A Moving Balls Approximation Method for Smooth Constrained Minimization

Marc Teboulle

School of Mathematical Sciences
Tel Aviv University

Joint work with Alfred Auslender, Paris/Lyon and Ron Shefi, Tel Aviv

Modern Trends in Optimization and Its Application

Workshop II: Numerical Methods for Continuous Optimization, October 11-15, 2010
IPAM - Institute for Pure and Applied Mathematics, UCLA, Los Angeles

Constrained Minimization

$$(P) \quad f_* := \inf\{f(x) : \ f_i(x) \leq 0, \ i = 1, \ldots, m, x \in \mathbb{R}^n\}.$$

- No *universal method* for solving (P).
- Different Data/Structures imply/require different algorithms.
- Interest is for $n \times m$ large. Thus we would like a method:
  ♣ requiring simple computational steps
  ♣ capable to produce rapidly optimal solutions within reasonable accuracy.
- **Early works for general NLP:**
  ♢ Linearization methods: (Zoutendjick (1960), Pschenichny (1978)...
  ♢ More popular: Quadratic approx. of objective and Linearized constraints: SQP methods.
- Here, we depart from *general* NLP. We focus on problems with **smooth data** and exploit this to build an **elementary and provably convergent efficient scheme** for solving (P).

Problem and Basic Assumptions

$$(P) \quad f_* := \inf\{f(x) : \ x \in \mathcal{F}\}, \ \ \mathcal{F} := \{x \in \mathbb{R}^n : \ f_i(x) \leq 0, \ i = 1, \dots m\}.$$

- **A1** $f, f_i : \mathbb{R}^n \to \mathbb{R}, \ i = 1 \dots m$ are differentiable functions with Lipschitz gradient with constants L_f, L_i, (Notation: $C_L^{1,1}$):

  $$\|\nabla f(x) - \nabla f(y)\| \leq L_f \|x - y\|, \ \|\nabla f_i(x) - \nabla f_i(y)\| \leq L_i \|x - y\|, \ \forall x, y \in \mathbb{R}^n.$$

- **A2** Problem (P) is feasible and

  $$\exists x^0 \in \mathcal{F} : \ C := \mathcal{F} \cap \{x : \ f(x) \leq f(x^0)\} \ \text{ is bounded.}$$

- **A3** The Mangasarian-Fromovitz CQ holds :

  $$\forall x \in \mathcal{F} : \ \exists d \in \mathbb{R}^n : \ \langle \nabla f_i(x), d \rangle < 0 \ \forall i \in I(x) := \{i \in [1, m] : \ f_i(x) = 0\}.$$

A2, A3 are standard assumptions in NLP and **A1** is for **smooth** NLP.

Objective

- We want to compute points satisfying first order necessary optimality conditions for problem (P), i.e., points in S:

$$S := \{x \in \mathcal{F} : \exists\, \lambda_i \geq 0 \text{ with } \nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x) = 0,\ \lambda_i f_i(x) = 0,\ \forall i\}$$

We would like an *elementary scheme*, suitable for large scale problems:

- Requires simple computational operations.
- Efficient for solving (P) to reasonable accuracy.
- We achieve this through a "**cocktail**" based on:
  - ♣ **a simple geometric idea**
  - ♣ **duality**
  - ♣ **an optimal gradient scheme**
  - ♣ **a simple active set technique.**

Main Approximation Tool

Main idea: Exploit the *smoothness* of the problem's data with functions in the class $C_L^{1,1}$, to:
- approximate the objective function
- construct "balls" that iteratively approximate the feasible set.

Use basic well known property of a $C_L^{1,1}$ function.

Lemma Let $g : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function with global Lipschitz gradient, and Lipschitz constant L_g. Then,

$$g(y) \leq g(x) + \langle y - x, \nabla g(x) \rangle + \frac{L_g}{2} \|x - y\|^2, \quad \forall x, y.$$

Geometrically, means that a $C^{1,1}$ function is approximated from above by a nice convex *quadratic*.
[Actually, also true from below, but with a *concave* quadratic.]

The Approximate Model

Consider the following approximations for (P). For each $x \in \mathcal{F}$, define:

$$h(x,y) := f(x) + \langle \nabla f(x), y - x \rangle + \frac{L_f}{2}||x-y||^2$$

$$C(x) := \{y \in \mathbb{R}^n : \ f_i(x) + \langle \nabla f_i(x), y - x \rangle + \frac{L_i}{2}||x-y||^2 \leq 0, \ i \in [1,m]\}.$$

- For any x, the function $y \to h(x,\cdot)$ is L_f-*strongly convex*
- $\exists$ a unique point $p(x)$ solving the strongly convex problem $(P(x))$:

For each $x \in \mathcal{F}$: $(P(x))$ $\qquad p(x) = \text{argmin}\{h(x,y) : \ y \in C(x)\}$

The next results will justify the use of $(P(x))$ as a natural approximation of the original problem (P).

Basic Properties of Problem $(P(x))$

$(P(x))$ $\quad p(x) = \text{argmin}\{h(x,y) : \ y \in C(x)\}$

Proposition 1 For any $x \in \mathcal{F}$
(i) $C(x)$ is a nonempty convex compact set.
(ii) $C(x) \subset \mathcal{F}$
(iii) Slater's condition holds for the constraints set $C(x)$.

Proposition 2 $\quad \mathbf{x} \in \mathbf{S}$ if and only if $\mathbf{x} = \mathbf{p}(\mathbf{x})$

Proposition 3 [Useful for Convergence Analysis]
(i) $p(C) \subset C$, i.e., the image of C under the map $p(\cdot)$ remains in C.
(ii) $p(\cdot)$ is continuous on C.

The Moving Balls Approximation Method- MBA

We have $x \in S \iff x = p(x)$.

Naturally calls for the following basic fixed point algorithm to solve (P), called the Moving Balls Approximation **MBA** algorithm.

MBA Algorithm Generate the sequence $\{x_k\}$ via:

$$x^0 \in \mathcal{F},\ x^k = p(x^{k-1}) = \operatorname{argmin}\{h(x^{k-1}, y) :\ y \in C(x^{k-1})\},\ k \geq 1.$$

Theorem

Let $\{x^k\}$ be the sequence generated by **MBA**. *Then,*
(i) the sequence of function values $\{f(x^k)\}$ is monotone nonincreasing;
(ii) the sequence $\{x^k\}$ is bounded, and any limit point of $\{x^k\}$ is in S.

Reformulation of $p(x) = \mathbf{argmin}\{h(x,y) : y \in C(x)\}$

Simple algebra shows that

$$p(x) = \Pi_{C(x)}\left(x - \frac{1}{L_f}\nabla f(x)\right), \qquad \Pi_{C(x)} \text{ projection onto } \; C(x)$$

$$\begin{aligned}
C(x) &:= \bigcap_{i=1}^{m} B_i(x) \text{ where the balls } B_i(\cdot) \text{ are given by,} \\
B_i(x) &= \{y \in \mathbb{R}^n : \|y - c_i(x)\|^2 \leq \rho_i^2(x)\}, \\
c_i(x) &:= x - \frac{1}{L_i}\nabla f_i(x), \text{ (center of the ball } B_i(x)) \\
\rho_i^2(x) &:= \frac{1}{L_i^2}\|\nabla f_i(x)\|^2 - \frac{2}{L_i} f_i(x), \text{ (squared radius of } B_i(x)).
\end{aligned}$$

How to efficiently compute the projection $p(x)$?

Computing $p(x)$ via Dual of $(P(x))$

$$\begin{aligned}
\text{Let } x \in \mathcal{F} \text{ and } s_0(y,x) &:= \langle \nabla f(x), y-x\rangle + \frac{L_f}{2}\|x-y\|^2, \\
s_i(y,x) &:= f_i(x) + \langle \nabla f_i(x), y-x\rangle + \frac{L_i}{2}\|x-y\|^2,\ i \in [1,m] \\
L_x(y,u) &:= s_0(y,x) + \sum_{i=1}^m u_i s_i(y,x), \text{ Lagrangian for } (P(x)).
\end{aligned}$$

The dual problem to $P(x)$ is then defined by

$$D(x): \quad \max\{d(u,x):\ u \in \mathbb{R}^m_+\} \text{ with } d(u,x) := \inf_y L_x(y,u).$$

A simple computation shows that a Lagrangian dual problem $D(x)$ to $(P(x))$ can be written as the convex minimization problem in $\mathbb{R}^m_+$:

$$(D(x)) \qquad q(x) := \inf_{u\in\mathbb{R}^m_+} \left\{ \sum_{i=1}^m \frac{\|\nabla f(x) + \sum_{i=1}^m u_i \nabla f_i(x)\|^2}{2(L_f + \sum_{i=1}^m L_i u_i)} - \sum_{i=1}^m u_i f_i(x) \right\}.$$

Primal-Dual Relations for the pair $(P(x)), (D(x))$

Proposition Let $x \in \mathcal{F}$. Then, strong duality holds for the pair of problems $(P(x)), (D(x))$ i.e.,

$$m(x) + q(x) = 0,$$

where $m(x), q(x)$ are the optimal value of $(P(x)), (D(x))$ respectively.

Moreover, if $u(x)$ solves $(D(x))$ then the optimal solution $p(x)$ of $P(x)$ is

$$p(x) = x - \frac{\nabla f(x) + \sum_{i=1}^{m} u_i(x) \nabla f_i(x)}{(L_f + \sum_{i=1}^{m} L_i u_i(x))}$$

- **Thus, basic step of MBA: $p(x)$ is given by an explicit formula.**
- **Looking for a simple and efficient algorithm to solve $D(x)$.**

Extension to Variational Inequalities

MBA can be easily extended to solve a class of variational inequalities with convex constraints described by $\mathcal{F}$.

- Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a monotone map on $\mathcal{F}$. The (VI) problem is:

$$(VI) \quad \text{Find } x^* \in \mathcal{F}, \text{ such that } \langle F(x^*), x - x^* \rangle \geq 0, \quad \forall x \in \mathcal{F}.$$

- Suppose that F is co-coercive on $\mathcal{F}$ i.e.

$$\exists c > 0 : \ \langle x - y, F(x) - F(y) \rangle \geq c \|F(x) - F(y)\|^2 \quad \forall x, y \in \mathcal{F}.$$

- Co-coercivity corresponds to f being a smooth convex function with Lipschitz gradient L_f, when replacing $\nabla f(x)$ with $F(x)$, and setting $L_f := 1/c$. Then, we can simply re-define for (VI) the map p by:

$$p(x) := \text{argmin}\{h(x, y) = \langle F(x), y - x \rangle + \frac{1}{2c} ||x - y||^2 : \ y \in C(x)\}$$

Convergence for variational inequalities

The sequence generated by **MBA** for solving (VI) then simply consists of

$$x^k = \text{argmin}\{\langle F(x^{k-1}), y - x^{k-1}\rangle + \frac{1}{2c}\|y - x^{k-1}\|^2 : \ y \in C(x^{k-1})\}, \ k \geq 1.$$

Theorem Let $\{x^k\}$ be the sequence generated by **MBA**, for problem (VI) with F assumed co-coercive. Then,

(i) the whole sequence $\{x^k\}$ converges to a solution of (VI).

(ii) Furthermore, if F is assumed strongly monotone, the rate of convergence is linear.

Similarly, we can establish more convergence results in the convex case, see [AST].

Back to Solve the Convex Dual $(D(x))$

- Dual Problem of $(P(x))$

$$(D(x)) \qquad q(x) := \inf_{u \in \mathbb{R}^m_+} \left\{ \sum_{i=1}^m \frac{\|\nabla f(x) + \sum_{i=1}^m u_i \nabla f_i(x)\|^2}{2(L_f + \sum_{i=1}^m L_i u_i)} - \sum_{i=1}^m u_i f_i(x) \right\}.$$

- Particular case of the convex minimization problem:
  (Conv) $\quad c_* := \inf\{c(u) : u \in U\}$, $c : \mathbb{R}^m \to \mathbb{R}$ convex and C^1, $U \subset \mathbb{R}^m$ closed convex.
- Many algorithms can be used: e.g., Interior point, gradient projection, etc..
- We want a simple and efficient scheme. Gradient projection methods are natural candidates, but are **slow**. The number of iterations k necessary to produce an approximation within accuracy ϵ i.e., such that

$$c(u^k) - c_* \leq \epsilon$$

  is of the order $O(\epsilon^{-1})$.

Optimal Projected Gradient

$$\text{(Conv)} \qquad c_* := \inf\{c(u) : u \in U\}, U \subset \mathbb{R}^m \text{ closed convex}$$

But with $c(\cdot)$ convex **and** $C^{1,1}$, i.e., with L_c- Lipschitz gradient.

♠ **(OPG) introduced by Nesterov (83). For a $C^{1,1}$ convex $c(\cdot)$,** produces

$$g(u^k) - g_* \simeq O(1/k^2).$$

i.e., faster by a **square root factor** versus the usual projected gradient, and yet equally simple.

Algorithm 1 Optimal Projected Gradient

Require: $y^1 = u^0 \in U, \nabla c(u), L_c$(can be iteratively computed).

1: $k \leftarrow 1,\ \alpha_k \leftarrow 1$
2: **while** stopping criteria **do**
3: $\quad u^k = \Pi_U(y^k - \frac{1}{L_c}\nabla c(y^k)),\ \alpha_{k+1} = (1 + \sqrt{1 + 4\alpha_k^2})/2$
4: $\quad y^{k+1} = u^k + \frac{\alpha_k - 1}{\alpha_{k+1}}(u^k - u^{k-1})$
5: **end while**
6: $u^* \leftarrow u^k$

Dual objective of $(D(x))$

- For any $x \in \mathcal{F}$, define:

$$\begin{aligned} A &= [\nabla f_1(x), \ldots, \nabla f_m(x)] \in \mathbb{R}^{n \times m}, \\ b &= -\nabla f(x), c := (-f_1(x), \ldots - f_m(x))^T, \\ d &= (L_1, \ldots, L_m)^T \end{aligned}$$

- Our dual objective is the sum of a *convex fraction* with a linear term:

$$g(u) = \frac{\|Au - b\|^2}{2L_f(1 + d^T u)} + c^T u, \qquad 1 + d^T u > 0 \; \forall u \in \mathbb{R}^m_+.$$

- Does not look encouraging...
- **Remarkably, the dual function $g(\cdot)$ has a gradient which is globally Lipschitz on $\mathbb{R}^m_+$**

The Lipshitz constant L_g is also explicitly given in terms of problem's data.

Global Lipschitz Gradient Property of Dual

In fact, this a consequence of a slightly more general result.

- Let $Q \in \mathbb{R}^{m\times m}$ be a symmetric matrix, $f \in \mathbb{R}^m, \gamma \in \mathbb{R}$ and $r \in \mathbb{R}^m_{++}$
- Let, $q(u) := \frac{1}{2}u^T Qu + f^T u + \gamma,\ l(u) := 1 + r^T u$
- For all $u \in \mathbb{R}^m_+$ consider the ratio

$$(*) \quad R : \mathbb{R}^m_+ \to \mathbb{R}, \ \ R(u) = \frac{q(u)}{l(u)}.$$

Clearly, our dual $g(\cdot)$ is a special case of $R(u)$.

Lemma Let $R : \mathbb{R}^m_+ \to \mathbb{R}$ be given by (*). Then, $R \in C^{1,1}_M(\mathbb{R}^m_+)$ with

$$M = \left(1 + \frac{\|r\|}{r_{\min}}\right)^2 \|Q\| + 2\left(1 + \frac{\|r\|}{r_{min}}\right)\|f\|\|r\| + 2|\gamma|\|r\|^2,$$

where $r_{\min} := \min_{1\le i\le m} r_i > 0$.

Solving Dual $(D(x))$ via OPG

Therefore, **OPG** can be applied to solve $(D(x))$ and we get the following iteration complexity bound.

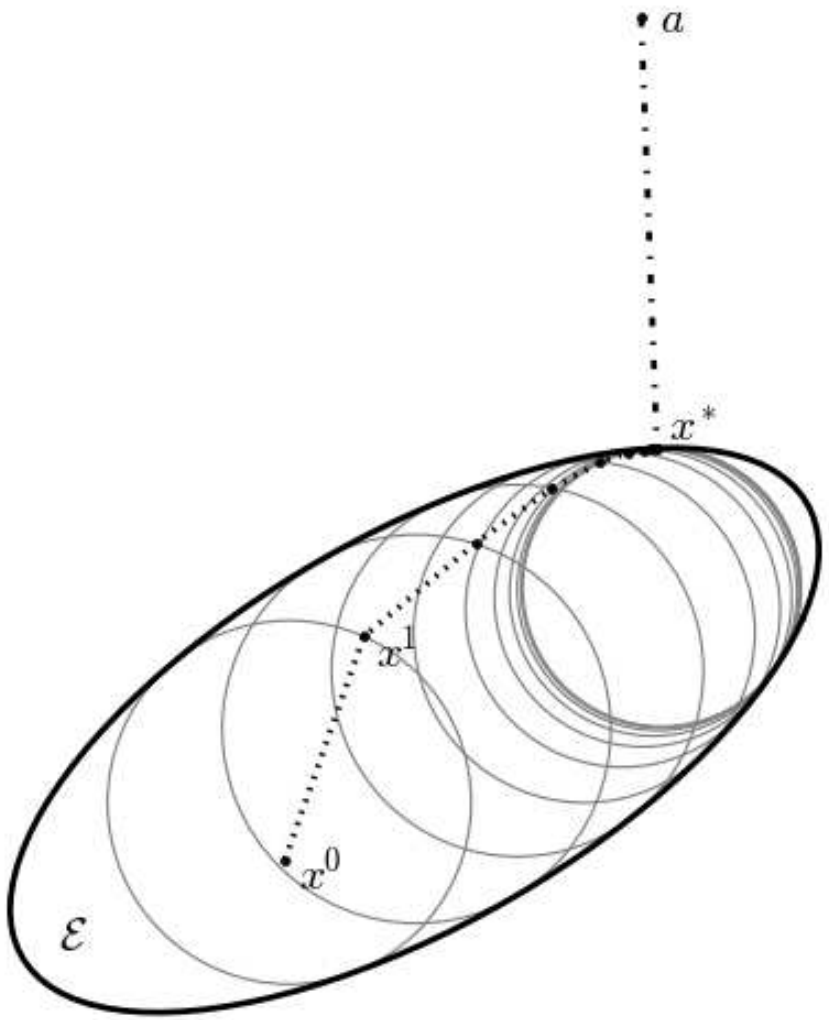
Theorem An ϵ-global optimal solution for $(D(x))$ satisfying $g(u^k) - g_* \leq \epsilon$ is reached in $O(\sqrt{\frac{L_g}{\epsilon}})$ iterations.

MBA is Simple – Does This Really Work?

- Solution of dual via OPG gives a simple and efficient scheme.
- We have a simple formula for $x^k = p(x^{k-1})$ via primal-dual relation.
- Does this really work?

- For large n: **Yes** and quite well as we shall see soon.
- For large m: **Not as good...:** a large number of balls is necessary to approximate $\mathcal{F}$....and the situation deteriorates....
- For instance, SQP performs much better than MBA when number of inequality constraints m grows significantly....This was not observed when increasing the dimension n of the variables...

Too many balls..?...



Projection of a onto an ellipsoid $\mathcal{E}$.

Do we really need all these balls?!...

.....Last Cocktail Ingredient ....

- This difficulty motivated us to consider a modification of **MBA** by introducing **an active set technique.**
- Essentially, it allows to **reduce drastically the number of balls** necessary to approximate the constraints set $\mathcal{F}$.
- The special structure of our problem allows to devise such a scheme which is as simple as **MBA**, and yet **significantly faster** for problems with a large number of constraints.

Some Notations/Definitons for MBA-AS

Let $\langle 1, m\rangle$ be the set of integers included in $[1, m]$.
Let I be a subset of $\langle 1, m\rangle$. For each $x \in \mathcal{F},\ \epsilon \geq 0$, set

$$C(x, I) := \begin{cases} \mathbb{R}^n, & \text{if } I = \emptyset; \\ \cap_{i \in I} B_i(x) & \text{otherwise} \end{cases}$$

$$\begin{aligned} p(x, I) &= \operatorname{argmin}\{f(x) + \langle \nabla f(x), y - x\rangle + \frac{L_f}{2}||x - y||^2 :\ y \in C(x, I)\} \\ I(x, \epsilon) &= \{i :\ -\epsilon \leq f_i(x) \leq 0\}, \quad I^c(x, \epsilon) = \{i :\ f_i(x) < -\epsilon\} \end{aligned}$$

When $I = I(x, \epsilon)$ we denote:

$$\begin{aligned} C(x, \epsilon) = C(x, I), \qquad & p(x, \epsilon) = p(x, I), \\ d(x, \epsilon) = p(x, \epsilon) - x, \qquad & \sigma(x, \epsilon) = \|d(x, \epsilon)\|. \end{aligned}$$

Key Result for MBA-AS

The special structure of our problem allows for easy formula and for establishing the key result.

Lemma Let $x \in \mathcal{F},\ \epsilon \geq 0$, and $x_\epsilon = x + \alpha(x,\epsilon)d(x,\epsilon)$. Then $x_\epsilon \in \mathcal{F}$ and we have,

$$\alpha(x,\epsilon) \geq \min\{\frac{\epsilon}{L\sigma(x,\epsilon)^2 + a(x,\epsilon)}, 1\}.$$

Notations/Defs for above: For $i \in \langle 1, m\rangle$ set:

$$\begin{aligned} a_i(x,\epsilon) &= \langle \nabla f_i(x), d(x,\epsilon)\rangle, \qquad a(x,\epsilon) = \max\{|a_i(x,\epsilon)| : 1 \leq i \leq m\}, \\ \alpha_i(x,\epsilon) &= \frac{-a_i(x,\epsilon) + \sqrt{a_i(x,\epsilon)^2 - 2L_i f_i(x)\sigma(x,\epsilon)^2}}{L_i\sigma(x,\epsilon)^2}. \end{aligned}$$

We set $L := \max\{L_i | i = 1,\ldots,m\}$ and

$$\alpha(x,\epsilon) := \begin{cases} 1, & \text{if } I^c(x,\epsilon) = \emptyset; \\ \min\{1, \min\{\alpha_i(x,\epsilon) :\ i \in I^c(x,\epsilon)\}\} & \text{otherwise} \end{cases}$$

Moving Balls Method with an Active Set Technique

Algorithm 2 MBA-AS

Require: $x_0 \in \mathcal{F}, f, f_i, g = \nabla f, g_i = \nabla f_i, L_f, L = \max\{L_1, \ldots, L_m, \}, \epsilon_0 > 0, \eta \in]0, 1[$.

1: $k \leftarrow 0$
2: **while** stopping criteria **do**
3: $c_i(x^k) \leftarrow x^k - g_i(x^k)/L_i$ {centers}
4: $\rho_i(x^k)^2 \leftarrow \frac{1}{L^2}||g_i(x^k)||^2 - \frac{2}{L_i} f_i(x^k)$ {radii}
5: $I(x^k, \epsilon_k),\ C(x^k, \epsilon_k) \leftarrow \cap_{I(x^k,\epsilon_k)} B_i(c_i(x^k), \rho_i(x^k))$
6: $p(x^k, \epsilon_k) \leftarrow \Pi_{C(x^k,\epsilon_k)}(x^k - g(x^k)/L_f)$
7: $\alpha_i(x^k, \epsilon_k)\ \forall i \in I^c(x^k, \epsilon_k)$
8: $\alpha(x^k, \epsilon_k) = \min\{1, \min\{\alpha_i(x^k, \epsilon_k) :\ i \in I^c(x^k, \epsilon_k)\}\}$
9: $x^{k+1} = x_{\epsilon_k} = x^k + \alpha(x^k, \epsilon_k)(p(x^k, \epsilon_k) - x^k)$
10: If $\sigma(x^k, \epsilon_k) := \|p(x^k, \epsilon_k) - x^k\| \leq \epsilon_k$ and $\alpha(x^k, \epsilon_k) = 1$, then $\epsilon_{k+1} = \eta\epsilon_k$; else $\epsilon_{k+1} = \epsilon_k$. $k \leftarrow k + 1$
11: **end while**
12: $x^* \leftarrow x^k$

As simple as MBA – Additional steps 7, 8 admit explicit formulas

Convergence of MBA-AS

MBA-AS shares the same convergence properties of **MBA**.

Theorem

Let $\{x^k\}$ *be the sequence generated by* **MBA-AS**. *Then,*
(i) the sequence $\{x^k\}$ *is feasible,*
(ii) the sequence of function values $\{f(x^k)\}$ *is monotone nonincreasing,*
(iii) the sequence $\{x^k\}$ *is bounded, and any limit point of* $\{x^k\}$ *is in* S.

Some Numerical Experiments

- **For convex quadratically constrained problems**
- Randomly generated *fully dense* data matrices with various condition numbers $\{10, 10^2, 10^3\}$
- Dimension ranges from $n = 50$ to $n = 1000$ for variables and
- from $m = 50$ to $m = 2000$ for the number of inequality constraints
- Our results are compared to 2 well known solvers:
  (i) CVXOPT a quadratic cone programming solver developed by Dahl and Vandenberghe
  (ii) An SQP solver from IMSL Library.
- All experiments performed on a very basic Laptop with 2GB internal memory

CVXOPT vs. MBA-AS – Fixed No. Var. $n = 50$

		CVXOPT		MBA-AS			
condition	m	*cpu*	*std*	*cpu*	*std*	max-iters	min-iters
10^1	50	0.96	0.06	0.22	0.11	12	4
	100	2.69	0.21	0.25	0.11	10	8
	200	7.06	0.51	0.23	0.04	10	7
	500	20.61	2.17	0.56	0.04	9	7
	1000	46.34	3.23	1.59	0.16	8	7
	2000	96.72	10.01	2.88	0.22	8	6
10^2	50	1.17	0.08	0.63	0.19	36	10
	100	2.51	0.39	0.67	0.34	38	5
	200	6.82	0.69	0.73	0.13	21	12
	500	20.73	1.68	0.93	0.10	17	13
	1000	43.46	4.94	1.96	0.24	17	12
	2000	88.30	7.93	4.22	0.21	13	11
10^3	50	0.89	0.07	0.86	0.22	41	23
	100	2.68	0.21	0.91	0.32	63	19
	200	6.26	0.54	1.34	0.74	45	17
	500	17.29	1.59	1.46	0.44	40	18
	1000	37.77	4.14	2.30	0.41	31	17
	2000	74.09	7.17	4.78	0.47	19	13

CVXOPT vs. MBA-AS – Fixed No. Constr. $m = 50$

		CVXOPT		MBA-AS			
condition	n	*cpu*	*std*	*cpu*	*std*	max-iters	min-iters
10^1	50	1.29	0.15	0.08	0.02	11	8
	100	7.69	0.78	0.18	0.06	11	8
	200	28.48	2.44	0.35	0.12	10	7
	500	221.15	15.42	1.11	0.62	7	5
	1000	1137.32	120.41	5.02	0.28	6	5
10^2	50	1.51	0.20	0.14	0.02	18	15
	100	8.44	0.86	0.29	0.07	23	15
	200	33.57	2.09	0.60	0.09	20	15
	500	225.68	13.20	2.39	0.42	16	13
	1000	988.36	47.70	10.01	0.62	15	12
10^3	50	1.17	0.13	0.19	0.05	30	17
	100	6.51	0.56	0.47	0.12	46	22
	200	26.69	2.35	1.44	0.31	41	25
	500	225.92	22.69	4.95	1.04	48	30
	1000	1123.80	121.04	17.38	1.32	36	25

SQP vs. MBA-AS – Fixed No. Variables $n = 50$

		SQP		MBA-AS			
condition	m	*cpu*	*std*	*cpu*	*std*	max-iters	min-iters
10^1	50	0.74	0.15	0.16	0.05	11	3
	100	1.84	0.36	0.18	0.04	11	3
	200	3.94	0.51	0.27	0.08	14	8
	500	10.31	1.16	0.54	0.03	9	8
	1000	20.99	3.24	1.02	0.06	8	7
	2000	36.66	7.31	1.81	0.16	8	6
10^2	50	1.25	0.10	0.31	0.17	39	8
	100	2.24	0.18	0.63	0.39	98	15
	200	4.35	0.43	0.64	0.15	23	9
	500	10.13	1.56	1.15	0.13	19	14
	1000	15.71	3.61	1.85	0.14	17	12
	2000	26.70	6.62	3.12	0.31	15	10
10^3	50	1.22	0.41	0.54	0.20	63	9
	100	2.12	0.27	1.06	0.63	128	18
	200	4.53	0.78	1.31	0.70	104	16
	500	12.03	5.50	2.28	1.46	95	15
	1000	12.18	4.22	2.48	0.28	24	15
	2000	21.55	6.36	4.21	0.41	20	12

SQP vs. MBA-AS – Fixed No. Constr. $m = 50$

		SQP		MBA-AS			
condition	n	*cpu*	*std*	*cpu*	*std*	max-iters	min-iters
10^1	50	1.24	0.20	0.08	0.02	11	9
	100	2.14	0.54	0.18	0.03	11	8
	200	2.59	0.57	0.34	0.09	10	8
	500	7.43	1.59	0.89	0.20	7	6
	1000	20.59	3.76	2.67	0.39	6	5
10^2	50	1.33	0.08	0.16	0.02	27	18
	100	4.31	0.68	0.30	0.05	24	19
	200	10.72	1.35	0.70	0.10	23	21
	500	37.14	7.21	3.07	0.44	21	19
	1000	90.55	13.67	8.34	0.97	17	15
10^3	50	1.31	0.14	0.23	0.05	47	25
	100	4.92	0.78	0.49	0.08	45	32
	200	18.65	3.82	1.53	0.25	60	38
	500	86.57	19.85	6.26	1.81	80	24
	1000	288.39	34.90	19.29	2.51	56	33

CVXOPT vs. MBA – Fixed No. Constr. $m = 50$

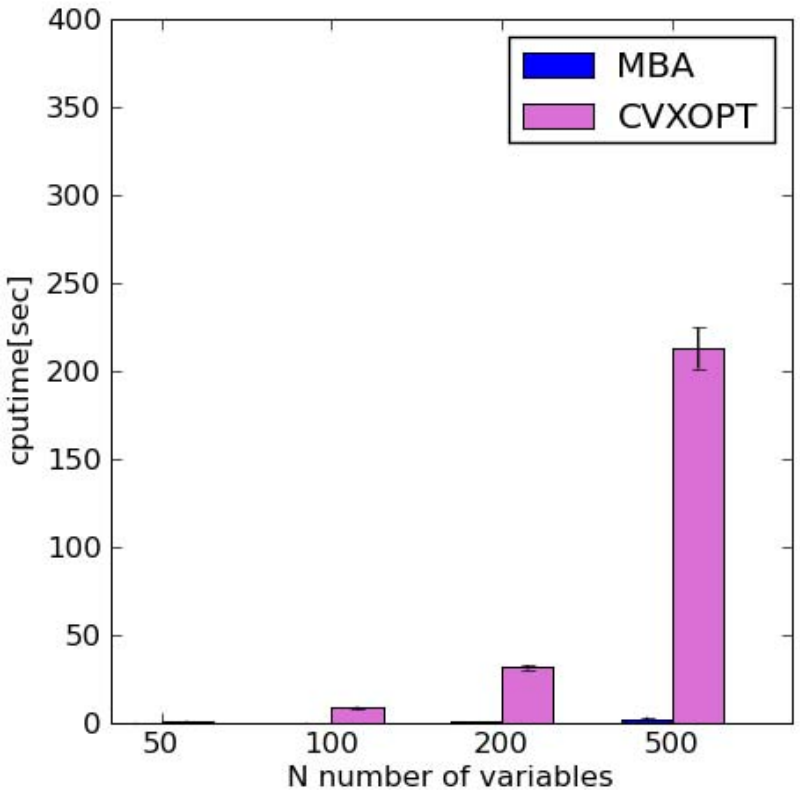
		SQP		MBA			
condition	n	*cpu*	*std*	*cpu*	*std*	max-iters	min-iters
10^1	50	1.05	0.10	0.09	0.05	9	7
	100	6.77	0.72	0.22	0.12	9	7
	200	24.52	1.54	0.43	0.23	8	6
	500	188.10	17.32	1.18	0.49	6	5
	1000	1056.74	59.82	5.44	1.03	5	3
10^2	50	1.36	0.13	0.23	0.09	20	12
	100	8.55	0.82	0.29	0.10	20	13
	200	31.65	1.26	0.66	0.25	20	14
	500	213.08	11.68	2.48	0.80	17	13
	1000	1027.87	34.57	8.70	0.52	14	12
10^3	50	1.14	0.09	0.26	0.08	31	18
	100	6.31	0.89	0.44	0.16	37	18
	200	28.90	3.52	1.16	0.35	43	23
	500	228.95	29.69	3.98	0.59	34	25
	1000	1075.36	81.68	16.59	1.48	37	27

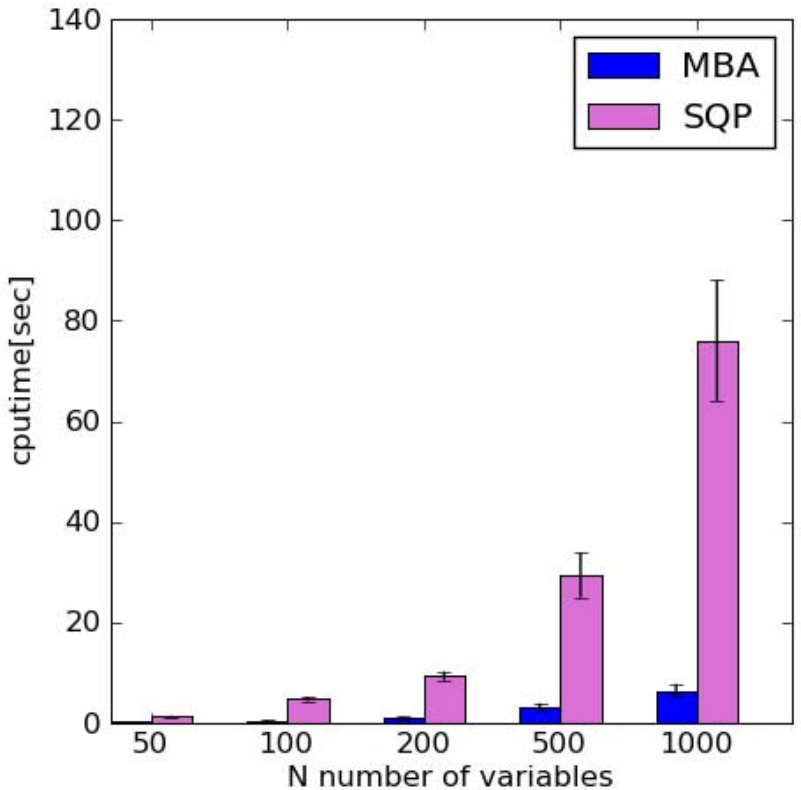
SQP vs. MBA – Fixed No. Constr. $m = 50$

		SQP		MBA			
condition	n	*cpu*	*std*	*cpu*	*std*	max-iters	min-iters
10^1	50	0.85	0.16	0.08	0.01	11	8
	100	1.05	0.21	0.30	0.14	9	8
	200	1.58	0.29	0.28	0.11	7	6
	500	5.79	1.52	0.71	0.04	6	5
	1000	18.83	4.09	1.03	0.12	6	5
10^2	50	1.34	0.12	0.29	0.06	30	18
	100	4.77	0.41	0.39	0.15	27	19
	200	9.38	0.87	0.95	0.31	23	19
	500	29.46	4.50	3.16	0.76	22	18
	1000	76.07	11.96	6.47	1.29	16	14
10^3	50	1.30	0.22	0.30	0.06	50	30
	100	4.77	0.45	0.55	0.11	49	32
	200	20.34	4.71	1.49	0.34	62	36
	500	83.03	20.19	5.91	0.74	55	39
	1000	196.96	21.86	16.46	0.91	48	41

Some Representative Results on Average Runs

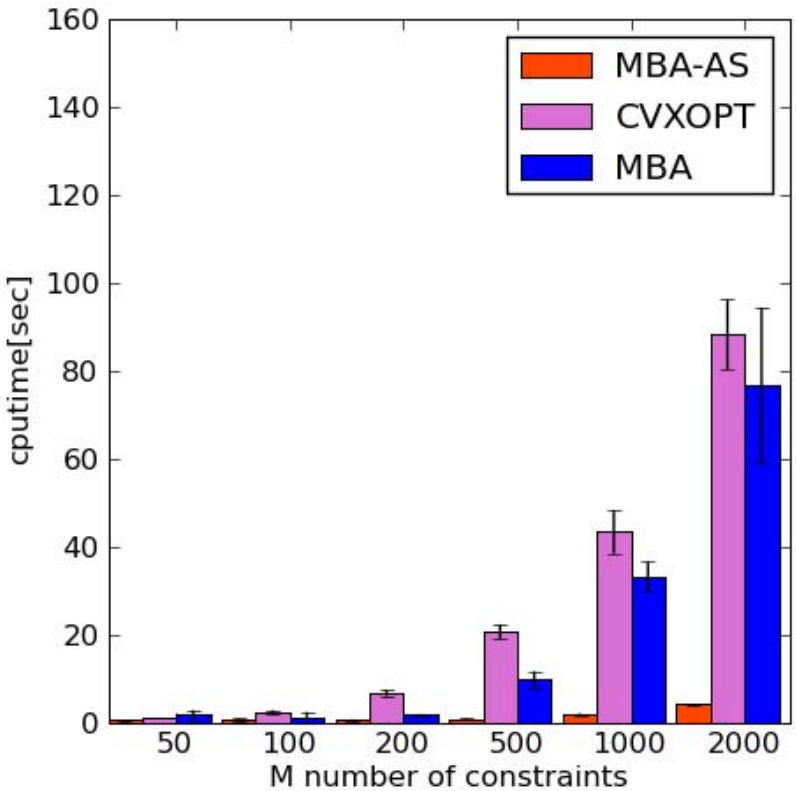
Comparison between SQP and CVXOPT to MBA:
Fixed number of constraints $m = 50$, and $n \nearrow$

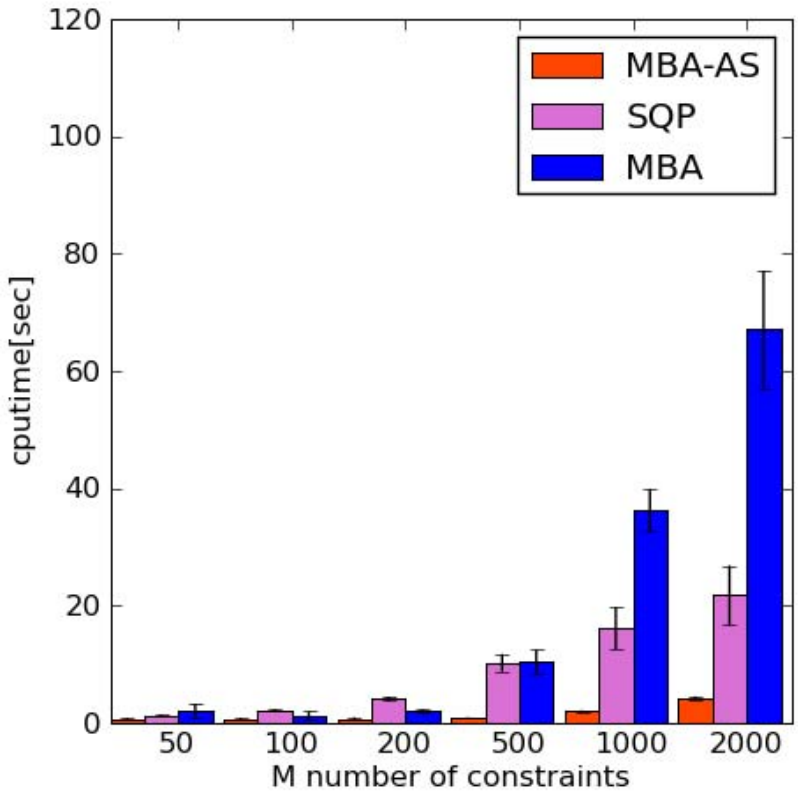




Some Representative Results on Average Runs

Comparison between SQP and CVXOPT to MBA and MBA-AS: Fixed number of variables $n = 50$, and $m \nearrow$





More Results and Details

A. Auslender, R. Shefi and M. Teboulle. A Moving Balls Approximation Method for Smooth Constrained Minimization.

To appear in SIAM J. Optimization.

http://www.math.tau.ac.il/∼teboulle/

Thank you for listening!