CBLIB 2014: A benchmark library for conic mixed-integer and continuous optimization

Henrik A. Friberg*

March 7, 2014

Abstract

The Conic Benchmark Library (CBLIB 2014) is a collection of more than a hundred conic optimization instances under a free and open license policy. It is the first extensive benchmark library for the advancing field of conic mixed-integer and continuous optimization, which is already supported by all major commercial solvers and spans a wide range of industrial applications. The library addresses the particular need for public test sets mixing cone types and allowing integer variables, but has all types of conic optimization in target. The CBF file format is embraced as standard, and tools are provided to aid integration with, or transformation to the input format of, any software package.

1 Introduction

A conic optimization problem is the problem of minimizing (or maximizing) a linear objective over a feasible region specified in terms of affine expressions, convex cones, and, if any, integer variables. It may be formulated as

$$\begin{array}{lll} \underset{x}{\text{minimize}} & c^T x & \\ \text{subject to} & A_i x - b_i \in \mathcal{K}_i, & \text{for } i = 1, \ldots, k, \\ & x_j \in \mathbb{Z}, & \text{for } j \in \mathcal{J}. \end{array} \tag{1}$$

The conic form (1) allows us to express all convex mixed-integer and continuous optimization problems without loss of generality [19]. In fact, only three types of cones (nonnegative orthant, quadratic cone and semidefinite cone) are enough to represent a broad range of applications [3]. These three cone types are called the real-valued symmetric cones, and are usually accompanied by equality constraints for convenience, which in (1) would be the cone of zeros $\{0\}^n$. As example, we take the classical Markowitz portfolio optimization problem [29] and compare the traditional and the conic form. The portfolio problem (2)

*PhD candidate at the Technical University of Denmark, Department of Wind Energy, and employee at MOSEK ApS. E-mail: haf@mosek.com.

maximizes expected return subject to the accepted risk γ and investable wealth ω. The return vector μ and covariance matrix $\Sigma = G^T G$, characterize the investments in consideration.

$$\begin{array}{lrcll} \underset{x}{\text{maximize}} & \mu^T x & & & \\ \text{subject to} & x^T \Sigma x & \leq & \gamma & \leftrightarrow \\ & e^T x & = & \omega & \leftrightarrow \\ & x & \geq & 0 & \leftrightarrow \end{array} \qquad \begin{array}{lrcl} \underset{x}{\text{maximize}} & \mu^T x & & \\ \text{subject to} & (\gamma^{1/2},\, Gx) & \in & \mathcal{Q}^{1+n}, \\ & e^T x - \omega & \in & \{0\}, \\ & x & \in & \mathbb{R}^n_+. \end{array} \tag{2}$$

The traditional form (on the left) use a convex functional to express the nonlinearity as $f(x) \leq 0$. The conic form (on the right) achieves the same using a nonlinear cone. In particular, the three affine expressions are constrained, respectively, to the quadratic cone $\mathcal{Q}^{1+n} = \{(r, x) \in \mathbb{R}^1_+ \times \mathbb{R}^n \mid r^2 \geq x^T x\}$, and the two linear cones, the set of zero and the nonnegative orthant.

One advantage of the conic form is that convexity is not a concern. Any input data yields a convex problem instance by definition, whereas the convexity of the traditional form depends on the parameters of $f(x)$, such as Σ in (2). Another advantage stems from the efficiency by which primal-dual interior-point methods are able to exploit the underlying structure of symmetric cones [32]. This advantage is reflected in the state-of-the-art optimization software, with high-performing implementations in all major commercial solvers; XPRESS [14], MOSEK [31], GUROBI [21] and CPLEX [23]. SEDUMI [39], SDPT3 [42] and SDPA [44] can moreover be mentioned as significant open source projects based on variants of the primal-dual method for symmetric cones proposed in [32]. MOSEK, SEDUMI and SDPT3 support all real-valued symmetric cones, while XPRESS, GUROBI and CPLEX omit support of the semidefinite cone. SDPA, in contrast, support only the semidefinite cone. Integer variables can be handled by all major commercial solvers, but not by any of the open source projects.

What is currently missing from this development, is an extensive, challenging and publicly available benchmark library keeping up with mainstream cone support. Benchmark libraries are known to have a great effect on stimulating improvements in reliability and performance in optimization software. The *NETLIB LP* [18] library, for instance, was the first electronically distributed benchmark library for continuous linear optimization and often attributed for its major effect on the development of LP solvers. Correspondingly, *MIPLIB* [28] has played a major role in the field of mixed-integer linear optimization. In review of benchmark libraries for conic optimization, *SDPLIB* [5] is worth noticing although focus is limited to the semidefinite cone. Mixed cone types were considered in the *7th DIMACS Implementation Challenge* [34], but the benchmark library established for this challenge has been inactive for years. These instances are furthermore difficult to use without MATLAB [40], and were reformulated at the time to fit a SEDUMI biased format. No benchmark libraries were found for conic mixed-integer optimization, although supported by all major commercial optimization software available today. This reveals a particular need to address mixed cone types and integer variables.

The Conic Benchmark Library (CBLIB) is an ongoing community developed project staying updated with the conic mixed-integer and continuous capabilities of mainstream solvers. Mixed cone types and integer variables fall under this category and has, among other things, been addressed by the library. CBLIB 2014 is the initial release with more than a hundred instances of varying difficulty. News and changes can be followed on `http://cblib.zib.de`, from where the project is being maintained and distributed.

The benchmark library embraces the recently developed Conic Benchmark Format (CBF) [17] as the standard file format for conic optimization problems. As opposed to all considered alternatives, CBF has the advantageous of being a portable text-based format, compact and easily parsable, and not biased towards the capabilities of any particular solver. It gracefully handles the concept of matrix-variables and matrix-inequalities, and is versatile enough to encompass other convex cones should such become part of the mainstream solver capabilities. Moreover, the CBF file format allows CBLIB to provide instances with free variables and change-sequences in problem data (where warmstarting is advantages), both of which represent ongoing computational studies of interior-point methods [2, 37].

The article is outlined as follows. In Section 2, the conic problem formulation used in CBLIB is formalized. Verification of the claims made by solvers (such as feasibility and optimality) is discussed, and the CBF file format is exemplified. Section 3 describes all of the problem instances in this benchmark library, as well as the tools distributed with them. A brief note on benchmarking is made in Section 4, and final remarks are made in Section 5.

Notation and cone definitions

The notation in this section uses $x = [x]^+ - [x]^-$ as the decomposition of a vector into its nonnegative and nonpositive parts. That is, element-wise, $[x]^+_j = \max(x_j, 0)$ and $[x]^-_j = \max(-x_j, 0)$. We use $\mathcal{S}^n \subset \mathbb{R}^{n\times n}$ as the subset of symmetric matrices, and $\langle X, Y\rangle = \sum_{ij} X_{ij}Y_{ij}$ as the standard trace inner product for such matrices. The Cartesian product, $\times$, is defined to satisfy

$$x_1 \in \mathcal{K}_{x_1},\ x_2 \in \mathcal{K}_{x_2} \quad \Longleftrightarrow \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathcal{K}_{x_1} \times \mathcal{K}_{x_2},$$

and a cone which is not the Cartesian product of smaller cones is said to be primitive. That is, $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$ is not primitive. The Euclidean distance from a point $\tilde{x}$ to its projection y in $\mathcal{K}$, is given by $\text{dist}(\tilde{x}, \mathcal{K}) = \min_{y\in\mathcal{K}} \|\tilde{x} - y\|_2$. These distances are listed in the paragraphs below for projections y as shown in [6]. The fractionality of a scalar $\tilde{x}$, is the minimum distance to a point in $\mathbb{Z}$ given by $\text{dist}(\tilde{x}, \mathbb{Z}) = |\tilde{x} - \text{round}(\tilde{x})|$.

Linear cones This family covers the set of reals $\mathbb{R}^n$, the set of zeros $\{0\}^n$, the nonnegative orthant $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n \mid x_j \geq 0 \text{ for } j = 1, \ldots, n\}$, and the

nonpositive orthant $\mathbb{R}^n_- = \{x \in \mathbb{R}^n \mid x_j \leq 0 \text{ for } j = 1, \ldots, n\}$. Infeasible points $\tilde{x}$, have a strictly positive Euclidean distance given elementwise over the primitive cones by $|\tilde{x}_j|$ for $\{0\}$, $[\tilde{x}_j]^-$ for $\mathbb{R}_+$, and $[\tilde{x}_j]^+$ for $\mathbb{R}_-$. Points in $\mathbb{R}^n$ are always feasible.

Second-order cones This family, also known as ice cream cones, covers the quadratic cone $\mathcal{Q}^{1+n} = \{(r, x) \in \mathbb{R}^1_+ \times \mathbb{R}^n \mid r^2 \geq x^T x\}$, and the rotated quadratic cone $\mathcal{Q}_r^{2+n} = \{(r, x) \in \mathbb{R}^2_+ \times \mathbb{R}^n \mid 2r_1 r_2 \geq x^T x\}$. Infeasible points $\tilde{x}$, have a strictly positive Euclidean distance given by

$$\operatorname{dist}(\tilde{x},\ \mathcal{Q}^n) = \begin{cases} \left[\frac{\tilde{x}_1 - \|\tilde{x}_{2:n}\|_2}{\sqrt{2}}\right]^- & \text{if } \tilde{x}_1 \geq -\|\tilde{x}_{2:n}\|_2, \\ \|\tilde{x}\|_2 & \text{otherwise,} \end{cases}$$

and

$$\operatorname{dist}(\tilde{x},\ \mathcal{Q}_r^n) = \operatorname{dist}\left(T\tilde{x},\ \mathcal{Q}^n\right), \text{ where } T = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & 0 & \mathrm{I} \end{bmatrix}.$$

Semidefinite cones Refers to the real-valued symmetric positive semidefinite cone $\mathcal{S}^n_+ = \{X \in \mathcal{S}^n \mid \lambda(X) \in \mathbb{R}^n_+\}$, where λ is the eigenvalue function. Infeasible matrix-points $\tilde{X}$, have a strictly positive Euclidean distance defined here by $\| [\lambda(\tilde{X})]^- \|_2$ (derived from the Schatten 2-norm). We point out an often encountered alternative, $\| [\lambda(\tilde{X})]^- \|_\infty$ (derived from the induced 2-norm), but leave the discussion on the best choice open.

2 Problem formulation

The optimization problems of CBLIB 2014 are all written in the conic form shown in (P). The coefficients of the linear objective function are given by $c \in \mathbb{R}^{n_x}$ and $C \in \mathcal{S}^{n_X}$, and the constraints are similarly described by $A \in \mathbb{R}^{n_g \times n_x}$, $\mathcal{F} : \mathcal{S}^{n_X} \to \mathbb{R}^{n_g}$ and $b \in \mathbb{R}^{n_g}$, as well as $\mathcal{H}^* : \mathbb{R}^{n_x} \to \mathcal{S}^{n_G}$ and $B \in \mathcal{S}^{n_G}$. This problem is known as the primal problem,

$$\begin{array}{lll} \underset{x,X}{\text{minimize}} & c^T x + \langle C, X \rangle & \\ \text{subject to} & Ax + \mathcal{F}(X) - b \in \mathcal{K}_g^{n_g}, & \\ & \mathcal{H}^*(x) - B \in \mathcal{S}_+^{n_G}, & \\ & x \in \mathcal{K}_x^{n_x},\ X \in \mathcal{S}_+^{n_X}, \text{ and } x_j \in \mathbb{Z} \text{ for } j \in \mathcal{J}. & \end{array} \tag{P}$$

Excluding the integer requirements of (P), it has a Lagrange dual problem (D) which, as we shall see in a moment, can be used to certify primal infeasibility or certify that a claimed solution is optimal. It can also be used in sensitivity analysis [16], or in variable generation [36] similar to column-generation in linear optimization. While the exact representation of the Lagrange dual problem is

not unique, the following formulation may be used as standard reference,

$$\begin{array}{lllllllll} \underset{y,Y}{\text{maximize}} & b^T y + \langle B, Y \rangle & & & & & & & \\ \text{subject to} & A^T y & + & \mathcal{H}(Y) & - & c & \in & -(\mathcal{K}_x^{n_x})^*, \\ & \mathcal{F}^*(y) & & & - & C & \in & -(\mathcal{S}_+^{n_X})^*, \\ & y \in (\mathcal{K}_g^{n_g})^*, \; Y \in (\mathcal{S}_+^{n_G})^*. & & & & & & & \end{array} \tag{D}$$

The linear operators from matrices to vectors, $\mathcal{F}(X)$ and $\mathcal{H}(Y)$, and their adjoint operators from vectors to matrices, $\mathcal{F}^*(x)$ and $\mathcal{H}^*(y)$, are defined according to

$$\mathcal{F}(X) = \begin{bmatrix} \langle F_1, X \rangle \\ \vdots \\ \langle F_{n_g}, X \rangle \end{bmatrix}, \qquad \mathcal{F}^*(y) = \sum_{i=1}^{n_g} y_i F_i,$$

where $F_i \in \mathcal{S}^{n_X}$ for $i = 1, \ldots, n_g$, and $H_j \in \mathcal{S}^{n_G}$ for $j = 1, \ldots, n_x$. The Lagrange dual problem (D) is further specified in terms of dual cones indicated by a superscripted star. The semidefinite cone is self-dual, e.g., $(\mathcal{S}_+^{n_X})^* = (\mathcal{S}_+^{n_X})$, and the same holds for most other cones mentioned here. The exception is the set of reals, $\mathbb{R}^n$, and the set of zeros, $\{0\}^n$, which are each others dual cone.

Solvers for (P) make claims about the status of its result, e.g., feasible or optimal, or about the status of the problem, e.g., infeasible. This status should be verified in benchmarks whenever possible, to catch the cases of failure. In this regard, anything more than verifying the feasibility of a point requires dual values from the solver.

> A point is said to be *feasible* with respect to (P) or (D), when the Euclidean distance to all cones and the fractionality of integer variables are zero. In practice, a certain tolerance has to be accepted due to the approximating nature of floating-point arithmetic.
>
> A feasible point to (P) is *certified optimal* when we are given a feasible point to (D) with the same objective value (within a tolerance). This is a direct consequence of weak duality.
>
> The problem (P) is *certified infeasible* when we are given a feasible point to the problem (D), modified such that c and C are fixed to zero, with a strictly positive objective value (above a tolerance). This is the conic generalization of the Farkas' lemma from linear optimization.
>
> The problem (P) has a *certified unbounded direction* (which may improve the objective value of any feasible point indefinitely) when we are given a feasible point to the problem (P), modified such that b and B are fixed to zero, with a strictly negative objective value (below a tolerance). This also implies that the problem (D) is certified infeasible.

The second and third certificate, depending the Lagrange dual problem (D), is only guaranteed to exist when (P) has no integer requirements, i.e., $\mathcal{J} = \emptyset$, and

a constraint qualification is satisfied. Instances without constraint qualification can be regarded as ill-posed, however, as it implies the existence of a less complex description of the same feasible region [41]. Basic conic duality theory and some extensions are found in [6, 33, 30].

In conclusion of this section, we show how to formulate an instance of the Markowitz portfolio optimization problem (2) in the CBF file format. We let the number of investments in consideration be $n = 2$, and write out the primal problem (P) with an objective sense changed from minimize to maximize,

$$\mathcal{K}_x^{n_x} = \mathbb{R}_+^2, \;\; \mathcal{K}_g^{n_g} = \mathcal{Q}^3 \times \{0\}, \;\; c = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \;\; A = \begin{bmatrix} 0 & 0 \\ G_{11} & G_{12} \\ 0 & G_{22} \\ 1 & 1 \end{bmatrix}, \;\; b = \begin{bmatrix} -\gamma^{1/2} \\ 0 \\ 0 \\ \omega \end{bmatrix}.$$

This translates directly into the CBF file shown in Table 1, bearing in mind that specified constants are always added to, never subtracted from, the affine expressions and the objective function. More details of the CBF file format can be found in [17].

```
VER
1

OBJSENSE
MAX

VAR
2 1
L+ 2

CON
4 2
Q 3
L= 1
```

```
OBJACOORD
2
0 μ1
1 μ2

ACOORD
5
1 0 G11
1 1 G12
2 1 G22
3 0 1
3 1 1

BCOORD
2
0 γ^1/2
3 −ω
```

Table 1: A portfolio optimization problem in the CBF file format.

3 The instance catalog

This section brings an overview of the problem instances in CBLIB 2014. A brief description of each instance is found in Table 2, along with references to the researchers who worked with and described the instances. Most instances have been found by data mining in the public domain, and the contributors of these instances to the CBLIB project have been recognized in the distributed benchmark library. Note that semidefinite cones are absent from this initial release due to our focus on second-order cones.

Instances	Origin and description
estein4_A, estein4_B, estein4_C, estein4_nr22, estein5_A, estein5_B, estein5_C, estein5_nr1, estein5_nr21	Drewes [12]. Minimum Steiner tree problem.
turbine07, turbine07_aniso, turbine54, turbine07GF, turbine54GF, turbine07_lowb, turbine07_lowb_aniso	Drewes [12]. Balancing high-speed rotating machinery with either the least axial weight locations, the least distinct weight sets ('GF' in name), or minimum imbalance ('lowb' in name).
nb, nb_L1, nb_L2, nb_L2_bessel	Coleman and Vanderbei [8]. Calibration of antenna arrays, suppressing signals that do not come from a chosen direction.
nql30, nql60, nql90, nql180, qssp30, qssp60, qssp90, qssp180	Andersen et al. [1], Christiansen and Andersen [7]. Collapse states for loaded plastic plates using the plain strain model ('nql' in name), or the supported plate model ('qssp' in name).
sched_50_50_orig, sched_50_50_scaled, sched_100_50_orig, sched_100_50_scaled, sched_100_100_orig, sched_100_100_scaled, sched_200_100_orig, sched_200_100_scaled	Skutella [38]. Job scheduling on parallel unrelated machines.
sssd-strong-15-4, sssd-strong-15-8, sssd-strong-20-4, sssd-strong-20-8, sssd-strong-25-4, sssd-strong-25-8, sssd-strong-30-4, sssd-strong-30-8, sssd-weak-15-4, sssd-weak-15-8, sssd-weak-20-4, sssd-weak-20-8, sssd-weak-25-4, sssd-weak-25-8, sssd-weak-30-4, sssd-weak-30-8	Bonami et al. [4], Elhedhli [13]. Stochastic service system design with M/M/1 queues using Strong formulation ('strong' in name), or weak formulation ('weak' in name).
uflquad-nopsc-10-100, uflquad-nopsc-10-150, uflquad-nopsc-20-100, uflquad-nopsc-20-150, uflquad-nopsc-30-100, uflquad-nopsc-30-150, uflquad-nopsc-30-200, uflquad-nopsc-30-300, uflquad-psc-10-100, uflquad-psc-10-150, uflquad-psc-20-100, uflquad-psc-20-150, uflquad-psc-30-100, uflquad-psc-30-150, uflquad-psc-30-200, uflquad-psc-30-300	Bonami et al. [4], Günlük et al. [20]. Separable quadratic uncapacitated facility location. Relaxation with cuts ('psc' in name) or without cuts ('nopsc' in name).
pp-n10-d10, pp-n10-d10000, pp-n100-d10, pp-n100-d10000, pp-n1000-d10, pp-n1000-d10000, pp-n100000-d10, pp-n100000-d10000	Ziegler [45]. Production planning.
chainsing-1000-1, chainsing-1000-2, chainsing-1000-3, chainsing-10000-1, chainsing-10000-2, chainsing-10000-3, chainsing-50000-1, chainsing-50000-2, chainsing-50000-3	Conn et al. [10], Kobayashi et al. [26]. The chained singular function (academic).
2013_dsNRL, 2013_wbNRL, 2013i_wbNRL, 2013_firL1, 2013_firL1Linfalph, 2013_firL1Linfeps, 2013_firL2a, 2013_firL2L1alph, 2013_firL2L1eps, 2013_firL2Linfalph, 2013_firL2Linfeps, 2013_firLinf	Coleman et al. [9]. Optimal design of a delta-sigma ('ds' in name), a wideband ('wb' in name) or a nonlinear-phase FIR ('fir' in name) filter.
classical_50_1, classical_50_2, classical_50_3, classical_200_1, classical_200_2, classical_200_3, robust_50_1, robust_50_2, robust_50_3, robust_100_1, robust_100_2, robust_100_3, robust_200_1, robust_200_2, robust_200_3, shortfall_50_1, shortfall_50_2, shortfall_50_3, shortfall_100_1, shortfall_100_2, shortfall_100_3, shortfall_200_1, shortfall_200_2, shortfall_200_3	Vielma et al. [43]. Portfolio optimization with cardinality constraints.

Table 2: Descriptions and references to the researchers who worked with and described the instances.

	Size			Conic domain entries: Linear			Conic domain entries: Second-order			Conic domains		Objective
Instances	var	map	nnz	bin	int	cont	bin	int	cont	lin	so	
2013_dsNRL	61822	1616	66668564			1616			61822	1616	20504	-9.6379E−06
2013_firL1	59706	20902	39787428			20902			59706	20902	19902	-3.6669E+00
2013_firL1Linfalph	119412	20903	79574856			20903			119412	20903	39804	-3.3116E+00
2013_firL1Linfeps	59173	30085	9873426			30086			59172	30086	19724	-1.5255E−02
2013_firL2L1alph	49612	30268	9985771			30269			49611	30269	9923	-2.4441E−01
2013_firL2L1eps	60708	20903	40288929			20903			60708	20903	19903	-3.0683E+00
2013_firL2Linfalph	91783	2002	121660011			2002			91783	2002	29928	-7.7910E−02
2013_firL2Linfeps	59636	24655	19108570			24655			59636	24655	11928	-1.0141E−02
2013_firL2a	10002	10001	50015001			10001			10002	10001	1	-1.4368E−01
2013_firLinf	59856	2001	79771354			2001			59856	2001	19952	-1.0022E−02
2013_wbNRL	40450	1042	39138234			38123			3369	38123	9	-3.8759E−05
2013i_wbNRL	63312	1710	101934231			59827			5195	59827	9	Dual infeasible
chainsing-1000-1	12976	9982	17966			13976			8982	13976	2994	3.0180E+01
chainsing-1000-2	9985	7988	14975			10985			6988	10985	1997	3.0180E+01
chainsing-1000-3	6991	5992	11981			7991			4992	7991	999	3.0180E+01
chainsing-10000-1	129976	99982	179966			139976			89982	139976	29994	3.0261E+02
chainsing-10000-2	99985	79988	149975			109985			69988	109985	19997	3.0261E+02
chainsing-10000-3	69991	59992	119981			79991			49992	79991	9999	3.0261E+02
chainsing-50000-1	649976	499982	899966			699976			449982	699976	149994	1.5134E+03
chainsing-50000-2	499985	399988	749975			549985			349988	549985	99997	1.5134E+03
chainsing-50000-3	349991	299992	599981			399991			249992	399991	49999	1.5134E+03
classical_50_1	152	255	2902	50		306			51	356	1	-9.4760E−02
classical_50_2	152	255	2902	50		306			51	356	1	-9.0528E−02
classical_50_3	152	255	2902	50		306			51	356	1	-8.8041E−02
classical_200_1	602	1005	41602	200		1206			201	1406	1	-1.1668E−01[v]
classical_200_2	602	1005	41602	200		1206			201	1406	1	-1.1009E−01[v]
classical_200_3	602	1005	41602	200		1206			201	1406	1	-1.0571E−01[v]
estein4_A	67	108	128	9		139			27	148	9	8.0137E−01
estein4_B	67	108	128	9		139			27	148	9	1.1881E+00
estein4_C	67	108	128	9		139			27	148	9	1.0727E+00
estein4_nr22	67	108	128	9		139			27	148	9	5.0329E−01
estein5_A	132	211	258	18		271			54	289	18	1.0454E+00
estein5_B	132	211	258	18		271			54	289	18	1.1932E+00
estein5_C	132	211	258	18		271			54	289	18	1.4991E+00
estein5_nr1	132	211	258	18		271			54	289	18	1.6644E+00
estein5_nr21	132	211	258	18		271			54	289	18	1.8182E+00
nb	2383	123	191519			127			2379	127	793	-5.0703E−02
nb_L1	3176	915	192312			1712			2379	1712	793	-1.3012E+01
nb_L2	4195	123	402285			127			4191	127	839	-1.6290E+00
nb_L2_bessel	2641	123	208817			127			2637	127	839	-1.0257E−01
nql30	4501	6380	20569			8181			2700	8181	900	-9.4602E−01
nql60	18001	25360	82539			32561			10800	32561	3600	-9.3504E−01
nql90	40501	56940	185909			73141			24300	73141	8100	-9.3136E−01
nql180	162001	227280	744419			292081			97200	292081	32400	-9.2764E−01
pp-n10-d10	50	31	59		10	41			30	51	10	7.2481E+01
pp-n10-d10000	50	31	59		10	41			30	51	10	1.4815E+03
pp-n100-d10	500	301	599		100	401			300	501	100	7.7729E+02[v]
pp-n100-d10000	500	301	597		100	401			300	501	100	1.9856E+04
pp-n1000-d10	5000	3001	5969		1000	4001			3000	5001	1000	7.3436E+03[v]
pp-n1000-d10000	5000	3001	5968		1000	4001			3000	5001	1000	2.1611E+05
pp-n100000-d10	500000	300001	597382		100000	400001			300000	500001	100000	0.0000E+00[a]
pp-n100000-d10000	500000	300001	597463		100000	400001			300000	500001	100000	0.0000E+00[a]
qssp30	7565	11255	44414			11256			7564	11256	1891	-6.4967E+00
qssp60	29525	44105	178814			44106			29524	44106	7381	-6.5627E+00
qssp90	65885	98555	403214			98556			65884	98556	16471	-6.5942E+00
qssp180	261365	391505	1616414			391506			261364	391506	65341	-6.6391E+00
robust_50_1	207	365	5564	51		417			104	468	2	-8.5695E−02
robust_50_2	207	365	5564	51		417			104	468	2	-1.4365E−01
robust_50_3	207	365	5564	51		417			104	468	2	-8.9803E−02
robust_100_1	407	715	21114	101		817			204	918	2	-7.2090E−02
robust_100_2	407	715	21114	101		817			204	918	2	-9.1574E−02
robust_100_3	407	715	21114	101		817			204	918	2	-1.1682E−01
robust_200_1	807	1415	82214	201		1617			404	1818	2	-1.4275E−01

robust_200_2	807	1415	82214	201	1617		404	1818	2	-1.2167E−01
robust_200_3	807	1415	82214	201	1617		404	1818	2	-1.2911E−01[v]
sched_50_50_orig	4979	2527	25488		5029		2477	5029	2	2.6673E+04[a]
sched_50_50_scaled	4977	2526	27985		5028		2475	5028	1	7.8520E+00
sched_100_50_orig	9746	4844	55291		9846		4744	9846	2	Primal infeasible[a]
sched_100_50_scaled	9744	4843	60288		9845		4742	9845	1	6.7165E+01
sched_100_100_orig	18240	8338	104902		18340		8238	18340	2	Primal infeasible[a]
sched_100_100_scaled	18238	8337	114899		18339		8236	18339	1	2.7331E+01
sched_200_100_orig	37889	18087	260503		38089		17887	38089	2	Primal infeasible[a]
sched_200_100_scaled	37887	18086	280500		38088		17885	38088	1	5.1812E+01
shortfall_50_1	205	361	5612	51	413		102	464	2	-1.1018E+00
shortfall_50_2	205	361	5612	51	413		102	464	2	-1.0952E+00
shortfall_50_3	205	361	5612	51	413		102	464	2	-1.0923E+00
shortfall_100_1	405	711	21212	101	813		202	914	2	-1.1063E+00
shortfall_100_2	405	711	21212	101	813		202	914	2	-1.1007E+00[v]
shortfall_100_3	405	711	21212	101	813		202	914	2	-1.1031E+00
shortfall_200_1	805	1411	82412	201	1613		402	1814	2	-1.1354E+00[v]
shortfall_200_2	805	1411	82412	201	1613		402	1814	2	-1.1253E+00[v]
shortfall_200_3	805	1411	82412	201	1613		402	1814	2	-1.1163E+00[v]
sssd-strong-15-4	125	180	372	72	197		36	269	12	3.2800E+05
sssd-strong-15-8	249	344	744	144	377		72	521	24	6.2251E+05
sssd-strong-20-4	145	205	432	92	222		36	314	12	2.8781E+05
sssd-strong-20-8	289	389	864	184	422		72	606	24	6.0035E+05
sssd-strong-25-4	165	230	492	112	247		36	359	12	3.1172E+05
sssd-strong-25-8	329	434	984	224	467		72	691	24	5.0114E+05[v]
sssd-strong-30-4	185	255	552	132	272		36	404	12	2.6413E+05
sssd-strong-30-8	369	479	1104	264	512		72	776	24	5.2877E+05[v]
sssd-weak-15-4	125	180	360	72	197		36	269	12	3.2800E+05
sssd-weak-15-8	249	344	720	144	377		72	521	24	6.2251E+05
sssd-weak-20-4	145	205	420	92	222		36	314	12	2.8781E+05
sssd-weak-20-8	289	389	840	184	422		72	606	24	6.0035E+05[v]
sssd-weak-25-4	165	230	480	112	247		36	359	12	3.1172E+05
sssd-weak-25-8	329	434	960	224	467		72	691	24	5.0075E+05[v]
sssd-weak-30-4	185	255	540	132	272		36	404	12	2.6413E+05
sssd-weak-30-8	369	479	1080	264	512		72	776	24	5.2877E+05[v]
turbine07	84	101	313		101	11	73	101	26	2.0000E+00
turbine07GF	87	124	444	12	124		75	136	25	3.0000E+00
turbine07_aniso	83	108	313		116	11	64	116	25	3.0000E+00
turbine07_lowb	212	354	621	56	424		86	480	27	8.9928E−01
turbine07_lowb_aniso	210	361	621	56	440		75	496	25	1.3945E+00
turbine54	366	477	2099		477	11	355	477	120	3.0000E+00
turbine54GF	369	500	2982	12	500		357	512	119	4.0000E+00
uflquad-nopsc-10-100	3011	5111	7010	10	5112		3000	5122	1000	5.4029E+02
uflquad-nopsc-10-150	4511	7661	10510	10	7662		4500	7672	1500	7.0965E+02
uflquad-nopsc-20-100	6021	10121	14020	20	10122		6000	10142	2000	3.9954E+02
uflquad-nopsc-20-150	9021	15171	21020	20	15172		9000	15192	3000	5.6872E+02[v]
uflquad-nopsc-30-100	9031	15131	21030	30	15132		9000	15162	3000	3.5524E+02[v]
uflquad-nopsc-30-150	13531	22681	31530	30	22682		13500	22712	4500	4.6816E+02[v]
uflquad-nopsc-30-200	18031	30231	42030	30	30232		18000	30262	6000	5.8356E+02[v]
uflquad-nopsc-30-300	27031	45331	63030	30	45332		27000	45362	9000	9.4070E+02[v]
uflquad-psc-10-100	3011	5111	8010	10	5112		3000	5122	1000	5.4029E+02
uflquad-psc-10-150	4511	7661	12010	10	7662		4500	7672	1500	7.0965E+02
uflquad-psc-20-100	6021	10121	16020	20	10122		6000	10142	2000	3.9954E+02
uflquad-psc-20-150	9021	15171	24020	20	15172		9000	15192	3000	5.6872E+02
uflquad-psc-30-100	9031	15131	24030	30	15132		9000	15162	3000	3.5524E+02
uflquad-psc-30-150	13531	22681	36030	30	22682		13500	22712	4500	4.6816E+02
uflquad-psc-30-200	18031	30231	48030	30	30232		18000	30262	6000	5.5492E+02
uflquad-psc-30-300	27031	45331	72030	30	45332		27000	45362	9000	7.6266E+02

[a(currancy)] Infeasibility meassures exceed 10^{-4} on some primitive cones or integer requirements (points not normalized).

[v(alue)] Objective neither claimed by a solver (mixed-integer case) nor certified by a feasible dual solution (continuous case), to be within an absolute gap of 10^{-4} or relative gap of 10^{-7} from optimality.

Table 3: CBLIB 2014 instance statistics.

The instance statistics are found in Table 3. For each instance the table shows the number of variables, affine expressions, and nonzero constraint coefficients (not counting constants and objective coefficients). It then shows the number of binary, general integer, and continuous entries of cones in each domain family, wherein affine expression entries (that is, entries of $\mathcal{K}_g^{n_g}$ from (P) as opposed to entries of $\mathcal{K}_x^{n_x}$) are always counted as continuous. Next follows a total of the linear and second-order cones. The last column indicates the instance status by reporting the primal objective value, whenever possible, of a point found using the default solver and parameter settings of MOSEK version 7 [31]. The reported objective is optimal unless otherwise indicated. Please see the footnotes of the table for the tolerances used on feasibility and optimality.

One may ask whether the last column of Table 3 can be replaced by exact results freed from tolerances? This question is addressed in [25] and [22] using interval arithmetic, and for the general case their conclusion is negative. Points that lie exactly on the boundary of a semidefinite cone are nontrivial to verify in practice, and for the interval of the optimal value to be finite, all variables are required to be bounded. Another approach is through symbolic-numeric quantifier elimination [24], generalizing the concept of Fourier-Motzkin elimination from linear optimization. However, this algorithm features doubly exponential complexity and is not practical for the instances of this benchmark library. This is in sharp contrast to continuous linear optimization in which exact solutions in rational arithmetic can be obtained fairly efficiently [27].

The library contains 121 instances out of which 80 are mixed-integer. Only eight of the 80 mixed-integer instances contain general integer variables, showing binary variables to be the most common as expected. This is in line with the mixed-integer linear instances of the *MIPLIB* library [28]. All instances contain second-order cones, but only three of the 80 mixed-integer instances require the entry of a second-order cone to be integer. Beware, that this latter observation is based solely on the domain of integer variables, and does not consider affine expression entries even though they may preserve the integrality of its variables.

The average number of entries per second-order cone is close to three in many of the instances. Elaborating on this, 86 of the 121 instances contain at least one 3-dimensional second-order cone out of which 20 have exactly one other and 66 have no other second-order cones. In the other end of the scale we find nine of the 121 instances with more than a thousand entries per second-order cone on average. The total of second-order cones range as low as one (eleven instances) to more than 100 000 (three instances).

Instance `2013i_wbNRL` is an example of the fact that it is quite normal to make mistakes or forget something in the first attempt to formulate a problem. In this particular case, the problem features a direction which may improve the objective value of any feasible point indefinitely. In other words, it can be certified to have an unbounded direction, and this direction is the best result a solver can produce to help the user locate such formulation errors.

Instances `sched_*_*_orig` are all indicated by Table 3 to be primal infeasible, albeit known from the scaled versions to be feasible. It so happens because these indicators were generated from the points returned by MOSEK, as the best guess possible when numerical difficulties hindered convergence. In this case, the returned points were simply closer to primal infeasibility certificates, than to satisfy the criteria for primal feasibility. These instances constitute a numerical challenge.

Instances `pp-n100000-d10` and `pp-n100000-d10000` are large mixed-integer second-order cone instances known to have feasible points. None of these points, however, were identified by MOSEK 7 within one hour. The default solver and parameter settings of CPLEX version 12 [23] were also attempted on these two instances, but did not yield a feasible point either. These instances constitute a combinatorial challenge.

3.1 Filtering out instances of interest

Cone support is not uniform across all solvers, and it is often the case that benchmarks focus on a subset of instances with certain characteristics. For this reason the Python script, `filter.py`, has been developed to filter out instances of interest. It takes a string as input, substitute `||*|...||` with the value of the filter `*` given arguments `...`, and evaluate it as a boolean expression. Instances evaluating to true are listed.

```
python filter.py "||int|| and ||cones|so|| and not ||psdcones||"
```
Mixed-integer second-order cone instances.

```
python filter.py "||entries|so|| / ||cones|so|| <= 4"
```
Instances with no more than four entries per second-order cone on average.

```
python filter.py "||cones|so|| == ||cones|so|==3||"
```
Instances where all second-order cones have exactly three entries.

The filtering mechanism can be extended to more advanced selection criteria by adding Python functions to the directory of filters. In this release, the filters extend to those listed in Table 4. Given a second string as argument, the script will evaluate and output the result of it. With an empty filter (always true), this can be used to generate tables of instance statistics.

```
python filter.py "" "[||name||, ||var|F||, ||int|so||, ||map|lin||]"
```
The name followed by the number of free variables, of integer second-order cone entries and of linear constraints (listed for all instances).

This second argument is also useful for becoming familiar with an instance, or with how a particular filter works.

name	cones	psdcones
minimize	– varcones	– psdvarcones
change	– mapcones	– psdmapcones
nnz	entries	psdentries
	– var	– psdvar
	– map	– psdmap
	int	
	– binary	

Table 4: General, scalar-variable and matrix-variable filters aligned in columns. The symbol '–' indicates that a filter is a restricted version of another filter.

3.2 Feeding instances into optimization software

A disadvantage of the CBF format is the lack of support in mainstream software. This concern has led to the development of tools which can aid integration with, or transformation to the input format of, most software packages. More specifically, the library is distributed with CBF parsers in various programming languages and a file converter tool.

Parsers of the CBF format has been written in the MATLAB, Python, and C++ programming languages. These parsers may be used to feed instances into optimization software through programming interfaces. An example of this concept has been made with the Python script, `run.py`, which uses the CBF parser in Python to feed instances into MOSEK [31] through its Python API. This script was, for example, used to generate the last column of Table 3 in the instance catalog. By default, the script is configured to save the optimization result of each instance with the extension, `.sol`. Subsequent analysis with the Python script, `summary.py`, is thus possible.

```
python run.py runmosek -l [FILENAME]
```
Runs MOSEK on all instances listed in the newline-separated file.

```
python summary.py -l [FILENAME]
```
Summary of results for all instances listed in the newline-separated file.

The file converter tool, named `cbftool`, uses the CBF parser written in C++ to convert instances into another file format. It is capable of transforming conic constraints, $Ax - b \in \mathcal{K}$, into $Ax - b = s$ and $s \in \mathcal{K}$, but is otherwise incapable of modifying problem formulations to match the limitations of a particular file format. Thus, although the sparse SDPA format [44] is supported by `cbftool`, only pure semidefinite instances can be converted to this format. The tool supports two extensions of the MPS format to facilitate second-order cones. The explicit second-order cone extension from 1999 [31], and the implicit quadratic inequality extension introduced in 2003 [23]. Examples are given below.

```
cbftool -o mps-mosek [FILENAME] [FILENAME] ...
```
Convert given instances to the explicit second-order cone extension of the MPS format.

```
cbftool -o mps-cplex -opath [FOLDER] [FILENAME] [FILENAME] ...
```
Convert given instances to the implicit quadratic inequality extension of the MPS format, and store the results in the specified folder.

4 Notes on benchmarking

On a general note, a benchmark comparing two or more optimization methods must provide answers to the fundamental questions:

What are the performance measures?

What are the acceptance criteria for the output of the optimization methods?

Acceptance criteria usually boils down to the tolerances within which near-feasible solutions and certificates are accepted. In case the selected acceptance criteria approve all results, and there is only one performance measure, e.g., computation time, the improvement (or speed-up) factor [15] is a simple and widely used measure of comparison. The improvement factor does not indicate variance, however, and when time limits are reached, or results fail to satisfy the acceptance criteria, performance profiles [11] becomes a much better basis of comparison.

The degree of infeasibility can be measured by the Euclidean distances and the fractionality of integer variables, introduced at the end of Section 1. These measures are recommended when the individual instances are studied, e.g., when comparing different formulations. This is the typical situation of problem owners, and implies that when instances are badly formulated, solvers will struggle and yield large infeasibility measures. On the other hand, when the individual solvers are studied, it is unfair to blame these for the occurrences of high infeasibility caused by badly formulated instances. In this latter case, the following precautions are therefore recommended:

- Treat each *primitive* cone separately. A cone may be the Cartesian product of other cones, such as when a semidefinite matrix has a block-diagonal construction. In such cases, each factor of the Cartesian product, e.g., each block of the semidefinite matrix, should be treated separately.
- *Normalize* affine expressions by the infinity norm of coefficients. That is, given the affine expression $Ax + b$ constrained to the nonegative orthant $\mathbb{R}_+$, or any other cone, one should compute the infeasibility measure for the normed point $(Ax + b)/\max(1, \|A\|_\infty, \|b\|_\infty)$.

Finally, we consider a common performance measure, computation time, usually measured by the wall-clock time on the same, or equivalent, hardware and software environments. Slight performance variability between two equivalent executions are to be expected, even for deterministic optimization methods, and are subject to statistical analysis as covered in [28]. Given that performance is to be summarized in a single time measurement, it is recommended only to include those parts of the optimization process to which a genuine interest lies. The time used to input and receive output through a slow medium such as a hard drive, for instance, is rarely of interest. Including this overhead is unfair to academic work focusing efforts on optimization, and misleading to commercial work offering faster alternatives through low-level APIs.

5 Final remarks

Conic optimization has become mainstream during the past ten years. Excellent commercial and open-source solvers are available, frequent advancements are being made, and its potential usage stretch all the way to general convex optimization. The lack of a public benchmark library is apparent, however, and CBLIB is the first to follow mainstream support of conic optimization including mixed cone types and integer variables. For the upcoming releases, we aim to add semidefinite instances, more second-order cone instance, and challenging infeasible instances as requested in, e.g., [35]. We wish for a widespread adoption of the CBF format, and to make the integration with, or transformation to the input format of, any software package as easy as possible. We moreover intend to categorize instances in test sets such as open, challenging and easy.

The conic benchmark library is a community project and grow through external submissions. Please consider contributing at `http://cblib.zib.de`.

Acknowledgments The author wish to thank all contributors of instances to the benchmark library: Erling D. Andersen, Hans D. Mittelman, Joachim Dahl and Rune Sandvik. Thanks to Ambros Gleixner and Thorsten Koch for maintaining the benchmarking library at Zuse Institute Berlin, and to Mathias Stolpe for his guidance. The author, Henrik A. Friberg, was funded by MOSEK ApS and the Danish Ministry of Science, Innovation and Higher Education through the Industrial PhD project "Combinatorial Optimization over Second-Order Cones and Industrial Applications".

References

[1] K. D. Andersen, E. Christiansen, and M. L. Overton. Computing Limit Loads by Minimizing a Sum of Norms. *SIAM Journal on Scientific Computing*, 19(3):1046–1062, 1998.

[2] M. F. Anjos and S. Burer. On handling free variables in interior-point methods for conic linear optimization. *SIAM Journal on Optimization*, 18(4):1310–1325, 2007.

[3] A. Ben-tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. Society for Industrial and Applied Mathematics, 2001.

[4] P. Bonami, M. Kilinc, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. *Mixed Integer Nonlinear Programming*, 154:1–39, 2012.

[5] B. Borchers. SDPLIB 1.2, a library of semidefinite programming test problems. *Optimization Methods and Software*, 11(1):683–690, 1999.

[6] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. `https://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf`.

[7] E. Christiansen and K. D. Andersen. Computation of collapse states with von Mises type yield condition. *International Journal for Numerical Methods in Engineering*, 46(8):1185–1202, 1999.

[8] J. O. Coleman and R. J. Vanderbei. Random-process formulation of computationally efficient performance measures for wideband arrays in the far field. In *Proceedings of the 42nd Midwest Symposium on Circuits and Systems*, volume 2, pages 761–764, 1999.

[9] J. O. Coleman, D. P. Scholnik, and J. J. Brandriss. A specification language for the optimal design of exotic FIR filters with second-order cone programs. *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers*, 1:341–345, 2002.

[10] A. R. Conn, N. I. M. Gould, and P. L. Toint. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, 50(182):399–430, 1988.

[11] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[12] S. Drewes. *Mixed Integer Second Order Cone Programming*. PhD thesis, Department of Mathematics, Technical University Darmstadt, 2009. `http://www3.mathematik.tu-darmstadt.de/fileadmin/home/users/82/DissertationDrewes.pdf`.

[13] S. Elhedhli. Service System Design with Immobile Servers, Stochastic Demand, and Congestion. *Manufacturing & Service Operations Management*, 8(1):92–97, 2006.

[14] Fair Isaac Corporation. Xpress-Optimizer Reference Manual, Release 25.01. Technical report, 2013. To appear online.

[15] P. J. Fleming and J. J. Wallace. How not to lie with statistics: the correct way to summarize benchmark results. *Communications of the ACM*, 29(3):218–221, 1986.

[16] R. W. Freund and F. Jarre. A sensitivity result for semidefinite programs. *Operations Research Letters*, 32(2):126–132, Mar. 2004.

[17] H. A. Friberg. The Conic Benchmark Format: Version 1 - Technical Reference Manual. Technical Report E-0047, Department of Wind Energy, Technical University of Denmark, 2014. `http://orbit.dtu.dk/services/downloadRegister/88492586/Conic_Benchmark_Format.pdf`.

[18] D. M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, 13:10–12, 1985.

[19] F. Glineur. Conic optimization: an elegant framework for convex optimization. *Belgian Journal of Operations Research, Statistics and Computer Science*, 41:5–28, 2001.

[20] O. Günlük, J. Lee, and R. Weismantel. MINLP strengthening for separable convex quadratic transportation-cost UFL. Technical report, IBM Research Report RC24213, 2007. `http://domino.watson.ibm.com/library/cyberdig.nsf/papers/F93A77B97033B1878525729F0051C343/$File/rc24213.pdf`.

[21] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual - Version 5.6. Technical report, 2013. `http://www.gurobi.com/documentation/5.6/reference-manual/refman.pdf`.

[22] V. Härter, C. Jansson, and M. Lange. VSDP: A Matlab toolbox for verified semidefinite-quadratic-linear programming. Technical report, Institute for Reliable Computing, Hamburg University of Technology, 2012. `http://www.ti3.tu-harburg.de/jansson/vsdp/VSDP2012Guide.pdf`.

[23] IBM Corporation. IBM ILOG CPLEX V12.1 - User's Manual for CPLEX. Technical report, 2013. `ftp://ftp.software.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf`.

[24] H. Iwane, H. Yanami, H. Anai, and K. Yokoyama. An effective implementation of symbolic–numeric cylindrical algebraic decomposition for quantifier elimination. *Theoretical Computer Science*, 479:43–69, 2013.

[25] C. Jansson. Guaranteed Accuracy for Conic Programming Problems in Vector Lattices. Technical report, Institute for Reliable Computing, Technical University Hamburg, 2007. `http://arxiv.org/pdf/0707.4366v1`.

[26] K. Kobayashi, S. Kim, and M. Kojima. Sparse second order cone programming formulations for convex optimization problems. *Journal of the Operations Research Society of Japan*, 51 (3):241–264, 2008.

[27] T. Koch. The final NETLIB-LP results. *Operations Research Letters*, 32(2):138–142, 2004.

[28] T. Koch, T. Achterberg, E. D. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. D. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2): 103–163, 2011.

[29] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[30] D. A. Morán R., S. S. Dey, and J. P. Vielma. A Strong Dual for Conic Mixed-Integer Programs. *SIAM Journal on Optimization*, 22(3):1136–1150, 2012.

[31] MOSEK ApS. The MOSEK C Optimizer API manual, Version 7.0. Technical report, 2013. `http://docs.mosek.com/7.0/capi.pdf`.

[32] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[33] G. Pataki. Strong duality in conic linear programming: Facial reduction and extended duals. In D. H. Bailey, H. H. Bauschke, P. Borwein, F. Garvan, M. Théra, J. D. Vanderwerff, and H. Wolkowicz, editors, *Computational and Analytical Mathematics*, volume 50 of *Springer Proceedings in Mathematics & Statistics*, pages 613–634. Springer New York, 2013.

[34] G. Pataki and S. H. Schmieta. The DIMACS library of semidefinite-quadratic-linear programs. Technical report, Computational Optimization Research Center, Columbia University, 2002. `http://dimacs.rutgers.edu/Challenges/Seventh/Instances/lib.ps`.

[35] I. Pólik and T. Terlaky. New stopping criteria for detecting infeasibility in conic optimization. *Optimization Letters*, 3(2):187–198, 2009.

[36] K. Sivaramakrishnan, G. Plaza, and T. Terlaky. A conic interior point decomposition approach for large scale semidefinite programming. Technical report, Department of Mathematics, North Carolina State University, 2005. `http://www.optimization-online.org/DB_HTML/2005/11/1250.html`.

[37] A. Skajaa, E. D. Andersen, and Y. Ye. Warmstarting the homogeneous and self-dual interior point method for linear and conic quadratic problems. *Mathematical Programming Computation*, 5(1):1–25, 2012.

[38] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48(2):206–242, 2001.

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

[40] The MathWorks, Inc. MATLAB Primer, R2013b. Technical report, 2013. `http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf`.

[41] L. Tunçel and H. Wolkowicz. Strong duality and minimal representations for cone optimization. *Computational Optimization and Applications*, 53(2):619–648, 2012.

[42] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95(2):189–217, 2003.

[43] J. P. Vielma, S. Ahmed, and G. L. Nemhauser. A Lifted Linear Programming Branch-and-Bound Algorithm for Mixed Integer Conic Quadratic Programs. *INFORMS Journal on Computing*, 20:438–450, 2008.

[44] M. Yamashita, K. Fujisawa, K. Nakata, M. Nakata, M. Fukuda, K. Kobayashi, and K. Goto. A high-performance software package for semidefinite programs: SDPA 7. Technical report, 2010. `http://www.optimization-online.org/DB_HTML/2010/01/2531.html`.

[45] H. Ziegler. Solving certain singly constrained convex optimization problems in production planning. *Operations Research Letters*, 1(6):246–252, 1982.