Math. Prog. Comp. (2016) 8:191–214
DOI 10.1007/s12532-015-0092-4

FULL LENGTH PAPER

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

**Henrik A. Friberg**[1,2]

Received: 8 March 2014 / Accepted: 17 August 2015 / Published online: 16 October 2015


**Abstract** The Conic Benchmark Library is an ongoing community-driven project aiming to challenge commercial and open source solvers on mainstream cone support. In this paper, 121 mixed-integer and continuous second-order cone problem instances have been selected from 11 categories as representative for the instances available online. Since current file formats were found incapable, we embrace the new Conic Benchmark Format as standard for conic optimization. Tools are provided to aid integration of this format with other software packages.

**Keywords** Problem instances · Conic programming · Mixed integer programming

**Mathematics Subject Classification** 90C90 · 90C25 · 90C11

## 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 constraints. It may be formulated as

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

✉ Henrik A. Friberg
metronware@gmail.com

1 Department of Wind Energy, Technical University of Denmark, Roskilde, Denmark

2 MOSEK ApS, Copenhagen, Denmark

The conic form (1) allows us to express all convex mixed-integer and continuous optimization problems without loss of generality [21], but this generality offers no advantages from a computational point of view. Instead, only three types of cones (nonnegative orthant, quadratic cone and semidefinite cone) are typically used to solve a broad range of applications [4]. 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 compare the traditional and the conic form of the classical Markowitz portfolio optimization problem [34]. The portfolio problem (2), maximizes expected return subject to the accepted risk $\gamma$ and investable wealth $\omega$. The return vector $\mu$ and covariance matrix $\Sigma = U^T U$, characterize the investments in consideration.

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

The traditional form on the left uses a convex functional, $f(x) \leq 0$, to express the nonlinearity of the risk constraint. The conic form on the right achieves the same using the quadratic cone, $\mathcal{Q}^{1+n} = \{(r, x) \in \mathbb{R}^1_+ \times \mathbb{R}^n \mid r^2 \geq x^T x\}$. The equation and variable nonnegativity of the traditional form are formulated using two linear cones, the set of zero and the nonnegative orthant. More advanced examples of conic reformulations are found in [1] and [4].

One advantage of the conic form is that convexity does not have to be investigated, since it follows from convexity of the cones involved. In contrast, the convexity of a nonlinear problem in the traditional form cannot be established based on structural information, but has to be verified using the input data, such as $\Sigma$ 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 [38]. This advantage is reflected in the state-of-the-art optimization software, with high-performing implementations in all major commercial solvers; XPRESS [18], MOSEK [37], GUROBI [25] and CPLEX [27]. The open source projects listed in [44] are furthermore mostly based on variants of the method proposed in [38], including the significant contributions of SEDUMI [50] and SDPT3 [52]. SEDUMI, SDPT3 and MOSEK support all real-valued symmetric cones, while XPRESS, GUROBI and CPLEX omit support of the semidefinite cone. Integer constraints can be handled by all listed commercial solvers, but not by any of the open source projects. Open source support for conic mixed-integer optimization, however, is actively being added to the constraint integer programming framework SCIP through cone solver plugins [35] and outer approximations [6].

What is essentially missing from this development is a proper and publicly available benchmark library. Benchmark libraries are known to have a great effect on stimulating improvements in reliability and performance in optimization software. The *NETLIB LP* [20] library, for instance, was the first electronically distributed benchmark library for continuous linear optimization and often attributed for its major effect on the devel-

opment of LP solvers. Correspondingly, *MIPLIB* [32] has played a major role in the field of mixed-integer linear optimization. In review of benchmark libraries for conic optimization, *SDPLIB* [8] and the library of structured semidefinite programming instances [15] are worth noticing although their focus is limited to the semidefinite cone. A mixture of different cone types was considered in the *7th DIMACS Implementation Challenge* [40], but the benchmark library established for this challenge has been inactive for years. The DIMACS instances are furthermore difficult to use without MATLAB [51], and were reformulated at the time to eliminate free variables even though the best way to handle free variables is still an open research question [3]. No benchmark libraries were found for conic mixed-integer optimization, although supported by all major commercial optimization software available today. The closest match is probably the BIQMAC library [56], containing pure-binary quadratic optimization problems which are second-order cone representable.

The Conic Benchmark Library (CBLIB) is an ongoing community-driven project, hosted at http://cblib.zib.de, with aims to stay updated with the conic mixed-integer and continuous capabilities of mainstream solvers. First, however, there are concrete areas to be nursed. As seen, mixed cone types and integer variables represent cases where current benchmark libraries do not challenge state-of-the-art solvers. Even worse, the shortage of these instances alongside infeasible, dual infeasible and facially reducible problems prevent proper testing of theoretical ideas as concluded in [45] and [22]. More fundamentally, however, is the need of a file format for these conic problems that is supported across all major solvers. With CBLIB 2014, we have taken the initial steps toward addressing these issues.

First of all, a focused effort was made on gathering applications of second-order cones, as we found it to be poorly represented by current benchmark libraries. In this effort, instances formulated with convex quadratic constraints have been ignored, as there is usually a natural second-order cone representation that only the problem owner can retrieve. Portfolio optimization (2) is a good example of this, where a normalized, trimmed and often rank-reduced data matrix $U$ is the origin of the commonly used sample covariance matrix $\Sigma = U^T U$. Today, with the help of contributors from various fields, CBLIB has become the largest collection of mixed-integer and continuous second-order cone instances available online under a free and open license policy.

Second of all, a detailed analysis of existing file formats were carried out eventually leading to the Conic Benchmark Format (CBF). Looking at the old MPS format [27, 37], several extensions have been proposed over time, two of which enable second-order cone support. MOSEK [37] uses an explicit cone extension, while CPLEX [27] reuses a quadratic extension by reformulating the cone as the intersection of a half-space and a non-convex quadratic constraint. This lack of consensus is less of an issue, however, compared to the overwhelming task of augmenting the MPS format with the matrix notation for coefficients, variables and inequalities needed to realize a semidefinite cone extension. As consequence, many are currently using either the SDPA format [57], simply describing a matrix inequality, or the SEDUMI format [50], which is a MATLAB-based binary format. The CBF format can be seen as an attempt to unify the SDPA and SEDUMI format under a common conic model (presented in Sect. 3) and in portable clear text. The format is furthermore designed to allow maximum

performance reading into C, Python and MATLAB which makes a transition to the format less cumbersome.

The article is outlined as follows. Preliminaries are provided in Sect. 2. In Sect. 3, the CBLIB standard reference for a conic problem is formalized and related to the CBF file format. In Sect. 4, we discuss the notion of feasibility and exact results in conic optimization, as well as the five basic solution certificates for continuous problems. Section 5 describes the selection of problem instances for this paper, as well as the tools distributed with them. Final remarks are made in Sect. 6.

## 2 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

$$x \in \mathcal{K}_x,\ y \in \mathcal{K}_y \Longleftrightarrow \begin{bmatrix} x \\ y \end{bmatrix} \in \mathcal{K}_x \times \mathcal{K}_y, \tag{3}$$

for column vectors and

$$X \in \mathcal{S}^{n_1},\ Y \in \mathcal{S}^{n_2} \Longleftrightarrow \begin{bmatrix} X & 0 \\ 0 & Y \end{bmatrix} \in \mathcal{S}^{n_1} \times \mathcal{S}^{n_2}, \tag{4}$$

for matrices. 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 [9]. The minimum distance to a point in $\mathbb{Z}$, also known as the fractionality of a scalar $\tilde{x}$, is 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, nicknamed the 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

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

and

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

We point out that the rotated quadratic cone is often encountered without the factor 2 in front of $r_1 r_2$. This is also called a restricted hyperbolic constraint, originating with [33]. We did not consider the restricted hyperbolic constraint as a separate cone, however, as it is not symmetric, making duality more cumbersome, and because its transformation to a rotated quadratic cone has no computational disadvantage.

*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 $\lambda$ 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.

## 3 Problem formulation

The simplicity of the conic form (1) is also its weakness in practice. It implies a constraint-oriented (as opposed to a column-oriented) representation, hides a lot of information, and is bloated with identity matrices, $A_i = \mathrm{I}$, to define variable domains as used, e.g., by conic form problems in standard form [9]. To approach the first issues we stack all affine maps, $g(x) = Ax - b = (g^1(x)^T, \ldots, g^{k_g}(x)^T)^T$ where $g^i(x) = A_i x - b_i$ from (1), and constrain them to the affine map cone $\mathcal{K}_g^{n_g} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_{k_g}$, with $k_g$ being the number of cones and $n_g$ the total number of affine map entries. The latter issue is addressed by introducing a variable domain cone $\mathcal{K}_x^{n_x} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_{k_x}$, with $k_x$ being the number of cones and $n_x$ the total number of variables. These changes lead to the conic form,

$$\begin{aligned} \underset{x}{\text{minimize}}\ & c^T x \\ \text{subject to}\ & Ax - b \in \mathcal{K}_g^{n_g}, \\ & x \in \mathcal{K}_x^{n_x},\ \text{and } x_j \in \mathbb{Z} \text{ for } j \in \mathcal{I}, \end{aligned} \tag{5}$$

for which dimensions can be specified as $A \in \mathbb{R}^{n_g \times n_x}$, $b \in \mathbb{R}^{n_g}$, $c \in \mathbb{R}^{n_x}$ and $|\mathcal{I}| = n_i$. The conic form (5) is still cumbersome and ambiguous, however, when it comes to semidefinite cones, as it implies the use of linear indexes into symmetric matrices. This requires a consensus regarding whether matrices are seen as column-stacked or row-stacked and whether the symmetric upper or lower triangular elements are skipped or not. To address this issue, the conic form (5) has been augmented with an explicit matrix notation. The affected variables are combined in a matrix, $X$, and explicitly

constrained to the semidefinite variable domain, $\mathcal{S}_+^{n_X}$, which is the Cartesian product of smaller semidefinite cones. Similarly, the affected affine maps are combine in a matrix-valued affine map, $G(x)$, and constrained to the semidefinite affine map domain, $\mathcal{S}_+^{n_G}$, which is the Cartesian product of smaller semidefinite cones. With these changes we finally arrive at the standard reference for the primal problem used in CBLIB,

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

where the linear operators from matrices to vectors, $\mathcal{F}(X)$, and from vectors to matrices, $\mathcal{H}^*(x)$, are defined by

$$\mathcal{F}(X) = \begin{bmatrix} \langle F_1, X \rangle \\ \vdots \\ \langle F_{n_g}, X \rangle \end{bmatrix}, \quad \mathcal{H}^*(x) = \sum_{j=1}^{n_x} x_j H_j.$$

These definitions match the usual semidefinite program in standard and inequality form [9], and the dimensions are given by $C \in \mathcal{S}^{n_X}$, $B \in \mathcal{S}^{n_G}$, $F_i \in \mathcal{S}^{n_X}$ for $i = 1, \ldots, n_g$, and $H_j \in \mathcal{S}^{n_G}$ for $j = 1, \ldots, n_x$. For continuous problems, the standard reference for the dual problem used in CBLIB is given by the Lagrange-dual of (P) stated similarly as

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

where the adjoint linear operators from vectors to matrices, $\mathcal{F}^*(y)$, and from matrices to vectors, $\mathcal{H}(Y)$, are defined by

$$\mathcal{F}^*(y) = \sum_{i=1}^{n_g} y_i F_i, \quad \mathcal{H}(Y) = \begin{bmatrix} \langle H_1, Y \rangle \\ \vdots \\ \langle H_{n_x}, Y \rangle \end{bmatrix}.$$

Note that the domains of (D) are specified in terms of dual cones indicated by a superscripted star. Nevertheless, this is easily dealt with as all cones mentioned in this paper are self-dual, e.g., $(\mathcal{S}_+^{n_X})^* = (\mathcal{S}_+^{n_X})$, with exception of the set of reals, $\mathbb{R}^n$, and the set of zeros, $\{0\}^n$, which are each others dual cone. Now note the negation of affine map domains in the maximization problem (D). Had the objective sense of (P) been to

maximize, this would have been a negation of variable domains in the minimization problem (D). To memorize this relation, it is always the variable domains of the minimization problem that is subject to the sign change. On a pedagogical remark, this dualization procedure is just as applicable and produces the same result as the sensible-odd-bizarre rules [5] for linear optimization problems, but extends to support nonlinear cones.

### 3.1 The file format

The instances of CBLIB 2014 are stored in the Conic Benchmark Format which has a technical specification [19] matching the conic form (P). As a matter of fact, it only differs in its choice of objective sense which can be changed from minimize to maximize. In this section we revisit the example from the introduction and comment on its formulation in the CBF file format.

With two investments and an upper triangular covariance factor $U$, the Markowitz portfolio optimization problem (2) can be written in the conic form (P) as follows.

$$\begin{aligned} \underset{x_0,x_1}{\text{maximize}} \quad & \begin{bmatrix} \mu_0 \\ \mu_1 \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \\ \text{subject to} \quad & \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ U_{00} & U_{01} \\ 0 & U_{11} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} - \begin{bmatrix} -\gamma^{1/2} \\ 0 \\ 0 \\ \omega \end{bmatrix} \in \mathcal{Q}^3 \times \{0\}, \\ & \qquad\qquad\qquad\qquad\qquad\qquad \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \in \mathbb{R}^2_+. \end{aligned} \tag{6}$$

This problem formulation translates into the CBF file format, shown in Table 1, as follows. First we comply with the technical specification [19], by specifying that the file is written in version 1 of the CBF format (line 4–5). This has to be the first non-commentary line of the file, and is followed by a description of the problem (6) separated into model structure and problem data.

Stated as model structure, the objective sense is to maximize (line 9–10). The problem has two variables in one cone (line 11–12), namely $\mathbb{R}^2_+$ (line 13), and there are four affine maps in two cones (line 14–15), namely $\mathcal{Q}^3$ (line 16) and $\{0\}$ (line 17).

Stated as problem data, the objective function has two nonzero coefficients (line 21–22), namely $\mu_0$ for the first variable $x_0$ (line 23) and $\mu_1$ for the second variable $x_1$ (line 24). Note that all data is specified on a sparse coordinate form like this, with indexes counting from zero. The problem has five nonzero constraint coefficients (line 25–26), listed as $U_{00}$ for the first variable $x_0$ in the second affine map $g_1$ (line 27), and so on. Finally, there are two nonzero constraint constants (line 32–33), namely $\gamma^{1/2}$ in the first affine map $g_0$ (line 34) and $-\omega$ in the fourth affine map $g_3$ (line 35).

**Table 1** A portfolio optimization problem in the CBF file format

```
01 ##########################
02 ## FILE INFORMATION   ##
03 ##########################

04 VER
05 1

06 ##########################
07 ## MODEL STRUCTURE    ##
08 ##########################

09 OBJSENSE
10 MAX

11 VAR
12 2 1
13 L+ 2

14 CON
15 4 2
16 Q 3
17 L= 1

18 ##########################
19 ## PROBLEM DATA       ##
20 ##########################

21 OBJACOORD
22 2
23 0 μ0
24 1 μ1

25 ACOORD
26 5
27 1 0 U00
28 1 1 U01
29 2 1 U11
30 3 0 1
31 3 1 1

32 BCOORD
33 2
34 0 γ^1/2
35 3 −ω
```

Relevant to the benchmarking of warmstarting capability for continuous optimization problems [48], the CBF format also introduced the `CHANGE` keyword. At the end of a problem data specification, it can be used to start a new problem data specification appending to or modifying the previous. These relative changes allows the solver to reuse internal data structures in its reoptimization after every change. In the portfolio optimization problem (6), this could be used benchmark the solvers ability to generate (risk, return)-points on an investment curve for incrementing values of $\gamma$.

## 4 Solution validation

Since numerical computations are performed in finite precision, small errors may accumulate throughout the solution procedure. When a solver terminates with a claimed feasible solution, it may thus deviate from the mathematically exact feasible region by some tolerances defined in the solver. While a user may want tolerances to meet the needs of a specific application, knowing that lowering them can cause numerical issues rather than better solutions, a benchmarker may instead want to align solvers with each other. In any case, it is sensible to test the final result against vendor-independent error measures.

The best way to test the validity of a solution is to translate it to its natural application-specific representation, such as a schedule, and verify it there. More generally, and especially for comparative studies, a better basis of comparison may, however, be given by the fractionality of integer variables and Euclidean distances to each cone.

These measures can for instance be used when the individual formulations are studied, as it implies that bad formulations cause solvers to struggle and yield large infeasibility measures. In contrast, 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:

- *Normalize affine expressions by the infinity norm of coefficients*. By definition of a cone, the constraint $Ax + b \in \mathcal{K}$ is invariant to positive scaling. Invariance to scaling-based reformulations can also be achieved in the infeasibility measure, by computing the Euclidean distance for the normed point $(Ax + b)/\max(1, \|\mathrm{vec}(A)\|_\infty, \|b\|_\infty)$.
- *Treat each primitive cone separately*. By definition of the Cartesian product (3), two conic constraints can be merged into one. Invariance to such reformulations can also be achieved in the infeasibility measure, by computing the Euclidean distance separately for each factor of the Cartesian product. For a block-diagonal semidefinite matrix, this corresponds to computing it separately for each block.

Without going into details, a solution can be validated in terms of these measures by comparing the Euclidean distances to some chosen absolute error tolerance. In case of fractionality, it is also common to allow some relative error such that 1 000 000.1 is accepted as integer feasible while 1.1 is not. A brief survey of this and other solution validation criteria is found in [10, Chapter 1].

A relevant question at this point is whether we are able to obtain any kind of exact results freed from such tolerances. This question is addressed in [29] and [26] 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 to compute a finite interval, guaranteed to contain the optimal value, all primal variables have to be bounded. Another approach is through symbolic-numeric quantifier elimination [28], generalizing the concept of Fourier–Motzkin elimination from linear optimization. This algorithm has doubly exponential complexity, however, 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 [31].

### 4.1 Validating status claims

Most solvers return from a successful termination with a claim such as *the solution is optimal* or *the problem is infeasible*. In conic continuous optimization, there exists simple certificates to support such claims. In terms of problem (P), the solver has

- *certified optimality* of a feasible point, when we are given a feasible point to problem (D) with the same objective value, $c^T x + \langle C, X\rangle = b^T y + \langle B, Y\rangle$ (within a tolerance). This is a direct consequence of weak duality.
- *certified infeasibility* when we are given a feasible point to problem (D), modified such that $c$ and $C$ are fixed to zero, with a strictly positive objective value, $b^T y + \langle B, Y\rangle > 0$ (above a tolerance). This is the conic generalization of the Farkas' lemma from linear optimization.

- *certified dual infeasibility* when we are given a feasible point to problem (P), modified such that $b$ and $B$ are fixed to zero, with a strictly negative objective value, $c^T x + \langle C, X \rangle < 0$ (below a tolerance). This is also a direction in which the objective value of any primal feasible point can be improved indefinitely.
- *certified facial reducibility* when we are given a feasible point to problem (D), modified such that $c$ and $C$ are fixed to zero, with a zero-valued objective value, $b^T y + \langle B, Y \rangle = 0$ (within a tolerance), and non-zero entries of any self-dual cone. This is a facial reduction certificate for (P) showing it to be ill-posed in the sense of [46].

These certificates all follow from the basic theory of conic duality [9] and facial reduction [41], and the list is complete. That is, if problem (P) cannot be certified as facially reducible (not dual facially reducible), it either has a feasible point that can be certified as optimal, an infeasibility certificate, or a dual infeasibility certificate.

Fortunately, the primal-dual homogeneous interior-point method [39] is capable of finding such certificates in all cases. Indeed, the only ill-posed case of the algorithm [39, Last paragraph on page 223] actually converges to a certificate for facial reducibility, as recently shown by [42]. Hence, there might be a cure for the numerical issues faced by all current implementations of the algorithm when used on facially reducible problems [23,55].

## 5 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.

The instance statistics are found in Table 3. For each instance the table shows the total number of variables (*var*), affine expressions (*map*), and nonzero constraint coefficients (*nnz*) not counting constants and objective coefficients. It then shows the number of primitive linear (*lin*) and second-order (*so*) cones counted separately for each cone dimension. Primitive linear cones are always one-dimensional, and for second-order cones the dimension is followed by a colon and its count in a comma-separated list. Next follows the number of binary variables defined in a linear ($b_{lin}$) and second-order ($b_{so}$) variable domain cone. Similarly, the table shows the number of general integer variables defined in a linear ($I_{lin}$) and second-order ($I_{so}$) variable domain cone. The last columns indicate the instance status. The column (*obj*) reports the best primal objective value, whenever possible, among the primal feasible points found using MOSEK version 7.1.0.12 [37] and CPLEX version 12.6.0.0 [27] on a 64-bit Linux platform. Default parameters settings were used in these runs, except for forcing single-threaded behavior, a time limit of one hour, as well as an absolute and relative optimality gap of zero for integer problems. Superscripts are appended to this column, *obj*, when solutions could not be validated using the tolerances on feasibility and optimality stated in the footnotes of the table. The same tolerances are used to

**Table 2** Description of packages in the CBLIB 2014 selection with references to the researchers who worked with and described the 121 instances

| Packages | Origin and description | Instances |
|---|---|---|
| chainsing | Conn et al. [14], Kobayashi et al. [30] | 9 |
| | The chained singular function (academic) | |
| estein | Drewes [16] | 9 |
| | Minimum Steiner tree problem | |
| filterdesign | Coleman et al. [13] | 12 |
| | Optimal design of a delta-sigma ('ds' in name), a wideband ('wb' in name) or a nonlinear-phase FIR ('fir' in name) filter | |
| nb | Coleman and Vanderbei [12] | 4 |
| | Calibration of antenna arrays, suppressing signals that do not come from a chosen direction | |
| Portfoliocard | Vielma et al. [53] | 24 |
| | Portfolio optimization with cardinality constraints | |
| pp | Ziegler [58] | 8 |
| | Production planning | |
| sched | Skutella [49] | 8 |
| | Job scheduling on parallel unrelated machines | |
| sssd | Bonami et al. [7], Elhedhli [17] | 16 |
| | Stochastic service system design with M/M/1 queues using Strong formulation ('strong' in name), or weak formulation ('weak' in name) | |
| strain | Andersen et al. [2], Christiansen and Andersen [11] | 8 |
| | Collapse states for loaded plastic plates using the plain strain model ('nql' in name), or the supported plate model ('qssp' in name) | |
| turbine | Drewes [16] | 7 |
| | 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) | |
| uflquad | Bonami et al. [7], Günlük et al. [24] | 16 |
| | Separable quadratic uncapacitated facility location. With cuts ('psc' in name) or without cuts ('nopsc' in name) | |

label the output of MOSEK (column *M*) and CPLEX (column *C*). A dash, `-`, means that the output neither validated as a primal feasible point nor a certificate of any kind. Moreover, `P` means primal feasibility, `O` means optimality is claimed (mixed-integer case) or certified (continuous case), and `DI` means that a dual infeasibility certificate was recognized.

Overall, the CBLIB 2014 selection of instances can be described as follows. 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 [32]. All instances contain second-order cones, but only three of

**Table 3** CBLIB 2014 instance statistics

| Instances | Size | | | Conic domains | | Binary | | Integer | | Status | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | var | map | nnz | lin | so | lin | so | lin | so | obj | M | C |
| *chainsing* | | | | | | | | | | | | |
| chainsing-1000-1 | 12,976 | 9982 | 17,966 | 13,976 | [3:2994] | | | | | 3.0180E+01 | o | o |
| chainsing-1000-2 | 9985 | 7988 | 14,975 | 10,985 | [3:1996, 1000:1] | | | | | 3.0180E+01 | o | o |
| chainsing-1000-3 | 6991 | 5992 | 11,981 | 7991 | [3:998, 1998:1] | | | | | 3.0180E+01 | o | o |
| chainsing-10000-1 | 129,976 | 99,982 | 179,966 | 139,976 | [3:29,994] | | | | | 3.0261E+02 | o | o |
| chainsing-10000-2 | 99,985 | 79,988 | 149,975 | 109,985 | [3:19,996, 10,000:1] | | | | | 3.0261E+02 | o | o |
| chainsing-10000-3 | 69,991 | 59,992 | 119,981 | 79,991 | [3:9998, 19,998:1] | | | | | 3.0261E+02 | o | o |
| chainsing-50000-1 | 649,976 | 499,982 | 899,966 | 699,976 | [3:149,994] | | | | | 1.5134E+03 | o | o |
| chainsing-50000-2 | 499,985 | 399,988 | 749,975 | 549,985 | [3:99,996, 50,000:1] | | | | | 1.5134E+03 | o | o |
| chainsing-50000-3 | 349,991 | 299,992 | 599,981 | 399,991 | [3:49,998, 99,998:1] | | | | | 1.5134E+03 | o | o |
| *estein* | | | | | | | | | | | | |
| estein4_A | 67 | 108 | 128 | 148 | [3:9] | 9 | 0 | 0 | 0 | 8.0137E−01 | o | o |
| estein4_B | 67 | 108 | 128 | 148 | [3:9] | 9 | 0 | 0 | 0 | 1.1881E+00 | o | o |
| estein4_C | 67 | 108 | 128 | 148 | [3:9] | 9 | 0 | 0 | 0 | 1.0727E+00 | o | o |
| estein4_nr22 | 67 | 108 | 128 | 148 | [3:9] | 9 | 0 | 0 | 0 | 5.0329E−01 | o | o |
| estein5_A | 132 | 211 | 258 | 289 | [3:18] | 18 | 0 | 0 | 0 | 1.0454E+00 | o | o |
| estein5_B | 132 | 211 | 258 | 289 | [3:18] | 18 | 0 | 0 | 0 | 1.1932E+00 | o | o |
| estein5_C | 132 | 211 | 258 | 289 | [3:18] | 18 | 0 | 0 | 0 | 1.4991E+00 | o | o |
| estein5_nr1 | 132 | 211 | 258 | 289 | [3:18] | 18 | 0 | 0 | 0 | 1.6644E+00 | o | o |
| estein5_nr21 | 132 | 211 | 258 | 289 | [3:18] | 18 | 0 | 0 | 0 | 1.8182E+00 | o | o |

**Table 3** continued

| Instances | Size | | | Conic domains | | Binary | | Integer | | Status | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | var | map | nnz | lin | so | lin | so | lin | so | obj | M | C |
| *filterdesign* | | | | | | | | | | | | |
| 2013_dsNRL | 61,822 | 1616 | 66,668,564 | 1616 | [3:20,503, 313:1] | | | | | −9.6379E−06 | O | O |
| 2013_firL1 | 59,706 | 20,902 | 39,787,428 | 20,902 | [3:19,902] | | | | | −3.6669E+00 | O | O |
| 2013_firL1Linfalph | 119,412 | 20,903 | 79,574,856 | 20,903 | [3:39,804] | | | | | −3.3116E+00 | O | - |
| 2013_firL1Linfeps | 59,173 | 30,085 | 9,873,426 | 30,086 | [3:19,724] | | | | | −1.5255E−02 | O | - |
| 2013_firL2L1alph | 49,612 | 30,268 | 9,985,771 | 30,269 | [3:9922, 19,845:1] | | | | | −2.4441E−01 | O | O |
| 2013_firL2L1eps | 60,708 | 20,903 | 40,288,929 | 20,903 | [3:19,902, 1002:1] | | | | | −3.0683E+00 | O | O |
| 2013_firL2Linfalph | 91,783 | 2002 | 121,660,011 | 2002 | [3:29,927, 2002:1] | | | | | −7.7910E−02 | O | O |
| 2013_firL2Linfeps | 59,636 | 24,655 | 19,108,570 | 24,655 | [3:11,927, 23,855:1] | | | | | −1.0141E−02 | O | O |
| 2013_firL2a | 10,002 | 10,001 | 50,015,001 | 10,001 | [10,002:1] | | | | | −1.4368E−01 | O | O |
| 2013_firLinf | 59,856 | 2001 | 79,771,354 | 2001 | [3:19,952] | | | | | −1.0022E−02 | O | - |
| 2013_wbNRL | 40,450 | 1042 | 39,138,234 | 38,123 | [38:7, 1035:1, 2068:1] | | | | | −3.8759E−05 | O | O |
| 2013i_wbNRL | 63,312 | 1710 | 101,934,231 | 59,827 | [51:4, 52:3, 1431:1, 3404:1] | | | | | Unbounded | DI | P |
| *nb* | | | | | | | | | | | | |
| nb | 2383 | 123 | 191,519 | 127 | [3:793] | | | | | −5.0703E−02 | O | O |
| nb_L1 | 3176 | 915 | 192,312 | 1712 | [3:793] | | | | | −1.3012E+01 | O | O |
| nb_L2 | 4195 | 123 | 402,285 | 127 | [3:838, 1677:1] | | | | | −1.6290E+00 | O | O |
| nb_L2_bessel | 2641 | 123 | 208,817 | 127 | [3:838, 123:1] | | | | | −1.0257E−01 | O | O |
| *portfoliocard* | | | | | | | | | | | | |
| classical_50_1 | 152 | 255 | 2902 | 356 | [51:1] | 50 | 0 | 0 | 0 | −9.4760E−02 | O | O |
| classical_50_2 | 152 | 255 | 2902 | 356 | [51:1] | 50 | 0 | 0 | 0 | −9.0528E−02 | O | O |
| classical_50_3 | 152 | 255 | 2902 | 356 | [51:1] | 50 | 0 | 0 | 0 | −8.8041E−02 | O | O |
| classical_200_1 | 602 | 1005 | 41,602 | 1406 | [201:1] | 200 | 0 | 0 | 0 | −1.1668E−01[V] | P | P |

**Table 3** continued

| Instances | Size | | | Conic domains | | Binary | | Integer | | Status | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | var | map | nnz | lin | so | lin | so | lin | so | obj | M | C |
| classical_200_2 | 602 | 1005 | 41,602 | 1406 | [201:1] | 200 | 0 | 0 | 0 | −1.1009E−01$^{v}$ | P | P |
| classical_200_3 | 602 | 1005 | 41,602 | 1406 | [201:1] | 200 | 0 | 0 | 0 | −1.0607E−01$^{v}$ | P | P |
| robust_50_1 | 207 | 365 | 5564 | 468 | [52:2] | 51 | 0 | 0 | 0 | −8.5695E−02 | O | O |
| robust_50_2 | 207 | 365 | 5564 | 468 | [52:2] | 51 | 0 | 0 | 0 | −1.4365E−01 | O | O |
| robust_50_3 | 207 | 365 | 5564 | 468 | [52:2] | 51 | 0 | 0 | 0 | −8.9803E−02 | O | O |
| robust_100_1 | 407 | 715 | 21,114 | 918 | [102:2] | 101 | 0 | 0 | 0 | −7.2090E−02 | O | P |
| robust_100_2 | 407 | 715 | 21,114 | 918 | [102:2] | 101 | 0 | 0 | 0 | −9.1574E−02 | O | O |
| robust_100_3 | 407 | 715 | 21,114 | 918 | [102:2] | 101 | 0 | 0 | 0 | −1.1682E−01 | O | O |
| robust_200_1 | 807 | 1415 | 82,214 | 1818 | [202:2] | 201 | 0 | 0 | 0 | −1.4275E−01 | O | P |
| robust_200_2 | 807 | 1415 | 82,214 | 1818 | [202:2] | 201 | 0 | 0 | 0 | −1.2167E−01 | O | P |
| robust_200_3 | 807 | 1415 | 82,214 | 1818 | [202:2] | 201 | 0 | 0 | 0 | −1.2911E−01$^{v}$ | P | P |
| shortfall_50_1 | 205 | 361 | 5612 | 464 | [51:2] | 51 | 0 | 0 | 0 | −1.1018E+00 | O | O |
| shortfall_50_2 | 205 | 361 | 5612 | 464 | [51:2] | 51 | 0 | 0 | 0 | −1.0952E+00 | O | O |
| shortfall_50_3 | 205 | 361 | 5612 | 464 | [51:2] | 51 | 0 | 0 | 0 | −1.0923E+00 | O | O |
| shortfall_100_1 | 405 | 711 | 21,212 | 914 | [101:2] | 101 | 0 | 0 | 0 | −1.1063E+00 | O | P |
| shortfall_100_2 | 405 | 711 | 21,212 | 914 | [101:2] | 101 | 0 | 0 | 0 | −1.1007E+00$^{v}$ | P | P |
| shortfall_100_3 | 405 | 711 | 21,212 | 914 | [101:2] | 101 | 0 | 0 | 0 | −1.1031E+00 | O | P |
| shortfall_200_1 | 805 | 1411 | 82,412 | 1814 | [201:2] | 201 | 0 | 0 | 0 | −1.1354E+00$^{v}$ | P | P |
| shortfall_200_2 | 805 | 1411 | 82,412 | 1814 | [201:2] | 201 | 0 | 0 | 0 | −1.1254E+00$^{v}$ | P | P |
| shortfall_200_3 | 805 | 1411 | 82,412 | 1814 | [201:2] | 201 | 0 | 0 | 0 | −1.1199E+00$^{v}$ | P | P |
| *pp* | | | | | | | | | | | | |
| pp-n10-d10 | 50 | 31 | 59 | 51 | [3:10] | 0 | 10 | 0 | 0 | 7.2481E+01 | O | O |
| pp-n10-d10000 | 50 | 31 | 59 | 51 | [3:10] | 0 | 10 | 0 | 0 | 1.4815E+03 | O | – |

**Table 3** continued

| Instances | Size | | | Conic domains | | Binary | | Integer | | Status | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | var | map | nnz | lin | so | lin | so | lin | so | obj | M | C |
| pp-n100-d10 | 500 | 301 | 599 | 501 | [3:100] | 0 | 100 | 0 | 0 | 7.7728E+02[v] | P | P |
| pp-n100-d10000 | 500 | 301 | 597 | 501 | [3:100] | 0 | 100 | 0 | 0 | 1.9856E+04 | O | – |
| pp-n1000-d10 | 5000 | 3001 | 5969 | 5001 | [3:1000] | 0 | 1000 | 0 | 0 | 7.3434E+03[v] | P | P |
| pp-n1000-d10000 | 5000 | 3001 | 5968 | 5001 | [3:1000] | 0 | 1000 | 0 | 0 | 2.1611E+05 | P | – |
| pp-n100000-d10 | 500,000 | 300,001 | 597,382 | 500,001 | [3:100,000] | 0 | 100,000 | 0 | 0 | 0.0000E+00[a] | – | – |
| pp-n100000-d10000 | 500,000 | 300,001 | 597,463 | 500,001 | [3:100,000] | 0 | 100,000 | 0 | 0 | 1.8348E+07[a] | – | – |
| *sched* | | | | | | | | | | | | |
| sched_50_50_orig | 4979 | 2527 | 25,488 | 5029 | [3:1, 2474:1] | | | | | 2.6673E+04[a] | – | – |
| sched_50_50_scaled | 4977 | 2526 | 27,985 | 5028 | [2475:1] | | | | | 7.8520E+00 | O | – |
| sched_100_50_orig | 9746 | 4844 | 55,291 | 9846 | [3:1, 4741:1] | | | | | 1.8189E+05 | – | O |
| sched_100_50_scaled | 9744 | 4843 | 60,288 | 9845 | [4742:1] | | | | | 6.7165E+01 | O | – |
| sched_100_100_orig | 18,240 | 8338 | 104,902 | 18,340 | [3:1, 8235:1] | | | | | 7.1737E+05 | – | O |
| sched_100_100_scaled | 18,238 | 8337 | 114,899 | 18,339 | [8236:1] | | | | | 2.7331E+01 | O | – |
| sched_200_100_orig | 37,889 | 18,087 | 260,503 | 38,089 | [3:1, 17,884:1] | | | | | 1.4136E+05 | – | O |
| sched_200_100_scaled | 37,887 | 18,086 | 280,500 | 38,088 | [17,885:1] | | | | | 5.1812E+01 | O | – |
| *sssd* | | | | | | | | | | | | |
| sssd-strong-15-4 | 125 | 180 | 372 | 269 | [3:12] | 72 | 0 | 0 | 0 | 3.2800E+05 | O | O |
| sssd-strong-15-8 | 249 | 344 | 744 | 521 | [3:24] | 144 | 0 | 0 | 0 | 6.2251E+05 | O | O |
| sssd-strong-20-4 | 145 | 205 | 432 | 314 | [3:12] | 92 | 0 | 0 | 0 | 2.8781E+05 | O | O |
| sssd-strong-20-8 | 289 | 389 | 864 | 606 | [3:24] | 184 | 0 | 0 | 0 | 6.0035E+05 | O | O |
| sssd-strong-25-4 | 165 | 230 | 492 | 359 | [3:12] | 112 | 0 | 0 | 0 | 3.1172E+05 | O | O |
| sssd-strong-25-8 | 329 | 434 | 984 | 691 | [3:24] | 224 | 0 | 0 | 0 | 5.0075E+05 | P | O |
| sssd-strong-30-4 | 185 | 255 | 552 | 404 | [3:12] | 132 | 0 | 0 | 0 | 2.6413E+05 | O | O |

**Table 3** continued

| Instances | Size | | | Conic domains | | Binary | | Integer | | Status | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | var | map | nnz | lin | so | lin | so | lin | so | obj | M | C |
| sssd-strong-30-8 | 369 | 479 | 1104 | 776 | [3:24] | 264 | 0 | 0 | 0 | 5.2876E+05 | P | O |
| sssd-weak-15-4 | 125 | 180 | 360 | 269 | [3:12] | 72 | 0 | 0 | 0 | 3.2800E+05 | O | O |
| sssd-weak-15-8 | 249 | 344 | 720 | 521 | [3:24] | 144 | 0 | 0 | 0 | 6.2251E+05 | O | O |
| sssd-weak-20-4 | 145 | 205 | 420 | 314 | [3:12] | 92 | 0 | 0 | 0 | 2.8781E+05 | O | O |
| sssd-weak-20-8 | 289 | 389 | 840 | 606 | [3:24] | 184 | 0 | 0 | 0 | 6.0034E+05 | P | O |
| sssd-weak-25-4 | 165 | 230 | 480 | 359 | [3:12] | 112 | 0 | 0 | 0 | 3.1172E+05 | O | O |
| sssd-weak-25-8 | 329 | 434 | 960 | 691 | [3:24] | 224 | 0 | 0 | 0 | 5.0075E+05 | P | O |
| sssd-weak-30-4 | 185 | 255 | 540 | 404 | [3:12] | 132 | 0 | 0 | 0 | 2.6413E+05 | O | O |
| sssd-weak-30-8 | 369 | 479 | 1080 | 776 | [3:24] | 264 | 0 | 0 | 0 | 5.2876E+05 | P | O |
| *strain* | | | | | | | | | | | | |
| nql30 | 4501 | 6380 | 20,569 | 8181 | [3:900] | | | | | −9.4602E−01 | O | O |
| nql60 | 18,001 | 25,360 | 82,539 | 32,561 | [3:3600] | | | | | −9.3504E−01 | O | O |
| nql90 | 40,501 | 56,940 | 185,909 | 73,141 | [3:8100] | | | | | −9.3136E−01 | O | O |
| nql180 | 162,001 | 227,280 | 744,419 | 292,081 | [3:32,400] | | | | | −9.2764E−01 | O | O |
| qssp30 | 7565 | 11,255 | 44,414 | 11,256 | [4:1891] | | | | | −6.4967E+00 | O | O |
| qssp60 | 29,525 | 44,105 | 178,814 | 44,106 | [4:7381] | | | | | −6.5627E+00 | O | O |
| qssp90 | 65,885 | 98,555 | 403,214 | 98,556 | [4:16,471] | | | | | −6.5942E+00 | O | O |
| qssp180 | 261,365 | 391,505 | 1,616,414 | 391,506 | [4:65,341] | | | | | −6.6391E+00 | O | O |
| *turbine* | | | | | | | | | | | | |
| turbine07 | 84 | 101 | 313 | 101 | [3:25, 9:1] | 0 | 0 | 11 | 0 | 2.0000E+00 | O | O |
| turbine07GF | 87 | 124 | 444 | 136 | [3:25] | 12 | 0 | 0 | 0 | 3.0000E+00 | P | O |
| turbine07_aniso | 83 | 108 | 313 | 116 | [3:25] | 0 | 0 | 11 | 0 | 3.0000E+00 | O | O |
| turbine07_lowb | 212 | 354 | 621 | 480 | [2:1, 3:25, 9:1] | 56 | 0 | 0 | 0 | 8.9930E−01 | O | O |

**Table 3** continued

| Instances | Size | | | Conic domains | | Binary | | Integer | | Status | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | var | map | nnz | lin | so | lin | so | lin | so | obj | M | C |
| turbine07_lowb_aniso | 210 | 361 | 621 | 496 | [3:25] | 56 | 0 | 0 | 0 | 1.3945E+00 | - | O |
| turbine54 | 366 | 477 | 2099 | 477 | [3:119, 9:1] | 0 | 0 | 11 | 0 | 3.0000E+00 | P | O |
| turbine54GF | 369 | 500 | 2982 | 512 | [3:119] | 12 | 0 | 0 | 0 | 4.0000E+00 | P | O |
| *uflquad* | | | | | | | | | | | | |
| uflquad-nopsc-10-100 | 3011 | 5111 | 7010 | 5122 | [3:1000] | 10 | 0 | 0 | 0 | 5.4029E+02 | O | O |
| uflquad-nopsc-10-150 | 4511 | 7661 | 10,510 | 7672 | [3:1500] | 10 | 0 | 0 | 0 | 7.0965E+02 | O | O |
| uflquad-nopsc-20-100 | 6021 | 10,121 | 14,020 | 10,142 | [3:2000] | 20 | 0 | 0 | 0 | 3.9954E+02 | O | O |
| uflquad-nopsc-20-150 | 9021 | 15,171 | 21,020 | 15,192 | [3:3000] | 20 | 0 | 0 | 0 | 5.6872E+02 | O | O |
| uflquad-nopsc-30-100 | 9031 | 15,131 | 21,030 | 15,162 | [3:3000] | 30 | 0 | 0 | 0 | 3.5524E+02 | O | P |
| uflquad-nopsc-30-150 | 13,531 | 22,681 | 31,530 | 22,712 | [3:4500] | 30 | 0 | 0 | 0 | 4.6816E+02[v] | P | P |
| uflquad-nopsc-30-200 | 18,031 | 30,231 | 42,030 | 30,262 | [3:6000] | 30 | 0 | 0 | 0 | 5.5491E+02[v] | P | P |
| uflquad-nopsc-30-300 | 27,031 | 45,331 | 63,030 | 45,362 | [3:9000] | 30 | 0 | 0 | 0 | 7.8479E+02[v] | P | P |
| uflquad-psc-10-100 | 3011 | 5111 | 8010 | 5122 | [3:1000] | 10 | 0 | 0 | 0 | 5.4029E+02 | O | O |
| uflquad-psc-10-150 | 4511 | 7661 | 12,010 | 7672 | [3:1500] | 10 | 0 | 0 | 0 | 7.0965E+02 | O | O |
| uflquad-psc-20-100 | 6021 | 10,121 | 16,020 | 10,142 | [3:2000] | 20 | 0 | 0 | 0 | 3.9954E+02 | O | O |
| uflquad-psc-20-150 | 9021 | 15,171 | 24,020 | 15,192 | [3:3000] | 20 | 0 | 0 | 0 | 5.6872E+02 | O | O |
| uflquad-psc-30-100 | 9031 | 15,131 | 24,030 | 15,162 | [3:3000] | 30 | 0 | 0 | 0 | 3.5524E+02 | O | O |
| uflquad-psc-30-150 | 13,531 | 22,681 | 36,030 | 22,712 | [3:4500] | 30 | 0 | 0 | 0 | 4.6816E+02 | O | O |
| uflquad-psc-30-200 | 18,031 | 30,231 | 48,030 | 30,262 | [3:6000] | 30 | 0 | 0 | 0 | 5.5491E+02 | O | O |
| uflquad-psc-30-300 | 27,031 | 45,331 | 72,030 | 45,362 | [3:9000] | 30 | 0 | 0 | 0 | 7.6035E+02 | O | O |

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

[v(alue)] Objective neither claimed by a solver to be within an absolute and relative gap of 0.0 from optimality (mixed-integer case), nor certified to be within an absolute gap of $10^{-4}$ or relative gap of $10^{-7}$ from optimality (continuous case)

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 might also be implied integer.

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 100000 (three instances).

We now elaborate on the differences between MOSEK and CPLEX as shown in Table 3, starting with instance `2013i_wbNRL`. This instance 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, and this direction is a dual infeasibility certificate. MOSEK found this certificate, while CPLEX terminated with a primal feasible point.

Another observation from Table 3 is that the ”best” formulation is not always clear. MOSEK terminated with primal infeasibilities on all `sched_*_*_orig` instances, but solved all of the `sched_*_*_scaled` instances just fine. Thus, what can be solved and not is exactly opposite to CPLEX, with `sched_50_50_orig` as the only exception for which CPLEX also terminated with primal infeasibilities. Only together, were they able to solve nearly all of the `sched` instances.

Numerical issues are unfortunately not an isolated case, however, as CPLEX also terminated with primal infeasibilities on `2013_firL1Linfalph` as well as on `2013_firL1Linfeps`. Moreover, MOSEK refused to claim optimality on `turbine07GF`, `turbine54` and `turbine54GF`, even though terminating in time with the optimal solutions, presumably because numerical issues forced it to skip subproblems rather than to prune them from the search tree. There were also integer problems where optimality was claimed, but infeasible solutions were returned. This happened for CPLEX on all of the `pp-*-d10000` instances and for MOSEK on `turbine07_lowb_aniso`. In one case, `sssd-weak-30-8`, MOSEK moreover seems to have cut off the optimal solution as it claimed optimality although an objective improvement of 5.4 in absolute and 1.0E−05 in relative measures could be achieved.

Finally, as indication of the hardness of these instances in terms different from numerical issues, neither CPLEX nor MOSEK were able to find any feasible solution to `pp-n100000-d10` in time. Interestingly, this is a trivial task to perform by hand as seen by consulting the mathematical model [58]. It is also worth pointing out that the continuous instances of the filterdesign package are absolutely huge, and CPLEX actually timed out on `2013_dsNRL` and `2013_firLinf`. This, despite actually outputting a valid solution and optimality certificate in the former case. On the integer problems, CPLEX and MOSEK timed out 20 and 21 times respectively. The 14 instances on which they both timed out is given by the special case

of `pp-n100000-d10` (no solutions found) and otherwise match when the letter `P` appears simultaneously in column *M* and *C* of Table 3.

### 5.1 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 all occurrences of `||*|...||` with the value of the filter `*` given arguments `...`, and evaluate it as a boolean expression. Instances evaluating to true are listed.

```
python filter.py "||cones|so|| == ||cones|so|==3||"
```

Instances where all second-order cones have exactly three entries. In this command, `||cones||` counts the number of conic domains, and takes two arguments to limit its scope. The first argument specifies a cone type following the CBF format, with linear cones `F`, `L+`, `L-`, `L=`, or all four, `lin`, as well as second-order cones `Q`, `QR`, or both, `so`. The second argument is a relation with cone dimension as left-hand-side.

```
python filter.py "||int|| and ||cones|so||
 and not ||psdcones||"
```

Mixed-integer second-order cone instances. This command uses Python boolean logic with `||int||` counting integer variables (the subset of binary variables is found by `||binary||`), and `||psdcones||` counting semidefinite cones.

```
python filter.py "||entries|so|| / ||cones|so|| <= 4"
```

Instances with no more than four entries per second-order cone on average. This command shows the use of Python mathematics, with `||entries||` summing the dimension of cones (here limited to second-order cones).

The script also accepts an execution argument, indicated by `-x`, whose result will be evaluated and printed. With an empty filter (always true), this can be used to generate tables of instance statistics.

```
python filter.py "" -x "[||path||, ||minimize||,
 ||var|F||, ||map|L=||]"
```

Instance statistics for all instances. The filter `||path||` is filepath (`||name||` is filename without extension) and `||minimize||` is whether the objective sense is to minimize. `||var||` and `||map||` are subsets of `||entries||` limited respectively to $\mathcal{K}_x^{n_x}$ and $\mathcal{K}_g^{n_g}$ from (P). Here, the former is further limited to free variables, and the latter to equality constraints.

The execution argument can be useful for exploring the instances and filters. Note that the filtering mechanism is implemented as a plugin system which can be extended by adding functions to the directory of filters in the distributed library.

### 5.2 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 [37] and CPLEX [27] through their through its Python API. This script was, for example, used to generate the last columns 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 -f [CBFFILE1] [CBFFILE2] ...`

Runs MOSEK on the listed instances, that is, `[CBFFILE1]`, `[CBFFILE2]`, and so on. A summary of these results can be shown by `python summary.py -f [CBFFILE1] [CBFFILE2] ....`

`python run.py runmosek -s [SET]`

Runs MOSEK on the instances in `[SET]`. This can be a subdirectory of *cbf* in the distributed library, or a file formatted as the default output of the `filter.py` script (a stripped version of `ref.csv` in the distributed library). The summary is shown by `python summary.py -s [SET]`.

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 [57] is supported by `cbftool`, nothing but matrix inequalities can be converted to this format. The tool supports the two extensions of the MPS format, mentioned in the introduction, to facilitate second-order cones. Examples of this are given below.

`cbftool -o mps-mosek [CBFFILE1] [CBFFILE2] ...`

Convert listed instances to the MPS format using the explicit second-order cone extension. Results are stored in the current directory.

`cbftool -o mps-cplex -opath [OUTPUTDIR] [CBFFILE1] [CBFFILE2] ...`

Convert listed instances to the MPS format using the quadratic extension with nonnegative variable bounds for second-order cones. Results are stored in `[OUTPUTDIR]`.

## 6 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. Several issues have been identified in the availability of benchmark libraries, however, which may potentially slow down progress. Some of these issues have been addressed with the release of CBLIB 2014. There is now a large collection of mixed-integer and continuous second-order cone instances, and a new CBF file format which unifies the SDPA and SEDUMI format under a common mathematical formulation.

Since this publication, CBLIB has been used by XPRESS [18] (mentioned in [43]), by MOSEK [37] and GUROBI [25] (private communication), as well as in the public benchmarks of Hans Mittelmann [36]. Moreover, the library has continued to grow from a few hundred to more than a thousand instances distributed online. While this expansion includes new applications of conic optimization, it mostly provides a wider variety of data for some of the mathematical models, and some very hard and unsolved problems which are not suited for performance benchmarks. CBLIB 2014 thus remains representative as a benchmark selection of the entire collection.

Future work includes categorizing the instances into test sets similar to the sets of open, challenging and easy instances found in MIPLIB [32]. Adding native support of the CBF format to the open source solvers and algebraic modeling tools is also of high value to the project. This has already started to happen with PICOS [47] as first mover. Finally, we are interested in instances with properties rare to the existing library such as infeasibilities (as requested in [45]) or integer variables in cones (as requested in [22]), or simply representing new applications of conic optimization.

*The Conic Benchmark Library, CBLIB, is a community project and grow through external submissions. Please consider contributing at* http://cblib.zib.de.

**Acknowledgments** The author owe a special thanks to Erling D. Andersen and Mathias Stolpe who supervised the work on CBLIB, and to Ambros Gleixner and Thorsten Koch for hosting and maintaining the benchmarking library at Zuse Institute Berlin. Another great thank you goes to the anonymous peer reviewers for their valuable feedback, and to the many who has contributed instances or given feedback to drive the benchmark library forward. The author, Henrik A. Friberg, was funded by MOSEK ApS and the Danish Ministry of Higher Education and Science through the Industrial PhD project "Combinatorial Optimization over Second-Order Cones and Industrial Applications".

## References

1. Alizadeh, F., Goldfarb, G.: Second-order cone programming. Math. Progr. **51**, 3–51 (2003)
2. Andersen, K.D., Christiansen, E., Overton, M.L.: Computing limit loads by minimizing a sum of norms. SIAM J. Sci. Comput. **19**(3), 1046–1062 (1998)
3. Anjos, M.F., Burer, S.: On handling free variables in interior-point methods for conic linear optimization. SIAM J. Optim. **18**(4), 1310–1325 (2007)
4. Ben-tal, A., Nemirovski, A.: Lectures on modern convex optimization: analysis, algorithms, and engineering applications, vol. 2 of MPS-SIAM series on optimization. SIAM. ISBN: 978-0-89871-491-3 (2001)
5. Benjamin, A.T.: Sensible rules for remembering duals–the SOB method. SIAM Rev. **37**(1), 85–87 (1995)

6. Berthold, T., Heinz, S., Vigerske, S.: Extending a CIP framework to solve MIQCPs. In: Mixed Integer Nonlinear Programming, volume 154 of the IMA Volumes in Mathematics and its Applications, pp. 427–444. Springer, New York. ISBN: 978-1-4614-1926-6 (2012)
7. Bonami, P., Kilinc, M., Linderoth, J.: Algorithms and software for convex mixed integer nonlinear programs. Mix. Int. Nonlinear Progr. **154**, 1–39 (2012)
8. Borchers, B.: SDPLIB 1.2, a library of semidefinite programming test problems. Optim. Methods Softw. **11**(1), 683–690 (1999)
9. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press. https://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf (2004)
10. Chinneck, J.W.: Feasibility and infeasibility in optimization. In: International Series in Operations Research and Management Science, vol. 118, Springer, US, Boston, MA. ISBN: 978-0-387-74931-0 (2008)
11. Christiansen, E., Andersen, K.D.: Computation of collapse states with von Mises type yield condition. Int. J. Numer. Meth. Eng. **46**(8), 1185–1202 (1999)
12. Coleman, J.O., Vanderbei, R.J.: 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, vol. 2, pp. 761–764 (1999)
13. Coleman, J.O., Scholnik, D.P., Brandriss, J.J.: 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)
14. Conn, A.R., Gould, N.I.M., Toint, P.L.: Testing a class of methods for solving minimization problems with simple bounds on the variables. Math. Comput. **50**(182), 399–430 (1988)
15. de Klerk, E., Sotirov, R.: A new library of structured semidefinite programming instances. Optim. Methods Softw. **24**(6), 959–971 (2009)
16. Drewes, S.: Mixed integer second order cone programming. PhD thesis, Department of Mathematics, Technical University Darmstadt. http://www3.mathematik.tu-darmstadt.de/fileadmin/home/users/82/DissertationDrewes.pdf (2009)
17. Elhedhli, S.: Service system design with immobile servers, stochastic demand, and congestion. Manuf Serv Oper Manag **8**(1), 92–97 (2006)
18. Fair Isaac Corporation. Xpress-optimizer reference manual, Release 20.00. Technical report XPRESS2013. http://www.fico.com/en/wp-content/secure_upload/Xpress-Optimizer-Refernce-Manual.pdf (2009)
19. Friberg, H.A.: The conic benchmark format: version 1 - technical reference manual. Technical Report E-0047, Department of Wind Energy, Technical University of Denmark. http://orbit.dtu.dk/services/downloadRegister/88492586/Conic_Benchmark_Format.pdf (2014)
20. Gay, D.M.: Electronic mail distribution of linear programming test problems. Math. Progr. Soc. COAL Newslett. **13**, 10–12 (1985)
21. Glineur, F.: Conic optimization: an elegant framework for convex optimization. Belg. J. Oper. Res. Stat. Comput. Sci. **41**, 5–28 (2001)
22. Góez, J.C.: Mixed integer second order cone optimization disjunctive conic cuts: theory and experiments. PhD thesis, Lehigh University (2013)
23. Gruber, G., Rendl, F.: Computational experience with ill-posed problems in semidefinite programming. Comput. Optim. Appl. **21**(2), 201–212 (2002)
24. Günlük, O., Lee, J., Weismantel, R.: MINLP strengthening for separable convex quadratic transportation-cost UFL. Technical report, IBM Research Report RC24213. http://domino.watson.ibm.com/library/cyberdig.nsf/papers/F93A77B97033B1878525729F0051C343/$File/rc24213.pdf (2007)
25. Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual - Version 5.6. Technical report. http://www.gurobi.com/documentation/5.6/reference-manual/refman.pdf (2013)
26. Härter, V., Jansson, C., Lange, M.: VSDP: A Matlab toolbox for verified semidefinite-quadratic-linear programming. Technical report, Institute for Reliable Computing, Hamburg University of Technology. http://www.ti3.tu-harburg.de/jansson/vsdp/VSDP2012Guide.pdf, (2012)
27. IBM Corporation. IBM ILOG CPLEX optimization studio V12.6.0 documentation. Technical report. www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.0 (2014)
28. Iwane, H., Yanami, H., Anai, H., Yokoyama, K.: An effective implementation of symbolic-numeric cylindrical algebraic decomposition for quantifier elimination. Theor. Comput. Sci. **479**, 43–69 (2013)

29. Jansson, C.: Guaranteed accuracy for conic programming problems in vector lattices. Technical report, Institute for Reliable Computing, Technical University Hamburg. arXiv:0707.4366v1 (2007)
30. Kobayashi, K., Kim, S., Kojima, M.: Sparse second order cone programming formulations for convex optimization problems. J. Oper. Res. Soc. Jpn **51**(3), 241–264 (2008)
31. Koch, T.: The final NETLIB-LP results. Oper Res Lett **32**(2), 138–142 (2004)
32. Koch, T., Achterberg, T., Andersen, E.D., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H.D., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math Program Comput **3**(2), 103–163 (2011)
33. Lobo, M.S., Vandenberghe, L., Boyd, S., Lebret, H.: Applications of second-order cone programming. Linear Algebra Appl. **284**, 193–228 (1998)
34. Markowitz, H.: Portfolio selection. J. Financ. **7**(1), 77–91 (1952)
35. Mars, S., Schewe, L.: An SDP-package for SCIP. Technical Report August, TU Darmstadt. http://www.opt.tu-darmstadt.de/~smars/scip_sdp.html (2012)
36. Mittelmann, H.D.: MISOCP and large SOCP benchmark. http://plato.asu.edu/ftp/socp.html (2014)
37. MOSEK ApS. The MOSEK C optimizer API manual, Version 7.0. technical report. http://docs.mosek.com/7.0/capi.pdf (2013)
38. Nesterov, Y., Nemirovskii, A.: Interior-point polynomial algorithms in convex programming, vol. 13. SIAM, Philadelphia. ISBN: 978-0-89871-319-0 (1994)
39. Nesterov, Y., Todd, M.J., Ye, Y.: Infeasible-start primal-dual methods and infeasibility detectors for nonlinear programming problems. Math. Progr. **84**(2), 227–267 (1999)
40. Pataki, G., Schmieta, S.H.: The DIMACS library of semidefinite-quadratic-linear programs. Technical report, Computational Optimization Research Center, Columbia University. http://dimacs.rutgers.edu/Challenges/Seventh/Instances/lib.ps (2002)
41. Permenter, F., Parrilo, P.A.: Partial facial reduction: simplified, equivalent SDPs via approximations of the PSD cone, (2014). arXiv:1408.4685
42. Permenter, F., Friberg, H.A., Andersen, E.D.: Solving conic optimization problems via self-dual embedding and facial reduction: a unified approach. http://www.optimization-online.org/DB_HTML/2015/09/5104.html (2015)
43. Perregaard, M.: Advances in convex quadratic integer optimization with xpress (2014). In: Presented at INFORMS Annual Meeting 2014
44. Pólik, I.: Conic optimization software. In: Cochran, J.J., Cox Jr, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (eds.) Wiley Encyclopedia of Operations Research and Management Science. Wiley, New York (2011)
45. Pólik, I., Terlaky, T.: New stopping criteria for detecting infeasibility in conic optimization. Optim. Lett. **3**(2), 187–198 (2009)
46. Renegar, J.: Incorporating condition measures into the complexity theory of linear programming. SIAM J. Optim. **5**(3), 506–524
47. Sagnol, G.: PICOS: a python interface for conic optimization solvers. http://picos.zib.de (2015)
48. Skajaa, A., Andersen, E.D., Ye, Y.: Warmstarting the homogeneous and self-dual interior point method for linear and conic quadratic problems. Math. Progr. Comput. **5**(1), 1–25 (2012)
49. Skutella, M.: Convex quadratic and semidefinite programming relaxations in scheduling. J. ACM **48**(2), 206–242 (2001)
50. Sturm, J.F.: Using sedumi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim Methods Softw **11–12**, 625–653 (1999)
51. The MathWorks, Inc. MATLAB Primer, R2013b. Technical report. http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf (2013)
52. Tütüncü, R.H., Toh, K.C., Todd, M.J.: Solving semidefinite-quadratic-linear programs using SDPT3. Math. Progr. **95**(2), 189–217 (2003)
53. Vielma, J.P., Ahmed, S., Nemhauser, G.L.: A lifted linear programming branch-and-bound algorithm for mixed integer conic quadratic programs. INFORMS J. Comput. **20**, 438–450 (2008)
54. Waki, H., Muramatsu, M.: Facial reduction algorithms for conic optimization problems. J. Optim. Theor Appl. **158**(1), 188–215 (2013)
55. Waki, H., Nakata, M., Muramatsu, M.: Strange behaviors of interior-point methods for solving semidefinite programming problems in polynomial optimization. Comput Optim Appl **53**(3), 823–844 (2011)

56. Wiegele, A.: Biq Mac library—a collection of max-cut and quadratic 0–1 programming instances of medium size Quadratic 0–1 Programming problems. Technical report, Alpen-Adria-Universität Klagenfurt. http://biqmac.uni-klu.ac.at/biqmaclib.html (2007)
57. Yamashita, M., Fujisawa, K., Nakata, K., Nakata, M., Fukuda, M., Kobayashi, K., Goto, K.: A high-performance software package for semidefinite programs: SDPA 7. Technical report. http://www.optimization-online.org/DB_HTML/2010/01/2531.html (2010)
58. Ziegler, H.: Solving certain singly constrained convex optimization problems in production planning. Oper Res. Lett. **1**(6), 246–252 (1982)