On the development of Interior Point software for Quadratic Programming

D. di Serafino, Second University of Naples, Italy
daniela.diserafino@unina2.it

joint work with

S. Cafieri, M. D’Apuzzo, V. De Simone, Second University of Naples
G. Toraldo, University of Naples “Federico II”

Work supported by MIUR, FIRB Project “Large-Scale Nonlinear Optimization” (n. RBNE01WBBB) and PRIN Project “Innovative Problems and Methods in Nonlinear Optimization” (n. 2005017083)
Outline

- Background and motivations

- Main issues in developing IP software
  - Our choices in PRQP – Potential Reduction software for Quadratic Programming
  - Focus on Linear Algebra

- Performance results and comparisons
**QP problem**

**Primal-dual formulation**

\[
\begin{align*}
\text{min } q(x) &= \frac{1}{2} x^T Q x + c^T x \\
\text{s.t. } A x - z &= b, \quad C x = d, \quad x + \nu = u \\
& \quad (x, z, \nu) \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{max } p(x) &= -\frac{1}{2} x^T Q x + b^T y + d^T \lambda - u^T t \\
\text{s.t. } Q x - A^T y - C^T \lambda - s + t &= -c, \\
& \quad (x, y, s, t) \geq 0
\end{align*}
\]

\[Q \in \mathbb{R}^{n \times n} \text{ SPSD, } A \in \mathbb{R}^{m \times n}, C \in \mathbb{R}^{p \times n}, m + p \leq n, \text{ large and sparse}\]

\[x, y, \lambda, t \text{ primal and dual variables, } s, z, \nu \text{ slack variables}\]
Currently available software for QP

<table>
<thead>
<tr>
<th>Software</th>
<th>LA Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPMPD</td>
<td>Direct</td>
</tr>
<tr>
<td>BQPD</td>
<td>Direct</td>
</tr>
<tr>
<td>CPLEX-barrier</td>
<td>Direct</td>
</tr>
<tr>
<td>HOPDM</td>
<td>Direct / Iterative</td>
</tr>
<tr>
<td>KNITRO</td>
<td>Direct / Iterative</td>
</tr>
<tr>
<td>LANCELOT</td>
<td>Direct / Iterative</td>
</tr>
<tr>
<td>LOQO</td>
<td>Direct</td>
</tr>
<tr>
<td>MOSEK</td>
<td>Direct</td>
</tr>
<tr>
<td>OOQP</td>
<td>Direct</td>
</tr>
<tr>
<td>PCx</td>
<td>Direct</td>
</tr>
<tr>
<td>QPA</td>
<td>Iterative</td>
</tr>
<tr>
<td>QPB</td>
<td>Iterative</td>
</tr>
<tr>
<td>SQOPT</td>
<td>Direct</td>
</tr>
<tr>
<td>Xpress-MP</td>
<td>Direct</td>
</tr>
</tbody>
</table>

... = Interior Point methods

GOAL
develop a software package for large-scale QP problems based on efficient Linear Algebra kernels
Developing IP software: main issues

- Choice of the method
  - Potential Reduction
- Solution of the Newton system
  - System reduction
  - Iterative solvers & preconditioners
  - Adaptive stopping criteria (inner iterations)
- Computation of the step length
- Starting point
- Degeneracy
- Non convexity
- Presolve
Choice of the method

- Interior Point
  - “Easy” implementation (w.r.t. Active Set methods)
  - Effectiveness on large-scale problems
  - Can be extended to general nonlinear optimization problems

- Potential Reduction (instead of more usual path-following)
  - Close theory and practice
  - Single function to measure the quality of a point and to decide how to improve it
  - Perturbation parameters automatically updated
  - Linear Algebra kernels shared with other IP methods

- Primal-dual

- Infeasible; feasible whenever possible
**Infeasible primal-dual PR framework**

(Kojima, Mizuno & Todd, 1995)

\[
\min \Phi(x, y, s, z, \nu, t) = \rho \ln(x^T s + y^T z + \nu^T t) - \sum_{i=1}^{n} \ln(x_i s_i) - \sum_{j=1}^{m} \ln(y_j z_j) - \sum_{i=1}^{n} \ln(\nu_i t_i)
\]

Potential function

(Tanabe, 1988; Todd & Ye, 1990)

\[
\Delta \geq \frac{\sigma}{\Delta^0}
\]

\[
\Delta^0, \sigma^0 \text{ corresponding to the initial point } w^0
\]

\[
\sigma = \left\| (r_{p_1}, r_{p_2}, r_{p_3}, r_d) \right\|_2
\]

\[
r_{p_1} = Ax - z - b, \quad r_{p_2} = x + \nu - u, \quad r_{p_3} = Cx - d
\]

\[
r_d = Qx - A^T y - C^T \lambda - s + t + c
\]

\[
\sigma = 0 \quad \text{feasible version}
\]
PR basic steps

1. Given the current interior iterate $w=(x,y,s,z,\lambda,v,t)$, apply a Newton step to the perturbed KKT conditions

$$J(w)\delta w = g(\Delta/\rho)$$

$J(w)$ = Jacobian of the KKT equations

$\Delta/\rho$ = perturbation parameter

2. Update $w$

$$\bar{w} = w + \bar{\vartheta} \delta w$$

with $\bar{\vartheta}$ suitably chosen
### KKT Linear System

The KKT linear system is given by:

\[
\begin{bmatrix}
Q & -A & -I & 0 & -C^T & 0 & I \\
A & 0 & 0 & -I & 0 & 0 & 0 \\
I & 0 & 0 & 0 & 0 & I & 0 \\
C & 0 & 0 & 0 & 0 & 0 & 0 \\
S & 0 & X & 0 & 0 & 0 & 0 \\
0 & Z & 0 & Y & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & T V
\end{bmatrix}
\begin{bmatrix}
\delta x \\
\delta y \\
\delta s \\
\delta z \\
\delta \lambda \\
\delta v \\
\delta \tau
\end{bmatrix}
= 
\begin{bmatrix}
-r_d \\
-r_{p_1} \\
-r_{p_2} \\
-r_{p_3} \\
-XSe_n + (\Delta/\rho)e_n \\
-YZe_m + (\Delta/\rho)e_m \\
-TVe_n + (\Delta/\rho)e_n
\end{bmatrix}
\]

where

\[
L = \text{diag}(l), \quad e_q = (1,1,...,1) \in \mathbb{R}^q
\]

This requires a large computational effort and is one of the critical issues in an effective implementation of IP methods.
KKT system reduction: augmented system

\[
\begin{bmatrix}
Q + E & -G^T \\
-\delta & -F
\end{bmatrix}
\begin{bmatrix}
\delta x_1 \\
\delta x_2
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
b_2
\end{bmatrix}
\]

symmetric indefinite

bound constraints
\[E = X^{-1}S, \; F = T^{-1}V, \; G = I,\]
\[\delta x_1 = \delta x, \; \delta x_2 = \delta t\]

inequality + bound constr.
\[E = X^{-1}S + V^{-1}T, \; F = Y^{-1}Z, \; G = A,\]
\[\delta x_1 = \delta x, \; \delta x_2 = \delta y\]

equality + bound constr.
\[E = X^{-1}S + V^{-1}T, \; F = 0, \; G = C,\]
\[\delta x_1 = \delta x, \; \delta x_2 = \delta y\]

ineq. + eq. + bound constr.
\[E = X^{-1}S + V^{-1}T, \; F = \begin{bmatrix} Y^{-1}Z & 0 \\ 0 & 0 \end{bmatrix}, \; G = \begin{bmatrix} A \\ C \end{bmatrix},\]
\[\delta x_1 = \delta x, \; \delta x_2 = \begin{bmatrix} \delta y \\ \delta \lambda \end{bmatrix}\]
KKT system reduction: normal equations

Primal ordering

\[(G(Q + E)^{-1}G^T + F)\delta x_2 = -b_2 - G(Q + E)^{-1}b_1\]

Dual ordering \((F\) nonsingular)\n
\[(Q + E + G^T F^{-1}G)\delta x_1 = b_1 - G^T F^{-1}b_2\]

symmetric positive definite
Augmented system vs normal equations

Augmented system

- Dimension \( n+m+p \)
- Symmetric indefinite (\( F \) nonsingular \( \rightarrow \) quasi-definite [Vanderbei, 1995])
- Sparse if \( Q \) and \( G \) are sparse
- Only the diagonal entries must be updated at each outer iteration

Normal equations

- Dimension \( n \) or \( m+p \)
- Symmetric positive definite
- May be dense even if \( Q \) and \( G \) are sparse (i.e. \( G \) has a dense column/row)
- All the entries must be updated at each outer iteration
Ill conditioning

\[
\begin{bmatrix}
Q + E & -G^T \\
-G & -F
\end{bmatrix}
\]

\[G(Q + E)^{-1}G^T + F = Q + E + G^T F^{-1}G\]

\[E = X^{-1}S + V^{-1}T\]

\[F = Y^{-1}Z \quad \text{or} \quad F = \begin{bmatrix} Y^{-1}Z & 0 \\ 0 & 0 \end{bmatrix}\]

approaching the solution, some entries of \(E\) and \(F\) may become very large, producing an increasing ill conditioning in the matrix
Linear Algebra solvers: direct vs iterative

Direct solvers

- Widely used in well-established IP software (Mosek, PCx, Loqo, OOQP, ...)
- Ill conditioning not a severe problem (S. Wright, 1997; M. Wright, 1998)
- Computational cost may become prohibitive for large-scale problems

Iterative solvers

- Increasing attention by IP community in the last years (implemented, e.g., in KNITRO, HOPDM)
- Require suitable preconditioning techniques
- Adaptive stopping criteria, according to the quality of the IP iterate
Solution of normal equations (bound constr.)

Direct solution
- Sparse Cholesky factorization
- Reordering

Iterative solution
- Conjugate Gradient method
- Limited-memory Incomplete Cholesky Factorization (ICF) preconditioner (Lin & Moré, 1999)
- Diagonal scaling (Pardalos et al., 1991) (+ reordering)
Limited-memory ICF preconditioner

$q = \text{fill-in parameter},$ specifies the amount of additional memory available for factorization: only the $nj+q$ largest entries are retained

$q = \text{fill-in parameter},$

Adaptive choice of the fill-in parameter

outer steps 1 and 2: $q = 2$

outer step $k+1:$

$$\text{if ( } \text{iter}_{CG}(k) > 1.2 \times \text{iter}_{CG}(k-1) \text{ and } \text{iter}_{CG}(k) > \text{it}_{min} \text{ )}$$

$$q = \min(q + 3, q_{max})$$

$q_{max} = 30, \quad \text{it}_{min} = n/2000$
Solution of augmented system (lin. + bound constr.)

Direct solution

- Bunch-Parlett $LDL^T$; $LDL^T$ + heuristics for pivot selection (Vanderbei, 1995)
- Reordering

Iterative solution

- Krylov methods: simplified QMR, CG (although the matrix is indefinite)
- Constraint Preconditioner
Constraint Preconditioner (CP)

\[
\begin{bmatrix}
Q + E & -G^T \\
-G & -F
\end{bmatrix}
\begin{bmatrix}
\delta x_1 \\
\delta x_2
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
b_2
\end{bmatrix}
\]

\[
P = \begin{bmatrix}
diag(Q + E) & -G^T \\
-G & -F
\end{bmatrix}
\]

- Indefinite preconditioner with the same structure as the augmented system matrix

- CP variants investigated by many researchers
  (Axelsson, 1979; Golub & Wathen, 1998; Luksan & Vlcek, 1998; Keller et al., 2000; Rozloznik & Simoncini, 2002; Durazzi & Ruggiero, 2003; Gondzio et al., 2004; Dollar et al., 2004-2006; di Serafino et al., 2004; Forsgren et al., 2005; ...)
CP: spectral properties

- $P^{-1}M$ has at least $m$ unit eigenvalues; the remaining ones are real positive

- If $\text{rank}(F) = j$, then $P^{-1}M$ has at least $2m-j$ unit eigenvalues

  (Keller at al., 2000; Durazzi & Ruggiero, 2003; Bergamaschi et al., 2004; Dollar 2005)

  ➔ When the iterate approaches the solution, if $q$ entries of $F$ get close to zero, then additional $q$ eigenvalues tend to be clustered around 1

  ➔ We expect that the preconditioner increases its effectiveness as the PR method progresses
Augmented system: inequality constraints

- Null starting guess
  - CP + Simplified QMR

- Suitable starting guess, i.e.
  - \( \delta x_1^{(0)} = 0, \quad \delta x_2^{(0)} = F^{-1}b_2 \)

CP+CG applied to Augmented System behaves as CG applied to Normal Equations (dual ordering) with preconditioner \( \tilde{P} = diag(Q + E) + G^T F^{-1} G \)

- No breakdown
- Convergence in at most \( n \) iterations

\[
\|e^i\|_M \leq 2 \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^i \|e^0\|_M, \quad \kappa = \lambda_{\text{max}} / \lambda_{\text{min}}, \quad \lambda_{\text{max}} (\lambda_{\text{min}}) \text{ max (min) eig. of } \tilde{P}^{-1} (Q + E + A^T F^{-1} A)
\]

(Cafieri, D’Apuzzo, De Simone, di Serafino, 2004)

CP allows to use CG although the system is indefinite
Augmented system: equal. (+ inequal. ) constr.

\[
\begin{bmatrix}
Q + X^{-1}S + V^{-1}T & -A^T & -C^T \\
-A & Y^{-1}Z & 0 \\
-C & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\delta x \\
\delta y \\
\delta \lambda
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix}
\]

- Suitable starting guess, i.e.

\[
C\delta x^{(0)} = b_3, \quad \delta y^{(0)} = Z^{-1}Y(A\delta x^{(0)} - b_2), \quad \delta \lambda^{(0)} = 0
\]

least squares problem

- Null starting guess

\[
\delta x^{(0)} = 0, \quad \delta y^{(0)} = 0, \quad \delta \lambda^{(0)} = 0
\]

GICOLAG Workshop On the development of IP software for Quadratic Programming 21
CG vs SQMR

- Conjugate Gradient
  - Lower computational cost (1 mat-vet product per iteration)
  - Less robust (numerical experiments on QP problems with linear inequality constraints)

- Simplified QMR
  - Higher computational cost (2 mat-vet products per iteration)
  - More robust
Building CP

\[ P = \begin{bmatrix} C & -G^T \\ -G & -F \end{bmatrix} = LDL^T \]

\[ C = \text{diag}(Q + E) \]

\[ P = \begin{bmatrix} C & -G^T \\ -G & -F \end{bmatrix} = \begin{bmatrix} I & 0 \\ -GC^{-1} & I \end{bmatrix} \begin{bmatrix} C & 0 \\ 0 & -N \end{bmatrix} \begin{bmatrix} I & -C^{-1}G^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} I & 0 \\ -GC^{-1} & L_0 \end{bmatrix} \begin{bmatrix} C & 0 \\ 0 & -D_0 \end{bmatrix} \begin{bmatrix} I & -C^{-1}G^T \\ 0 & L_0^T \end{bmatrix} \]

\[ N = F + GC^{-1}G^T = L_0D_0L_0^T \]
Approximating CP

- Inexact CP
  - Approximation of $G$ (Bergamaschi, Gondzio et al., 2006)
  - ICF of $F + GC^{-1}G^T$ (Benzi & Simoncini, 2006)
  - Incomplete Schilders’ factorization (Dollar, Gould & Wathen, 2004)

- Keep CP fixed for some iterations (reuse it)
  - The idea is not new
    (normal equations: Carpenter & Shanno, 1993; Karmakar & Ramakrishnan, 1991)
Reusing CP (work in progress)

while (PR stopping criterion not satisfied) do

... 

\( k = k+1 \)
build and factorize \( P^k \)
apply to the KKT system the Krylov solver with \( P^k \)
\( j = k; \)
\( l = 1; \)
while (\( \delta^k > \varepsilon \cdot TOL_{PR} \) and \( iit(k) \leq 2 \cdot iit(k-1) \) and \( l \leq l_{max} \)) do

apply to the KKT system the Krylov solver with \( P^j \)
\( k = k+1 \)
\( l = l+1 \)
endwhile 

... 

endwhile 

\( \delta^k = \Delta / (1 + |p(x)|), \quad \varepsilon = 10, \)
\( iit(k) = \# \text{ inner iter. at outer step } k, \quad l_{max} \text{ dynamically set} \)
Convergence of the inexact PR method

\[ \rho \geq 2n + m + \sqrt{2n + m} \]

\[ w^0 = \gamma\alpha(e,e,e,e,0,e,e), \quad \alpha > 0, \quad 0 < \gamma < 1, \quad \|w^*\|_{\infty} \leq \alpha, \quad w^* \text{ opt. solution} \]

\[ \|r\| \leq \frac{\sqrt{3}}{4} \cdot \frac{\Delta}{\rho}, \quad r = \text{residual of the approx. KKT solution} \]

a step length \( \overline{\vartheta} = \overline{\vartheta}(\gamma) \) exists such that

\[ \Phi(\overline{w}) - \Phi(w) < -\delta, \quad \overline{w} = w + \overline{\vartheta}\delta w, \quad \delta > 0 \]

\[ \overline{\Delta} \geq (1 - \overline{\vartheta})\Delta \]

\[ \overline{\Delta} / \Delta^0 \geq \overline{\sigma} / \sigma^0 \]

(Cafieri, D'Apuzzo, De Simone, di Serafino, Toraldo, 2005)

\[ \forall \varepsilon \in (0,1) \quad \exists K : \quad \Delta^k \leq \varepsilon \quad \forall k \geq K, \quad K = O((\rho - 2n - m) |\ln \varepsilon|) \]

polynomial convergence
Adapting CG/SQMR accuracy to PR iterates

CG/SQMR stopping criterion: $\| r \| \leq TOL$

- **Theory**
  $$TOL = \frac{\sqrt{3}}{4} \cdot \frac{\Delta}{\rho}$$

- **Computational study**
  - Reduce the number of inner iterations
  - Avoid slowdown in decreasing the infeasibility

  $TOL = \begin{cases} 
  \min \left\{ \tau \frac{\Delta}{\rho}, \ TOL_{PR} \right\}, & \text{if } \delta \leq \varepsilon \cdot TOL_{PR} \\
  \tau \frac{\Delta}{\rho}, & \text{otherwise} 
  \end{cases}$

  With safeguard strategies

  $\delta = \Delta / (1 + | p(x) |)$

  $\tau = 1, \ 10$;  $\varepsilon = 10$

(Cafieri, D’Apuzzo, De Simone, di Serafino, 2005)
Perform a centering step

\[
\min \left\{ x_i s_i, y_j z_j, v_i t_i \right\} < \eta \frac{\Delta}{2n+m}
\]

- Keeps the current infeasibility
- Increases the duality gap
Computing the step length

\[ \bar{w} = w + \bar{\vartheta} \delta w \]

- **Theory**

  \[ \begin{align*}
  \bar{\vartheta} &= \max \left\{ \vartheta \in (0, \tilde{\vartheta}] : \Delta \geq (1 - \vartheta) \Delta \right\} \\
  \tilde{\vartheta} &= \arg \min \Phi, \quad \vartheta_{\max} = \max \left\{ \vartheta \geq 0 : w + \vartheta \delta w > 0 \right\}
  \end{align*} \]

- **Implementation**

  \[ \bar{\vartheta} \in (0, \beta \cdot \vartheta_{\max}], \quad \beta \approx 1, \quad \text{such that} \quad \Delta \geq (1 - \vartheta) \Delta \]  
  (backtracking)
Starting Point

Theory

- \( \alpha^0 > 0, \ 0 < \gamma < 1 \) such that \( w^0 = \alpha^0(e,e,e,e,0,e,e) \) and \( \|w^*\|_\infty \leq \alpha^0 / \gamma \)

  a "suitable" step length \( \vartheta = \vartheta(\gamma) \) exists

- \( \eta \alpha^0 \max\{ \| (x,s) \|_1, \| (y,z) \|_1, \| (v,t) \|_1 \} > (2/\gamma)\Delta, \ \eta = \eta(\vartheta) \)

  \( w^* \) such that \( \|w^*\|_\infty \leq \alpha / \gamma \) does not exists

  reduce \( \gamma \) to continue the PR algorithm

Implementation

- Start from \( w^0 = \alpha^0(e,e,e,e,0,e,e) \) such that \( \Delta^0 \geq \chi \sigma^0, \ \chi \geq 10 \)

- If the step length becomes “too small”, increase \( \alpha \) by a factor \( \omega \geq 10 \) and restart

Very small steps may be chosen!

Computation experiments have shown that the restart often takes place in the first few PR iterations. But more investigation is needed!
Stopping criteria & main algorithmic parameters

\[ \delta \leq TOL_{PR}, \quad \max\left\{ \sigma_{p_1}, \sigma_{p_2}, \sigma_{p_3}, \sigma_d \right\} \leq TOL_{2PR}, \quad it \leq \text{maxit} \]

\[ \delta^k = \frac{\Delta^k}{1 + \| p(x) \|}, \quad \sigma_{p_1}^k = \frac{\| r_{p_1}^k \|}{1 + \| b \|}, \quad \sigma_{p_2}^k = \frac{\| r_{p_2}^k \|}{1 + \| u \|}, \quad \sigma_{p_3}^k = \frac{\| r_{p_1}^k \|}{1 + \| d \|}, \quad \sigma_{p_1}^k = \frac{\| r_d^k \|}{1 + \| c \|} \]

\( it = \# \) of PR iterations

\[ \rho = c(2n + m), \quad c = \begin{cases} 10 & \text{if } 2m + n < 15000 \\ 6 & \text{otherwise} \end{cases} \]

\( \beta = 0.99 \) (step length : \( \bar{\vartheta} \in (0, \beta \cdot \vartheta_{\text{max}}) \) such that \( \bar{\Delta} \geq (1 - \vartheta)\Delta \))

\( \eta = 10^{-3} \) (centering : \( \min \{ x_i s_i, y_j z_j, v_t \} < \eta \Delta / (2n + m) \))

\( \chi = 10, \ \omega = 10, \ 100 \) (starting point : \( w^0 = \alpha^0 (e, e, e, 0, e, e) \) such that \( \Delta^0 \geq \chi \sigma^0 \), \( \alpha^0 \leftarrow \omega \alpha^0 \))

\( TOL_{PR} = 10^{-7}, \quad TOL_{2PR} = 10^{-8}, \quad \text{maxit} = 80 \)
PRQP software

PRQP
Potential Reduction software for Quadratic Programming

- Infeasible/feasible primal-dual PR algorithm
- Different solvers for the KKT system: direct, CG, SQMR
- Exact and reused Constraint Preconditioner, limited-memory ICF for bound constrained problems
- Computational strategies previously described

Linear Algebra kernels
- BLAS level 1
- ICFS (CG + limited-memory ICF, sparse matrix-vector product)
- HSL MA27 (sparse $LDL^T$ + solve)
- Custom SQMR and sparse matrix-matrix product
Testing environment

- Test problems: 48 convex QP problems from CUTEr
  - $1000 \leq n \leq 50000$
  - High sparsity of Hessian and constraint matrices (≥99%)
  - 30 problems with diagonal Hessian matrix (the Constraint Preconditioner is the augmented system matrix!)

- Comparisons with other packages
  - KNITRO: direct, CG
  - MOSEK
  - Comparisons using time performance profiles (Dolan & Moré, 2002)

- Computational environment
  - 2.53 GHz Pentium IV, 1.256 GB RAM, 128 KB L1 cache
  - Linux Red Hat 9.0 O.S., g77 and gcc 3.2.2 compilers
Comparison of PRQP versions

All the test problems
Comparison of PRQP versions

Problems with non-diagonal Hessian

All the test problems
Comparison of PRQP versions (cont’d)

Problems with linear inequality and box constraints: CG vs SQMR

Failures (PR does not converge in 80 iterations)
PRQP-QMR: 1; PRQP-CG: 4
### PRQP versions: some details

<table>
<thead>
<tr>
<th>PROBLEM</th>
<th>n/m/p</th>
<th>PRQP-DIR</th>
<th>PRQP-QMR</th>
<th>PRQP-CG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IT</td>
<td>TIME</td>
<td>IT</td>
</tr>
<tr>
<td>AUG3DQP*</td>
<td>27543/0/8000</td>
<td>20</td>
<td>4.68e+1</td>
<td>20</td>
</tr>
<tr>
<td>CVXQP1</td>
<td>1000/0/500</td>
<td>14</td>
<td>1.22e+1</td>
<td>15</td>
</tr>
<tr>
<td>CVXQP1</td>
<td>10000/0/5000</td>
<td>19</td>
<td>1.23e+4</td>
<td>19</td>
</tr>
<tr>
<td>CVXQP3</td>
<td>1000/0/750</td>
<td>18</td>
<td>3.80e+1</td>
<td>19</td>
</tr>
<tr>
<td>CVXQP3</td>
<td>10000/0/7500</td>
<td>---§</td>
<td>-----</td>
<td>16</td>
</tr>
<tr>
<td>DTOC3*</td>
<td>14999/0/9998</td>
<td>19</td>
<td>6.87e-1</td>
<td>19</td>
</tr>
<tr>
<td>LISWET7*</td>
<td>10002/1000/0</td>
<td>37</td>
<td>1.38e+0</td>
<td>37</td>
</tr>
<tr>
<td>LISWET11*</td>
<td>10002/1000/0</td>
<td>40</td>
<td>1.52e+0</td>
<td>40</td>
</tr>
</tbody>
</table>

* = diagonal Hessian, § = too high memory requirements, ♦ = max IT achieved
Comparison of PRQP, KNITRO and MOSEK

31 test problems
Comparison of PRQP, KNITRO and MOSEK

8 problems with non-diagonal Hessian

31 test problems
## PRQP, KNITRO and MOSEK: some details

<table>
<thead>
<tr>
<th>PROBLEM</th>
<th>n/m/p</th>
<th>PRQP-DIR</th>
<th>PRQP-QMR</th>
<th>KNIT-DIR</th>
<th>KNIT-CG</th>
<th>MOSEK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IT TIME</td>
<td>IT/IIT TIME</td>
<td>IT TIME</td>
<td>IT/IIT TIME</td>
<td>IT TIME</td>
</tr>
<tr>
<td>CVXQP1</td>
<td>1000/0/500</td>
<td>14</td>
<td>15/166</td>
<td>11</td>
<td>10/1571</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.22e+1</td>
<td>1.57e-1</td>
<td>1.27e+0</td>
<td>6.40e-1</td>
</tr>
<tr>
<td>CVXQP1</td>
<td>10000/0/5000</td>
<td>19</td>
<td>19/290</td>
<td>13</td>
<td>80♦/9932</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.23+e4</td>
<td>1.58e+1</td>
<td>8.04e+2</td>
<td>2.04e+2</td>
</tr>
<tr>
<td>HAGER3</td>
<td>5001/0/2500</td>
<td>14</td>
<td>14/49</td>
<td>12</td>
<td>10/16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.18e-1</td>
<td>2.33e-1</td>
<td>6.50e-1</td>
<td>6.72e+2</td>
</tr>
<tr>
<td>HUESTIS*</td>
<td>5000/0/2</td>
<td>14</td>
<td>14/14</td>
<td>21</td>
<td>80♦/1040</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.09e-1</td>
<td>1.38e-1</td>
<td>6.70e-1</td>
<td>7.00e-2</td>
</tr>
<tr>
<td>LISWET2*</td>
<td>10002/1000/0</td>
<td>19</td>
<td>19/21</td>
<td>32</td>
<td>25/193</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7.18e-1</td>
<td>8.92e-1</td>
<td>3.92e+0</td>
<td>1.16</td>
</tr>
<tr>
<td>LISWET11*</td>
<td>10002/1000/0</td>
<td>40</td>
<td>40/41</td>
<td>34</td>
<td>43/221</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.52e+0</td>
<td>1.90e+0</td>
<td>4.08e+0</td>
<td>1.28e+0</td>
</tr>
</tbody>
</table>

♦ = diagonal Hessian,  ♦♦ = max IT achieved

GICOLAG Workshop  On the development of IP software for Quadratic Programming
Preliminary results by reusing CP

Adaptive choice of reusing CP during the PR algorithm
Conclusions …

- Potential Reduction may be competitive with other IP algorithms
- Iterative linear algebra solvers (with suitable preconditioners and adaptive stopping criteria) are key ingredients for achieving efficiency
- Further investigation is needed to improve the starting point strategy

... and future work

- Extension to nonconvex problems
- Dealing with degeneracy
- Presolving
- ...

Extension to nonconvex problems

Only limited attention has been devoted to exploiting the QP problem features

- **Algorithms involving the use of trust regions**

- **Algorithms using suitable modifications of the Hessian of the objective function (or of the Hessian of the Lagrangian)**
  (Absil & Tits, 2005)

- **Convergence usually proved for feasible algorithms**
Dealing with degeneracy

- The constraint matrix is rank deficient
  - presolve analysis
  - regularization + safeguard strategies

- There are weakly active constraints
  - ad hoc strategies to prevent slowdown of convergence