Smoothing methods for Second-Order Cone Programs/Complementarity Problems

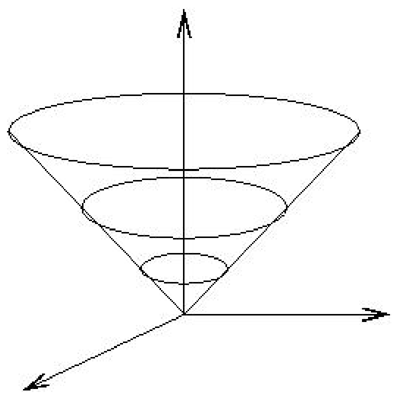
Paul Tseng
University of Washington, Seattle

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

$$\begin{array}{ll} \min & g(x) \\ \text{s.t.} & Ax = b \\ & x \in K \end{array}$$

$A \in \Re^{m \times n}$, $b \in \Re^m$

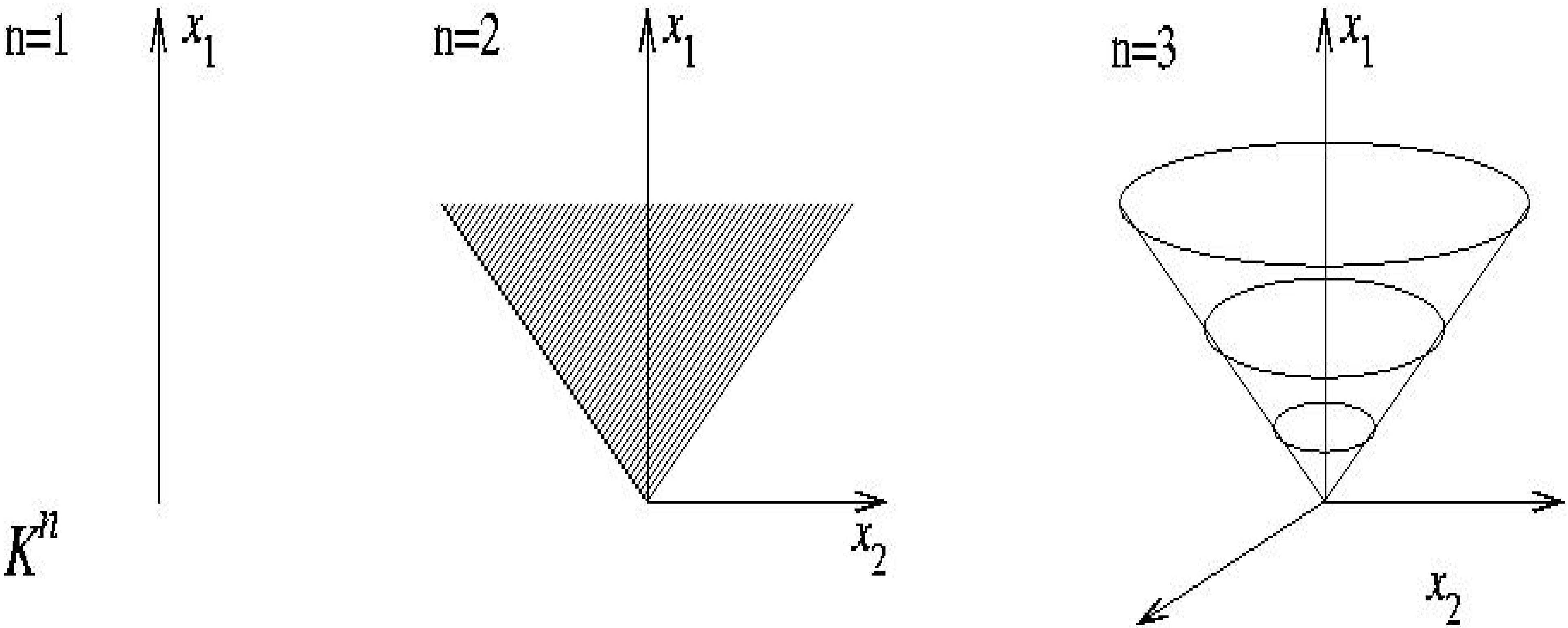
$g : \Re^n \longrightarrow \Re$, convex, twice cont. diff.

$K = K^{n_1} \times \cdots \times K^{n_p}$

$K^{n_i} \stackrel{\text{def}}{=} \left\{ x_i = \begin{bmatrix} x_{1i} \\ x_{2i} \end{bmatrix} \in \Re \times \Re^{n_i - 1} : \|x_{2i}\|_2 \le x_{1i} \right\}$

Special cases? LP, SOCP,...

SOC K^n



Suff. Optim. Conditions

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

$$\Longleftrightarrow$$

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

with

$$\begin{aligned} F(\zeta) &= \quad d + (I - A^T(AA^T)^{-1}A)\zeta \\ G(\zeta) &= \nabla g(F(\zeta)) - A^T(AA^T)^{-1}A\zeta \qquad (Ad = b) \end{aligned}$$

SOCCP

Find $\zeta \in \Re^n$ satisfying

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

$F, G : \Re^n \to \Re^n$ smooth

$\nabla F(\zeta), -\nabla G(\zeta)$ column-monotone $\forall \zeta \in \Re^n$, 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

$$x_i \cdot y_i \stackrel{\text{def}}{=} \begin{bmatrix} x_{1i} \\ x_{2i} \end{bmatrix} \cdot \begin{bmatrix} y_{1i} \\ y_{2i} \end{bmatrix} = \begin{bmatrix} x_i^T y_i \\ x_{1i} y_{2i} + y_{1i} x_{2i} \end{bmatrix}$$

(Jordan product assoc. with K^{n_i})

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

Fact (Fukushima,Luo,T '02):

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

Thus, SOCCP is equivalent to

$$\boxed{\phi_{\text{FB}}(F(\zeta), G(\zeta)) = 0}$$

ϕ_{FB} is strongly semismooth (Sun,Sun '03)

Unconstr. Smooth Min. Reformulation

$$\min \; f_{\text{FB}}(\zeta) \stackrel{\text{def}}{=} \|\phi_{\text{FB}}(F(\zeta), G(\zeta))\|_2^2$$

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

For monotone NCP ($K = \Re^n_+$),

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

(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 $\nabla f_{\text{FB}}(\zeta) = 0$.

Numerical Experience on Convex SOCP

$$\begin{aligned} x = \quad F(\zeta) = & \quad d + (I - P)\zeta \\ y = \quad G(\zeta) = & \quad \nabla g(F(\zeta)) - P\zeta \end{aligned}$$

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 $f_{\rm FB}(\zeta)$, using Armijo stepsize rule, with $\zeta^{\rm init} = 0$. Stop when

$$\max\{f_{\rm FB}(\zeta), |x^T y|\} \le \texttt{accur}.$$

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

$$\begin{aligned} f_{\rm FB}(\zeta) &= \psi_{\rm FB}(x,y) \\ \nabla f_{\rm FB}(\zeta) &= (I-P)\nabla_x \psi_{\rm FB}(x,y) - P\nabla_y \psi_{\rm FB}(x,y) \end{aligned}$$

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

DIMACS Challenge SOCPs

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

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

Problem Name	SeDuMi (pars.eps=1e-5) iter/cpu	L-BFGS-Chol (accur=1e-5) iter/cpu
nb	19/7.6	1042/16.5
`nb-L2`	11/11.1	330/9.2
`nb-L2-bessel`	11/5.3	108/1.7

Table 1: (cpu times are in sec on an HP DL360 workstation, running Matlab 6.1)

Regularized Sum-of-Norms Problems

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

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

$h(w) = 1^T w + \frac{1}{3}\|w\|_3^3$ (cubic reg.)

Reformulate as a convex SOCP:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{M} z_i + h(w) \\ \text{subject to} & A_i w + s_i = b_i, \quad (z_i, s_i) \in K^{m_i+1}, \; _{i=1,\ldots,M}, \quad w \in \Re_+^\ell. \end{array}$$

Problem ℓ, M, r (m, n)	BFGS-Chol iter/cpu	CG-PR-Chol iter/cpu	L-BFGS-Chol iter/cpu
500,10,10 (56,566)	352/24.6	1703/6.6	497/2.4
500,50,10 (283,833)	546/85.1	3173/69.0	700/12.4
500,10,50 (246,756)	272/36.3	1290/23.0	371/5.6

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

Smoothing Newton Step

$$\phi^{\mu}_{\mathrm{FB}}(x,y) \stackrel{\mathrm{def}}{=} (x^2+y^2+\mu^2 e)^{1/2} - x - y$$

with $e = (\underbrace{1,0,..,0}_{n_1}, ..., \underbrace{1,0,..,0}_{n_p})^T$, $\quad \mu > 0$ (Fukushima,Luo,T '02)

Given ζ, choose $\mu > 0$ and solve

$$\nabla\phi^{\mu}_{\mathrm{FB}}(F(\zeta),G(\zeta))^T \Delta\zeta = -\phi_{\mathrm{FB}}(F(\zeta),G(\zeta))$$

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.

SOCP from Dist. Geometry Optim (ongoing work..)

n pts in $\Re^d$ ($d = 2, 3$).

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

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

Estimate $x_1, ..., x_m$.

Problem (nonconvex):

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

Convex relaxation:

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

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

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

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 $-[\nabla^2_{x_i x_i} f_\mu]^{-1}\nabla_{x_i} f_\mu$, with Armijo stepsize rule, and re-iterate.
- Decrease μ when $\|\nabla_{x_i} f_\mu\| = O(\mu)\ \forall i$.

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

Simulation Results

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

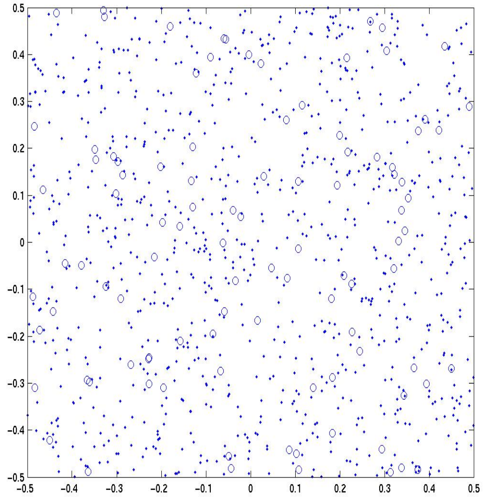
two pts are nhbrs if dist $< .06$.

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

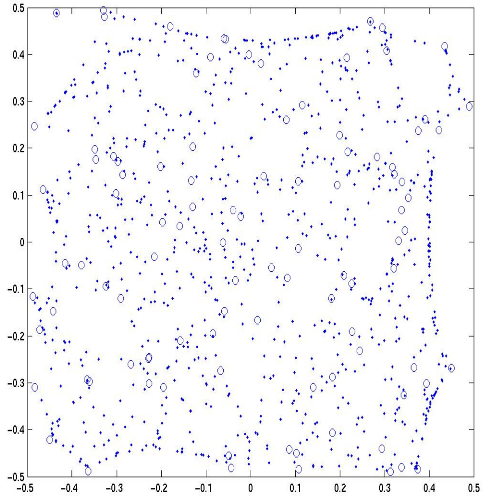
n	SOCP dim	SeDuMi cpu/Err	Inexact BCD cpu/Err
1000	21472×33908	330/.48	373/.48
2000	84440×130060	12548/.57	2090/.52

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

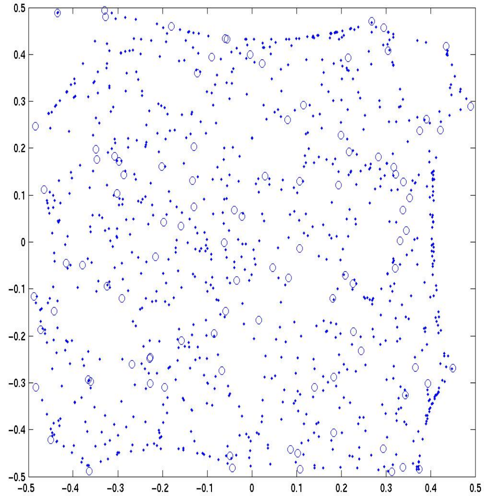
Err $= \sum_{i=1}^{m} \|x_i - \tilde{x}_i\|_2^2$.



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!