Ecole Polytechnique de Montréal

Black-Box Optimization with the NOMAD Software

Sébastien Le Digabel
Mark Abramson
Charles Audet
John Dennis

2010–02–22

Presentation outline

MADS overview

Presentation outline

Presentation outline

MADS overview

Selected algorithmic features

NOMAD

Presentation outline

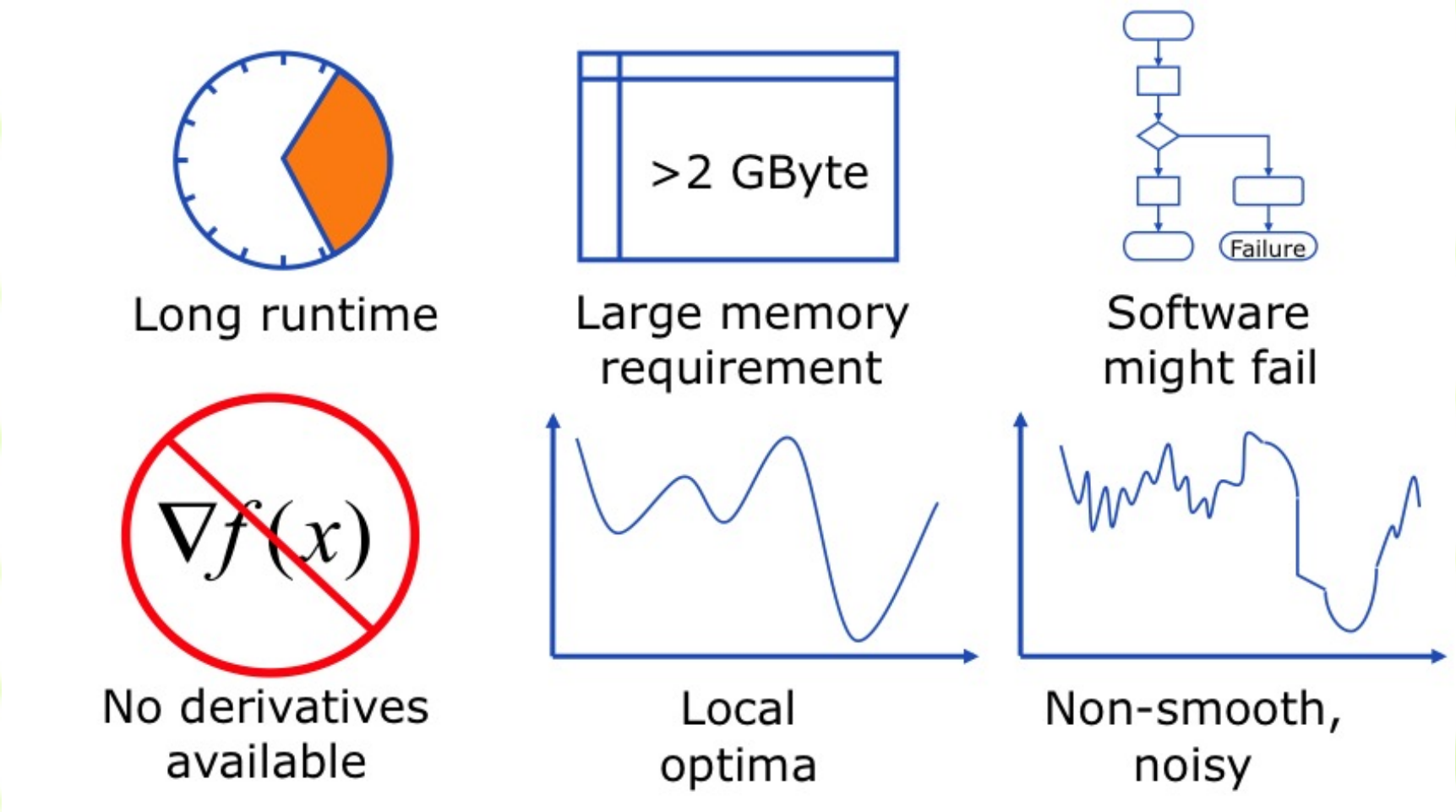
Black-box optimization problems

We consider the nonsmooth optimization problem:

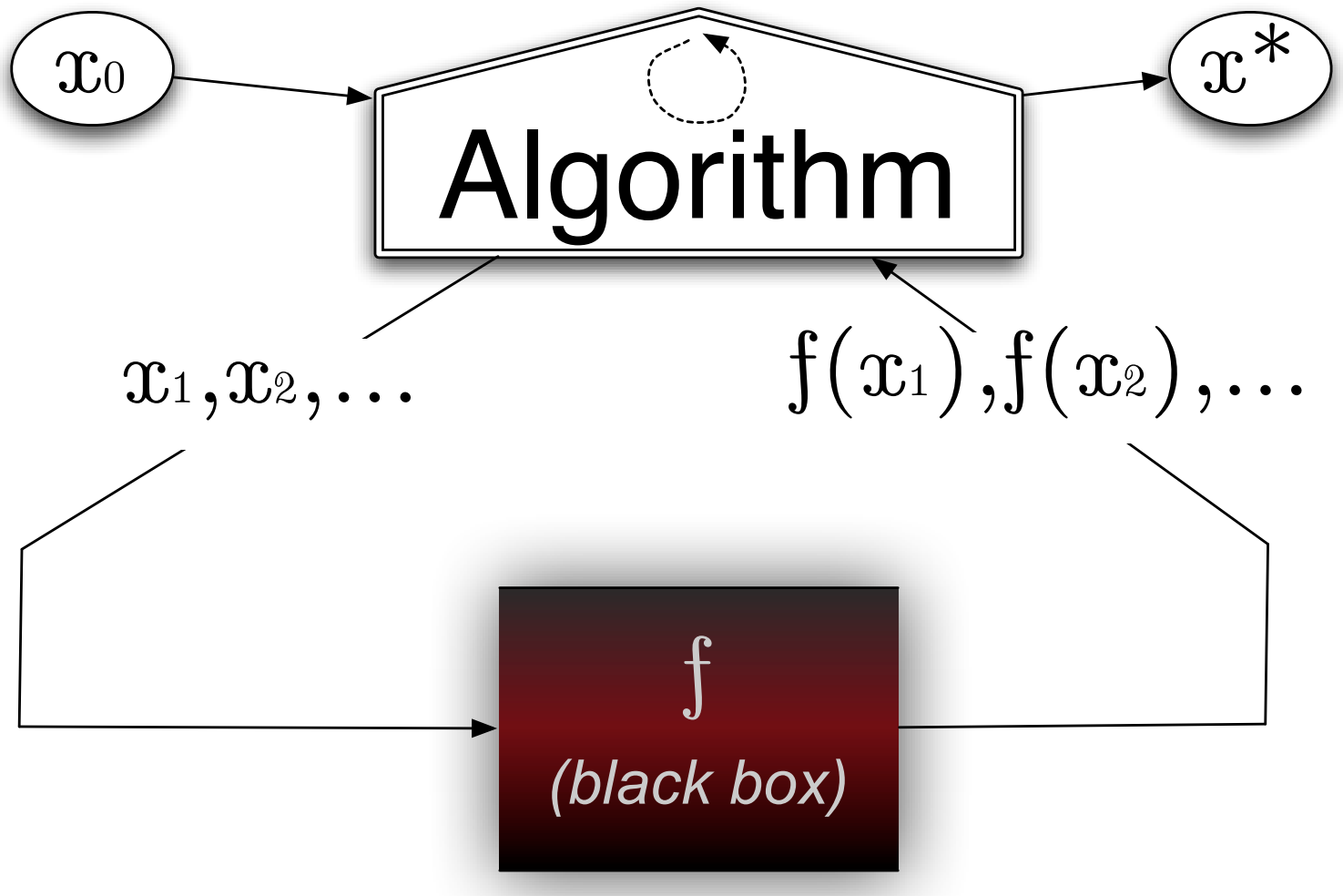
$$\begin{aligned} \text{Minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \Omega, \end{aligned}$$

where evaluation of the functions are usually the result of a computer code (a black-box)

Black-boxes as illustrated by J. Simonis [ISMP 2009]



MADS and direct search principle



Mesh Adaptive Direct Search (MADS)

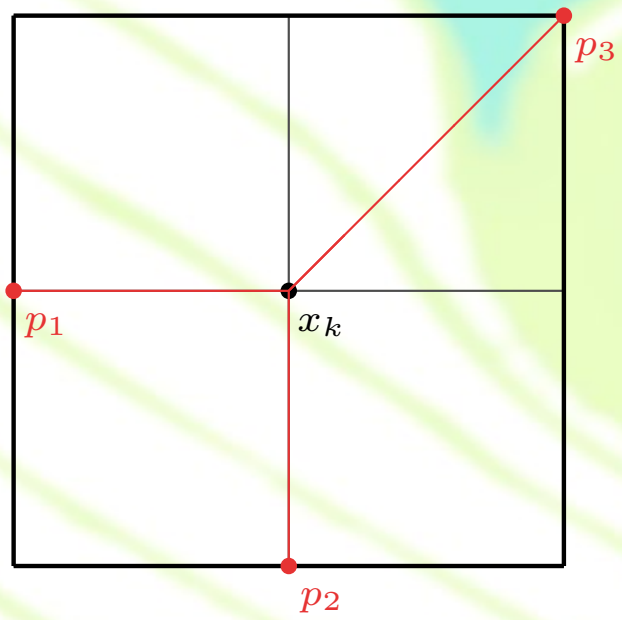
- Iterative algorithm that evaluates the black-box functions at some trial points.
- Trial points are generated on the mesh

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

  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)

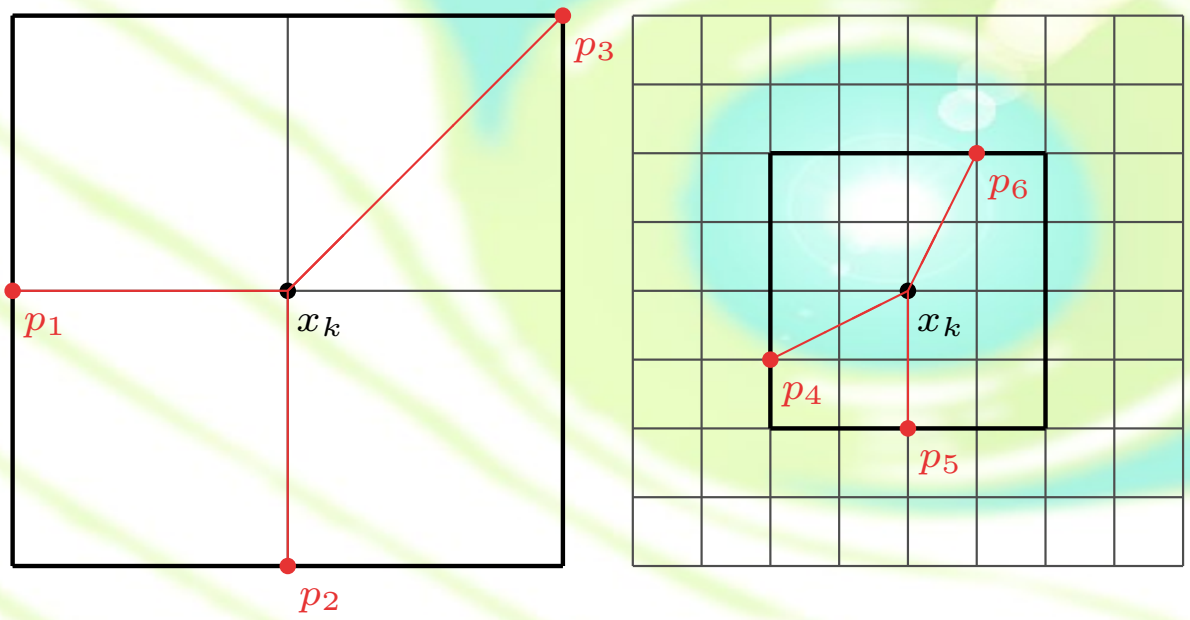
$\Delta_k = 1$



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

Poll illustration (successive fails and mesh shrink)

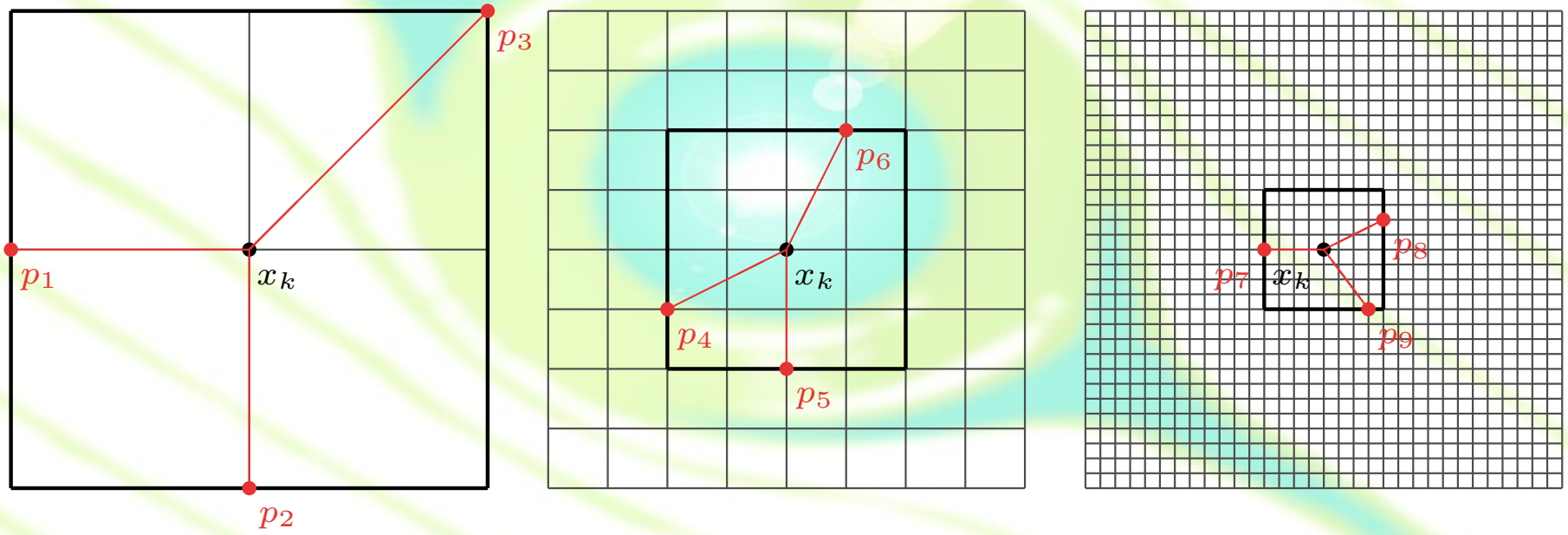
$\Delta_k = 1$ $\Delta_{k+1} = 1/4$



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

Types of directions used in the poll step

- Coordinate directions.
- 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 black-box 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)

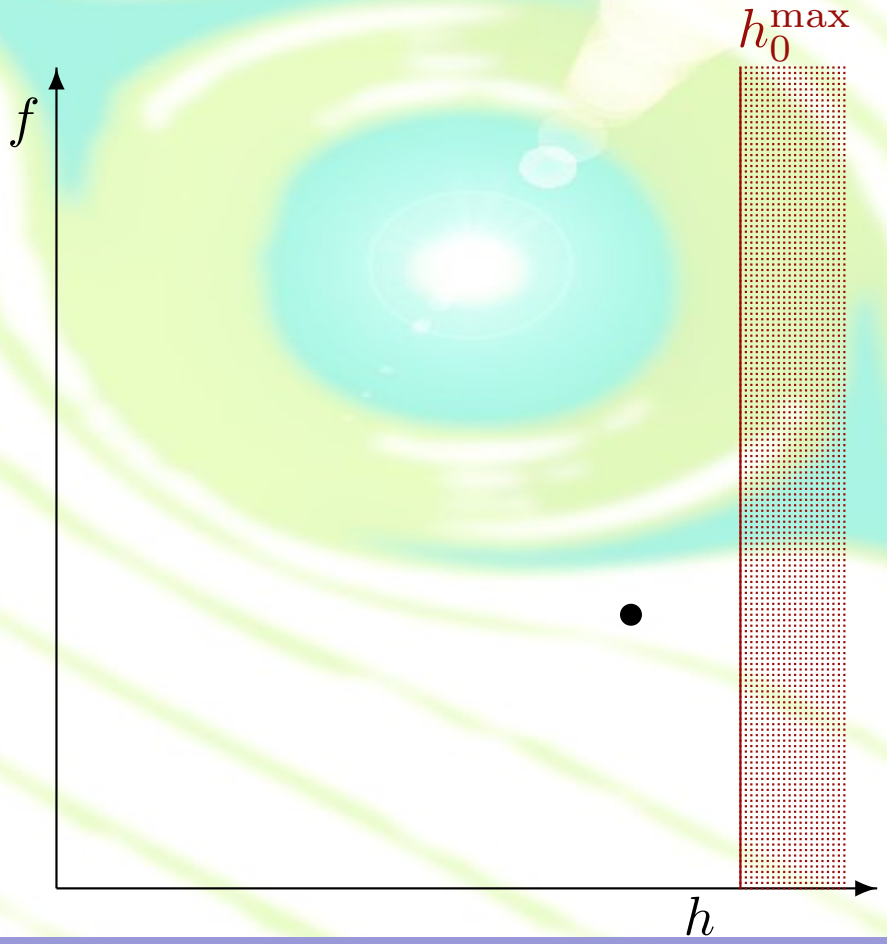
  Defined for the relaxable constraints.
  As in the filter methods of Fletcher and Leyffer, it uses the non-negative constraint violation function $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$

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

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

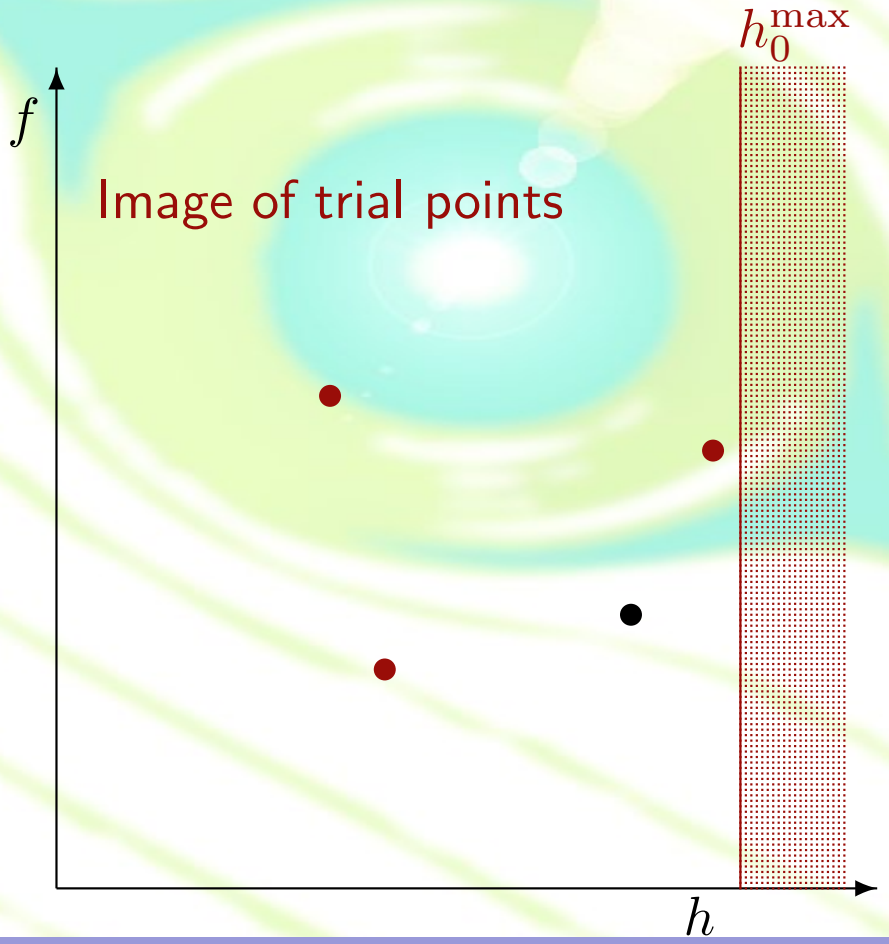
Three strategies to deal with constraints

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



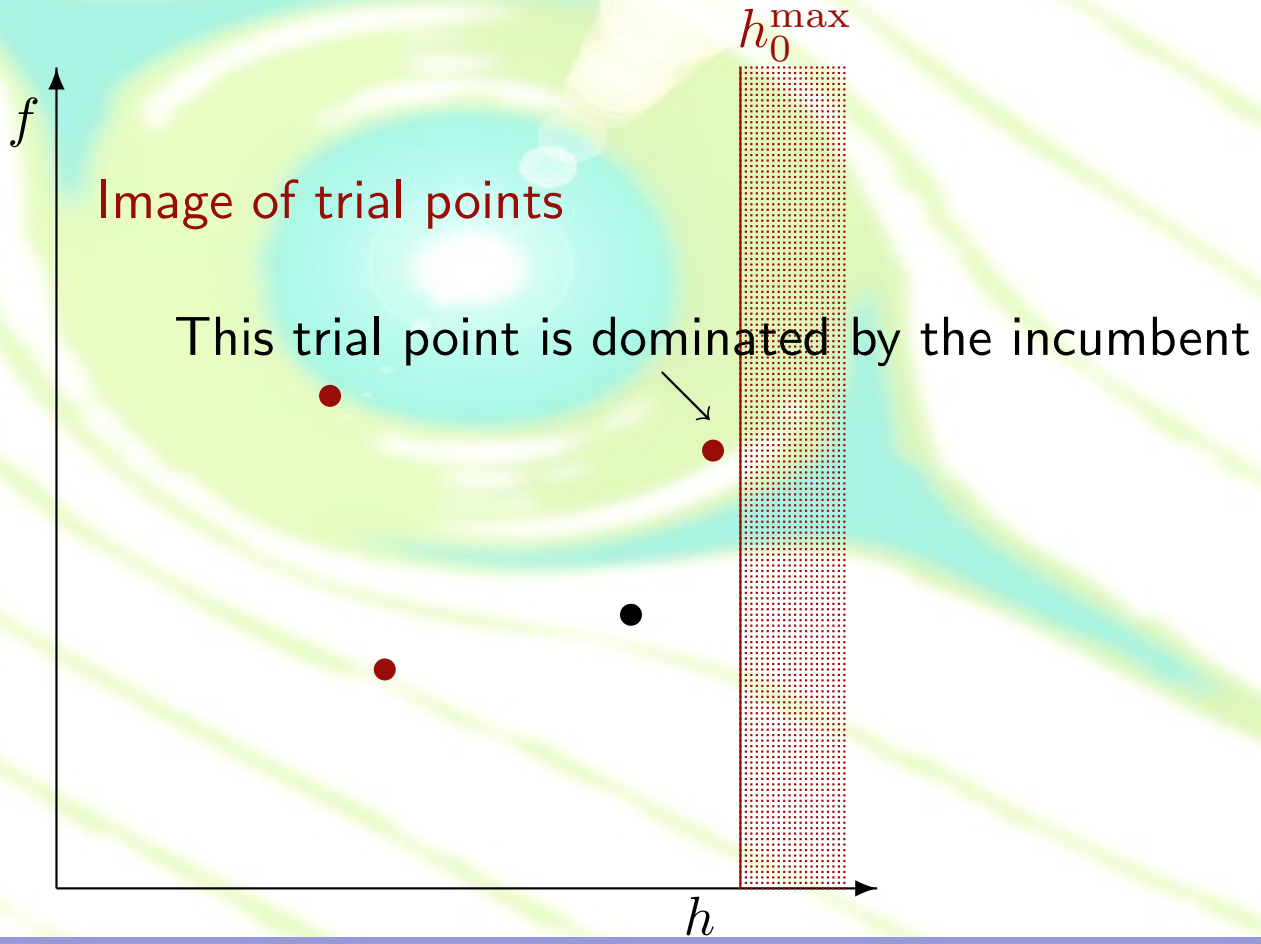
Three strategies to deal with constraints

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



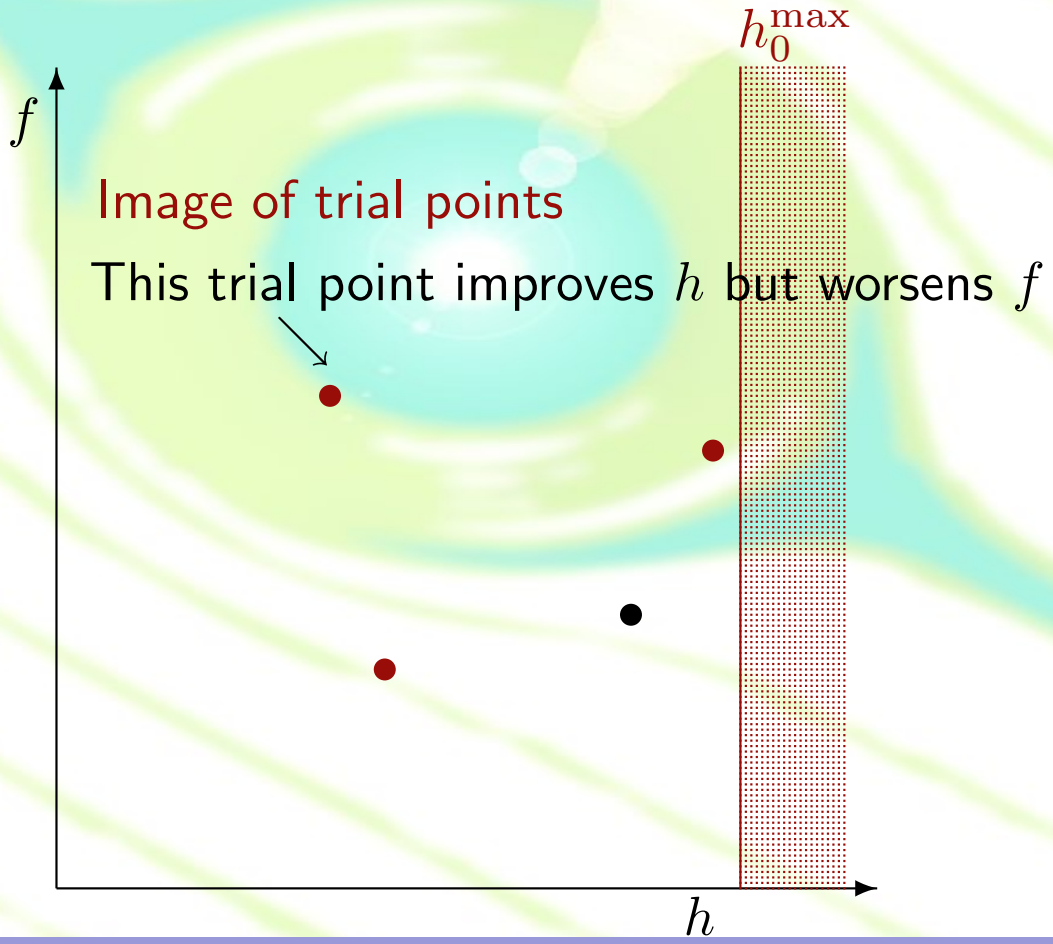
Three strategies to deal with constraints

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



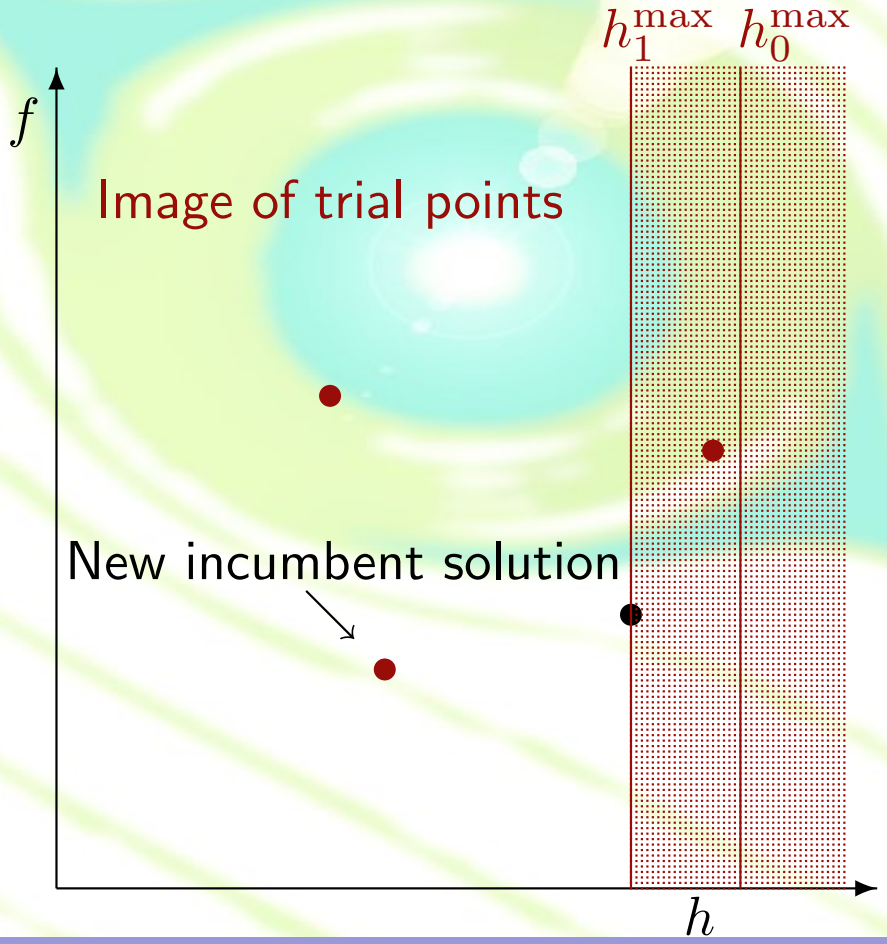
Three strategies to deal with constraints

- 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 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 can change. NOMAD deals with such cases (by associating 'signatures' to trial points).
- An extended poll is conducted 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.
- There are several types of surrogates...

Non adaptive surrogate $s(x)$

- Defined and given by the user at the beginning of an optimization.
- Is never updated or recalibrated.
- Depends only on x.
- It is a simplification (or relaxation) of the true function. (simpler physical model, smaller epsilons and/or max number of iterations in internal numerical methods).
- NOMAD currently considers only this type of surrogates.

Non adaptive surrogate with variable precision $v(x, p)$:

- Defined and given by the user at the beginning of an optimization.
- Is never recalibrated.
- Not an interpolation. Relaxation of the truth.
- p is a precision parameter that is decided by the algorithm (epsilons used in intern numerical methods). Should be linked to the mesh size parameter.

Adaptive surrogates $d(x, S)$ or $d(x, p, S)$:

- Can be user-defined or automatically constructed.
- Based on interpolation techniques (Kriging).
- S is a set of interpolation points (true function evaluations): can be user-chosen from the set of all true evaluations, or automatically selected.
- The surrogate is recalibrated periodically.
- p is a precision parameter.

Hybrid use of non adaptive and adaptive surrogates

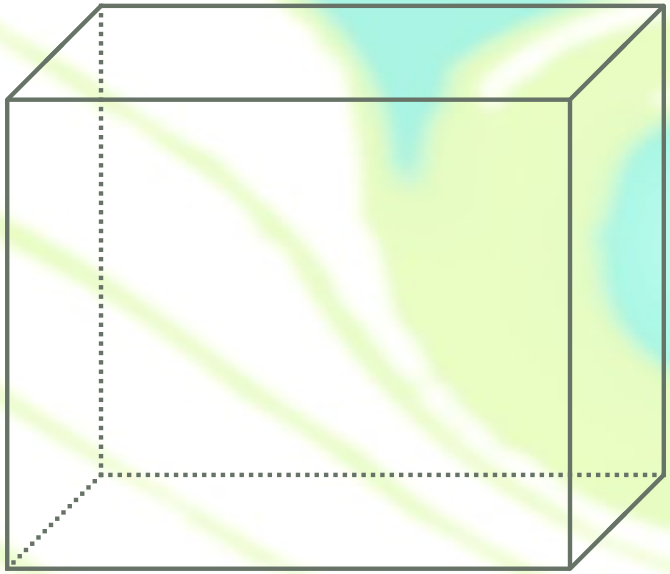
- The user defines $v(x, p)$, a non adaptive surrogate with variable precision.
- The method constructs an adaptive surrogate of the error $e(x, p, s) = f(x) - v(x, p)$.
- The surrogate considered is $v(x, p) + e(x, p, s)$.

Biobjective optimization

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

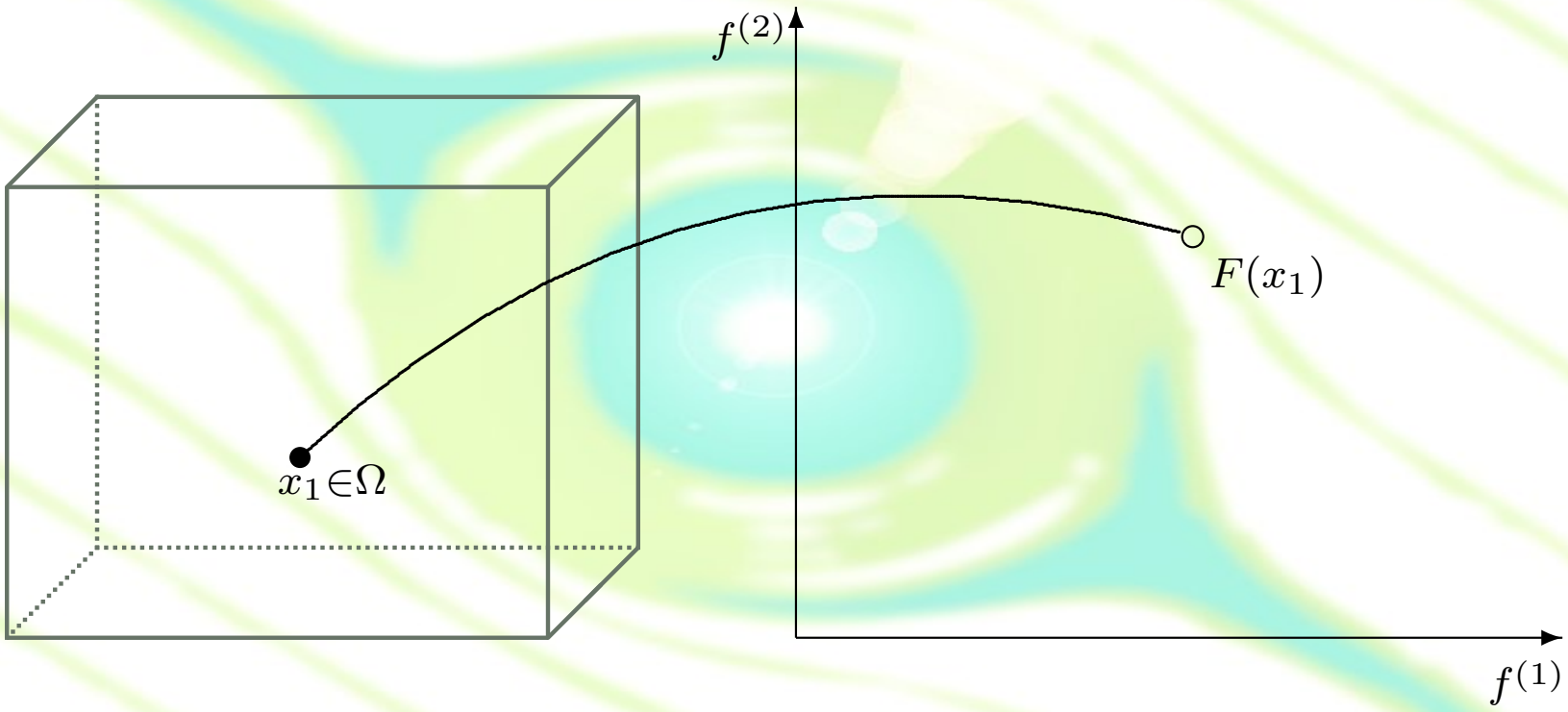
- $u \in \Omega$ dominates $v \in \Omega$ ($u \prec v$) if and only if $f^{(1)}(u) \leq f^{(1)}(v)$ and $f^{(2)}(u) \leq f^{(2)}(v)$ with at least one strict inequality.
- $u \in \Omega$ is Pareto optimal if and only if there is no $w \in \Omega$ such that $w \prec u$.
- We aim at finding the Pareto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

Example with $F : \mathbb{R}^3 \rightarrow \mathbb{R}^2$



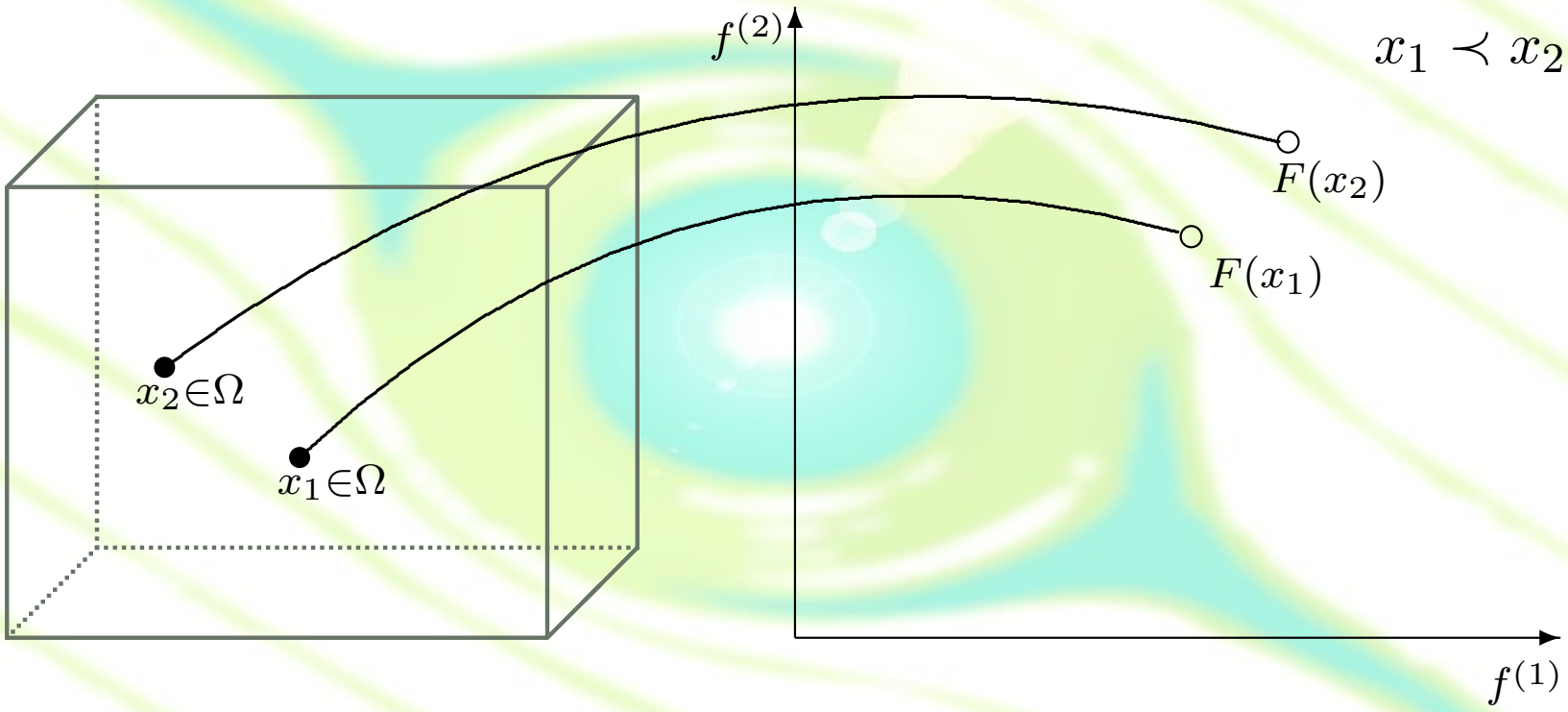
Domain : $\Omega \subset \mathbb{R}^3$

Example with $F : \mathbb{R}^3 \to \mathbb{R}^2$



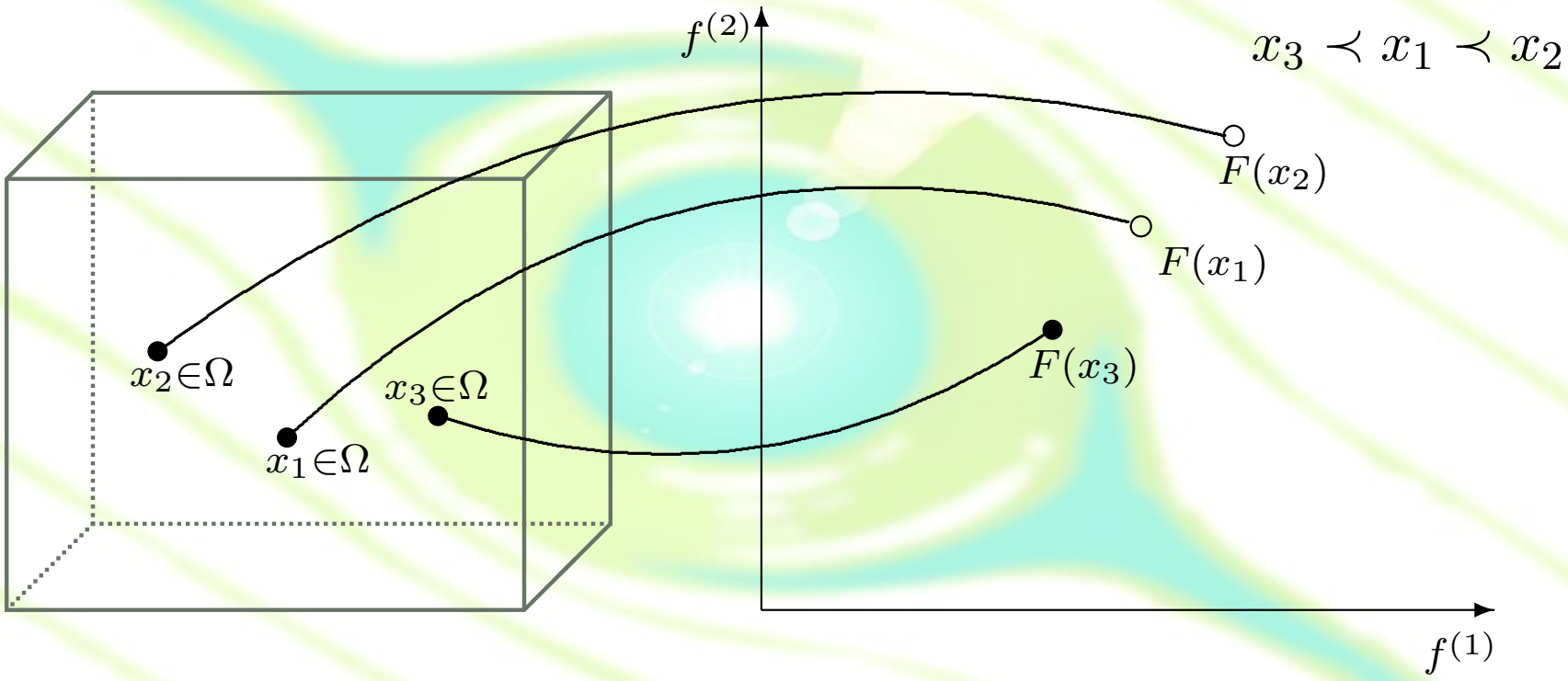
Domain : $\Omega \subset \mathbb{R}^3$

Example with $F : \mathbb{R}^3 \to \mathbb{R}^2$



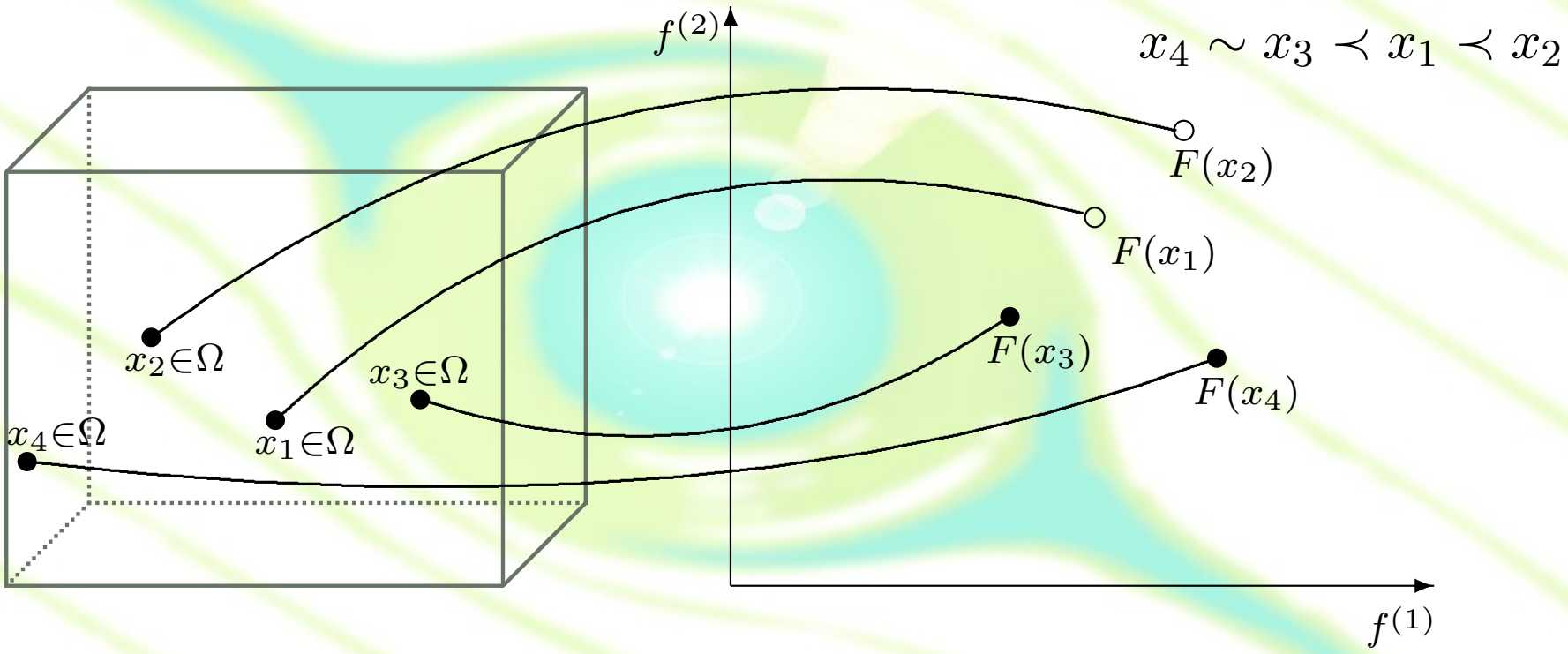
Domain : $\Omega \subset \mathbb{R}^3$

Example with $F : \mathbb{R}^3 \to \mathbb{R}^2$



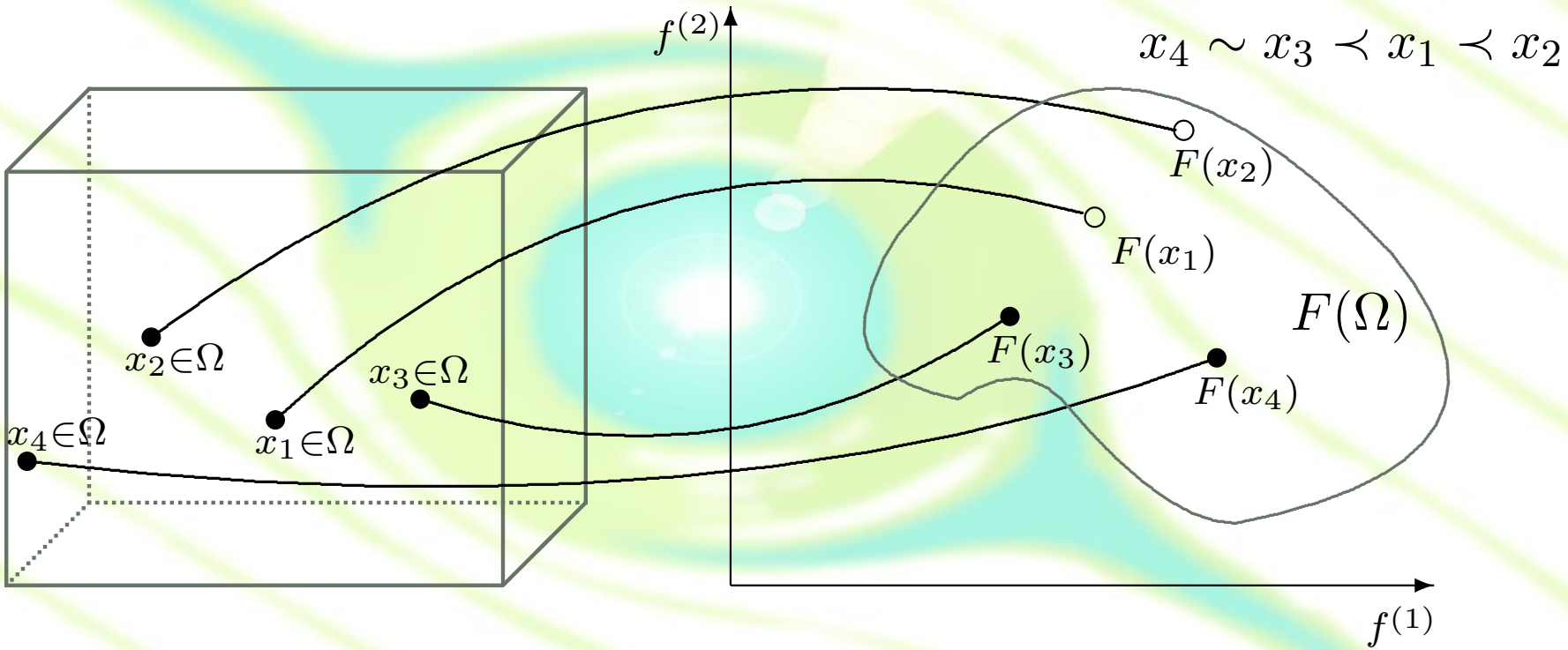
Domain : $\Omega \subset \mathbb{R}^3$

Example with $F : \mathbb{R}^3 \to \mathbb{R}^2$



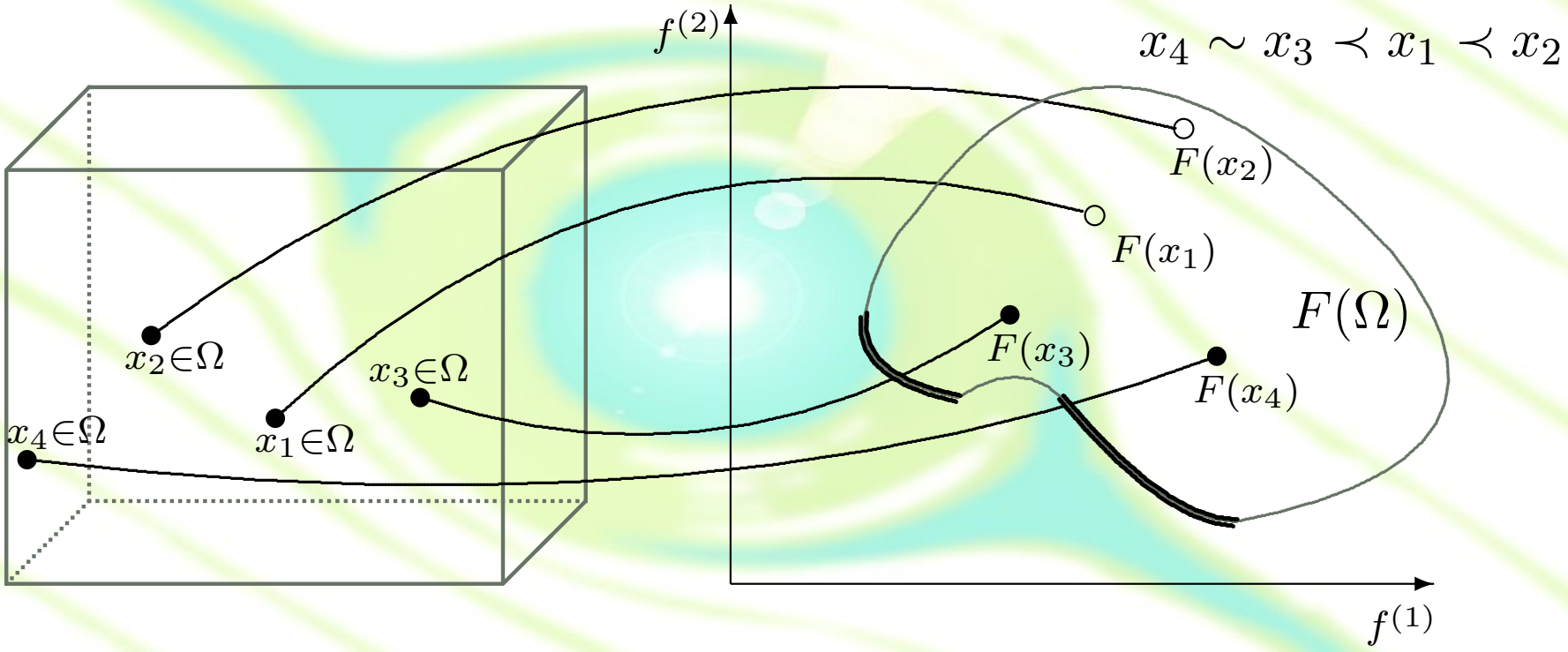
Domain : $\Omega \subset \mathbb{R}^3$

Example with $F : \mathbb{R}^3 \to \mathbb{R}^2$



Domain : $\Omega \subset \mathbb{R}^3$ Image of Ω in the objective space $\mathbb{R}^2$

Example with $F : \mathbb{R}^3 \to \mathbb{R}^2$



Domain : $\Omega \subset \mathbb{R}^3$ Image of Ω in the objective space $\mathbb{R}^2$

Single-objective subproblems

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

- We use the single-objective version of NOMAD.
- We launch this algorithm on a series of subproblems.
- 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

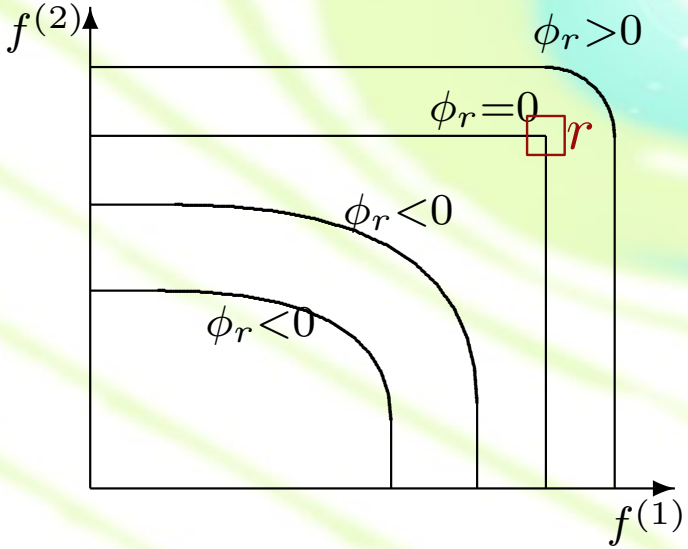
$$\phi_r(F(x)) := \begin{cases} -\prod_{q=1}^{p} (r_q - f^{(q)}(x))^2 & \text{if } F(x) \leq r, \\ \sum_{q=1}^{p} \left((f^{(q)}(x) - r_q)_+\right)^2 & \text{otherwise.} \end{cases}$$

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

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

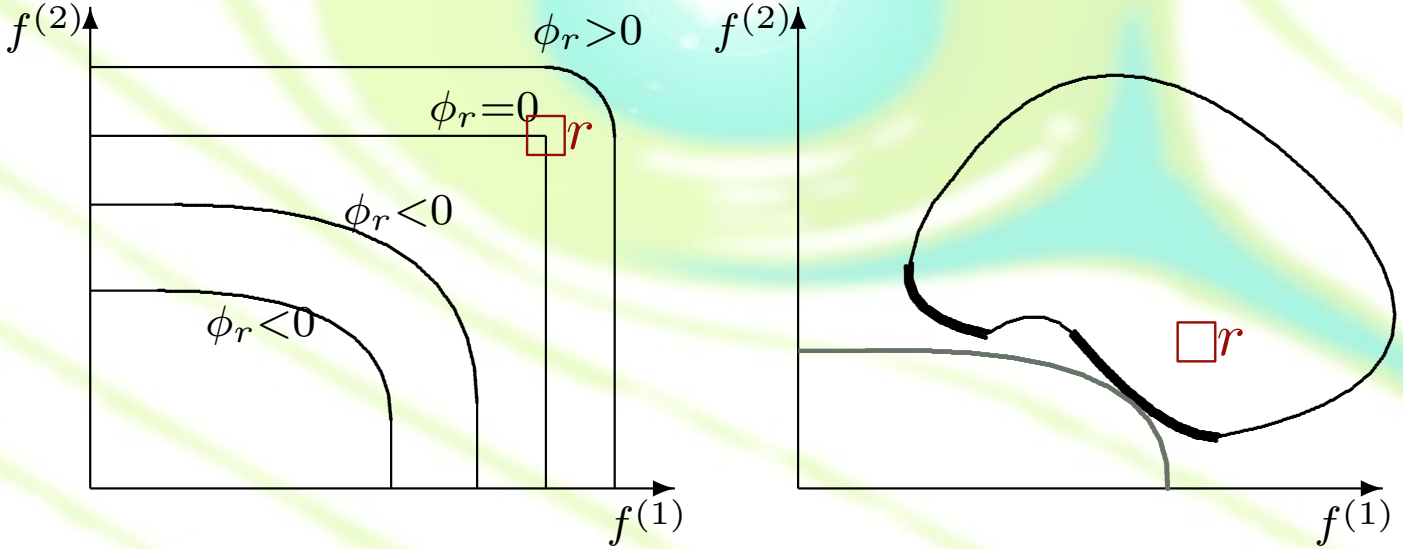
When minimized, it generates a Pareto solution that dominates r.



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

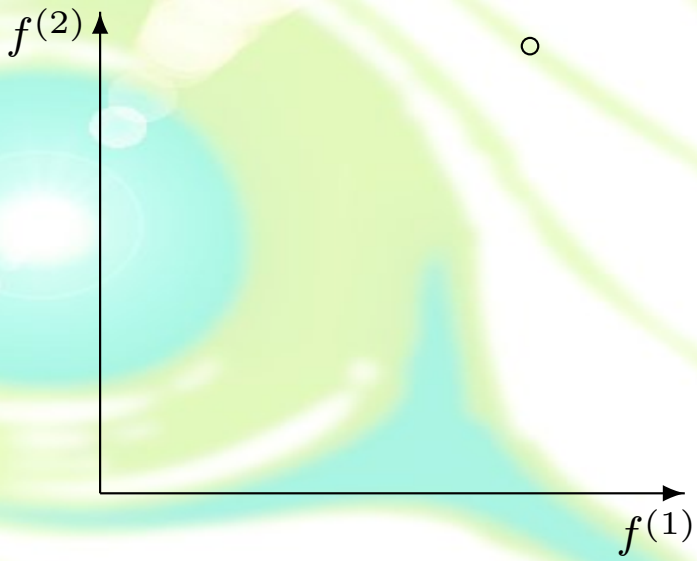
$$\phi_r(F(x)) := \begin{cases} -\prod_{q=1}^{p} (r_q - f^{(q)}(x))^2 & \text{if } F(x) \leq r, \\ \sum_{q=1}^{p} \left((f^{(q)}(x) - r_q)_+\right)^2 & \text{otherwise.} \end{cases}$$

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



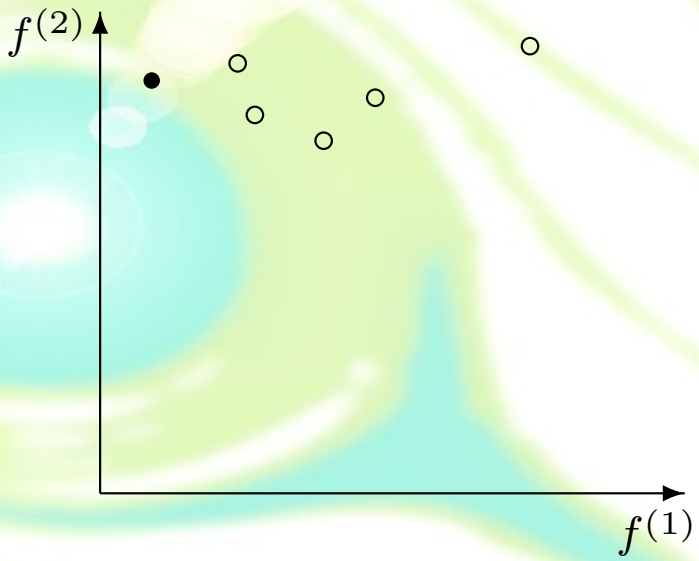
Every Pareto solution can be obtained by setting r appropriately.

BiMADS algorithm



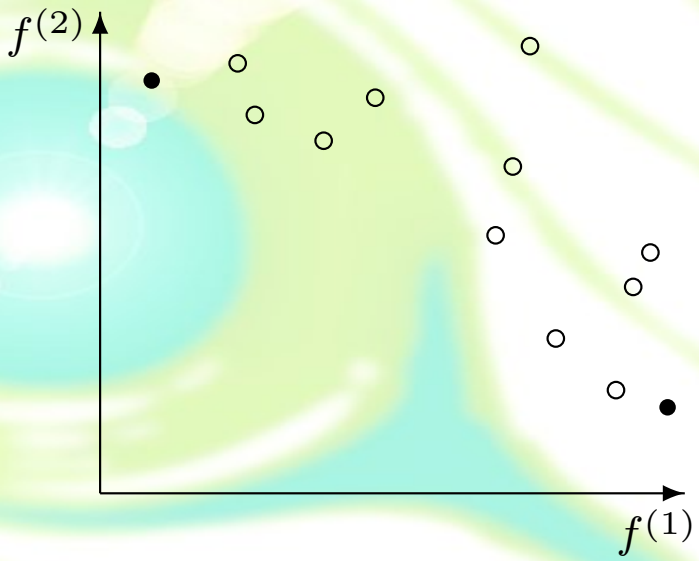
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.

BiMADS algorithm



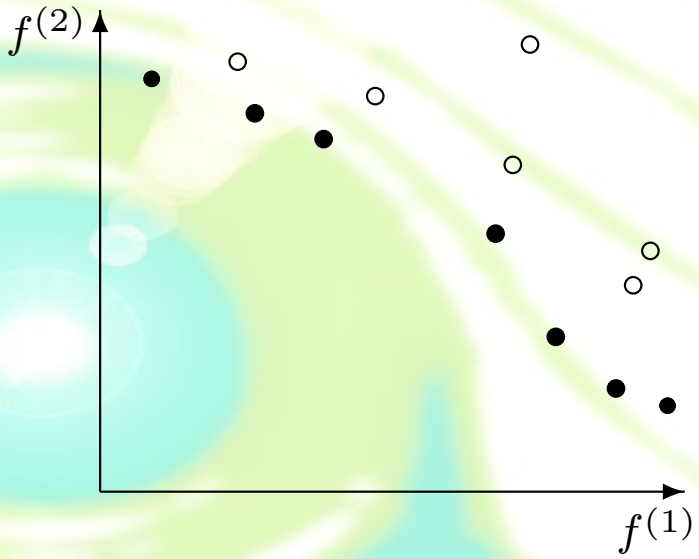
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.

BiMADS algorithm



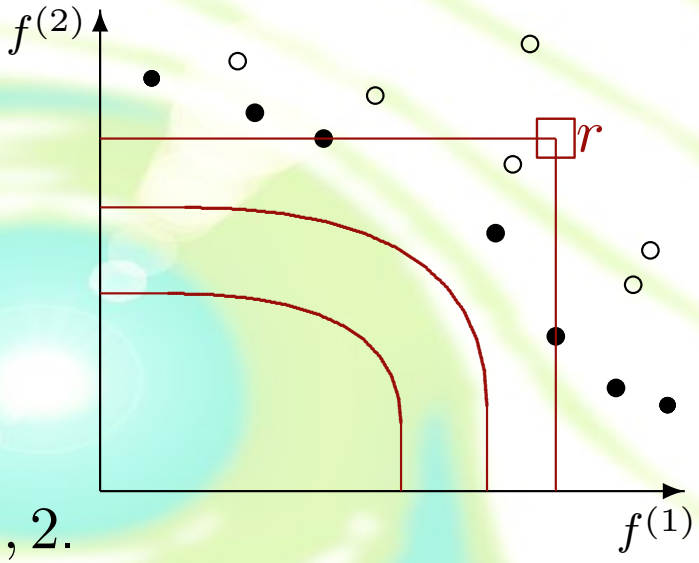
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.

BiMADS algorithm



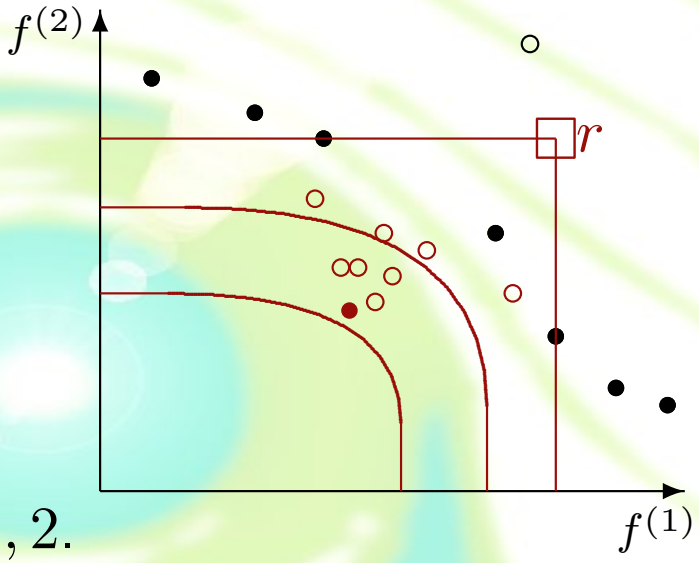
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.

BiMADS algorithm



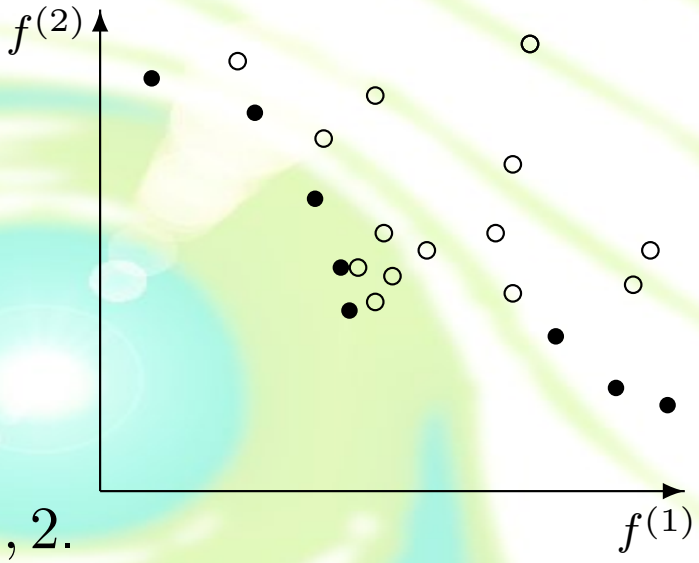
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.
  - SINGLE-OBJECTIVE MINIMIZATION:
    Solve the single-objective problem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

BiMADS algorithm



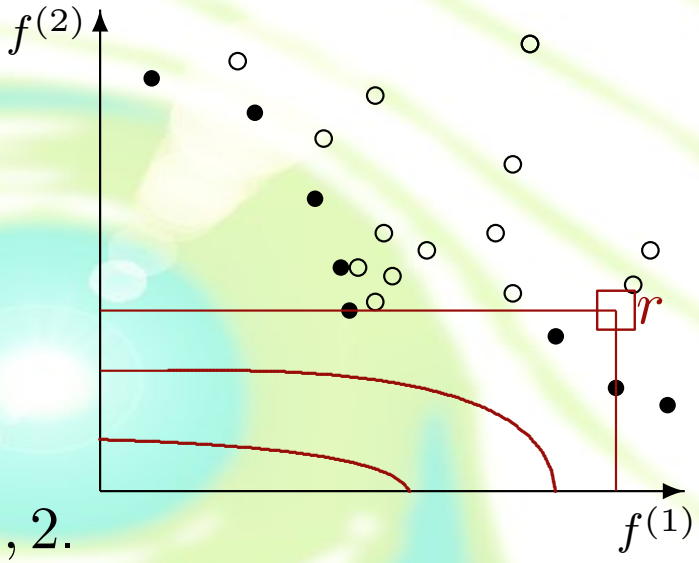
- INITIALIZATION:
  Solve $\min_{x \in \Omega} f^{(q)}(x)$ for $q = 1, 2$.
- MAIN ITERATIONS:
  - REFERENCE POINT DETERMINATION:
    Use the set of feasible ordered undominated points generated so far to generate a reference point r.
  - SINGLE-OBJECTIVE MINIMIZATION:
    Solve the single-objective problem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

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 $\min_{x \in \Omega} \phi_r(F(x))$.

Biobjective optimization: example

pMADS

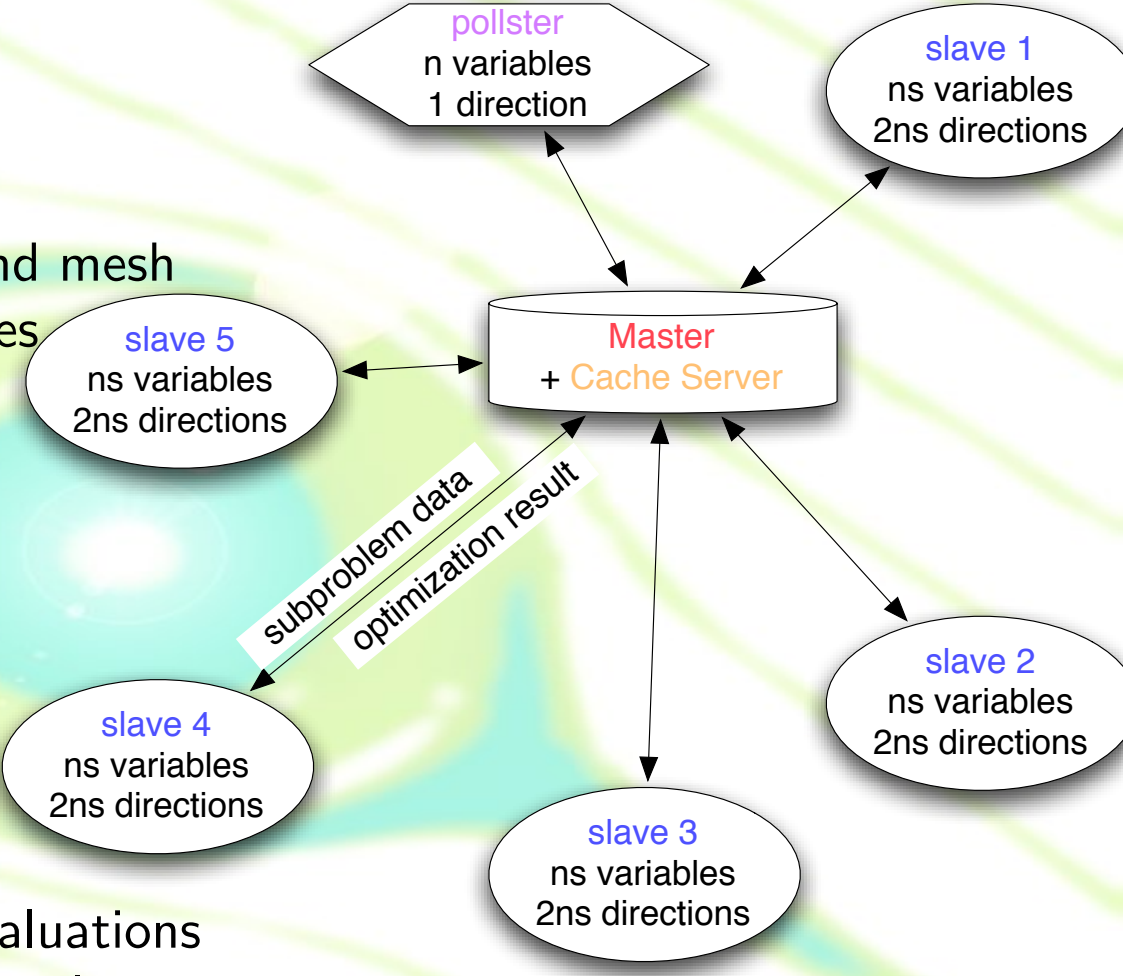
- Idea: simply evaluate the trial points in parallel.
- Synchronous version:
  - The iteration is ended only when all the evaluations in progress are terminated.
  - Processes can be idle between two evaluations.
  - The algorithm is identical to the scalar version.
- Asynchronous version:
  - If a new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

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 black-box evaluations
  - allows the "cache search" in slave processes



NOMAD (Nonsmooth 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 GPL license.
- Complete user guide available in the package.
- TOMS paper in revision.

NOMAD – history and team

- Developed since 2000.
- Algorithm designers:
  - M. Abramson, C. Audet, J. Dennis, and myself.
- Developers:
  - Versions 1 and 2: G. Couture.
  - Version 3 (2008): M. Sylla and Q. Reynaud, and myself.
- Current version: 3.3.
- Next version (March): 3.4 (parallelism).

Main 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).
  - Cache search (used in parallel versions).
  - User search.

Main 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).

NOMAD installation

- Pre-compiled executables are available for Windows and Mac.
- Installation programs copy these executables.
- On Unix/Linux, after download, launch the `configure` command that creates a makefile for compilation.
- Two ways to use NOMAD: batch mode or library mode.

Black-box 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 g1 g2` (objective and two constraints).

Important parameters

- Necessary parameters: dimension (n), the black-box characteristics, and the starting point (x_0).
- All algorithmic parameters have default values. The most important are:
  - Maximum number of black-box evaluations,
  - Starting point (more than one can be defined),
  - Types of directions (more than one can be defined),
  - Initial mesh size,
  - Constraints types,
  - Latin-Hypercube sampling,
  - Seeds.
- See the user guide for the description of all parameters, or use the `nomad -h` option.

Run NOMAD

- `> nomad parameters.txt`
- Example on the laptop.

Advanced functionalities (library mode)

- No system calls: the code executes faster (more than twice).
- Easy to program multiple runs in parallel with different seeds.
- The user can program a custom search strategy.
- The user can pre-process all evaluation points before they are evaluated.
- The user can decide the priority in which trial points are evaluated.
- The user can indicate user-functions that will be called at some events (new success, new iteration, new MADS run in bi-objective optimization)

Examples included in the NOMAD package

These examples illustrate other possibilities:

- Multi-start from points generated with LH sampling.
- Compatibility with previous versions of NOMAD.
- Problems used in library mode and coded as:
  - a Windows DLL,
  - a GAMS program,
  - a CUTEr problem,
  - a Matlab function,
  - a FORTRAN code.
- A GUI prototype in JAVA.

Discussion

- We have presented the MADS algorithm, some of its features, and the NOMAD implementation.
- Future versions of NOMAD:
  - 3.4 (March): Parallelism (pMADS, Coop-MADS and PSD-MADS).
  - 3.5: First version of adaptive surrogates.
- References:
  - MADS references available from the NOMAD website.
  - NOMAD tests: Globalizations strategies (COAP) and OrthoMADS (SIOPT).