Ecole Polytechnique de Montréal

Blackbox Optimization

(with MADS and NOMAD)

Sébastien Le Digabel
Charles Audet
John Dennis

ICiS, Snowbird
2010–08–06

Presentation outline

MADS overview

Presentation outline

MADS overview

Selected algorithmic features

Presentation outline

MADS overview

Selected algorithmic features

NOMAD

Presentation outline

MADS overview

Selected algorithmic features

NOMAD

Discussion

MADS overview

Selected algorithmic features

NOMAD

Discussion

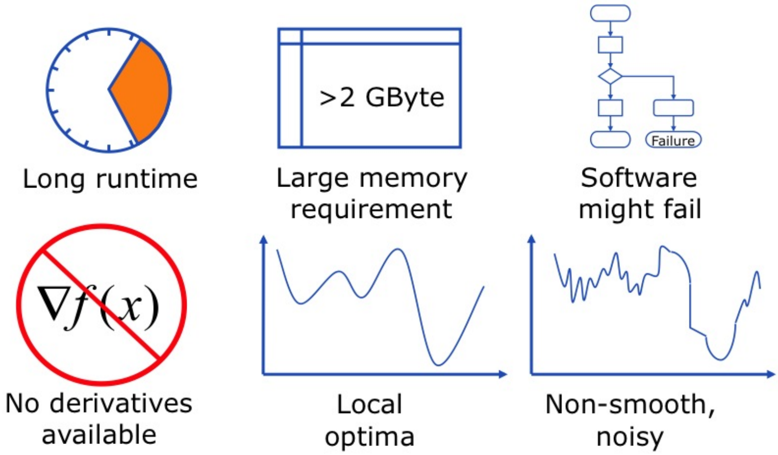
Blackbox optimization problems

We consider the nonsmooth optimization problem:

$$\begin{array}{ll} \text{Minimize} & f(x) \\ \text{subject to} & x \in \Omega, \end{array}$$

where evaluation of the functions are usually the result of a computer code (a blackbox)

Blackboxes as illustrated by J. Simonis [ISMP 2009]



Mesh Adaptive Direct Search (MADS)

- Iterative algorithm that evaluates the blackbox functions at some trial points.
- Trial points are generated on the mesh

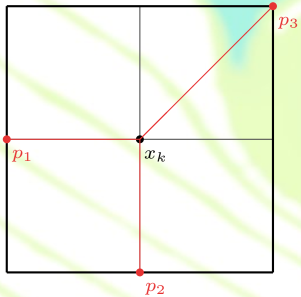
$$M(\Delta_k) = \left\{x_k + \Delta_k D z : z \in \mathbb{N}^{n_D}\right\} \subset \mathbb{R}^n$$

where x_k is the current iterate, $\Delta_k \in \mathbb{R}^+$ is the mesh size parameter, and D a fixed set of n_D directions in $\mathbb{R}^n$.

- **Search step:** trial points can be generated anywhere on the mesh. It is typically user-provided but can also be generic.
- **Poll step:** directions are used to generate poll trial points. The normalized directions become dense in the unit sphere.
- **Updates step:** the mesh size is reduced if no new iterate is found.
- Algorithm backed by a convergence analysis based on the Clarke Calculus for nonsmooth functions.

Poll illustration (successive fails and mesh shrink)

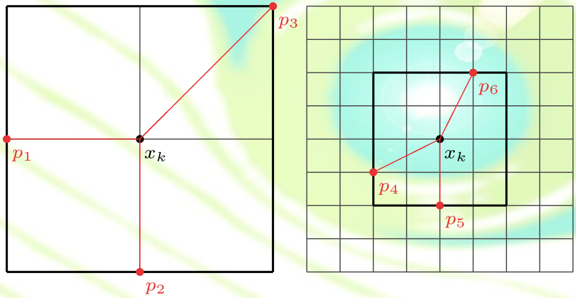
$\Delta_k = 1$



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

Poll illustration (successive fails and mesh shrink)

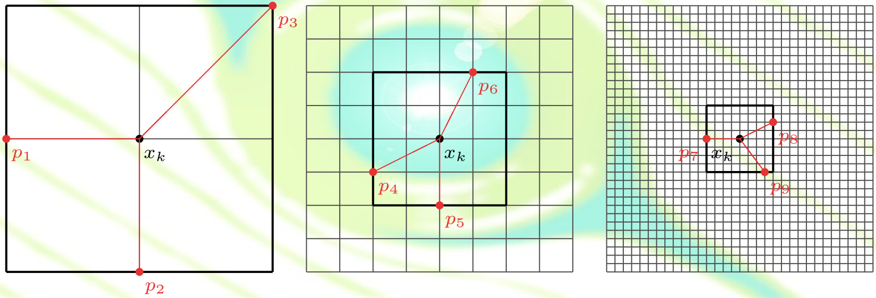
$\Delta_k = 1$ $\Delta_{k+1} = 1/4$



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 overview

Selected algorithmic features

NOMAD

Discussion

Types of directions used in the poll step

- Coordinate directions (compass search).
- LT-MADS directions: use randomness and are not orthogonal.
- Ortho-MADS directions: orthogonal directions based on the quasi-random Halton sequence (results are reproducible).
- LT-MADS and Ortho-MADS directions become dense in the whole space of variables.

Constraints handling

The domain: $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$

Constraints can be relaxable, unrelaxable or hidden.

- Unrelaxable constraints define X

  Cannot be violated by any trial point.
  For example, logical conditions on the variables indicating if the simulation may be launched.

Constraints handling

The domain: $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$

Constraints can be relaxable, unrelaxable or hidden.

- Unrelaxable constraints define X
- Relaxable constraints $c_j(x) \leq 0$

  Can be violated, and $c_j(x)$ provides a measure of how much the constraint is violated. A budget for example.

Constraints handling

The domain: $\Omega = \{x \in X : c_j(x) \leq 0, j \in J\} \subset \mathbb{R}^n$

Constraints can be relaxable, unrelaxable or hidden.

- Unrelaxable constraints define X
- Relaxable constraints $c_j(x) \leq 0$
- Hidden constraints

  Is a convenient term to denote the set of points in the feasible region for the relaxable or unrelaxable constraints at which the blackbox fails to return a value for one of the problem functions.
  A typical example is when the simulation fails to return a value.

Three strategies to deal with constraints

- Extreme barrier (EB)

Treats the problem as being unconstrained,
by replacing the objective function $f(x)$ by

$$f_\Omega(x) := \begin{cases} f(x) & \text{if } x \in \Omega, \\ \infty & \text{otherwise.} \end{cases}$$

The problem

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

is then solved.
Remark : If $x \notin X$ (the non-relaxable constraints), then the costly evaluation of $f(x)$ is not performed.

Three strategies to deal with constraints

- Extreme barrier (EB)
- Progressive barrier (PB)

  Defined for the relaxable constraints.
  As in the filter methods of Fletcher and Leyffer, it uses the non-negative constraint violation function $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$

$$h(x) := \begin{cases} \sum_{j \in J} (\max(c_j(x), 0))^2 & \text{if } x \in X, \\ \infty, & \text{otherwise.} \end{cases}$$

  At iteration k, points with $h(x) > h_k^{\max}$ are rejected by the algorithm, and $h_k^{\max}$ decreases toward 0 as $k \to \infty$.

Three strategies to deal with constraints

- Extreme barrier (EB)
- Progressive barrier (PB)
- Progressive-to-Extreme Barrier (PEB)

  Initially treats a relaxable constraint by the progressive barrier. Then, if polling around the infeasible poll center generates a new infeasible incumbent that satisfies a constraint violated by the poll center, then that constraint moves from being treated by the progressive barrier to the extreme barrier.

Categorical variables

- Categorical variables are discrete unrelaxable variables.
- The set of possible values cannot be ordered and the user must provide a neighborhood definition.
- For some problems, according to the different values of the categorical variables, the number of variables may change. NOMAD deals with such cases (by associating *signatures* to trial points).
- An extended poll is performed using the neighborhood method in order to try new values for the categorical variables.

Surrogates

- A surrogate of the function f is a function that we hope shares some similarities with f but is much cheaper to evaluate.
- Usage in direct search methods:
  - Allows a sophisticated search step.
  - Criterion to sort trial points for evaluation.
  - Find a starting point.
  - Searches may consider only surrogate evaluations and f is evaluated only at the most promising points.
- NOMAD currently considers only non adaptive surrogates.

Biobjective optimization

$$\begin{array}{ll} \text{minimize} & F(x) = (f^{(1)}(x), f^{(2)}(x)) \\ \text{subject to} & x \in \Omega \end{array}$$

- $u \in \Omega$ dominates $v \in \Omega$ $(u \prec v)$ if and only if $f^{(1)}(u) \leq f^{(1)}(v)$ and $f^{(2)}(u) \leq f^{(2)}(v)$ with at least one strict inequality.
- $u \in \Omega$ is Pareto optimal if and only if there is no $w \in \Omega$ such that $w \prec u$.
- Objective: find the Pareto set or the set of optimal trade-off solutions.

Single-objective subproblems

In order to approximate the Pareto front of a biobjective problem,

- The single-objective version of NOMAD is executed on a series of subproblems.
- Nonlinear single-objective reformulations $\phi_r(F(x))$ are considered.
- The solution of each of these subproblems produces a local approximation of the Pareto front.

We do not use weights. Instead, let $r \in \mathbb{R}^2$ denote a reference point in the objective space. We introduce the objective function

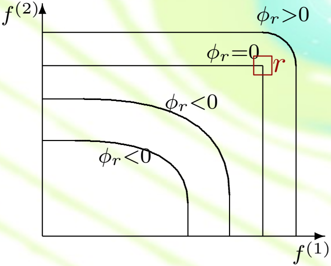
$$\phi_r(F(x)) := \begin{cases} -\prod_{q=1}^{p} (r_q - f^{(q)}(x))^2 & \text{if } F(x) \le r, \\ \sum_{q=1}^{p} \left((f^{(q)}(x) - r_q)_+\right)^2 & \text{otherwise.} \end{cases}$$

When minimized, it generates a Pareto solution that dominates r.

We do not use weights. Instead, let $r \in \mathbb{R}^2$ denote a reference point in the objective space. We introduce the objective function

$$\phi_r(F(x)) := \begin{cases} -\prod_{q=1}^{p} (r_q - f^{(q)}(x))^2 & \text{if } F(x) \le r, \\ \sum_{q=1}^{p} \left((f^{(q)}(x) - r_q)_+\right)^2 & \text{otherwise.} \end{cases}$$

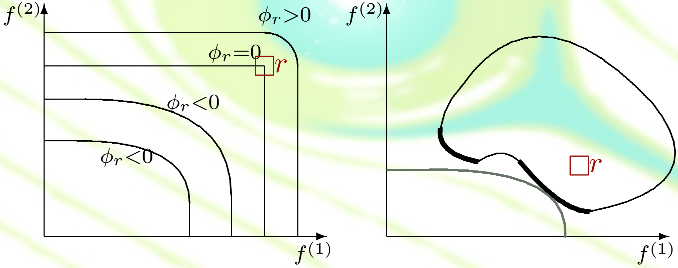
When minimized, it generates a Pareto solution that dominates r.



We do not use weights. Instead, let $r \in \mathbb{R}^2$ denote a reference point in the objective space. We introduce the objective function

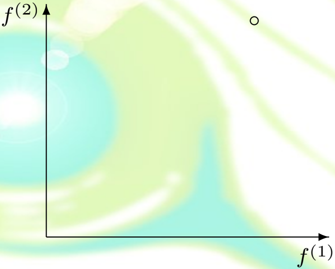
$$\phi_r(F(x)) := \begin{cases} -\prod_{q=1}^{p} (r_q - f^{(q)}(x))^2 & \text{if } F(x) \le r, \\ \sum_{q=1}^{p} \left((f^{(q)}(x) - r_q)_+\right)^2 & \text{otherwise.} \end{cases}$$

When minimized, it generates a Pareto solution that dominates r.



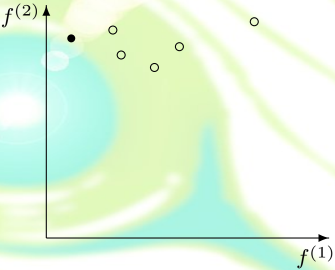
Every Pareto solution can be obtained by setting r appropriately.

BiMADS algorithm



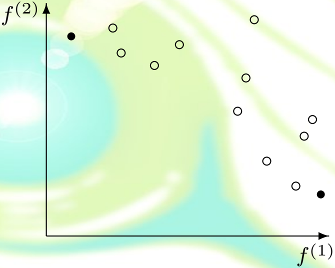
- **Initialization:**
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.

BiMADS algorithm



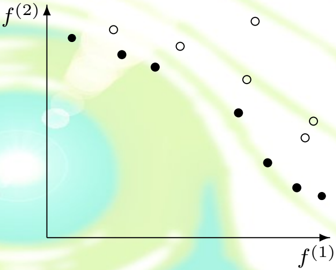
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.

BiMADS algorithm



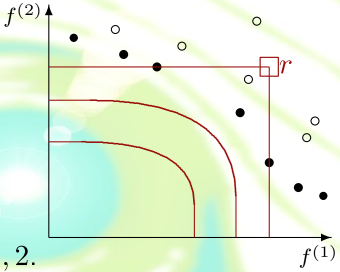
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.

BiMADS algorithm



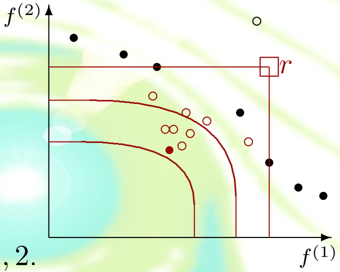
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.

BiMADS algorithm



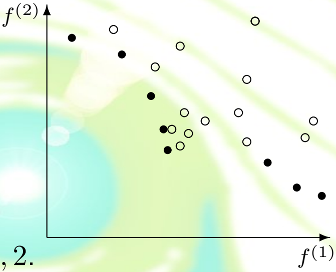
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.
  - SINGLE-OBJECTIVE MINIMIZATION:
    Solve the single-objective problem $\min_{x \in \Omega} \phi_r(F(x))$.

BiMADS algorithm



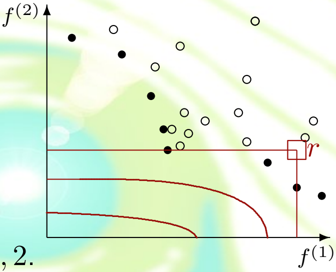
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.
  - SINGLE-OBJECTIVE MINIMIZATION:
    Solve the single-objective problem $\min_{x \in \Omega} \phi_r(F(x))$.

BiMADS algorithm



- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.
  - SINGLE-OBJECTIVE MINIMIZATION:
    Solve the single-objective problem $\min_{x \in \Omega} \phi_r(F(x))$.

BiMADS algorithm



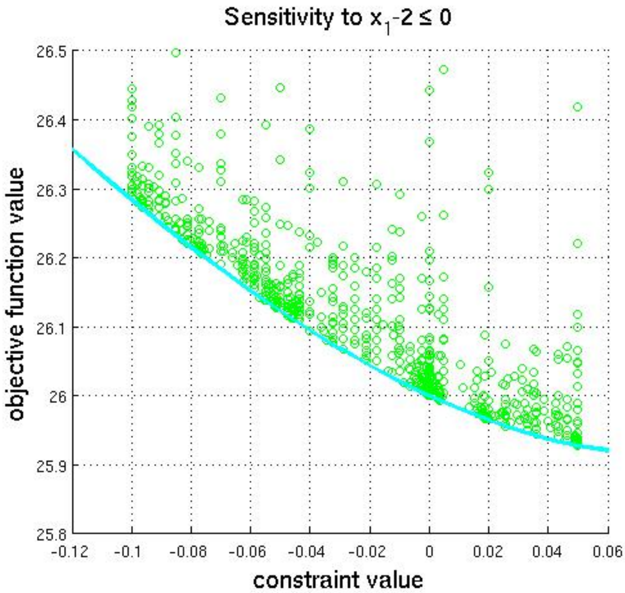
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.
  - SINGLE-OBJECTIVE MINIMIZATION:
    Solve the single-objective problem $\min_{x \in \Omega} \phi_r(F(x))$.

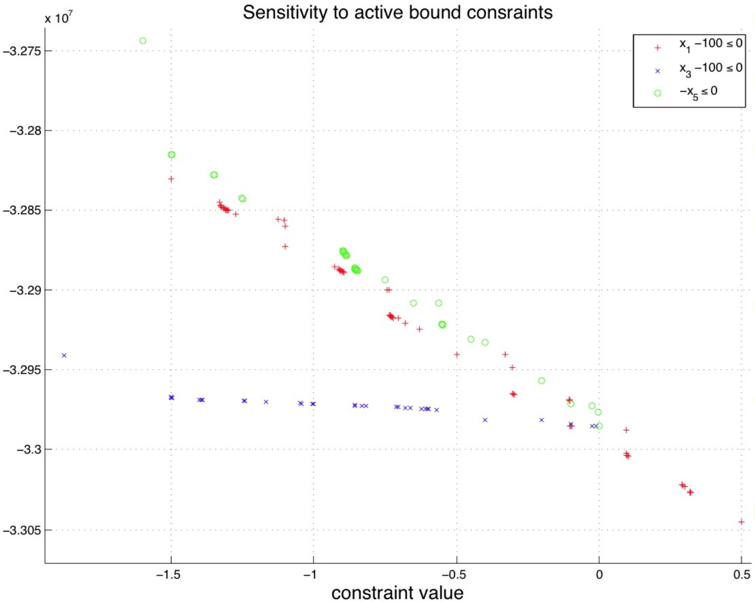
Sensitivity analysis

- Getting an optimizer is often insufficient for engineers; Sensitivity analysis for constraints is a useful tool to grasp more knowledge and see which constraints are important and may be relaxed or tighten.
- Simple analysis: after an execution the cache is inspected and a plot $c_j(x)$ v.s $f(x)$ is drawn.
- Detailed analysis: original problem with constraint $c_j(x) \leq 0$ is replaced with biobjective problem

$$\min_{x \in \Omega_j} \quad (c_j(x),\ f(x))$$
$$\text{s.t.} \quad \underline{c}_j \leq c_j(x) \leq \overline{c}_j$$

where Ω_j is the feasible set Ω minus the constraint.


Sensitivity to x1-2 ≤ 0
26.5
26.4
26.3
26.2
26.1
26
25.9
25.8
objective function value
-0.12
-0.1
-0.08
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
constraint value


Sensitivity to active bound consraints
x 10^7
-3.275
-3.28
-3.285
-3.29
-3.295
-3.3
-3.305
-1.5
-1
-0.5
0
0.5
constraint value
x_1 -100 ≤ 0
x_3 -100 ≤ 0
-x_5 ≤ 0

pMADS

- Idea: simply evaluate the trial points in parallel.
- Synchronous version:
  - The iteration is ended only when all the evaluations in progress are terminated.
  - Processes can be idle between two evaluations.
  - The algorithm is identical to the scalar version.
- Asynchronous version:
  - 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 are considered when they are finished.
  - The algorithm is slightly reorganized.

PSD-MADS

- **PSD:** Parallel Space Decomposition.
- 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, Tsitsiklis 1989] and on the Parallel Variable Distribution [Ferris, Mangasarian 1994].
- Objective: solve larger problems ($\simeq 50 - 500$ instead of $\simeq 10 - 20$).
- Asynchronous method.
- Convergence analysis.

MADS overview

Selected algorithmic features

NOMAD

Discussion

NOMAD (Nonlinear Optimization with MADS)

- C++ implementation of MADS.
- Standard C++, no other package needed.
- Runs on Linux, Unix, Mac OS X and Windows.
- Command-line and library interfaces.
- Distributed under the LGPL license.
- User guide and Doxygen documentation.

Functionalities (1/2)

- Single or Biobjective optimization.
- Variables:
  - Continuous, integer, binary, categorical.
  - Periodic.
  - Fixed.
  - Groups of variables.
- Searches:
  - Latin-Hypercube (LH).
  - Variable Neighborhood Search (VNS).
  - User search.

Functionalities (2/2)

- Constraints treated with 4 different methods:
  - Extreme Barrier.
  - Progressive Barrier (default).
  - Progressive-to-Extreme Barrier.
  - Filter method.
- Several direction types:
  - Coordinate directions.
  - LT-MADS.
  - OrthoMADS.
  - Hybrid combinations.
- Non adaptive surrogate functions: used to order the poll trial points, in VNS search, and in the extended poll for categorical variables.

(all items correspond to published or submitted papers).

Examples and tools included in the NOMAD package

- Examples include: Multi-start, GUI in java, interfaces with Windows DLL, AMPL, GAMS, CUTEr, MATLAB, FORTRAN.
- Tools: PSD-MADS, COOP-MADS and sensitivity analysis tools.

MADS overview

Selected algorithmic features

NOMAD

Discussion

Discussion

- We have presented the MADS algorithm, some of its features, and the NOMAD implementation.
- Future versions of NOMAD should include adaptive surrogates.
- References:
  - MADS references available from the NOMAD website.
  - NOMAD paper in TOMS accepted this morning :-)
  - NOMAD tests: Globalizations strategies (COAP) and OrthoMADS (SIOPT).