Computational NSO
(NonSmooth Optimization)
a tutorial focusing on bundle methods

Claudia Sagastizábal
(visiting researcher)

mailto:sagastiz@impa.br, http://www.impa.br/~sagastiz

SPCOM Tutorial, Adelaide, Feb 8th, 2015
– Computational NSO: what does it mean?
– Why special NSO methods?
– How is the oracle information used?
– Subgradient Methods
– Cutting-plane methods
– Bundle Methods
– Comparing the methods
– Going Beyond: opening the black box
– Inexact models for f
– Controlling the impact of noise
– Putting in place an on-demand accuracy scheme
– Stochastic Programming Applications in Energy
Computational NSO: what does it mean?

For the unconstrained problem

$$\min \ f(x),$$

where \( f \) is convex but not differentiable at some points
Computational NSO: what does it mean?

For the unconstrained problem

$$\min f(x),$$

where $f$ is convex but not differentiable at some points, we shall define algorithms based on information provided by an oracle or “black box”

$$x \xrightarrow[]{} f(x) \quad g(x) \in \partial f(x)$$
Computational NSO: what does it mean?

For the unconstrained problem

$$\min f(x),$$

where $f$ is convex but not differentiable at some points, we shall define algorithms based on information provided by an oracle or “black box”

Relation with this morning tutorial?
\[ f(x_0) = f(x_0) \]

\[ \nabla f(x_0) \]

\[ \text{mínimo local} \]
$$f(x_0) = f(g(x_0))$$
In NSO the skier is blind
Computational NSO: what does it mean?

For the unconstrained problem

$$\min f(x),$$

where $f$ is convex but not differentiable at some points, we shall define **algorithms** based on information provided by an oracle or “black box”

$$\nabla f(x) \in \partial f(x).$$
What do we mean by an algorithm?

An example
What do we mean by an algorithm?

An example
What do we mean by an algorithm?

An example

source http://comofas.com/
What do we mean by an algorithm?

An example

[source http://comofas.com/]
What do we mean by an algorithm?

An example
What do we mean by an algorithm?

An example
What do we mean by an algorithm?

An example

source http://comofas.com/
What do we mean by an algorithm?

An example

What do we mean by an algorithm?
What do we mean by an algorithm?

repeat until ...??

repeat until ...??
What do we mean by an algorithm?

An algorithm is a sequence of steps that are repeated until satisfaction.
What do we mean by an algorithm?

An algorithm is a sequence of steps that are repeated until satisfaction of a stopping test.
Back to Computational NSO

For the unconstrained problem

$$\min f(x),$$

where $f$ is convex but not differentiable at some points,

we look for algorithms based on information provided by an oracle or “black box”

endowed with reliable stopping tests
A quick overview of Convex Analysis

An example of a convex nonsmooth function
A quick overview of Convex Analysis

An example of a convex nonsmooth function
A quick overview of Convex Analysis

An example of a convex nonsmooth function

\[ \partial f(x) = \{ \nabla f(x) \} \]

= \{ slopes of linearizations supporting \( f \), tangent at \( x \) \}
A quick overview of Convex Analysis

An example of a convex nonsmooth function
A quick overview of Convex Analysis

An example of a convex nonsmooth function
A quick overview of Convex Analysis

An example of a convex nonsmooth function
A quick overview of Convex Analysis

An example of a convex nonsmooth function
A quick overview of Convex Analysis

An example of a convex nonsmooth function

\[ \partial f(x) = \{ g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all } y \} \]
A quick overview of Convex Analysis

An example of a convex nonsmooth function

\[ \partial f(x) = \{ g \in \mathbb{R}^n : f(y) \geq f(x) + g^\top (y - x) \text{ for all } y \} \]

= \{ slopes of linearizations supporting \( f \), tangent at \( x \) \}
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL})\)
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ |\nabla f(x_k)| = 1, \forall x \neq 0 \]

\[ \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \[ |\nabla f(x_k)| \leq \text{TOL} \quad (\leftrightarrow |g(x_k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL})\]
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]
\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \[ |\nabla f(x^k)| \leq TOL \quad (\leftrightarrow |g(x^k)| \leq TOL) \]
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]
\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \(|\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL})\]
Why special NSO methods?

Smooth optimization methods do not work

\[
\begin{align*}
\text{abs} \quad f(x) &= |x| \\
|\nabla f(x^k)| &= 1, \quad \forall x \neq 0 \\
\partial f(0) &= [-1, 1]
\end{align*}
\]

Smooth stopping test fails: \( |\nabla f(x^k)| \leq \text{TOL} \) \((\leftrightarrow |g(x^k)| \leq \text{TOL})\)
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails**: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \quad \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]
Why special NSO methods?

Smooth optimization methods **do not work**

\[ f(x) = |x| \]

\[ |\nabla f(x^k)| = 1, \quad \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test **fails:** \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]

Finite difference approximations **fail** *(no automatic differentiation)*
Why special NSO methods?

Smooth optimization methods **do not work**

\[
\begin{align*}
\text{abs} \quad f(x) &= |x| \\
|\nabla f(x^k)| &= 1, \quad \forall x \neq 0 \quad \partial f(0) = [-1, 1]
\end{align*}
\]

Smooth stopping test **fails**: \(|\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL})

Finite difference approximations **fail**

Linesearches get trapped in kinks and **fail**
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]
\[ |\nabla f(x_k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \(|\nabla f(x^k)| \leq TOL \quad (\leftrightarrow |g(x^k)| \leq TOL)\)

Linesearches get trapped in kinks and fail
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]
\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq TOL \quad (\leftrightarrow |g(x^k)| \leq TOL) \]

Finite difference approximations fail

Linesearches get trapped in kinks and fail

\(-g(x^k)\) may not provide descent
Why special NSO methods?

Smooth optimization methods do not work

\[ f(x) = |x| \]
\[ |\nabla f(x^k)| = 1, \forall x \neq 0 \quad \partial f(0) = [-1, 1] \]

Smooth stopping test fails: \[ |\nabla f(x^k)| \leq \text{TOL} \quad (\leftrightarrow |g(x^k)| \leq \text{TOL}) \]

Finite difference approximations fail

Linesearches get trapped in kinks and fail

\(-g(x^k)\) may not provide descent
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

Subgradient Methods
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

Subgradient Methods

0. Choose $x^1$ and set $k = 1$.
1. Call the oracle at $x^k$.
2. Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.
3. Make $k = k + 1$ and loop to 1.
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

**Subgradient Methods**

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$.
2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.
3 Make $k = k + 1$ and loop to 1.

Is this a good “recipe”?
Subgradient Methods

0  Choose $x^1$ and set $k = 1$.
1  Call the oracle at $x^k$.
2  Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.
3  Make $k = k + 1$ and loop to 1.

SG methods are the algorithmic version of this road sign.
Subgradient Methods

0  Choose \( x^1 \) and set \( k = 1 \).
1  Call the oracle at \( x^k \).
2  Compute \( x^{k+1} = x^k - t_k g(x^k) \) for a suitable stepsize \( t_k > 0 \).
3  Make \( k = k + 1 \) and loop to 1.

SG methods are
the algorithmic version
of this road sign

\[ \cdots \text{something is missing}! \]
Subgradient Methods

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$.
2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.
3 Make $k = k + 1$ and loop to 1.

SG methods are the algorithmic version of this road sign not a good recipe
Subgradient Methods

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$.
2 Compute $x^{k+1} = x^k - t_k g(x^k)$ for a suitable stepsize $t_k > 0$.
3 Make $k = k + 1$ and loop to 1.

SG methods are the algorithmic version of this road sign:

not a good recipe

Non-monotone!
Subgradient Methods: why a “not-good” recipe

- Non-monotone functional values, but converges
- because distance to solution set decreases for $t_k$ sufficiently small
- Lacks a stopping test
Subgradient Methods: why a “not-good” recipe

- Non-monotone functional values, but converges
- because distance to solution set decreases for $t_k$ sufficiently small
- Lacks a stopping test

…does not use all available information
Subgradient Methods: why a “not-good” recipe

\[
\begin{align*}
&\text{Non-monotone functional values, but converges} \\
&\text{because distance to solution set decreases for } t_k \text{ sufficiently small} \\
&\text{Lacks a stopping test} \\
&\text{…does not use all } \text{available} \text{ information}
\end{align*}
\]
Subgradient Methods: why a “not-good” recipe

Non-monotone functional values, but converges because distance to solution set decreases for $t_k$ sufficiently small

Lacks a stopping test

…does not use all available information

SG methods are like caipirinha without cachaça
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

Black box information defines linearizations.
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

Black box information defines linearizations that put together create a model $M$ of the function $f$.

The model is used to define iterates and to put in place a reliable stopping test.
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

Black box information defines linearizations that put together create a model $M$ of the function $f$.

$$x^i \quad \rightarrow \quad f(x^i) = f(x^i) \quad \Rightarrow \quad f^i + g^i \top (x - x^i)$$
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

Black box information defines linearizations that put together create a model $M$ of the function $f$.

$$
\chi^i \quad \Rightarrow \quad f^i = f(\chi^i), \quad g^i = g(\chi^i) \quad \Rightarrow \quad M(x) = \max_i \{ f^i + g^i \top (x - \chi^i) \} 
$$
How is the oracle information used?

We look for algorithms based on information provided by an oracle endowed with reliable stopping tests.

Black box information defines linearizations that put together create a model $M$ of the function $f$.

$$x^i ightarrow f(x^i) \quad g(x^i) \in \partial f(x)$$

$$x^i \rightarrow M(x) = \max_i \{ f^i + g^i \top (x - x^i) \}$$

(just an example, many other models are possible)
Cutting-plane methods

To minimize $f$ (unavailable in an explicit manner), minimize its model

$$M(x) = \max_i \left\{ f^i + g^i \top (x - x^i) \right\}$$

Improve the model at each iteration
Cutting-plane methods

To minimize \( f \) (unavailable in an explicit manner), minimize its model

\[
M(x) = \max_i \left\{ f_i + g_i^\top (x - x_i) \right\}
\]

Improve the model at each iteration:

\[
M_{k+1}(x) = \max_{i \leq k+1} \left\{ f_i + g_i^\top (x - x_i) \right\}
= \max \left( M_k(x), f^{k+1} + g^{k+1\top} (x - x^{k+1}) \right)
\]

where \( x^{k+1} \) minimizes \( M_k \)
Cutting-plane methods

To minimize $f$ (unavailable in an explicit manner), minimize its model $M(x) = \max_i \left\{ f^i + g^i \top (x - x^i) \right\}$

Improve the model at each iteration:

$$M_{k+1}(x) = \max_{i \leq k+1} \left\{ f^i + g^i \top (x - x^i) \right\}$$
$$= \max \left( M_k(x), f^{k+1} + g^{k+1} \top (x - x^{k+1}) \right)$$

where $x^{k+1}$ minimizes $M_k$

Instead of $x^* \in \arg \min f(x)$ at one shot
Cutting-plane methods

To minimize $f$ (unavailable in an explicit manner), minimize its model

$$M(x) = \max_i \left\{ f^i + g^i \top (x - x^i) \right\}$$

Improve the model at each iteration:

$$M_{k+1}(x) = \max_{i \leq k+1} \left\{ f^i + g^i \top (x - x^i) \right\}$$

$$= \max \left( M_k(x), f^{k+1} + g^{k+1} \top (x - x^{k+1}) \right)$$

where $x^{k+1}$ minimizes $M_k$

Instead of $x^* \in \arg\min f(x)$ at one shot,

$x^{k+1} \in \arg\min M_k(x)$ iteratively
Cutting-plane methods

Artificial bounding at least for the first iterations
Cutting-plane methods

$f(x)$

$X$

$x^1$
Cutting-plane methods

\[ f(x) \]

\[ X \]

\[ x^1 \]

\[ x^2 \]
Cutting-plane methods

$f(x)$

$x^1$ $x^2$ $x^3$
Cutting-plane methods
Cutting-plane methods

\[ f(x) \]
Cutting-plane methods

\[ \{ M_k(x^{k+1}) \} \text{ increases} \]
Cutting-plane methods

\[ \{M_k(x^{k+1})\} \text{ increases but not necessarily the functional values:} \]
\[ f(x^5) > f(x^4) \]
Cutting-plane methods

\{M_k(x^{k+1})\} increases but not necessarily the functional values: \( f(x^5) > f(x^4) \). Stopping test measures \( \delta_k := f(x^k) - M_{k-1}(x^k) \)
Cutting-plane Methods

0  Choose $x^1$ and set $k = 1$.
1  Call the oracle at $x^k$.
2  Compute $x^{k+1} \in \text{arg min}_x M_k(x)$
3  $M_{k+1}(\cdot) = \max \left( M_k(\cdot), f^k + g^{k\top} (\cdot - x^k) \right)$, $k = k + 1$, loop to 1.
Cutting-plane Methods

0. Choose $x^1$ and set $k = 1$.
1. Call the oracle at $x^k$. If $f(x^k) - M_{k-1}(x^k) \leq \text{tol}$ STOP
2. Compute $x^{k+1} \in \arg\min_X M_k(x)$
3. $M_{k+1}(\cdot) = \max \left( M_k(\cdot), f^k + g^{k\top}(\cdot - x^k) \right)$, $k = k + 1$, loop to 1.

CP methods are an improved algorithmic version of the Aussie sign.

a better recipe
Cutting-plane Methods

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$. If $f(x^k) - M_{k-1}(x^k) \leq \text{tol}$ STOP
2 Compute $x^{k+1} \in \arg\min_x M_k(x)$
3 $M_{k+1}(\cdot) = \max\left(M_k(\cdot), f^k + g^k \top (\cdot - x^k)\right)$, $k = k + 1$, loop to 1.

CP methods are an improved algorithmic version of the Aussie sign converges, but can stall and
Cutting-plane Methods

0. Choose $x^1$ and set $k = 1$.
1. Call the oracle at $x^k$. If $f(x^k) - M_{k-1}(x^k) \leq \text{tol}$ STOP
2. Compute $x^{k+1} \in \operatorname{arg \min}_x M_k(x)$
3. $M_{k+1}(\cdot) = \max \left( M_k(\cdot), f^k + g^k \top (\cdot - x^k) \right)$, $k = k + 1$, loop to 1.

CP methods are
an improved algorithmic version
of the Aussie sign

CP methods are like caipirinha with a few drops of cachaca
Cutting-plane Methods

0 Choose $x^1$ and set $k = 1$.
1 Call the oracle at $x^k$. If $f(x^k) - M_{k-1}(x^k) \leq \text{tol}$ STOP
2 Compute $x^{k+1} \in \arg\min_x M_k(x)$
3 $M_{k+1}(\cdot) = \max\left(M_k(\cdot), f^k + g^{k^\top}(\cdot - x^k)\right)$, $k = k + 1$, loop to 1.

CP methods are
an improved algorithmic version
of the Aussie sign

CP methods are like caipirinha with a few drops of cachaca
can be improved!
Cutting-plane Methods: why not the best recipe

\[
\begin{aligned}
&\text{Non-monotone functional values, but converges because } \\
&\quad \liminf \left( f(x^k) - M_{k-1}(x^k) \right) \rightarrow 0 \\
&\text{Has a stopping test, but LP size grows indefinitely eventually numerical errors prevail.}
\end{aligned}
\]

\[
x^{k+1} \in \arg\min_{x} M_{k}(x) \quad \text{with} \quad M_{k}(x) = \max_{i \leq k} \{ f^i + g^i \top (x - x^i) \}
\]

and \( X \) polyhedral
Cutting-plane Methods: why not the best recipe

- Non-monotone functional values, but converges because \( \liminf f(x^k) - M_{k-1}(x^k) \to 0 \)
- Has a stopping test, but LP size grows indefinitely eventually numerical errors prevail.

\[ x^{k+1} \in \arg \min_X M_k(x) \] 

with \( M_k(x) = \max_{i \leq k} \{ f^i + g^i \top (x - x^i) \} \) and \( X \) polyhedral

is equivalent to solving a linear programming problem

\[
\begin{align*}
\min & \quad r \\
\text{s.t.} & \quad r \in \mathbb{R}, x \in X \\
& \quad r \geq f^i + g^i \top (x - x^i) \text{ for } i \leq k
\end{align*}
\]
Cutting-plane Methods: why not the best recipe

Non-monotone functional values, but converges because
\[ \liminf \left( f(x^k) - M_{k-1}(x^k) \right) \to 0 \]
Has a stopping test, but LP size grows indefinitely eventually numerical errors prevail.

\[ x^{k+1} \in \arg\min_X M_k(x) \quad \text{with} \quad M_k(x) = \max_{i \leq k} \{ f^i + g^i \top (x - x^i) \} \]
and \( X \) polyhedral

is equivalent to solving a linear programming problem

\[
\begin{align*}
\min & \quad r \\
\text{s.t.} & \quad r \in \mathbb{R}, x \in X \\
& \quad r \geq f^i + g^i \top (x - x^i) \quad \text{for } i \leq k \text{ grows with iterations}
\end{align*}
\]
Ingredients for the best recipe

- CP brings in the concept of a model, which gives a stopping test ($\delta^k$)
- CP still non-monotone

Monotonicity defeats instability and oscillations
Ingredients for the best recipe

- CP brings in the concept of a model, which gives a stopping test \( (\delta^k) \)
- CP still non-monotone

Monotonicity defeats instability and oscillations: the sequence of function values at green-spot iterates converges
Ingredients for the best recipe

- CP brings in the concept of a model, which gives a stopping test ($\delta^k$)
- CP still non-monotone

Monotonicity defeats instability and oscillations: the sequence of function values at green-spot iterates converges

- Bundle Methods select green-spot iterates using a descent rule
Ingredients for the best recipe

- CP brings in the concept of a model, which gives a stopping test ($\delta^k$)
- CP still non-monotone

Monotonicity defeats instability and oscillations: the sequence of function values at green-spot iterates converges

- Bundle Methods select green-spot iterates using a descent rule

$$f(\hat{x}^{k+1}) \leq f(\hat{x}^k) - m\delta_k \text{ where } \delta_k \text{ is a positive quantity } < f(\hat{x}^k)$$

limit points of the serious-step subsequence $\{\hat{x}^k\}$ minimize $f$
Bundle Methods

\[ f(x) \]
Bundle Methods
Bundle Methods
Bundle Methods
Bundle Methods
Bundle Methods

0 Choose $\chi^1$, set $k = 1$, and let $\hat{\chi}^1 = \chi^1$.
1 Compute $\chi^{k+1} \in \arg\min M_k(\chi) + \frac{1}{2t_k}|\chi - \hat{\chi}^k|^2$
2 If $\delta_k := f(\hat{\chi}^k) - M_k(\chi^{k+1}) \leq \text{tol} \quad \text{STOP}$
3 Call the oracle at $\chi^{k+1}$.
   
   If $f(\chi^{k+1}) \leq f(\hat{\chi}^k) - m\delta_k$, set $\hat{\chi}^{k+1} = \chi^{k+1}$ • (Serious Step)
   Otherwise, maintain $\hat{\chi}^{k+1} = \hat{\chi}^k$ (Null Step)
4 Define $M_{k+1}$, $t_{k+1}$, make $k = k + 1$, and loop to 1.
Bundle Methods

Unlike CP $M_{k+1}(\cdot) = \max \left( M_k(\cdot), f^k + g^{k^\top}(\cdot - x^k) \right)$, now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min M_k(x) + \frac{1}{2t_k} |x - \hat{x}^k|^2$ with $M_k(x) = \max_{i \leq k} \{ f^i + g^i(\cdot - x^i) \}$ is equivalent to a QP:

$$\begin{cases} 
\min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k} |x - \hat{x}^k|^2 \\
\text{s.t.} & r \geq f^i + g^{i^\top}(x - x^i) \text{ for } i \leq k 
\end{cases}$$

A posteriori, the solution remains the same if …
Bundle Methods

Unlike CP $M_{k+1}(\cdot) = \max \left( M_k(\cdot), f^k + g^{k^\top}(\cdot - x^k) \right)$, now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min M_k(x) + \frac{1}{2t_k} |x - \hat{x}^k|^2$ with $M_k(x) = \max_{i \leq k} \{f_i + g_i^\top(x - x^i)\}$ is equivalent to a QP:

$$\begin{cases} 
\min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k} |x - \hat{x}^k|^2 \\
\text{s.t.} & r \geq f^i + g_i^\top(x - x^i) \text{ for } i \leq k 
\end{cases}$$

A posteriori, the solution remains the same if all, or . . .
Bundle Methods

Unlike CP $M_{k+1}(\cdot) = \max\left( M_k(\cdot), f^k + g^{k\top}(\cdot - x^k) \right)$, now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min M_k(x) + \frac{1}{2t_k} |x - \hat{x}^k|^2$ with $M_k(x) = \max_{i \leq k} \{f^i + g^i(\cdot - x^i)\}$ is equivalent to a QP:

$$\begin{cases} 
\min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k} |x - \hat{x}^k|^2 \\
\text{s.t.} & r \geq f^i + g^i(\cdot - x^i) \text{ for active } i's 
\end{cases}$$

A posteriori, the solution remains the same if all, or active, or . . .
Bundle Methods

Unlike CP $M_{k+1}(\cdot) = \max\left(M_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$, now the choice of the new model is more flexible: $x^{k+1} \in \arg\min M_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $M_k(x) = \max_{i \leq k}\{f^i + g^{i\top}(x - x^i)\}$ is equivalent to a QP:

$$\begin{cases} 
\min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\
\text{s.t.} & r \geq \sum_{i} \bar{\alpha}^i (f^i + g^{i\top}(x - x^i))
\end{cases}$$

A posteriori, the solution remains the same if all, or active, or the optimal convex combination
Bundle Methods

Unlike CP $\mathcal{M}_{k+1}(\cdot) = \max \left( \mathcal{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k) \right)$, now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathcal{M}_k(x) + \frac{1}{2t_k} |x - \hat{x}^k|^2$ with $\mathcal{M}_k(x) = \max_{i \leq k} \{ f^i + g^i\top(x - x^i) \}$ is equivalent to a QP:

$$\begin{cases} 
\min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & r + \frac{1}{2t_k} |x - \hat{x}^k|^2 \\
\text{s.t.} & r \geq \sum_i \bar{\alpha}^i (f^i + g^i\top(x - x^i)) 
\end{cases}$$

A posteriori, the solution remains the same if all, or active, or the optimal convex combination

$$BM \mathcal{M}_{k+1}(\cdot) = \max \left( \max_{\text{active}}, f^k + g^{k\top}(\cdot - x^k) \right)$$

aggregate
Bundle Methods

Unlike CP $\mathbf{M}_{k+1}(\cdot) = \max\left(\mathbf{M}_k(\cdot), f^k + g^{k\top}(\cdot - x^k)\right)$, now the choice of the new model is more flexible:

$x^{k+1} \in \arg\min \mathbf{M}_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$ with $\mathbf{M}_k(x) = \max_{i \leq k}\{f^i + g^i(\cdot - x^i)\}$ is equivalent to a QP:

\[
\begin{align*}
\min_{r \in \mathbb{R}, x \in \mathbb{R}^n} & \quad r + \frac{1}{2t_k}|x - \hat{x}^k|^2 \\
\text{s.t.} & \quad r \geq \sum_i \bar{\alpha}_i (f^i + g^i(\cdot - x^i))
\end{align*}
\]

Same solution if all, or active, or the optimal convex combination

$\mathbf{M}_k(\cdot)$

$\textbf{BM}\ M_{k+1}(\cdot) = \max\left(\max_{\text{active}} \text{aggregate}, f^k + g^{k\top}(\cdot - x^k)\right)$

Bundle Compression: QP with 2 constraints
Bundle Methods

When \( k \rightarrow \infty \), the algorithm generates two subsequences.

Convergence analysis addresses the mutually exclusive situations

- either the SS subsequence is infinite
- or there is a last SS, followed by infinitely many null steps
Bundle Methods

When $k \to \infty$, the algorithm generates two subsequences.

Convergence analysis addresses the mutually exclusive situations

- either the SS subsequence is infinite (limit point minimizes $f$)
- or there is a last SS, followed by infinitely many null steps (last SS minimizes $f$ and null $\to$ last SS)
Comparing the methods: bundle and SG

Typical performance on a battery of Unit Commitment problems
Comparing the methods: bundle and CP

On a battery of probabilistically constrained problems

CP is fast to reach a few digits of accuracy, then stalls
Comparing the methods: bundle and CP

On a battery of probabilistically constrained problems

- CP is fast to reach a few digits of accuracy, then stalls
- Bundle is consistently 3 times faster
Comparing the methods

**SG** ok if low precision - for instance in combinatorial optimization

**CP** ok if not many iterations - usually not the case

**Bundle** ok if f complex and high precision is required
Comparing the methods

**SG** ok if low precision - for instance in combinatorial optimization

**CP** ok if not many iterations - usually not the case

**Bundle** ok if f complex and high precision is required
Can we do any better?
Can we do any better??
Can we do any better??

YES, WE CAN
Bundle Methods with on-demand accuracy
the new generation

(or the perfect caipirinha)
First, the bad news

For a convex nonsmooth function, solving

$$\min f(x)$$

with a black box method

is doomed to slow convergence speed: complexity is $O\left(\frac{1}{\sqrt{k}}\right)^k$ iterations.
First, the bad news

For a convex nonsmooth function, solving

$$\min f(x)$$

with a black box method

is doomed to slow convergence speed: complexity is $O\left(\frac{1}{\sqrt{k}}\right)$ $k$ iterations

Better performance possible by exploiting structure
First, the bad news

For a convex nonsmooth function, solving

$$\min f(x)$$

with a black box method

$$f(x) \quad g(x) \in \partial f(x)$$

is doomed to slow convergence speed: complexity is $O\left(\frac{1}{\sqrt{k}}\right)$ $k$ iterations

Better performance possible by exploiting structure

For instance, for strongly convex $f$ complexity drops to $O\left(\frac{1}{k}\right)$
First, the bad news

For a convex nonsmooth function, solving

\[ \min f(x) \]

with a black box method

is doomed to slow convergence speed: complexity is \( O\left(\frac{1}{\sqrt{k}}\right) \) \( k \) iterations

Note: complexity results assume black box always called as above
How does structure appear?

– Explicitly
  as a sum
  as a composition

– Implicitly
  U-Lagrangian
  VU-decomposition
  partly smooth functions
How does structure appear?

– Explicitly
  as a sum
  as a composition

– Implicitly
  U-Lagrangian
  VU-decomposition
  partly smooth functions
How does structure appear?

– Explicitly
  – as a sum
  – as a composition

≠ black boxes

– Implicitly
  U-Lagrangian
  VU-decomposition
  partly smooth functions
How does structure appear?

– Explicitly
  as a sum
  as a composition
  ≠ black boxes

– Implicitly
  U-Lagrangian
  VU-decomposition
  partly smooth functions
  digging tools
Explicit Structure:
Opening the Black Box
A convex partly nonsmooth function

For $x \in \mathbb{R}^n$, given matrices $A \succeq 0$, $B \succ 0$,

$$f(x) = \sqrt{x^\top Ax + x^\top Bx}$$

has a unique minimizer at 0.

On $\mathcal{N}(A)$ the function is not differentiable, and the first term vanishes: $f|_{\mathcal{N}(A)}$ looks smooth.
This function has several interesting structures

If no structure at all

\[ f(x) = \sqrt{x^\top Ax} + x^\top Bx \]
This function has several interesting structures

If no structure at all

\[ f(x) = \sqrt{x^\top Ax + x^\top Bx} \]

This defines the black box:

\[ x \rightarrow f(x) \rightarrow g(x) \in \partial f(x) \]
This function has several interesting structures

**Sum structure**

\[ f(x) = f_1(x) + f_2(x) \]

\[ \begin{cases} 
  f_1(x) = \sqrt{x^\top Ax} \\
  f_2(x) = x^\top Bx 
\end{cases} \]
This function has several interesting structures

Sum structure

\[ f(x) = f_1(x) + f_2(x) \]

with

\[
\begin{align*}
  f_1(x) &= \sqrt{x^\top Ax} \\
  f_2(x) &= x^\top Bx
\end{align*}
\]

This defines a **sum black box**:

\[ f_1(x), f_2(x) \]

\[ g_j(x) \in \partial f_j(x)_{j=1,2} \]
This function has several interesting structures

Composite structure

\[ f(x) = (h \circ c)(x) \text{ with } \]
\[ c(x) = (x, x^\top B x) \in \mathbb{R}^{n+1} \]
\[ h(C) = \sqrt{C_{1:n}^\top A C_{1:n} + C_{n+1}} \]

for \( C \) smooth and \( h \) positively homogeneous
This function has several interesting structures

**Composite structure**

\[ f(x) = (h \circ c)(x) \]  
with  
\[
\begin{cases} 
  c(x) = (x, x^\top B x) \in \mathbb{R}^{n+1} \\
  h(C) = \sqrt{C_{1:n}^\top A C_{1:n} + C_{n+1}} 
\end{cases}
\]

for \( C \) smooth and \( h \) positively homogeneous

This defines a **composite black box**:

\[ C := c(x) \text{ and } h(C) \]

Jacobian \( Dc(x) \) and  
\[ G(C) \in \partial h(C) \]
This function has several interesting structures

Inexact information

Suppose not all of A/B is known/accessible,

so that only estimates are available for f
This function has several interesting structures

**Inexact information**

Suppose not all of $A/B$ is known/accessible,

so that only estimates are available for $f$

This defines a **noisy black box**:

\[ f_x \approx f(x) \]

\[ g_x \approx g(x) \in \partial f(x) \]
Structured models for \( f \)

No structure

\[
M(x) = \max_i \left\{ f_i^i + g_i^i \top (x - x^i) \right\} = \max_i \left\{ (f_1^i + f_2^i) + (g_1^i + g_2^i) \top (x - x^i) \right\}
\]

Sum structure

\[
M(x) = \max_i \left\{ f_1^i + g_1^i \top (x - x^i) \right\} + \max_i \left\{ f_2^i + g_2^i \top (x - x^i) \right\}
\]
Structured models for $f$

No structure

$$M(x) = \max_i \left\{ f^i + g^i \top (x - x^i) \right\}$$
$$= \max_i \left\{ (f^i_1 + f^i_2) + (g^i_1 + g^i_2) \top (x - x^i) \right\}$$

Sum structure

$$M(x) = \max_i \left\{ f^i_1 + g^i_1 \top (x - x^i) \right\}$$
$$+ \max_i \left\{ f^i_2 + g^i_2 \top (x - x^i) \right\}$$

Larger QP
Structured models for $f$

Composite structure

\[ M(x) = \max_i \left\{ G^i \top \left( c(\hat{x}) + Dc(\hat{x})(x - \hat{x}) \right) \right\} \]

\approx h(c(\hat{x}) + Dc(\hat{x})(x - \hat{x}))
Structured models for $f$

Composite structure

$$M(x) = \max_i \left\{ G^i \top \left( c(\hat{x}) + Dc(\hat{x})(x - \hat{x}) \right) \right\}$$

$$\approx h(c(\hat{x}) + Dc(\hat{x})(x - \hat{x}))$$

Good near $\hat{x}$
Inexact models for \( f \)

\[ M(x) = \max_i \left\{ f_i + g_i^\top (x - x_i) \right\} \]
Inexact models for $f$

Inexact information

$$M(x) = \max_i \left\{ f_i + g_i^\top(x - x_i) \right\}$$

excessive noise is attenuated via stepsize $t_k$
Bundle Methods with Inexact Information

\[ f(x) \]

\[ x^1 \]
Bundle Methods with Inexact Information

\[ f(x) \]

\[ \mathbf{M}_1(\cdot) + \frac{1}{2t_1} \| \cdot - \mathbf{x}^1 \|^2 \]
Bundle Methods with Inexact Information

\[ f(x) \]

\[ \mathcal{M}_2(\cdot) + \frac{1}{z_{\ell_2}} | \cdot - \chi_2 |^2 \]

\[ \mathcal{E}_f \]

\[ \mathcal{E}_g \]

\[ \chi^1 \]

\[ \chi^2 \]
Bundle Methods with Inexact Information
Controlling the impact of noise

\[
x^{k+1} = \arg\min_x \mathbf{M}(x) + \frac{1}{2t_k} |x - \hat{x}|^2
\]

Now linearizations may be inexact:

\[
\begin{align*}
  x^j & \rightarrow f^j = f_{x^j} \\
  g^j & = g_{x^j}
\end{align*}
\implies \mathbf{M}(x) = \max_{j \leq i} \left\{ f^j + g^j \top (x - x^j) \right\}
\]

And the model may be “wrong”

*If too wrong: noise needs to be attenuated*
Controlling the impact of noise

\[ \chi^{k+1} = \arg \min_{\chi} M(\chi) + \frac{1}{2t_k} |\chi - \hat{x}|^2 \]

now linearizations may be inexact:

\[ \chi^j \quad \overset{f^j = f_{\chi^j}}{\longrightarrow} \quad \overset{g^j = g_{\chi^j}}{\Longrightarrow} \quad M(\chi) = \max_{j \leq i} \left\{ f^j + g^j \top (\chi - \chi^j) \right\} \]

and the model may be “wrong”

**Noise attenuated by increasing** \( t \), **hence lowering** QP value
Detecting excessive noise by checking $\delta_k$
Detecting excessive noise: $\delta_k < 0$
Detecting excessive noise by checking $\delta_k$
Detecting excessive noise by checking $\delta_k$
Controlling the impact of noise:

oracles with on-demand accuracy

\[ x^{k+1} = \arg \min_x M(x) \frac{1}{2t_k} |x - \hat{x}|^2 \]

now linearizations may be inexact:

\[ \chi_j \]

\[ f_j = f_{\chi_j} \quad \Rightarrow \quad M(x) = \max_{j \leq i} \left\{ f_j + g_j^\top (x - \chi_j) \right\} \]

If we have the ability of computing \( f_x/g_x \) with more or less accuracy compute (asympt.) exactly SS and do not waste time in Null
On-demand accuracy scheme

Explicit structure, induced by some decomposition method

by Lagrangian relaxation

by Benders decomposition

Principle: if a problem is difficult to solve directly, solve instead a sequence of easier subproblems.

The master program has often a nonsmooth objective function
On-demand accuracy scheme

Explicit structure, induced by some decomposition method

by Lagrangian relaxation

by Benders decomposition

**Principle:** if a problem is difficult to solve directly,
solve instead a sequence of easier subproblems.

Separate subproblems allow for fast oracle calculations:
f/g defined as the sum of \( N \) terms
Lagrangian Relaxation Example

Real-life optimization problems

\[
\begin{align*}
\text{(primal)} & \quad \begin{cases}
\max & \sum_{j \in J} -c_j^j(p_j) \\
\text{subject to} & \quad p_j \in P_j, j \in J \\
& \quad \sum_{j \in J} g_j^-p_j = 0
\end{cases}
\end{align*}
\]
Lagrangian Relaxation Example

Real-life optimization problems

\[
\begin{align*}
\text{(primal)} \quad \left\{ \begin{array}{l}
\min & \sum_{j \in J} C^j(p^j) \\
p^j \in P^j, j \in J \\
\sum_{j \in J} g^j(p^j) = 0 \\
\end{array} \right.
\end{align*}
\]

Often exhibit separable structure after dualization
Lagrangian Relaxation Example

Real-life optimization problems

\[
\begin{aligned}
&\text{max} & & \sum_{j \in J} -c^j(p^j) \\
&\text{s.t.} & & p^j \in P^j, j \in J \\
& & & \sum_{j \in J} g^j(p^j) = 0 \\
\end{aligned}
\]

often exhibit separable structure after dualization
Lagrangian Relaxation Example

Real-life optimization problems

\[
\begin{align*}
\text{(primal)} \quad & \max \sum_{j \in J} -c^j(p^j) \\
& \text{subject to } \sum_{j \in J} g^j(p^j) = 0 \quad \text{subject to } \left\langle x, g^j(p^j) \right\rangle \\
& p^j \in \mathcal{P}^j, j \in J
\end{align*}
\]

often exhibit separable structure after dualization

\[
\begin{align*}
\text{(dual)} \quad & \min_x \sum_{j \in J} \left\{ \max_{p^j} -c^j(p^j) + \left\langle x, g^j(p^j) \right\rangle \right\} \\
& p^j \in \mathcal{P}^j
\end{align*}
\]
Lagrangian Relaxation Example

Real-life optimization problems

\[
\begin{align*}
\text{(primal)} & \quad \max \sum_{j \in J} -C_j(p_j) \\
& \quad \text{subject to } \sum_{j \in J} g_j(p_j) = 0, \quad p_j \in \mathcal{P}_j, j \in J
\end{align*}
\]

often exhibit separable structure after dualization

\[
\begin{align*}
\text{(dual)} & \quad \min_x \sum_{j \in J} f_j(x) \\
& \quad f_j(x) := \begin{cases} \\
& \quad \max \quad -C_j(p_j) + \langle x, g_j(p_j) \rangle \\
& \quad \text{subject to } p_j \in \mathcal{P}_j
\end{cases}
\end{align*}
\]
Energy management problems

Typically, evaluating \( f^j(x) := \left\{ \begin{array}{ll}
\max & -C^j(p^j) + \langle x, g^j(p^j) \rangle \\
P^j & \in P^j
\end{array} \right. \)

corresponds to local subproblems, related to one power plant, requiring sometimes heavy calculations
Energy management problems

Typically, evaluating $f^j(x) := \begin{cases} \max & -C^j(p^j) + \langle x, g^j(p^j) \rangle \\ p^j & \in \mathcal{P}^j \end{cases}$

corresponds to local subproblems, related to one power plant, requiring sometimes heavy calculations.

One subgradient for free: $g^j(p^j(x))$ once a solution $p^j(x)$ is available
Often, most of the CPU time is spent in the oracle calculations. For mid-term power generation planning:

\[
\min_{x} f(x) \quad \text{subject to} \quad x \in \mathbb{R}^n
\]

Scenario tree with 50,000 nodes

Nuclear subproblems are LPs with 100,000 variables and 300,000 constraints, consuming **99%** total running time
Often, most of the CPU time is spent in the oracle calculations. For mid-term power generation planning:

\[ \min f(x) \quad x \in \mathbb{R}^n \]

Can we skip/solve approximately nuclear subproblems, consuming LESS running time without losing accuracy?
Can we adapt the oracle response to the solver needs?

**dual**

\[
\begin{align*}
\text{min } f(x) \\
x \in \mathbb{R}^n
\end{align*}
\]

**1st-stage problem**

\[
\begin{align*}
\text{min } c^T x + Q(x) \\
x \in \mathcal{X}
\end{align*}
\]

**2nd-stage subproblems**

\[
\begin{align*}
Q_1(x) \\
Q_N(x)
\end{align*}
\]

**unit subproblems (primal)**

\[
\begin{align*}
x & \xRightarrow{50} 50 \\
x & \xRightarrow{5} 5 \\
x & \xRightarrow{60} 60
\end{align*}
\]
Can we adapt the oracle response to the solver needs?

\[ \min f(x) \quad x \in \mathbb{R}^n \]

unit subproblems (primal)

Now the oracle returns **INEXACT** values

\[ \min c^T x + Q(x) \quad x \in \mathcal{X} \]

2nd-stage subproblems

\[ Q_x / g_x \]

1st-stage problem
Can we adapt the oracle response to the solver needs? **YES!**

with a NSO method capable of handling **oracles with On-demand Accuracy**
Can we adapt the oracle response to the solver needs? **YES!**

with a NSO method capable of handling oracles with **On-demand Accuracy** created over noisy black-boxes

\[ f_x \approx f(x) \]
\[ g_x \approx g(x) \in \partial f(x) \]

when we have the ability of computing \( f_x / g_x \) with more or less accuracy
Oracle with on-demand accuracy

For $f^j(x) := \max \left\{ -\mathcal{C}^j(p^j) + \langle x, g^j(p^j) \rangle \right\}$

we design a noisy black box that gets additional input:

- an error bound $\varepsilon$
- a descent target $\gamma$

such that

\[
\begin{align*}
    f_x &= f(x) - \eta(x) \\
    g_x &\in \partial \eta(x) f(x) \\
    \eta(x) &\leq \varepsilon
\end{align*}
\]

for all $x$, with $\eta(x) \geq 0$

if $x$ gave enough descent: $f_x \leq \gamma$
Oracle with on-demand accuracy

For $f^j(x) := \max \left\{ \max_{p^j \in P^j} -C^j(p^j) + \langle x, g^j(p^j) \rangle \right\}$

we design a noisy black box that gets additional input:

**on demand!**

an **error bound** $\varepsilon$ and a **descent target** $\gamma$ such that

\[
\begin{align*}
  f_x &= f(x) - \eta(x) \\
  g_x &= \partial \eta(x) f(x) \\
  \eta(x) &\leq \varepsilon
\end{align*}
\]

for all $x$, with $\eta(x) \geq 0$ unknown

if $x$ gave enough descent: $f_x \leq \gamma$
Classical Bundle Method

0  Choose $x^1$, set $k = 1$ $\hat{x}^1 = x^1$.
1  Compute $x^{k+1} \in \arg\min M_k(x) + \frac{1}{2t_k} |x - \hat{x}^k|^2$
2  If $\delta_k = f(\hat{x}^k) - M_k(x^{k+1}) \leq \text{tol} \quad \text{STOP}$
3  Call the oracle at $x^{k+1}$.

If $f(x^{k+1}) \leq f(\hat{x}^k) - m\delta_k$, set $\hat{x}^{k+1} = x^{k+1}$ • (Serious Step)
Otherwise, maintain $\hat{x}^{k+1} = \hat{x}^k$ (Null Step)

4  Define $M_{k+1}, t_{k+1}$, make $k = k + 1$, and loop to 1.
Partly Exact Bundle Method

0 Choose $x^1$, $\varepsilon_1$, set $k = 1 \hat{x}^1 = x^1$.

1 Compute $x^{k+1} \in \arg\min M_k(x) + \frac{1}{2t_k} |x - \hat{x}^k|^2$

2 If $\delta_k = f_{\hat{x}^k} - M_k(x^{k+1})$ “is too negative" $t_{k+1} = 10t_k$, go to 1

Otherwise, if $\delta_k \leq \text{tol}$ STOP

3 Call the oracle at $x^{k+1}$ with $\gamma = f_{\hat{x}^k} - m\delta_k$, decreasing $\varepsilon_k$

4 Define $M_{k+1}$, $t_{k+1}$, make $k = k + 1$, and loop to 1.
Partly Exact Bundle Method

0. Choose $x^1$, $\varepsilon_1$, set $k = 1$, $\hat{x}^1 = x^1$.

1. Compute $x^{k+1} \in \arg\min M_k(x) + \frac{1}{2t_k}|x - \hat{x}^k|^2$

2. If $\delta_k = f_{\hat{x}^k} - M_k(x^{k+1})$ “is too negative" $t_{k+1} = 10t_k$,
   go to 1

   Otherwise, if $\delta_k \leq \text{tol}$ STOP

3. Call the oracle at $x^{k+1}$ with $\gamma = f_{\hat{x}^k} - m\delta_k$, decreasing $\varepsilon_k$

4. Define $M_{k+1}$, $t_{k+1}$, make $k = k + 1$, and loop to 1.

   as $\varepsilon_k \to 0$, $f_{\hat{x}^k} \to f(\hat{x}^k)$, the method finds exact solutions!
$f(x)$

$h_1 \geq h_2 \geq h_3 \cdots \geq 0$
Oracle with on-demand accuracy: versatility

\[
\begin{align*}
    f_x &= f(x) - \eta(x) \\
g_x &\in \partial_{\eta(x)} f(x) \\
\eta(x) &\leq \varepsilon
\end{align*}
\]
for all \( x \), with \( \eta(x) \geq 0 \)

We control both \( \varepsilon \) and \( \gamma \), which can vary with \( x \):

- \( \varepsilon_x = 0 \) and \( \gamma_x = +\infty \) is an exact oracle.
- \( \varepsilon_x \to 0 \) along the iterative process and \( \gamma_x = +\infty \) is an asymptotically exact oracle
- \( \varepsilon_x = 0 \) with finite \( \gamma_x \) gives a partly inexact oracle
- \( \varepsilon_x > 0 \) unknown, but bounded, with \( \gamma_x = +\infty \) is an inexact oracle
Theoretical Results

Convex proximal bundle methods in depth: a unified analysis for inexact oracles

W. de Oliveira, C. Sagastizábal, C. Lemaréchal

MathProg 148, pp 241-277, 2014

General and versatile convergence theory for inexact oracles, including

- asymptotically exact ones (driving $\epsilon$ to 0).
- inexact oracles (convergence within accuracy bound)
- lower an dupper oracles
- previous exact bundle variants
- new ones
Application in Energy I

Mid-term planning for power generation

Scenario tree with 50,000 nodes

Nuclear LPs with 100,000 variables and 300,000 constraints
Application in Energy I

Mid-term planning for power generation

Incremental Bundle

Dual

\[
\begin{align*}
\min f(x) \\
x \in \mathbb{R}^n
\end{align*}
\]

price \( x \)

\( \leq 50 \)

\( \geq 60 \)

\( f_x \text{ and } g_x \)

\( j^\text{th} \) unit subproblem

Primal

Skips Nuclear LPs (alternating) \( \equiv \) noisy black box

25% less CPU time than exact bundle, same accuracy
Application in Energy II

2-stage stochastic linear programs

L-shaped decomposition into N scenarios
Application in Energy II

2-stage stochastic linear programs

Skips 80% LPs solution ≡ noisy black box
4 times faster than L-shaped, same accuracy
Applications in Energy III

Maximize revenue of hydro producers keeping reservoir levels between min-zones with 90% confidence (numerical integration in dimension 192!)

Comparison with previous values obtained by Wim van Ackooij, from R&D at EDF on several instances from Val d’Isère (Alpes), using a method by A. Prékopa.

Huge reduction in CPU times: drops from almost 3h to 3 minutes
Closing remarks

- Thanks to Welington de Oliveira and Marc Schmidt for some of the images.

- Credits to some co-authors: Welington de Oliveira, Claude Lemaréchal, Wim van Ackooij

- **Warning:** This tutorial does not intend to encourage drinking caipirinha.
Closing remarks

- Thanks to Welington de Oliveira and Marc Schmidt for some of the images.

- Credits to some co-authors: Welington de Oliveira, Claude Lemaréchal, Wim van Ackooij

- **Warning:** This tutorial does not intend to encourage drinking caipirinha.
  
  It is rather meant to facilitate the use of modern (on-demand accuracy) bundle methods.

Any doubts or questions, just e-mail me
To learn more

(exact) Bundle books

Inexact Bundle theory

Inexact Bundle variants with applications

and my web-page: http://www.impa.br/~sagastiz
ICSP 2016
XIV International Conference on Stochastic Programming

June 25-July 01, 2016
Búzios, Brazil

Save the date!