Derivative free parametric optimization concepts for material scientists

Rodolphe Le Riche
CNRS and Ecole des Mines de Saint-Etienne

ARCHIMAT 2011 workshop
First International school on architectured materials, Autrans, France, May 2011

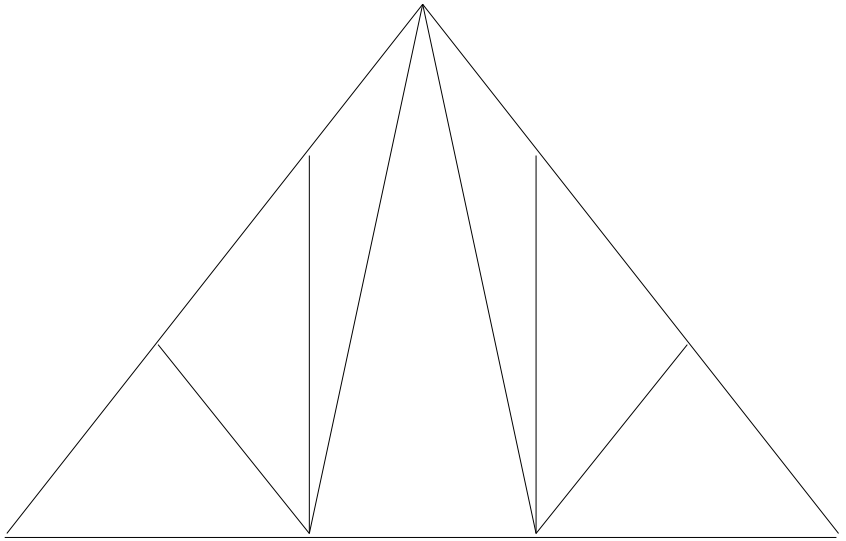
Purpose of this presentation

A short introduction to optimization for material scientists :

- When is optimization useful ?
- Get oriented in the jungle of optimization algorithms,
- show some basic optimization principles (derivatives free),
- provide references to state-of-the-art methods,
- where to add problem specific knowledge in tackling difficult optimization cases.

Goal of parametric numerical optimization

Ex : 15 bars truss, each bar chosen out of 10 profiles
→ 10^{15} possible trusses. How to choose ?



How to choose ? The modeling, formulation, optimization steps

1. Have a model, y , (analytical, finite elements, coupled sub-models …) of the object you need to optimize.
2. Formulate the optimization problem

$$\min_{x \in S} \; f(y(x))$$
$$g(y(x)) \leqslant 0$$

x : optimization variables
f : objective functions
g : optimization constraints
f , g : optimization (performance) criteria

3. ***Try*** to solve the problem, either analytically (e.g., Karush Kuhn and Tucker conditions) or using optimization algorithms.
[4. Almost never right the first time : go back to 1]

Outline

1. Introduction : examples of optimization uses in material engineering, general numerical optimization points, why derivative free ?

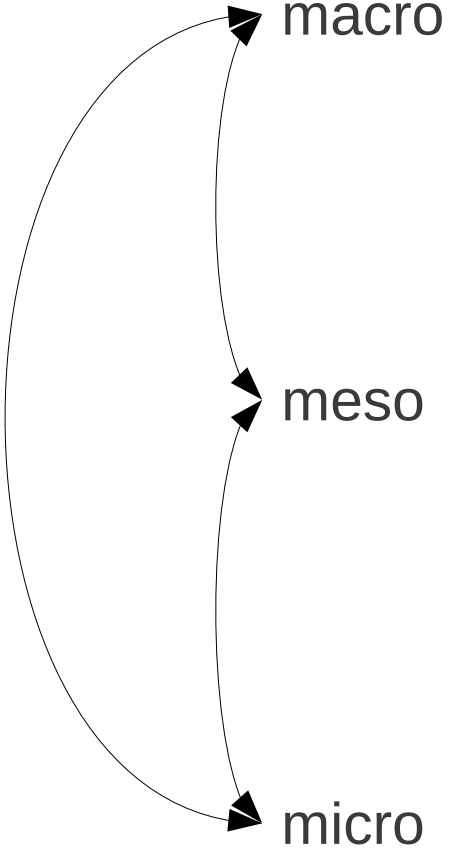
2. Deterministic optimization

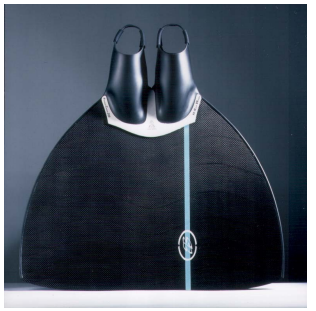
3. Stochastic optimization

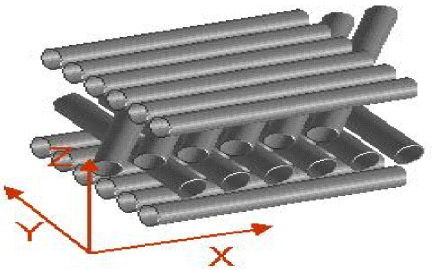
4. Adding problem specific knowledge

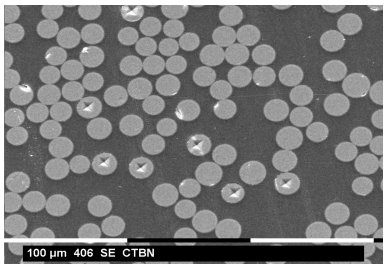
Application examples (1)
Structural design at various scales

x, f and g can be defined at any combination of scales





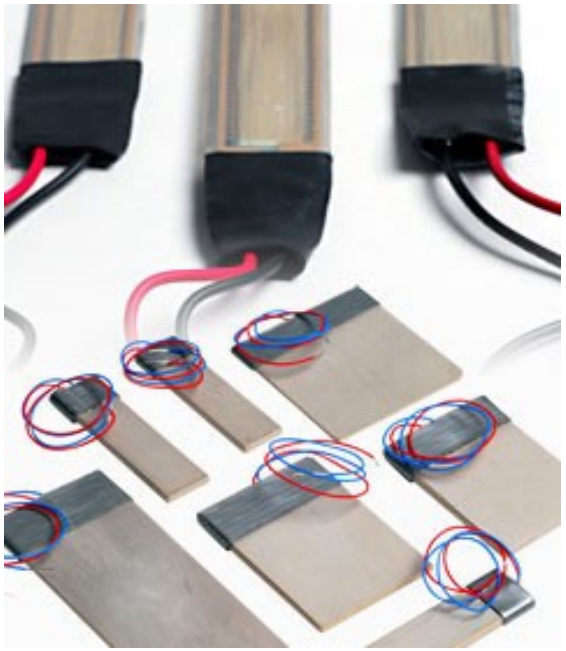




Application examples (2) : Multi-disciplinary design

x, f and g concern many disciplines

Coupled material design and control



(from Advanced Ceramics Inc.)

Coupled manufacturing and design

(composite tow fiber placement machine at NLR)

→ more parameters, beyond the scope of mechanics of materials.

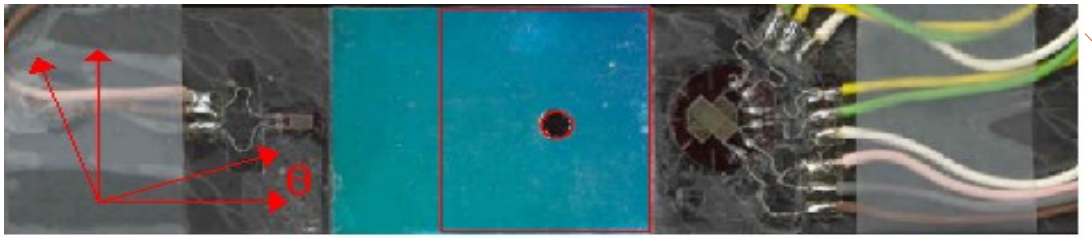
Application examples (3)
constitutive behaviour identification

$$\dot{\sigma} = C\left(\dot{\varepsilon}, \chi \; ; \; x\right)$$

χ , internal variables

x , constitutive law parameters

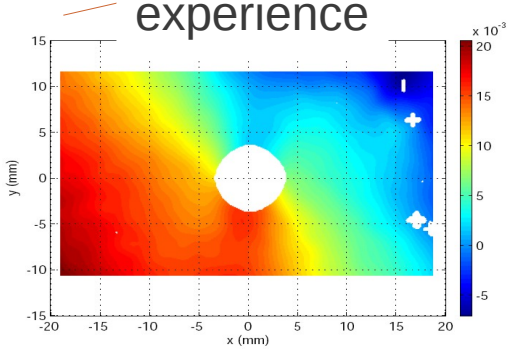
$$min_{x \in S}\, \text{distance}\left(\text{Experiment} \; , \; \text{Model}\left(x\right) \right)$$



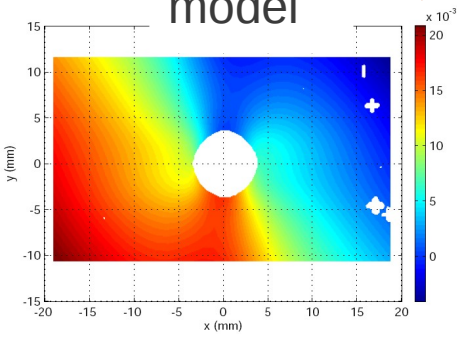
full-field measures

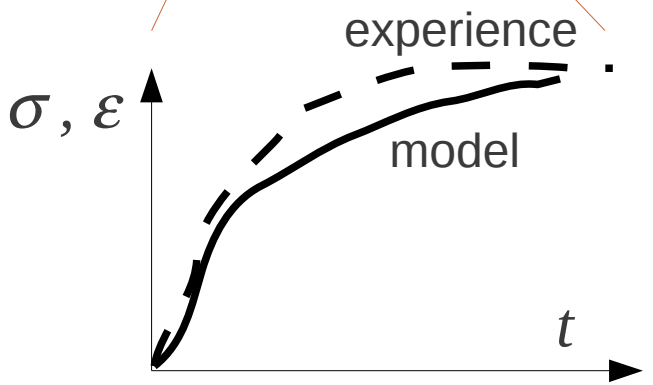
point-wise measures

experience



model





Outline

1. Introduction : examples of optimization uses in material engineering, general numerical optimization points, why derivative free ?

2. Deterministic optimization

3. Stochastic optimization

4. Adding problem specific knowledge

Analytical optimization

Some optimization problems can be solved analytically or with 1 iteration methods. Expl. quadratic programming problems, $n<1000$:

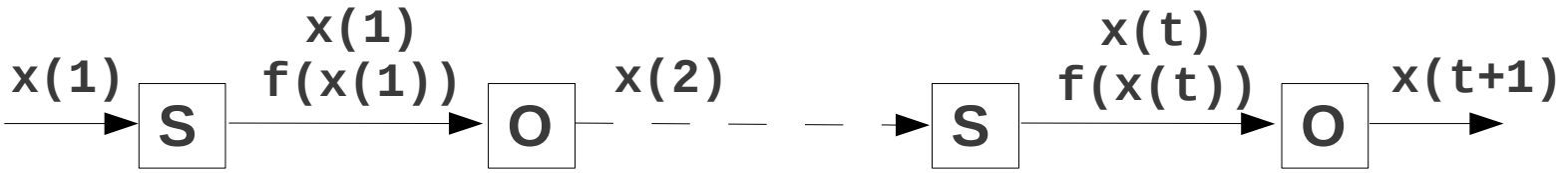
$$x \in S \subset \mathbb{R}^n$$
$$f(x) = \frac{1}{2} x^T H x + A x + B \quad , \quad H > 0$$
$$g(x) = C x + D$$

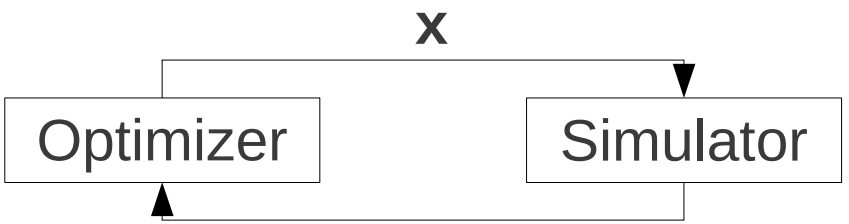
Otherwise, use iterative optimization algorithms, dubbed « optimizers ».

Optimization programs

An optimizer is an algorithm that iteratively proposes new x's based on past trials in order to approximate the solution to the optimization problem

x(t+1) = Optimizer[x(1),f(x(1)) , … , x(t),f(x(t))]





f(y(x)) + (FJB) **dfdx(y(x)) + ...**

The ***cost*** of the optimization is the number of calls to the simulator y (usually = number of calls to f)

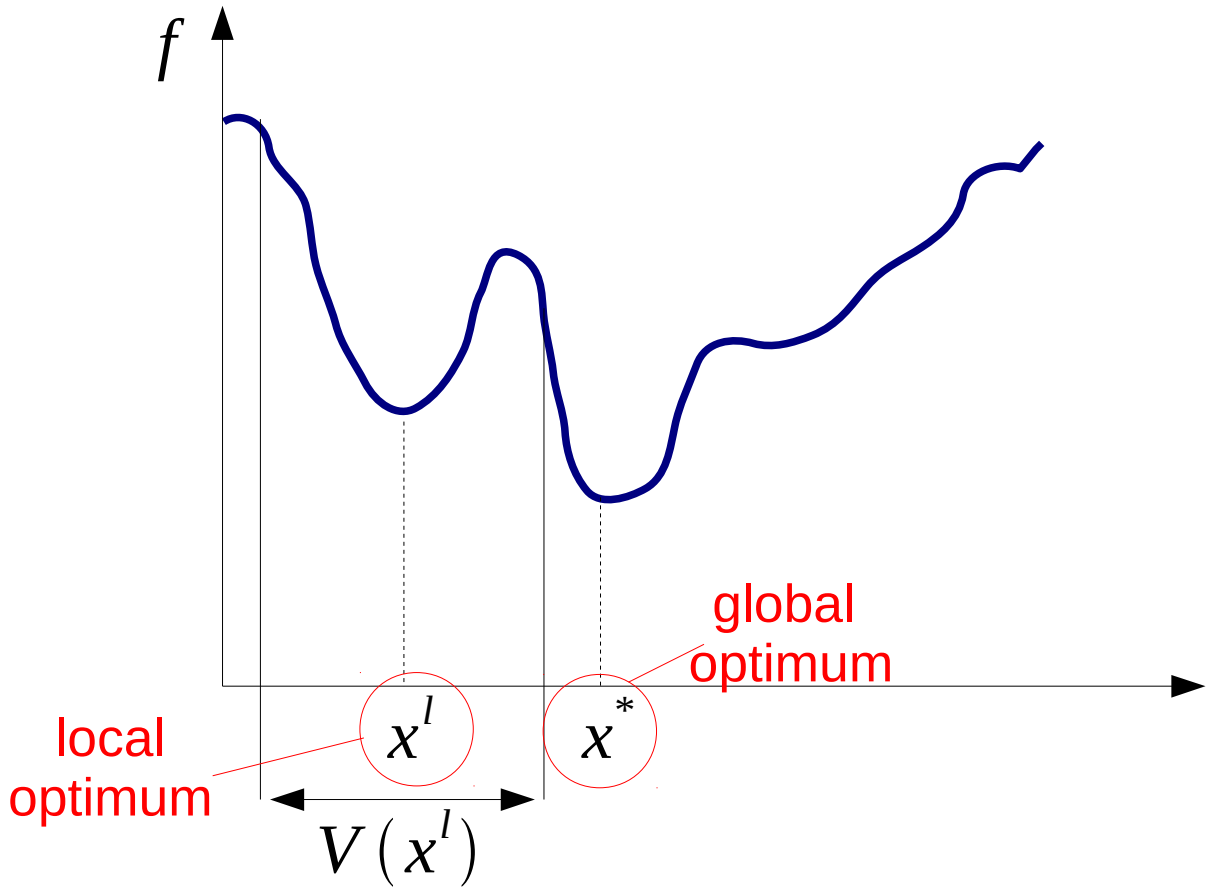
Local vs. global optimization (1)

Here, we focus on

$$\min_{x \in S} f(x)$$

$$S \subset \mathbb{R}^n \text{ or } \mathbb{N}^n \text{ or } \left\{\mathbb{R}^{n1} \cup \mathbb{N}^{n2}\right\}$$

i.e., continuous or integer or mixte optimization.
We don't discuss g.



Local vs. global optimization (2)

Local and global optimizers have different goals, hence working principles.

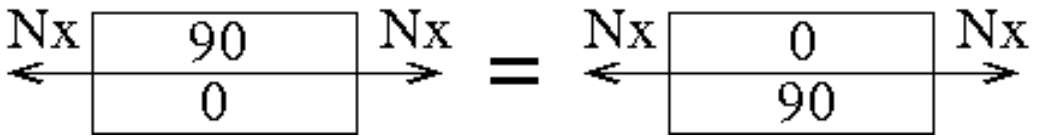
Local optimizers use local information (typically gradient based in R^n) and aim at ***efficiently*** finding a neighbouring optimum.

Global optimizers ***compromise on efficiency and exploration*** of the search space S. Global search is computationally more costly than local search but is sometimes needed (large n, periodic and noisy functions).

Composites often need global optimizers

because stacking sequences offer an excess in design variables.
Expl :

$max_{\theta_i} \; A_{11}$, longitudinal laminate stiffness



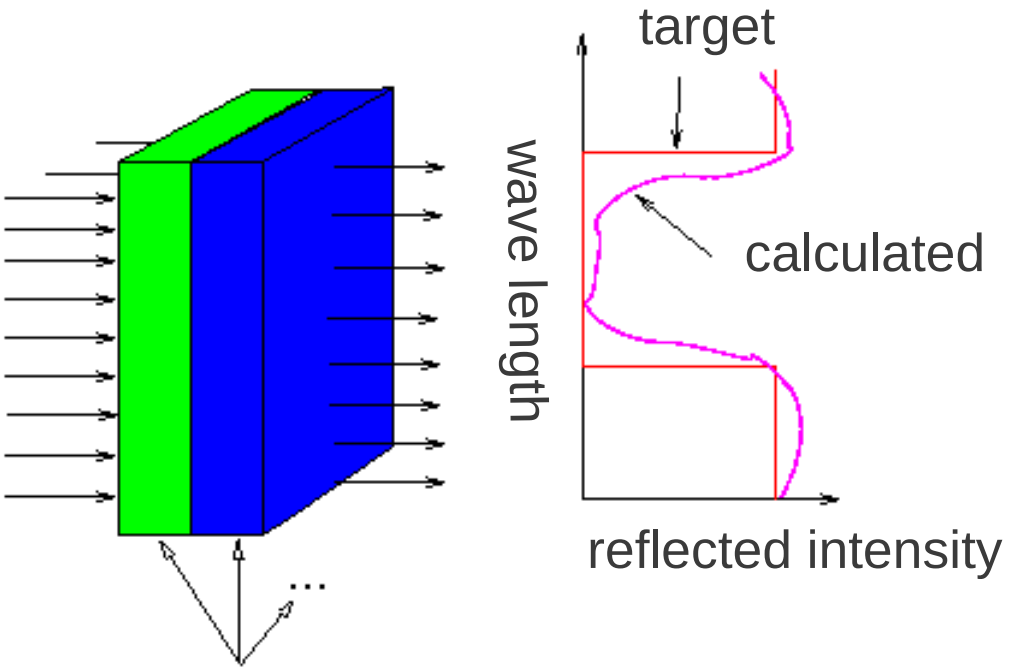
In composite design, there are often local optima.

Expl., $N_y/N_x = 0.5$, length=20 in., width=5 in., graphite-epoxy

sequence	buckling	strength
$(90_2/\pm 45_2/90_2/\pm 45/90_2/\pm 45_6)_s$	9998.19	10394.81
$((90_2/\pm 45_2)_2/90_2/\pm 45/90_2/\pm 45_3)_s$	9997.60	10187.93
$(\pm 45/90_4/\pm 45/90_2/\pm 45_5/90_2/\pm 45)_s$	9976.58	10187.93

Global optimization need, 2nd example : optical filters

(after T. Bäck)

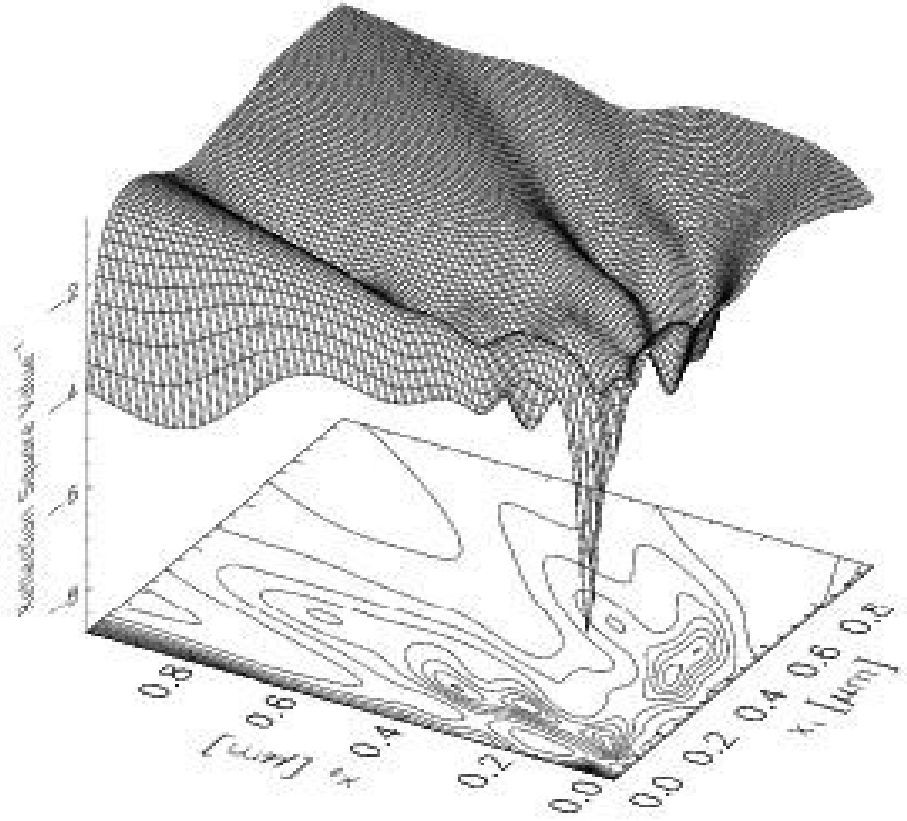


filters (material, thickness, numbers)

$$min_{\text{nb., mat., thickn.}} \int_{\lambda_m}^{\lambda_M} [R_{\text{calc.}}(\lambda) - R_{\text{target}}(\lambda)]^2 d\lambda$$

Optical fibers example (cont.)

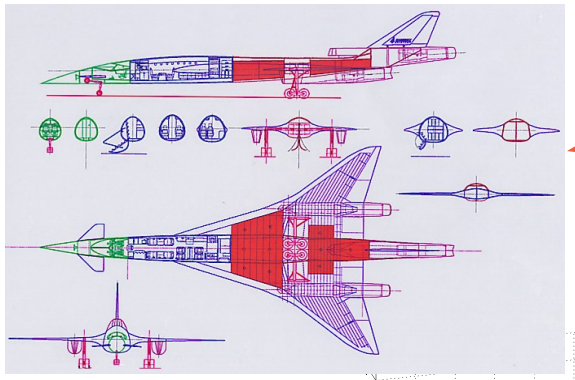
Distance objective function landscape.
x and y are two thicknesses, z the distance:



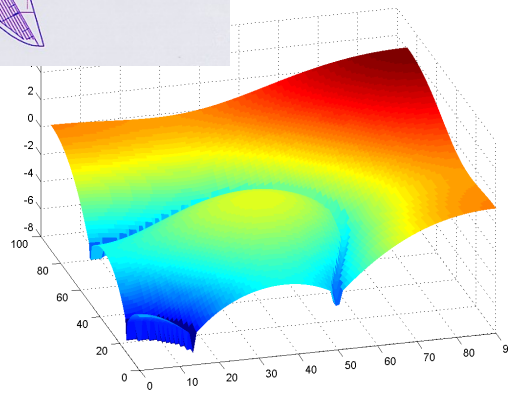
→ a local optimizer may get trapped at a local optimum.

Make can make optimizing difficult

Goal : $min_{x \in S \subset \mathbb{R}^n} f(x)$

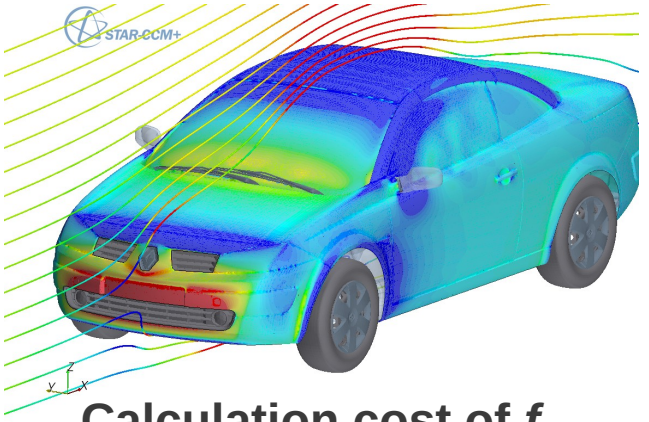


Number of variables, *n*

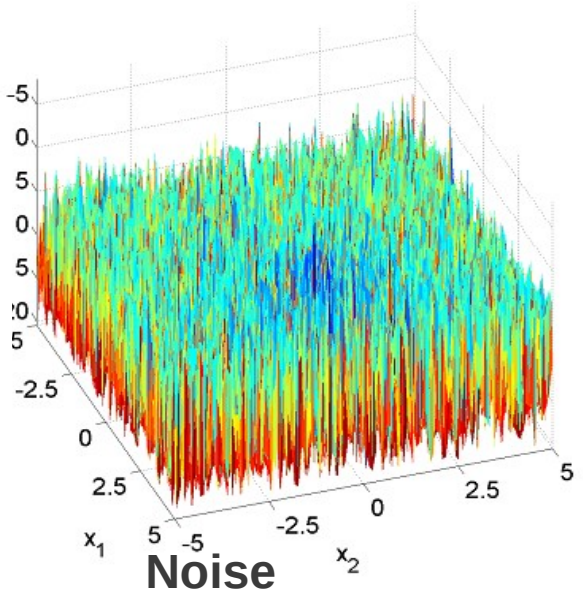


local optima
(expl. composites)

Ill conditioning



Calculation cost of *f*



Noise

difficulties

Derivative free optimizers

When a gradient, an Hessian, … (sensitivity analysis) is available, use it ! → See Franz-Joseph Barthold's presentation

Derivative free optimizers are useful

when gradients, $\nabla_x f$, $\nabla_x g$, are not available,

when gradients do not exist, $x \in S \subset \mathbb{N}^n$
or f or g are non differentiable (C^0).

Outline

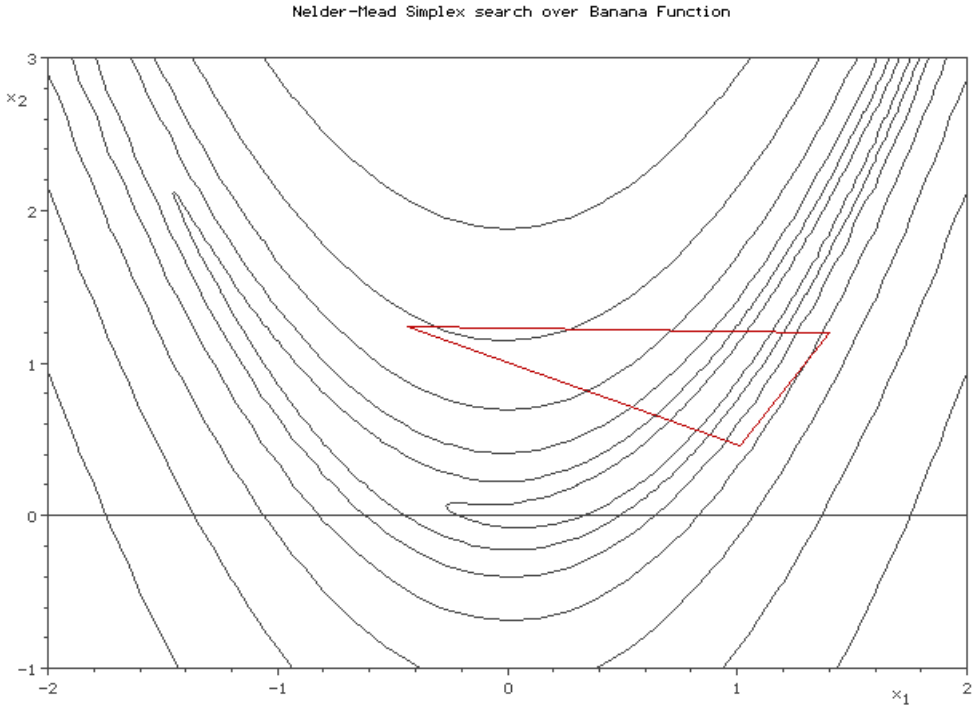
1. Introduction
2. Deterministic optimization : local, global.
3. Stochastic optimization
4. Adding problem specific knowledge

A basic (yet fine) local derivative free algorithm in R^n : Nelder-Mead

Nelder, John A.; R. Mead (1965). "A simplex method for function minimization".

Principle : a simplex (a polytope of n+1 vertices in n dimensions) undergoes geometrical transformations to go downhill. It is a « pattern search method ».

2D example :



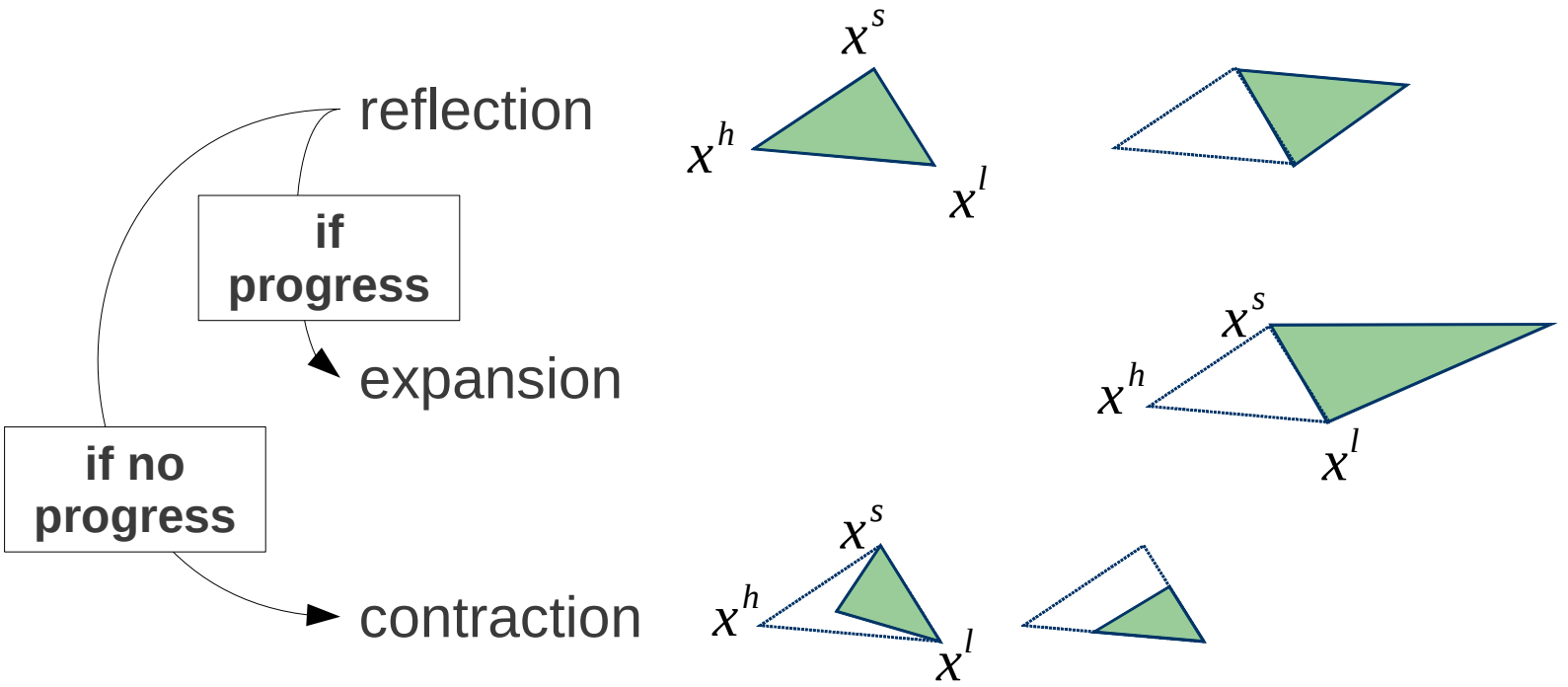
(from Simiprof, wikipedia, 2011)

Nelder-Mead (2)

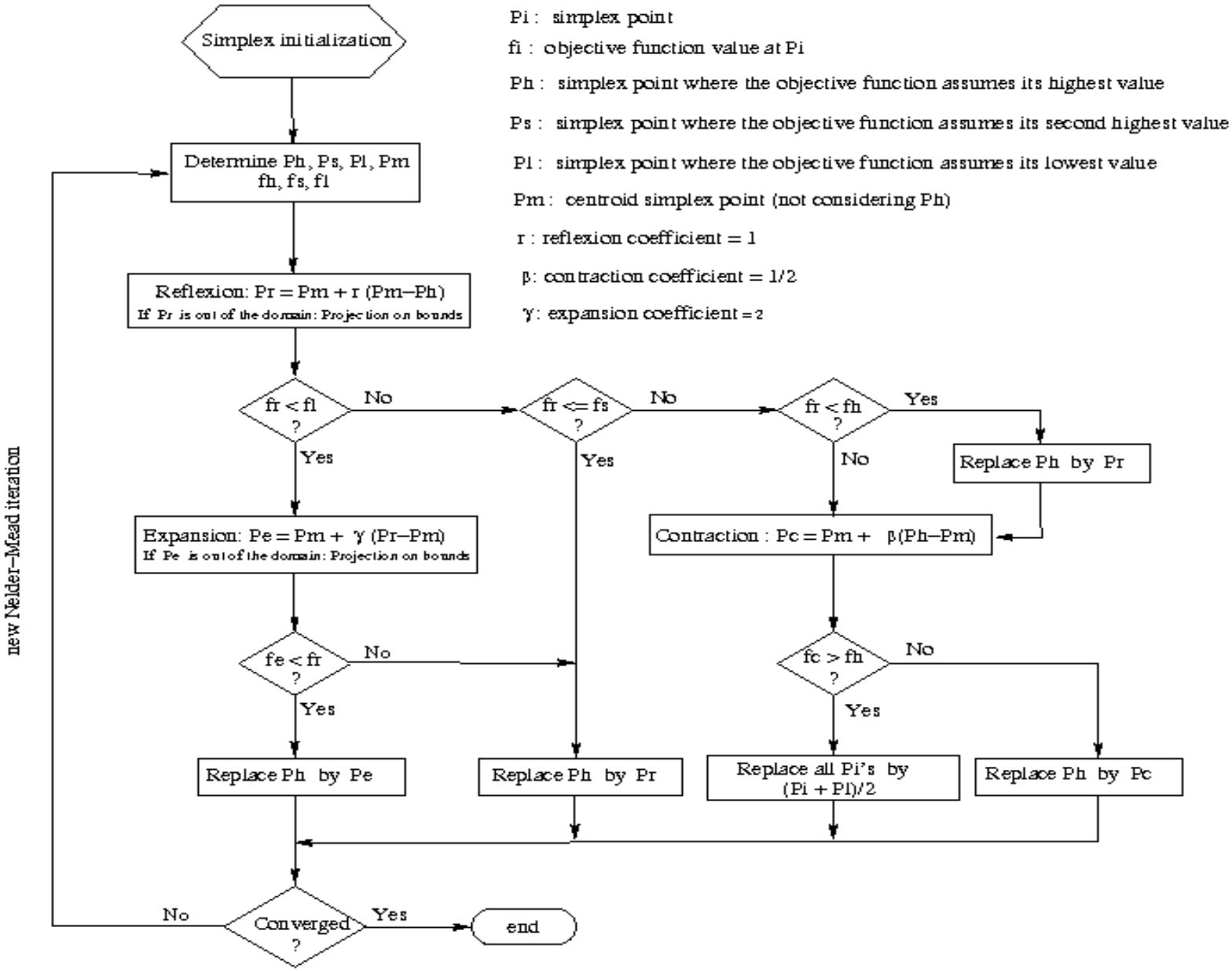
The vertices of the simplex are $n+1$ points of the search space where the objective function is evaluated

$$f(x^l) \leqslant \ldots \leqslant f(x^s) \leqslant f(x^h)$$

The geometrical transformations are :



Nelder-Mead (3) : flow chart (hard to read ;-( )

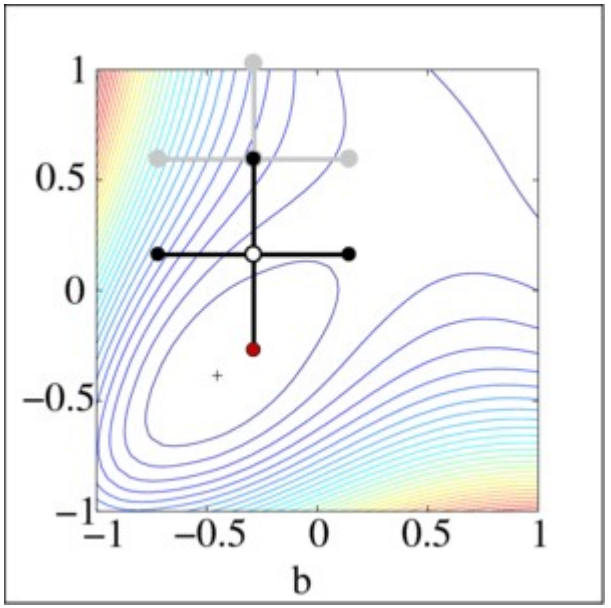


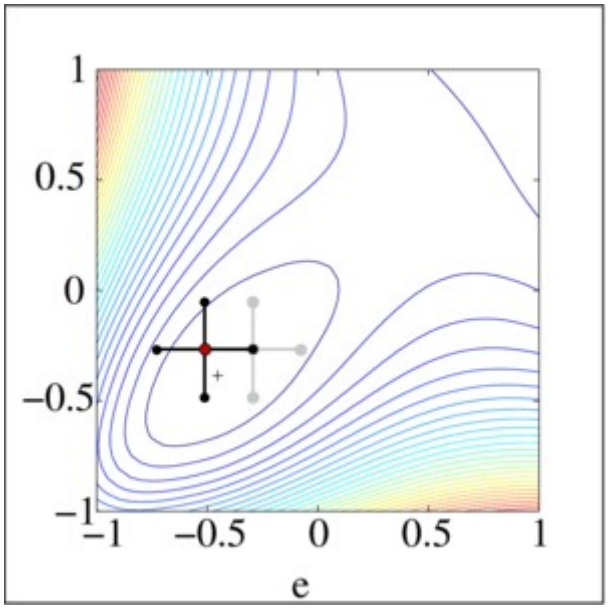
Nelder-Mead (4)

Fine up to n=10, does something for you up to n=25.

May converge prematurly (aligned vortices → loss of dimension).

There are other convergent (slightly more expensive) pattern search methods. Cf. Dolan, E.D.; Lewis, R.M.; Torczon, V.J. (2003). "On the local convergence of pattern search". SIAM Journal on Optimization. Expl.:





Recommended local derivative free optimizers in R^n

NEWUOA, M.J.D. Powell, 2004 : derivative free efficient optimizer based on a series of quadratic local approximations. Up to 160 variables.

Improved Nelder-Mead, e.g., GBNM (Globalized and Bounded Nelder-Mead algo., Luersen and Le Riche, 2004) : bounds on variables, restarts to avoid degeneracy and make it global, adaptive penalty for constraints. Up to 25 variables.

Outline

1. Introduction
2. Deterministic optimization : local, global.
3. Stochastic optimization
4. Adding problem specific knowledge

Global optimization : general points

Global optimization algorithms strike a compromise between an extensive ***exploration*** of the search space S (search in the volume ≠ line search of the gradient) and an ***exploitation*** (or ***intensification***) of already gathered information to make the search more rapid.

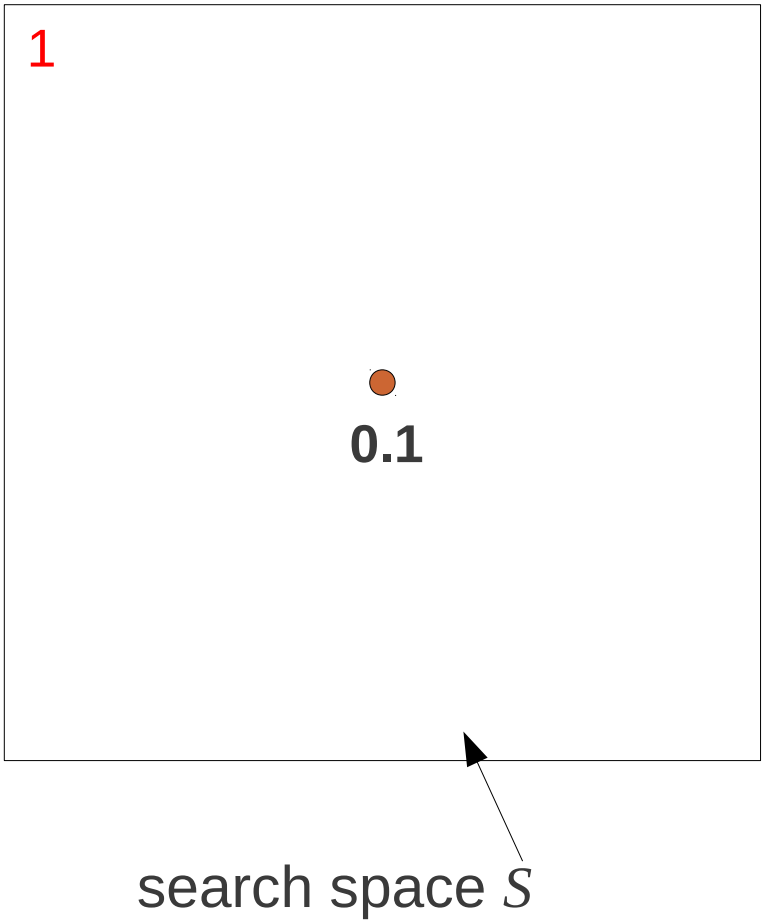
Global optimizers tend to be more costly but more robust than local optimizers.

Global derivative free deterministic optimization in R^n : DIRECT

(D.R. Jones et al., 1993)

DIRECT = *DIviding RECTangles* = a strategy to divide S into rectangular subregions which balances local and global search. « Lipschitzian optimization without Lipschitz constant ».

DIRECT : 2D example (1/3), Initialization

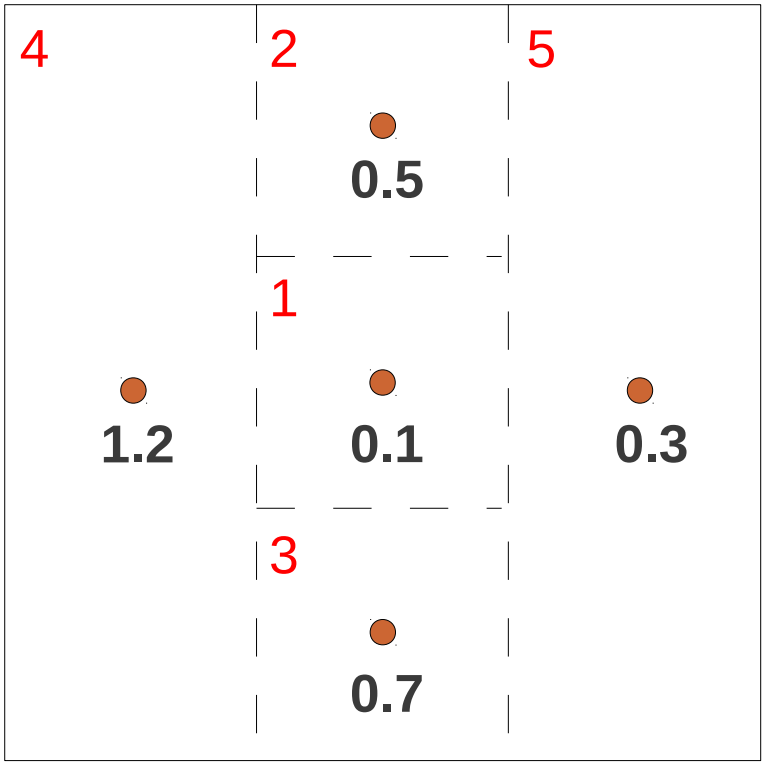


Rectangles
[f at center ; largest side Δ] :

1 : [0.1 ; 1]

DIRECT : 2D example (2/3)

The initial rectangle has been divided in thirds (a center stays a center).



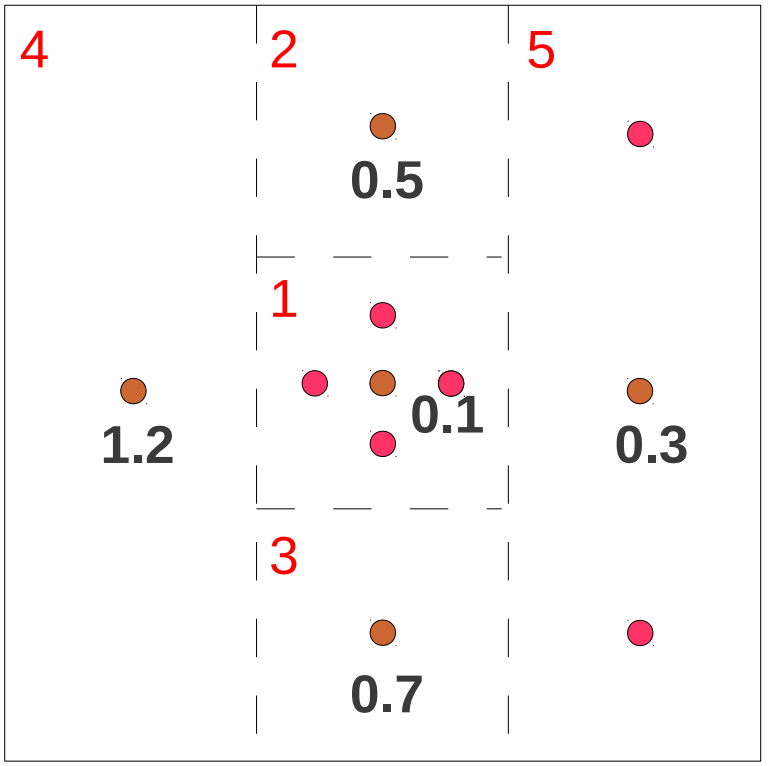
Rectangles
[center ; largest side Δ] :

1 : [0.1 ; 0.33]
2 : [0.5 ; 0.33]
3 : [0.7 ; 0.33]
4 : [1.2 ; 1.00]
5 : [0.3 ; 1.00]

Which rectangles should now be divided (so that f is evaluated at their centers) ?

DIRECT : 2D example (3/3)

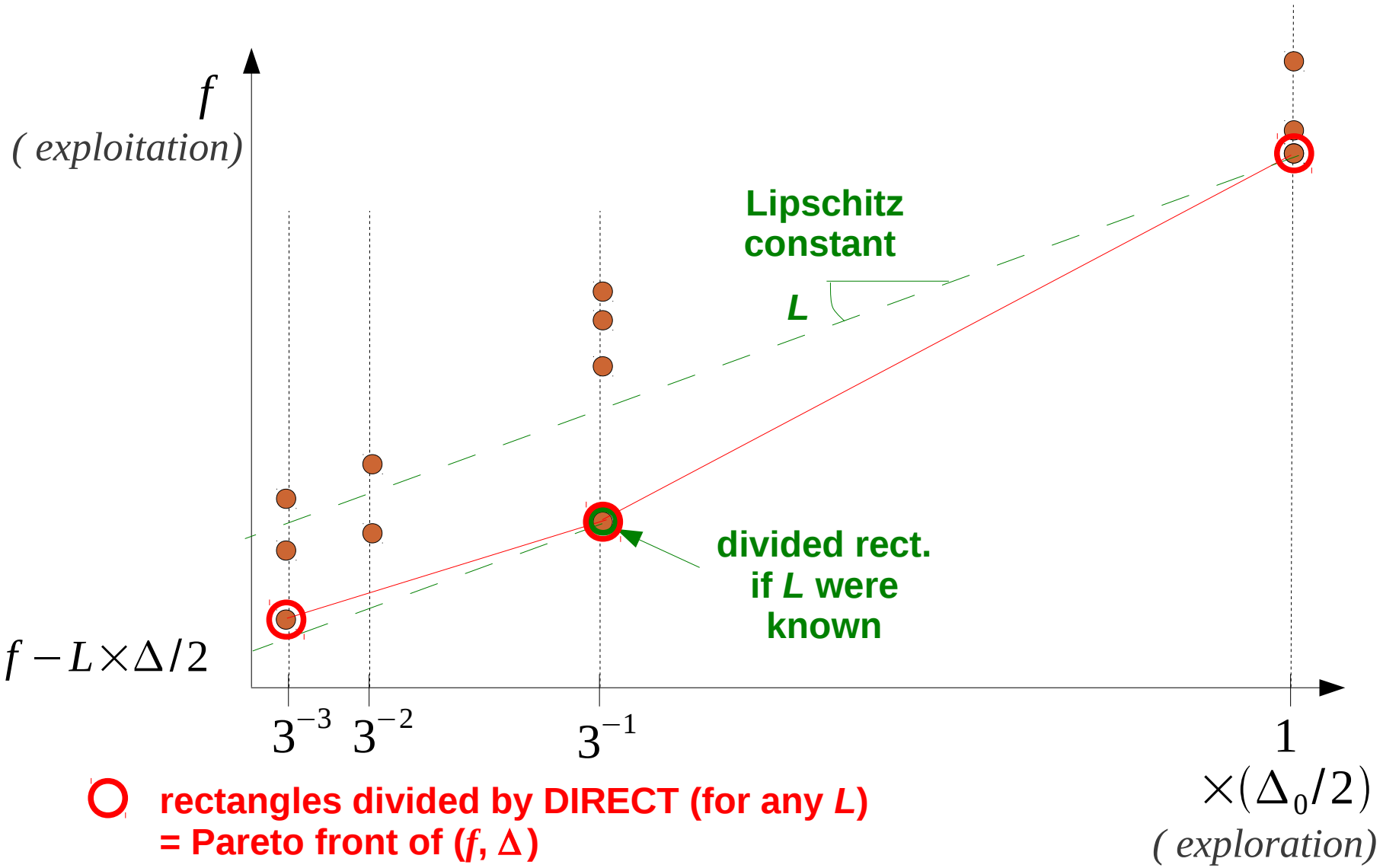
- the best rectangles (low f) ← exploitation, 1.
- the unknown rectangles (largest side) ← exploration, 5.



Rectangles
[center ; largest side Δ *]* :

1 : [0.1 ; 0.33]
2 : [0.5 ; 0.33]
3 : [0.7 ; 0.33]
4 : [1.2 ; 1.00]
5 : [0.3 ; 1.00]

DIRECT : multi-objective interpretation



○ rectangles divided by DIRECT (for any L)
= Pareto front of (f, Δ)

DIRECT : simplified flow chart

Initializations

While $t < t_{max}$ do,

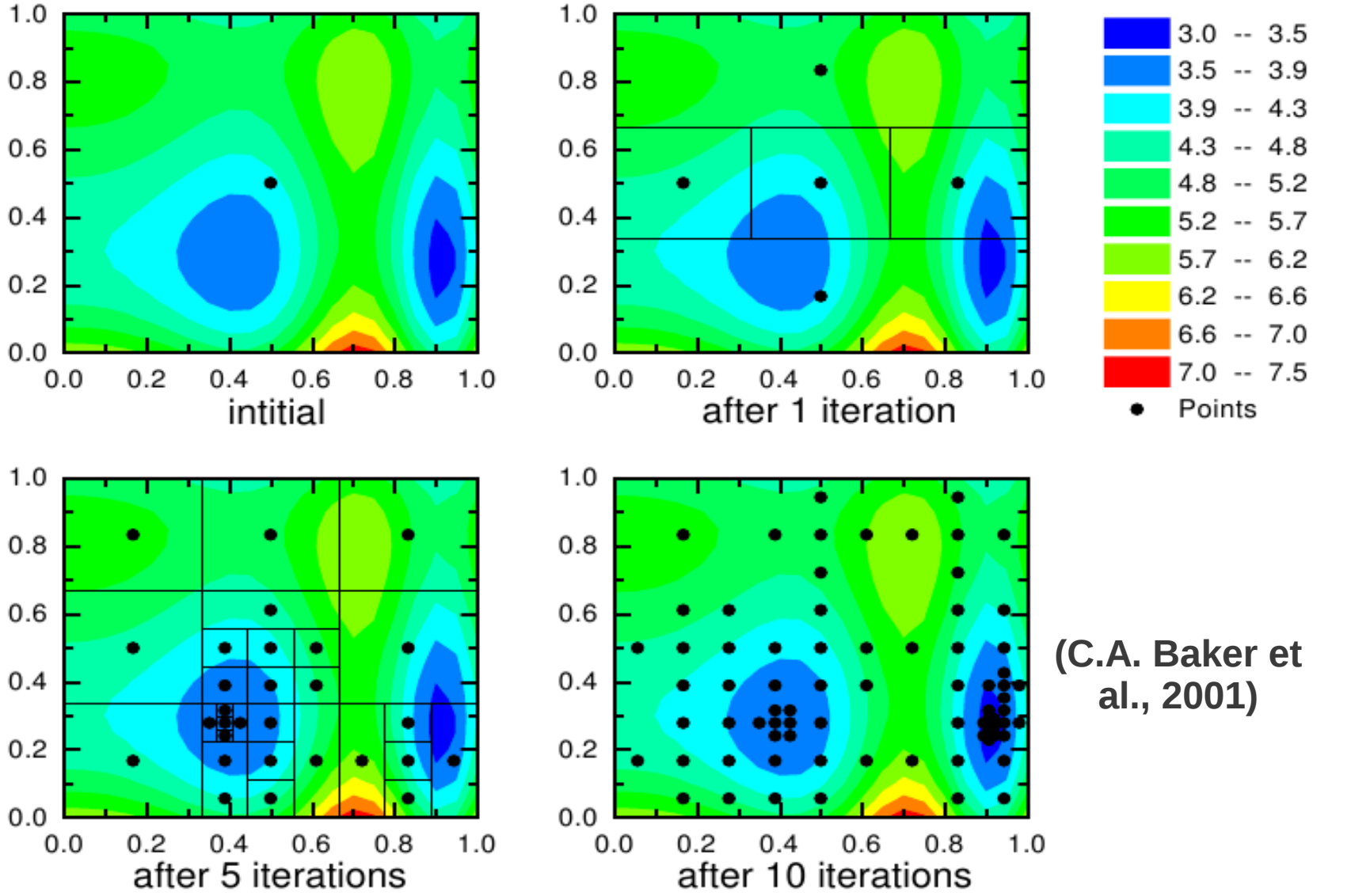
- Find the set of Pareto optimal rectangles, P
- For each rectangle j in P

  Divide along the largest sides of j and calculate f at the new centers

  Update the rectangles list, f_{min}, x_{min} and t
- End for each

End while

DIRECT : 2D example



(C.A. Baker et al., 2001)

DIRECT : conclusions

DIRECT converges to the global optimum when t (the number of calls to f) $\rightarrow \infty$ because it creates a dense series of points in S.

The determination of the Pareto front (sorting) has an $O(t)$ algorithmic complexity which is typically negligible w.r.t. the simulation costs (finite elements ...) .

At constant number of calls to f the accuracy of DIRECT degrades rapidly with the number of optimization variables n. OK up to $n \approx 10$.

No interaction between the f values at different rectangles explains the poor scaling in dimension $\rightarrow$ EGO (Efficient Global Optimization, D.R. Jones, 1998) is another, more mature deterministic global optimization method in R^n based on kriging, ok up to $n \approx 30$.

n

Outline

1. Introduction
2. Deterministic optimization
3. Stochastic optimization : continuous, discrete.
4. Adding problem specific knowledge

Introduction to stochastic optimization

Random numbers are versatile search engines (work both in R^n and / or N^n). They can also yield efficient methods.

Let $p^t(x)$ denote the probability density function of x at iteration t (e.g., after t evaluations of f). It represents the belief at t that the optimum x^* is at x.

How to « sample $p^t(x)$ » once (Scilab notation) ?

- if x is uniform between m and M, $X \sim U[m,M]$, call

```
x = m + rand(n,1).*(M-m)
```

- if x is (multi-)Gaussian with mean m and covariance matrix C, $X \sim N(m,C)$,call

```
x = m + grand(1,'mn',0,C)
```

Flow chart of a general stochastic optimizer

- **Initialize** t **and** $p^t(x)$
- **Sample** $x^{t+1} \sim p^t(x)$
- **Calculate** $f(x^{t+1})$
- **Update the distribution**

$$p^{t+1}(x) = Update(\, x^1, f(x^1), \ldots, x^{t+1}, f(x^{t+1}) \,)$$

or more often $p^{t+1}(x) = Update(\, p^t(x), x^{t+1}, f(x^{t+1}) \,)$

- **Stop or [** $t = t+1$ **and go back to Sample]**

A simple example in R^n : ES-(1+1)

Initializations : x, $f(x)$, m, C, t_{max}.

While $t < t_{max}$ do,

Sample $N(m,C)$ --> x'

Calculate $f(x')$, $t = t+1$

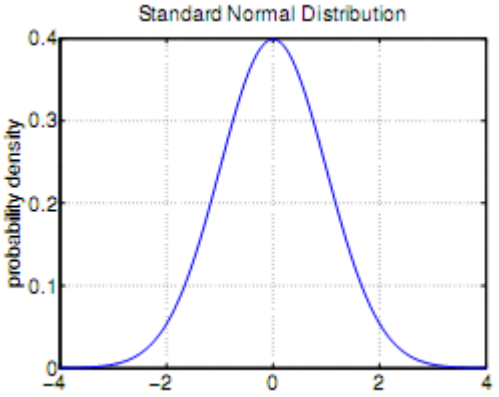
If $f(x')<f(x)$, $x = x'$, $f(x) = f(x')$ Endif

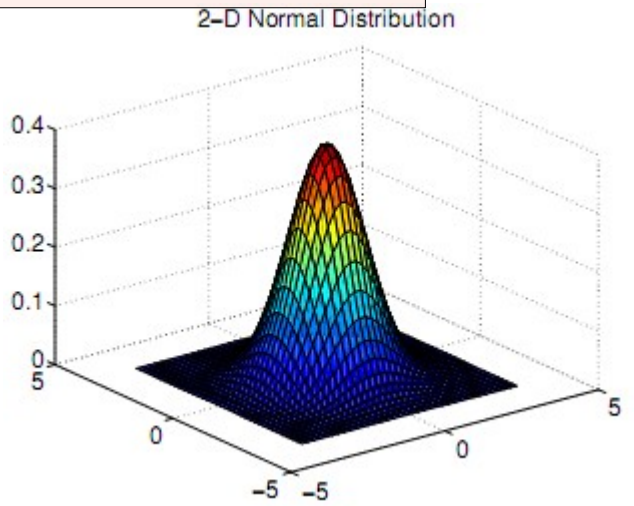
Update m (e.g., $m=x$) and C

End while

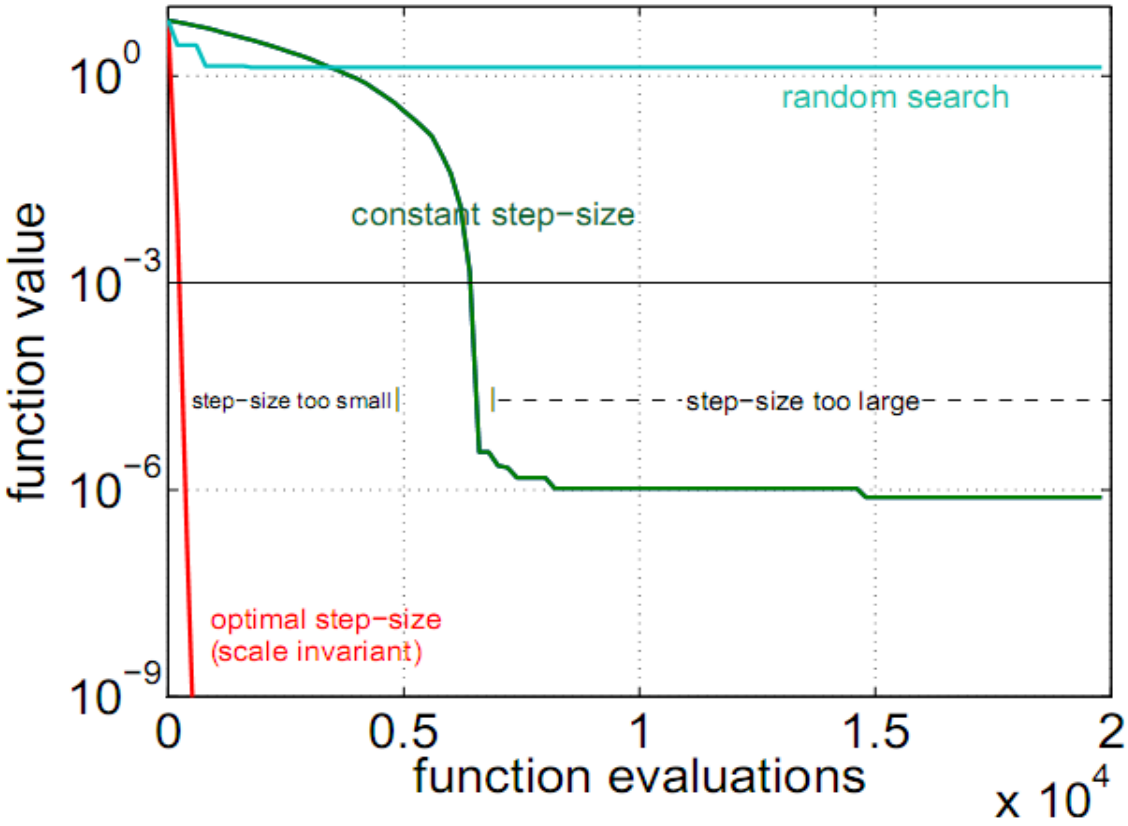
Normal law

$N(m,C)$





Adapting the step size (C^2) is important



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

(A. Auger et N. Hansen, 2008)

Above isotropic ES(1+1) : $\boldsymbol{C} = \sigma^2 \mathrm{I}$, σ is the step size.
With an optimal step size ($\approx \|x\| / n$) on the sphere function, performance degrades only in $O(n)$ (better than DIRECT).

Stochastic optimizer in R^n with a population : simplified CMA-ES

(N. Hansen et al., since 1996, now with A. Auger)

CMA-ES = *Covariance Matrix Adaptation Evolution Strategy* = optimization through sampling and updating of a multi-normal distribution.

A fully populated covariance matrix is build : pairwise variable interaction learned. Can adapt the step in any direction.

The state-of-the-art evolutionary / genetic optimizer for continuous variables.

flow-chart of CMA-ES

CMA-ES is an evolution strategy *ES-(μ,λ)* :

Initializations : m, C, t_{max}, μ , λ

While $t < t_{max}$ do,

Sample $N(m,C)$ --> $x^1,\ldots,x^\lambda$

Calculate $f(x^1),\ldots,f(x^\lambda)$, $t = t+\lambda$

Rank : $f(x^{1:\lambda}),\ldots,f(x^{\lambda:\lambda})$

Update m and C *with the* μ bests, $x^{1:\lambda}$,...,$x^{\mu:\lambda}$

End while

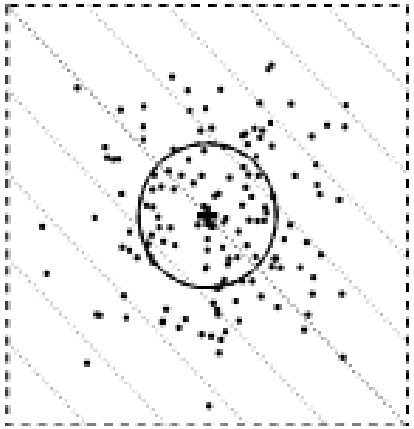
m *et* C are updated with

- the best **steps** (as opposed to points),
- a **time cumulation** of these best steps.

simplified CMA-ES : adapting C^2 with the last good steps

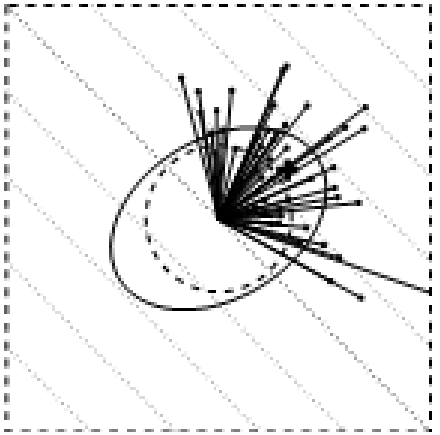
(A. Auger et N. Hansen, 2008)

Initialization : $m \in S$, $C = I$, $c_{cov} \approx 2/n^2$



sampling

$$x^i = m + y^i$$
$$y^i \propto N(0, C)$$
$$i = 1, \ldots, \lambda$$

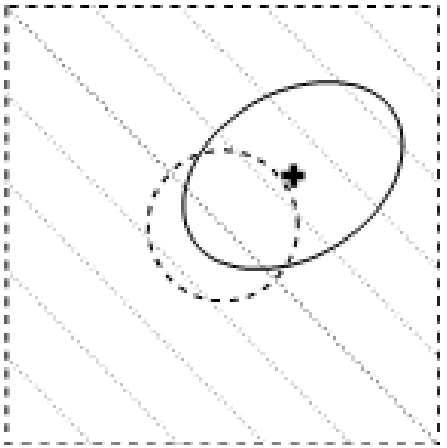


selection

$$y_w = \frac{1}{\mu} \sum_{i=1}^{\mu} y^{i:\lambda}$$

rank 1 C update

$$C \leftarrow (1 - c_{cov}) C + c_{cov}\, \mu\, y_w\, y_w^T$$



update m

$$m \leftarrow m + y_w$$

The state-of-the-art CMA-ES

(A. Auger and N. Hansen, *A restart CMA evolution strategy with increasing population size*, 2005)

Additional features :

- Steps weighting, $y_w = \sum_{i=1}^{\mu} w_i \, y^{i:\lambda}$
- Time cumulation of the steps.
- Simultaneous rank 1 and μ covariance adaptations.
- Use of a global scale factor, $C \rightarrow \sigma^2 \; C$.
- Restarts with increasing population sizes.

Has been used up to $n = 100$ continuous variables.

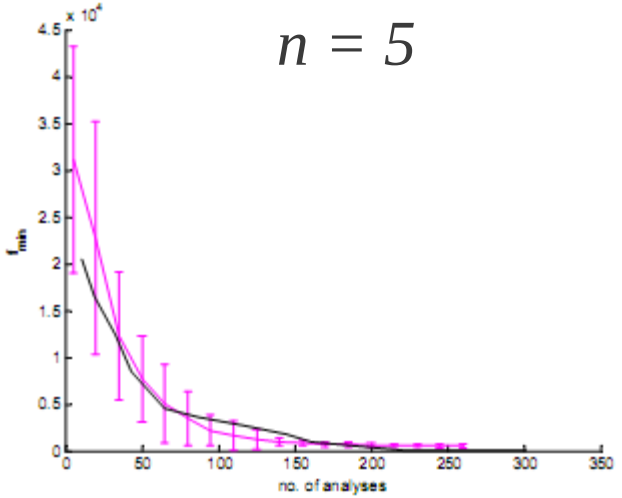
Comparison : DIRECT vs. ES-(1+1)

Sphere function.

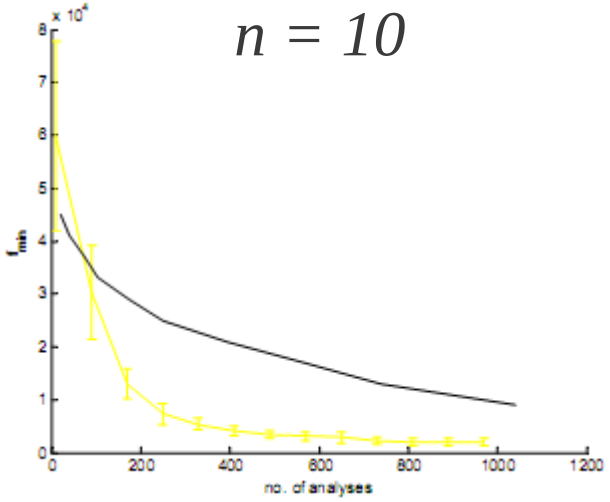
ES(1+1) with a constant step size σ = 0.1, 15 repetitions, coloured lines.
DIRECT : black line.

→ DIRECT does not scale well with ↑ n.

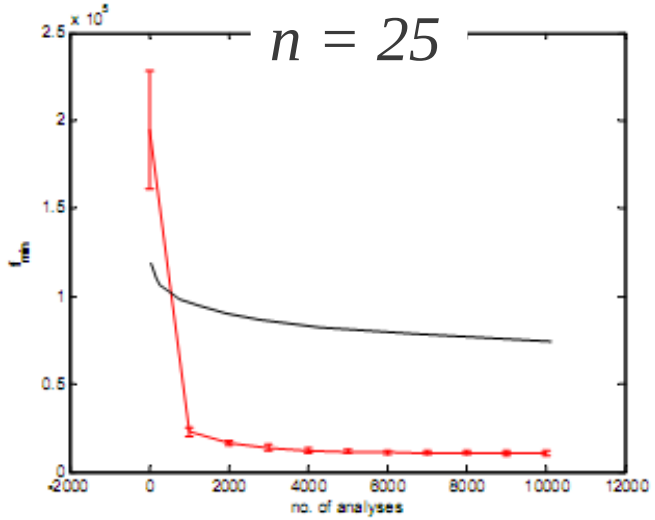
$n = 5$



$n = 10$



$n = 25$



Outline

1. Introduction
2. Deterministic optimization
3. Stochastic optimization : continuous, discrete.
4. Adding problem specific knowledge

The Univariate Marginal Density Algorithm (UMDA)

(Baluja 1994 – as PBIL – and Mühlenbein 1996)

$x \in S \equiv \{1,2,\ldots,A\}^n$ (alphabet of cardinality A)

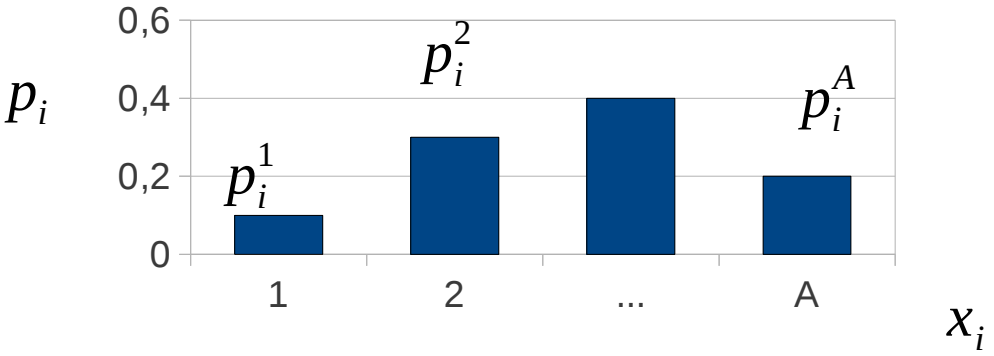
e.g. $\{-45^o\ ,\ 0^o\ ,\ 45^o\ ,\ 90^o\}^n$ (fiber orientations)

e.g. $\{\text{matl1}\ ,\ \ldots\ ,\ \text{matlA}\}^n$ (material choice)

The algorithm is that of a population based stochastic optimization (see CMA-ES) with different sampling and updating of p^t.

p^t assumes that the variables are independent (drop ^{t}),

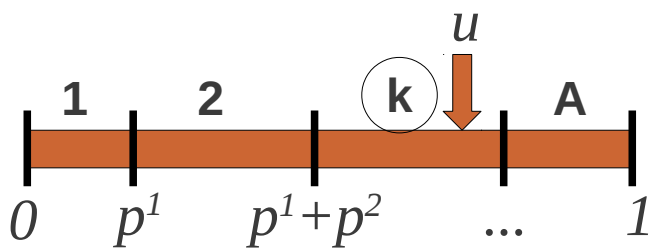
$$p(x) = \prod_{i=1}^{n} p_i(x_i)$$



$$\sum_{j=1}^{A} p_i^j = 1$$

UMDA (2)

Sampling :



For $i=1,n \;\; u_i \sim U[0,1]$

If $0 \le u_i \le p_i^1 \;\Rightarrow\; x_i=1$

If $\sum_{j=1}^{k} p_i^j \le u_i < \sum_{j=1}^{k+1} p_i^j \;\Rightarrow\; x_i=k$

If $\sum_{j=1}^{A-1} p_i^j \le u_i \le 1 \;\Rightarrow\; x_i=A$

Learning :

Select the μ best points out of λ , $f(x^{1:\lambda}) \le f(x^{2:\lambda}) \le \ldots \le f(x^{\mu:\lambda})$

p_i^j is the frequency of j at position i in the μ bests

$$p_i^j = \frac{\sum_{k=1:\lambda}^{\mu:\lambda} I(x_i^k = j) + \varepsilon}{\mu(1+\varepsilon)} \quad , \quad I(x_i^k = j)=1 \text{ if } x_i^k = j \;\; , \;\; =0 \text{ otherwise}$$

$$p_i^j \ge \frac{\varepsilon}{\mu(1+\varepsilon)}$$ (minimum frequency for ergodicity)

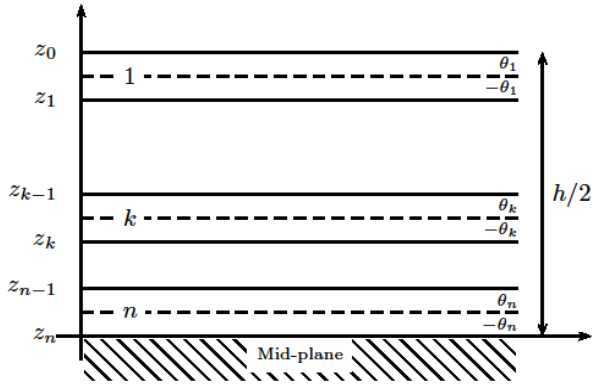
Application to a laminate frequency problem (1)

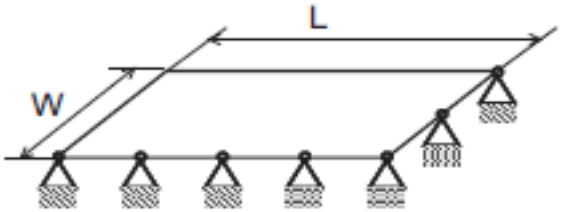
(from Grosset, L., Le Riche, R. and Haftka, R.T., A double-distribution statistical algorithm for composite laminate optimization, SMO, 2006)

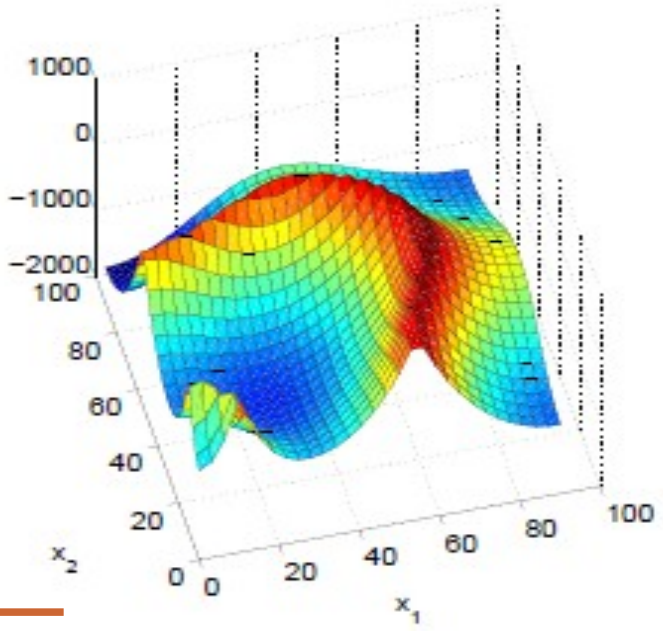
$max_x\ f_1(x_1,\ldots,x_{15})$, the first eigenfreq. of a simply supported plate

such that $0.48 \le \nu_{eff}(x) \le 0.52$

where $x_i \in \{0^o, 15^o, \ldots, 90^o\}$







the constraint is enforced by penalty and creates a narrow ridge in the design space

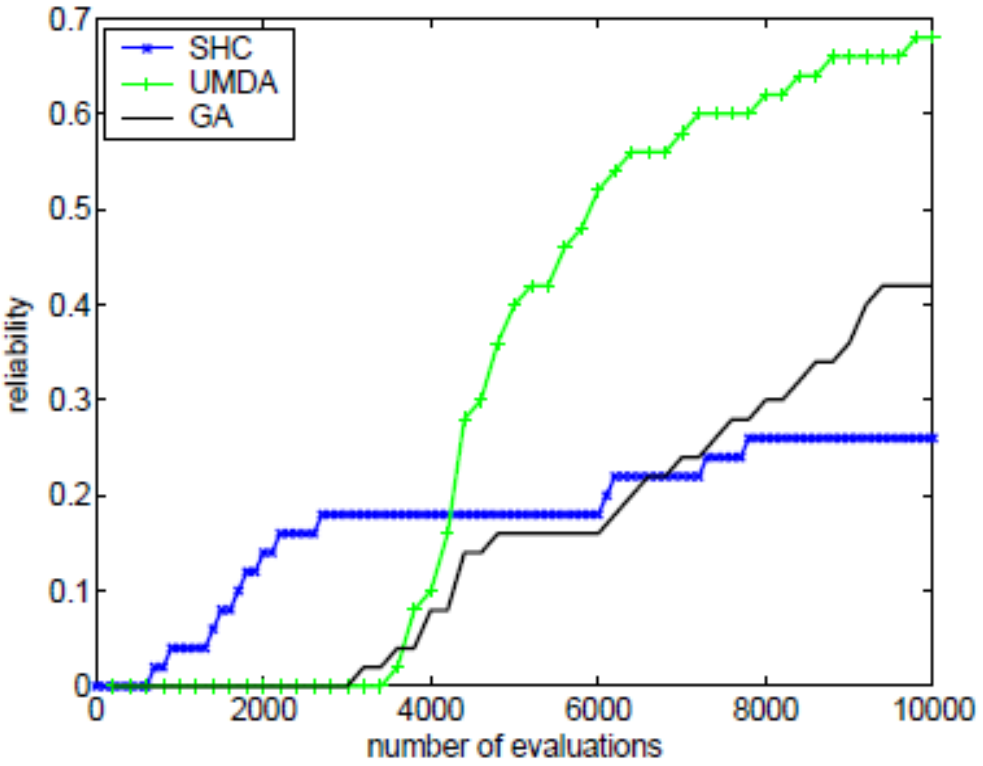
Application to a laminate frequency problem (2)

Optimum : $\left[90_4^o/\pm 75^o/\pm 60_2^o/\pm 45_5^o/\pm 30_5^o\right]_s$

Compare UMDA to a GA (genetic algorithm) and SHC (Stochastic Hill Climber)

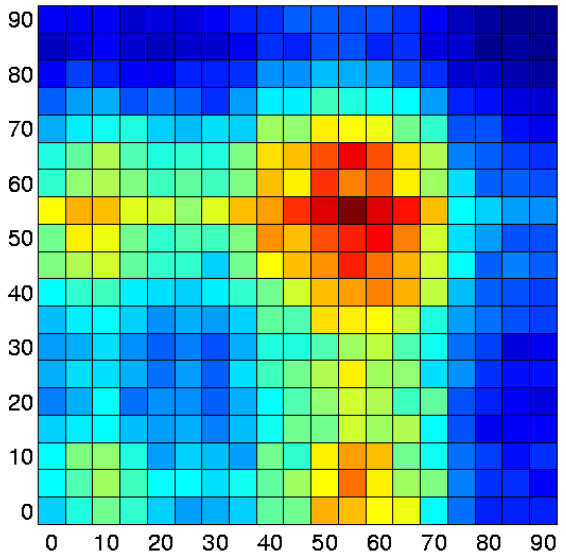
Reliability = probability of finding the optimum at a given cost.

UMDA performs fairly well on this problem.

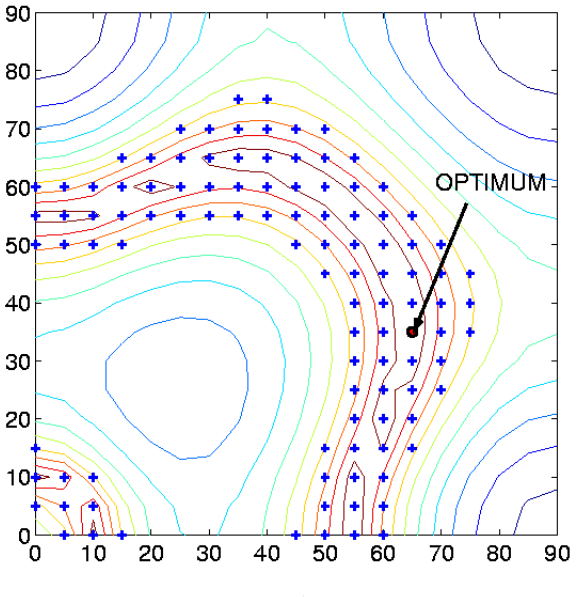


Application to a laminate frequency problem (3)

density learned by UMDA (2D)

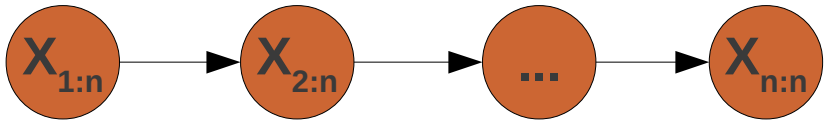


contour lines of the penalized objective function



Stochastic discrete optimization : learning the variables dependencies

More sophisticated discrete optimization methods attempt to learn the couplings between variables. For example, with pairwise dependencies :



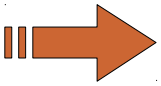
$$p(x) = p(x_{1:n})\, p(x_{2:n}|x_{1:n}) \ldots p(x_{n:n}|x_{n-1:n})$$

Trade-off : richer probabilistic structures better capture the objective function landscape but they also have more parameters → need more f evaluations to be learned (// complex constitutive equations).

MIMIC (Mutual Information Maximizing Input Clustering) algorithm : De Bonnet, Isbell and Viola, 1997.
BMDA (Bivariate Marginal Distribution Algorithm) : Pelikan and Muehlenbein, 1999.

Outline

1. Introduction
2. Deterministic optimization
3. Stochastic optimization
4. Adding problem specific knowledge



Adding problem specific knowledge to optimization problems

Problem specific knowledge can be used to

• change the formulation of the optimization problem to improve its mathematical structure (conditionning, make it quadratic, …),
Ex : $x \equiv 1/EI$ for a displacement of a bended beam,

• decompose the problem into a series of easier subproblems,

• use a low fidelity simulator to capture variables dependencies in stochastic optimization,
(implicit equivalent : add specific knowledge to an evolutionary algorithm through coding, crossover, mutation choices)

• More importantly : calculate sensitivities (see Franz-Joseph Barthold's talk).

Example in composites
Use of the lamination parameters

- $\mathbf{V} \equiv$ Lamination parameters = geometric contribution of the plies to the stiffness.
- E.g. in-plane problem:

$$\begin{Bmatrix} A_{11} \\ A_{22} \\ A_{12} \\ A_{66} \end{Bmatrix} = h \begin{bmatrix} U_1 & U_2 & U_3 \\ U_1 & -U_2 & U_3 \\ U_5 & 0 & -U_3 \\ U_4 & 0 & -U_3 \end{bmatrix} \begin{Bmatrix} 1 \\ V_1^* \\ V_3^* \end{Bmatrix}$$

h: total laminate thickness, U_i's: material invariants

- Symmetric balanced laminates $[\pm\theta_1, \pm\theta_2, \ldots, \pm\theta_n]_s$:

$$V^*_{\{1,3\}} = \frac{2}{h}\int_0^{h/2} \{\cos 2\theta, \cos 4\theta\} dz = \frac{1}{n}\sum_{k=1}^{n} \{\cos 2\theta_k, \cos 4\theta_k\}$$

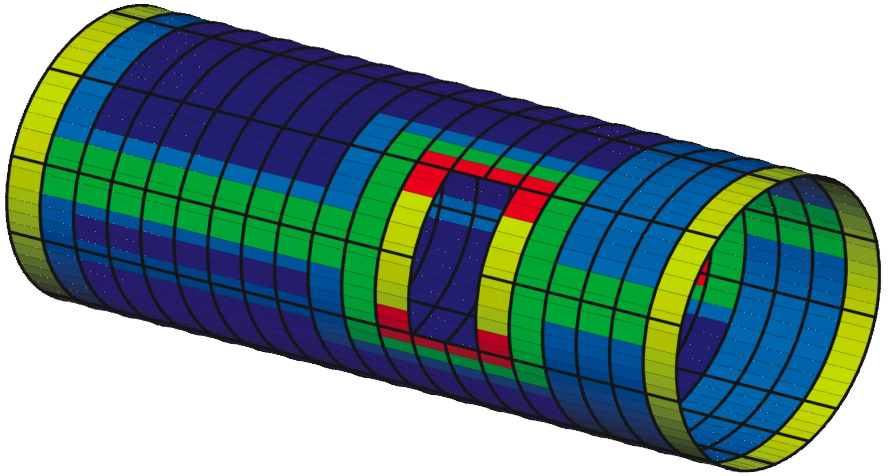
Simplifications : fewer *V*'s than fiber angles. Often, the *V*'s are taken as continuous.

Example in composites : Use of lamination parameters in problem decomposition

(Liu, Haftka, and Akgün, « Two-level composite wing structural optimization using response surfaces », 2000.
Merval, Samuelides and Grihon, « Lagrange-Kuhn-Tucker coordination for multilevel optimization of aeronautical structures », 2008.)

Initial problem :

Optimize a composite structure made of several assembled panels by changing each ply orientation
→ many discrete variables



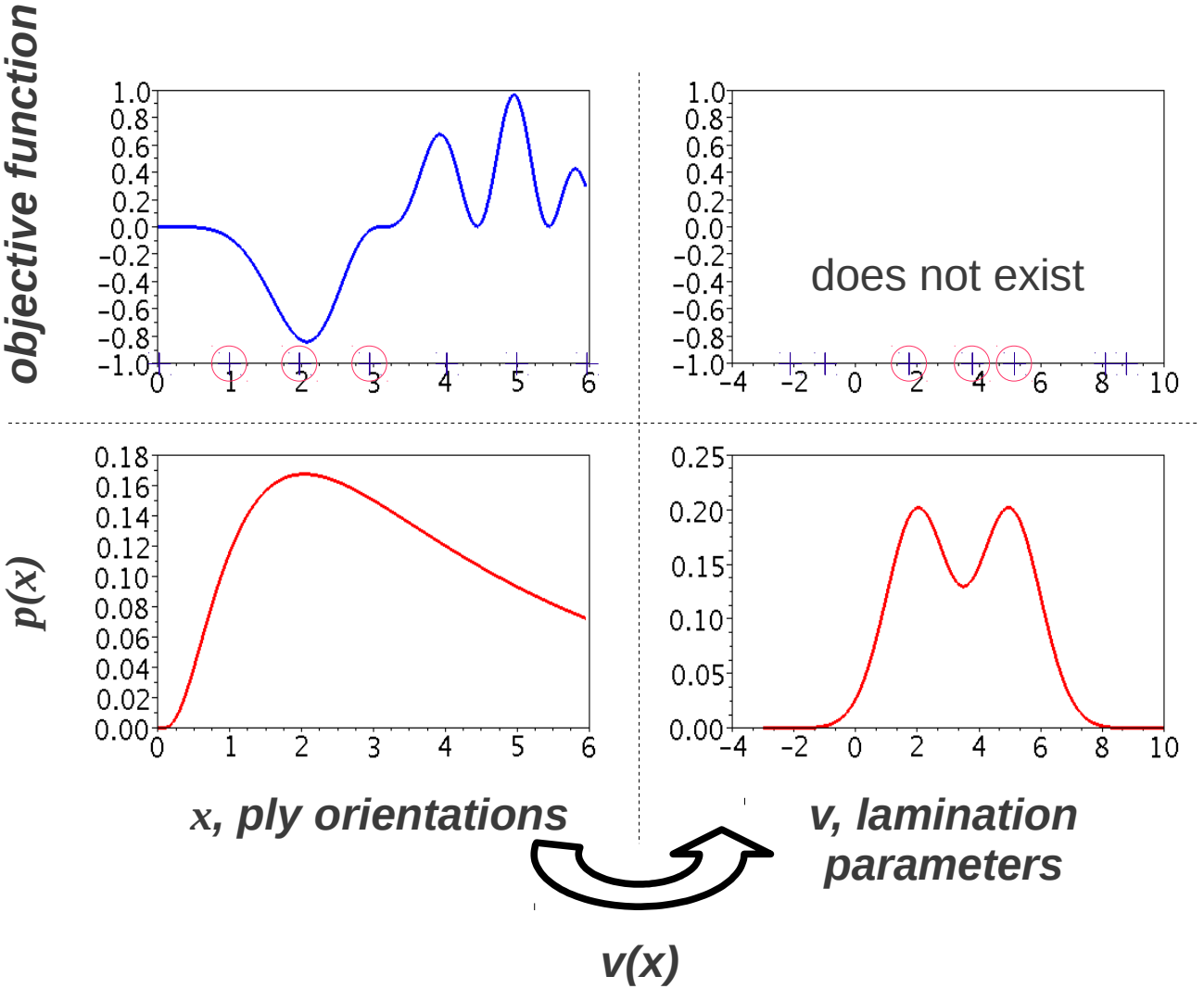
Decomposed problem :

Structure level
Optimize a composite structure made of several assembled panels by changing the lamination parameters of each panel
→ few continuous variables

optimal *V*'s

Laminate level
Minimize the distance to target lamination parameters by changing the ply orientations
→ few discrete variables

Expl : Use lamination parameters to capture dependencies in stochastic optimization (1)



(DDOA, Double Density Optimization Algorithm, Grosset, Le Riche et Haftka, SMO 2006)

$$p_{DDOA}(x) \sim p_X(x).p_V(v(x))$$

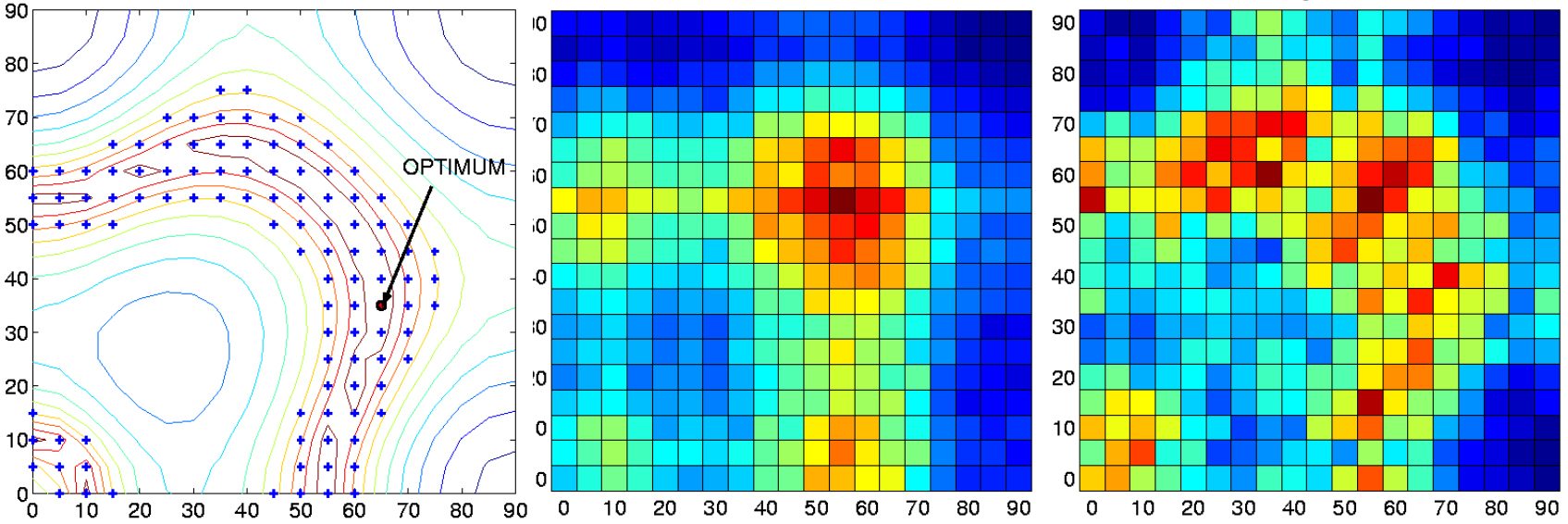
Expl : Use lamination parameters to capture dependencies in stochastic optimization (2)

- $p_X(x)$ and $p_V(v)$ can be simple densities, without variables couplings (→ easy to learn), yet $p_{DDOA}(x)$ is a coupled density.

f(x) and selected points

$$p_X(x) = \prod_{i=1}^{n} p_i(x_i)$$

$p_{DDOA}(x)$



- One half of the algorithm searches in a low dimension space.
- DDOA can be applied to other problems (use low fidelity model for *v*).