Parallel Space Decomposition of the Mesh Adaptive Direct Search algorithm

Sébastien Le Digabel
Charles Audet
John Dennis

ICCOPT-MOPTA, August 2007

Presentation Outline

Presentation Outline

Presentation Outline

- Introduction
- MADS and PSD Overview
- Adaptation of PSD for MADS
- Convergence analysis
- Preliminary Results
- Discussion

Presentation Outline

Presentation Outline

Presentation Outline

Summary of the talk

- MADS is a direct search algorithm for nonsmooth optimization
- Parallel Space Decomposition (PSD) methods based on the block-Jacobi method [Bertsekas, Tsitsiklis 1989] are generic and parallel frameworks for optimization
- We propose to apply a PSD method to MADS in order to solve medium-size problems ($50 < n < 500$) $\rightarrow$ PSD-MADS

Target Class of Problems

$$\min_{x \in \Omega \subseteq \mathbb{R}^n} f(x)$$

where

- $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$
- function f and constraints defining Ω are black-box functions
  - nonsmooth
  - noisy
  - problematic derivative approximation
  - can possibly fail to evaluate
  - costly to evaluate (seconds, minutes, days)
  - usually the result of a computer code

MADS Overview

- **M**esh **A**daptive **D**irect **S**earch [Audet, Dennis 2005]
- NOMAD project: www.gerad.ca/nomad [Audet, Couture]
- Extends the **G**eneralized **P**attern **S**earch [Torczon 1997]
- Direct Search method: derivative are not evaluated nor approximated
- Iterative algorithm in two steps where the black-box functions are evaluated at some trial points, which are either accepted as new iterates or rejected

MADS Mesh

- All trial points at iteration k are constructed to lie on a mesh

$$M(\Delta_k) = \left\{ x_k + \Delta_k Dz : z \in \mathbb{N}^{n_D} \right\} \subset \mathbb{R}^n$$

  where $\Delta_k \in \mathbb{R}^+$ is the **mesh size parameter** and D a fixed set of n_D directions in $\mathbb{R}^n$
- After each iteration, Δ_k is reduced when no new iterate has been found (iteration failure)

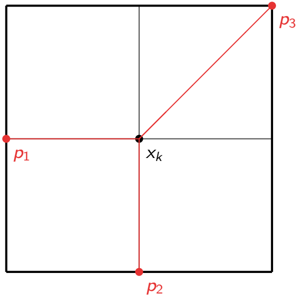
Poll and Search

At iteration k: two steps: the Poll and the Search

- Poll
  - local exploration on the mesh near the best current iterate x_k
  - use of MADS directions (at least one is necessary to ensure convergence)
  - MADS directions are not the fixed set of directions D
- Search
  - global and flexible exploration strategy
  - has only to generate a finite number of trial points lying on the mesh

Poll illustration (successive fails and mesh shrink)

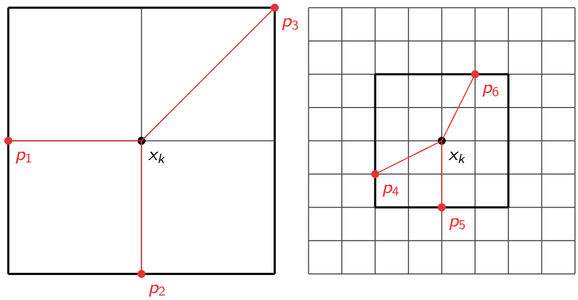
$\Delta_k = 1$



trial points$=\{p_1, p_2, p_3\}$

Poll illustration (successive fails and mesh shrink)

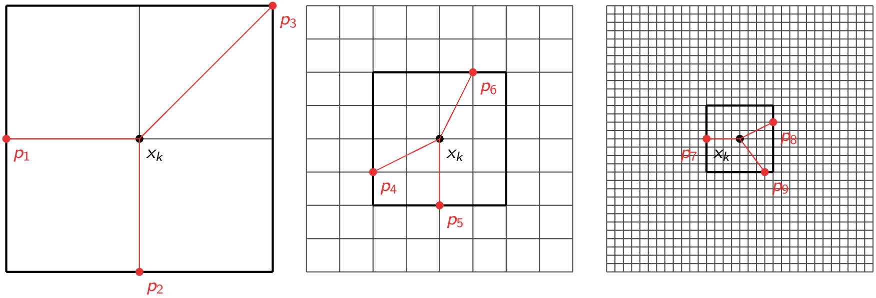
$\Delta_k = 1$ $\Delta_{k+1} = 1/4$



trial points$=\{p_1, p_2, p_3\}$ $= \{p_4, p_5, p_6\}$

Poll illustration (successive fails and mesh shrink)

$\Delta_k = 1$ $\Delta_{k+1} = 1/4$ $\Delta_{k+2} = 1/16$



trial points$=\{p_1, p_2, p_3\}$ $= \{p_4, p_5, p_6\}$ $= \{p_7, p_8, p_9\}$

MADS Convergence

- Constraints are handled with the barrier approach: if $x \notin \Omega$, $f(x)$ is considered to be $+\infty$
- A hierarchical convergence based on f differentiability analysis is available for MADS with barrier
- **Main convergence result** : MADS leads to a Clarke stationnary point $\hat{x} \in \Omega$ if f is Lipschitz near $\hat{x}$:
  $f^{\circ}(\hat{x}; d) \geq 0$ for all $d \in T_{\Omega}^{Cl}(\hat{x})$
- **Corollary for unconstrained case** : if the function is strictly differentiable, then $\nabla f(\hat{x}) = 0$

PSD methods

- Block-Jacobi method in [Bertsekas, Tsitsiklis 1989]
- Generic and parallel optimization framework
- Idea: each process works on a subproblem and has responsability of small groups of variables
- Iterative algorithm with two steps:
  - **decomposition**: subproblems with a reduced number of variables are optimized in parallel
  - **synchronization**: results of subproblems are gathered; a new iterate is constructed
- Parallel Variable Distribution [Ferris, Mangasarian 1994]

PSD methods

Subproblem $\mathcal{P}_P : \min_{x \in \Omega_p(x^*)} f(x)$

$\Omega_p(x^*) = \{x \in \Omega : x_i = x_i^* \ \forall i \in \{1, 2, ..., n\} \setminus N_p\}$

Initializations

x_0, lists N_p of subproblems variables

Iteration k

[1] **Parallel Decomposition [slave s_p]**

optimizes subproblem $\mathcal{P}_p$ with $x^* = x_k$

$y_i \leftarrow$ solution of optimization

[2] **Synchronization [master]**

$x_{k+1} \leftarrow$ new iterate from all solutions y_p's

$k \leftarrow k + 1$

goto [1] until some stopping condition is met

Adaptation of PSD for MADS (PSD-MADS)

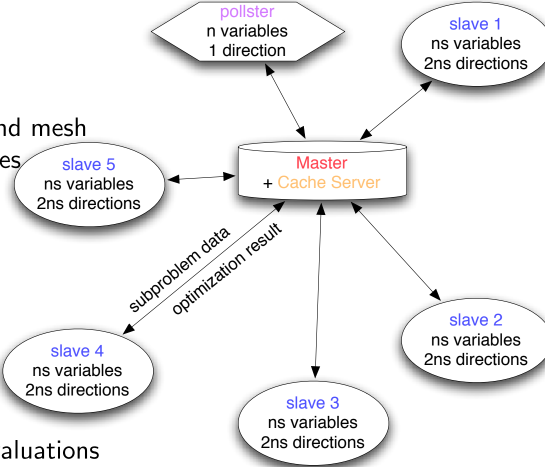
- MADS is used to optimize subproblems
- The synchronization step is removed
- Main parameters:
  - bbe : maximum number of black-box evaluations for each MADS optimization (does not include cache hits)
  - ns : number of variables in subproblems

Slaves = Regulars slaves + Pollster slave

- Regular slaves: solve subproblems with standard MADS directions on a reduced number of variables (N_p)
- Pollster slave
  - all n variables are considered
  - single-polls: only one direction per poll
  - terminates after one iteration
  - the set of normalized poll directions grows dense in the unit-sphere

Processes occupation

- Master
  - receives all slave's signals
  - updates current solution and mesh
  - decides subproblem variables
  - sends subproblem data
- Slaves (regular and pollster)
  - receive subproblem data
  - optimize subproblem
  - send optimization data
- Cache server
  - memorizes all black-box evaluations
  - allows the "cache search" for regular slaves



Choice of the sets N_p

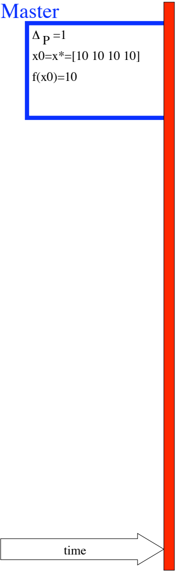
- N_p are the sets of variables for subproblem $\mathcal{P}_p$ optimized by regular slave s_p
- Original PSD method:
  - fixed sets N_p for all iterations
  - sets N_p had to form a partition of $\{1, 2, ..., n\}$
- PSD-MADS:
  - sets are not required to form a partition of $\{1, 2, ..., n\}$
  - sets may change for a same slave between two optimizations
  - Implementation: sets are randomly and uniformly chosen
  - Implementation: all sets have the same size ns

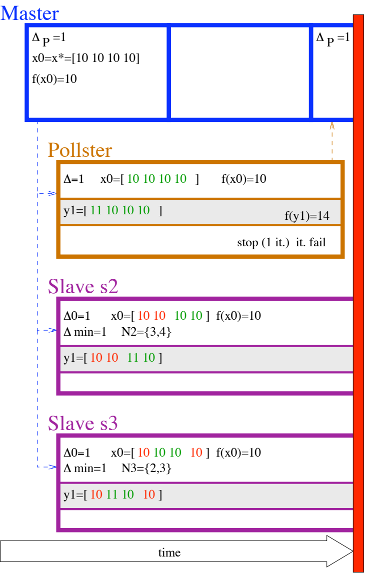
Convergence from the pollster's perspective

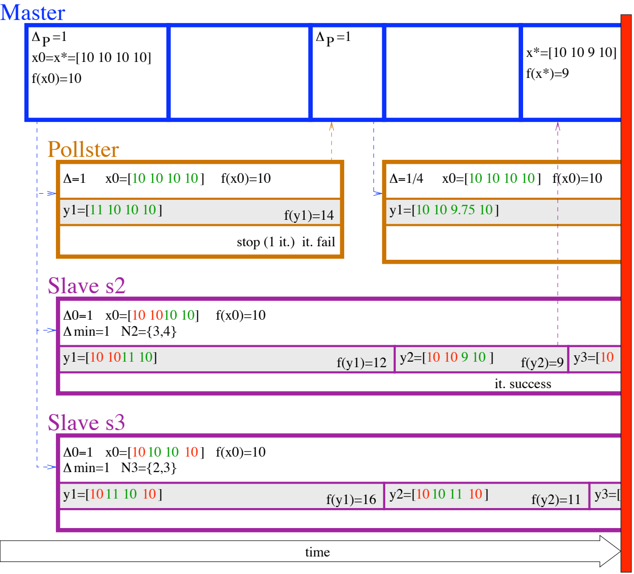
- The pollster runs a complete MADS algorithm on the original problem :
  - The poll contains a single direction
  - Search
    - consists in obtaining the best cache point
    - by construction, slaves generate a finite number of points on the pollster mesh
  - Global mesh size Δ_P: link between pollster and slaves:

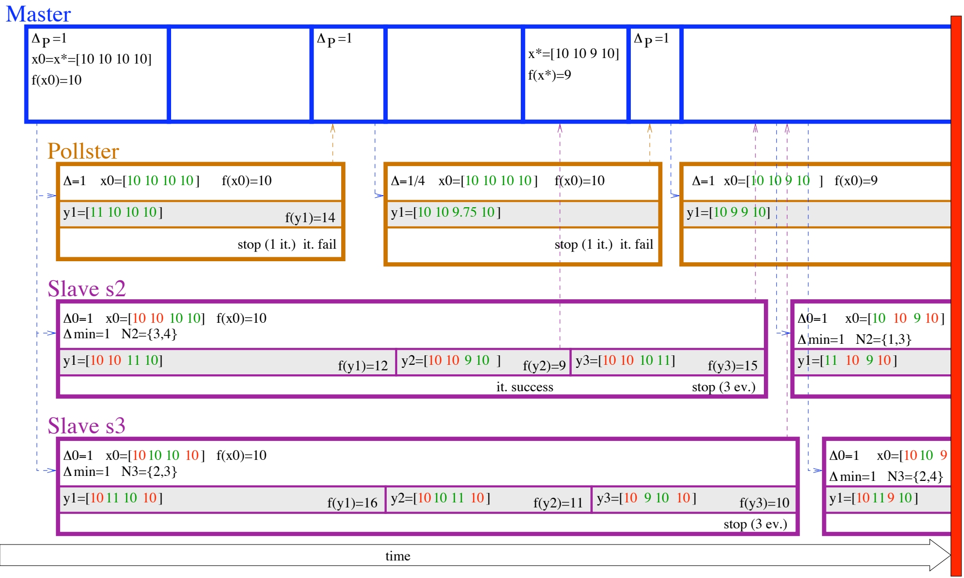
$$\Delta_{pollster} \leq \Delta_P \leq \Delta_{reg.slaves}$$

- All MADS convergence conditions are verified:
  the MADS theoretical convergence analysis holds


Master
Δ P =1
x0=x*=[10 10 10 10]
f(x0)=10
time


Master
Δ P =1
x0=x*=[10 10 10 10]
f(x0)=10
Δ P =1
Pollster
Δ=1 x0=[10 10 10 10] f(x0)=10
y1=[11 10 10 10] f(y1)=14
stop (1 it.) it. fail
Slave s2
Δ0=1 x0=[10 10 10 10] f(x0)=10
Δ min=1 N2={3,4}
y1=[10 10 11 10]
Slave s3
Δ0=1 x0=[10 10 10 10] f(x0)=10
Δ min=1 N3={2,3}
y1=[10 11 10 10]
time


Master
Δ P =1
x0=x*=[10 10 10 10]
f(x0)=10
Δ P =1
x*=[10 10 9 10]
f(x*)=9
Pollster
Δ=1 x0=[10 10 10 10] f(x0)=10
y1=[11 10 10 10] f(y1)=14
stop (1 it.) it. fail
Δ=1/4 x0=[10 10 10 10] f(x0)=10
y1=[10 10 9.75 10]
Slave s2
Δ0=1 x0=[10 1010 10] f(x0)=10
Δ min=1 N2={3,4}
y1=[10 1011 10] f(y1)=12
y2=[10 10 9 10] f(y2)=9
y3=[10
it. success
Slave s3
Δ0=1 x0=[10 10 10 10] f(x0)=10
Δ min=1 N3={2,3}
y1=[10 11 10 10] f(y1)=16
y2=[10 10 11 10] f(y2)=11
y3=[
time



Master
Δp=1
x0=x*=[10 10 10 10]
f(x0)=10
Δp=1
x*=[10 10 9 10]
f(x*)=9
Δp=1
Pollster
Δ=1 x0=[10 10 10 10] f(x0)=10
y1=[11 10 10 10] f(y1)=14
stop (1 it.) it. fail
Δ=1/4 x0=[10 10 10 10] f(x0)=10
y1=[10 10 9.75 10]
stop (1 it.) it. fail
Δ=1 x0=[10 10 9 10] f(x0)=9
y1=[10 9 9 10]
Slave s2
Δ0=1 x0=[10 10 10 10] f(x0)=10
Δmin=1 N2={3,4}
y1=[10 10 11 10] f(y1)=12
y2=[10 10 9 10] f(y2)=9
y3=[10 10 10 11] f(y3)=15
it. success
stop (3 ev.)
Δ0=1 x0=[10 10 9 10]
Δmin=1 N2={1,3}
y1=[11 10 9 10]
Slave s3
Δ0=1 x0=[10 10 10 10] f(x0)=10
Δmin=1 N3={2,3}
y1=[10 11 10 10] f(y1)=16
y2=[10 10 11 10] f(y2)=11
y3=[10 9 10 10] f(y3)=10
stop (3 ev.)
Δ0=1 x0=[10 10 9
Δmin=1 N3={2,4}
y1=[10 11 9 10]
time

Test Problem G2 from [Hedar, Fukushima 2006]

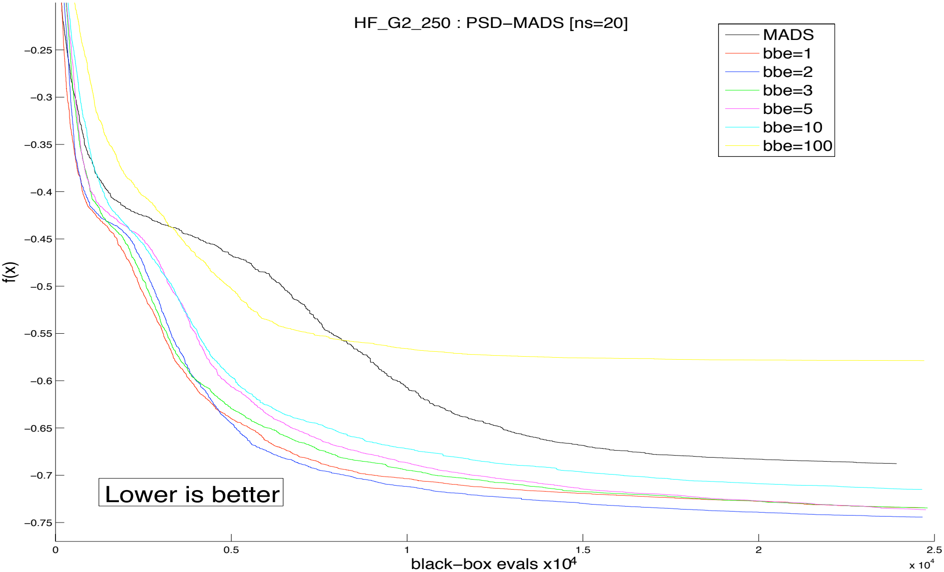
$$\min_{x\in\mathbb{R}^n} f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4 x_i - 2\prod_{i=1}^{n} \cos^2 x_i}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

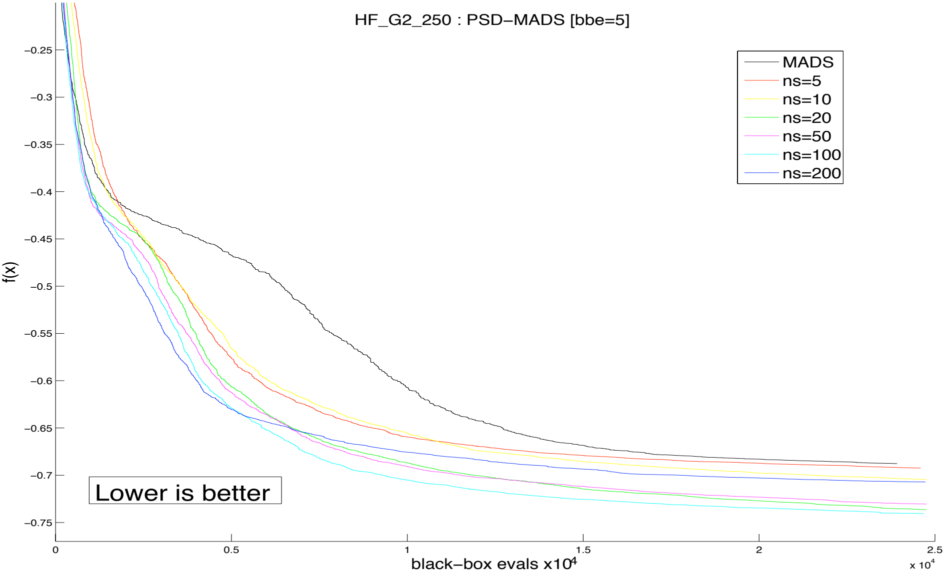
$$s.t. \begin{cases} g_1(x) = -\prod_{i=1}^{n} x_i + 0.75 \leq 0 \\ g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0 \end{cases}$$

$n = 250$, $0 \leq x_i \leq 10$, $x_0 = [5\ 5\ \ldots\ 5]^T$

Testing protocols

- Graphs showing the number of evaluations *v.s* the objective function value
- Each plot is an average of 30 runs
- Different PSD-MADS runs are compared to the basic parallel version of MADS
- PSD-MADS parameters tested: *bbe* and *ns*
- Budget of 25000 evaluations
- 14 processes


HF_G2_250 : PSD−MADS [ns=20]
MADS
bbe=1
bbe=2
bbe=3
bbe=5
bbe=10
bbe=100
f(x)
−0.25
−0.3
−0.35
−0.4
−0.45
−0.5
−0.55
−0.6
−0.65
−0.7
−0.75
0
0.5
1
1.5
2
2.5
x 10^4
black−box evals x10^4
Lower is better


HF_G2_250 : PSD−MADS [bbe=5]
MADS
ns=5
ns=10
ns=20
ns=50
ns=100
ns=200
Lower is better
f(x)
−0.25
−0.3
−0.35
−0.4
−0.45
−0.5
−0.55
−0.6
−0.65
−0.7
−0.75
0
0.5
1
1.5
2
2.5
x 10^4
black−box evals x10^4

Discussion

- PSD-MADS: new algorithm applying the PSD parallel framework to MADS
- Promising results for a large problem
- Convergence results of MADS still hold
- Work in progress:
  - PVD synchronization $\longrightarrow$ new PSD-MADS recomposition
  - include the PVD "forget-me-not" terms
  - compare results with APPS: Asynchronous Parallel Pattern Search [Hough, Kolda, Torczon 2001]
- Questions ?

Discussion

- PSD-MADS: new algorithm applying the PSD parallel framework to MADS
- Promising results for a large problem
- Convergence results of MADS still hold
- Work in progress:
  - PVD synchronization $\rightarrow$ new PSD-MADS recomposition
  - include the PVD “forget-me-not” terms
  - compare results with APPS: Asynchronous Parallel Pattern Search [Hough, Kolda, Torczon 2001]
- Questions ?