KNITRO NLP Solver

Todd Plantenga,
Ziena Optimization, Inc.

US-Mexico Workshop on Optimization
January 2007, Huatulco, Mexico

What is KNITRO?

- Commercial software for general NLP
  - Fully embeddable software
  - AMPL, AIMMS, Excel, GAMS, MATLAB, Mathematica
  - Written in C/C++
  - Callable from C, C++, Fortran, Java
  - Windows, Linux (32-bit, 64-bit), MacOSX, Solaris
- Ziena Optimization, Inc. started in 2001
  - *http://www.ziena.com*
- KNITRO Team:
  - Richard Waltz, Todd Plantenga, Jorge Nocedal, Robert Fourer, Richard Byrd

KNITRO Design Approach

- Multiple, complementary algorithms needed
  - Robustness
  - Efficiency (CPU time, function evaluations)
  - Practicality (derivative availability, warm start)

- Simultaneous development of **Interior Point** and **Active Set** methods
- Integrated algorithms (robustness, crossover)
- Special adaptations for LP, QP, MPCC, etc.
- Extendable framework (multi-start)

3 Different Algorithms

- Interior Point / Barrier
  - **Interior / Direct** (line search)
    (2003 - Waltz, Morales, Nocedal, Orban)
  - **Interior / CG** (trust region)
    (1999 - Byrd,Hribar,Nocedal)
- Active Set SLQP
  - **Active Set** (trust region)
    (2004 - Byrd, Gould, Nocedal, Waltz)

Interior Point Approaches

$$\begin{array}{ll} \min\limits_{x \in \mathbf{R}^n} & f(x) \\ \text{subject to} & g_i(x) \geq 0 \quad i \in I \\ & h_j(x) = 0 \quad j \in E \end{array}$$

- Solve barrier subproblems:

$$\begin{array}{ll} \min\limits_{x,s} & f(x) - \mu \sum\limits_{i \in I} \ln s_i \\ \text{subject to} & g_i(x) - s_i = 0 \quad i \in I \\ & h_j(x) = 0 \quad j \in E \\ & x \in \mathbf{R}^n \end{array}$$

Force $\mu \to 0, \; s_i > 0$

KNITRO

Interior Point / Direct Algorithm

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & 0 & A_g & A_h \\ 0 & \Sigma & -I & 0 \\ A_g^T & -I & 0 & 0 \\ A_h^T & 0 & 0 & \delta_h I \end{bmatrix} \begin{bmatrix} d_x \\ d_s \\ -\Delta\lambda_g \\ -\Delta\lambda_h \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}_k \\ \nabla_s \mathcal{L}_k \\ g_k - s_k \\ h_k \end{bmatrix}$$

with $\Sigma = S^{-1}\Lambda_g$

using δ_h for regularization

- Line search using l_2-norm merit function
- Fall back to trust region Interior Point / **CG**
  - Negative curvature in null space of KKT matrix
  - Exceed maximum number of backtracks
  - Step length becomes too short
  - 49% of CUTE problems

Interior Point / CG Algorithm

- Step ***d*=*u*+*v***, with ***v*** in range space of constraint Jacobian

$$\text{Let} \quad u = (u_x, u_s)$$

$$\text{Let} \quad W = \begin{bmatrix} \nabla^2_{xx}\mathcal{L} & 0 \\ 0 & \Sigma \end{bmatrix}, \quad \Sigma = S^{-1}\Lambda_g$$

$$\begin{aligned} \min_u \quad & \nabla f_k^T u_x - \mu_k S_k^{-1} u_s + u^T W_k v_k + \frac{1}{2} u^T W_k u \\ \text{subject to} \quad & \\ & \begin{bmatrix} A_g^T & -I \\ A_h^T & 0 \end{bmatrix} \begin{bmatrix} u_x \\ u_s \end{bmatrix} = 0 \\ & \|(u_x, S_k^{-1} u_s)\|_2^2 \leq \Delta^2 - \|v_k\|_2^2 \\ & u_s \geq -\tau S_k e - v_s \end{aligned}$$

- Restrict ***u*** to null space using an exact projection
- Find inexact minimizer using CG
- Slacks are scaled (preconditioned)

IP / CG Preconditioning

- Tangent step ***u*** needs preconditioning for:
  - Slacks (scale $\tilde{u}_s = S_k^{-1} u_s$)
  - Hessian $\nabla^2_{xx}\mathcal{L}$
  - Constraints:

$$\min_u \; t^T u + \frac{1}{2} u^T W u \quad \text{s.t.} \quad Au = b$$

CG equations: $r_0 = W u_0 + t$

$$\begin{array}{rclcrcl} u_{i+1} & = & u_i + \alpha p_i & , & r_{i+1} & = & r_i + \alpha W p_i \\ z_{i+1} & = & M r_{i+1} & , & p_{i+1} & = & -z_{i+1} + \beta p_i \end{array}$$

where constraint preconditioner M

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$$

guarantees u in null space of A

IP / CG Hessian Preconditioning

- Tangent step preconditioner:

$$\begin{bmatrix} P & 0 & A_g & A_h \\ 0 & D_s & -I & 0 \\ A_g^T & -I & 0 & 0 \\ A_h^T & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_x \\ z_s \\ y_g \\ y_h \end{bmatrix} = \begin{bmatrix} r_x \\ r_s \\ 0 \\ 0 \end{bmatrix} \qquad \begin{array}{ccc} P & \approx & \nabla^2_{xx}\mathcal{L} \\ D_s & \approx & \Sigma \end{array}$$

- Block elimination gives:

$$\begin{bmatrix} P + A_g D_s A_g^T & A_h \\ A_h^T & 0 \end{bmatrix} \begin{bmatrix} z_x \\ y_g \end{bmatrix} = \begin{bmatrix} r_x + A_g r_s \\ 0 \end{bmatrix}$$

If no equalities: $LL^T \approx \nabla^2_{xx}\mathcal{L} + A_g \Sigma A_g^T$

- Incomplete Cholesky results:
  - Works well when only bound constraints
  - Small improvement for general inequalities

Interior Point Characteristics

Strengths

- Efficiency, speed
- Relative insensitivity to degeneracy
- Flexible derivative options

Weaknesses

- NLP start point, initial point strategy
- Warm start
- Updating barrier parameter μ

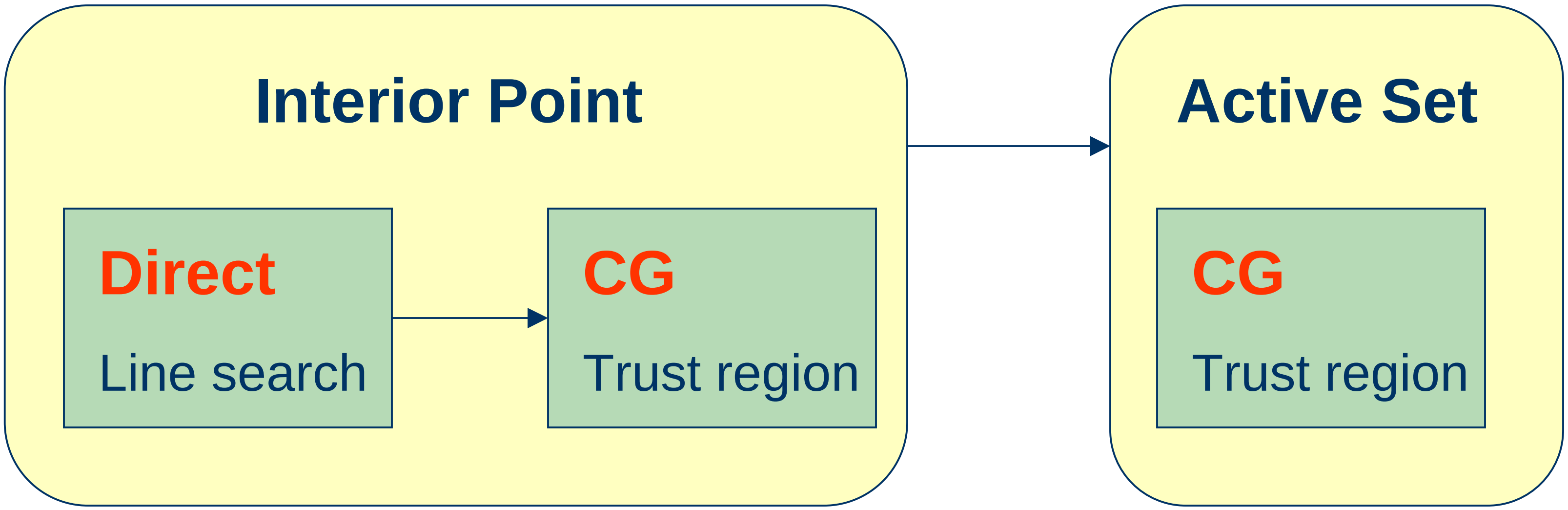
Active Set SLQP Algorithm

- Minimize l_1-norm penalty function (Fletcher, 1989)

$$f(x) + \nu \left(\|h(x)\|_1 + \sum_i \max\{0, -g(x)_i\} \right)$$

- Solve LP to identify active set
  - Box trust region $|d|_\infty \leq \Delta^{LP}$
  - Update Δ^{LP} with special rules
  - Update ν every iterate, heuristic can decrease based on $|\lambda|_\infty$
- Solve EQP using CG
  - Spherical trust region $|d|_2 \leq \Delta^{EQP}$
  - Iterate is $d^C + \alpha(d^{EQP} - d^C)$

KNITRO Algorithm Interactions



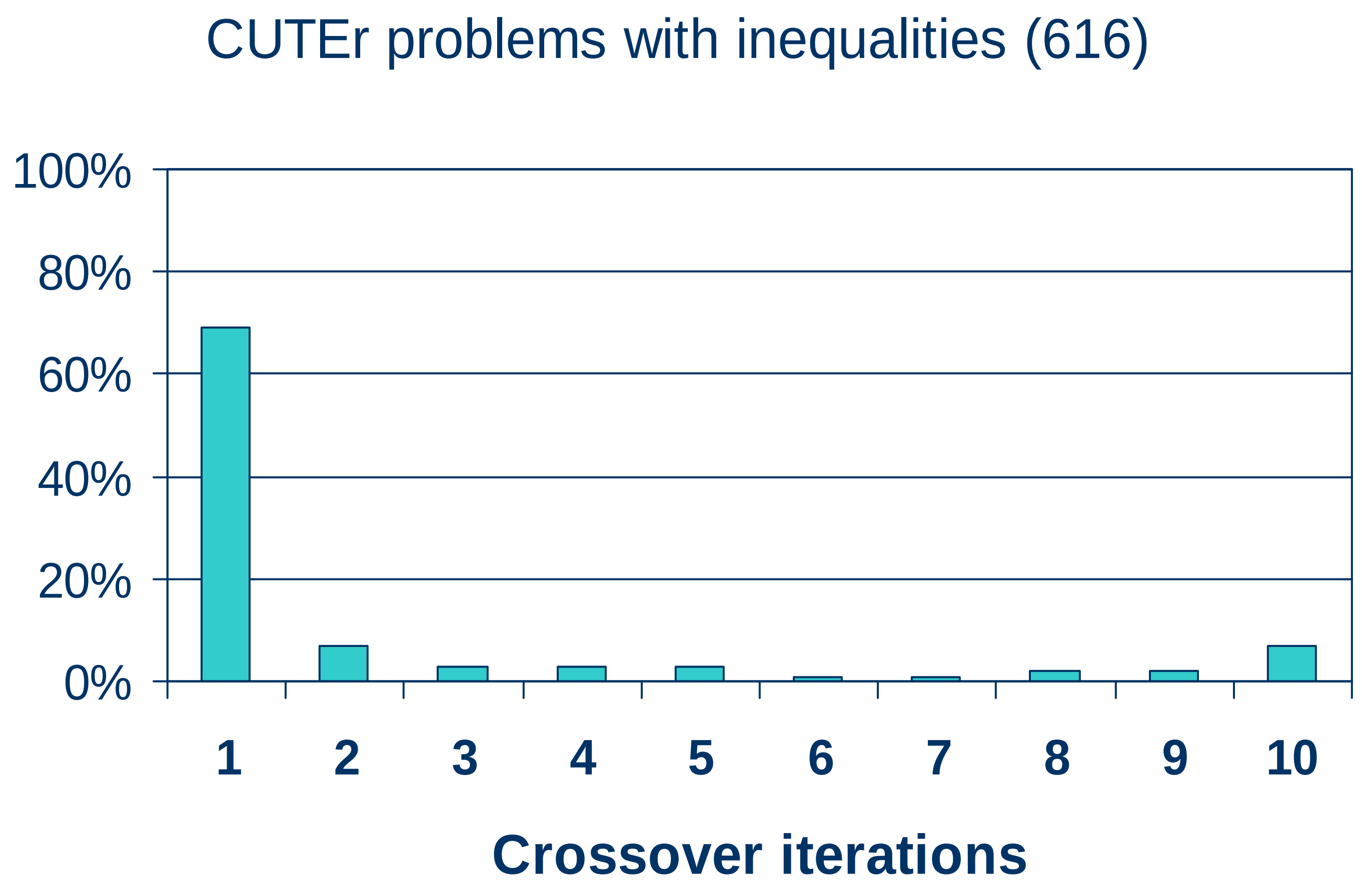
- IP/**Direct** reverts to IP/**CG** for robustness
- Interior Point can cross over to Active Set for:
  - More accurate solution point
  - More accurate sensitivities

Crossover to Active Set

- Improve the interior point solution
  - Identify active set of inequalities
  - More accurate sensitivities
  - Solution on constraints better in some applications
- First, guess active set and try EQP
- Else, carefully initialize LP trust region:

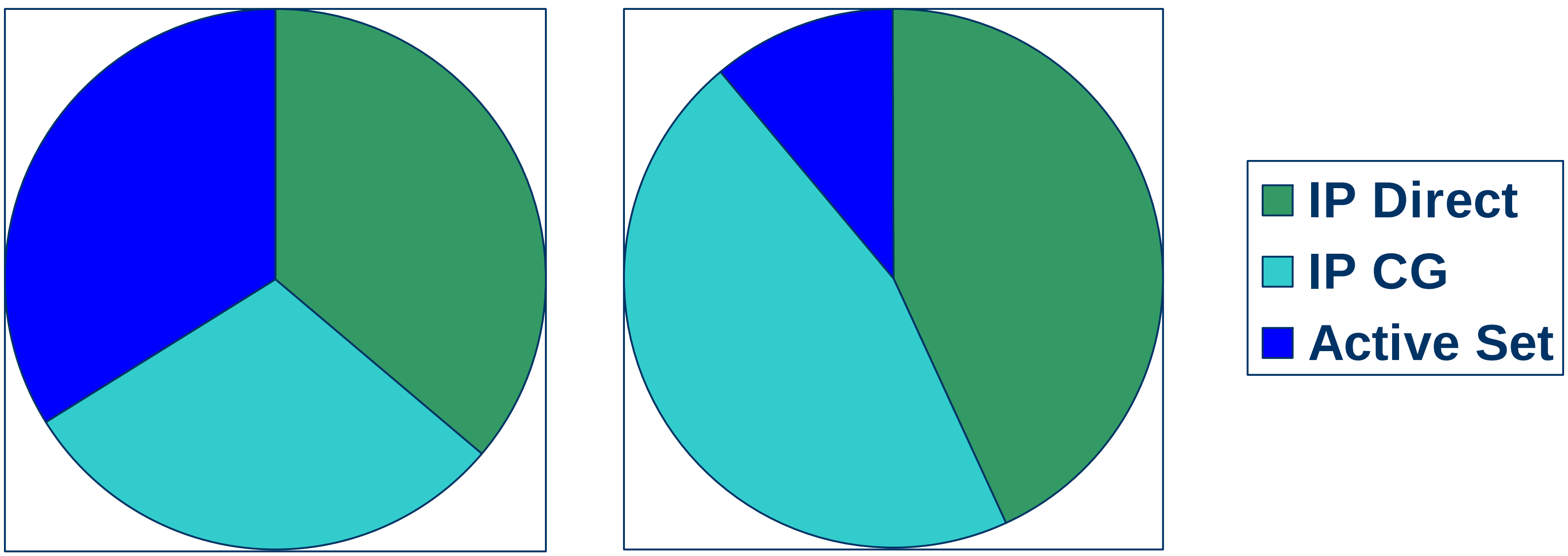
$$\Delta_0^{LP} = \min \left\{ \frac{g_i}{\|\nabla g_i\|}, \frac{h_j}{\|\nabla h_j\|} \;:\; i, j \;\; \text{inactive} \right\}$$

Crossover Success Rate



3 Solvers are more Efficient

- 920 test problems
- "Best" solver when local solution found:



Function Evaluations **CPU Time**

Test Problem Tuning

Beware of "fragile" test problems

- Example, change BLAS Level 1 library
  - 2814 cases (938 problems, 3 algorithms)
    - **78.5%** reach same objective in same num fn evaluations
    - **9.9%** reach same objective, but different num fn evals (average change is zero)
    - **4.4%** find different objective (by at least 1%)
    - **1.3%** fail to find a local solution
- Other examples
  - Windows vs. Linux
  - Windows debug vs. production (C runtime)

KNITRO

KNITRO Summary

- Strengths
  - 3 algorithms: efficiency, practicality
  - Integrated approach: robustness, crossover
- Improvements
  - Preconditioning (IP/CG and Active Set)
  - Active Set robustness
- Weaknesses
  - No automatic parameter tuning
  - Stops at infeasible stationary point
  - Regularization for Active Set, IP/CG
  - Need second-order KKT conditions