasibility)

In the summer of 2010 a new version will be released based on a hybrid of Quasi-Newton and Newton methods with a Hessian-Update technique.

2.5 SDPT3

Authors: K.C. Toh, M. Todd, R. Tütüncü
Version: 4.0, 2/2009
Available: yes; from `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`
Key paper: [26]
Features: primal-dual method, infeasible primal-dual and homogeneous self-dual formulations, Lanczos steplength computation
Language; Input format: Matlab+C or Fortran; SDPA
Error computations: yes
Solves: SDP, SOCP

This code is designed to solve conic programming problems whose constraint cone is a product of semidefinite cones, second-order cones, and/or non-negative orthants. It employs a predictor-corrector primal-dual path-following method, with either the H..K..M or the NT search direction. The basic code is written in Matlab, but key subroutines in Fortran and C are incorporated via Mex files. Routines are provided to read in problems in either SeDuMi or SDPA format. Sparsity and block diagonal structure are exploited, but the latter needs to be given explicitly.

The algorithm is stopped if:

- primal infeasibility is suggested because $b^T y/\|\mathcal{A}^* y + z\| > 10^8$; or
- dual infeasibility is suggested because $-\langle c, x\rangle/\|\mathcal{A}x\| > 10^8$; or
- sufficiently accurate solutions have been obtained:

  $$\text{rel\_gap} := \frac{\langle x, z\rangle}{\max\{1, (\langle c, x\rangle + b^T y)/2\}}$$

  and

  $$\text{infeas\_meas} := \max\left[\frac{\|\mathcal{A}x - b\|}{\max\{1, \|b\|\}}, \frac{\|\mathcal{A}^* y + z - c\|}{\max\{1, \|c\|\}}\right]$$

  are both below 10^{-8}; or
- slow progress is detected, measured by a rather complicated set of tests including

  $$\langle x, z\rangle/n < 10^{-4} \quad \text{and} \quad \text{rel\_gap} < 5 * \text{infeas\_meas};$$

  or

- numerical problems are encountered, such as the iterates not being positive definite, the Schur complement matrix not being positive definite, or the step sizes falling below 10^{-6}.

Remarks:

- SDPT3 is a general-purpose code based on a polynomial-time interior-point method.
- It should obtain reasonably accurate solutions to problems of small and medium size (for problems with semidefinite constraints, up to around a thousand constraints involving matrices of order up to around a thousand, and for sparse problems with only second-order/linear cones, up to around 20,000 constraints and 50,000 variables), and can solve some larger problems.
- Because it uses a primal-dual strategy, forms the Schur complement matrix for the Newton equations, and employs direct methods, it is unlikely to compete favorably with alternative methods on large-scale problems.

New Features:

- free variables
- determinant maximization problems
- SDP with complex data
- 3-parameter homogeneous self-dual model of SQLP

2.6 MOSEK

Author: E. Andersen
Version: 6.0, 7/2010
Available: yes; from `http://www.mosek.com/`
Key paper: [2]
Features: Solves large scale sparse SOCPs. Is parallelized. Solves mixed-integer SOCPs
Interfaces: C, Java, .NET, MATLAB, Python
Modeling Language interfaces: GAMS, AIMMS, AMPL
Input formats: OPF, Extended MPS
Error computations: yes
Solves: SOCP

The conic quadratic optimizer implemented in MOSEK employs the NT search direction. The other main features of the implementation are that it is based on a homogeneous and self-dual model, handles the rotated quadratic cone directly, employs a Mehrotra type predictor-corrector extension and sparse linear algebra to improve the computational efficiency.

Other features:

- Has an extensive presolve facility to reduce problem size before optimizing
- Exploits special structure in rotated quadratic cones

- Exploits fixed and upper bounded variables
- Detects primal and dual infeasibility reliably

Future plans: Support for SDPs

2.7 LOQO

Authors: H.Y. Benson, R. Vanderbei
Version: 6.07, 9/2006
Available: yes; from `http://www.princeton.edu/~rvdb/`
Key paper: [4]
Features: NLP approach
Language; Input format: C; SDPA, Matlab, AMPL
Error computations: no
Solves: SOCP

LOQO is a software package for solving general (smooth) nonlinear optimization problems of the form

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) = 0, \quad i \in \mathcal{E} \\ & h_i(x) \geq 0, \quad i \in \mathcal{I}, \end{array}$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $\mathcal{E}$ is the set of equalities, $\mathcal{I}$ is the set of inequalities, $g : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{E}|}$, and $h : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{I}|}$. It implements an infeasible-primal-dual path-following method and requires that the problem be smooth, that is f and h be twice differentiable, and g be an affine function. Even though LOQO can handle nonconvex problems in general, it performs better with convex problems, where f is convex, g are affine, and h are concave functions.

Stopping criteria

$$\begin{aligned} \|\mathcal{A}x - b\|_2 &\leq 10^{-7} \\ \|\mathcal{A}^* y + z - c\|_2 &\leq 10^{-8} \\ \log_{10}\left(\frac{|\langle c, x\rangle - b^T y|}{|\langle c, x\rangle| + 1}\right) &\leq -8 \end{aligned}$$

- LOQO can handle other types of nonlinear constraints in the problem.
- A weakness results from the use of the Cholesky factorization. LOQO works best with a sparse problem, and it does not exploit the special structure of an SOCP.
- Its SDP approach leads to dense problems.

2.8 SDPLR

Authors: S. Burer, R. Monteiro
Version: 1.03beta, 6/2009

Available: yes; from `http://dollar.biz.uiowa.edu/~sburer/doku.php`
Key paper: [8]
Features: Augmented Lagrangian algorithm applied to an NLP formulation of an SDP arising by replacing the primal variable by a low-rank factorization.
Language; Input format: C; SDPA, SDPLR
Error computations: yes
Solves: SDP

The code SDPLR is an infeasible, nonlinear programming method for solving any standard form primal SDP.

The main idea of SDPLR is to eliminate the primal positive semidefiniteness constraint $X \succeq 0$ by employing an implicit factorization $X = RR^T$ that is valid for at least one optimal solution (but not necessarily for all feasible solutions). Using a theorem of Pataki, R is chosen to be an $n \times r$ matrix, where r is the smallest integer satisfying $r(r+1)/2 \geq m$ and m is the number of primal constraints (aside from the constraint $X \succeq 0$). A stationary point $\bar{R}$ of the new nonlinear problem is then found using a first-order augmented Lagrangian method, and in practice, such a stationary point reliably gives an optimal SDP solution $\bar{X} = \bar{R}\bar{R}^T$.

SDPLR is an infeasible method that is terminated once the norm of the primal infeasibility has been reduced to a value of 10^{-6}, which, in accordance with the augmented Lagrangian algorithm, indicates an approximate stationary point.

- The main strength of SDPLR is that it is a first-order algorithm and hence can attack large-scale SDPs.
- The infeasible nature of SDPLR is a strong disadvantage.
- SDPLR does have a convergence theory that explains its practical convergence, though the theory is based on some assumptions of sequences, etc, and is not as strong as, for example, the theory of regular interior-point methods. The theory is contained in [9].
- SDPLR works with density of the dual matrix S and moreover works with a small number of columns r . As a result, the iterations of SDPLR are extremely fast.

2.9 ConicBundle

Authors: C. Helmberg
Version: 0.3.5, 4/2010
Available: yes; from `http://www.tu-chemnitz.de/~helmberg/ConicBundle`
Key papers: [15, 16, 13, 3]
Features: nonsmooth convex optimization with special purpose models for Second Order Cone and Semidefinite Optimization and additional support for Lagrangian relaxation/decomposition
Language; Input format: C++; C++
Error computations: no; only norm of an aggregate subgradient

Solves: Convex optimization problems given by first order oracles.
Special oracles and models are provided for eigenvalue optimization problems of the form

$$\min y \in \mathbb{R}^m a\lambda_{\max}(C - \sum_{i=1}^{m} A_i y_i) + b^T y$$

The design variables y_i may be box constrained, C and the A_i are given real symmetric matrices, $b \in \mathbb{R}^m$ allows to specify a linear cost term, and $a\langle 0$ is a constant multiplier for the maximum eigenvalue function $\lambda_{\max}(\cdot)$. The code is intended for large scale problems and allows to exploit structural properties of the matrices such as sparsity and low rank structure. From a Lagragian relaxation perspective, the code solves the dual to programs of the form $\max\{< C, X >: X \succeq 0, < A_i, X >= b_i, i = 1, \ldots, m\}$ with bounded trace, $\mathrm{tr}X \leq a$, provides support for several positive matrix variabls as well as for adding cutting planes and accumulates solution information to form a sparse or primal aggregate X^* that may be used as an approximation to an optimal X. The code is a dynamic bundle method constructing a minorizing model of the Lagrangian dual based on subgradient information in the style of the proximal bundle method of [17]. Function values and subgradients of the nonsmooth convex function $f(y) := a\lambda_{\max}(C - \sum_{i=1}^{m} A_i y_i) + b^T y$ are determined via computing the maximum eigenvalue and a corresponding eigenvector by a Lanczos method. The code never factorizes the matrix $C - \sum_{i=1}^{m} A_i y_i$ but uses matrix vector multiplications exclusively. Thus, there is no danger of increasing memory requirements or computational work by additional fill-in. Experimentally, the method seems to exhibit fast convergence if the semidefinite subcone that is used to generate the semidefinite cutting surface model, spans the optimal solution of the corresponding primal problem. If the subcone, however, is too small, the typical tailing off effect of subgradient methods is observed once a relative precision of roughly 10^{-3} has been reached.

2.10 SMCP

Authors: E. Andersen, L. Vandenberghe
Version: 0.3a, 5/2010
Available: yes; from `http://abel.ee.ucla.edu/smcp/`
Key papers: [1]
Features: primal and dual scaling methods for sparse matrix cone programs
Language; Input format: C; Python, SDPA or CVXOPT matrix
Error computations: yes
Solves: SDP
SMCP implements feasible start primal and dual barrier methods for sparse matrix cone programs (which include SDP, SOCP, and LP as special cases). The algorithm takes advantage of chordal sparsity when solving the Newton equations. It is implemented in Python/C and relies on the Python extensions

CVXOPT and CHOMPACK for linear algebra and chordal matrix computations.

Stopping criteria:

$$\frac{\|b - \mathcal{A}x\|_2}{\max\{1, \|b\|_2\}} \leq \epsilon_{\text{feas}}, \qquad \frac{\|\mathcal{A}^{\text{adj}}y + z - z\|_F}{\max\{1, \|c\|_F\}} \leq \epsilon_{\text{feas}}, \qquad x \succ_{K^*} 0, \qquad z \succ_K$$

and

$$\langle x, z\rangle \leq \epsilon_{\text{abs}}, \quad \left(\min\{\langle c, x, -b^T y\rangle\} \leq 0, \frac{\langle x, z\rangle}{-\min\{\langle c, x, -b^T y\rangle\}}\right) \leq \epsilon_{\text{rel}}$$

where $\epsilon_{\text{feas}} = 10^{-8}$ and $\epsilon_{\text{rel}} = \epsilon_{\text{abs}} = 10^{-6}$ are the default tolerances.

2.11 CPLEX

Authors: IBM
Version: 12.2, 6/2010
Available: yes; from `http://www.ibm.com (academic license)`
Key papers: see documentation
Features: specialized SOCP solver
Language: C, C++, C#, VB.net, Java, Matlab, Microsoft Excel, Python
Input format: extended MPS
Error computations: yes
Solves: standard and rotated SOCP versions
In addition to linear and quadratic progams, and their mixed integer versions, IBM ILOG CPLEX also solves programs involving standard and rotated cones, and convex quadratic constraints, both the continuous and the mixed integer versions.

Rotated cones and convex quadratic constraints are handled by transforming them into standard cone form. The standard cone form is then solved by a specialized SOCP solver. This solver is based on the homogeneous self dual formulation and Nesterov-Todd scaling. The formulation is solved in scaled space. A Cholesky factorization is used to solve the normal system and dense columns are handled in product form. The solver uses the predictor corrector technique.

The code is written C, and uses very efficient techniques for performing sparse linear algebra.

The stopping criterion is based on the primal, dual and relative gap error. It is controlled by the qcpconvergence tolerance parameter which is 10^{-7} by default. The quality of the solutions can be displayed using the IBM ILOG CPLEX solution quality queries.

3 Comments

While no tables with comparisons as in [18] will be given here, a few remarks are in order for a potential user of one of the codes. Based on the current status

for SDP problems CSDP, SeDuMi, SDPA, and SDPT3 are all quite stable. SMCP will be extended from its current feasible version but is partly already competitive as shown in [1]. We do not comment here on DSDP since a rewrite of it is imminent. Parallelization, especially of SDPA for both distributed and, now becoming very important, for shared memory, leads to a significant speed up. Also, the many ill-conditioned SDPs are currently solved best with the multiple precision versions of SDPA. They have been installed by us for use through [22] and also have been instrumental in our recent research [21], [12].

For problems of special structure or more general than SDP/SOCP, the codes SDPLR, PENNON and ConicBundle are very useful. For some larger SDPs one may also consider a Newton-augmented Lagrangian code with iterative linear algebra, NCGAL [29]. LOQO is not quite competitive for conic optimization problems while for SOCP the other applicable codes as the corresponding benchmark in [20] shows perform quite comparably and, different from the situation for SDP, even for larger problem sizes.

Finally, no mention is made here of the meanwhile large pool of application-specific conic software, partly by the same authors, such as for sensor network localization, sparse minimization etc.

For additional information we refer to the related literature, for example, to a survey article with a similar scope [23] as well as to several articles in this volume.

References

1. Andersen, M. S., J. Dahl, and L. Vandenberghe: Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. Submitted to Mathematical Programming Computation
2. Andersen, E. D., Roos, C., Terlaky, T.: On implementing a primal-dual interior-point method for conic quadratic optimization. Mathematical Programming (series B), **95** 249–277 (2003)
3. Armbruster, M., Fügenschuh, M., Helmberg, C., Martin, A.: A Comparative Study of Linear and Semidefinite Branch-and-Cut Methods for Solving the Minimum Graph Bisection Problem. in Integer Programming and Combinatorial Optimization, A. Lodi, A. Panconesi, G. Rinaldi eds., Lecture Notes in Computer Science, Springer, 112–124 (2008)
4. Benson, H.Y., Vanderbei, R.J.: Solving Problems with Semidefinite and Related Constraints Using Interior-Point Methods for Nonlinear Programming. Mathematical Programming Ser. B, **95**, 279–302 (2003)
5. Benson, S.J, Ye,, Y.: Algorithm 875: DSDP5—software for semidefinite programming. ACM Trans. Math. Software 34 no. 3, Art. 16 (2008)
6. Borchers, B.: CSDP, A C library for semidefinite programming. Optimization Methods and Software **11** 613–623 (1999)
7. Borchers, B., Young, J.: Implementation of a Primal-Dual Method for SDP on a Shared Memory Parallel Architecture. Comp. Opt. Appl. **37**, 355–369 (2007)
8. Burer, S., Monteiro, R.D.C.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Mathematical Programming (series B), **95** 329–357 (2003)

9. Burer, S., Monteiro, R.D.C.: Local minima and convergence in low-rank semidefinite programming. Mathematical Programming (series A), **103** 427–444 (2005)
10. DIMACS 7th Challenge website, `http://dimacs.rutgers.edu/Challenges/Seventh/`
11. Fujisawa, K., Fukuda, M., Kobayashi, K., Kojima, M., Nakata, K., Nakata, M., Yamashita, M.: SDPA (SemiDefinite Programming Algorithm) User's Manual — Version 7.0.5 Research Report B-448, Dept. of Math. and Comp. Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, January 2010.
12. Gijswijt, D. C., Mittelmann, H. D., Schrijver, A. Semidefinite code bounds based on quadruple distances, submitted
13. Helmberg, C.: A Cutting Plane Algorithm for Large Scale Semidefinite Relaxations. in The Sharpest Cut, ed. M. Grötschel, MPS-SIAM Series on Optimization, 233–256 (2004)
14. Helmberg, C., Rendl, F., Vanderbei, R.J., Wolkowicz, H.: An interior-point method for semidefinite programming, SIAM Journal on Optimization **6**, 342–361 (1996)
15. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM Journal on Optimization **10**, 673–696 (2000)
16. Helmberg, C., Kiwiel, K. C.: A Spectral Bundle Method with Bounds. Mathematical Programming Ser. B, **93**, 173–194 (2002)
17. Kojima, M., Shindoh, S., Hara, S.: Interior-point methods for the monotone semidefinite linear complementarity problems. SIAM Journal on Optimization **7**, 86–125 (1997)
18. Mittelmann, H. D.: An independent benchmarking of SDP and SOCP solvers. Mathematical Programming (series B), **95** 407–430 (2003)
19. Mittelmann, H.D.: Decision Tree for Optimization Software. `http://plato.la.asu.edu/guide.html`
20. Mittelmann, H.D.: Benchmarks for Optimization Software. `http://plato.la.asu.edu/bench.html`
21. Mittelmann, H. D., Vallentin, F.: High Accuracy Semidefinite Programming Bounds for Kissing Numbers. to appear in Experimental Mathematics (2010)
22. NEOS Server for Optimization. `http://www-neos.mcs.anl.gov/neos/`
23. Polik, I.: Conic optimization software. In Encyclopedia of Operations Research, Wiley (2010)
24. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software **11**, 625–653 (1999)
25. Sturm, J.F.: Central region method, in High Performance Optimization, J.B.G. Frenk and C. Roos and T. Terlaky and S. Zhang (eds.), Kluwer Academic Publishers, 157–194 (2000)
26. Tütüncü, R.H., Toh, K.C., Todd: Solving semidefinite-quadratic-linear programs using SDPT3. Mathematical Programming Ser. B, **95**, 189–217 (2003)
27. Yamashita, M., Fujisawa, K., Nakata, K., Nakata, M., Fukuda, M., Kobayashi, K., Goto, K.: A high-performance software package for semidefinite programs: SDPA 7. Research Report B-460, Dept. of Math. and Comp. Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, January 2010.
28. Ye, Y., Todd, M.J., Mizuno, S.: An $O(\sqrt{n}L)$-iteration homogeneous and self–dual linear programming algorithm. Mathematics of Operations Research **19**, 53–67 (1994)

29. Zhao, X.-Y., Sun, D., Toh, K.-C.: A Newton-CG Augmented Lagrangian Method for Semidefinite Programming. SIAM Journal on Optimization **20**, 1737–1765 (2010)