Parallel versions of the mesh adaptive direct search algorithm

Sébastien Le Digabel Antoine Lesage-Landry Samuel Mendoza

DFOS'24, 2024-06-27

Presentation outline

Introduction

MADS and NOMAD

Parallel versions of MADS

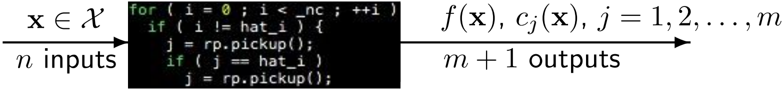
Computational tests

Conclusion

Context: Blackbox Optimization (BBO)

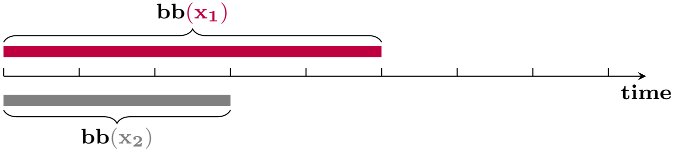
$$\min_{\mathbf{x} \in \mathcal{X}} \quad f(\mathbf{x}) \text{ s.t. } \mathbf{x} \in \Omega = \{\mathbf{x} \in \mathcal{X} : c_j(\mathbf{x}) \leq 0, j = 1, 2, \ldots, m\}$$

$\mathcal{X}$ is a n-dimensional space and the evaluations of f and the c_j's are provided by a blackbox:



- Each call to the blackbox may be time-expensive and time-heterogeneous
- The evaluation can fail
- Sometimes $f(\mathbf{x}) \neq f(\mathbf{x})$
- Derivatives are not available and cannot be approximated

Heterogeneous blackboxes



- Numerical methods may take longer to converge close to an optimal point: This occurs with the solar problem [Andrés-Thió et al., 2024]
- CPU-time related functions [Abramson et al., 2012]
- This may have great impact for parallel methods

Motivation

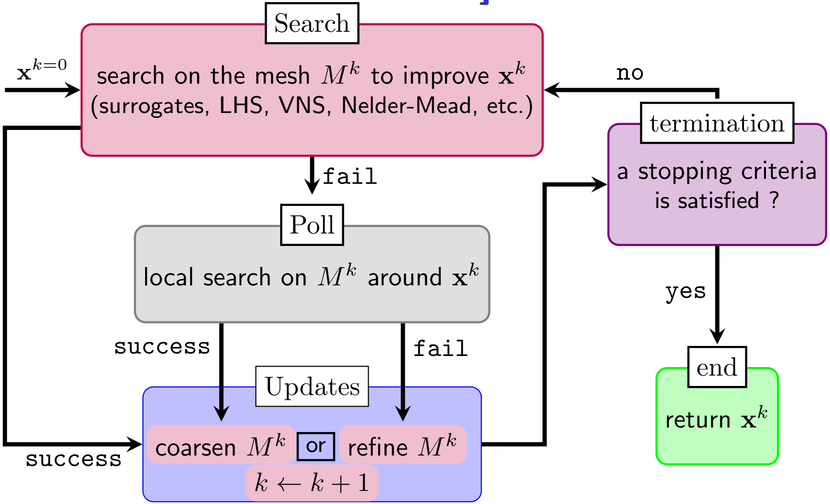
Different ways of parallelizing BBO:

- Parallelize the blackbox
- Parallelize the algorithm: Subject of this work
- Hybrid solution: Share processes between the blackbox and the algorithm
- With multiple available processes, it is also possible to construct large lists of trial points

Parallelize MADS: 4 methods:

- pMADS-S
- pMADS-A
- COOP-MADS
- PSD-MADS

MADS [Audet and Dennis, Jr., 2006]



[0] Initializations $(\mathbf{x}^0, \delta^0)$

[1] Iteration k

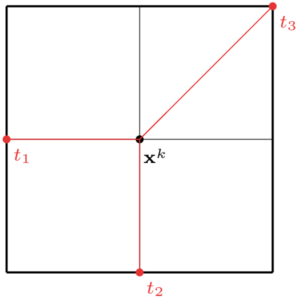
- **[1.1] Search**
  - select a finite number of mesh points
  - evaluate candidates opportunistically
- **[1.2] Poll** (if Search failed)
  - construct poll set $P_k = \{\mathbf{x}^k + \delta^k d : d \in D_k\}$
  - sort(P_k)
  - evaluate candidates opportunistically

[2] Updates

- if success
  - $\mathbf{x}^{k+1} \leftarrow$ success point
  - increase δ^k
- else
  - $\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k$
  - decrease δ^k
- $k \leftarrow k + 1$, stop or go to **[1]**

MADS illustration with $n = 2$: Poll step

$$\delta^k = \Delta^k = 1$$



poll trial points$=\{t_1, t_2, t_3\}$

δ^k is the mesh size parameter

Δ^k is the frame size parameter

we keep $\delta^k < \Delta^k$ typically with $\Delta^k = \sqrt{\delta^k}$

and $\delta^{k+1} \leftarrow \delta^k \times 4$ (success)

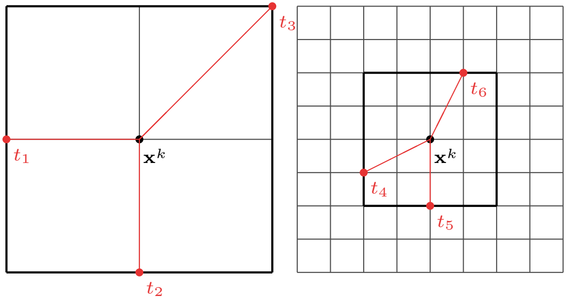
or $\delta^{k+1} \leftarrow \delta^k / 4$ (fail)

MADS illustration with $n = 2$: Poll step

$\delta^k = \Delta^k = 1$

$\delta^{k+1} = 1/4$

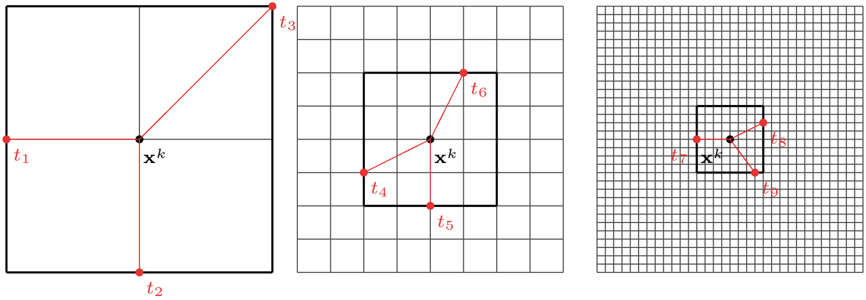
$\Delta^{k+1} = 1/2$



poll trial points$=\{t_1, t_2, t_3\}$ $= \{t_4, t_5, t_6\}$

MADS illustration with $n = 2$: Poll step

$\delta^k = \Delta^k = 1$ $\delta^{k+1} = 1/4$ $\Delta^{k+1} = 1/2$ $\delta^{k+2} = 1/16$ $\Delta^{k+2} = 1/4$



poll trial points$=\{t_1, t_2, t_3\}$ $= \{t_4, t_5, t_6\}$ $= \{t_7, t_8, t_9\}$

NOMAD (Nonlinear Optimization with MADS)

- C++ implementation of the MADS algorithm [Audet and Dennis, Jr., 2006]
- Standard C++. Runs on Linux, Mac OS X and Windows
- Parallel versions with MPI and OpenMP
- MATLAB versions; Multiple interfaces (Python, Excel, etc.)
- Open and free – LGPL license
- Download at https://www.gerad.ca/nomad or https://github.com/bbopt/nomad
- Support at nomad@gerad.ca
- Related articles in TOMS [Le Digabel, 2011, Audet et al., 2022]

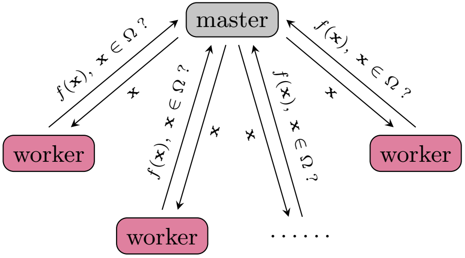
NOMAD: Parallel versions

- Current versions: 3.9 (June 2018) and 4.4.0 (January 2024)
- V3: based on MPI
- V4: based on OpenMP
- Parallel implementations are more mature in NOMAD V3
- Both versions can generate large lists of trial points if many processes are available, using enriched poll directions and sampling

Parallel MADS (pMADS)

Implementation (NOMAD V3):

- Evaluate the trial points in parallel: Straightforward opportunity for speedup
- Master/worker paradigm over MPI communications
- The efficiency of the opportunistic strategy is affected



pMADS versions

pMADS-S (synchronous):

- The iteration is over only when all the evaluations in progress are terminated
- Processes can be idle between two evaluations

pMADS-A (asynchronous):

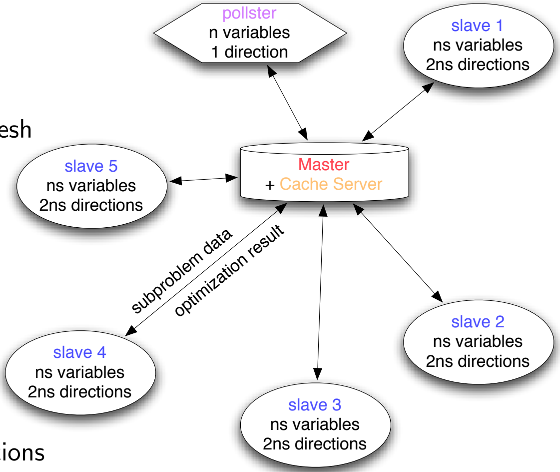
- If a new best point is found, the iteration is terminated even if there are evaluations in progress. New trial points are then generated
- Processes never wait between two evaluations
- "Old" evaluations (stragglers) are considered when they are finished
- Processes are never idled during an iteration

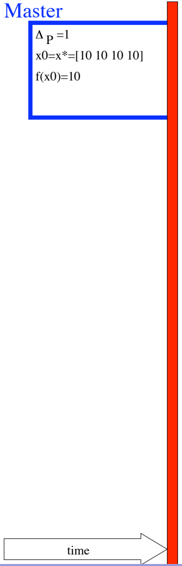
PSD-MADS

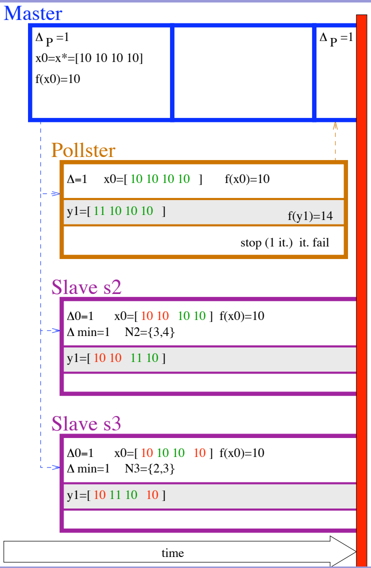
- **PSD:** Parallel Space Decomposition [Audet et al., 2008b]
- Idea: each process executes a MADS algorithm on a subproblem and has responsibility of small groups of variables
- Based on the block-Jacobi method [Bertsekas and Tsitsiklis, 1989] and on the Parallel Variable Distribution [Ferris and Mangasarian, 1994]
- Objective: solve larger problems ($\simeq 50 - 500$ instead of $\simeq 10 - 20$)
- Choice of subproblems:
  - Random
  - Overlapping is authorized
  - Small size: $n = 2$
  - Small budget of evaluations ($\simeq$ 10)

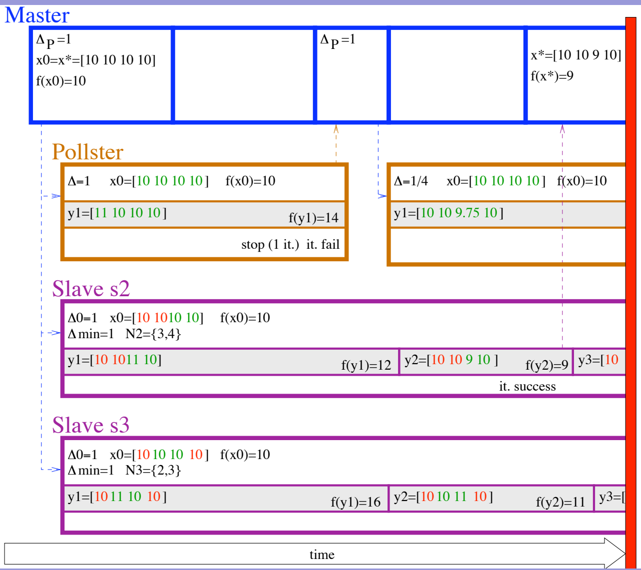
PSD-MADS: processes

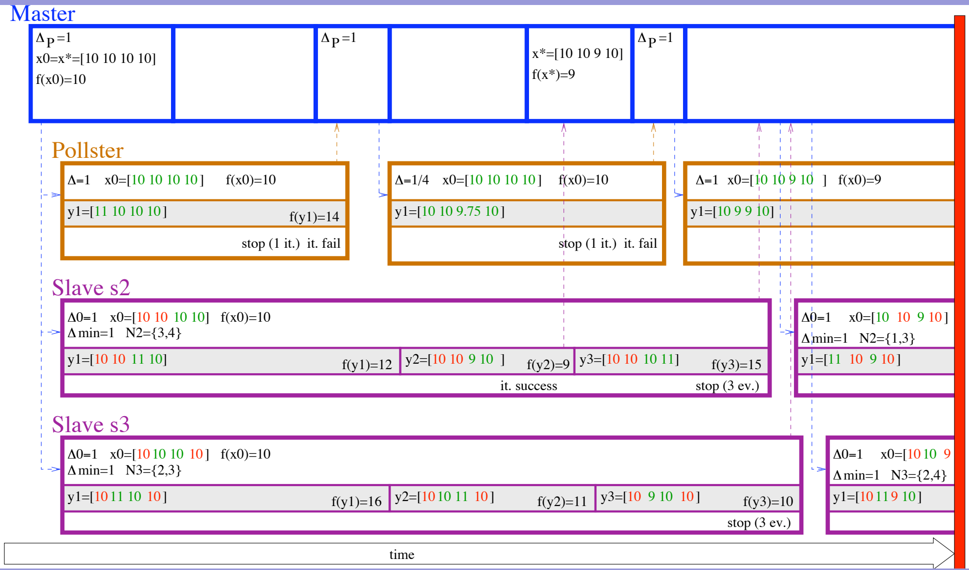
- Master
  - receives all workers signals
  - updates current solution and mesh
  - decides subproblem variables
  - sends subproblem data
- Workers ~~Slaves~~
  - receive subproblem data
  - optimize subproblem
  - send optimization data
- Cache server
  - memorizes all black-box evaluations
  - allows the "cache search" in the pollster




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

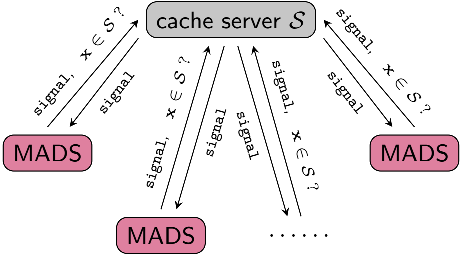

Master
Δ P =1
x0=x*=[10 10 10 10]
f(x0)=10
Δ P =1
Pollster
Δ=1 x0=[10 10 10 10] f(x0)=10
y1=[11 10 10 10] f(y1)=14
stop (1 it.) it. fail
Slave s2
Δ0=1 x0=[10 10 10 10] f(x0)=10
Δ min=1 N2={3,4}
y1=[10 10 11 10]
Slave s3
Δ0=1 x0=[10 10 10 10] f(x0)=10
Δ min=1 N3={2,3}
y1=[10 11 10 10]
time


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
Δ P =1
x0=x*=[10 10 10 10]
f(x0)=10
Δ P =1
x*=[10 10 9 10]
f(x*)=9
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
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
Δ min=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]
Δ min=1 N2={1,3}
y1=[11 10 9 10]
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=[10 9 10 10] f(y3)=10
stop (3 ev.)
Δ0=1 x0=[10 10 9
Δ min=1 N3={2,4}
y1=[10 11 9 10]
time

Cooperative MADS: COOP-MADS

- Uses a simplified version of the PSD-MADS parallel framework
- Processes run in parallel on the original problem with different seeds in order to produce different behaviours
- The cache server $\mathcal{S}$ allows to share evaluations
- (Almost) asynchronous method

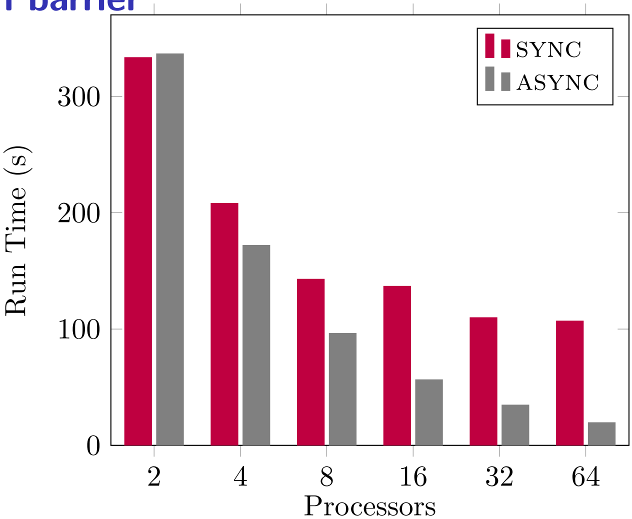


Convergence

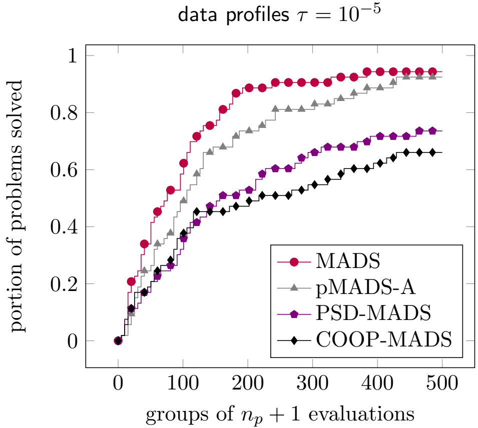
- pMADS-S: Same as MADS
- pMADS-A: Almost same as MADS except when stragglers give new successes: In that case, convergence is ensured by setting the mesh size parameter back to the "old" value
- PSD-MADS: Convergence based on the pollster that considers all variables. Convergence is ensured by the MADS framework: no conditions on the subproblems definitions
- COOP-MADS: Same as MADS

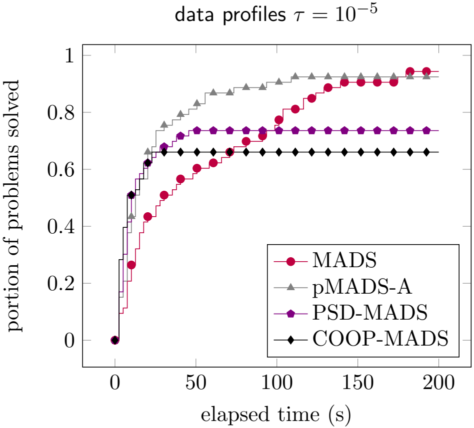
Impact of the synchronisation barrier

- solar4: $n = 29$, $m = 16$
- Heterogeneous blackbox
- pMADS-S (SYNC) vs pMADS-A (ASYNC)
- 64 processors



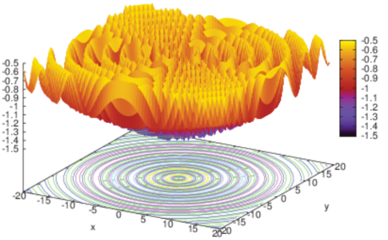
Moré-Wild smooth test problems

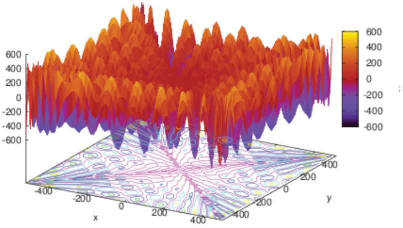




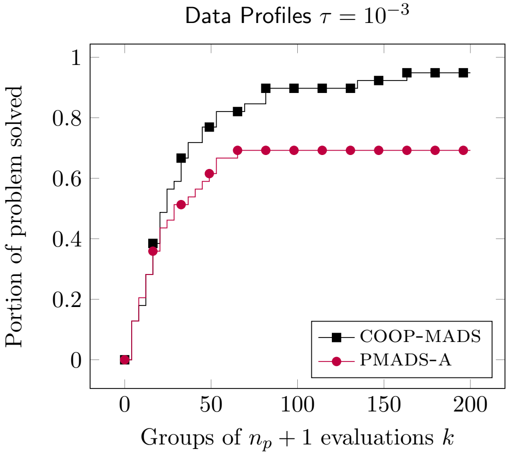
Multimodal problems for testing COOP-MADS

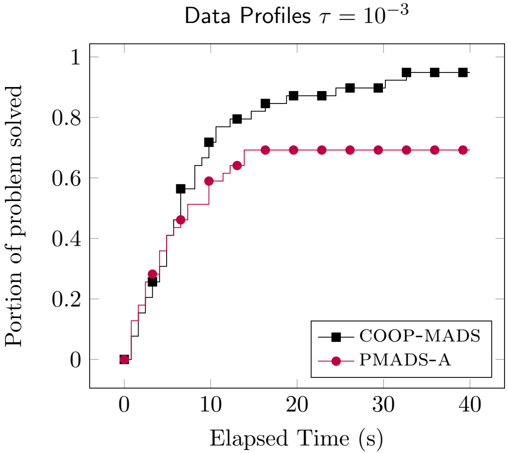
- 42 multimodal problems from [Jamil and Yang, 2013]
- $n = 2$
- budget of 1,600 evaluations
- 4 MPI processes



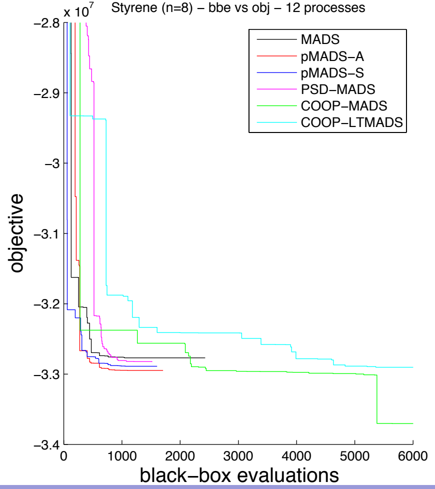


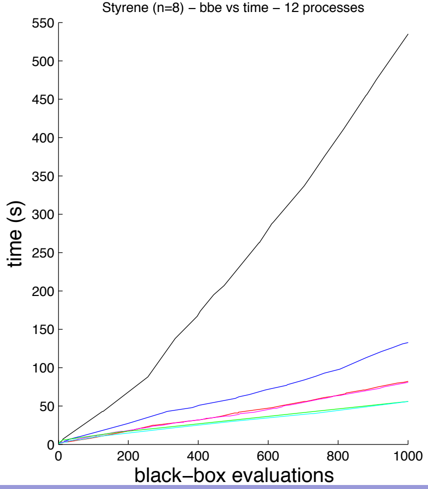
COOP-MADS on multimodal problems





STYRENE, n=8, 6,000 evaluations, with 13 processes





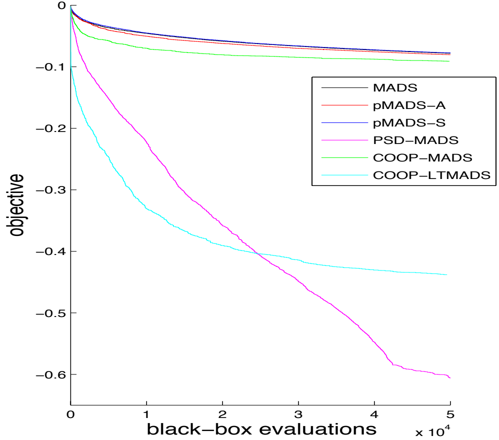
Test Problem G2 [Hedar and Fukushima, 2006]

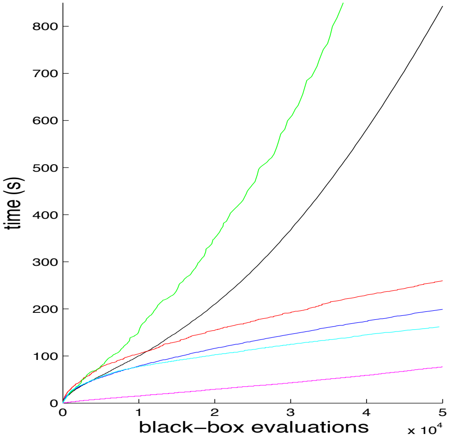
$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \left| \frac{\sum_{i=1}^{n} \cos^4 x_i - 2 \prod_{i=1}^{n} \cos^2 x_i}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

$$s.t. \begin{cases} c_1(\mathbf{x}) = -\prod_{i=1}^{n} x_i + 0.75 \leq 0 \\ c_2(\mathbf{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0 \end{cases}$$

$n = 500,\ 0 \leq \mathbf{x} \leq 10,\ \mathbf{x}^0 = [5\ 5\ \dots\ 5]^\top$

Problem G2, n=500, 50,000 evaluations





Summary

- We presented four parallel versions of the MADS algorithm:
  - pMADS-S and pMADS-A
  - PSD-MADS
  - COOP-MADS
- These versions are mature in NOMAD V3
- Achieve good improvements depending on several contexts (size, time-heterogeneity, multimodality, etc.)
- Future work:
  - More benchmarking
  - Full availability in NOMAD V4
  - Use of smarter subspaces in PSD-MADS for large problems

References I

Abramson, M., Asaki, T., J.E.Dennis, Jr., Magallanez, Jr., R., and Sottile, M. (2012).
An Efficient Class of Direct Search Surrogate Methods for Solving Expensive Optimization Problems with CPU-time-related Functions.
Structural Multidisciplinary Optimization, 45(1):53–64.

Andrés-Thió, N., Audet, C., Diago, M., Gheribi, A., Le Digabel, S., Lebeuf, X., Lemyre Garneau, M., and Tribes, C. (2024).
`solar`: A solar thermal power plant simulator for blackbox optimization benchmarking.
Technical Report G-2024-37, Les cahiers du GERAD.

Audet, C., Béchard, V., and Le Digabel, S. (2008a).
Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search.
Journal of Global Optimization, 41(2):299–318.

Audet, C. and Dennis, Jr., J. (2006).
Mesh Adaptive Direct Search Algorithms for Constrained Optimization.
SIAM Journal on Optimization, 17(1):188–217.

References II

Audet, C., Dennis, Jr., J., and Le Digabel, S. (2008b).
Parallel Space Decomposition of the Mesh Adaptive Direct Search Algorithm.
SIAM Journal on Optimization, 19(3):1150–1170.

Audet, C. and Hare, W. (2017).
Derivative-Free and Blackbox Optimization.
Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland.

Audet, C., Le Digabel, S., Rochon Montplaisir, V., and Tribes, C. (2022).
Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm.
ACM Transactions on Mathematical Software, 48(3):35:1–35:22.

Bertsekas, D. and Tsitsiklis, J. (1989).
Parallel and distributed computation: numerical methods.
Prentice-Hall, Upper Saddle River, NJ, USA.

Ferris, M. and Mangasarian, O. (1994).
Parallel Variable Distribution.
SIAM Journal on Optimization, 4(4):815–832.

References III

Hedar, A.-R. and Fukushima, M. (2006).
Derivative-free filter simulated annealing method for constrained continuous global optimization.
Journal of Global Optimization, 35(4):521–549.

Jamil, M. and Yang, X.-S. (2013).
A literature survey of benchmark functions for global optimisation problems.
International Journal of Mathematical Modelling and Numerical Optimisation, 4(2):150–194.

Le Digabel, S. (2011).
Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm.
ACM Transactions on Mathematical Software, 37(4):44:1–44:15.

Moré, J. and Wild, S. (2009).
Benchmarking Derivative-Free Optimization Algorithms.
SIAM Journal on Optimization, 20(1):172–191.