Blackbox optimization: Part 2/4: Algorithms

Sébastien Le Digabel

GERAD
GROUPE D'ÉTUDES ET DE RECHERCHE EN ANALYSE DES DÉCISIONS

POLYTECHNIQUE MONTRÉAL
TECHNOLOGICAL UNIVERSITY

ETICS 2023

BBO research team at GERAD/Polytechnique

Professors (C. Audet, Y. Diouane and SLD)

Research associate (C. Tribes)

Postdocs / Students / Graduates

Presentation outline

Introduction

The MADS algorithm

MADS features

The NOMAD software package

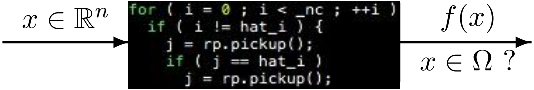
Conclusion

Blackbox / Derivative-Free Optimization

We consider

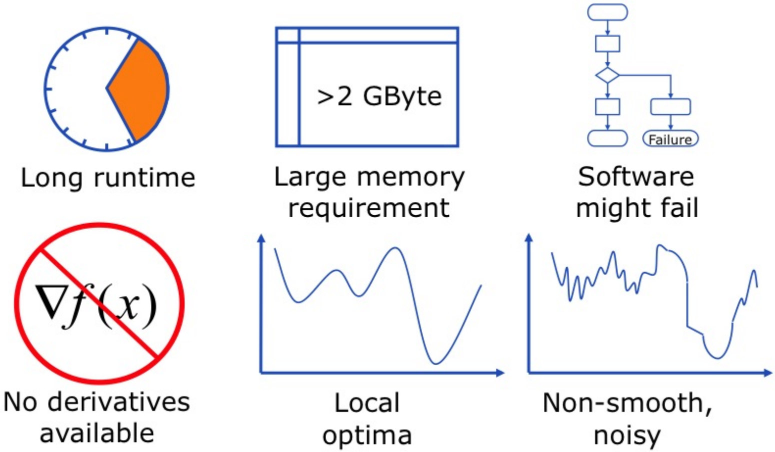
$$\min_{x \in \Omega} \quad f(x)$$

where the evaluations of f and the functions defining Ω are the result of a computer simulation (a blackbox)



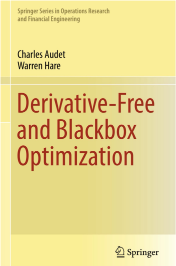
- Each call to the simulation may be expensive
- The simulation can fail
- Sometimes $f(x) \neq f(x)$
- Derivatives are not available and cannot be approximated

Blackboxes as illustrated by a Boeing engineer



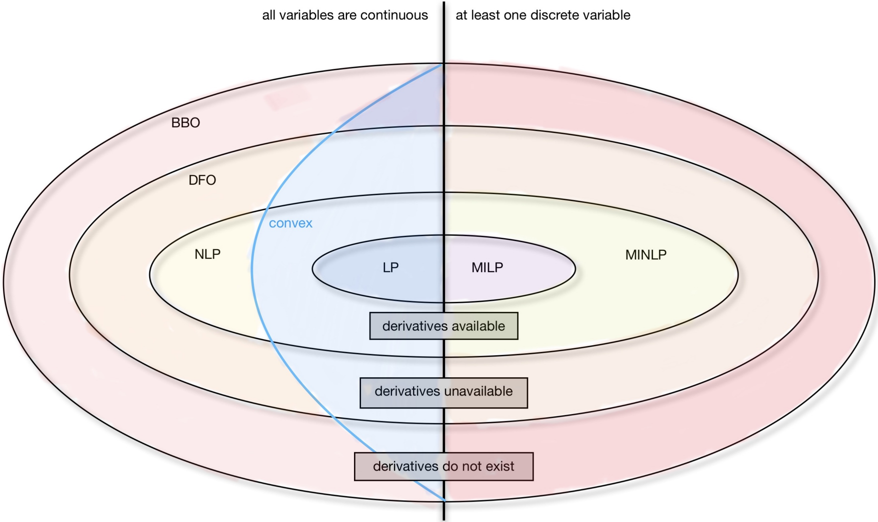
Terms

- *"Derivative-Free Optimization (DFO) is the mathematical study of optimization algorithms that do not use derivatives"* [Audet and Hare, 2017]
  - Optimization without using derivatives
  - Derivatives may exist but are not available
  - Obj./constraints may be analytical or given by a blackbox



- *"Blackbox Optimization (BBO) is the study of design and analysis of algorithms that assume the objective and/or constraints functions are given by blackboxes"* [Audet and Hare, 2017]
  - A simulation, or a blackbox, is involved
  - Obj./constraints may be analytical functions of the outputs
  - Derivatives may be available (ex.: PDEs)
  - Sometimes referred as *Simulation-Based Optimization (SBO)*

Optimization: Global view



Introduction

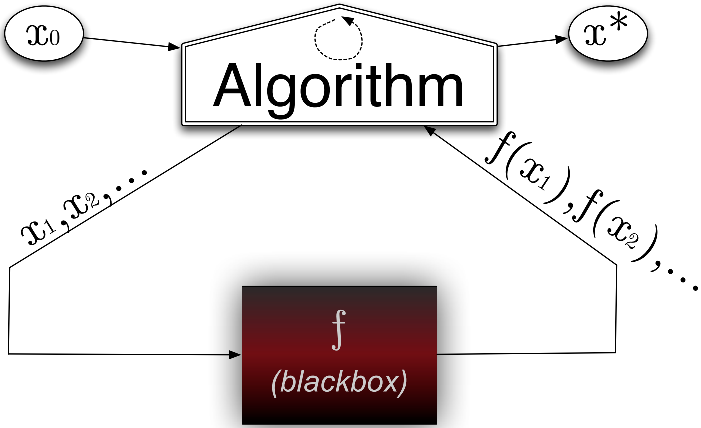
The MADS algorithm

MADS features

The NOMAD software package

Conclusion

Typical setting



Unconstrained case, with one initial starting solution

Algorithms for blackbox optimization

A method for blackbox optimization should ideally:

- Be efficient given a limited budget of evaluations
- Be robust to noise and blackbox failures
- Natively handle general constraints
- Deal with multiobjective optimization
- Deal with integer and categorical variables
- Easily exploit parallelism
- Have a publicly available implementation
- Have convergence properties ensuring first-order local optimality in the smooth case – otherwise why using it on more complicated problems?

Families of methods

- *"Computer science"* methods:
  - Heuristics such as genetic algorithms
  - No convergence properties
  - Cost a **lot** of evaluations
  - Should be used only in last resort for desperate cases
- Statistical methods:
  - Design of experiments
  - Bayesian optimization: EGO algorithm based on surrogates and expected improvement
  - Still limited in terms of dimension
  - Does not natively handle constraints
  - Good to use these tools in conjonction with DFO methods
- Derivative-Free Optimization methods (DFO)

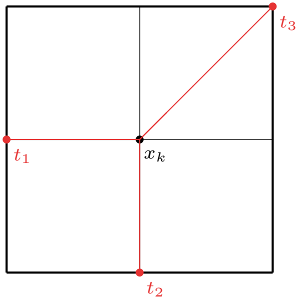
DFO methods

- **Model-based methods:**
  - Derivative-Free Trust-Region methods
  - Based on quadratic models or radial-basis functions
  - Use of a trust-region
  - Better for $\{$ DFO $\setminus$ BBO $\}$
  - Not resilient to noise and *hidden constraints*
  - Not easy to parallelize
- **Direct-search methods:**
  - Classical methods: Coordinate search, Nelder-Mead – the *other* simplex method
  - Modern methods: Generalized Pattern Search, Generating Set Search, Mesh Adaptive Direct Search (MADS)

So far, the size of the instances (variables and constraints) is typically limited to $\simeq 50$, and we target local optimization

MADS illustration with $n = 2$: Poll step

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



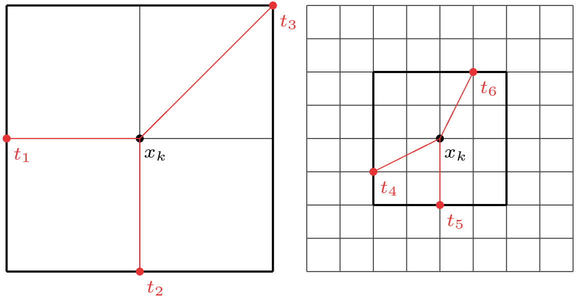
poll trial points=$\{t_1, t_2, t_3\}$

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

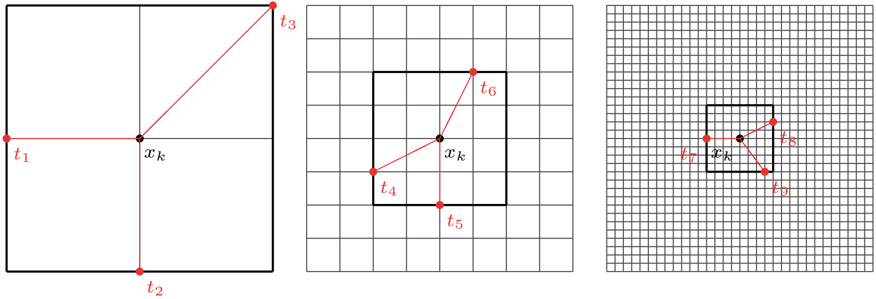
$\delta^k = \Delta^k = 1$

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

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

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

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



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

[0] Initializations (x_0, δ^0)
[1] Iteration k
- **[1.1] Search** (flexible part)
  - select a finite number of mesh points
  - evaluate candidates opportunistically
- **[1.2] Poll** (if Search failed) ("rigid" part)
  - construct poll set $P_k = \{x_k + \delta^k d : d \in D_k\}$
  - sort(P_k)
  - evaluate candidates opportunistically

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

The MADS algorithm [Audet and Dennis, Jr., 2006]

Hierarchical convergence

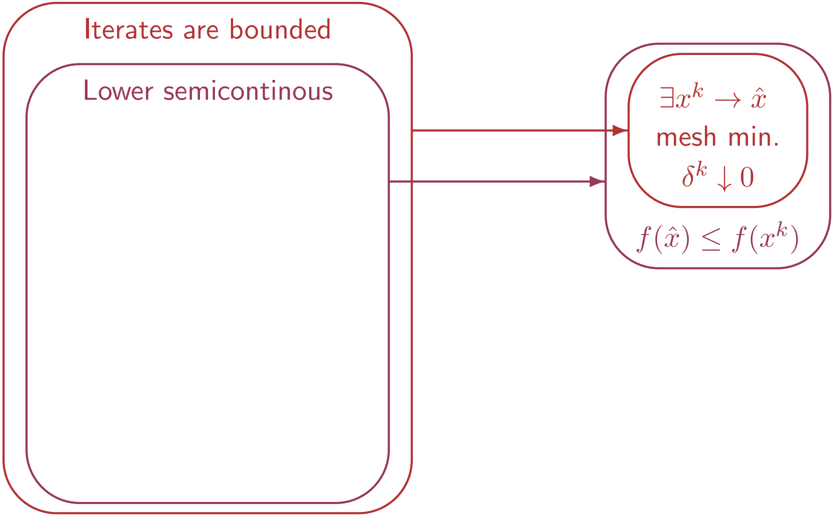
Iterates are bounded

$\exists x^k \to \hat{x}$

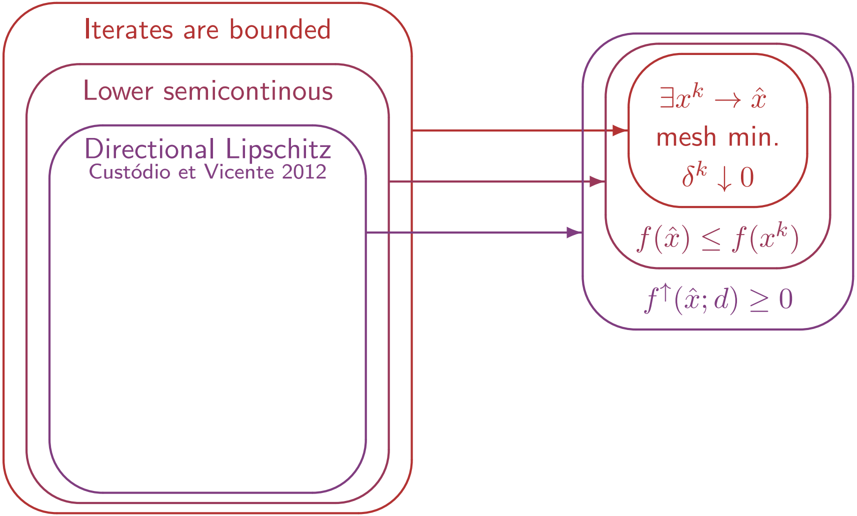
mesh min.

$\delta^k \downarrow 0$

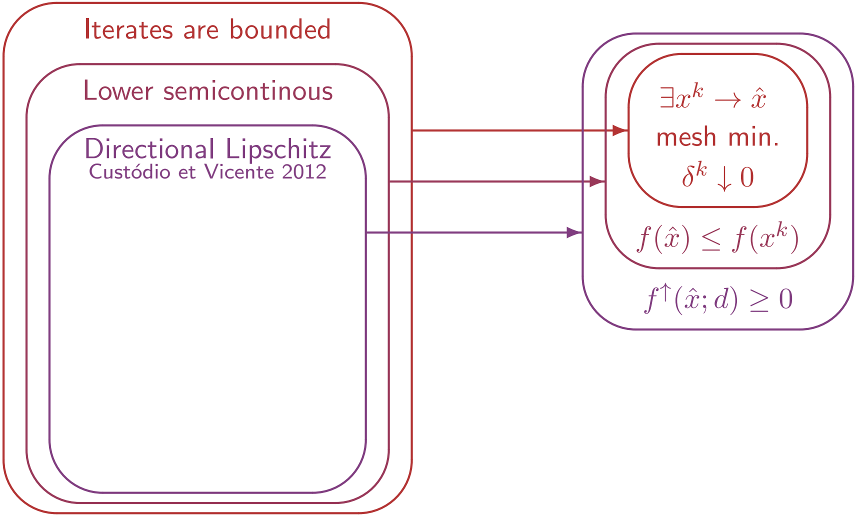
Hierarchical convergence



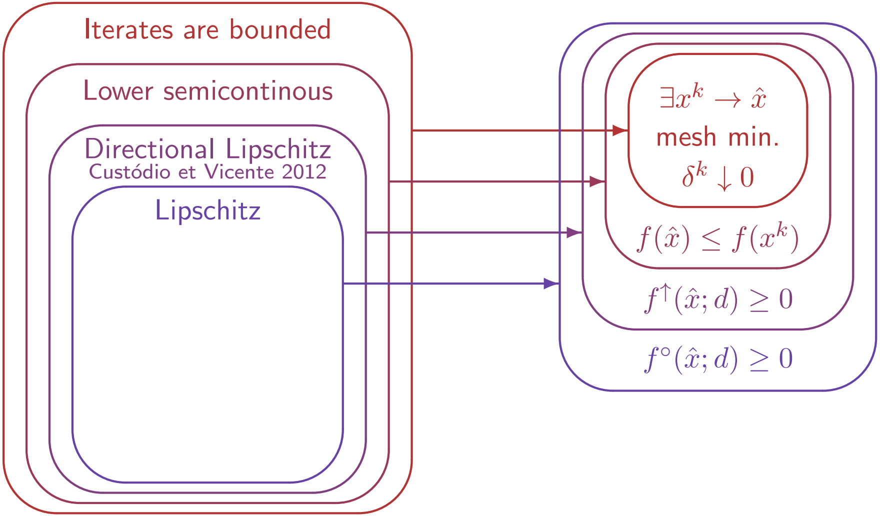
Hierarchical convergence



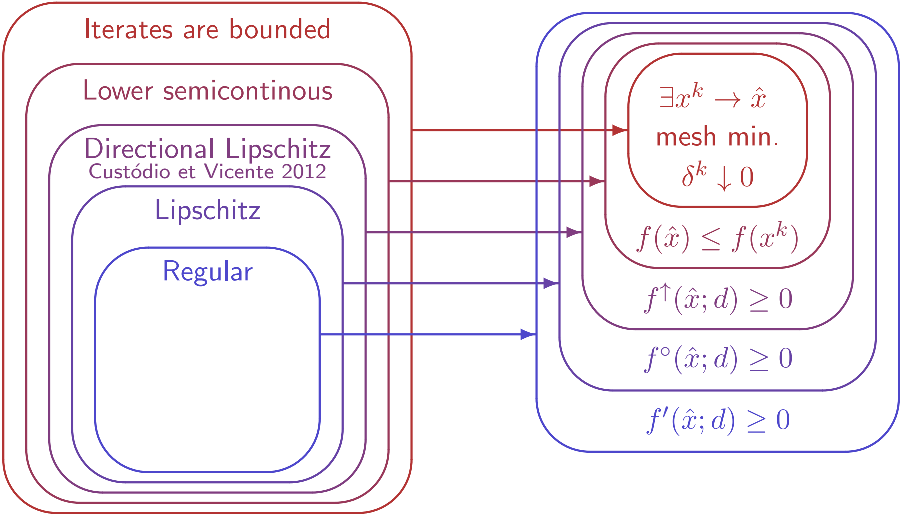
Hierarchical convergence



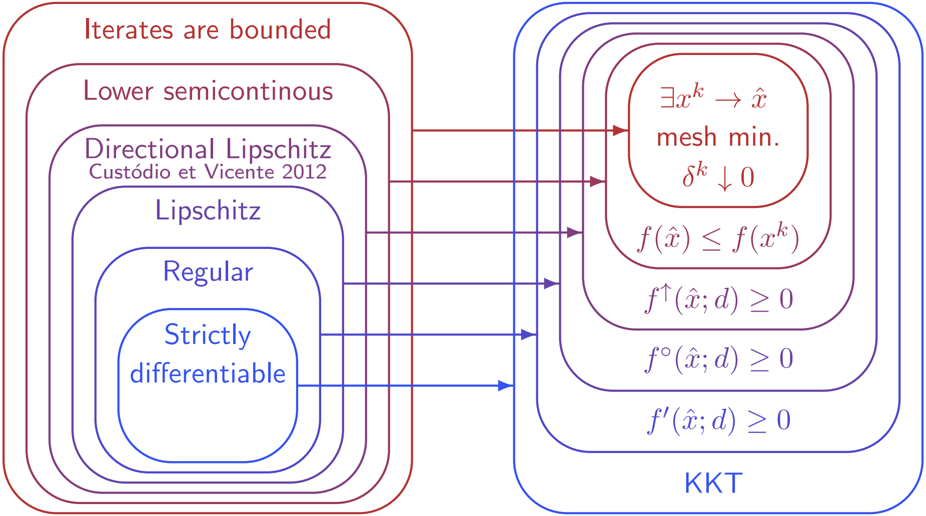
Hierarchical convergence



Hierarchical convergence



Hierarchical convergence



Special features of MADS

- Constraints handling with the Progressive Barrier technique [Audet and Dennis, Jr., 2009]
- Surrogates [Talgorn et al., 2015]
- Categorical/Meta variables [Audet et al., 2023]
- Granular and discrete variables [Audet et al., 2019]
- Global optimization [Audet et al., 2008a]
- Parallelism [Le Digabel et al., 2010, Audet et al., 2008b]
- Multiobjective optimization [Audet et al., 2008c, Bigeon et al., 2021]
- Sensitivity analysis [Audet et al., 2012]
- Handling of stochastic blackboxes [Alarie et al., 2021, Audet et al., 2021]

Introduction

The MADS algorithm

MADS features

The NOMAD software package

Conclusion

MADS features

In the following slides, we focus on these MADS features:

- Constraints handling
- Granular variables
- Surrogates
- Multiobjective optimization
- Parallelism

Constraints – with taxonomy of [Le Digabel and Wild, 2023]

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

- $\mathcal{X}$ corresponds to unrelaxable constraints

  Cannot be violated;

  Example: $x > 0$ when $\log x$ is used inside the simulation

Constraints – with taxonomy of [Le Digabel and Wild, 2023]

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

- $\mathcal{X}$ corresponds to unrelaxable constraints
- $c_j(x) \leq 0$: Relaxable and quantifiable constraints

May be violated at intermediate designs

$c_j(x)$ measures the violation

Example: cost $\leq$ budget

Constraints – with taxonomy of [Le Digabel and Wild, 2023]

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

- $\mathcal{X}$ corresponds to unrelaxable constraints
- $c_j(x) \leq 0$: Relaxable and quantifiable constraints
- Hidden constraints

  when the simulation fails, even for points in Ω

  Example:

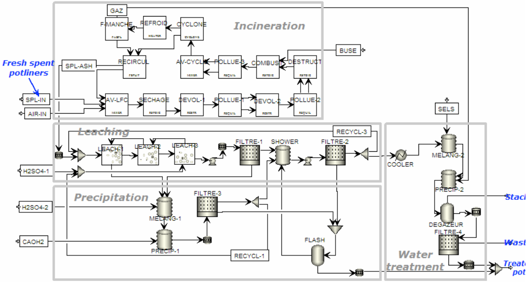
  ```
  Segmentation fault
  Bus error
  ERROR 42
  DIVISION BY ZERO
  ```

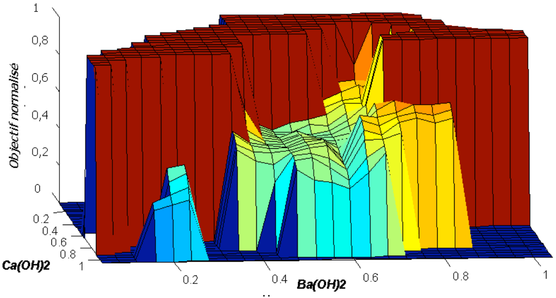
Constraints – with taxonomy of [Le Digabel and Wild, 2023]

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

- $\mathcal{X}$ corresponds to unrelaxable constraints
- $c_j(x) \leq 0$: Relaxable and quantifiable constraints
- Hidden constraints

Example: Chemical process:





7 variables, 4 constraints. The ASPEN software fails on 43% of the calls

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: this strategy can also be applied to a priori constraints in order to avoid the costly evaluation of $f(x)$

Three strategies to deal with constraints

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

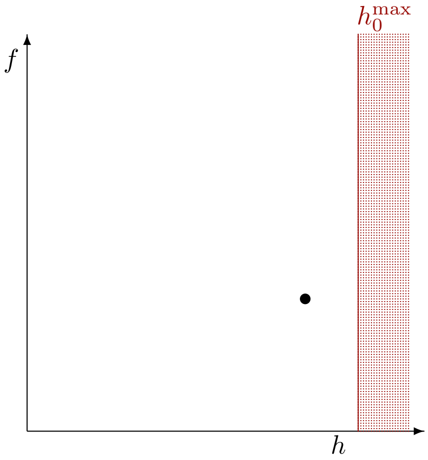
  Defined for relaxable and quantifiable constraints.
  As in the filter methods of Fletcher and Leyffer, it uses the non-negative constraint violation function $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$

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

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

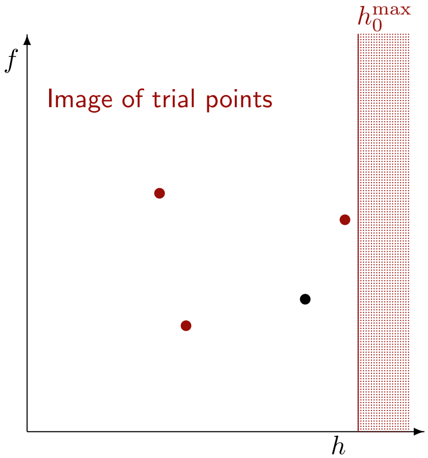
Three strategies to deal with constraints

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



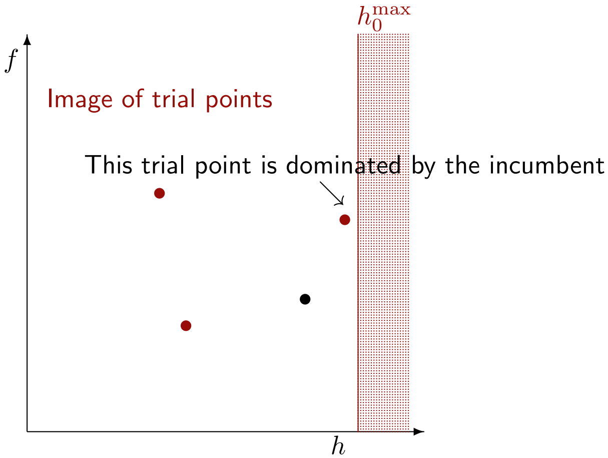
Three strategies to deal with constraints

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



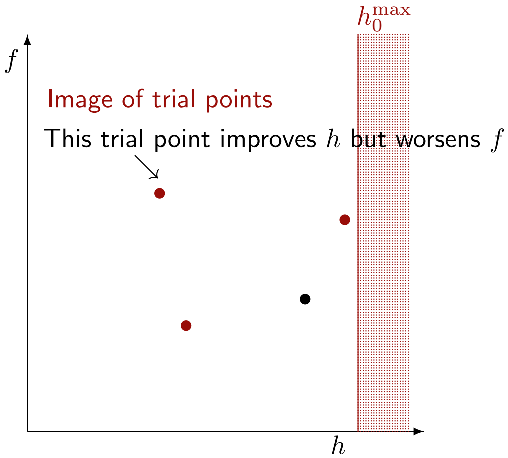
Three strategies to deal with constraints

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



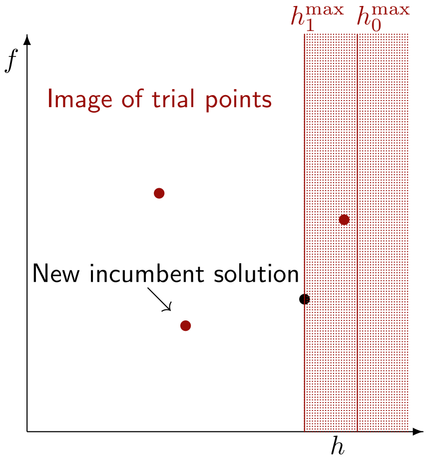
Three strategies to deal with constraints

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



Three strategies to deal with constraints

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



Three strategies to deal with constraints

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

  Initially treats a relaxable+quantifiable 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

Discrete variables in MADS

- MADS has been designed for continuous variables
- Some theory exists for categorical variables [Abramson, 2004]
- So far: Only a patch allows to handle integer variables: Rounding + minimal mesh size of one
- In [Audet et al., 2019], we present direct search methods with a natural way of handling discrete variables
- This lead to a new way of handling the mesh for a controlled number of decimals $\rightarrow$ granular variables

Mesh refinement on $\min(x-1/3)^2$

Δ^k	x^k
1	0
0.5	0.5
0.25	0.25
0.125	0.375
0.0625	0.3125
0.03125	0.34375
0.015625	0.328125
0.0078125	0.3359375
0.00390625	0.33203125
0.001953125	0.333984375

Mesh refinement on $\min(x - 1/3)^2$

Δ^k	x^k
1	0
0.5	0.5
0.25	0.25
0.125	0.375
0.0625	0.3125
0.03125	0.34375
0.015625	0.328125
0.0078125	0.3359375
0.00390625	0.33203125
0.001953125	0.333984375

alternately

Δ^k	x^k
1	0
0.5	0.5
0.2	0.4
0.1	0.3
0.05	0.35
0.02	0.34
0.01	0.33
0.005	0.335
0.002	0.332
0.001	0.333

Idea:

Instead of dividing Δ^k by 2, change it so that

10×10^b refines to 5×10^b

5×10^b refines to 2×10^b

2×10^b refines to 1×10^b

Mesh refinement on $\min(x - 1/3)^2$

Δ^k	x^k
1	0
0.5	0.5
0.25	0.25
0.125	0.375
0.0625	0.3125
0.03125	0.34375
0.015625	0.328125
0.0078125	0.3359375
0.00390625	0.33203125
0.001953125	0.333984375

alternately

Δ^k	x^k
1	0
0.5	0.5
0.2	0.4
0.1	0.3
0.05	0.35
0.02	0.34
0.01	0.33
0.005	0.335
0.002	0.332
0.001	0.333

Idea: Instead of dividing Δ^k by 2, change it so that

10×10^b refines to 5×10^b

5×10^b refines to 2×10^b

2×10^b refines to 1×10^b

To get three decimals, one simply sets the granularity to 0.001. Integer variables are treated by setting the granularity to $\mathcal{G} = 1$

Poll and mesh size parameter update

- The poll size parameter Δ^k is updated as
  $10 \times 10^b \quad \longleftrightarrow \quad 5 \times 10^b \quad \longleftrightarrow \quad 2 \times 10^b \quad \longleftrightarrow \quad 1 \times 10^b$
- The fine underlying mesh is defined with the mesh size parameter
  $$\delta^k = \begin{cases} 1 & \text{if } \Delta^k \geq 1 \\ \max\{10^{2b}, \mathcal{G}\} & \text{otherwise, i.e. } \Delta^k \in \{1,2,5\} \times 10^b \end{cases}$$
- Example: Granularity of $\mathcal{G} = 0.005$:

δ^k	Δ^k
1	5
1	2
1	1
0.01	0.5
0.01	0.2
0.01	0.1
0.005	0.05
0.005	0.02
0.005	0.01
0.005	0.005 $\leftarrow$ `stop`

Static versus dynamic surrogates

- Static surrogate: A cheaper model defined a priori by the user. It is used as a blackbox. Typically a simplified physics model. Variable fidelity may be considered.
- Dynamic surrogate: Model managed by the algorithm, based on past evaluations. It can be periodically updated.

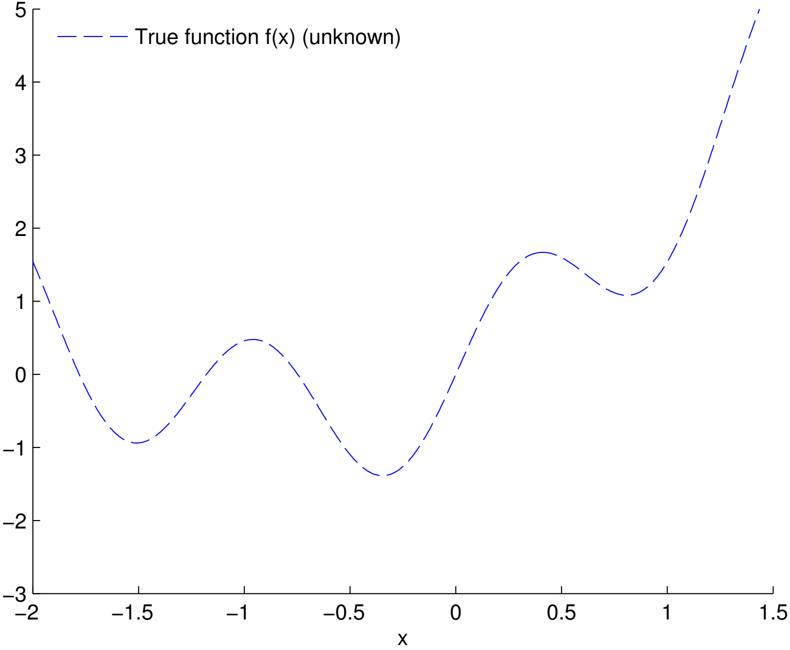
In the remaining, we focus on dynamic surrogates

Surrogate-assisted optimization

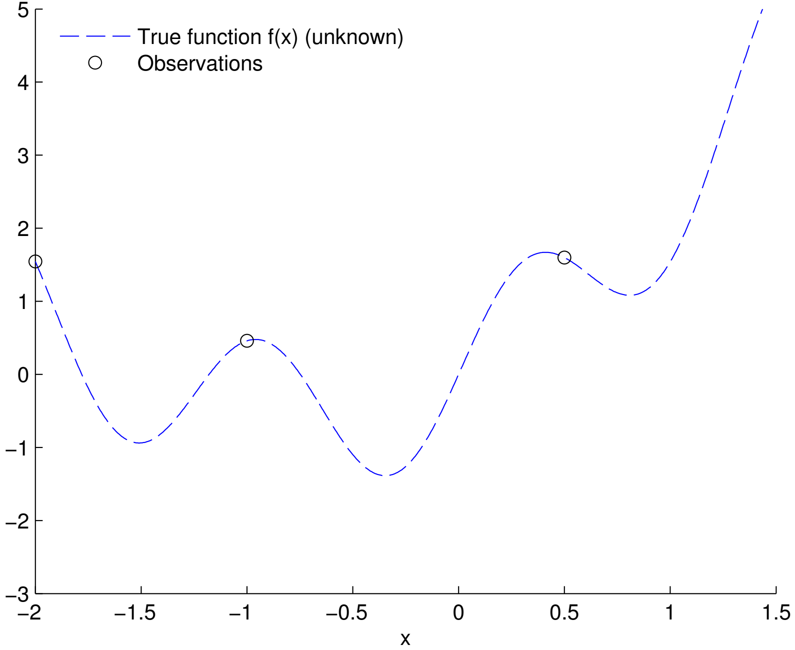
1. Use $[\mathbf{X}, f(\mathbf{X})]$ to build a surrogate $\hat{f}$ of the function f
2. Find $x_S \in \underset{x}{\operatorname{argmin}}\ \hat{f}(x)$ (or minimize another criteria such as the EI)
3. Evaluate $f(x_S)$
4. $\mathbf{X} \leftarrow \mathbf{X} \cup \{x_S\}$
5. Go back to Step **1.**

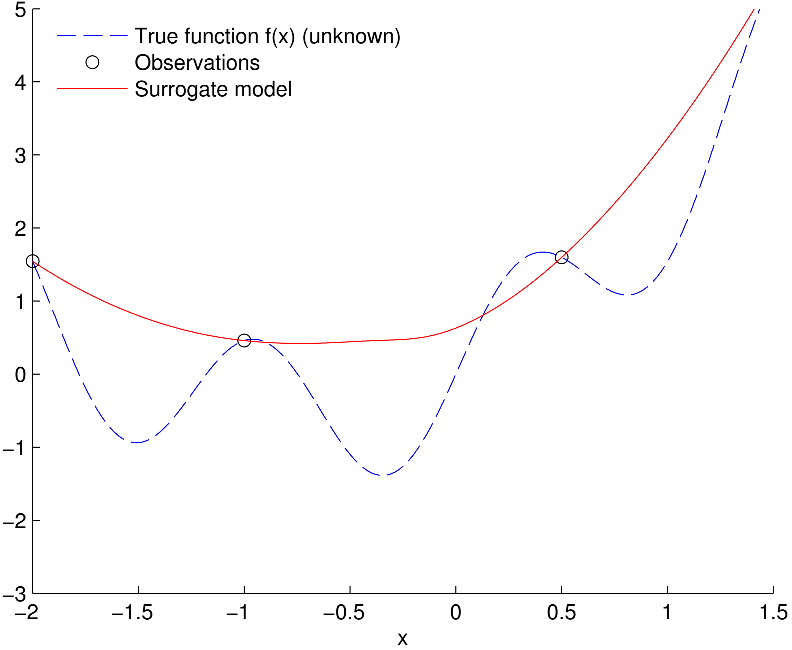
For constrained problems the same method can be used for constrained problems:

- Build the models of the constraints
- $x_S \leftarrow$ minimizer of $\hat{f}$ subject to the constraints $\hat{c}_j \leq 0$, $j = 1, 2, \ldots, m$

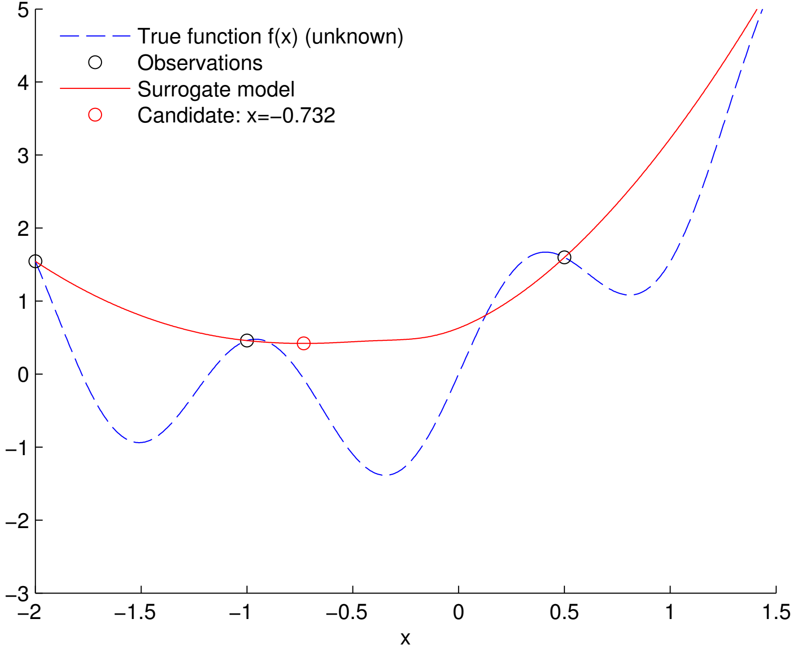


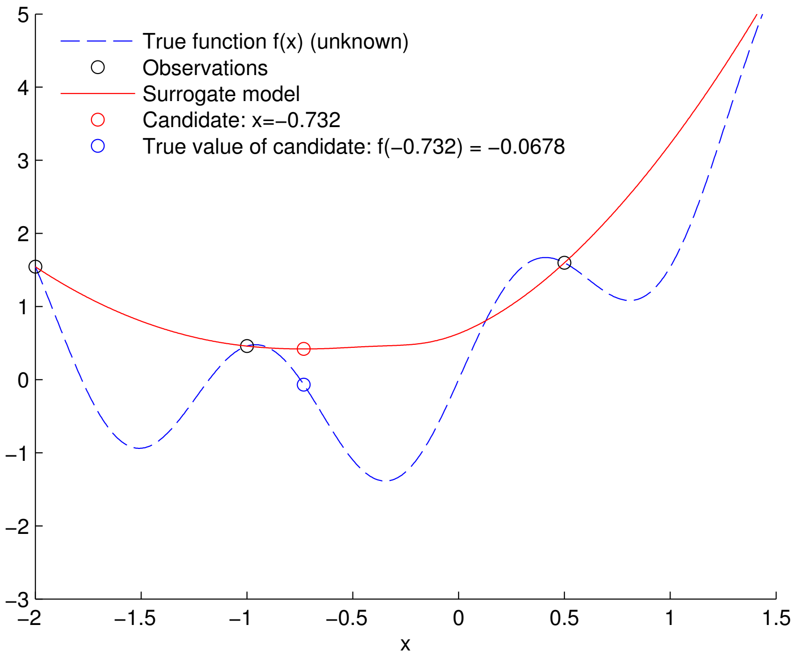
True function f(x) (unknown)
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

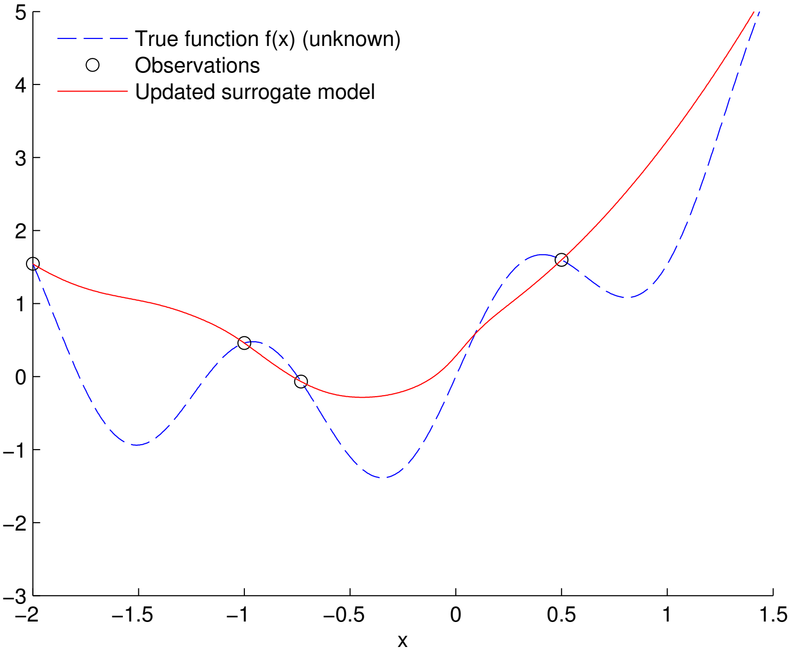

True function f(x) (unknown)
Observations
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

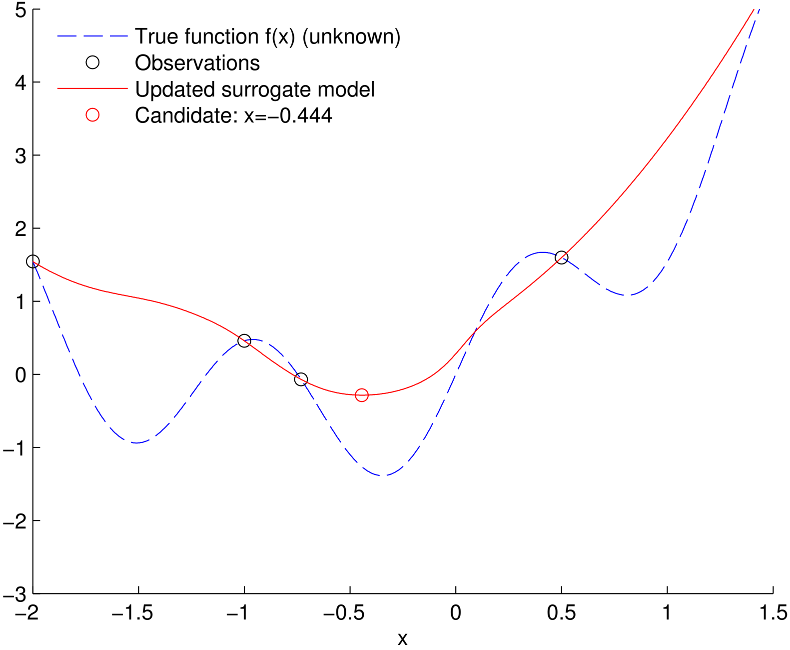


True function f(x) (unknown)
Observations
Surrogate model
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

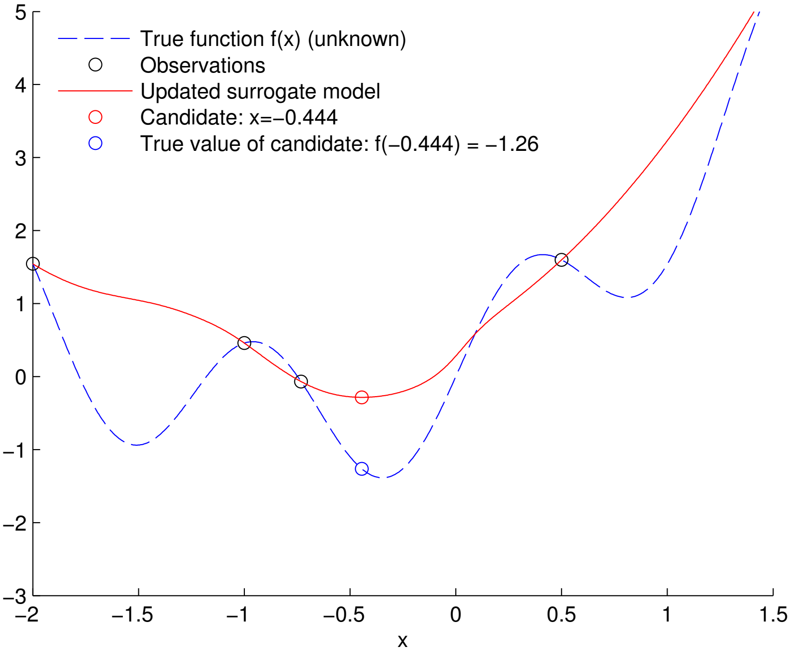

True function f(x) (unknown)
Observations
Surrogate model
Candidate: x=−0.732
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

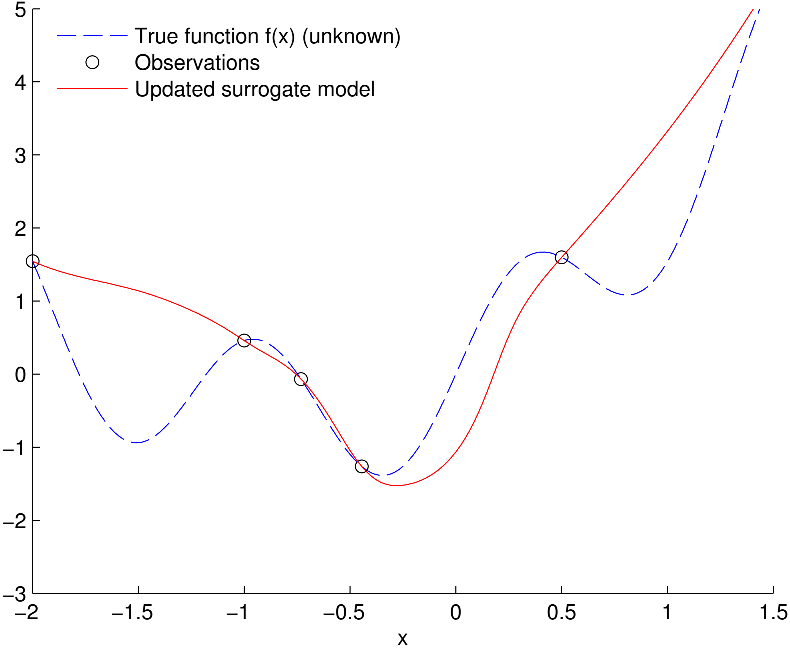

True function f(x) (unknown)
Observations
Surrogate model
Candidate: x=−0.732
True value of candidate: f(−0.732) = −0.0678
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x


True function f(x) (unknown)
Observations
Updated surrogate model
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

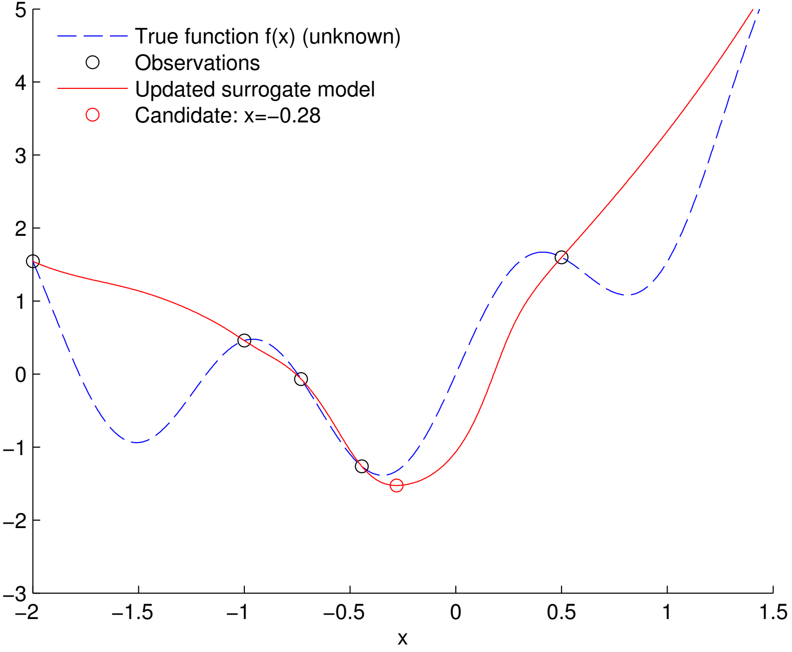


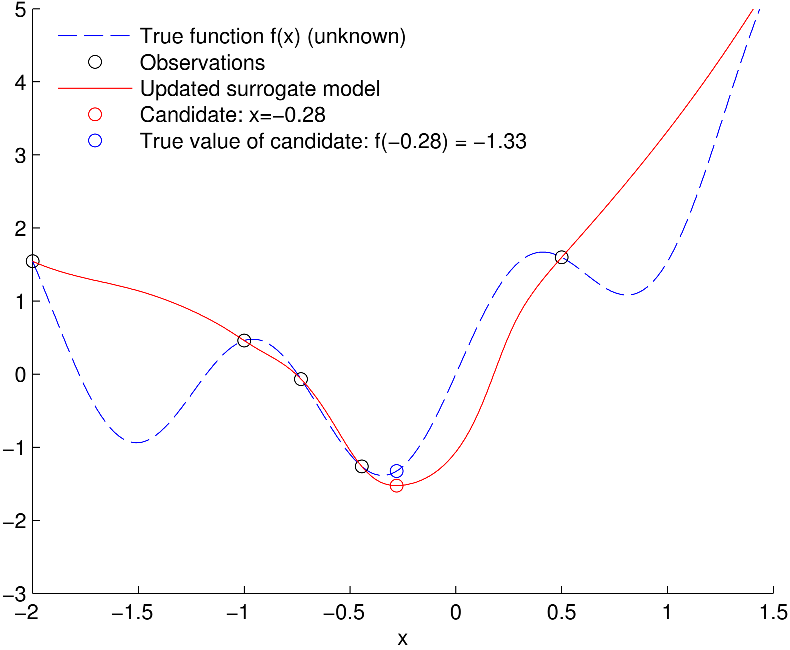
True function f(x) (unknown)
Observations
Updated surrogate model
Candidate: x=−0.444
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

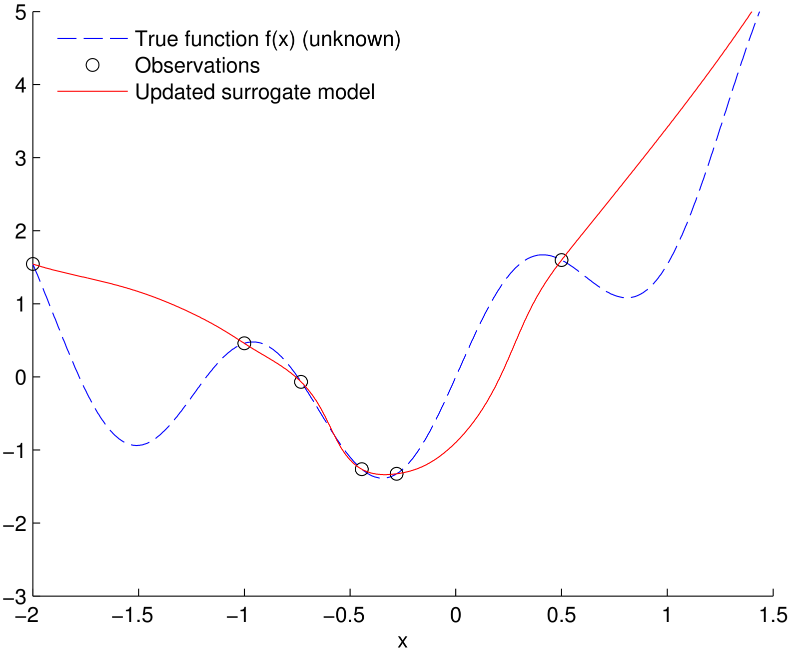

True function f(x) (unknown)
Observations
Updated surrogate model
Candidate: x=−0.444
True value of candidate: f(−0.444) = −1.26
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x



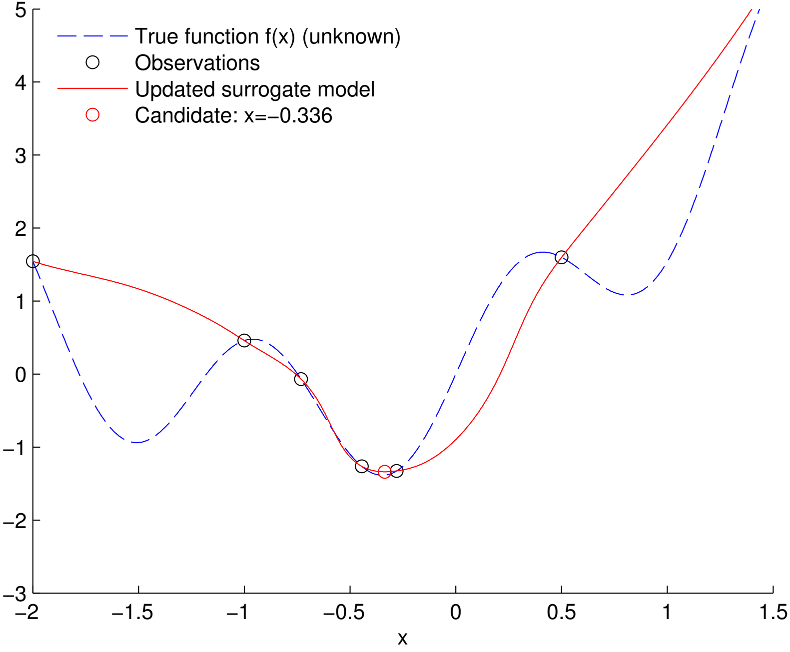
True function f(x) (unknown)
Observations
Updated surrogate model
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x


True function f(x) (unknown)
Observations
Updated surrogate model
Candidate: x=−0.28
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

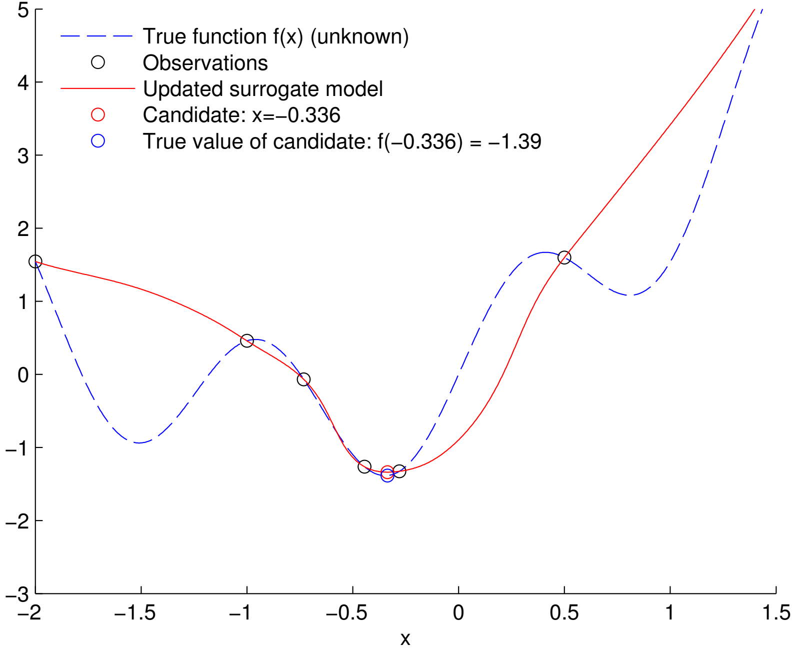

True function f(x) (unknown)
Observations
Updated surrogate model
Candidate: x=−0.28
True value of candidate: f(−0.28) = −1.33
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

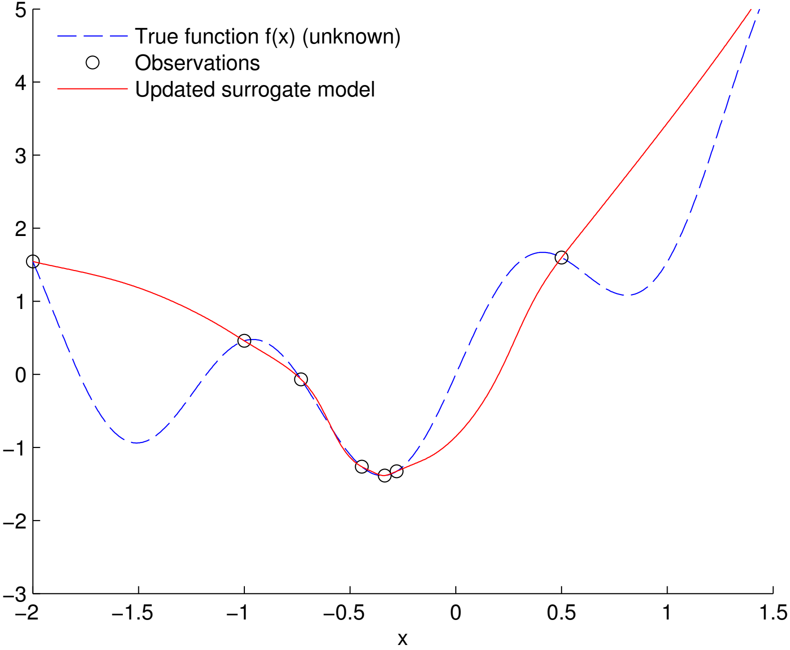


True function f(x) (unknown)
Observations
Updated surrogate model
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

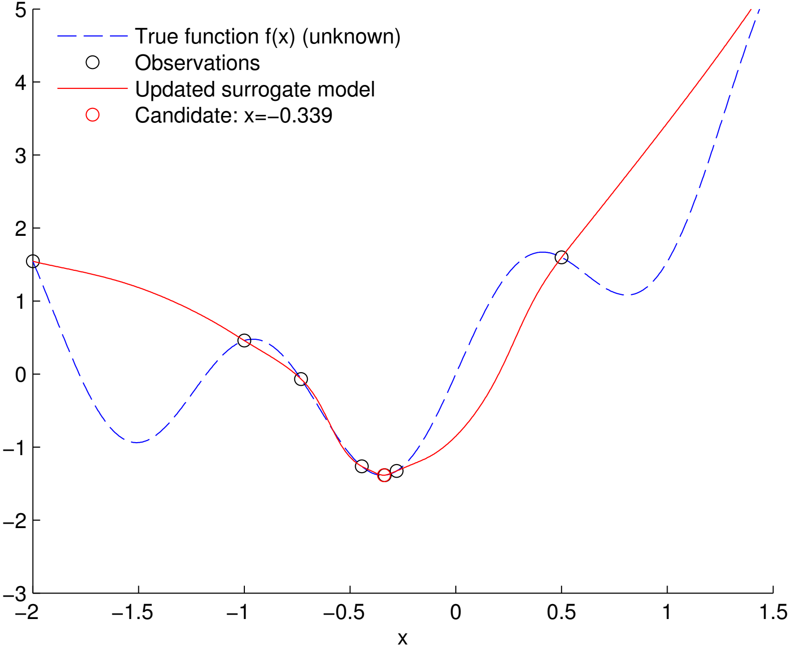


True function f(x) (unknown)
Observations
Updated surrogate model
Candidate: x=−0.336
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x


True function f(x) (unknown)
Observations
Updated surrogate model
Candidate: x=−0.336
True value of candidate: f(−0.336) = −1.39
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x



True function f(x) (unknown)
Observations
Updated surrogate model
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x


True function f(x) (unknown)
Observations
Updated surrogate model
Candidate: x=−0.339
5
4
3
2
1
0
−1
−2
−3
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x

Surrogate-assisted optimization in MADS

1. Initialization:
   - Initial design (x_0)
   - Initial mesh and poll sizes (δ^0, Δ^0)

2. Search
   - Build the **surrogates** $\hat{f}$ and $\{\hat{c}_j\}_{j=1,2,\ldots,m}$
   - $\mathbf{x}_S \leftarrow$ solution of the surrogate problem, projected on the current mesh
   - If $\mathbf{x}_S$ is a success, repeat the search

3. Poll
   - Construct the poll candidates
   - Use the **surrogates** to order the poll candidates
   - Evaluate the poll candidates *opportunistically*

4. If no stopping criteria is met, go back to Step 2.

What is a good model for surrogate-assisted optimization

- Good model of the objective f: respects the **order** between two candidates:

$$f(\mathbf{x}) \leq f(\mathbf{x}') \Leftrightarrow \hat{f}(\mathbf{x}) \leq \hat{f}(\mathbf{x}') \text{ for all } \mathbf{x}, \mathbf{x}' \in \mathcal{X}$$

- Good model of a constraint c_j: respects the **sign** of the function:

$$c_j(\mathbf{x}) \leq 0 \Leftrightarrow \hat{c}_j(\mathbf{x}) \leq 0 \text{ for all } \mathbf{x} \in \mathcal{X}$$

Multiobjective optimization

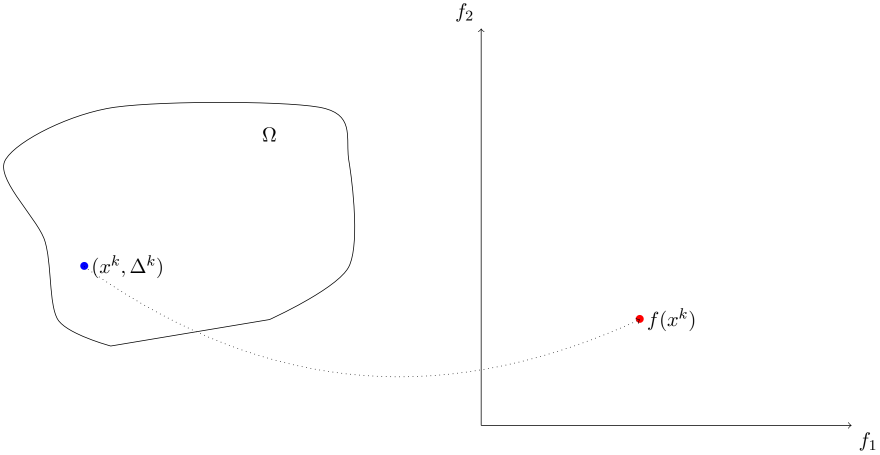
The problem:

$$\min_{x \in \Omega} f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$

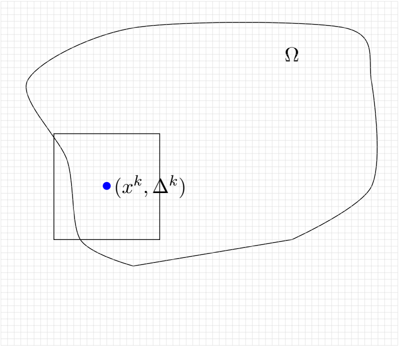
The DMulti-MADS algorithm [Bigeon et al., 2021]:

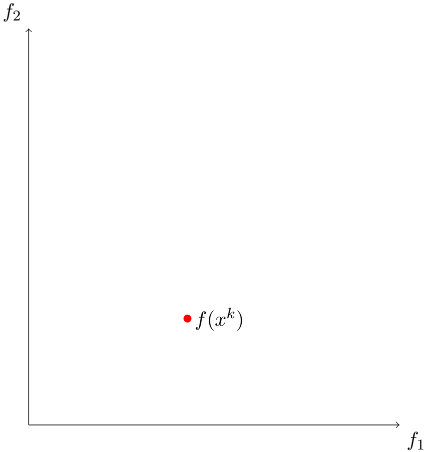
- Strongly inspired by DMS [Custódio et al., 2011] and BiMADS [Audet et al., 2008c]
- Handles more than 2 objectives
- Convergence to a set of locally Pareto optimal points
- Implemented in NOMAD v4 [Audet et al., 2022]

DMulti-MADS: an iteration

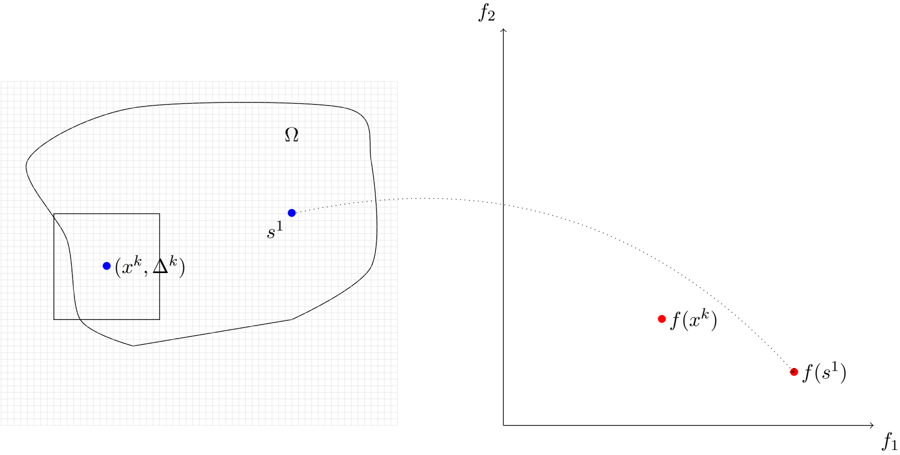


DMulti-MADS: an iteration



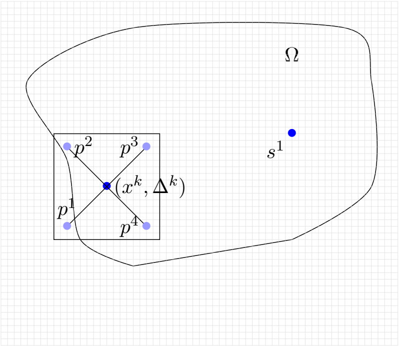


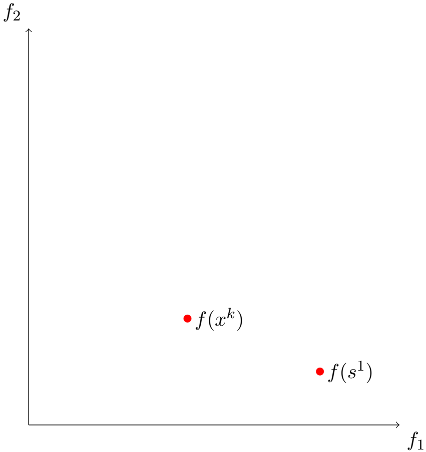
DMulti-MADS: an iteration



Search step

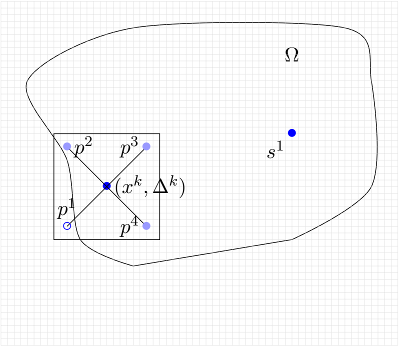
DMulti-MADS: an iteration

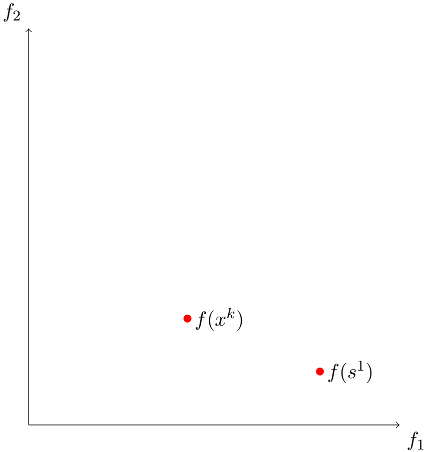




Poll step

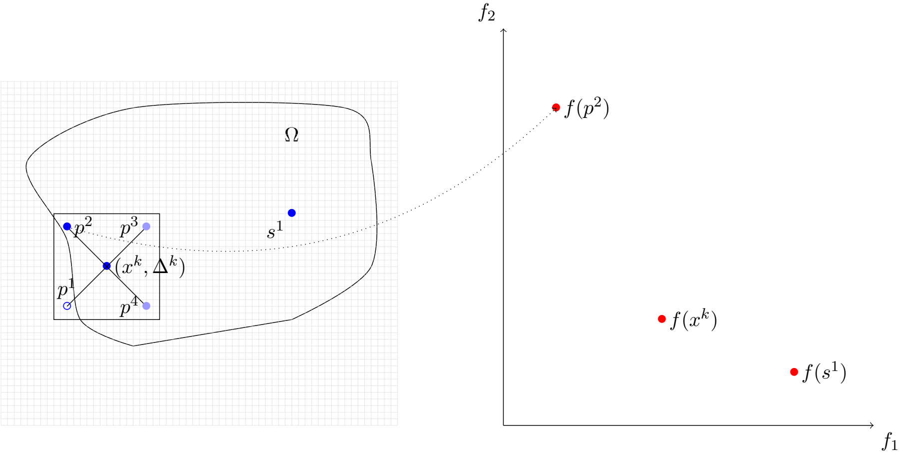
DMulti-MADS: an iteration





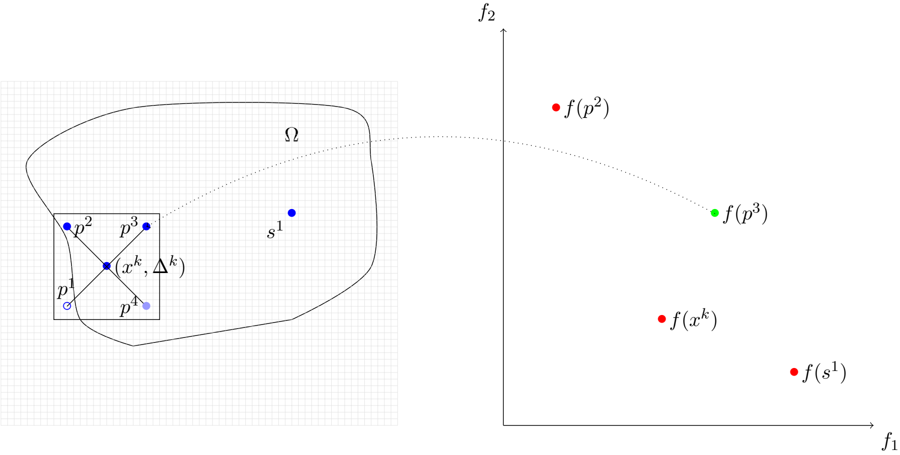
Poll step

DMulti-MADS: an iteration



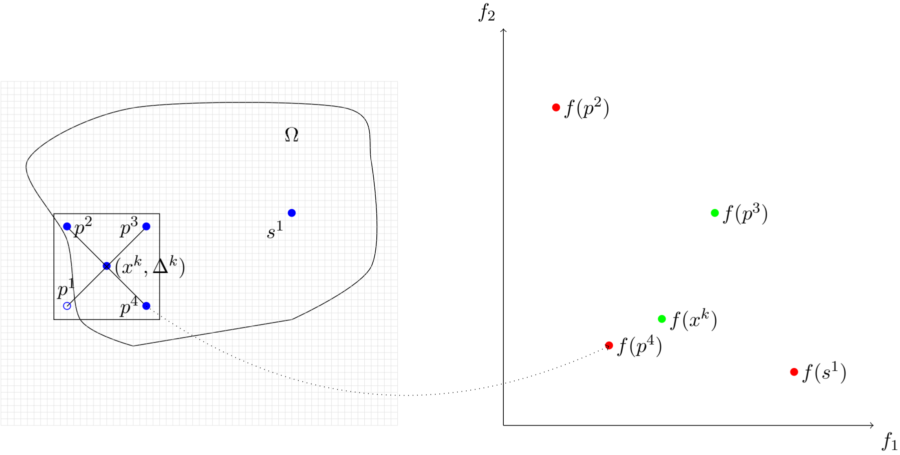
Poll step

DMulti-MADS: an iteration



Poll step

DMulti-MADS: an iteration



Poll step

First parallel method: pMADS

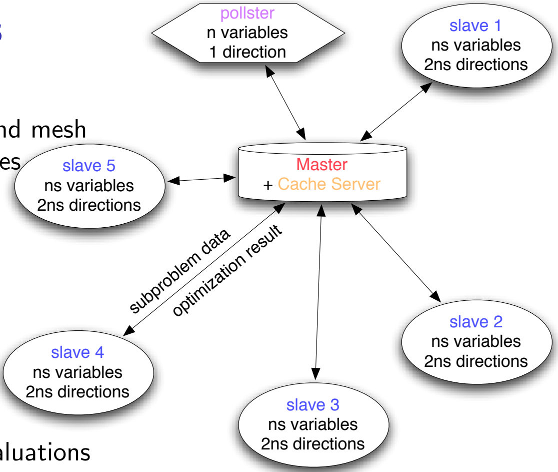
- Idea: simply evaluate the trial points in parallel
- Synchronous version:
  - The iteration is ended only when all the evaluations in progress are terminated
  - Processes can be idle between two evaluations
  - The algorithm is identical to the scalar version
- Asynchronous version:
  - If a new best 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 [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 $\simeq 10 - 20$)
- Asynchronous method
- Convergence analysis

PSD-MADS: processes

- Master
  - receives all slave's signals
  - updates current solution and mesh
  - decides subproblem variables
  - sends subproblem data
- Slaves
  - receive subproblem data
  - optimize subproblem
  - send optimization data
- Cache server
  - memorizes all blackbox evaluations
  - allows the "cache search" in slave processes



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
- MATLAB versions; Multiple interfaces (Python, Julia, etc.)
- Open and free – LGPL license
- Download at `https://www.gerad.ca/nomad`
- Support at `nomad@gerad.ca`
- Related articles in TOMS [Le Digabel, 2011] and [Audet et al., 2022]

Main functionalities (1/2)

- Single or biobjective optimization
- Variables:
  - Continuous, integer, binary, categorical, granular
  - Periodic
  - Fixed
  - Groups of variables
- Searches:
  - Latin-Hypercube
  - Variable Neighborhood Search
  - Nelder-Mead Search
  - Quadratic models
  - Statistical surrogates
  - User search

Main functionalities (2/2)

- Constraints treated with 4 different methods:
  - Progressive Barrier (default)
  - Extreme Barrier
  - Progressive-to-Extreme Barrier
  - Filter method
- Several direction types:
  - Coordinate directions
  - LT-MADS
  - OrthoMADS
  - Hybrid combinations
- Sensitivity analysis

$\rightarrow$ default values for all parameters

$\rightarrow$ all items correspond to published or submitted papers

Blackbox conception (batch mode)

- Command-line program that takes in argument a file containing x, and displays the values of $f(x)$ and the $c_j(x)$'s
- Can be coded in any language
- Typically: `> bb.exe x.txt` displays `f c1 c2` (objective and two constraints)

Run NOMAD

```
> nomad parameters.txt
```

```
[iota ~/Desktop/2018_UQAC_NOMAD/demo_NOMAD/mac] > ../nomad.3.8.1/bin/nomad parameters.txt

NOMAD - version 3.8.1 has been created by {
        Charles Audet         - Ecole Polytechnique de Montreal
        Sebastien Le Digabel - Ecole Polytechnique de Montreal
        Christophe Tribes     - Ecole Polytechnique de Montreal
}

The copyright of NOMAD - version 3.8.1 is owned by {
        Sebastien Le Digabel - Ecole Polytechnique de Montreal
        Christophe Tribes     - Ecole Polytechnique de Montreal
}

NOMAD v3 has been funded by AFOSR, Exxon Mobil, Hydro Québec, Rio Tinto and
IVADO.

NOMAD v3 is a new version of NOMAD v1 and v2. NOMAD v1 and v2 were created
and developed by Mark Abramson, Charles Audet, Gilles Couture, and John E.
Dennis Jr., and were funded by AFOSR and Exxon Mobil.

License    : '$NOMAD_HOME/src/lgpl.txt'
User guide: '$NOMAD_HOME/doc/user_guide.pdf'
Examples   : '$NOMAD_HOME/examples'
Tools      : '$NOMAD_HOME/tools'

Please report bugs to nomad@gerad.ca

Seed: 0

MADS run {

        BBE     OBJ

        4       0.0000000000
        21      -1.0000000000
        23      -3.0000000000
        51      -4.0000000000
        563     -4.0000000000

} end of run (mesh size reached NOMAD precision)

blackbox evaluations                     : 563
best infeasible solution (min. violation): ( 1.000000013 1.000000048 0.9999999797 0.999999992 -4 ) h=1.10134e-13 f=-4
best feasible solution                   : ( 1 1 1 1 -4 ) h=0 f=-4
```

Summary

- Blackbox optimization motivated by industrial applications
- Algorithmic features backed by mathematical convergence analyses and published in optimization journals
- NOMAD: Software package implementing MADS
- Open source; LGPL license
- Features: Constraints, biobjective, global optimization, surrogates, several types of variables, parallelism
- Fast support at nomad@gerad.ca
- NOMAD has become a baseline for benchmarking DFO algorithms

References I

Abramson, M. (2004).
Mixed Variable Optimization of a Load-Bearing Thermal Insulation System Using a Filter Pattern Search Algorithm.
Optimization and Engineering, 5(2):157–177.

Alarie, S., Audet, C., Bouchet, P.-Y., and Le Digabel, S. (2021).
Optimisation of stochastic blackboxes with adaptive precision.
SIAM Journal on Optimization, 31(4):3127–3156.

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.

Audet, C. and Dennis, Jr., J. (2009).
A Progressive Barrier for Derivative-Free Nonlinear Programming.
SIAM Journal on Optimization, 20(1):445–472.

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., Dennis, Jr., J., and Le Digabel, S. (2012).
Trade-off studies in blackbox optimization.
Optimization Methods and Software, 27(4–5):613–624.

References II

Audet, C., Dzahini, K., Kokkolaras, M., and Le Digabel, S. (2021).
Stochastic mesh adaptive direct search for blackbox optimization using probabilistic estimates.
Computational Optimization and Applications, 79(1):1–34.

Audet, C., Hallé-Hannan, E., and Le Digabel, S. (2023).
A General Mathematical Framework for Constrained Mixed-variable Blackbox Optimization Problems with Meta and Categorical Variables.
Operations Research Forum, 4(12).

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.

Audet, C., Le Digabel, S., and Tribes, C. (2019).
The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables.
SIAM Journal on Optimization, 29(2):1164–1189.

Audet, C., Savard, G., and Zghal, W. (2008c).
Multiobjective Optimization Through a Series of Single-Objective Formulations.
SIAM Journal on Optimization, 19(1):188–210.

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

References III

Bigeon, J., Le Digabel, S., and Salomon, L. (2021).
DMulti-MADS: Mesh adaptive direct multisearch for bound-constrained blackbox multiobjective optimization.
Computational Optimization and Applications, 79(2):301–338.

Custódio, A., Madeira, J., Vaz, A., and Vicente, L. (2011).
Direct multisearch for multiobjective optimization.
SIAM Journal on Optimization, 21(3):1109–1140.

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

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

Le Digabel, S., Abramson, M., Audet, C., and Dennis, Jr., J. (2010).
Parallel Versions of the MADS Algorithm for Black-Box Optimization.
In *Optimization days*, Montréal.
Slides available at `https://www.gerad.ca/Sebastien.Le.Digabel/talks/2010_JOPT_25mins.pdf`.

Le Digabel, S. and Wild, S. (2023).
A taxonomy of constraints in black-box simulation-based optimization.
Technical Report G-2015-57, Les cahiers du GERAD.
To appear in *Optimization and Engineering*.

References IV

Talgorn, B., Le Digabel, S., and Kokkolaras, M. (2015).
Statistical Surrogate Formulations for Simulation-Based Design Optimization.
Journal of Mechanical Design, 137(2):021405–1–021405–18.

Vicente, L. and Custódio, A. (2012).
Analysis of direct searches for discontinuous functions.
Mathematical Programming, 133(1-2):299–325.