COSMO.jl

A conic operator splitting method for large convex conic problems

Michael Garstka · Paul Goulart · Mark Cannon

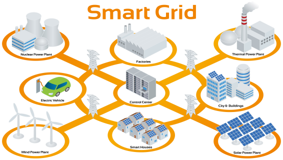
JuMP-dev workshop, Santiago, Chile
13th March 2019

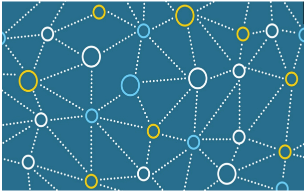
Motivation

Why do we care about solving large convex conic programs?

Motivation

Why do we care about solving large convex conic programs?





Overview

Conic Problem Format

ADMM Algorithm

Example: Nearest correlation matrix

Chordal decomposition of PSD constraints

Example: Block-arrow structured SDPs

Implementation

Problem Format

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

- Decision variables: $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$
- Problem data: real matrices $P \succeq 0$, A, and real vectors q, b
- Convex cone $\mathcal{K}$ which can be a Cartesian product of cones:

$$\mathcal{K} = \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \cdots \times \mathcal{K}_N^{m_N}, \quad \text{where} \sum_{i=1}^{N} m_i = m$$

Problem Format

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^TPx + q^Tx \\ \text{subject to} & Ax + s = b \\ & s \in \{0\}^{m_1} \times \mathbb{R}_+^{m_2} \end{array}$$

Linear Program

- Decision variables: $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$
- Problem data: real matrices $P \succeq 0$, A, and real vectors q, b
- Convex cone $\mathcal{K}$ which can be a Cartesian product of cones:

$$\mathcal{K} = \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \cdots \times \mathcal{K}_N^{m_N}, \quad \text{where} \sum_{i=1}^{N} m_i = m$$

Problem Format

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & Ax + s = b \\ & \text{mat}(s) \succeq 0 \end{array}$$

Semidefinite Program

- Decision variables: $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$
- Problem data: real matrices $P \succeq 0$, A, and real vectors q, b
- Convex cone $\mathcal{K}$ which can be a Cartesian product of cones:

$$\mathcal{K} = \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \cdots \times \mathcal{K}_N^{m_N}, \quad \text{where} \sum_{i=1}^{N} m_i = m$$

Generic ADMM

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

- Augmented Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2,$$

- ADMM steps:

Generic ADMM

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

- Augmented Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2,$$

- ADMM steps:

$$x^{k+1} := \underset{x}{\text{argmin}}\, L_\rho(x, z^k, y^k)$$

Generic ADMM

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

- Augmented Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2,$$

- ADMM steps:

$$x^{k+1} := \underset{x}{\text{argmin}}\, L_\rho(x, z^k, y^k)$$

$$z^{k+1} := \underset{z}{\text{argmin}}\, L_\rho(x^{k+1}, z, y^k)$$

Generic ADMM

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

- Augmented Lagrangian:

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2,$$

- ADMM steps:

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \operatorname*{argmin}_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Splitting method

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

$$\text{minimize} \quad \overbrace{\tfrac{1}{2}\tilde{x}^T P \tilde{x} + q^T \tilde{x} + I_{Ax+s=b}(\tilde{x}, \tilde{s})}^{f(\tilde{x}, \tilde{s})} + \overbrace{I_{\mathcal{K}}(s)}^{g(x,s)}$$

$$\text{subject to } (\tilde{x}, \tilde{s}) = (x, s)$$

ADMM algorithm

Input: Initial values x^0, s^0, y^0, step sizes σ, ρ

Do

$$(\tilde{x}^{k+1}, \tilde{s}^{k+1}) = \underset{\tilde{x}, \tilde{s}}{\operatorname{argmin}}\, L_\rho\left(\tilde{x}, \tilde{s}, x^k, s^k, y^k\right)$$

equality constrained QP

$$x^{k+1} = \tilde{x}^{k+1}$$

$$s^{k+1} = \Pi_{\mathcal{K}}\left(\tilde{s}^{k+1} + \frac{1}{\rho} y^k\right)$$

projection onto $\mathcal{K}$

$$y^{k+1} = y^k + \rho\left(\tilde{s}^{k+1} - s^{k+1}\right)$$

while *termination criteria not satisfied*

Solving the equality constrained quadratic program

Equality constrained QP:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\tilde{x}^T P \tilde{x} + q^T \tilde{x} + \frac{\sigma}{2}\|\tilde{x} - x^k\|_2^2 + \frac{\rho}{2}\|\tilde{s} - s^k + \frac{1}{\rho}y^k\|_2^2 \\ \text{subject to} \quad & A\tilde{x} + \tilde{s} = b \end{aligned}$$

KKT system:

$$\begin{bmatrix} P + \sigma I & A^T \\ A & -\frac{1}{\rho}I \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ \nu^{k+1} \end{bmatrix} = \begin{bmatrix} -q + \sigma x^k \\ b - s^k + \frac{1}{\rho}y^k \end{bmatrix}$$

- always quasi-definite
- factorisation can be cached

$$\tilde{s}^{k+1} = s^k - \frac{1}{\rho}\left(\nu^{k+1} + y^k\right)$$

ADMM algorithm

Input: Initial values x^0, s^0, y^0, step sizes σ, ρ

Do

$$(\tilde{x}^{k+1}, \tilde{s}^{k+1}) = \underset{\tilde{x},\tilde{s}}{\text{argmin}}\, L_\rho\left(\tilde{x}, \tilde{s}, x^k, s^k, y^k\right)$$

equality constrained QP

$$x^{k+1} = \tilde{x}^{k+1}$$

$$s^{k+1} = \Pi_{\mathcal{K}}\left(\tilde{s}^{k+1} + \frac{1}{\rho} y^k\right)$$

projection onto $\mathcal{K}$

$$y^{k+1} = y^k + \rho\left(\tilde{s}^{k+1} - s^{k+1}\right)$$

while *termination criteria not satisfied*

Projection onto $\mathcal{K}$

The update equation for s becomes a projection onto $\mathcal{K}$:

$$s^{k+1} = \Pi_{\mathcal{K}}\left(\tilde{s}^{k+1} + \frac{1}{\rho} y^k\right)$$

- Projections for LPs, QPs are computationally cheap
- Projection onto positive semidefinite cone requires an eigenvalue decomposition
- Algorithms for the eigen decomposition of a N-by-N matrix have a complexity of $\mathcal{O}(N^3)$

Example: Nearest correlation matrix problem

- Given data matrix $C \in \mathbb{R}^{n \times n}$ find the nearest correlation matrix X:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|X - C\|_F^2 \\ \text{subject to} & X_{ii} = 1, \quad i = 1, \ldots, n \\ & X \in \mathbb{S}_+^n, \end{array}$$

- The objective function can be rewritten as

$$\frac{1}{2}\|X - C\|_F^2 = \frac{1}{2}x^\top x - c^\top x + \frac{1}{2}c^\top c$$

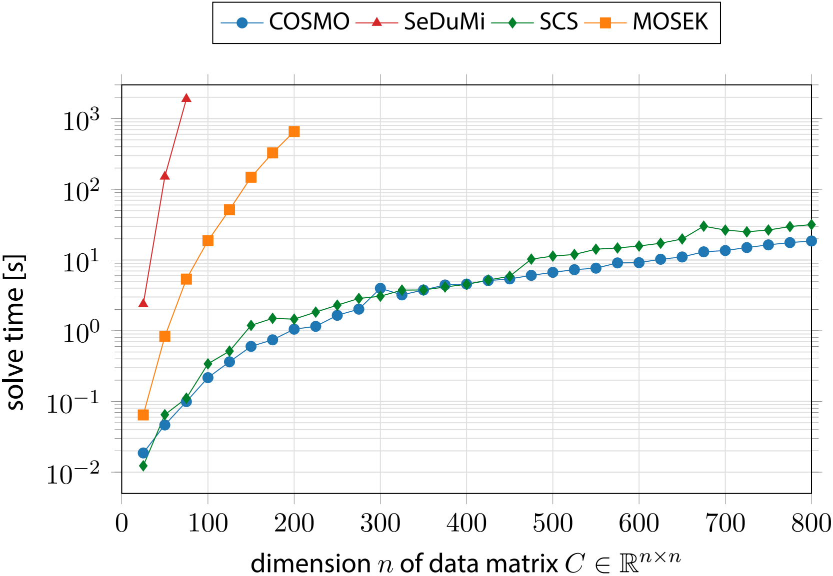
with $x = \text{vec}(X)$ and $c = \text{vec}(C)$

Example: Nearest correlation matrix problem

- We can solve this with a few lines of code with JuMP and COSMO:

```
C = rand(rng, n, n);
c = vec(C);

m = JuMP.Model(with_optimizer(COSMO.Optimizer));
@variable(m, X[1:n, 1:n], PSD);
x = vec(X);

@objective(m, Min, 0.5 * x' * x  - c' * x + 0.5 * c' * c)

for i = 1:n
  @constraint(m, X[i, i] == 1.)
end

JuMP.optimize!(m)
```

Example: Nearest correlation matrix problem



Chordal Decomposition

- **Main idea**: Replace large structured PSD constraint on S by a number of smaller PSD constraints on its subblocks

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \sum_{i=1}^{m} \mathcal{A}_i x_i + S = B \\ & S \in \mathbb{S}^r_+ \end{array}$$

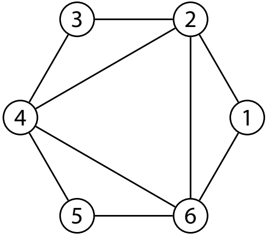
$$\begin{bmatrix} S_{11} & S_{12} & 0 & 0 & 0 & S_{16} \\ S_{21} & S_{22} & S_{23} & S_{24} & 0 & S_{26} \\ 0 & S_{32} & S_{33} & S_{34} & 0 & 0 \\ 0 & S_{42} & S_{43} & S_{44} & S_{45} & S_{46} \\ 0 & 0 & 0 & S_{54} & S_{55} & S_{56} \\ S_{61} & S_{62} & 0 & S_{64} & S_{65} & S_{66} \end{bmatrix}$$

Chordal Decomposition

- **Main idea**: Replace large structured PSD constraint on S by a number of smaller PSD constraints on its subblocks

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \sum_{i=1}^{m} \mathcal{A}_i x_i + S = B \\ & S \in \mathbb{S}_+^r \end{array}$$

$$\begin{bmatrix} S_{11} & S_{12} & 0 & 0 & 0 & S_{16} \\ S_{21} & S_{22} & S_{23} & S_{24} & 0 & S_{26} \\ 0 & S_{32} & S_{33} & S_{34} & 0 & 0 \\ 0 & S_{42} & S_{43} & S_{44} & S_{45} & S_{46} \\ 0 & 0 & 0 & S_{54} & S_{55} & S_{56} \\ S_{61} & S_{62} & 0 & S_{64} & S_{65} & S_{66} \end{bmatrix}$$

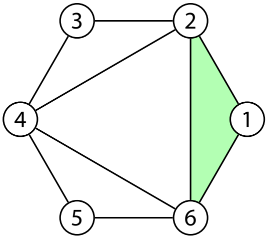


Chordal Decomposition

- **Main idea**: Replace large structured PSD constraint on S by a number of smaller PSD constraints on its subblocks

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \sum_{i=1}^{m} \mathcal{A}_i x_i + S = B \\ & S \in \mathbb{S}_+^r \end{array}$$

$$\begin{bmatrix} S_{11} & S_{12} & 0 & 0 & 0 & S_{16} \\ S_{21} & S_{22} & S_{23} & S_{24} & 0 & S_{26} \\ 0 & S_{32} & S_{33} & S_{34} & 0 & 0 \\ 0 & S_{42} & S_{43} & S_{44} & S_{45} & S_{46} \\ 0 & 0 & 0 & S_{54} & S_{55} & S_{56} \\ S_{61} & S_{62} & 0 & S_{64} & S_{65} & S_{66} \end{bmatrix}$$

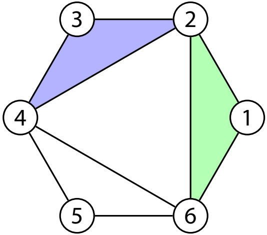


Chordal Decomposition

- **Main idea**: Replace large structured PSD constraint on S by a number of smaller PSD constraints on its subblocks

$$
\begin{aligned}
\text{minimize} \quad & \tfrac{1}{2}x^T P x + q^T x \\
\text{subject to} \quad & \sum_{i=1}^{m} \mathcal{A}_i x_i + S = B \\
& S \in \mathbb{S}_+^r
\end{aligned}
$$

$$
\begin{bmatrix}
S_{11} & S_{12} & 0 & 0 & 0 & S_{16} \\
S_{21} & S_{22} & S_{23} & S_{24} & 0 & S_{26} \\
0 & S_{32} & S_{33} & S_{34} & 0 & 0 \\
0 & S_{42} & S_{43} & S_{44} & S_{45} & S_{46} \\
0 & 0 & 0 & S_{54} & S_{55} & S_{56} \\
S_{61} & S_{62} & 0 & S_{64} & S_{65} & S_{66}
\end{bmatrix}
$$

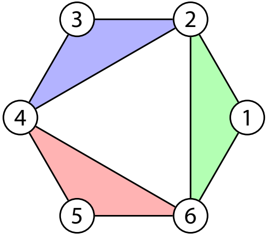


Chordal Decomposition

- **Main idea**: Replace large structured PSD constraint on S by a number of smaller PSD constraints on its subblocks

$$
\begin{aligned}
\text{minimize} \quad & \tfrac{1}{2}x^T P x + q^T x \\
\text{subject to} \quad & \sum_{i=1}^{m} \mathcal{A}_i x_i + S = B \\
& S \in \mathbb{S}^r_+
\end{aligned}
$$

$$
\begin{bmatrix}
S_{11} & S_{12} & 0 & 0 & 0 & S_{16} \\
S_{21} & S_{22} & S_{23} & S_{24} & 0 & S_{26} \\
0 & S_{32} & S_{33} & S_{34} & 0 & 0 \\
0 & S_{42} & S_{43} & S_{44} & S_{45} & S_{46} \\
0 & 0 & 0 & S_{54} & S_{55} & S_{56} \\
S_{61} & S_{62} & 0 & S_{64} & S_{65} & S_{66}
\end{bmatrix}
$$



Chordal Decomposition

- Represent aggregate sparsity pattern of S by a graph $G(V, E)$

Theorem (Agler's theorem)

Let $G(V, E)$ be a chordal graph with a set of maximal cliques $\{C_1, \dots, C_p\}$. Then $S \in \mathbb{S}^n_+(E, 0)$ if and only if there exist matrices $S_\ell \in \mathbb{S}^{|C_\ell|}_+$ for $\ell = 1, \dots, p$ such that

$$S = \sum_{\ell=1}^{p} T_\ell^T S_\ell T_\ell.$$

Chordal Decomposition

$$
\begin{aligned}
\text{minimize} \quad & \tfrac{1}{2} x^T P x + q^T x \\
\text{subject to} \quad & \sum_{i=1}^{m} \mathcal{A}_i x_i + S = B \\
& S \in \mathbb{S}_+^r(E, 0)
\end{aligned}
$$

$\Downarrow$ Agler's theorem

$$
\begin{aligned}
\text{minimize} \quad & \tfrac{1}{2} x^T P x + q^T x \\
\text{subject to} \quad & \sum_{i=1}^{m} \mathcal{A}_i x_i + \sum_{\ell=1}^{p} T_\ell^T S_\ell T_\ell = B \\
& S_\ell \in \mathbb{S}_+^{|C_\ell|}, \quad \ell = 1, \dots, p
\end{aligned}
$$

Chordal Decomposition with JuMP and COSMO

in Jupyter notebook

Example: Block-arrow structured SDPs

$$
\begin{aligned}
\text{minimize} \quad & q^T x \\
\text{subject to} \quad & \sum_{i=1}^{m} \mathcal{A}_i x_i + S = B \\
& S \in \mathbb{S}^r_+
\end{aligned}
$$

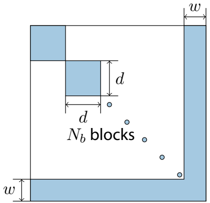
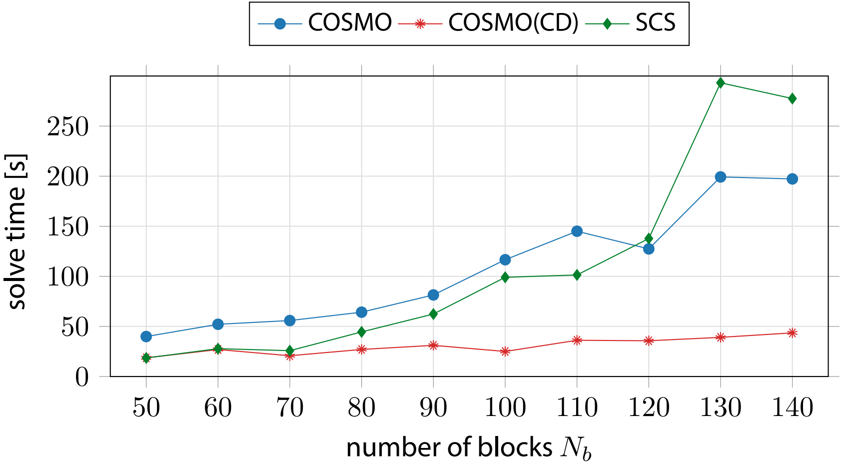


Figure: Parameters of block-arrow sparsity pattern.

Example: Block-arrow structured SDPs

- Benchmark problems: $d = 10$, $m = 100$, $N_b = 50 - 140$



Conclusion:

- open source ADMM-based solver written in Julia
- supports quadratic objectives
- supports LPs, QPs, SOCPs, SDPs
- infeasiblity detection
- chordal decomposition of PSD constraints
- allows user-defined convex sets
- supports MOI v0.8 / JuMP v0.19

Conclusion:

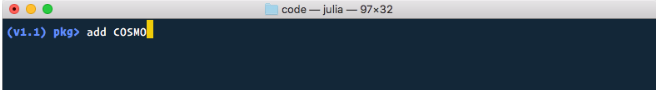
- open source ADMM-based solver written in Julia
- supports quadratic objectives
- supports LPs, QPs, SOCPs, SDPs
- infeasiblity detection
- chordal decomposition of PSD constraints
- allows user-defined convex sets
- supports MOI v0.8 / JuMP v0.19

Future work:

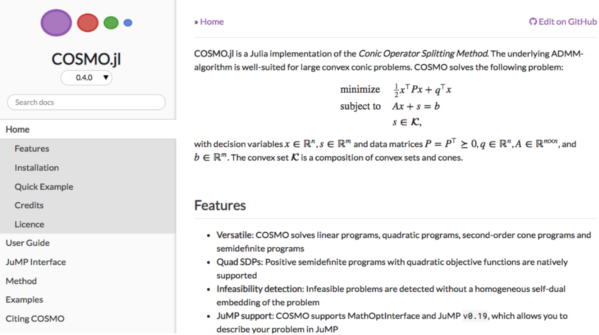
- Acceleration methods
- Approximate projections
- Parallel Implementation of projections

COSMO.jl Package

- Installation via the Julia package manager



- Code and documentation available at:
`https://github.com/oxfordcontrol/COSMO.jl`



COSMO.jl

0.4.0

Search docs

- Home
  - Features
  - Installation
  - Quick Example
  - Credits
  - Licence
- User Guide
- JuMP Interface
- Method
- Examples
- Citing COSMO

» Home Edit on GitHub

COSMO.jl is a Julia implementation of the *Conic Operator Splitting Method*. The underlying ADMM-algorithm is well-suited for large convex conic problems. COSMO solves the following problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Px + q^\top x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K}, \end{array}$$

with decision variables $x \in \mathbb{R}^n, s \in \mathbb{R}^m$ and data matrices $P = P^\top \succeq 0, q \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. The convex set $\mathcal{K}$ is a composition of convex sets and cones.

Features

- Versatile: COSMO solves linear programs, quadratic programs, second-order cone programs and semidefinite programs
- Quad SDPs: Positive semidefinite programs with quadratic objective functions are natively supported
- Infeasibility detection: Infeasible problems are detected without a homogeneous self-dual embedding of the problem
- JuMP support: COSMO supports MathOptInterface and JuMP `v0.19`, which allows you to describe your problem in JuMP

We want to solve the following semidefinite program:

$$\begin{array}{ll} \text{minimize} & q^T x \\ \text{subject to} & Ax + S = B, \quad S \succeq 0 \end{array}$$

where A and B have the same structure:

$$A = B = \begin{bmatrix} X & X & 0 & 0 \\ X & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{bmatrix}$$

Lets formulate the problem in JuMP and solve it with COSMO:

In [7]:

```
using COSMO, JuMP, LinearAlgebra

# Define problem data

A =
[0.128183  0.612346  0.0       0.0;
 0.612346  0.744476  0.526152  0.817133;
 0.0       0.526152  0.404581  0.454653;
 0.0       0.817133  0.454653  0.535701];

 B =
[0.67846   0.924571  0.0  0.0;
 0.924571  1.60899   0.794429  1.23378;
 0.0       0.794429  1.09579   0.686474;
 0.0       1.23378   0.686474  1.29377];

 q = -1.0907161041533153;
```

In [8]:

```
model = JuMP.Model(with_optimizer(COSMO.Optimizer, decompose = true, verbose = t
rue));

@variable(model, x);
@objective(model, Min, q * x);
@constraint(model, B - A .* x in JuMP.PSDCone());
JuMP.optimize!(model);
```

```
------------------------------------------------------------------
             COSMO - A Quadratic Objective Conic Solver
                         Michael Garstka
                University of Oxford, 2017 - 2018
------------------------------------------------------------------

Problem:  x ∈ R^{14},
          constraints: A ∈ R^{29x14} (38 nnz), b ∈ R^{29},
          matrix size to factor: 43x43 (1849 elem, 119 nnz)
Sets:     ZeroSet{Float64} of dim: 16
          PsdCone{Float64} of dim: 9
          PsdCone{Float64} of dim: 4
Decomp:   Num of original PSD cones: 1
          Num decomposable PSD cones: 1
          Num PSD cones after decomposition: 2
Settings: ϵ_abs = 1.0e-04, ϵ_rel = 1.0e-04,
          ϵ_prim_inf = 1.0e-06, ϵ_dual_inf = 1.0e-04,
          ρ = 0.1, σ = 1.0e-6, α = 1.6,
          max_iter = 2500,
          scaling iter = 10 (on),
          check termination every 40 iter,
          check infeasibility every 40 iter
Setup Time: 0.9ms

Iter:   Objective:      Primal Res:     Dual Res:       Rho:
40      -1.9240e+00     6.9916e-04      8.2750e-06      1.0000e-01
80      -1.9238e+00     3.1361e-13      6.3127e-13      1.0000e-01

------------------------------------------------------------------
>>> Results
Status: Solved
Iterations: 80
Optimal objective: -1.9238
Runtime: 0.008s (7.67ms)
```

You can see that the original 4x4 JuMP.PSDCone - constraint was decomposed into 2 smaller PsdCones of dimension 3x3 and 2x2