Smoothing methods for Second-Order Cone Programs/Complementarity Problems

Paul Tseng
University of Washington, Seattle

SIAM Conf. Optim

May, 2005
Talk Outline

- I. Second-Order Cone (SOC) Program and Complementarity Problem
  - Unconstrained Diff. Min. Reformulation
  - Numerical Experience
- II. SOCP from Dist. Geometry Optim
  - Simulation Results
Convex SOCP

\[
\begin{align*}
\min & \quad g(x) \\
\text{s.t.} & \quad Ax = b \\
& \quad x \in K
\end{align*}
\]

\(A \in \mathbb{R}^{m \times n}, \ b \in \mathbb{R}^m\)

\(g : \mathbb{R}^n \to \mathbb{R}, \) convex, twice cont. diff.

\(K = K^{n_1} \times \cdots \times K^{n_p}\)

\(K^{n_i} \overset{\text{def}}{=} \left\{ x_i = \begin{bmatrix} x_{1i} \\ x_{2i} \end{bmatrix} \in \mathbb{R} \times \mathbb{R}^{n_i-1} : \|x_{2i}\|_2 \leq x_{1i} \right\}\)

**Special cases?** LP, SOCP,...
SMOOTHING METHODS FOR SOCP/SOCP

SOC $K^n$

$n=1$

$n=2$

$n=3$

$K^n$
Suff. Optim. Conditions

\[ x \in K, \quad y \in K, \quad x^T y = 0, \]
\[ Ax = b, \quad y = \nabla g(x) - A^T \zeta_d \]

\[ \iff \]

\[ x \in K, \quad y \in K, \quad x^T y = 0, \]
\[ x = F(\zeta), \quad y = G(\zeta) \]

with

\[ F(\zeta) = d + (I - A^T (AA^T)^{-1} A) \zeta \]
\[ G(\zeta) = \nabla g(F(\zeta)) - A^T (AA^T)^{-1} A \zeta \quad (Ad = b) \]
SOCCP

Find $\zeta \in \mathbb{R}^n$ satisfying

$$x \in K, \quad y \in K, \quad x^T y = 0,$$

$$x = F(\zeta), \quad y = G(\zeta)$$

$F, G : \mathbb{R}^n \to \mathbb{R}^n$ smooth

$\nabla F(\zeta), -\nabla G(\zeta)$ column-monotone $\forall \zeta \in \mathbb{R}^n$, i.e.,

$$\nabla F(\zeta) u - \nabla G(\zeta) v = 0 \Rightarrow u^T v \geq 0$$

Special cases? convex SOCP, monotone NCP,...
How to solve SOCCP?

For LP, simplex methods and interior-point methods.

For SOCP, interior-point methods.

For convex SOCP and column-monotone SOCCP?

Interior-point methods not amenable to warm start. Non-interior methods?
Nonsmooth Eq. Reformulation

\[ x_i \cdot y_i \overset{\text{def}}{=} \begin{bmatrix} x_{1i} \\ x_{2i} \end{bmatrix} \cdot \begin{bmatrix} y_{1i} \\ y_{2i} \end{bmatrix} = \begin{bmatrix} x_i^T y_i \\ x_{1i} y_{2i} + y_{1i} x_{2i} \end{bmatrix} \]  
(Jordan product assoc. with \( K^{n_i} \))

\[
\phi_{FB}(x, y) \overset{\text{def}}{=} \left[ (x_i^2 + y_i^2)^{1/2} - x_i - y_i \right]_{i=1}^p
\]

**Fact** (Fukushima, Luo, T’02):

\[
\phi_{FB}(x, y) = 0 \iff x \in K, y \in K, x^T y = 0
\]

Thus, SOCCP is equivalent to

\[
\phi_{FB}(F(\zeta), G(\zeta)) = 0
\]

\( \phi_{FB} \) is strongly semismooth (Sun, Sun ’03)
**Unconstr. Smooth Min. Reformulation**

\[
\min f_{FB}(\zeta) \overset{\text{def}}{=} \| \phi_{FB}(F(\zeta), G(\zeta)) \|^2_2
\]

\(F, G\) smooth and \(\nabla F(\zeta), -\nabla G(\zeta)\) column-monotone \(\forall \zeta \in \mathbb{R}^n\) (e.g., LP, SOCP, convex SOCP, monotone NCP)

For monotone NCP \((K = \mathbb{R}_+^n)\),

\[f_{FB}\text{ is smooth, and } \nabla f_{FB}(\zeta) = 0 \iff \zeta \text{ is a soln}\]

(Geiger, Kanzow '96)

The same holds for SOCCP. (J.-S. Chen, T '04)

**Advantage?** Any method for unconstrained diff. min. (e.g., CG, BFGS, L-BFGS) can be used to find \(\nabla f_{FB}(\zeta) = 0\).
Numerical Experience on Convex SOCP

\[ x = F(\zeta) = d + (I - P)\zeta \]
\[ y = G(\zeta) = \nabla g(F(\zeta)) - P\zeta \]

with \( P = A^T(AA^T)^{-1}A, \ Ad = b. \)

(Solve \( \min \|Ax - b\| \) to find \( d \))

- Implement in Matlab CG-PR, BFGS, L-BFGS (memory=5) to minimize \( f_{FB}(\zeta) \), using Armijo stepsize rule, with \( \zeta^{\text{init}} = 0 \). Stop when

\[ \max\{f_{FB}(\zeta), |x^Ty|\} \leq \text{accur}. \]

- Let \( \psi_{FB}(x, y) \overset{\text{def}}{=} \|\phi_{FB}(x, y)\|^2_2 \). Then

\[ f_{FB}(\zeta) = \psi_{FB}(x, y) \]
\[ \nabla f_{FB}(\zeta) = (I - P)\nabla_x \psi_{FB}(x, y) - P\nabla_y \psi_{FB}(x, y) \]

Compute \( P\zeta \) using Cholesky factorization of \( AA^T \) or using preconditioned CG. Compute \( \psi_{FB}(x, y) \) and \( \nabla \psi_{FB}(x, y) \) within Fortran Mex files.
**DIMACS Challenge SOCPs**

- Problem names and statistics:
  - \( nb \) \( (m = 123, n = 2383, K = (K^3)^{793} \times \mathbb{R}_+^4) \)
  - \( nb-L2 \) \( (m = 123, n = 4195, K = K^{1677} \times (K^3)^{838} \times \mathbb{R}_+^4) \)
  - \( nb-L2\)-bessel \( (m = 123, n = 2641, K = K^{123} \times (K^3)^{838} \times \mathbb{R}_+^4) \)

Compare iters/cpu(sec)/accuracy with Sedumi 1.05 (Sturm '01), which implements a predictor-corrector interior-point method.

<table>
<thead>
<tr>
<th>Problem Name</th>
<th>SeDuMi (pars.eps=1e-5) iter/cpu</th>
<th>L-BFGS-Chol (accur=1e-5) iter/cpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>nb</td>
<td>19/7.6</td>
<td>1042/16.5</td>
</tr>
<tr>
<td>nb-L2</td>
<td>11/11.1</td>
<td>330/9.2</td>
</tr>
<tr>
<td>nb-L2-bessel</td>
<td>11/5.3</td>
<td>108/1.7</td>
</tr>
</tbody>
</table>

*Table 1: (cpu times are in sec on an HP DL360 workstation, running Matlab 6.1)*
Smooth methods for SOCP/SOCCP

Regularized Sum-of-Norms Problems

\[
\min_{w \geq 0} \sum_{i=1}^{M} \|A_iw - b_i\|_2 + h(w),
\]

\[A_i \sim U[-1, 1]^{m_i \times \ell}, \quad b_i \sim U[-5, 5]^{m_i}, \quad m_i \sim U\{2, 3, ..., r\} \quad (r \geq 2).\]

\[h(w) = 1^T w + \frac{1}{3}\|w\|_3^3 \quad \text{(cubic reg.)}\]

Reformulate as a convex SOCP:

\[
\begin{aligned}
& \text{minimize} & & \sum_{i=1}^{M} z_i + h(w) \\
& \text{subject to} & & A_iw + s_i = b_i, \quad (z_i, s_i) \in K^{m_i+1}, \quad i=1,...,M, \quad w \in \mathbb{R}_+^{\ell}.
\end{aligned}
\]

<table>
<thead>
<tr>
<th>Problem</th>
<th>BFGS-Chol iter/cpu</th>
<th>CG-PR-Chol iter/cpu</th>
<th>L-BFGS-Chol iter/cpu</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\ell, M, r \quad (m, n))</td>
<td>(500,10,10 \quad (56,566))</td>
<td>(500,50,10 \quad (283,833))</td>
<td>(500,10,50 \quad (246,756))</td>
</tr>
<tr>
<td></td>
<td>352/24.6</td>
<td>1703/6.6</td>
<td>497/2.4</td>
</tr>
<tr>
<td></td>
<td>546/85.1</td>
<td>3173/69.0</td>
<td>700/12.4</td>
</tr>
<tr>
<td></td>
<td>272/36.3</td>
<td>1290/23.0</td>
<td>371/5.6</td>
</tr>
</tbody>
</table>

Table 2: (cpu times are in sec on an HP DL360 workstation, running Matlab 6.5.1, with accur=1e-3)
**Smoothing Newton Step**

\[
\phi_{\text{FB}}^\mu(x, y) \overset{\text{def}}{=} (x^2 + y^2 + \mu^2 e)^{1/2} - x - y
\]

with \( e = (1, 0, ..., 0, ..., 1, 0, ..., 0)^T \), \( \mu > 0 \)  

(Fukushima, Luo, T ’02)

Given \( \zeta \), choose \( \mu > 0 \) and solve

\[
\nabla \phi_{\text{FB}}^\mu(F(\zeta), G(\zeta))^T \Delta \zeta = -\phi_{\text{FB}}^\mu(F(\zeta), G(\zeta))
\]

Use \( \Delta \zeta \) to accelerate convergence.

This requires more work per iteration. Use it judiciously.
Observations

For our unconstrained smooth merit function approach:

**Advantage:**

- Less work/iteration, simpler matrix computation than interior-point methods.
- Applicable to convex SOCP and column-monotone SOCCP.
- Useful for warm start?

**Drawback:**

- Many more iters. than interior-point methods.
- Lower solution accuracy.
\textbf{SOCP from Dist. Geometry Optim} \textit{(ongoing work..)}

$n$ pts in $\mathbb{R}^d$ ($d = 2, 3$).

Know $x_{m+1}, \ldots, x_n$ and Eucl. dist. estimate for pairs of ‘neighboring’ pts

$$d_{ij} > 0 \quad \forall (i, j) \in A \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}.$$ 

Estimate $x_1, \ldots, x_m$.

\textbf{Problem (nonconvex):}

$$\min_{x_1, \ldots, x_m} \sum_{(i, j) \in A} \left| \|x_i - x_j\|^2_2 - d_{ij}^2 \right|$$
Convex relaxation:

\[
\min_{x_1, \ldots, x_m} \sum_{(i,j) \in A} \max\{0, \|x_i - x_j\|_2^2 - d_{ij}^2\}
\]

This is an unconstrained (nonsmooth) convex program, can be reformulated as an SOCP. Alternatives?

Smooth approx.:\[
\max\{0, t\} \approx \mu h\left(\frac{t}{\mu}\right) \quad (\mu > 0)
\]
h smooth convex, \(\lim_{t \to -\infty} h(t) = \lim_{t \to \infty} h(t) - t = 0\).

We use \(h(t) = ((t^2 + 4)^{1/2} + t)/2\) (CHKS).
Smooth Approximation of Convex Relaxation

\[
\min_{x_1, \ldots, x_m} \ f_\mu(x_1, \ldots, x_m) \overset{\text{def}}{=} \sum_{(i,j) \in A} \mu h \left( \frac{\|x_i - x_j\|^2 - d_{ij}^2}{\mu} \right)
\]

Solve the smooth approximation using Inexact Block Coordinate Descent:

- If \( \|\nabla x_i f_\mu\| = \Omega(\mu) \), then update \( x_i \) by moving it along the Newton direction 
  \[-[\nabla^2_{x_i x_i} f_\mu]^{-1} \nabla x_i f_\mu \text{, with Armijo stepsize rule, and re-iterate.} \]

- Decrease \( \mu \) when \( \|\nabla x_i f_\mu\| = O(\mu) \ \forall i \).

\( \mu^{\text{init}} = 1e^{-3} \). \( \mu^{\text{end}} = 2e^{-6} \). Decrease \( \mu \) by a factor of 5. Code in Matlab.
Simulation Results

Uniformly generate $\tilde{x}_1, \ldots, \tilde{x}_n$ in $[-.5, .5]^2$, $m = 0.9n$

two pts are nhbrs if dist $< .06$.

Set $d_{ij} = \|\tilde{x}_i - \tilde{x}_j\|$ (Biswas, Ye ‘03)

<table>
<thead>
<tr>
<th>$n$</th>
<th>SOCP dim</th>
<th>SeDuMi cpu/Err</th>
<th>Inexact BCD cpu/Err</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>21472 × 33908</td>
<td>330/.48</td>
<td>373/.48</td>
</tr>
<tr>
<td>2000</td>
<td>84440 × 130060</td>
<td>12548/.57</td>
<td>2090/.52</td>
</tr>
</tbody>
</table>

Table 3: (cpu times are in secs on a Linux PC cluster, running Matlab 6.1.)

$\text{Err} = \sum_{i=1}^{m} \|x_i - \tilde{x}_i\|_2^2$.
True soln \((m = 900, n = 1000)\)

SOCP soln found by SeDuMi  
SOCP soln found by Inexact BCD
Observations

For our smoothing-Inexact BCD approach:

- Better cpu time than using SeDuMi. Add barrier term to find analytic center soln.

- Computation easily distributes.

- Code in Fortran (instead of Matlab) to improve time?
Lastly...

Thanks, Christian, for lending the use of your laptop!