Conic Optimization
—definition, software, applications—

Michal Kočvara

School of Mathematics, The University of Birmingham

Cambridge, 20 March 2014
Conic optimization

“generalized” mathematical optimization problem

\[
\begin{align*}
\min & \quad f(x) \\
\text{subject to} & \quad g_i(x) \in K_i, \quad i = 1, \ldots, m
\end{align*}
\]

\(K_i \rightarrow \text{convex cone } \subset \mathbb{R}^{n_i}, \quad n_i \leq n\)

\(f, g_i \rightarrow \begin{cases} 
\text{linear} & \rightarrow \text{linear conic optimization, LCO} \\
\text{convex} & \rightarrow \text{convex conic optimization, CCO} \\
\text{non-convex} & \rightarrow \text{non-convex conic optimization, NCO}
\end{cases}\)
“Good” cones

Non-negative orthant $K \equiv \mathbb{R}_+^m$

Leads to standard optimization problems:

$$(g_1(x), \ldots, g_m(x)) \in \mathbb{R}_+^m \iff g_i(x) \geq 0, \ i = 1, \ldots, m$$
“Good” cones

Lorentz cone (second-order, ice cream):

\[ K \equiv L^m = \left\{ x = (x_1, \ldots, x_{m-1}; x_m)^T \in \mathbb{R}^m : \sqrt{\sum_{i=1}^{m-1} x_i^2} \leq x_m \right\} \]

Meaning:

\[(y_1, \ldots, y_k, z) \in L^{k+1} \iff \|y\| \leq z\]
“Good” cones

Cone of positive semidefinite matrices:

\[ K \equiv S^+_m = \left\{ A \in \mathbb{R}^{m \times m}; \quad A = A^T, \quad x^T A x \geq 0 \quad \forall x \in \mathbb{R}^m \right\} \]

Meaning:

\[ A \in S^+_m \iff A \succeq 0 \iff A \text{ positive semidefinite} \]
Linear conic optimization

\[ \min c^T x \]
subject to
\[ A^{(i)} x - b^{(i)} \in K_i, \quad i = 1, \ldots, m \]

\[ K_i \left\{ \begin{array}{c}
\text{non-negative orthant} \rightarrow \text{linear optimization, LP} \\
\text{second-order cone} \rightarrow \text{s.-o. conic optimization, SOCP} \\
\text{semidefinite cone} \rightarrow \text{semidefinite optimization, SDP}
\end{array} \right. \]

Algorithms

\[ K_i \left\{ \begin{array}{c}
\text{non-negative orthant} \rightarrow \text{simplex, interior-point} \\
\text{second-order cone} \rightarrow \text{interior-point, augmented Lagr.} \\
\text{semidefinite cone} \rightarrow \text{interior-point, augmented Lagr.}
\end{array} \right. \]
Linear conic optimization

\[
\begin{align*}
\text{min } & \quad c^T x \\
\text{subject to } & \quad A^{(i)} x - b^{(i)} \in K_i, \quad i = 1, \ldots, m
\end{align*}
\]

\[K_i \quad \left\{ \begin{array}{l}
\text{non-negative orthant} \rightarrow \text{linear optimization, LP} \\
\text{second-order cone} \rightarrow \text{s.-o. conic optimization, SOCP} \\
\text{semidefinite cone} \rightarrow \text{semidefinite optimization, SDP}
\end{array} \right. \]

Software

\[\left\{ \begin{array}{l}
\text{non-negative orthant} \rightarrow \text{CPLEX, Gurobi, MOSEK, PENNON,\ldots} \\
\text{second-order cone} \rightarrow \text{MOSEK, SeDuMi, SDPT3, PENNON,\ldots} \\
\text{semidefinite cone} \rightarrow \text{MOSEK, SeDuMi, SDPT3, PENNON,\ldots}
\end{array} \right. \]
Convex conic optimization

\[
\begin{align*}
\min & \quad f(x) \\
\text{subject to} & \quad g_i(x) \in K_i, \quad i = 1, \ldots, m
\end{align*}
\]

\(f, g_i\) — (smooth) convex functions

Algorithms, software

**CVX** (Michael Grant, Steve Boyd):

convex functions (like log, exp) supported by using a “successive approximation” approach; approximation by linear conic optimization, solved by LCO software.

The solver is called multiple times to refine the solution to the required accuracy.

May be slow; not all convex functions supported.
Convex conic optimization

\[ \begin{align*}
\min \ f(x) \\
\text{subject to} \quad g_i(x) &\in K_i, \quad i = 1, \ldots, m
\end{align*} \]

\(f, g_i\) — (smooth) convex functions

Algorithms, software

Interior-point methods supported theoretically (Nesterov-Nemirovski)

No efficient implementation.

Augmented Lagrangian method

PENNON
Non-convex conic optimization

\[
\begin{align*}
\min f(x) \\
\text{subject to} \\
g_i(x) &\in K_i, \quad i = 1, \ldots, m
\end{align*}
\]

\[K_i\ldots\text{nonnegative orthant} \rightarrow \text{nonlinear optimization}\]
SQP, interior-point methods
SNOPT, IPOPT, KNITRO,\ldots

\[K_i\ldots\text{second-order or semidefinite cone}\]

Algorithms, software PENNON
Linear semidefinite programming (SDP)

“generalized” mathematical optimization problem

$$\begin{align*}
\min & \quad c^T x \\
\text{subject to} & \quad \sum x_i A_i + B \succeq 0 \\
\inf & \quad \text{Tr}(BY) \\
\text{subject to} & \quad \text{Tr}(A_i Y) = b_i, \quad i = 1, \ldots, n \\
& \quad Y \succeq 0
\end{align*}$$
Linear Semidefinite Programming

Vast area of applications...

- LP and CQP is SDP
- eigenvalue optimization
- robust programming
- control theory
- relaxations of integer optimization problems
- approximations to combinatorial optimization problems
- structural optimization
- chemical engineering
- machine learning
- many many others...
Why nonlinear SDP?

Problems from

- Structural optimization
- Control theory
Other Applications, Availability

- polynomial matrix inequalities (with Didier Henrion)
- nearest correlation matrix
- sensor network localization
- approximation by nonnegative splines
- financial mathematics (with Ralf Werner)
- Mathematical Programming with Equilibrium Constraints (with Danny Ralph)
- structural optimization with matrix variables and nonlinear matrix constraints (PLATO-N EU FP6 project)
- approximation of arrival rate function of a non-homogeneous Poisson process (F. Alizadeh, J. Eckstein)
- Conic Geometric Programming (Venkat Chandrasekaran)

Many other applications. . . . . . any hint welcome

Free academic version of the code PENNON available
PENLAB available for download from my website
Case study: Topology optimization with binary variables
Truss topology design (TTD)

\[
\min_{t \in \mathbb{R}^m, \, u \in \mathbb{R}^n} f^T u
\]

s.t. \( K(t)u = f, \quad \sum_{i=1}^{m} t_i \leq V, \quad t_i \geq 0, \quad i = 1, 2, \ldots, m \)

\[
K(t) = \sum_{i=1}^{m} t_i K_i, \quad K_i = \frac{E_i}{l_i^2} \gamma_i \gamma_i^T
\]

- \( t_i \) ... bar volumes
- \( u_i \) ... displacements
- \( f^T u \) ... compliance ("inverse stiffness")
Truss topology design (TTD)

Manufacturing constraints: only few (integer) values of $t_i$ are allowed (available).

Assume: $t_i \in \{0, 1\}$ for all $i$:

$$\begin{align*}
\min_{t \in \mathbb{R}^m, \ u \in \mathbb{R}^n} & \quad f^T u \\
\text{s.t.} & \quad K(t)u = f, \quad \sum_{i=1}^m t_i \leq V, \quad t_i \in \{0, 1\}, \quad i = 1, 2, \ldots, m
\end{align*}$$

This is a Mixed-Integer NLP.

Continuous relaxation:

$$\begin{align*}
\min_{t \in \mathbb{R}^m, \ u \in \mathbb{R}^n} & \quad f^T u \\
\text{s.t.} & \quad K(t)u = f, \quad \sum_{i=1}^m t_i \leq V, \quad t_i \in [0, 1], \quad i = 1, 2, \ldots, m
\end{align*}$$
Truss topology design (TTD)


Truss topology design (TTD)

The authors, in particular, use the fact that the (dual) relaxed problem can be written as a convex QCQP

$$\min_{u,r,\lambda} -f^T u + \lambda V - \sum r_j$$

s.t. $u^T K_i u - \lambda + r_j \leq 0, \quad i = 1, 2, \ldots, m$

$$\lambda \geq 0$$

Using the fact that $K_i = b_i b_i^T$, $b_i \in \mathbb{R}^n$, this can be further formulated as a convex QP and solved very efficiently.

The authors use a number of tricks and structure of the problem to derive an efficient MINLP branch-and-bound algorithm. They solve problems with up to 750 binary variables (and 90 continuous variables).
Topology optimization

Images courtesy of FE-Design and BMW Motoren GmbH
Topological optimization

Binary:

\[
\min_{t \in \mathbb{R}^m, u \in \mathbb{R}^n} f^T u \\
\text{s.t.} \quad K(t)u = f, \quad \sum_{i=1}^{m} t_i \leq V, \quad t_i \in \{0, 1\}, \quad i = 1, 2, \ldots, m
\]

This is a MINLP.

The same formulation as in TTD but:

- \(K_i\) are not dyadic products anymore
- larger problems required

Problems with 100 binary variables are state-of-the-art
Truss topology design

Using Schur complement theorem, TTD can be written as

$$\min_{t, \gamma} \gamma$$

subject to

$$t \in \{0, 1\}, \quad i = 1, \ldots, m$$

$$Z(f; t, \gamma) \succeq 0$$

$$\sum_{i=1}^{m} t_i \leq V$$

with

$$Z(f; t, \gamma) = \begin{pmatrix} \gamma & f^T \\ f & K(t) \end{pmatrix}.$$

This is a mixed-integer linear SDP

How to solve it? Use YALMIP!
Truss topology design

YALMIP is a modelling language for advanced modeling and solution of convex and nonconvex optimization problems. It is implemented as a free toolbox for MATLAB.

The modelling language supports a large number of optimization classes, ...

Interfaces to most available conic solvers.

BNB is an implementation of a standard branch and bound algorithm for mixed integer linear/quadratic/second order cone and semidefinite programming solver. The solver relies on external solvers for solving the node problems.
YALMIP code for binary truss design

\begin{verbatim}
m=par.m; n=par.n; BI=par.BI; ff=par.f; vol=6.1;

  t=binvar(m,1); compl=sdpvar(1,1);

  Astiff=zeros(n,n);
  for i=1:m, Astiff=Astiff+t(i)*BI(i,:)'*BI(i,:); end

  Z = [compl -ff'; -ff Astiff]; F = set(Z > 0);
  F = F + set(sum(t)<=vol);

  solvesdp(F,compl,options);

  t = double(t); pic(par,t);
\end{verbatim}
Example t4x4

Figure: Initial design and relaxed solution
Example t4x4

Figure: Relaxed solution — binary solution
YALMIP output

>> tic; mincombb; toc
* Starting YALMIP integer branch & bound.
* Lower solver : PENSDP-PENOPT
* Upper solver : rounder

<table>
<thead>
<tr>
<th>Node</th>
<th>Upper</th>
<th>Gap(%)</th>
<th>Lower</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Inf</td>
<td>Inf</td>
<td>1.344E-003</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>Inf</td>
<td>Inf</td>
<td>1.344E-003</td>
<td>64</td>
</tr>
<tr>
<td>64</td>
<td>5.771E-003</td>
<td>54.56</td>
<td>1.344E-003</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>161</td>
<td>5.771E-003</td>
<td>49.13</td>
<td>1.639E-003</td>
<td>158</td>
</tr>
<tr>
<td>164</td>
<td>5.771E-003</td>
<td>49.09</td>
<td>1.641E-003</td>
<td>161</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>276</td>
<td>5.771E-003</td>
<td>48.65</td>
<td>1.666E-003</td>
<td>273</td>
</tr>
<tr>
<td>277</td>
<td>5.771E-003</td>
<td>48.65</td>
<td>1.666E-003</td>
<td>274</td>
</tr>
<tr>
<td>278</td>
<td>1.667E-003</td>
<td>0.00</td>
<td>1.666E-003</td>
<td>0</td>
</tr>
</tbody>
</table>

+ 278 Finishing. Cost: 0.0016666
Elapsed time is 50.861633 seconds.
YALMIP output

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>obj-bin</th>
<th>obj-rel</th>
<th>iter</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>86</td>
<td>24</td>
<td>1.6667</td>
<td>1.3639</td>
<td>278</td>
<td>51 sec</td>
</tr>
</tbody>
</table>

Achtziger-Stolpe (faster computer, dedicated algorithm)

<table>
<thead>
<tr>
<th>iter</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>663</td>
<td>44 sec</td>
</tr>
</tbody>
</table>

CPLEX (MIQP)

<table>
<thead>
<tr>
<th>iter</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>11200</td>
<td>76 sec</td>
</tr>
</tbody>
</table>
The message?

- Many nonlinear problems can be formulated as linear (or convex) conic problems — use it!
- Codes for mixed integer linear conic problems underdeveloped (YALMIP, BARON, ???) — and unused(?)