Journées Franciliennes de Recherche Opérationnelle
Conservatoire National des Arts et Métiers, Paris, France

How the augmented Lagrangian algorithm can deal with an infeasible convex quadratic optimization problem — Motivation, analysis, implementation

J.Ch. Gilbert (INRIA-Paris, France)

Joint work with

Alice Chiche (EDF ↷ Artelys, France)
Émilie Joannopoulos (INRIA-Paris ↷ Sherbrooke Univ., Canada)

June 23, 2016

Outline

Outline

Convex quadratic optimization

The QP to solve

The QP to solve

The problem to solve

$$(P) \quad \begin{cases} \inf_{x \in \mathbb{R}^n} q(x) \\ l \leqslant Ax \leqslant u, \end{cases} \tag{1}$$

where q is a convex quadratic function defined at $x \in \mathbb{R}^n$ by

$$q(x) = g^\mathsf{T} x + \frac{1}{2} x^\mathsf{T} H x$$

and

- $g \in \mathbb{R}^n$
- $H \succcurlyeq 0$ (NP-hard otherwise, (P) encompasses linear optimization),
- A is $m \times n$,
- $l,\ u \in \bar{\mathbb{R}}^m$ satisfy $l < u$.

Also equality constraints in all solvers.

Can one still make progress in convex quadratic optimization?

The problem is polynomial and can be solved by

- active-set methods $\to$ probably non-polynomial,
- interior-point methods $\to$ polynomial,
- nonsmooth methods $\to$ polynomial on subclasses,
- other methods (including the augmented Lagrangian method).

Has this discipline been fully explored in the XXth century?

Observation 1. Odd behavior of Quadprog (Matlab). If the data is

$$g = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & 2 & 1 \end{pmatrix}, \quad x \geqslant \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix},$$

Quadprog-`active-set` answers

```
Exiting:  the solution is unbounded and at infinity;
Function value:  3.20000e+33
```

Very odd, since the problem has a *unique* solution, which is

$$x = \begin{pmatrix} -1 \\ -1 \\ 2 \end{pmatrix} \qquad \text{and} \qquad \text{val}(P) = -1.5.$$

It is a benign flaw, since if $H \curvearrowright H + \varepsilon I$, Quadprog finds a near solution.

Convex quadratic optimization

Can one still make progress in convex quadratic optimization?

Quadprog-`reflective-trust-region` (default algorithm) answers

```
Optimization terminated:  relative function value changing by
 less than OPTIONS.TolFun.
Function value:  -1.5
```

Correct answer!

Conclusion: the good algorithm may depend on the problem.

Convex quadratic optimization

Can one still make progress in convex quadratic optimization?

Observation 2. On the *solvable* convex QPs of the CUTEst collection:

- first group: 138 problems, solvers in Fortran or C++,
- second group: 58 problems ($n \leqslant 500$), solver in Matlab.

Solvers	% failure	% too slow	% infeasibility	% other
Qpa (AS)	30 %	15 %	15 %	–
Qpb (IP)	20 %	5 %	2 %	13 %
Ooqp (IP)	54 %	1 %	12 %	41 %
Quadprog (AS)	33 %	12 %	19 %	2 %

- "too slow": requires more than 600 seconds,
- "infeasibility": wrong diagnosis of infeasibility,
- "other": "too small stepsize", "too small direction", "ill-conditioning", and "unknown".

The problem does not come from some very difficult QPs.
For example, on the CUTEst problem QSCTAP1 ($n = 480$, $n_b = 480$ lower bounds, $m_I = 180$ lower bounds, $m_E = 120$):

- Qpa claims that the problem is unbounded,
- Qpb claims that the problem has a solution,
- Ooqp claims that the problem is infeasible,
- Quadprog stops on a too large number of iterations ($\geqslant 10^4$).

$\Longrightarrow$ Still progress to do.

Observation 3 (more important).

Most (all?) solvers do not give appropriate information
when the QP is special, they just return a flag.

- Special means $\text{val}(P) \notin \mathbb{R}$ below:
  - $\text{val}(P) \in \mathbb{R} \iff$ the problem has a solution (Frank-Wolfe [8; 1956]),
  - $\text{val}(P) = -\infty \iff$ the problem is feasible and unbounded,
  - $\text{val}(P) = +\infty \iff$ the problem is infeasible.
- Appropriate means useful when the QP solver is used in the SQP algorithm for solving a nonlinear optimization problem.

Goal of this study

- Having a robust and efficient active-set-like convex QP solver for the SQP algorithm.
  - Efficient of course!
  - Robust $\Longrightarrow$ deals appropriately with the special cases.
  - Other terms require to recall the definition of the SQP algorithm.

The SQP algorithm for solving a nonlinear optimization problem

- A standard generic nonlinear optimization problem consists in

$$(P_{EI}) \quad \begin{cases} \inf_x \; f(x) \\ c_E(x) = 0 \\ c_I(x) \leqslant 0, \end{cases}$$

  where $f : \mathbb{R}^n \to \mathbb{R}$, $c_E : \mathbb{R}^n \to \mathbb{R}^{m_E}$, and $c_I : \mathbb{R}^n \to \mathbb{R}^{m_I}$ are smooth (possibly non convex).

- The osculating quadratic problem to (P_{EI}) at (x_k, λ_k) is the problem in d:

$$(\text{OQP}) \quad \begin{cases} \inf_d \; \nabla f(x_k)^{\mathsf{T}} d + \frac{1}{2} d^{\mathsf{T}} \nabla^2_{xx} \ell(x_k, \lambda_k) d \\ c_E(x_k) + c_E'(x_k) d = 0 \\ c_I(x_k) + c_I'(x_k) d \leqslant 0, \end{cases}$$

  whose multipliers are $\lambda_k^{\mathrm{QP}} := \lambda_k + \mu$.

- One iteration of the local SQP/SQO algorithm: from (x_k, λ_k) to (x_{k+1}, λ_{k+1})
  - If possible, solve (OQP), to get d_k and λ_k^{QP}.
  - Update $x_{k+1} := x_k + d_k$ and $\lambda_{k+1} := \lambda_k^{\mathrm{QP}}$.

- Remarks
  - There is a sequence of QP's to solve
    $\Longrightarrow$ interest to have a good QP solver.
  - The (OQP) is NP-hard without convexity
    $\Longrightarrow$ interesting to take $M_k \succcurlyeq 0$ approximating $\nabla^2_{xx}\ell(x_k, \lambda_k)$.
  - If strict complementarity holds at the searched solution of (P_{EI}), the active constraints of (OQP) are those of (P_{EI})
    $\Longrightarrow$ active-set is interesting (only a single linear system to solve per iteration asymptotically).

Outline

Towards the AL algorithm

- The problem is transformed by using an auxiliary variable y:

$$(P) \quad \left\{ \begin{array}{l} \inf_{x\in\mathbb{R}^n} q(x) \\ l \leqslant Ax \leqslant u \end{array} \right. \qquad \curvearrowright \qquad (P') \quad \left\{ \begin{array}{l} \inf_{(x,y)\in\mathbb{R}^n\times\mathbb{R}^m} q(x) \\ Ax = y \\ l \leqslant y \leqslant u. \end{array} \right.$$

- Equality constraints penalized by the augmented Lagrangian

$$\ell_r(x,y,\lambda) := q(x) + \lambda^{\mathsf T}(Ax-y) + \frac{r}{2}\|Ax-y\|^2.$$

- At each iteration the AL algorithm [14, 15, 16, 3, 1, 18, 19; 1969-74] solves

$$\inf_{(x,y)\in\mathbb{R}^n\times[l,u]} \ell_r(x,y,\lambda). \tag{2}$$

- The AL algorithm makes sense if it is easier to solve (2) than (P).

The AL algorithm for a solvable convex QP

One iteration, from $(\lambda_k, r_k) \in \mathbb{R}^m \times \mathbb{R}_{++}$ to (λ_{k+1}, r_{k+1}):

- Compute (if possible, exit otherwise)

$$(x_{k+1}, y_{k+1}) \in \operatorname*{arg\,min}_{(x,y)\in\mathbb{R}^n\times[l,u]} \ell_{r_k}(x,y,\lambda_k).$$

- Update the multipliers

$$\lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where } s_{k+1} := y_{k+1} - Ax_{k+1}.$$

- Stop if

$$s_{k+1} \simeq 0.$$

- Update $r_k \curvearrowright r_{k+1} > 0$: $\rho_k := \|s_{k+1}\|/\|s_k\|$ and

$$r_{k+1} := \max\left(1, \frac{\rho_k}{\rho_{\rm des}}\right) r_k.$$

Understanding the AL algorithm I
Update rule of λ_k

One iteration, from $(\lambda_k, r_k) \in \mathbb{R}^m \times \mathbb{R}_{++}$ to (λ_{k+1}, r_{k+1}):

- Compute (if possible, exit otherwise)
$$(x_{k+1}, y_{k+1}) \in \operatorname*{arg\,min}_{(x,y)\in\mathbb{R}^n\times[l,u]} \ell_{r_k}(x, y, \lambda_k).$$
- Update the multipliers
$$\boxed{\lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where } s_{k+1} := y_{k+1} - Ax_{k+1}.}$$
- Stop if
$$s_{k+1} \simeq 0.$$
- Update $r_k \curvearrowright r_{k+1} > 0$: $\rho_k := \|s_{k+1}\|/\|s_k\|$ and
$$r_{k+1} := \max\left(1, \frac{\rho_k}{\rho_{\rm des}}\right) r_k.$$

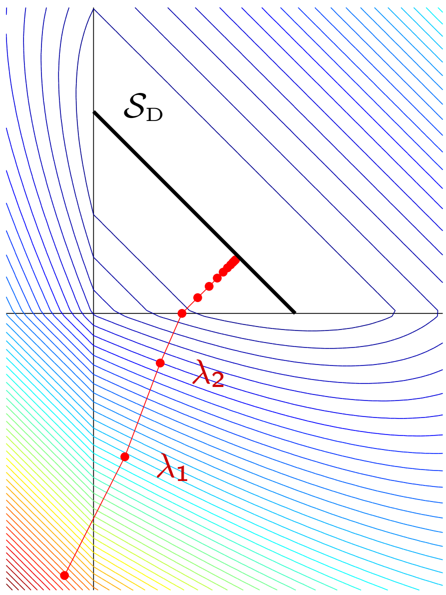
The secrets are in the dual space

- The dual function $\delta : \mathbb{R}^m \to \overline{\mathbb{R}}$, defined at $\lambda \in \mathbb{R}^m$ by
$$\delta(\lambda) := -\inf_{(x,y)\in\mathbb{R}^n\times[l,u]} \Big(q(x) + \lambda^\mathsf{T}(Ax - y)\Big).$$
  - δ is convex, closed, and $\delta > -\infty$,
  - $\operatorname{dom}\delta \neq \varnothing \iff \delta \not\equiv +\infty \iff \delta \in \overline{\operatorname{Conv}}(\mathbb{R}^m)$,
  - piecewise quadratic (quadratic on each orthant).
- If $(P) \equiv (P')$ has a solution:
$$0 \in \partial\delta(\bar\lambda) \iff \bar\lambda \text{ is a dual solution to } (P').$$
- The AL algorithm looks for a
$$\bar\lambda \in \operatorname*{arg\,min} \delta.$$

- AL algorithm = proximal algorithm on δ [17; 1973].
  - If $\delta \in \overline{\text{Conv}}(\mathbb{R}^m)$ and $r_k > 0$, this means that
    $$\lambda_{k+1} = \arg\min_{\lambda \in \mathbb{R}^m} \left(\delta(\lambda) + \frac{1}{2r_k} \|\lambda - \lambda_k\|^2 \right).$$
    One writes $\lambda_{k+1} = \text{prox}_{\delta, r_k}(\lambda_k)$.
  - The optimality condition $0 \in \partial\delta(\lambda_{k+1}) + \frac{1}{r_k}(\lambda_{k+1} - \lambda_k)$ and
    $$\lambda_{k+1} = \lambda_k - r_k s_{k+1}$$
    imply that
    $$s_{k+1} := y_{k+1} - Ax_{k+1} \text{ is in } \partial\delta(\underbrace{\lambda_{k+1}}_{\text{not } \lambda_k!}).$$
    Hence it is an implicit subgradient method.
  - Hence by looking for a λ such that $0 \in \partial\delta(\lambda)$, the AL algorithm tries to vanish the constraint $y - Ax$.

AL iterates minimizing the dual function for a solvable QP

- δ is piecewise quadratic
  $$\delta(\lambda) = \frac{1}{2}\lambda^{\mathsf{T}} S \lambda + (v + y_\lambda)^{\mathsf{T}} \lambda + C^{\text{st}}$$
- $\mathcal{S}_{\mathrm{D}} := \arg\min \delta$
- $\partial\delta(\lambda_{k+1})$ contains
  $$\frac{\lambda_k - \lambda_{k+1}}{r_k} = y_{k+1} - Ax_{k+1}$$
- small r_k's in the figure



Understanding the AL algorithm II
Update rule of r_k

One iteration, from $(\lambda_k, r_k) \in \mathbb{R}^m \times \mathbb{R}_{++}$ to (λ_{k+1}, r_{k+1}):

- Compute (if possible, exit otherwise)
$$(x_{k+1}, y_{k+1}) \in \underset{(x,y)\in\mathbb{R}^n\times[l,u]}{\arg\min} \ \ell_{r_k}(x, y, \lambda_k).$$
- Update the multipliers
$$\lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where } s_{k+1} := y_{k+1} - Ax_{k+1}.$$
- Stop if
$$s_{k+1} \simeq 0.$$
- Update $r_k \curvearrowright r_{k+1} > 0$: $\rho_k := \|s_{k+1}\|/\|s_k\|$ and
$$\boxed{r_{k+1} := \max\left(1, \frac{\rho_k}{\rho_{\rm des}}\right) r_k.}$$

- The update rule of r_k is based on the following global linear convergence result [6; 2005].
  - If (P) has a solution, then the dual solution set $\mathcal{S}_{\rm D} \neq \varnothing$ and
$$\boxed{\begin{array}{c}\forall\, \beta > 0, \quad \exists\, L > 0, \quad \text{dist}_{\mathcal{S}_{\rm D}}(\lambda_0) \leqslant \beta \quad \text{implies that} \\ \forall\, k \geqslant 1, \quad \|s_{k+1}\| \leqslant \min\left(1, \frac{L}{r_k}\right) \|s_k\|,\end{array}} \tag{3}$$
    where $s_k := y_k - Ax_k$.
  - (3) comes from a quasi-global error bound on the dual solution set $\mathcal{S}_{\rm D}$:
$$\boxed{\begin{array}{c}\text{for any bounded set } \mathcal{B} \subset \mathbb{R}^m, \text{ there is an } L > 0, \text{ such that} \\ \forall\, \lambda \in \mathcal{S}_{\rm D} + \mathcal{B}: \quad \text{dist}_{\mathcal{S}_{\rm D}}(\lambda) \leqslant L \left(\inf_{s\in\partial\delta(\lambda)} \|s\| \right).\end{array}} \tag{4}$$
  - The Lipschitz constant L is difficult to deduce from the data ...

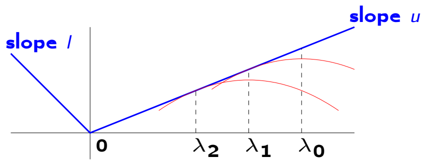
The Lipschitz constant L is difficult to deduce from the data ...

- Let $m = 1$ and $l < 0 < u$. Consider the problem

$$\left\{ \begin{array}{l} \inf 0 \\ l \leqslant 0x \leqslant u, \end{array} \right.$$

- The dual function reads

$$\delta(\lambda) = \left\{ \begin{array}{ll} l\lambda & \text{if } \lambda \leqslant 0 \\ u\lambda & \text{if } \lambda > 0. \end{array} \right.$$



- Hence $\mathcal{S}_{\rm D} = \{0\}$ and the quasi-global error bound reads

$$\forall\, B > 0, \quad \exists\, L > 0, \quad |\lambda| \leqslant B \quad \Longrightarrow \quad |\lambda| \leqslant \left\{ \begin{array}{ll} -Ll & \text{if } \lambda < 0 \\ 0 & \text{if } \lambda = 0 \\ Lu & \text{if } \lambda > 0. \end{array} \right.$$

- Therefore, for $\mathcal{B}$ fixed, $L \nearrow \infty$ when $l \nearrow 0$ or $u \searrow 0$ (fix λ in the error bound).

The rule of the *nonlinear* solver Algencan [2; 2014]:

$$r_0 = P_{[10^{-8},10^{+8}]}\left(10\,\frac{\max(1,|q(x_0)|)}{\max(1,\|Ax_0 - y_0\|^2)}\right).$$

- Motivation: balancing the objective and constraint parts of the ℓ_2 penalty function.
- In the previous example, the rule yields (whatever is l and u):

$$r_0 = 10.$$

- It does not catch the following fact:

for some problems, the appropriate r depends on
the distance from the optimal constraint value $A\bar{x}$ to $[l, u]^c$.

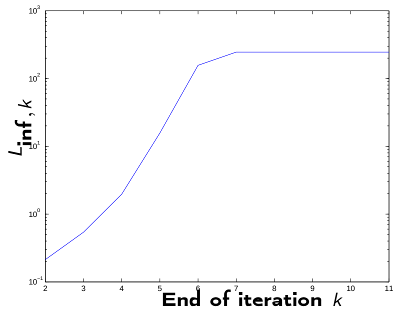
The AL algorithm
The AL algorithm for a solvable convex QP

In Oqla/Qpalm, L is guessed and r_k is set by the observation of $\rho_k := \|s_{k+1}\|/\|s_k\|$, thanks to the global linear convergence:

$$\forall\, \beta > 0, \quad \exists\, L > 0, \quad \mathrm{dist}_{\mathcal{S}_\mathrm{D}}(\lambda_0) \leqslant \beta \quad \text{implies that} \quad \forall\, k \geqslant 1, \quad \|s_{k+1}\| \leqslant \frac{L}{r_k}\, \|s_k\|.$$

- Lower bound of L:

$$L_{\mathrm{inf},k} := \max_{1 \leqslant i \leqslant k} \rho_i r_i.$$



- Setting of r_{k+1}:

$$r_{k+1} = \frac{L_{\mathrm{inf},k}}{\rho_\mathrm{des}}.$$

- With $\rho_\mathrm{des} = 1/10$, convergence occurs in 10..15 AL iterations.

The AL algorithm
The AL algorithm for a solvable convex QP

Understanding the AL algorithm III
Effect of the update rule of r_k for infeasible QPs

If the QP is infeasible:

- $\|s_k\| \searrow \sigma > 0$ and

$$\rho_k := \frac{\|s_{k+1}\|}{\|s_k\|} \to 1,$$

- the rule (increases r_k whenever $\rho_k > \rho_\mathrm{des}$ $[\rho_\mathrm{des} < 1]$) $\Longrightarrow r_k \nearrow \infty$,
- the AL subproblems become ill-conditioned,
- could stop when $r_k \geqslant \bar{r}$, but
  - difficult to find a universal threshold $\bar{r}$,
  - no information on the problem on return.

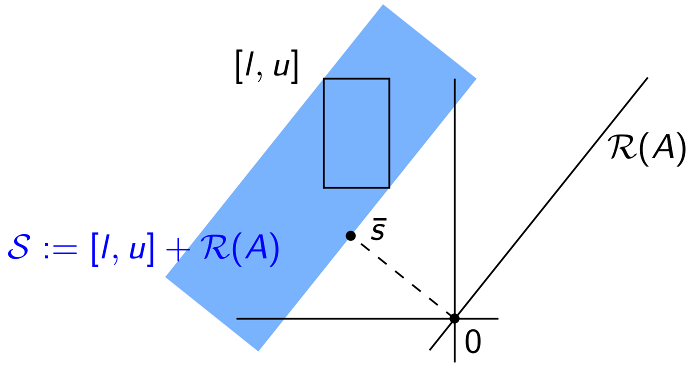
Can one have a global linear convergence when the QP is infeasible?

The smallest feasible shift

- It is always possible to find a shift $s \in \mathbb{R}^m$ such that

$$l \leqslant Ax + s \leqslant u \text{ is feasible for } x \in \mathbb{R}^n.$$

- These feasible shifts are exactly those in $\mathcal{S} := [l, u] + \mathcal{R}(A)$:



- The smallest feasible shift $\bar{s} := \arg\min\{\|s\| : s \in \mathcal{S}\}$.

$$\bar{s} = 0 \qquad \Longleftrightarrow \qquad \text{(P) is feasible.}$$

The closest feasible problem

The shifted QPs (feasible iff $s \in \mathcal{S}$, may be unbounded)

$$(P_s) \quad \begin{cases} \inf_x \ q(x) \\ l \leqslant Ax + s \leqslant u \end{cases} \qquad \text{and} \qquad (P'_s) \quad \begin{cases} \inf_x \ q(x) \\ Ax + s = y \\ l \leqslant y \leqslant u. \end{cases} \tag{5}$$

The closest feasible problems (feasible, may be unbounded)

$$(P_{\bar{s}}) \quad \begin{cases} \inf_x \ q(x) \\ l \leqslant Ax + \bar{s} \leqslant u. \end{cases} \qquad \text{and} \qquad (P'_{\bar{s}}) \quad \begin{cases} \inf_x \ q(x) \\ Ax + \bar{s} = y \\ l \leqslant y \leqslant u. \end{cases} \tag{6}$$

Claims clarified below ([21, 4])

- *The AL algorithm actually "solves" the closest feasible problem* $(P_{\bar{s}})$.
- *The speed of convergence is globally linear.*

The AL algorithm

Detection of unboundedness ($\mathrm{val}(P) = -\infty$)

When is the AL algorithm well defined?

Proposition ([4])

For the convex QP (1), the following properties are equivalent:

(*i*) $\mathrm{dom}\,\delta \neq \varnothing \qquad (\iff \delta \not\equiv +\infty \iff \delta \in \overline{\mathrm{Conv}}(\mathbb{R}^m))$,

(*ii*) *for some/any $s \in \mathcal{S}$, the shifted QP (5) is solvable,*

(*iii*) *for some/any $r > 0$ and $\lambda \in \mathbb{R}^m$, the AL subproblem (2) is solvable,*

(*iv*) *there is no $d \in \mathbb{R}^n$ such that $g^\mathsf{T} d < 0$, $Hd = 0$, and $Ad \in [l, u]^\infty$.*

- C^∞ denotes the asymptotic/recession cone of a convex set C.
- A direction like d in (*iv*) is called here an unboundedness direction.
- The failure of these conditions can be detected on the first AL subproblem (2), by finding a direction d such that

$$g^\mathsf{T} d < 0, \qquad Hd = 0, \qquad \text{and} \qquad Ad \in [l, u]^\infty.$$

- Fundamental assumption: (*i*)-(*iv*) holds from now on.

The AL algorithm

Convergence for an infeasible QP ($\mathrm{val}(P) = +\infty$)

Feasibility and dual function

- No duality gap:

$$\boxed{\text{the QP is feasible} \quad \iff \quad \delta \text{ is bounded below.}}$$

  - [$\Rightarrow$] (contrapositive) true for any convex problem by weak duality.
  - [$\Leftarrow$] (contrapositive) $\delta \not\equiv +\infty$ and $\delta \to -\infty$ along $\bar{s} \neq 0$ ($\mathcal{S}$ is closed).

- Consequence for a convex QP:

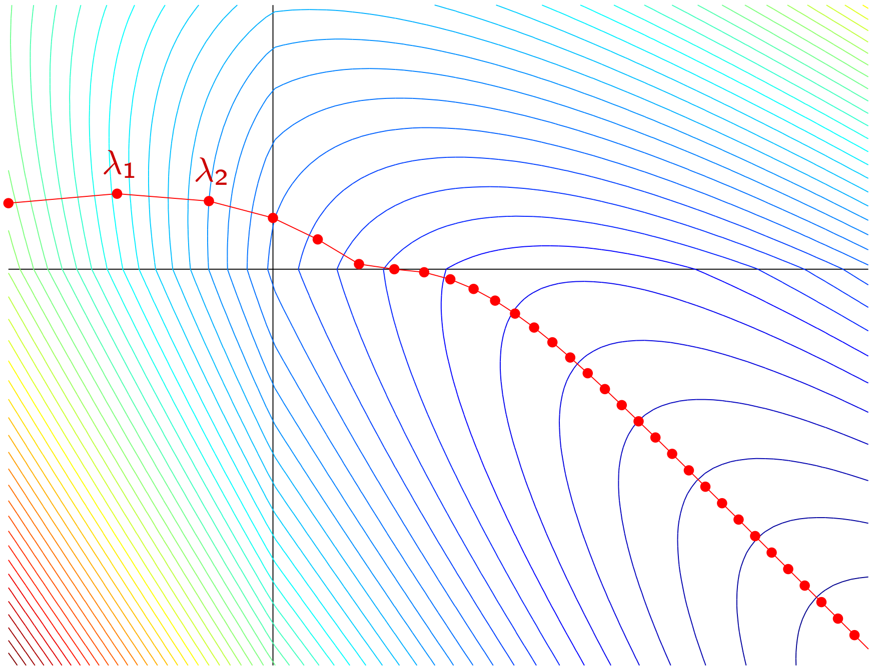
$$\begin{array}{rcl} \text{the QP is infeasible} & \Longrightarrow & \delta \text{ is unbounded below} \\ & \Longrightarrow & \{\lambda_k\} \text{ blows up} \\ & & \text{(by the proximal interpretation).} \end{array}$$

- One can say more.

The AL algorithm

Convergence for an infeasible QP ($\text{val}(P) = +\infty$)

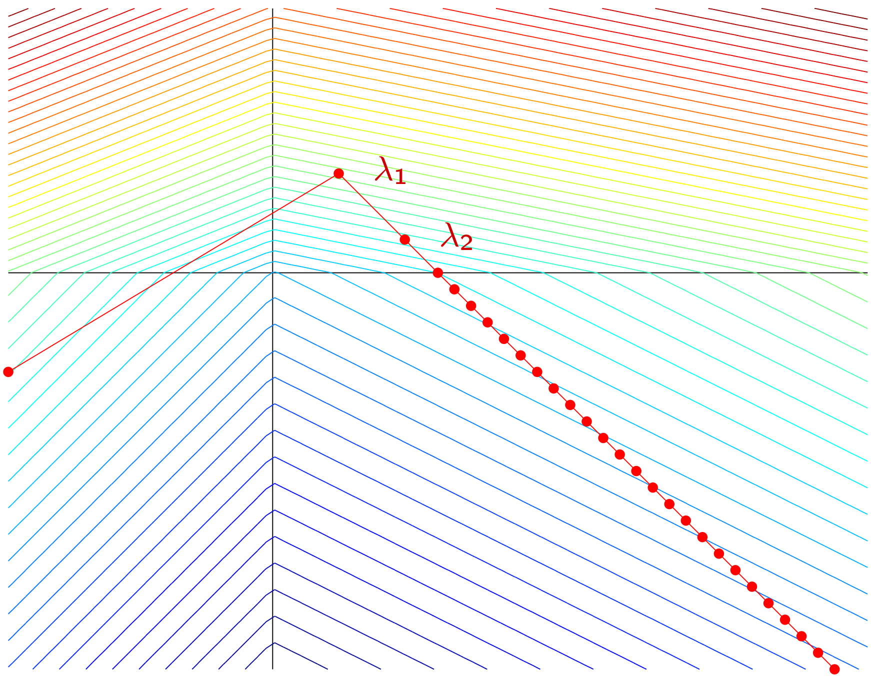
Level curves of the dual function δ (infeasible QP, $H \succ 0$)



The AL algorithm

Convergence for an infeasible QP ($\text{val}(P) = +\infty$)

Level curves of the dual function δ (infeasible QP, $H = 0$)



A surprising identity [4; 2016]

When $\operatorname{dom}\delta \neq \varnothing$,

$$\boxed{\mathcal{S} = \mathcal{R}(\partial\delta).}$$

- Surprising since
  - $\mathcal{S}$ only depends on the constraints of the QP,
  - δ also depends on the objective of the QP.
- We already know that $\mathcal{S} \cap \mathcal{R}(\partial\delta) \neq \varnothing$:

$$\mathcal{S} = [l, u] + \mathcal{R}(A) \ni s_{k+1} := y_{k+1} - Ax_{k+1} \in \partial\delta(\lambda_{k+1}) \subset \mathcal{R}(\partial\delta).$$

Convergence $s_k \to \bar{s}$ [21; 1987]

- "Intuitive proof"

$$\mathcal{S} = [l, u] + \mathcal{R}(A) \ni s_k := y_k - Ax_k \in \partial\delta(\lambda_k) \subset \mathcal{R}(\partial\delta).$$

  - Trust the proximal algo: $y_k - Ax_k \to$ the smallest element in $\mathcal{R}(\partial\delta)$.
  - Now $\mathcal{S} = \mathcal{R}(\partial\delta) \Longrightarrow$ the smallest element in $\mathcal{R}(\partial\delta)$ is $\bar{s}$.
  - Hence $s_k := y_k - Ax_k \to \bar{s}$.

The AL algorithm

Convergence for an infeasible QP (val(P) $= +\infty$)

Global linear convergence $s_k \to \bar{s}$ [4; 2016]

$(P_{\bar{s}})$ with solution $\Rightarrow$ the dual solution set of $(P_{\bar{s}})$, namely

$$\tilde{\mathcal{S}}_{\mathrm{D}} := \{\lambda \in \mathbb{R}^m : \bar{s} \in \partial\delta(\lambda)\}$$

is nonempty and

$$\boxed{\begin{array}{c}\forall\,\beta > 0, \quad \exists\, L > 0, \quad \operatorname{dist}_{\tilde{\mathcal{S}}_{\mathrm{D}}}(\lambda_0) \leqslant \beta \quad \text{implies that} \\ \forall\, k \geqslant 1, \quad \|s_{k+1} - \bar{s}\| \leqslant \frac{L}{r_k}\, \|s_k - \bar{s}\|.\end{array}} \tag{7}$$

Comments:

- Similar to the solvable case, but with $s_k \curvearrowright s_k - \bar{s}$,
- $\bar{s}$ is not known $\Rightarrow$ more difficult to design an update rule for r_k: instead of $s_k - \bar{s}$, observe $s'_k := s_k - s_{k-1} \to 0$ globally linearly.

The AL algorithm

The revised AL algorithm

The revised AL algorithm

Set $\lambda_0 \in \mathbb{R}^m$, $r_0 > 0$, $\rho'_{\mathrm{des}} \in \,]0, 1[$, and repeat for $k = 0, 1, 2, \ldots$

- Compute (if possible, exit with a direction of unboundedness otherwise)

$$(x_{k+1}, y_{k+1}) \in \underset{(x,y)\in\mathbb{R}^n\times[l,u]}{\arg\min}\ \ell_{r_k}(x, y, \lambda_k).$$

- Update the multipliers

$$\lambda_{k+1} = \lambda_k - r_k s_{k+1}, \quad \text{where } s_{k+1} := y_{k+1} - Ax_{k+1}.$$

- Stop if

$$A^{\mathsf{T}}(Ax_{k+1} - y_{k+1}) \simeq 0 \qquad \text{and} \qquad P_{[l,u]}(Ax_{k+1}) \simeq y_{k+1}.$$

- Update $r_k \curvearrowright r_{k+1} > 0$: $s'_k := s_k - s_{k-1}$, $\rho'_k := \|s'_{k+1}\| / \|s'_k\|$, and

$$r_{k+1} := \max\left(1, \frac{\rho'_k}{\rho'_{\mathrm{des}}}\right) r_k.$$

Outline

Numerical results

The codes Oqla and Qpalm and the selected test-problems

Oqla and Qpalm

Implementation of the revised AL algorithm in two solvers [10], soon freely available at https://who.rocq.inria.fr/Jean-Charles.Gilbert:

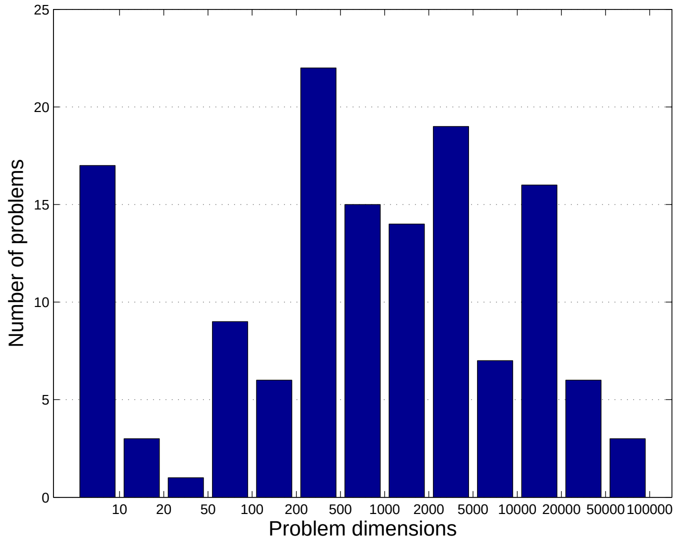
- Oqla
  - in C++,
  - fast execution, but slow implementation,
  - OO → easy to take into account new data structures, like Ooqp [9] (dense, sparse, ℓ-BFGS, ...),
  - AL subproblems solved by an active-set (AS) method,
  - more than 1 year of work for one engineer!
- Qpalm
  - in Matlab,
  - AL subproblems solved by an AS method,
  - fast implementation, easy to try new ideas, but slow execution.

Main objective of these tests: is it worth continuing working on the development of Oqla/Qpalm?

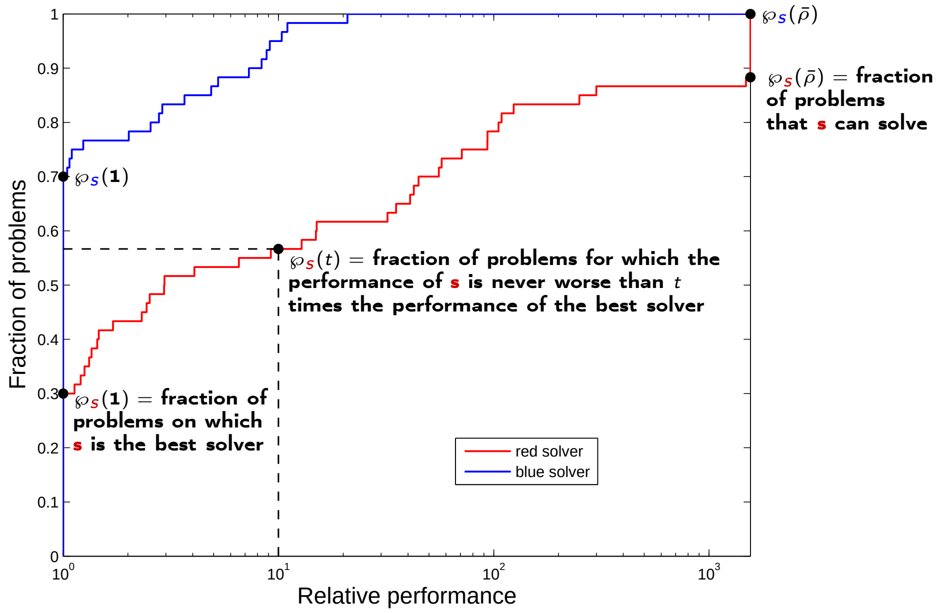
Selected Cutest problems

Comparison made on the Cutest collection of test-problems [13].

- 138 convex quadratic problems (all solvable, but 4?),
- 58 problems among them, with $n \leqslant 500$ (for comparison in Matlab).

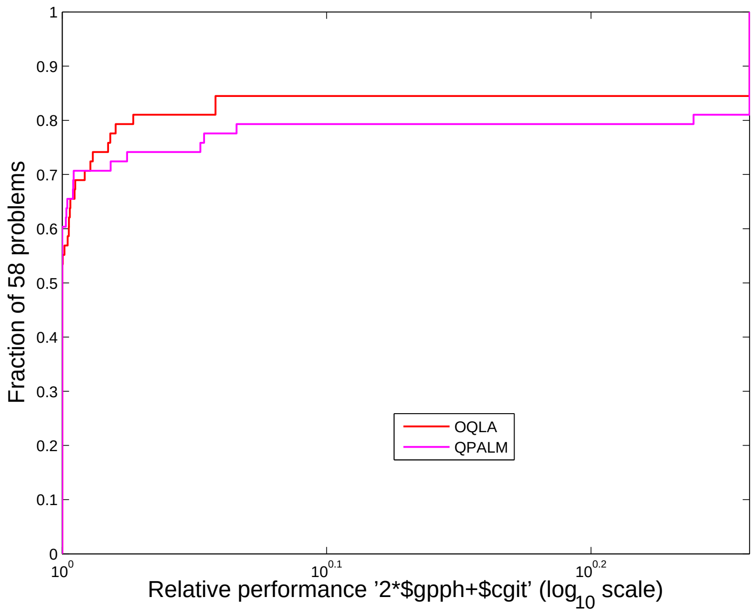


Reading performance profiles [7]



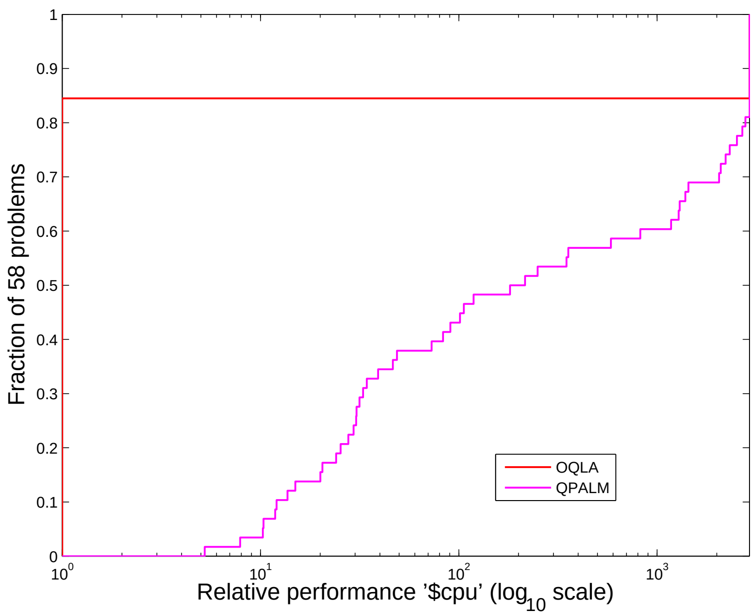
Performance profiles drawn with Libopt [11].

Comparison of Oqla and Qpalm on iteration counters



- Close to each other (see x-axis $[10^{0.05} \simeq 1.12]$ and y-axis [even scores]).
- Difference in failures due to the slowness of Qpalm in Matlab (or still not clear).

Comparison of Oqla and Qpalm on CPU time



- Oqla (in C++) is 10..2000 times faster than Qpalm (in Matlab).

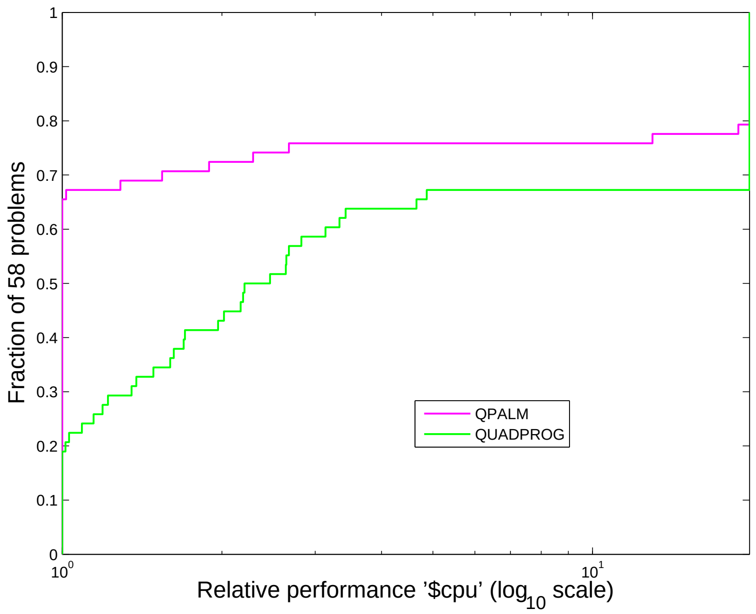
Numerical results
Comparison with active-set methods

Two more codes, which use active-set methods:

- Oqla
  - the standard QP solver of the Matlab optimization toolbox [20],
  - Options `'Algorithm'` → `'active-set'` and `'LargeScale'` → `'off'` ⟹ active-set method.
- Qpa
  - free code,
  - from the Galahad library [12],
  - in Fortran,
  - uses preprocessing and preconditioning?

Numerical results
Comparison with active-set methods

Comparison of Qpalm and Oqla on CPU time

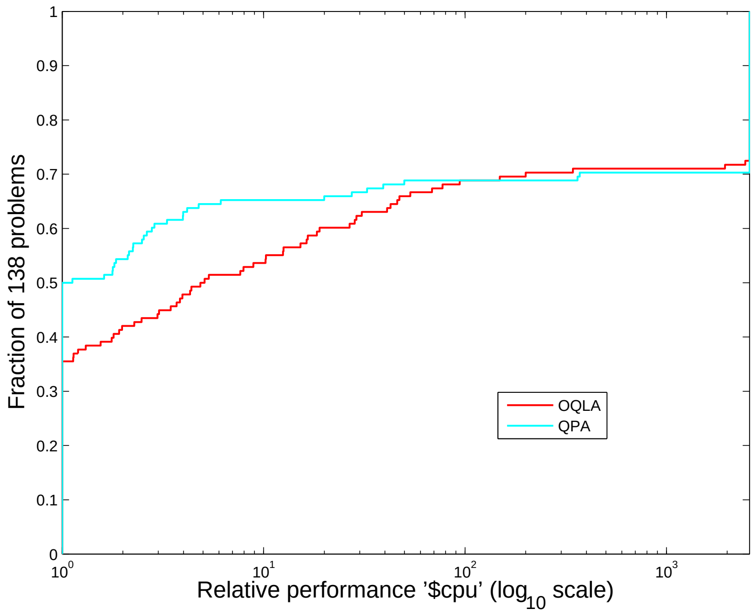


- Qpalm is often twice faster than Oqla (but not always faster).
- Qpalm is more robust than Oqla (81 % success to 67 %).
- Progress is still possible with Qpalm.

Numerical results

Comparison with active-set methods

Comparison of Oqla and Qpa on CPU time



- Qpa is more often faster than Oqla, but not significantly.
- Oqla and Qpa have the same robustness (73 % and 71 % success respectively).
- Progress is still possible with Oqla.

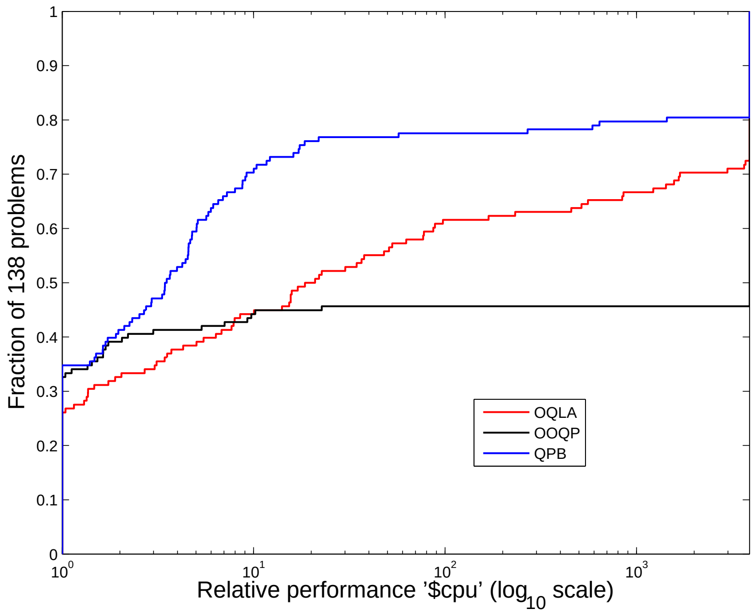
Numerical results

Comparison with interior-point methods

Two more codes, which use interior-point methods:

- Ooqp
  - free code,
  - written by Gertz and Wright in 2003 [9],
  - to show the interest of an OO implementation.
- Qpb
  - free code,
  - from the Galahad library [12],
  - in Fortran,
  - uses preprocessing and preconditioning?

Comparison of Oqla, Ooqp, and Qpb on CPU time

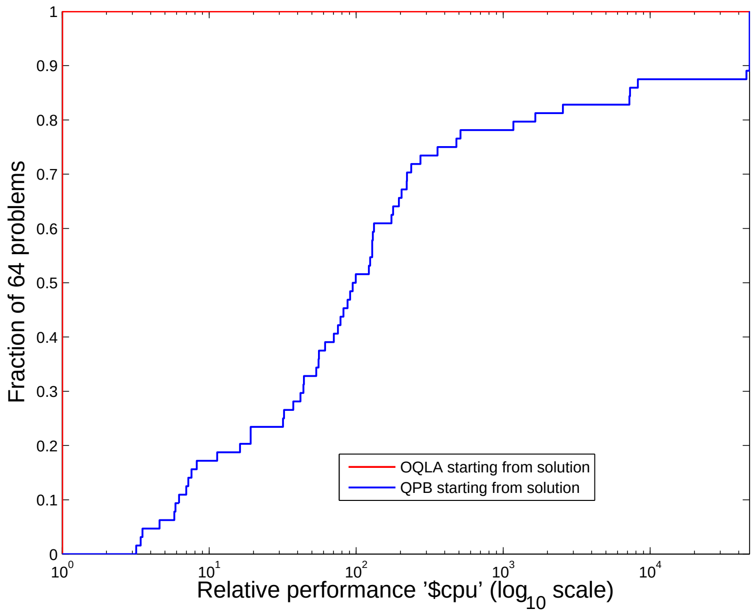


- IP methods are clearly faster than our AL+AS method (in particular with Ooqp).
- Poor robustness of Ooqp $\Longrightarrow$ careful implementation yields much improvement?
- Oqla is located between Qpb and Ooqp in terms of robustness.

Behaviors in an SQP framework

- Recall that one iteration of the SQP algorithm computes a PD solution $(d^{\mathrm{QP}}, \lambda^{\mathrm{QP}})$ of the OQP
$$\min_{l' \leqslant Ad \leqslant u'} \left(g^{\mathsf{T}} d + \frac{1}{2} d^{\mathsf{T}} H d \right)$$
and then updates (locally) the PD variables (x, λ) by
$$x_+ := x + d^{\mathrm{QP}} \qquad \text{and} \qquad \lambda_+ := \lambda^{\mathrm{QP}}.$$
- Close to the solution to the nonlinear problem, $x_+ \simeq x$ and $\lambda_+ \simeq \lambda$, therefore a good guess of the PD solution to the QP is available:
$$(0, \lambda).$$
- Hence, it makes sense to see how the QP solvers behave when the starting point is close to the solution to the QP.

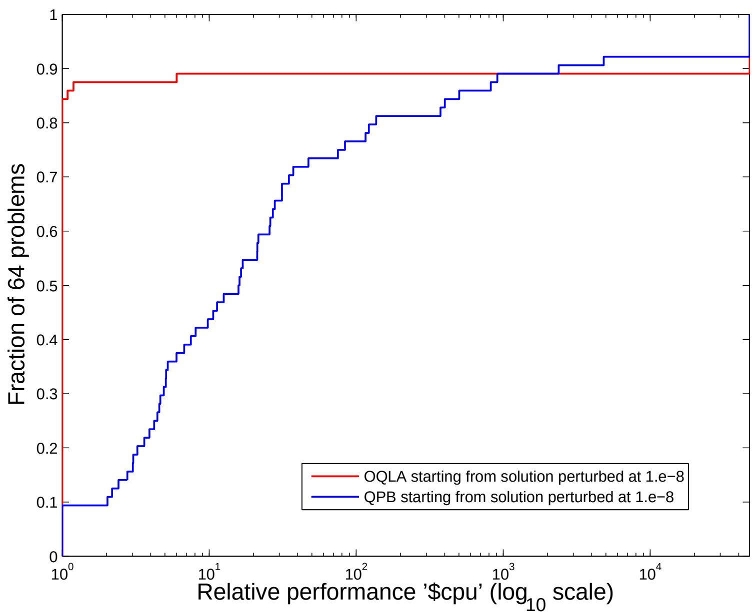
Oqla vs. Qpb, starting from a primal-dual solution, on CPU time



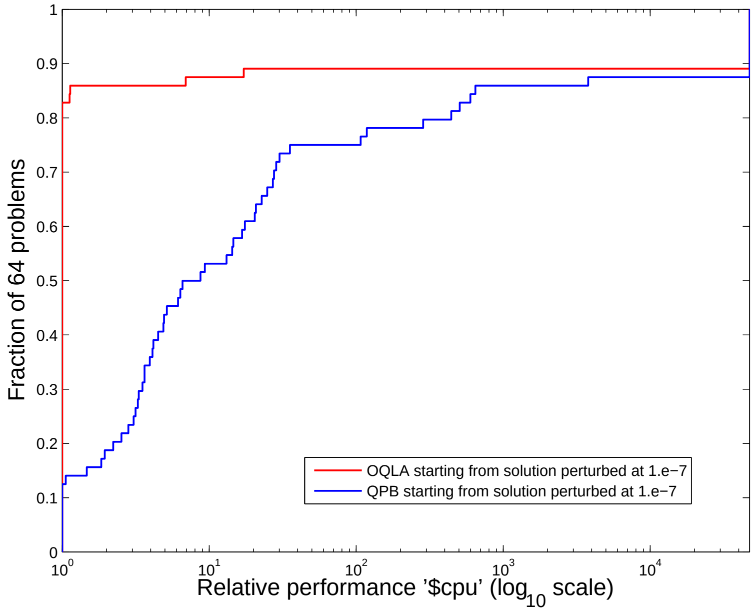
- Motivation: see whether Oqla can take advantage of a good starting point,
- 64 problems, for which an accurate primal-dual solution has been found,
- Qpb has no warm restart.

Numerical results
Comparison with interior-point methods

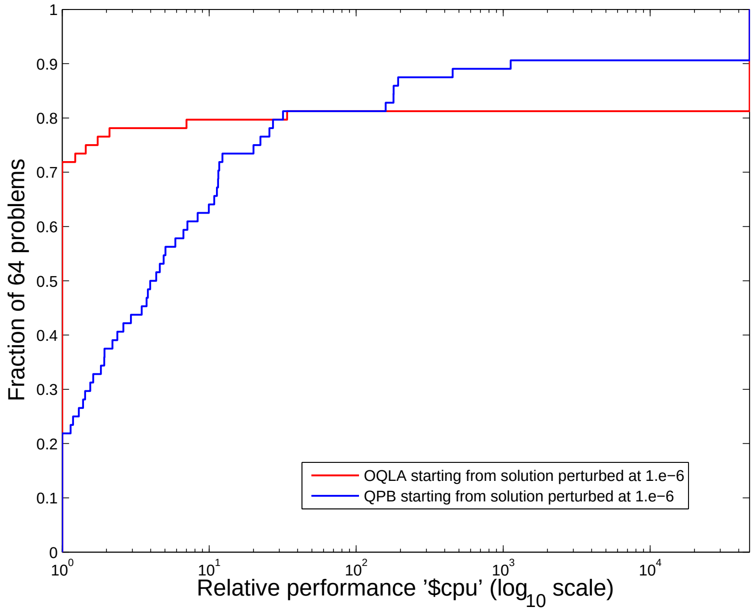
Oqla vs. Qpb, starting from a perturbed (10^{-8}) primal-dual solution



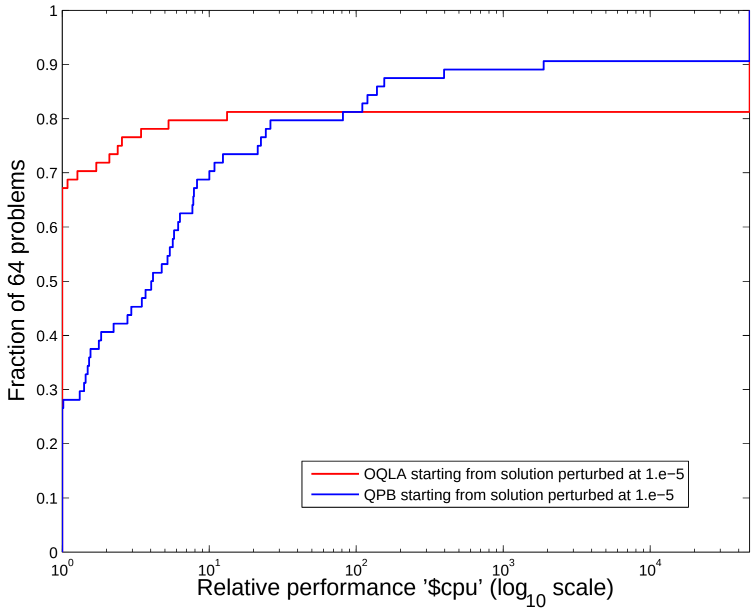
Oqla vs. Qpb, starting from a perturbed (10^{-7}) primal-dual solution



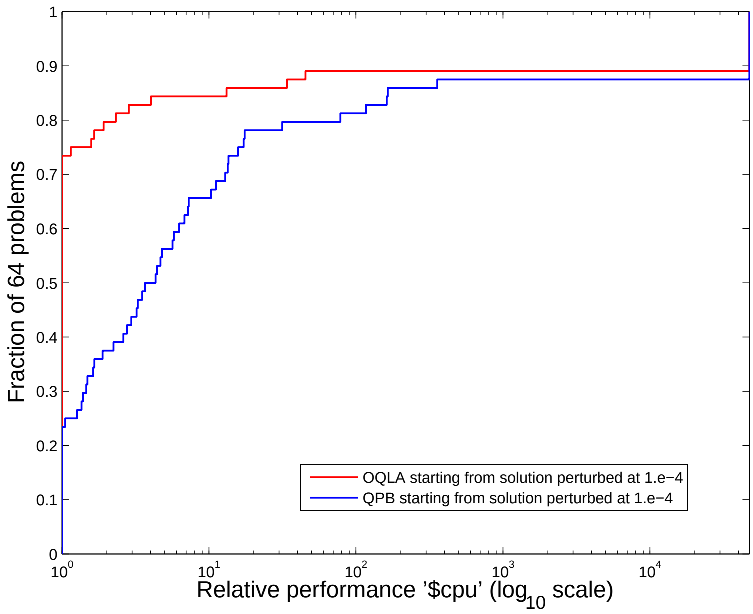
Oqla vs. Qpb, starting from a perturbed (10^{-6}) primal-dual solution



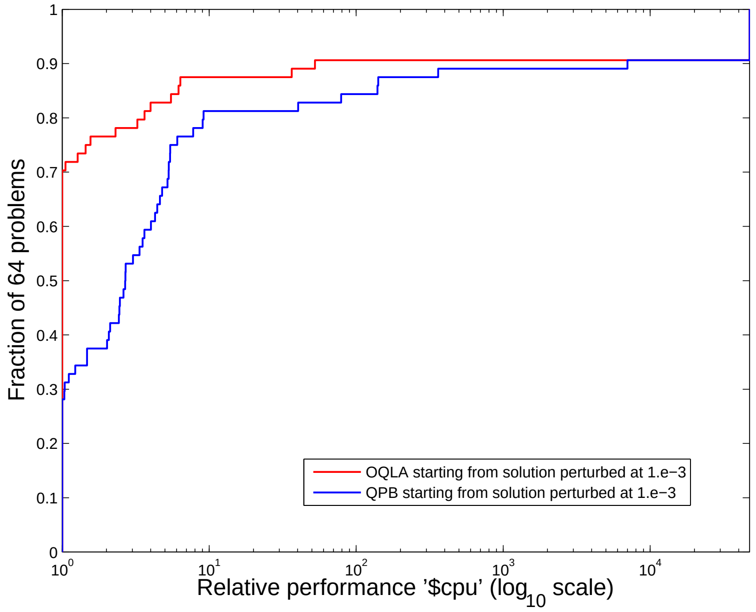
Oqla vs. Qpb, starting from a perturbed (10^{-5}) primal-dual solution



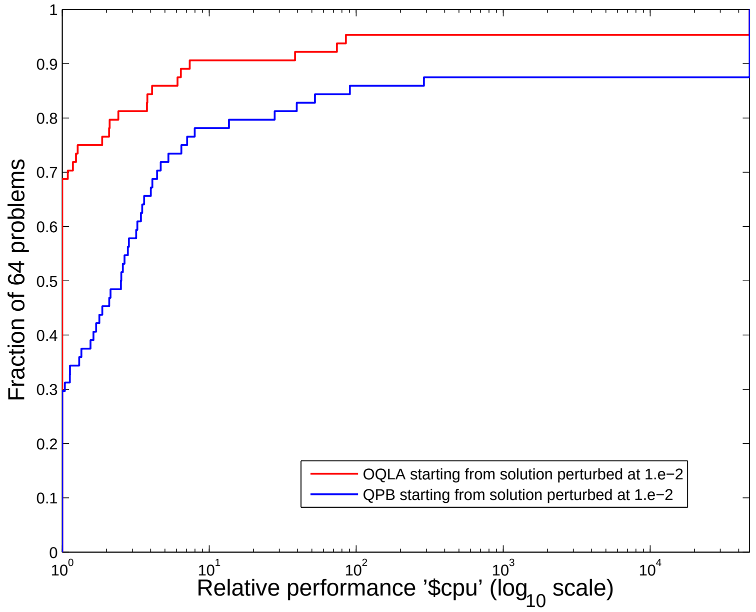
Oqla vs. Qpb, starting from a perturbed (10^{-4}) primal-dual solution



Oqla vs. Qpb, starting from a perturbed (10^{-3}) primal-dual solution



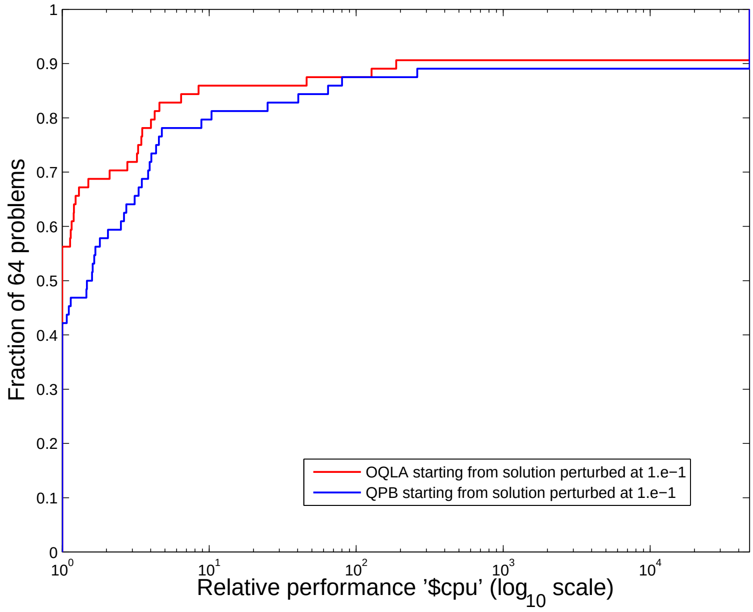
Oqla vs. Qpb, starting from a perturbed (10^{-2}) primal-dual solution



Numerical results

Comparison with interior-point methods

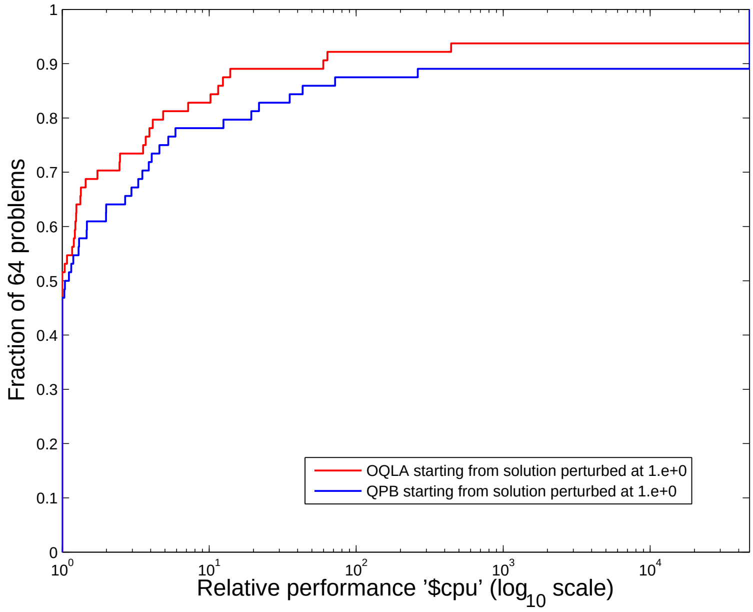
Oqla vs. Qpb, starting from a perturbed (10^{-1}) primal-dual solution



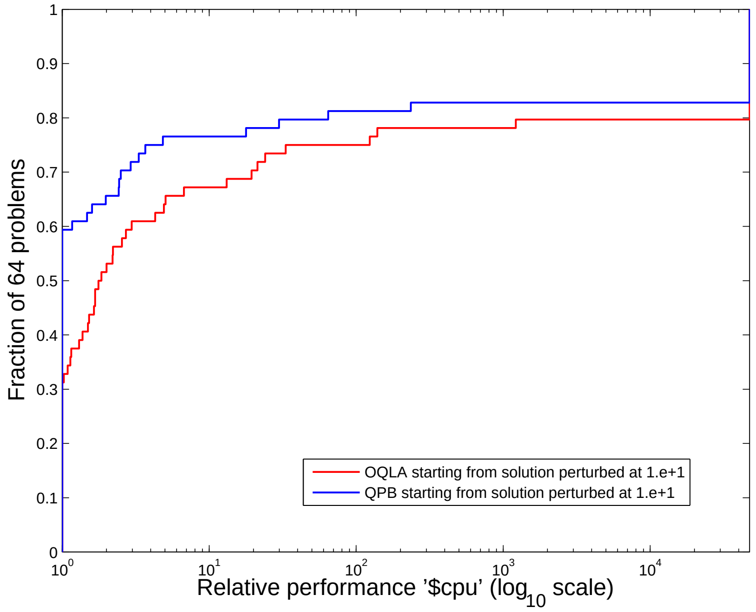
Numerical results

Comparison with interior-point methods

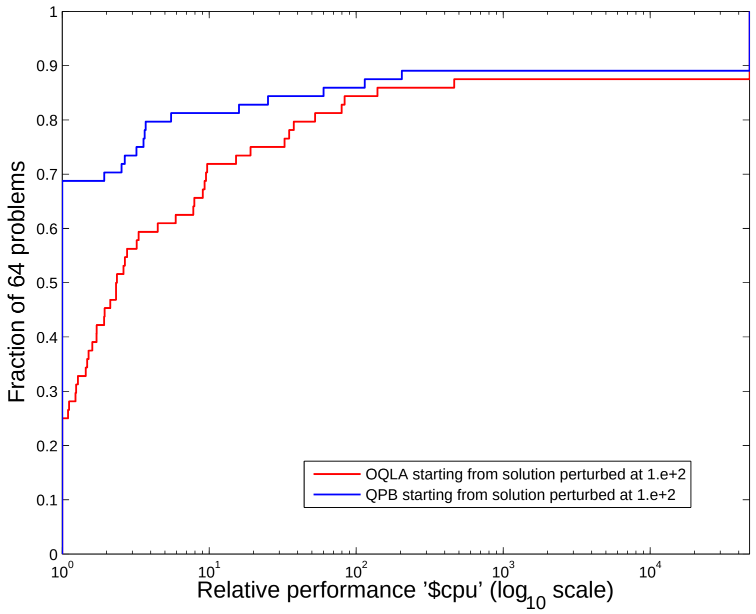
Oqla vs. Qpb, starting from a perturbed (10^{0}) primal-dual solution



Oqla vs. Qpb, starting from a perturbed (10^1) primal-dual solution



Oqla vs. Qpb, starting from a perturbed (10^2) primal-dual solution



- Conclusion: for perturbations less than 100 %, the AL+AS solver Oqla is "more often better" than the IP solver Qpb.

Outline

Discussion and future work

Discussion

- Oqla/Qpalm give interesting answers on infeasbile or unbounded QPs.
- Oqla and Qpalm are not ridiculous, with respect to well established active-set solvers (Qpa), and sometimes clearly better (Oqla).
- The present version of Oqla/Qpalm is not as efficient as the IP solver Qpb, but much more robust than Ooqp.
- Oqla/Qpalm can take advantage of an estimate of the solution (not the case of the other tested IP solvers) $\Longrightarrow$ nice for SQP.
- Still many possible improvements:
  - using preprocessing,
  - inexact minimization of the AL subproblems (2), while keeping the global linear convergence,
  - trying other solvers of the AL subproblems (2), like IP or Newton-min,
  - . . . .

Future work

- Can one preserve the global linear convergence of the AL algorithm when the AL subproblems (2) are solved inexactly?
- Try to use one (a few) interior point step(s) to solve the AL subproblems (2), in order to obtain polynomiality.
- Improve nonsmooth methods and use them to solve the AL subproblems (2), in order to gain in efficiency.
- Extend the result of Dean and Glowinski [5] to convex inequality constrained QP: for *stricty* convex QP with the *single equality constraint* $Ax = b$, the Lagrangian relaxation

$$\begin{aligned} x_k &= \arg\min_{x\in\mathbb{R}^n} \; q(x) + \lambda_k^{\mathsf{T}}(Ax-b) \\ \lambda_{k+1} &= \lambda_k + \alpha_k(Ax_k - b), \end{aligned}$$

  where α_k is chosen is a compact of $]0, 2/\mu_1[$, generates iterates that converge globally linearly to the unique solution to the closest feasible problem

$$\left\{ \begin{array}{l} \inf_x \; q(x) \\ A^{\mathsf{T}}(Ax-b) = 0. \end{array} \right.$$

Future work (continued)

- Show the global linear convergence of an AL algorithm for the more general problem (+ constraint qualification):

$$\left\{ \begin{array}{l} \inf_{x\in\mathbb{E}} \; \langle g, x\rangle + \frac{1}{2}\langle Hx, x\rangle \\ Ax \in C \\ x \in X. \end{array} \right.$$

  Two interesting instances:
  - $\mathbb{E} = \mathbb{R}^n$, $C = [l, u]$, $X =$ ball $\Longrightarrow$ trust region problem,
  - $\mathbb{E} = \mathcal{S}^n$, $H = 0$, $C = \{b\}$, $X = \mathcal{S}^n_+$ $\Longrightarrow$ linear SDP problem.

The end

Thank you very much for your attention!

Main references

- F. Delbos and J.Ch. Gilbert (2005). Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems. *Journal of Convex Analysis*, 12, 45-69.
- A. Chiche, J.Ch. Gilbert (2016). How the augmented Lagrangian algorithm can deal with an infeasible convex quadratic optimization problem. *Journal of Convex Analysis*, 23(2), 425-459.
- J.Ch. Gilbert, É. Joannopoulos (2016). OQLA/QPALM - Convex quadratic optimization solvers using the augmented Lagrangian approach, able to deal with infeasibility and unboundedness. Technical report, INRIA, BP 105, 78153 Le Chesnay, France. (To appear)

K.J. Arrow, F.J. Gould, S.M. Howe (1973).
A generalized saddle point result for constrained optimization.
Mathematical Programming, 5, 225–234.
[doi].

E.G. Birgin, J.M. Martínez (2014).
Practical Augmented Lagrangian Methods for Constrained Optimization.
SIAM Publication, Philadelphia.
[doi].

J.D. Buys (1972).
Dual algorithms for constrained optimization.
PhD Thesis, Rijksuniversiteit te Leiden, Leiden, The Netherlands.

A. Chiche, J.Ch. Gilbert (2016).
How the augmented Lagrangian algorithm can deal with an infeasible convex quadratic optimization problem.
Journal of Convex Analysis, 23(2), 425–459.
[pdf].

E.J. Dean, R. Glowinski (2006).
An augmented Lagrangian approach to the numerical solution of the Dirichlet problem for the elliptic Monge-Ampère equation in two dimensions.
Electronic Transactions on Numerical Analysis, 22, 71–96.
[pdf].

F. Delbos, J.Ch. Gilbert (2005).
Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems.
Journal of Convex Analysis, 12, 45–69.
[preprint] [editor].

E.D. Dolan, J.J. Moré (2002).
Benchmarking optimization software with performance profiles.
Mathematical Programming, 91, 201–213.
[doi].

M. Frank, P. Wolfe (1956).
An algorithm for quadratic programming.
Naval Research Logistics Quarterly, 3, 95–110.

65 / 65

[doi].

E.M. Gertz, S. Wright (2003).
Object-oriented software for quadratic programming.
ACM Transactions on Mathematical Software, 29, 58–81.
[doi].

J.Ch. Gilbert, É. Joannopoulos (2015).
OQLA/QPALM - Convex quadratic optimization solvers using the augmented Lagrangian approach, with an appropriate behavior on infeasible or unbounded problems.
Research report, INRIA, BP 105, 78153 Le Chesnay, France.
(to appear).

J.Ch. Gilbert, X. Jonsson (2008).
LIBOPT – An environment for testing solvers on heterogeneous collections of problems.
Submitted to *ACM Transactions on Mathematical Software*.

N.I.M. Gould, D. Orban, Ph.L. Toint (2003).
GALAHAD: a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization.
Technical report, Rutherford Appleton Laboratory, Oxfordshire OX11 0QX.
[internet].

N.I.M. Gould, D. Orban, Ph.L. Toint (2013).
Cutest: a constrained and unconstrained testing environment with safe threads.
Technical report, Rutherford Appleton Laboratory, Didcot, Oxfordshire OX11 0QX.

M.R. Hestenes (1969).
Multiplier and gradient methods.
Journal of Optimization Theory and Applications, 4, 303–320.

M.J.D. Powell (1969).
A method for nonlinear constraints in minimization problems.
In R. Fletcher (editor), *Optimization*, pages 283–298. Academic Press, London.

R.T. Rockafellar (1971).
New applications of duality in convex programming.
In *Proceedings of the 4th Conference of Probability, Brasov, Romania*, pages 73–81.

R.T. Rockafellar (1973).
A dual approach to solving nonlinear programming problems by unconstrained optimization.
Mathematical Programming, 5, 354–373.
[doi].

R.T. Rockafellar (1973).
The multiplier method of Hestenes and Powell applied to convex programming.
Journal of Optimization Theory and Applications, 12, 555–562.
[doi].

R.T. Rockafellar (1974).
Augmented Lagrange multiplier functions and duality in nonconvex programming.
SIAM Journal on Control, 12, 268–285.

Quadprog (2014).
Quadratic programming.
[internet].

J.E. Spingarn (1987).
A projection method for least-squares solutions to overdetermined systems of linear inequalities.
Linear Algebra and its Applications, 86, 211–236.
[doi].