Squaring the engineering optimization circle : distributed global optimization algorithms for computationally expensive problems

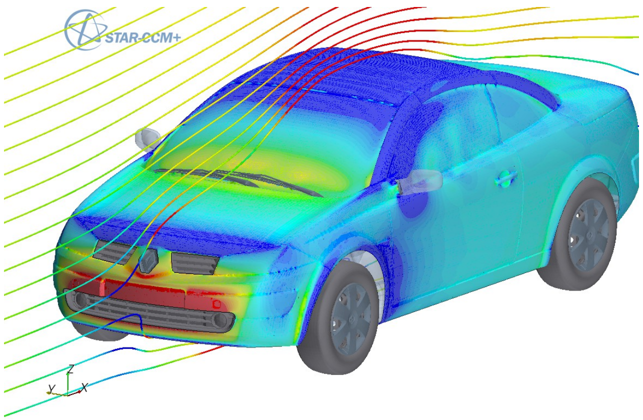
Rodolphe Le Riche[1,2] , Ramunas Girdziusas[2], Diane Villanueva[2], Gauthier Picard[2] and David Ginsbourger[3]

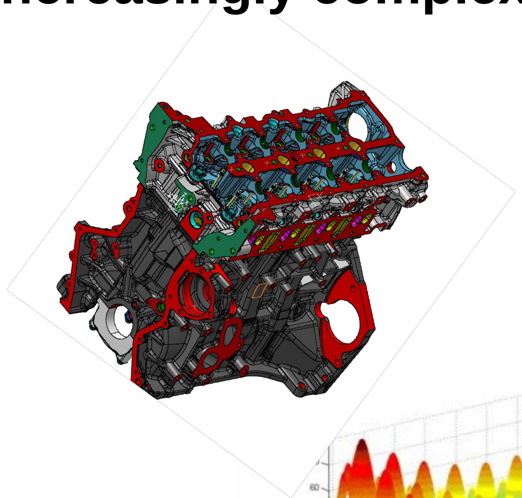
[1] CNRS ; [2] Ecole des Mines de Saint-Etienne ;
[3] Univ. of Bern ;

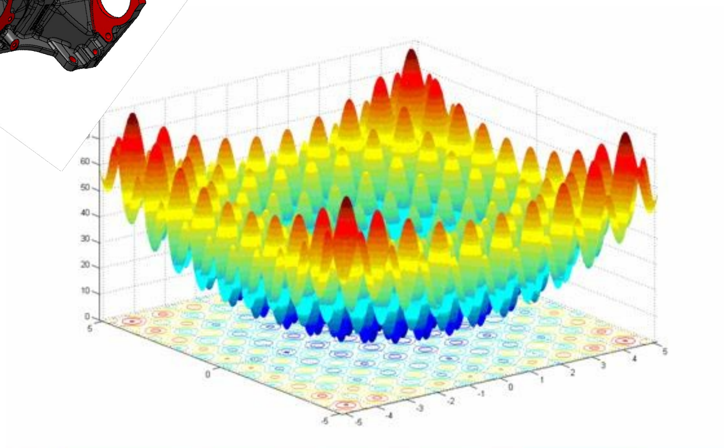
RIO 2012, Univ. de Valenciennes

Globally optimal solutions and realistic models : squaring the engineeting optimization circle

There is a demand for optimizing increasingly complex models ...







…. and finding the globally best solution(s)

A realistic simulation takes typically 0,5h
In 10D, global optimization needs (say) 10000 evaluations → ~208 days

Globally optimal solutions and realistic models : squaring the engineeting optimization circle

The computational cost is an endless obstacle to engineering optimization.

Approaches :

- reducing the cost of the simulation : reduced models, metamodels.
- making the optimization problem easier : reducing the number of design variables.
- Having more efficient global optimization algorithms.
- taking advantage of parallel computing infrastructures.

some topics we have been looking at lately

Outline of the talk

metamodel ← kriging

parallelized global optimization algorithms ← parallelized Expected Improvement (EI) algorithms, dynamic partitioning (agents).

1. Introduction to kriging and optimization
2. Synchronous parallel EI
3. Asynchronous parallel EI
4. Embarrassingly parallel EI algorithms
5. An agent-based dynamic partitioning algorithm

centralized

decision

decentralized

Related work

Many in evolutionary computing, e.g., Branke et al., *Distribution of evolutionary algorithms in heterogeneous networks, GECCO 2004* : island models and migration schemes adapted to heterogenous computing ressources. Not adapted to expensive objective functions.

Local pattern search : E.g., Kolda, *Revisiting asynchronous parallel pattern search for nonlinear optimization*, SIAM J. Optimization, 2005.

Deterministic global optimization : E.g., Regis and Shoemaker, *Parallel radial basis function methods for the global optimization of expensive functions*, Eur. J. of OR, 2007. ← Closest contribution to the current work, yet synchronous.

Problem statement, notation

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

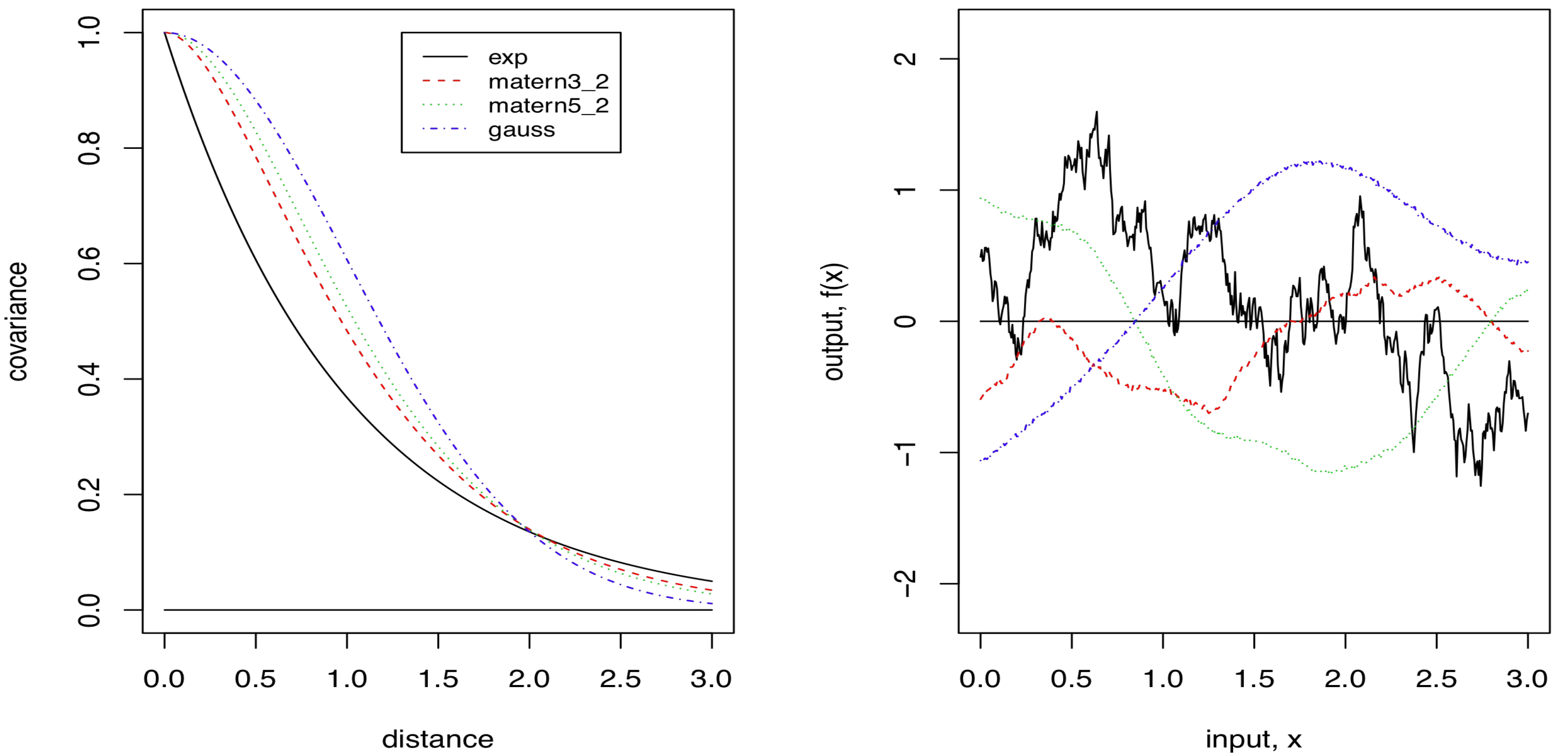
$$g(y(x)) \leq 0$$

x : design variables
y : numerical simulator (analytical, finite elements, coupled sub-models …).
f and g : optimization criteria (objective function and constraints)

Working assumptions : kriging (1)

Assume that *f(x)* can be seen as a trajectory of a stationary Gaussian random process, *F(x)*.

A fairly large class of functions can be represented in this way. They are parameterized by the covariance *Cov(F(x),F(x'))*.



(O. Roustant, HDR, 2011)

Working assumptions : kriging (2)

More precisely, assume that f(x) can be represented by a stationary **conditioned** Gaussian random process (+ linear trend) : kriging,

$$F(x) = a_0 + a_1\mu_1(x) + \ldots + a_L\mu_L(x) + Z(x) \;\Big|\; \begin{pmatrix} F(x^1) = f(x^1) \\ \ldots \\ F(x^m) = f(x^m) \end{pmatrix}$$

$$\Rightarrow \boxed{\left[F(\boldsymbol{x}^{\text{new}}) \mid F(\boldsymbol{x}) = f(\boldsymbol{x})\right] \sim \mathrm{N}\left(m_k(\boldsymbol{x}^{\text{new}}), C_k(\boldsymbol{x}^{\text{new}})\right)}$$

m_k and C_k are analytically known.

For example if the linear trend is known (simple kriging) :

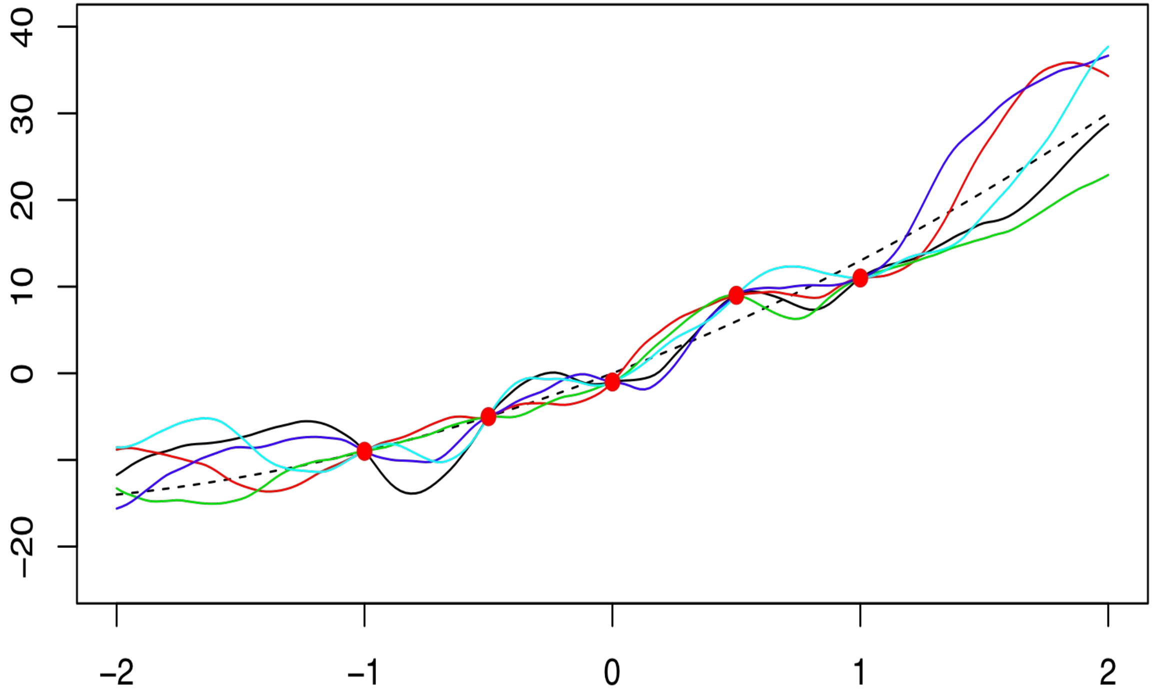
$$m_k(\boldsymbol{x}^{\text{new}}) = \mu(\boldsymbol{x}^{\text{new}}) + c^T(\boldsymbol{x}^{\text{new}})\, C^{-1}\left(f(\boldsymbol{x}) - \mu(\boldsymbol{x})\right)$$
$$C_k(\boldsymbol{x}^{\text{new}}) = C(\boldsymbol{x}^{\text{new}}) - c^T(\boldsymbol{x}^{\text{new}})\, C^{-1} c(\boldsymbol{x}^{\text{new}})$$

Kriging example

Red bullets are calculated points, $\left(x_i, f\left(x_i\right)\right)$

Paths of $\left[F\left(\boldsymbol{x}^{\mathbf{new}}\right) \mid F\left(\boldsymbol{x}\right)=f\left(\boldsymbol{x}\right)\right]$ in colour,

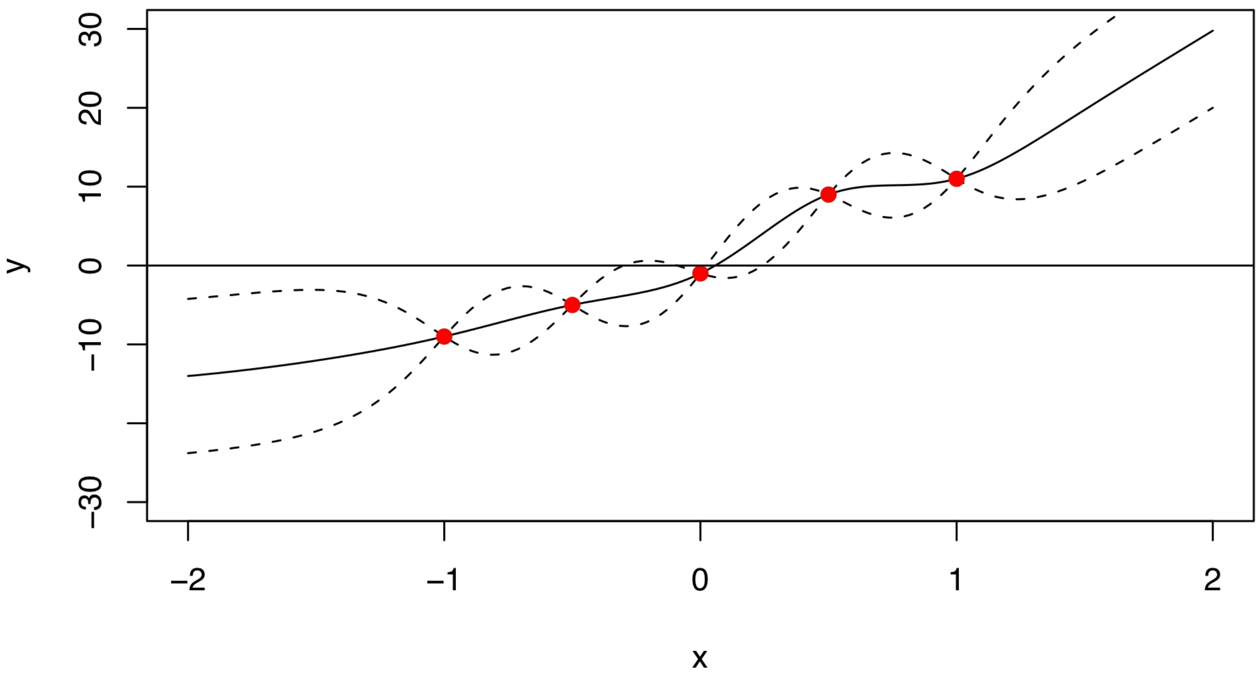
$\mu\left(x\right)$ black dotted line.



Kriging example

$m_k(x)$, kriging average, black line,

$\pm s_k(x)$, $\pm$std dev. prediction interval, dotted lines.

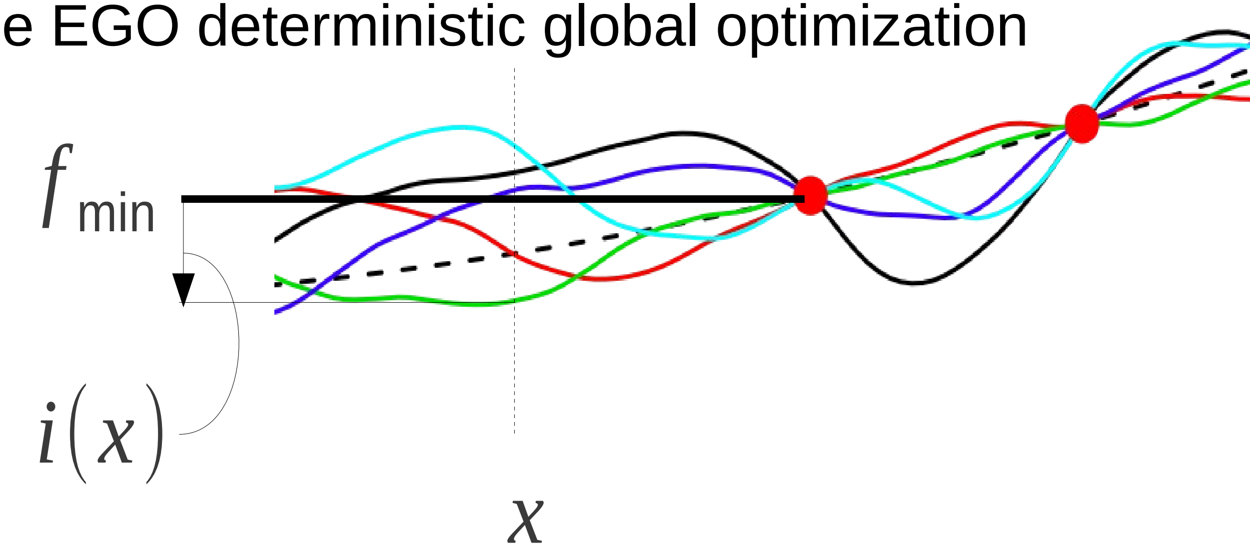


(one point-) Expected improvement

A natural measure of progress : the improvement,

$$I(x) = \left[f_{\min} - F(x)\right]^{+} \mid F(\mathbf{x}) = f(\mathbf{x}) \quad , \quad \text{where } [.]^{+} \equiv max(0,.)$$

- The expected improvement is known analytically.
- It is a parameter free measure of the exploration-intensification compromise.
- Its maximization defines the EGO deterministic global optimization algorithm (D. Jones, 1998).
  (sequential)

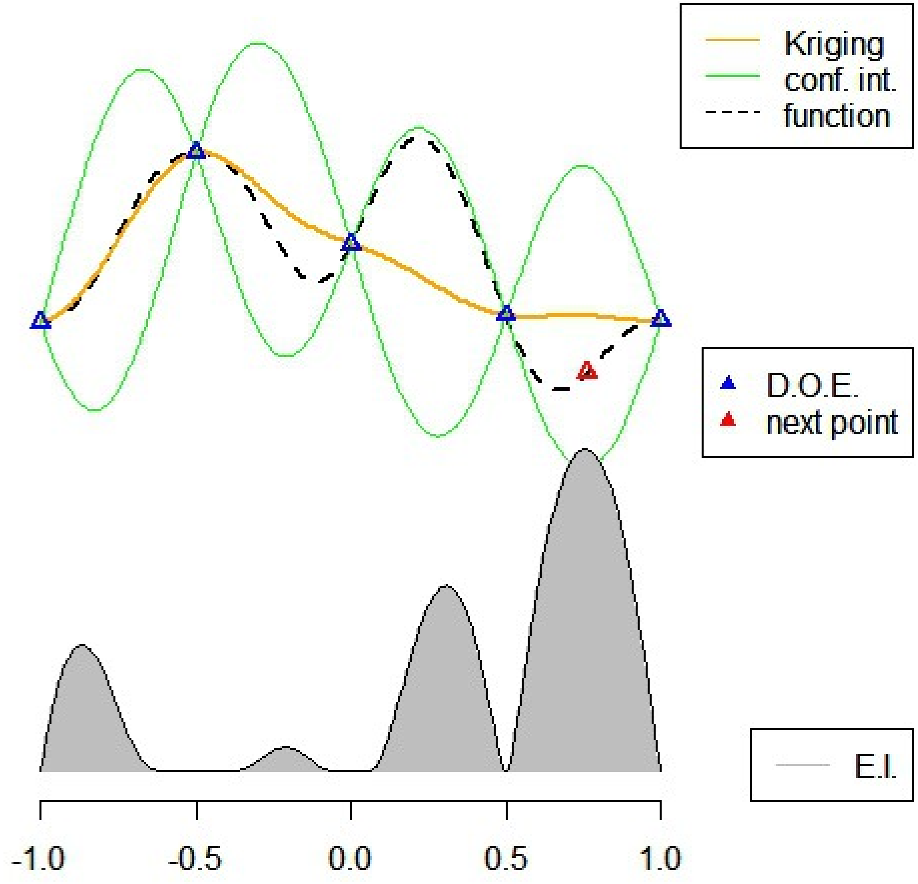


$$EI(x) = s_k(x) \times \left(u(x)\Phi(u(x)) + \varphi(u(x))\right) \quad , \quad \text{where } u(x) = \frac{f_{\min} - m_k(x)}{s_k(x)}$$

One EGO iteration

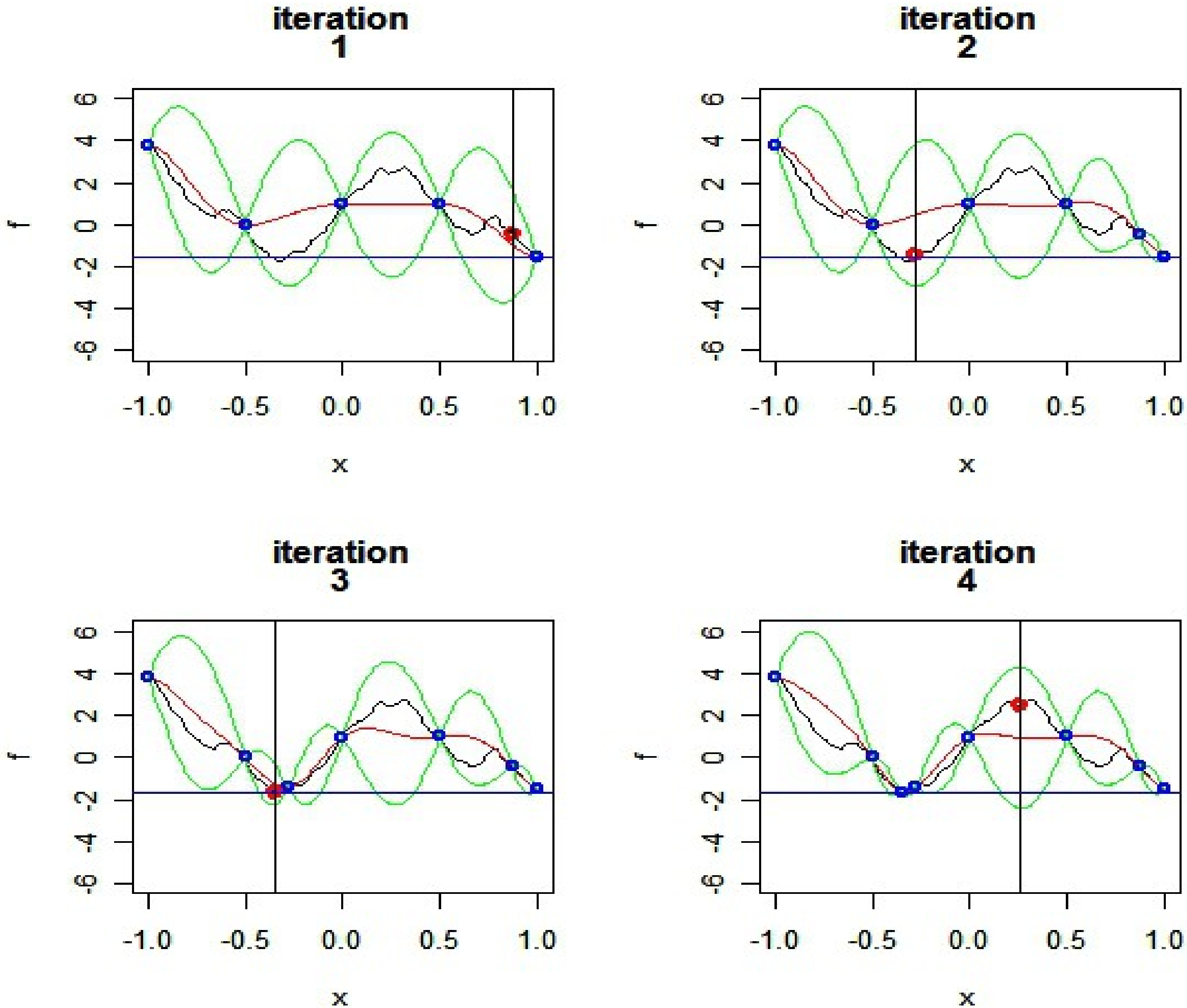
At each iteration, EGO adds to the t known points the one that maximizes EI,

$$x^{t+1} = arg\,max_x\, EI(x)$$



then, the kriging model is updated ...

EGO : example


iteration
1
iteration
2
iteration
3
iteration
4
f
x
-1.0
-0.5
0.0
0.5
1.0
6
4
2
0
-2
-4
-6

Outline of the talk

1. Introduction to kriging and optimization
2. Synchronous parallel EI
3. Asynchronous parallel EI
4. Embarrassingly parallel EI algorithms
5. An agent-based dynamic partitioning algorithm

From EGO to asynchronous parallel EI algorithm Selected bibliography of the team

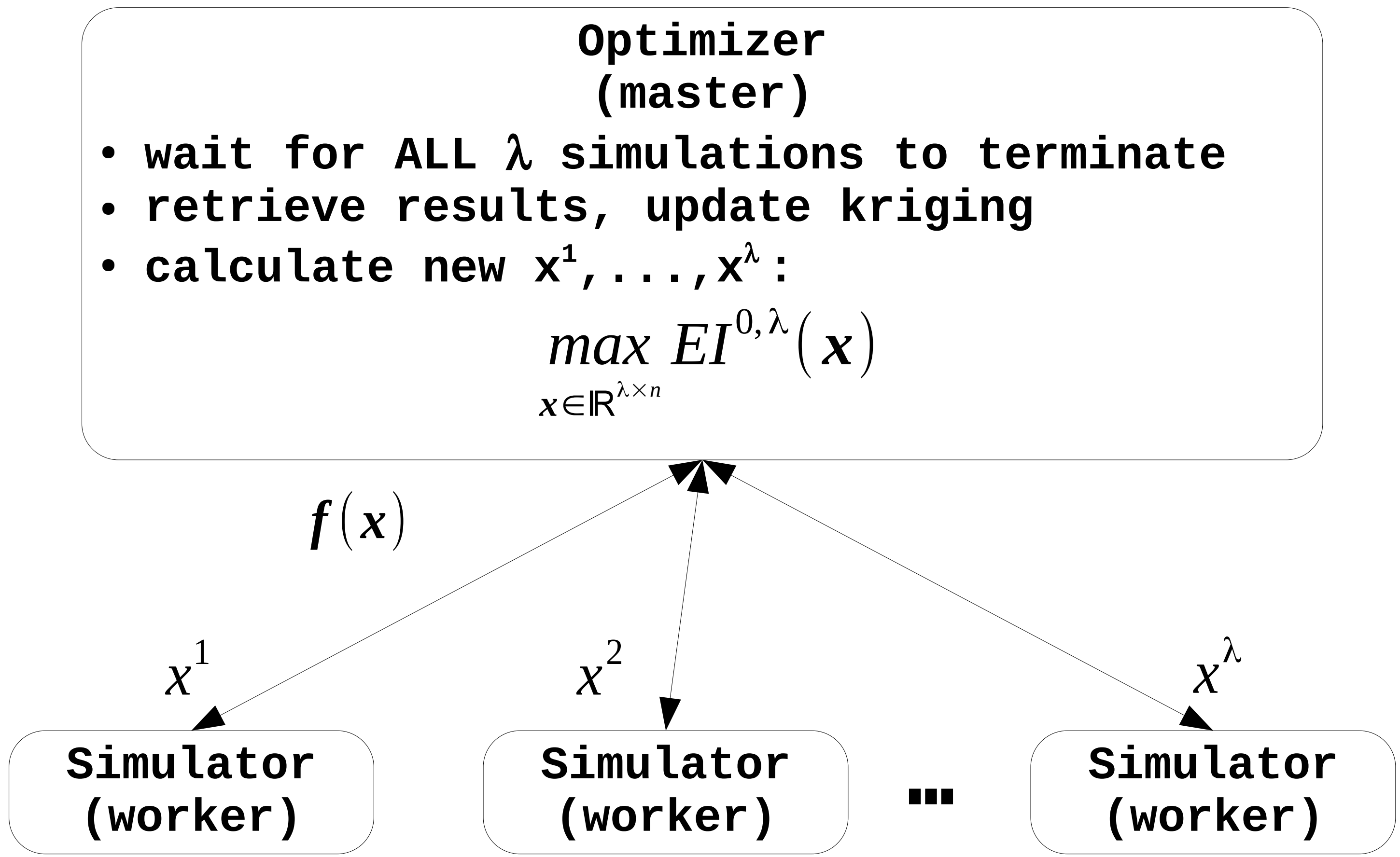
• D. Ginsbourger, R. Le Riche and L. Carraro, Kriging is well-suited to parallelize optimization, CIEOP, 2010.

• D. Ginsbourger, J. Janusevskis and R. Le Riche, Dealing with asynchronicity in parallel Gaussian Process based global optimization, Technical report hal-00507632, 2010.

• Janusevskis, J., Le Riche, R., Ginsbourger, D. and R. Girdziusas, Expected improvements for the asynchronous parallel global optimization of expensive functions : potentials and challenges, to be published in Learning and Intelligent Optimization, selected articles from the LION 6 Conference (Paris, Jan. 16-20, 2012), LNCS 7219, Lecture Notes in Computer Science series, Springer Verlag, Aug. 2012

Synchronous parallel EI : flow chart

A master-worker structure between computing nodes :



Synchronous parallel EI : criterion

λ nodes are available for new simulations at $x^1,\ldots,x^\lambda \; (\equiv \boldsymbol{x})$

→ choose $\boldsymbol{x}$ so that they maximize the synchronous λ points EI

$$EI^{0,\lambda}(\boldsymbol{x}) = E\left[f_{\text{min}} - min(F(\boldsymbol{x}))\right]^+ \mid F(x^{1\ldots m}) = f(x^{1\ldots m})$$

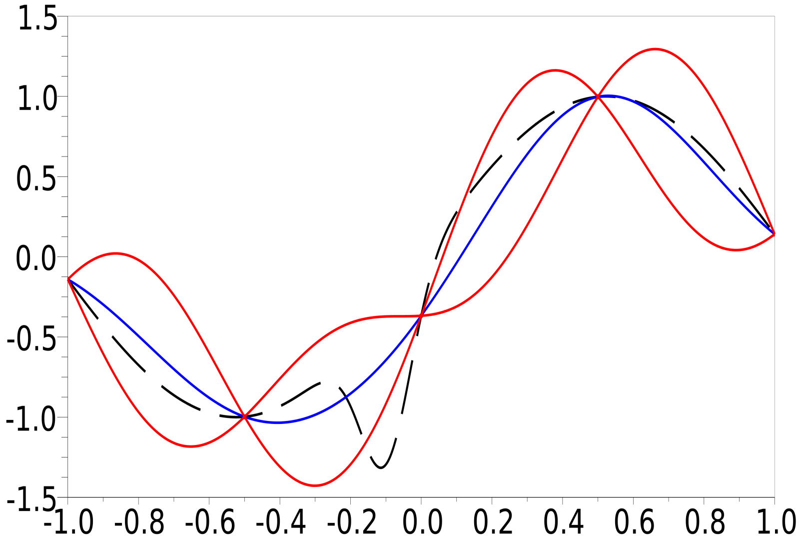
Compare to the sequential 1 point EI, from the EGO algorithm :

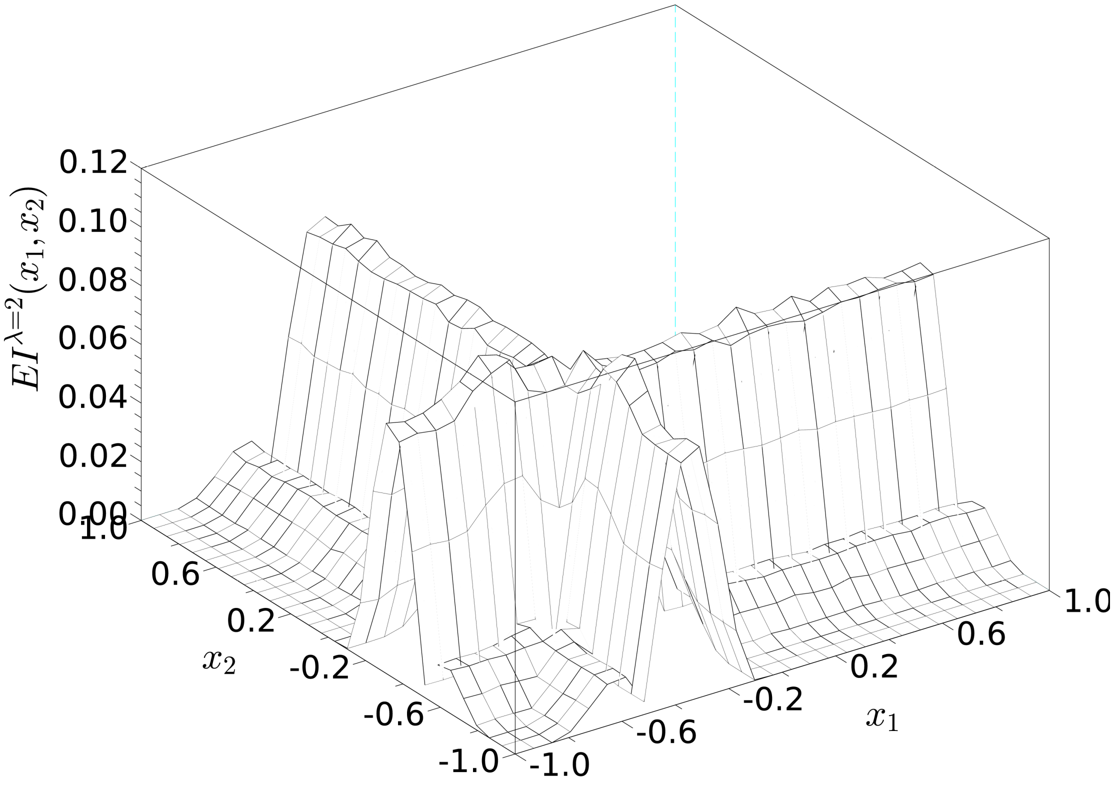
$$EI(x) \equiv EI^{0,1}(x) = E\left[f_{\text{min}} - F(x)\right]^+ \mid F(x^{1\ldots m}) = f(x^{1\ldots m})$$

Numerical estimation of $EI^{0,\lambda}$

$EI^{\mu,\lambda}$ not known analytically (excepted $EI^{0,1}$ and $EI^{0,2}$, recent efficient estimations by C. Chevalier and D. Ginsbourger at Bern Univ.) : Monte Carlo estimation.

Expl : 1D function (black dotted), $EI^{0,2}(x_{1,}\, x_2)$
10000 MC simulations



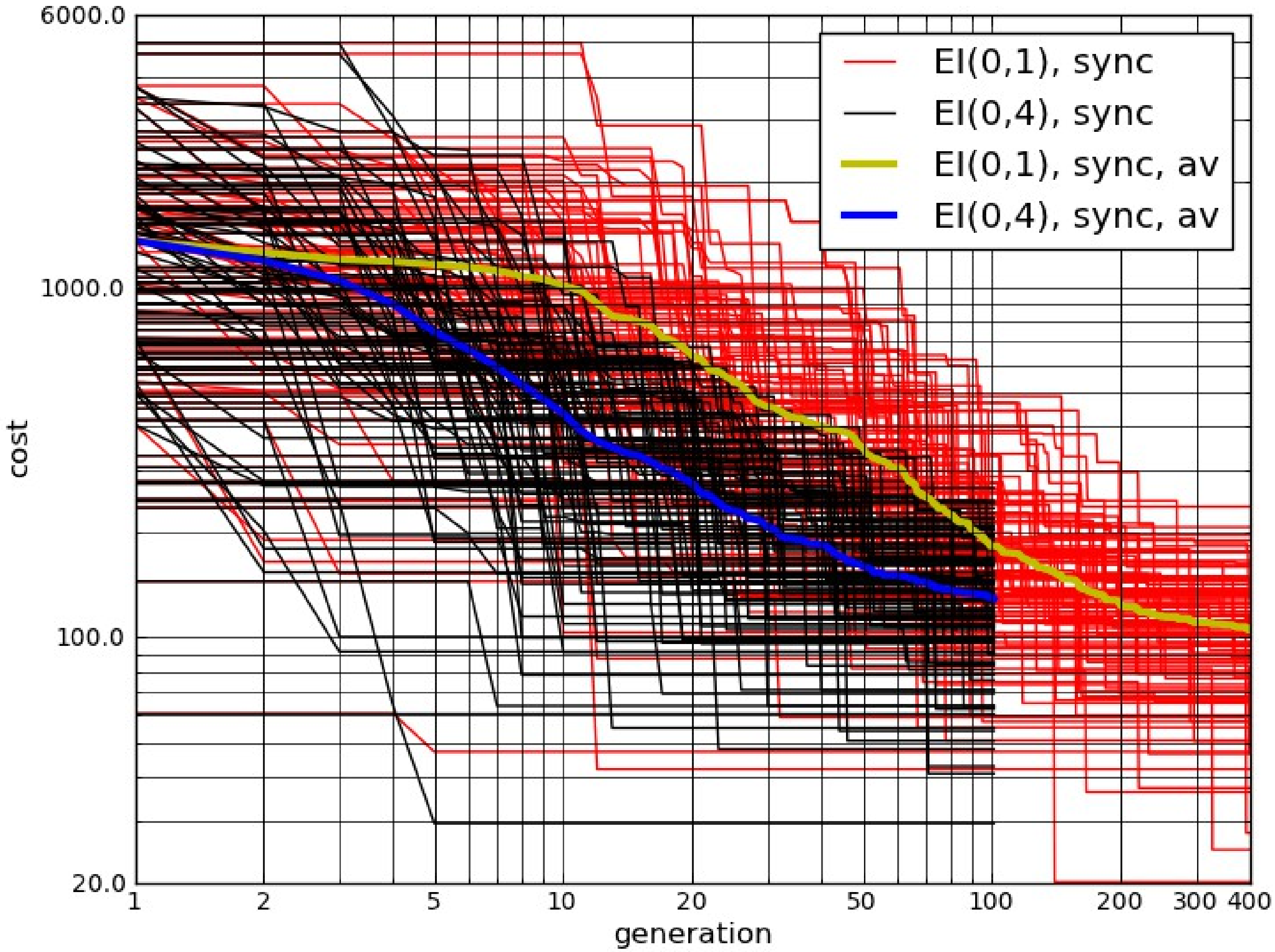


Test functions

Label	Cost function	Domain	Minimal value	Modality
"michalewicz2d"	$\sum_{i=1}^{2} \sin(x_i) \sin^2(i x_i^2/\pi)$	$[0,5]^2$	-1.841	multimodal
"rosenbrock6d"	$\sum_{i=1}^{5} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	$[0,5]^6$	0	unimodal
"rank1approx9d"	$\|\mathbf{A}_{4\times 5} - \mathbf{x}_{1\ldots 4}\mathbf{x}_{5\ldots 9}^T\|_2,\, a_{ij} \sim U(0,1)^1$	$[-1,1]^9$	0.712	bimodal

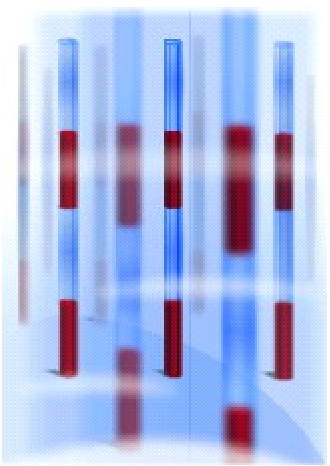
Results with $EI^{0,\mu}$

(6D Rosenbrock function)

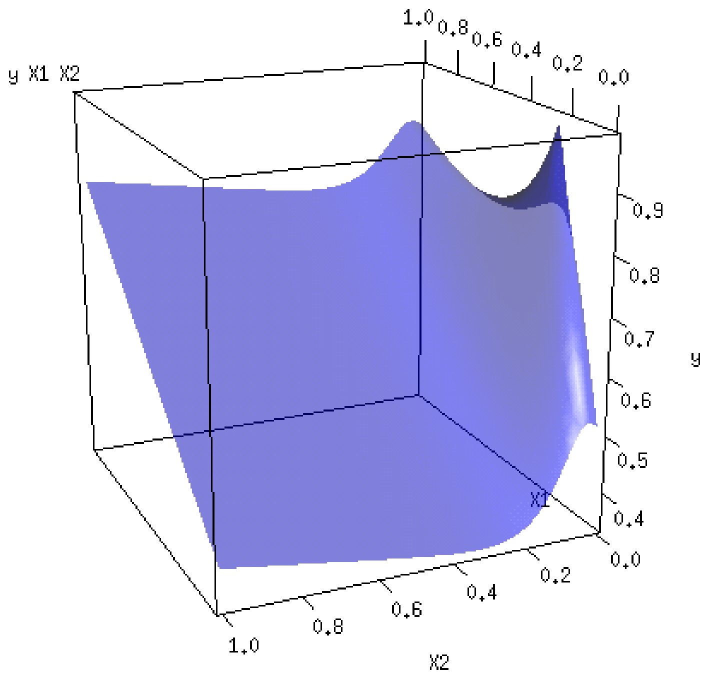


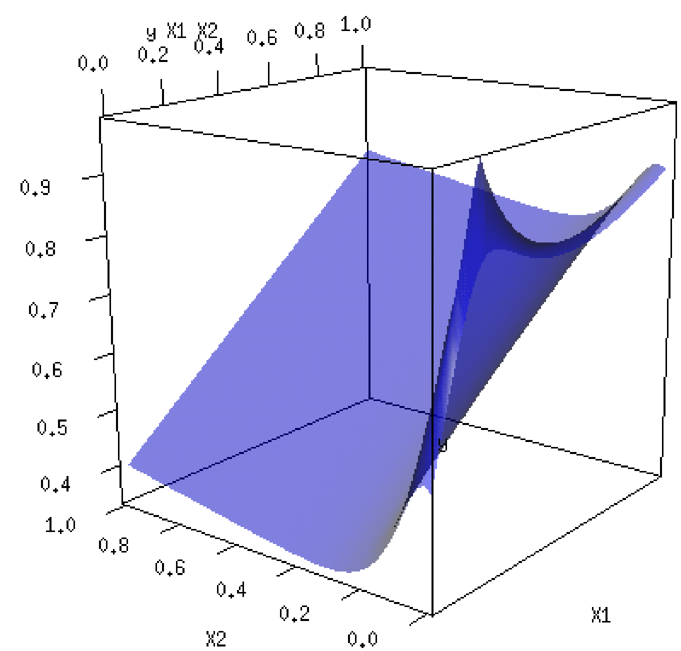
Results with $EI^{0,\lambda}$: nuclear safety test case

(from Yann Richet, IRSN)



- Maximization of 2D criticality model to check safety.
- Plutonium powder in storage can arrays ==> neutronic interaction between neutronic cans.
- x_1 : density of water between cans
- x_2 : density of plutonium powder
- Y : neutronic reactivity of the system (>1.0 means uncontrolled chain reaction, to be avoided)
- « true » maximum is ~0.99 .



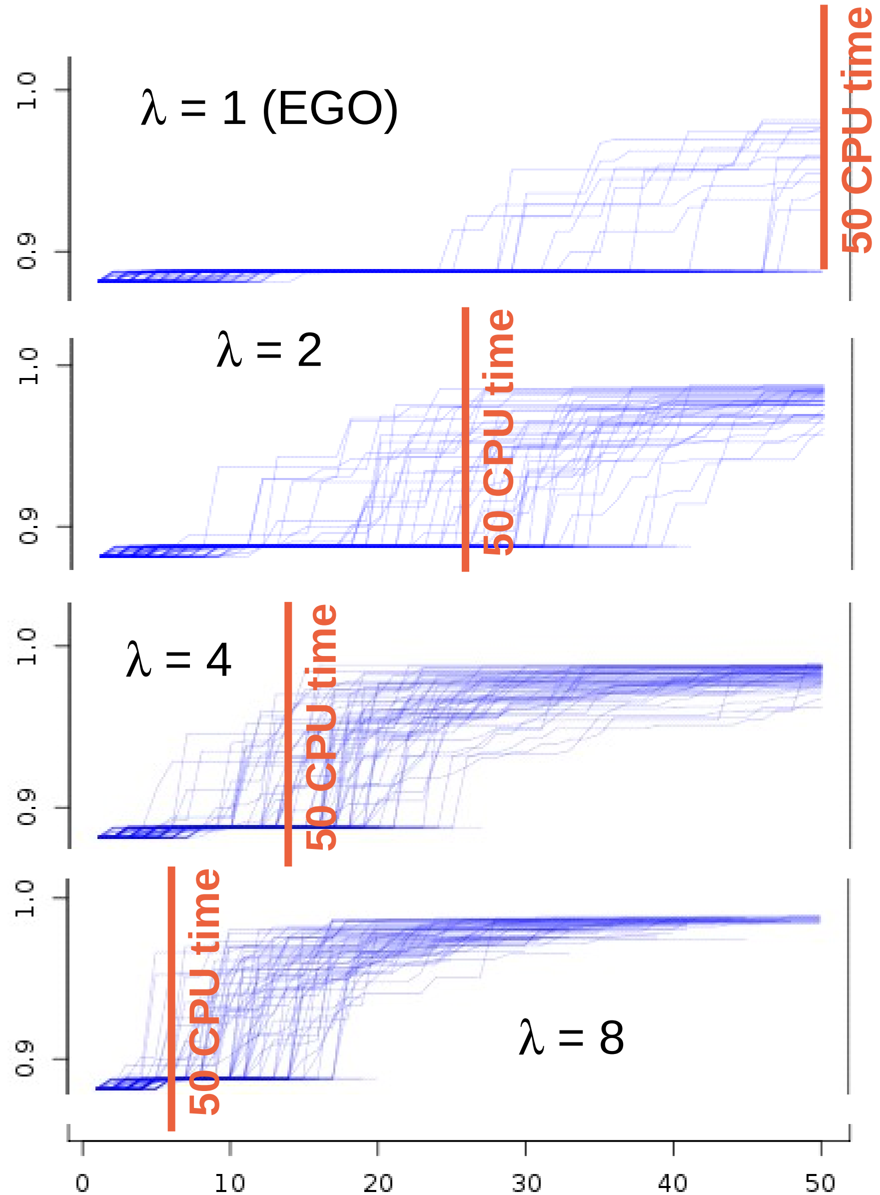


Results with $EI^{0,\lambda}$: nuclear safety test case

(from Yann Richet, IRSN)

100 EGO runs with different starting LHS (9 points + 4 corners)

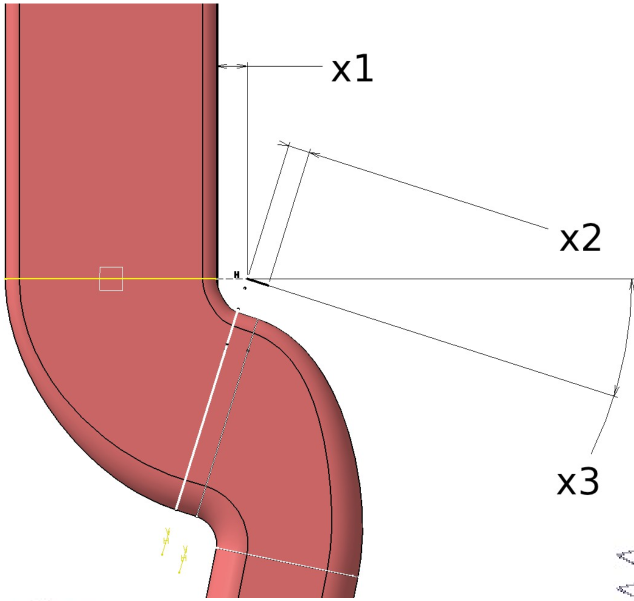
End EGO when
either max(EI(x)) <1.e-20
or > 50 iterations

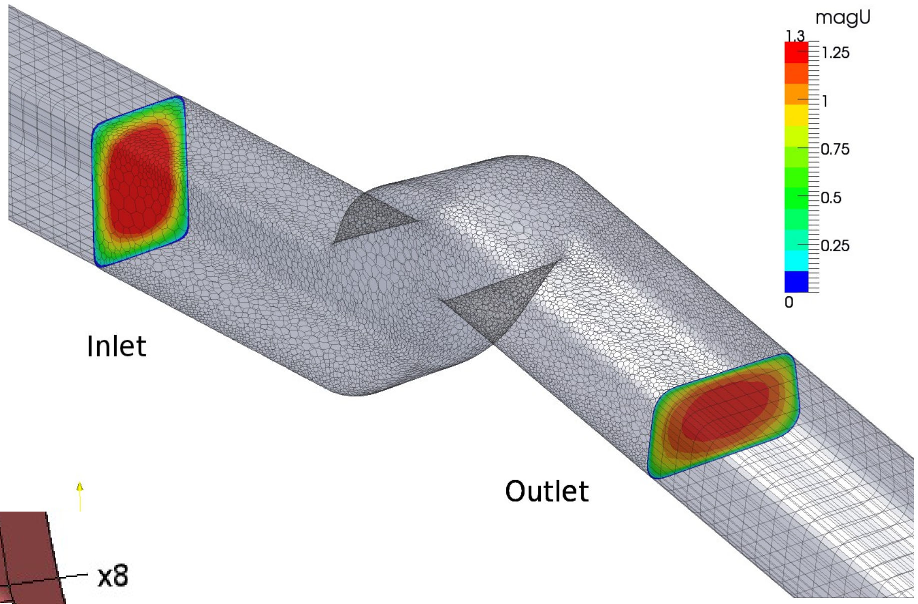


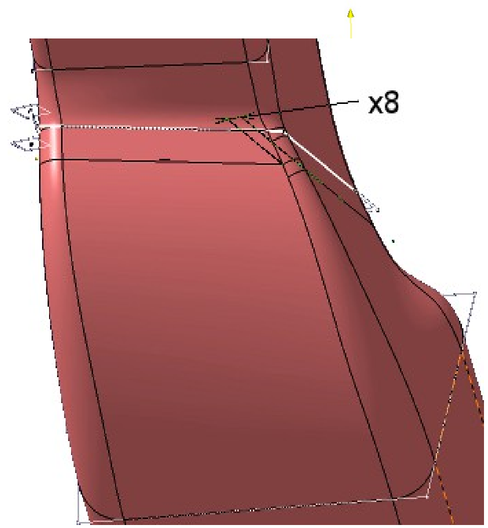
generations ~ wall clock time
if little T spread

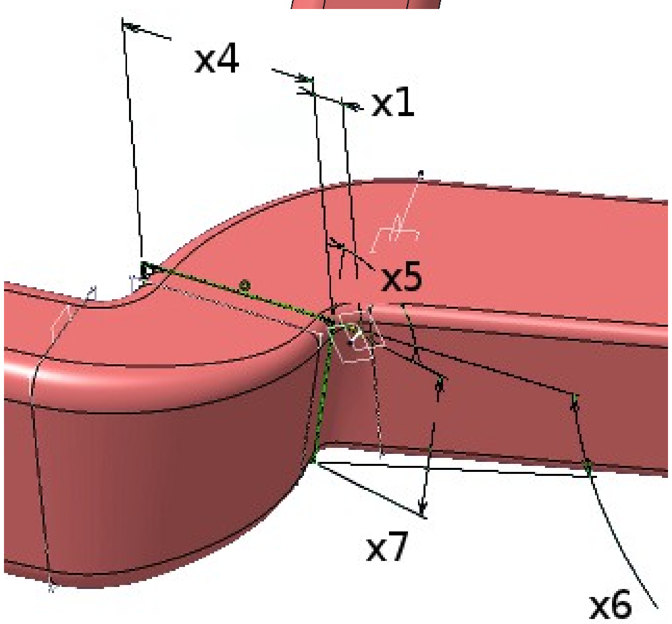
Results with $EI^{0,\lambda}$: air duct design (1)

(from ANR / OMD2 project)









$\boldsymbol{min_{x1,\ldots,x8}}$ pressure loss

viscous, turbulent, incompressible flow (Re=4000, ν=1.6 10^{-4} m^2/s)

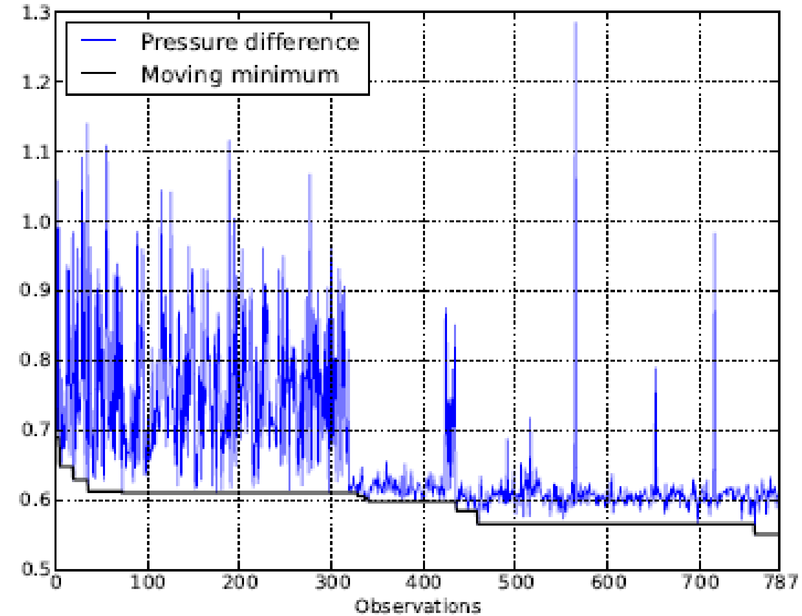
Complete simulation workflow :
Catia (CAO) + StarCCM (mesh) + Openfoam (FE analysis), from 15 to 40 min.
Run on PACA Grid.

Results with $EI^{0,\lambda}$: air duct design (2)

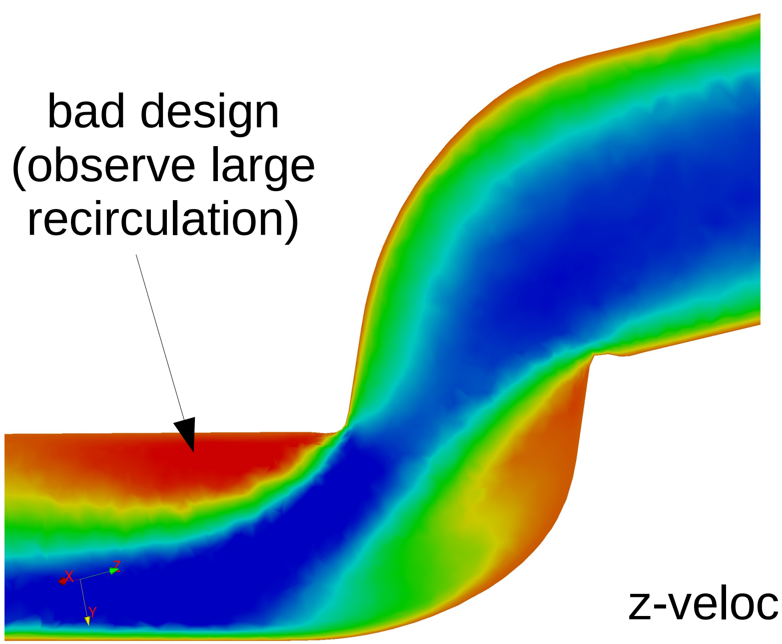
λ=4
320 LHS points in initial DOE

Simulation crashes : from 15 % at the beginning to 60 % at the end.

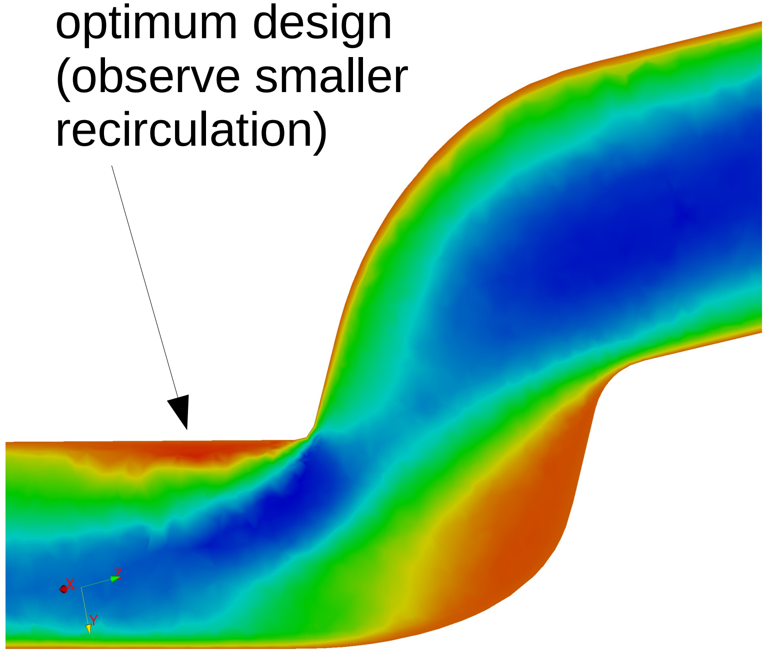
← multi-points EI are more robust to simulation crashes than single point EI.



bad design (observe large recirculation)



optimum design (observe smaller recirculation)



z-velocity maps

Limitations of $EI^{0,\mu}$

The number of nodes that can be used is limited by the problem to be solved

$$\max_{\boldsymbol{x} \in \mathbb{R}^{\lambda \times n}} EI^{0,\lambda}(\boldsymbol{x})$$

which is in dimension $\lambda \times n$.

The computing nodes have different speeds and the simulations different durations.

Time model :

λ nodes

T : time for 1 simulation, random variable , $T \sim U[t_{min}, t_{max}]$

$t_O =$ time for 1 optimization

T_{WC} : wall clock time for 1 generation

$$T_{WC} = t_O + \max_{i=1,\lambda}(t^i) \quad , \quad E(T_{WC}) = t_O + \frac{\lambda}{\lambda+1}(t_{max} - t_{min}) + t_{min} \sim O(t_O + t_{max})$$

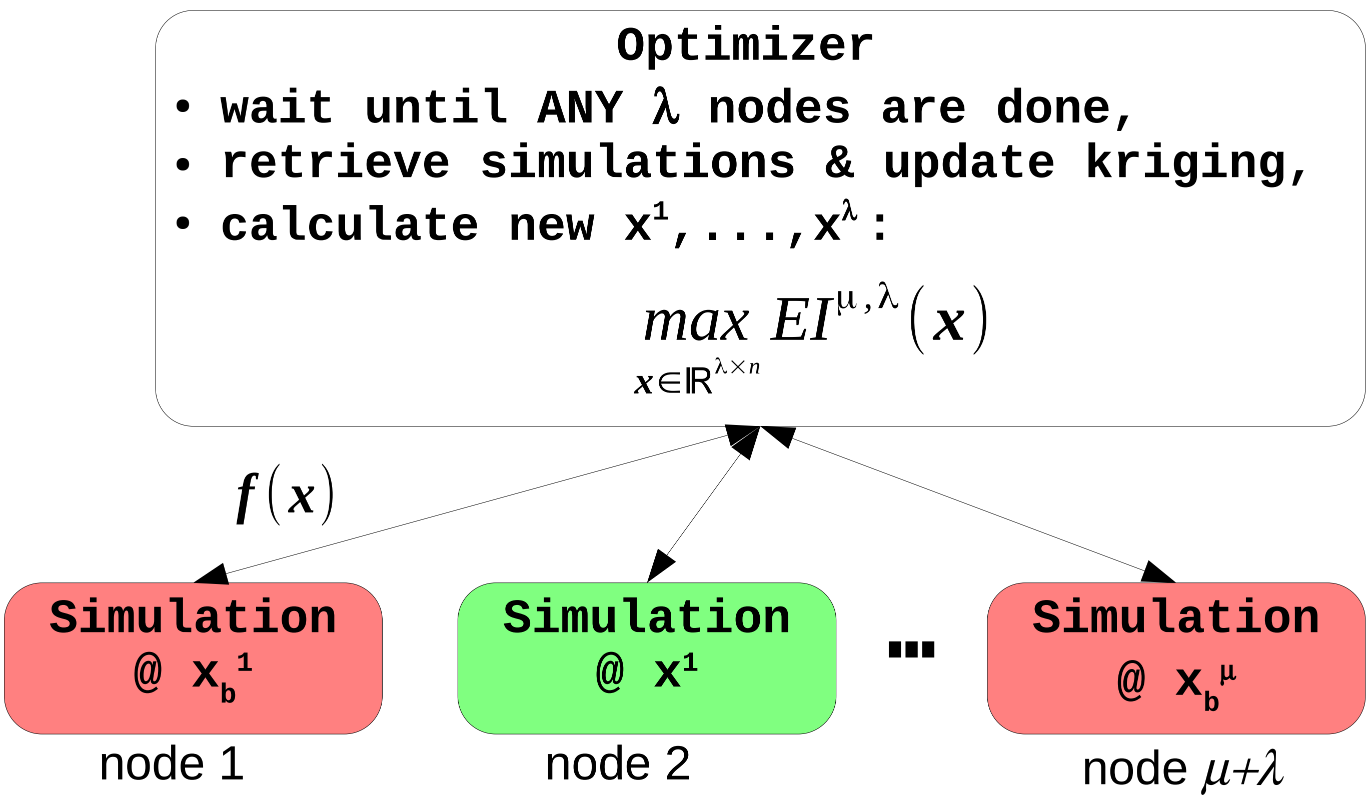
Outline of the talk

1. Introduction to kriging and optimization
2. Synchronous parallel EI
3. Asynchronous parallel EI
4. Embarassingly parallel EI algorithms
5. An agent-based dynamic partitioning algorithm

Asynchronous parallel EI : flow chart

- It allows to use $m=\lambda+\mu$ nodes (actually ok for any optimizer that is not sensitive to the order of return of the points).

- $EI^{\mu,\lambda}$ takes full account of past and on-going simulations.



Asynchronous parallel EI : criterion

λ nodes are available for new simulations at $x^1,\ldots,x^\lambda \ (\equiv \boldsymbol{x})$
μ nodes are busy running simulations at $x_b^1,\ldots,x_b^\mu \ (\equiv \boldsymbol{x_b})$

$$EI^{\mu,\lambda}(\boldsymbol{x}) = E\left[min\left(f_{\text{min}}, F(\boldsymbol{x_b})\right) - min\left(F(\boldsymbol{x})\right)\right]^+ \ | \ F(x^{1\ldots m}) = f(x^{1\ldots m})$$

Recall the 1 point sequential EI and the synchronous EI :

$$EI(x) \equiv EI^{0,1}(x) = E\left[f_{\text{min}} - F(x)\right]^+ \ | \ F(x^{1\ldots m}) = f(x^{1\ldots m})$$

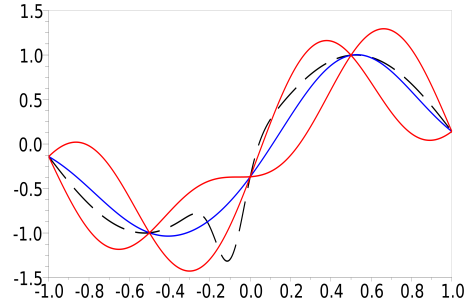
$$EI^{0,\lambda}(\boldsymbol{x}) = E\left[f_{\text{min}} - min\left(F(\boldsymbol{x})\right)\right]^+ \ | \ F(x^{1\ldots m}) = f(x^{1\ldots m})$$

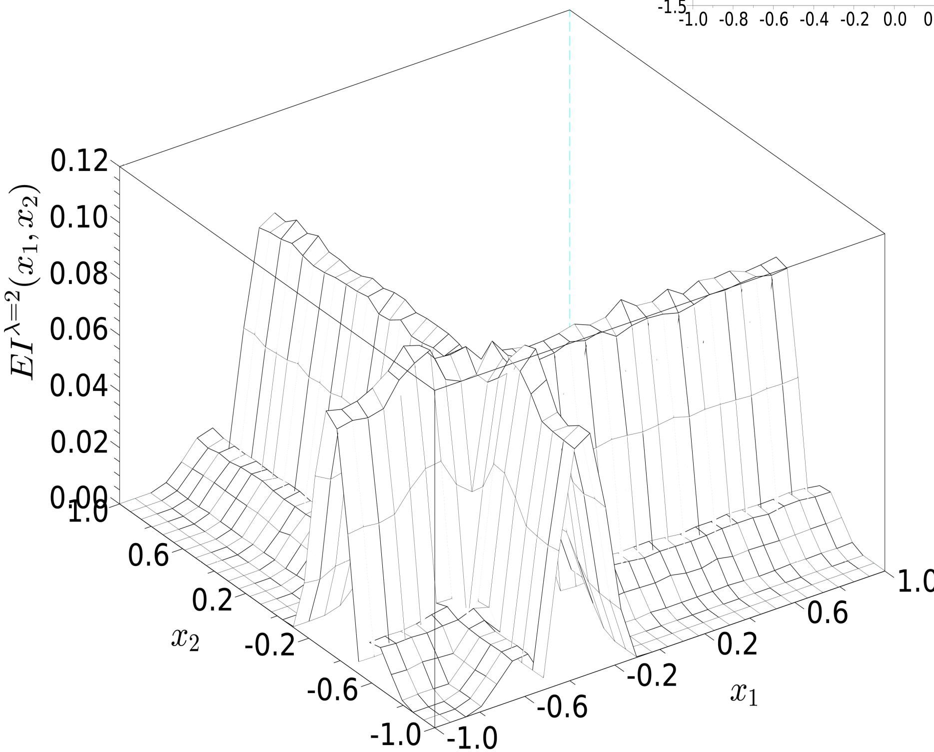
Property : $EI^{\mu,\lambda}(\boldsymbol{x}) \rightarrow 0^+$ as $\boldsymbol{x} \rightarrow \boldsymbol{x_b}$
(the search is pushed away from already sampled points which are being evaluated)

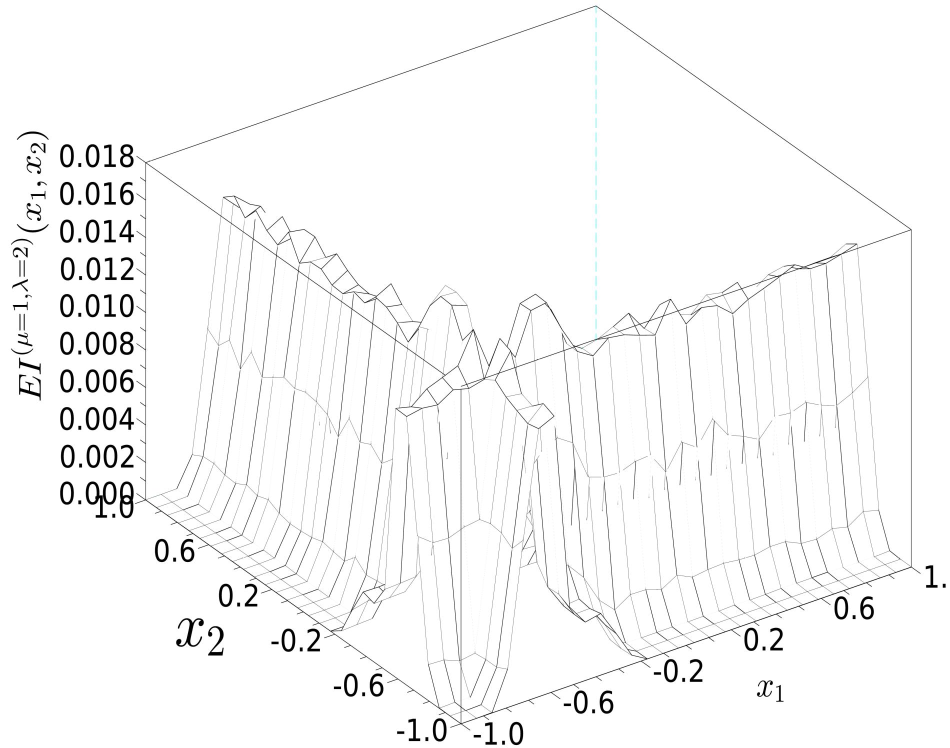
Numerical estimation of $EI^{\mu,\lambda}$

Expl : 1D function (black dotted), $EI^{1,2}(x_1, x_2)$, $x_b = -0.34$

10000 Monte Carlo simulations

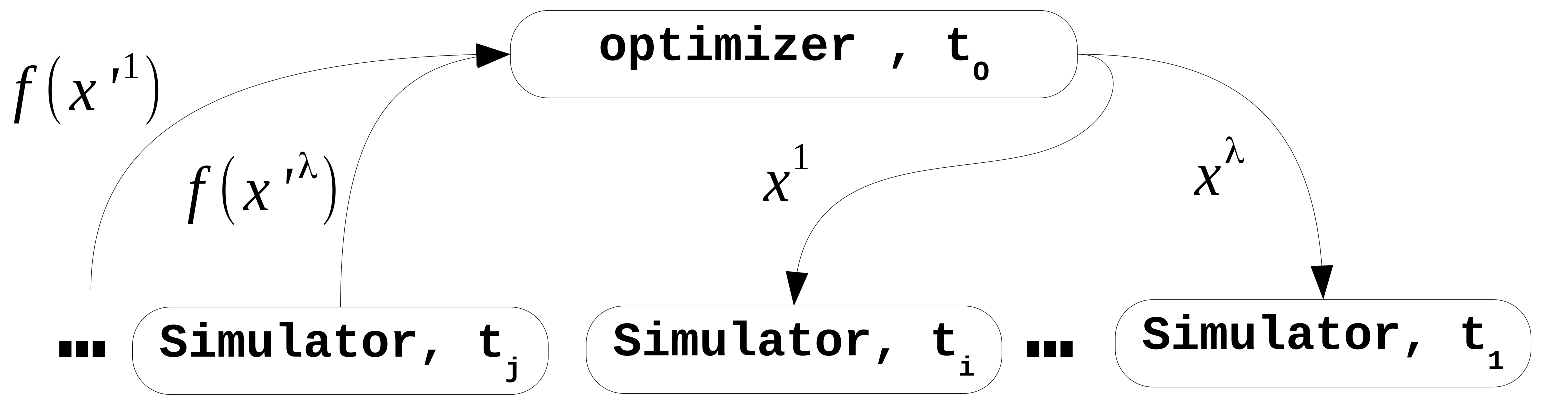






Time model of $EI^{\mu,\lambda}$ (1)

If time simulations $\gg$ time optimizer, use $EI^{\mu,1}$,
if time simulations $\ll$ time optimizer, use $EI^{0,\lambda}$,
otherwise, use $EI^{\mu,\lambda}$ for task allocation.



$M=\lambda+\mu$ nodes
T : time for 1 simulation, random variable , $T\sim U[t_{min}, t_{max}]$
t_O = time for 1 optimization

T_{WC} : wall clock time for 1 generation. Model :
$T_{WC} = t_O + t_{\lambda:M}$, then $t_{\lambda+1\ldots M:M} \leftarrow max[0 , t_{\lambda+1\ldots M:M} - T_{WC}]$

Time model of $EI^{\mu,\lambda}$ (2)

$M = \lambda + \mu$ nodes

T : time for 1 simulation, random variable, $T \sim U[t_{min}, t_{max}]$, $t_{min} = 10$

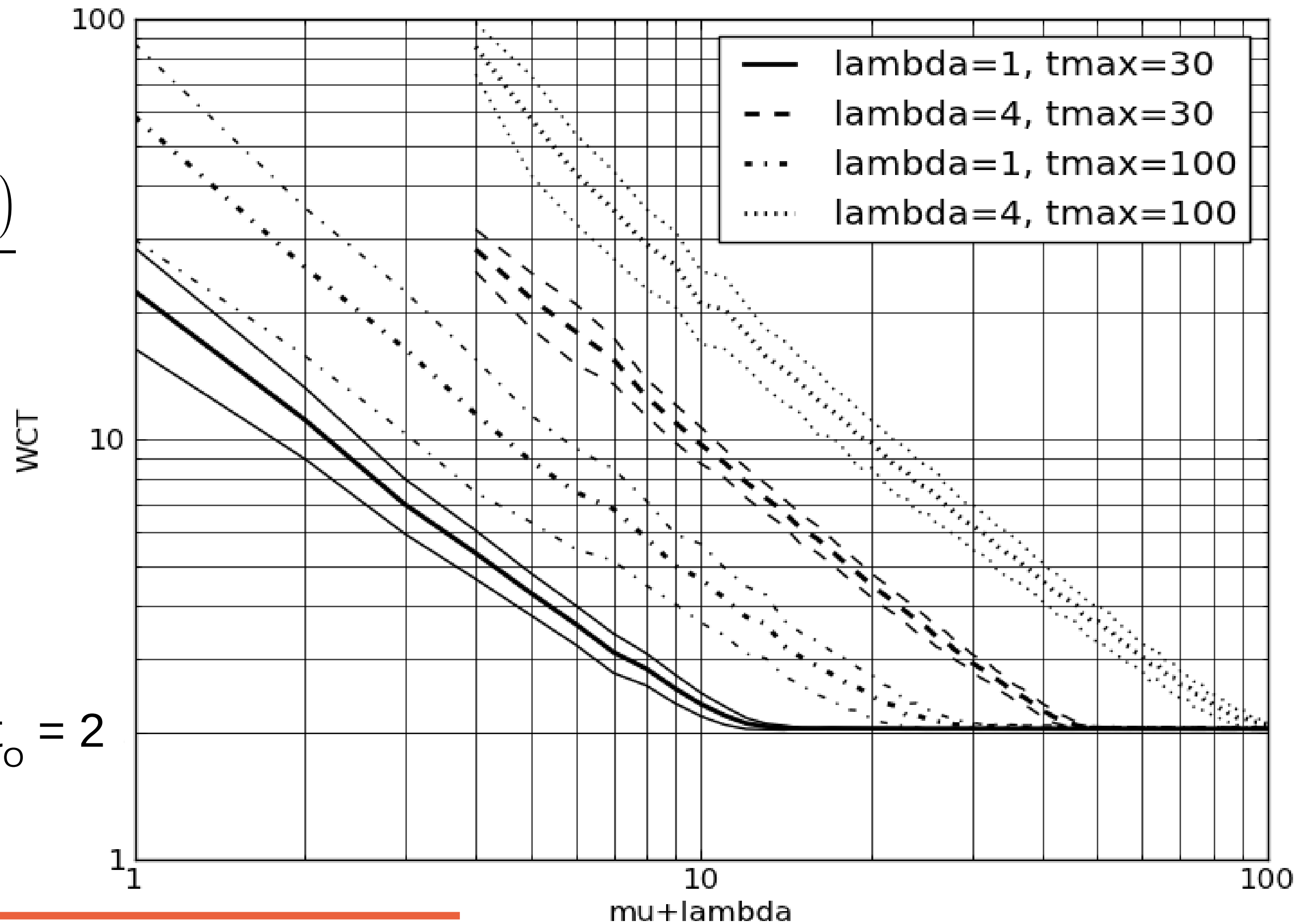
t_O = time for 1 optimization

T_{WC} : wall clock time for 1 generation. Model :

$T_{WC} = t_O + t_{\lambda:M}$, then $t_{\lambda+1\ldots M:M} \leftarrow max[0 \, , \, t_{\lambda+1\ldots M:M} - T_{WC}]$

$$E(T_{WC}) \approx t_O + \frac{E(T_{\lambda:M})}{M}$$

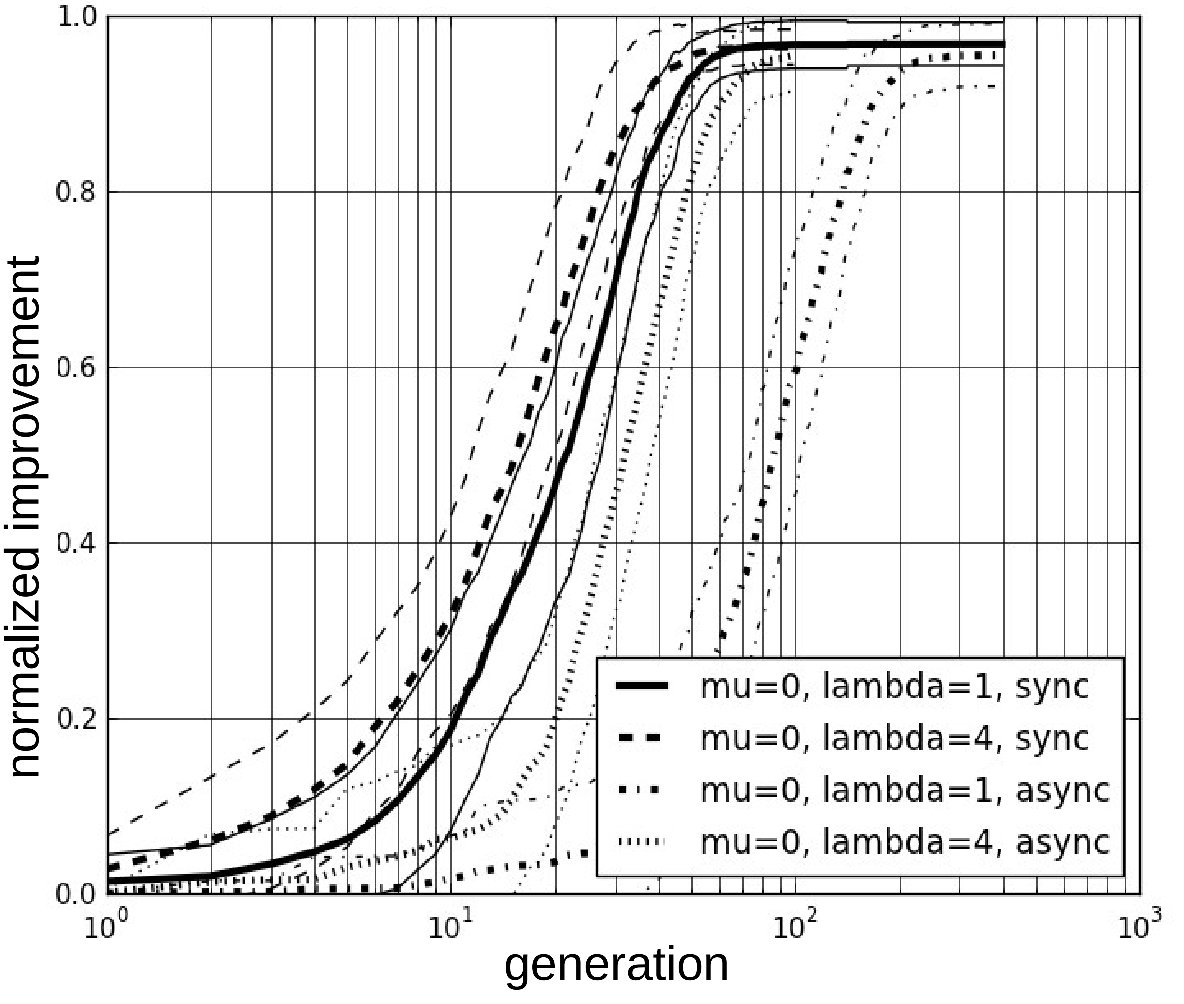
Scaling :
$E(T_{WC}) \underset{M\to\infty}{\to} t_O$



$t_O = 2$

Synchronous vs. asynchronous EI's (1)

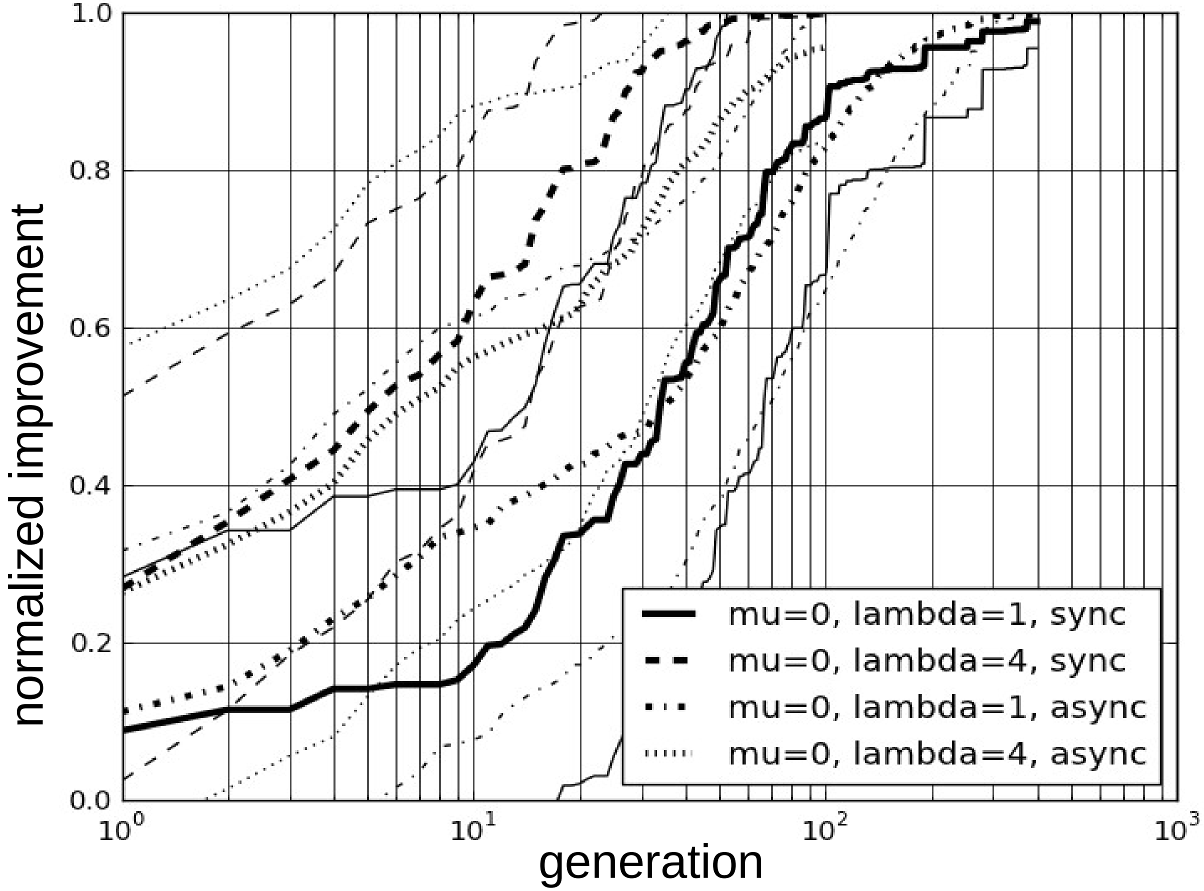
Ex : Rank1, 9D
(idem on Michalewicz
2D and Rosenbrock 6D)



Generation wise, asynchrony slows down the search because all demanded points are not evaluated.
But the wall clock time is much lower (× 0.093 and 0.13 for λ=1 and 4, M=32)

Synchronous vs. asynchronous EI's (2)

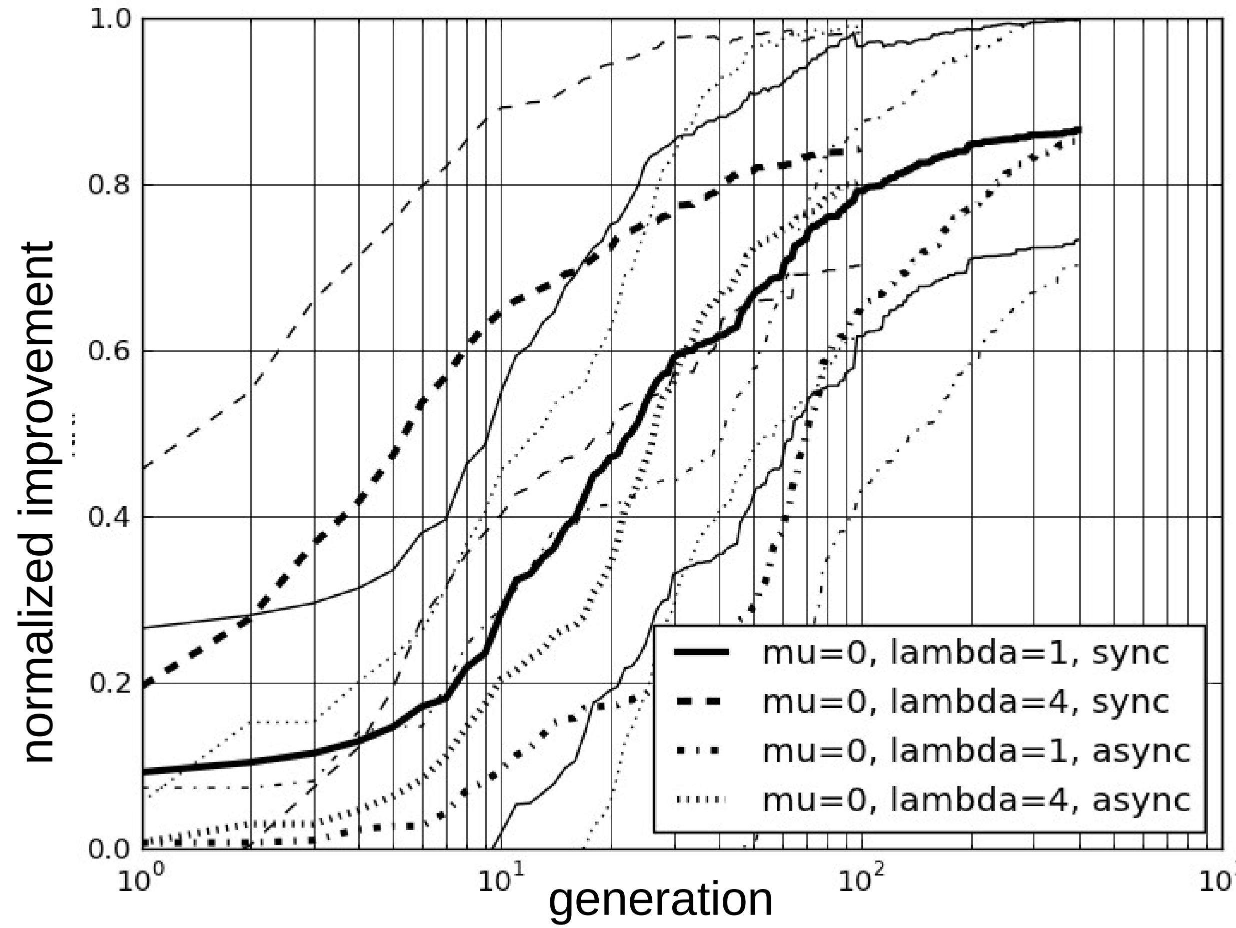
Ex : Michalewicz 2D



Generation wise, asynchrony slows down the search because all demanded points are not evaluated.
But the wall clock time is much lower (× 0.093 and 0.13 for λ=1 and 4, M=32)

Synchronous vs. asynchronous EI's (3)

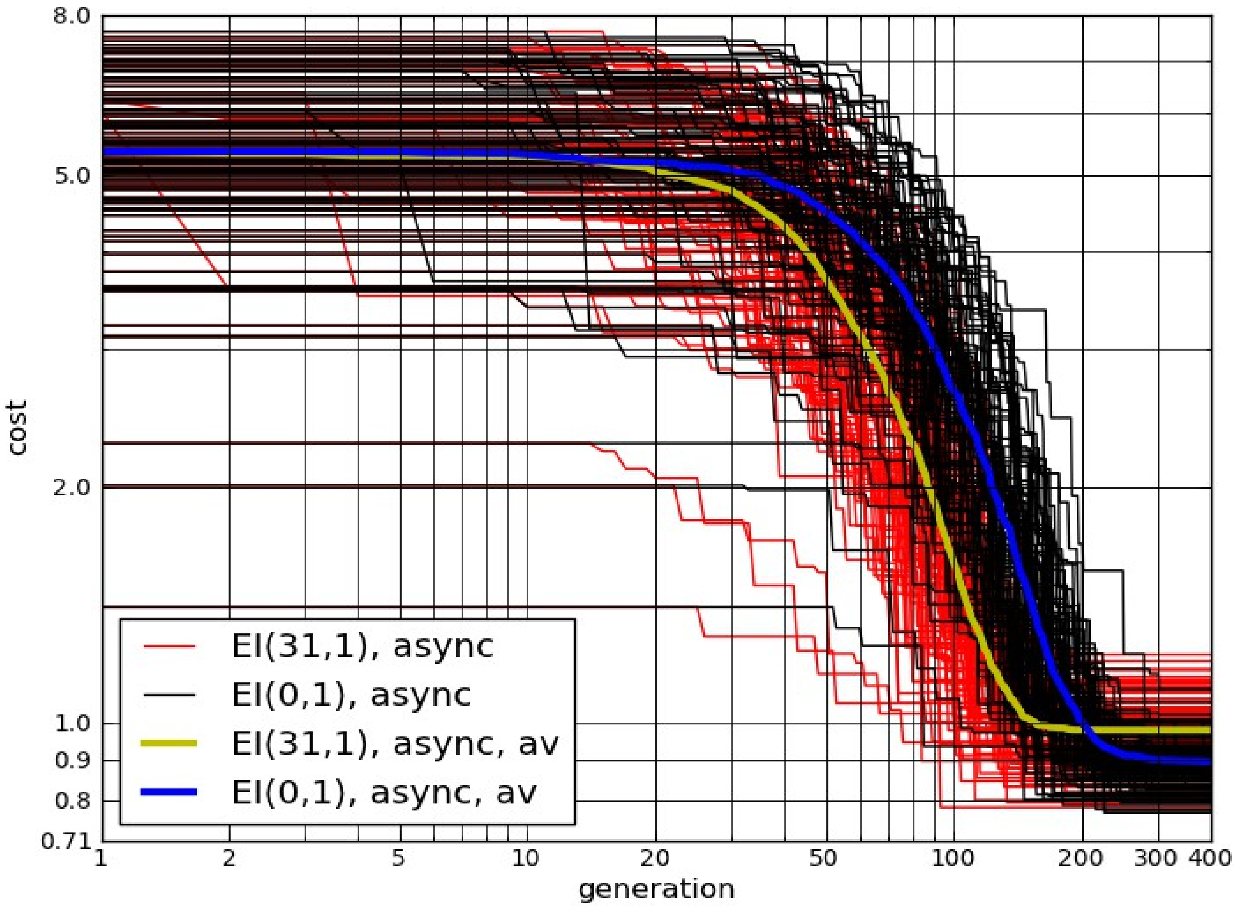
Ex : Rosenbrock 6D



Generation wise, asynchrony slows down the search because all demanded points are not evaluated.
But the wall clock time is much lower (× 0.093 and 0.13 for λ=1 and 4, M=32)

Effect of the μ busy points in $EI^{\mu,\lambda}$

100 runs, $EI^{0,1}$ asynchronous vs. $EI^{31,1}$ asynchronous, rank1 function in 9D



$EI^{31,1}$ is slightly faster than $EI^{0,1}$ because it avoids sending duplicates to the nodes for evaluation.

Partial conclusions

• We have presented an asynchronous parallel expected improvement algorithm for global optimization.

• Thanks to kriging and parallelization, it is adapted to computationally costly objective functions (and not adapted to high dimensions).

• It has a master-slave structure, with one optimizer only.

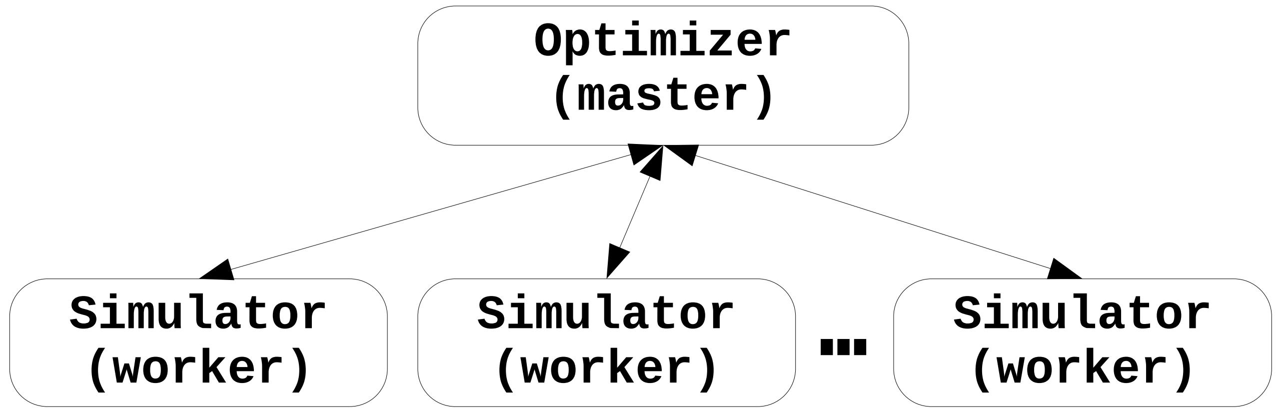
• Scaling : when the number of nodes increases the optimization time becomes the blocking factor.

• Solutions :
  • design fast mono-optimizers
  • **design algorithms with many optimizers.** ← **discussed next**

Algorithms with multiple optimizers

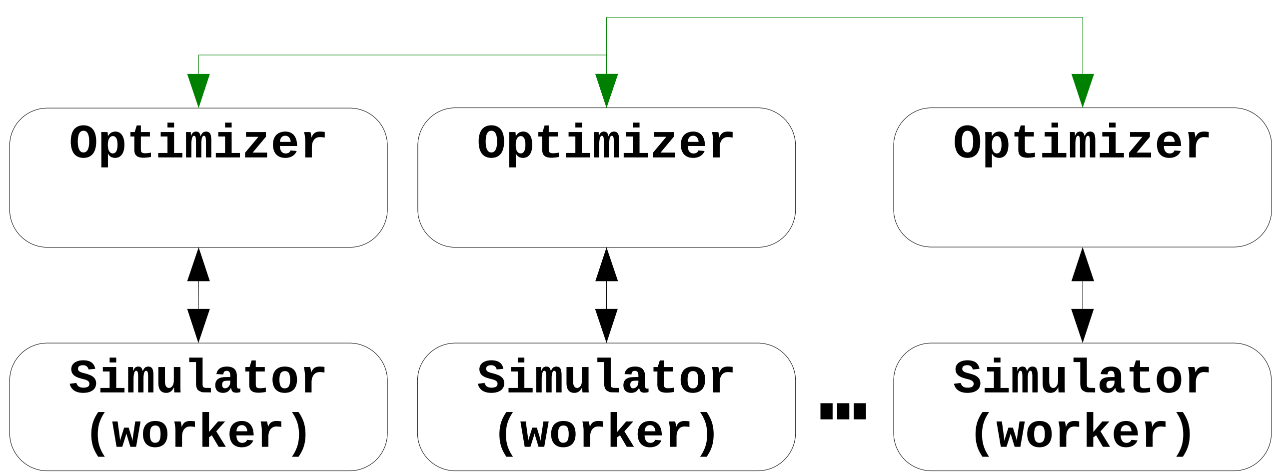
one optimizer

+ : decision with all information and ressources
- : t_o does not scale with M



Many optimizers
with coordination

+ : both optimizer and simulators scale with M
- : decision with partial information / limited ressources

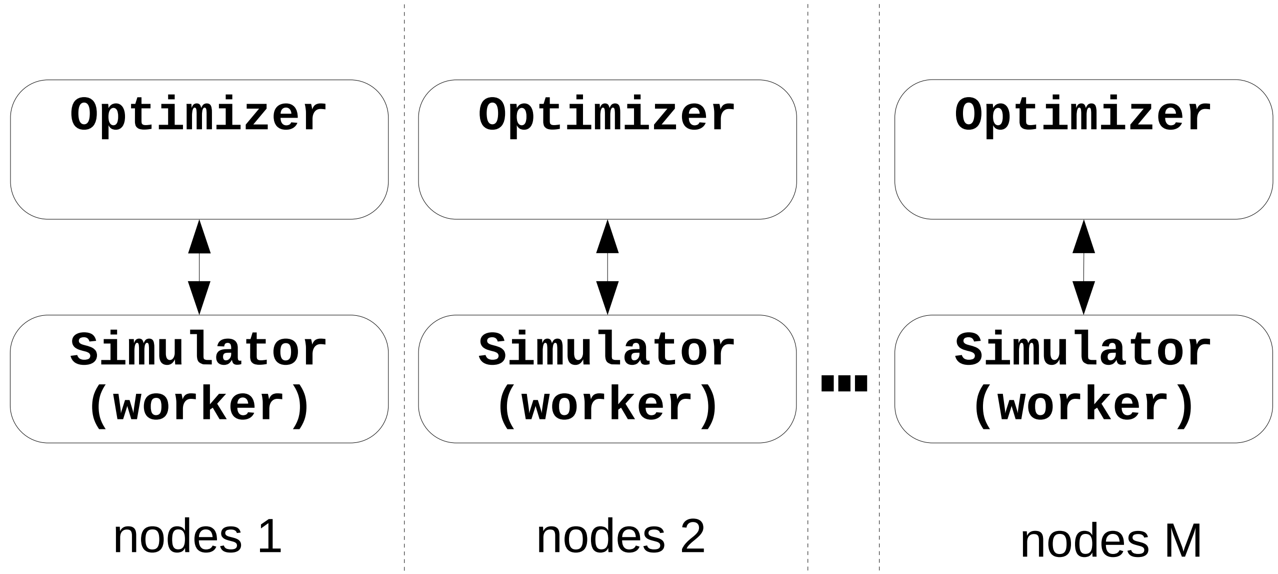


Outline of the talk

1. Introduction to kriging and optimization
2. Synchronous parallel EI
3. Asynchronous parallel EI
4. Embarrassingly parallel EI algorithms
5. An agent-based dynamic partitioning algorithm

Embarrassingly parallel EI algorithms

embarrassingly
=
no coordination

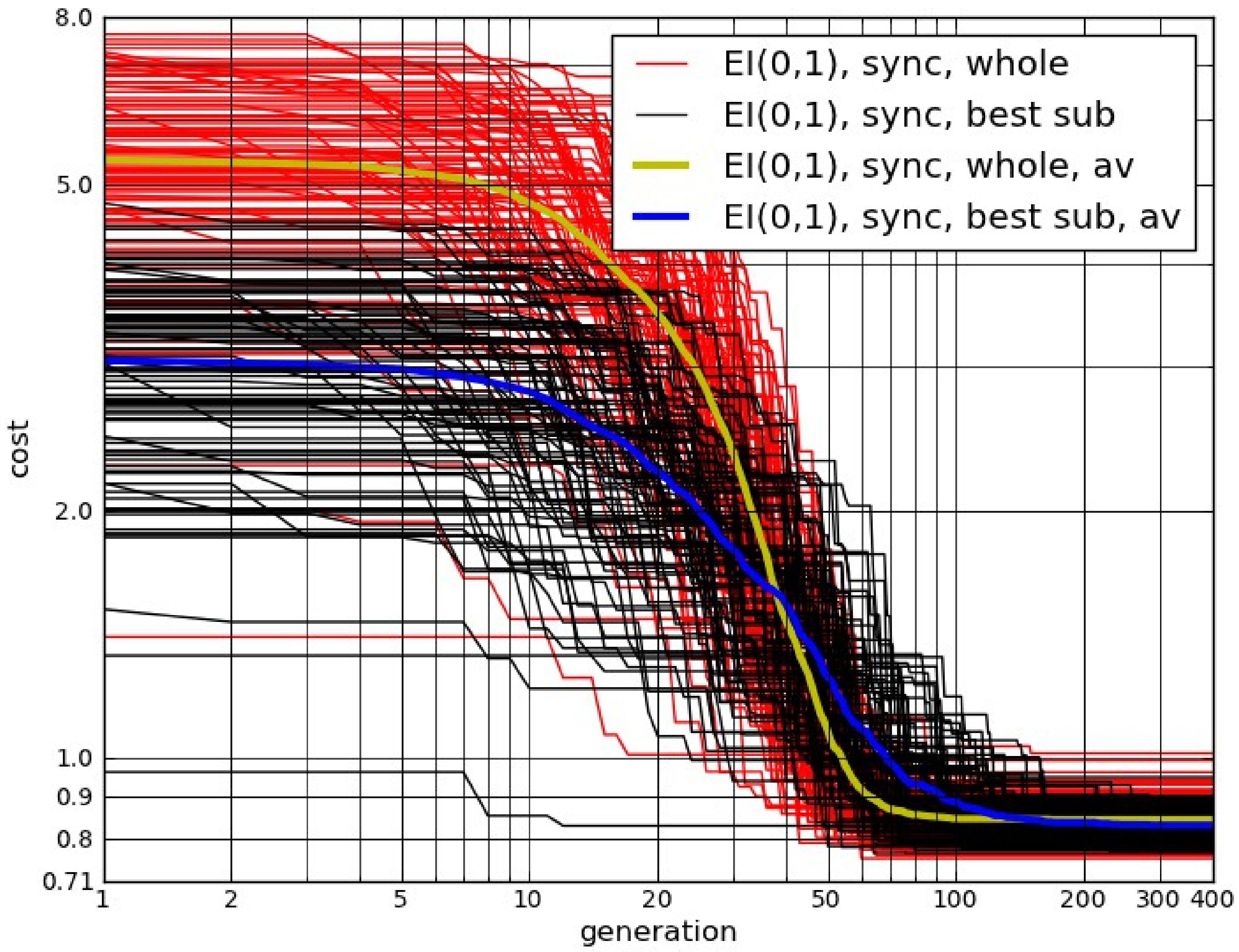


How to do it ?

• Change the initial DOE ← too costly (*size(DOE)* × *M* simulations to start).
• Divide the design space S into M fixed subdomains ← M-1 optimizations are useless, no observed gain.
• **Use M different covariance functions** ← interesting research direction.

Examples follow.

Fixed search space partition : example



9D Rank1 test function, 100 runs.

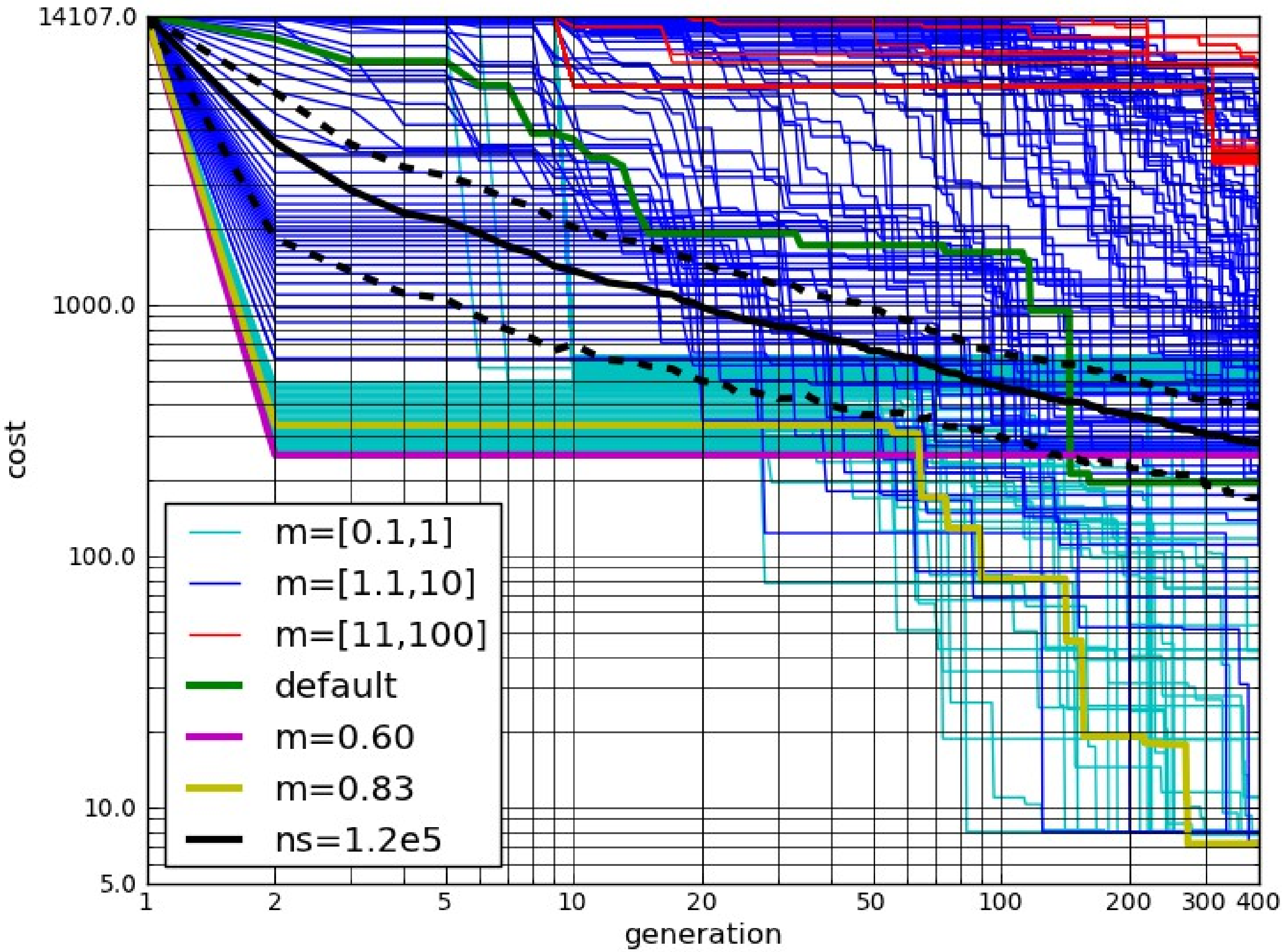
→ No gain observed after 40 iterations.

Different covariance functions : example

9D Rosenbrock test function

EI(0,1) = EGO
300 values of fixed covariance scale parameter (Gaussian kernel)

ns : random search for 300*400 trials.



→ Good covariance functions (e.g., m=0.83 here) yield very efficient optimizations.
Simple parallel implementation is a rough way to estimate them.

Outline of the talk

1. Introduction to kriging and optimization
2. Synchronous parallel EI
3. Asynchronous parallel EI
4. Embarrassingly parallel EI algorithms
5. An agent-based dynamic partitioning algorithm

Multiple optimizers with coordination : An agent-based dynamic partitioning algorithm

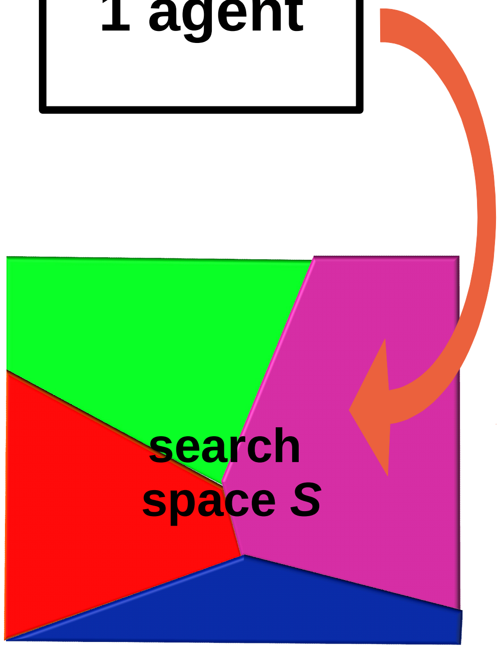
Villanueva, Le Riche, Picard and Haftka, *Dynamic partitioning for balancing exploration and exploitation in constrained optimization*, 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Sept. 17-19, 2012, Indianapolis, USA.

Multiple optimizers with coordination : An agent-based dynamic partitioning algorithm

1 subregion
+ 1 surrogate
+ 1 local constrained optimizer
+ 1 simulator

=

1 agent



Agents work in parallel to collectively solve the optimization problem :

$$\min_{x \in S \subset \mathbb{R}^n} f(x)$$
$$g(x) \leq 0$$

Agent coordination through :

- update of the partition
- agent creation
- agent deletion

(let's say 1 agent is affected to a set of computing nodes)

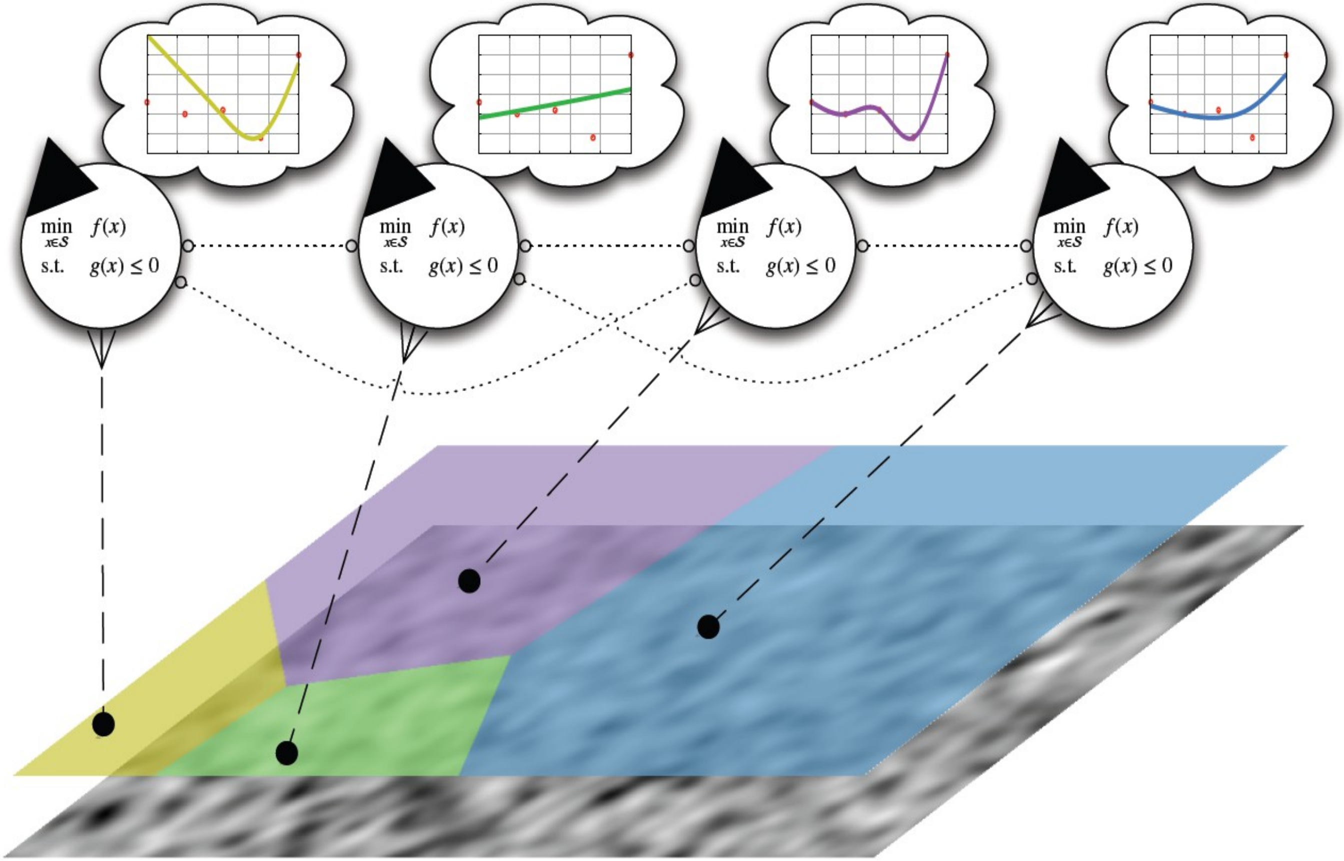
Agent-based dynamic partitioning algorithm
Goals

Solve a global optimization problem AND locate local optima
A method that can be used for expensive problems (thanks to the surrogates)

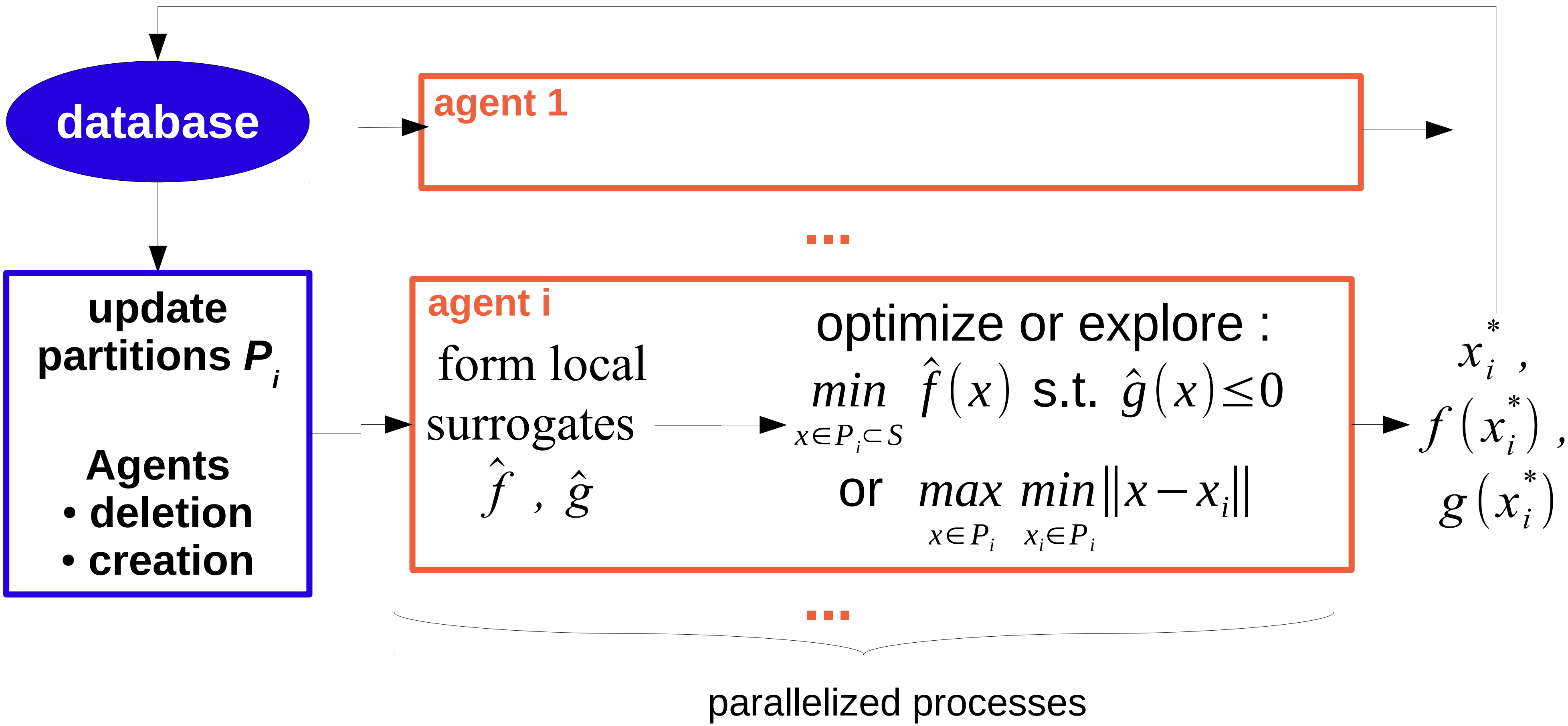
The search space partitioning allows :

1) to share the effort of finding local optima

2) to have surrogates defined locally (better for non stationary problems).



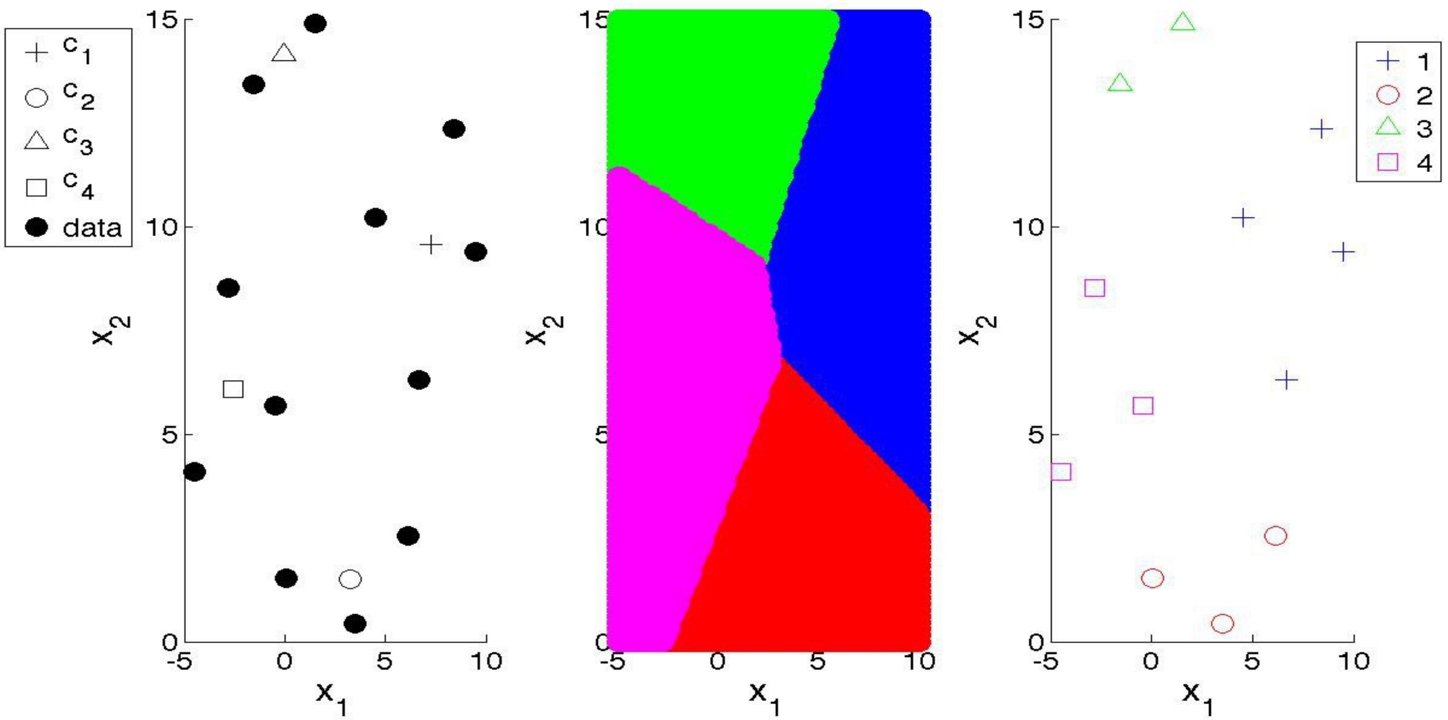
Agent-based dynamic partitioning algorithm
Global flow chart



optimize : SQP.
surrogates : polynomial response surface (orders 1, 2 and 3), kriging (linear or quad. trend), chosen based on cross-validation error.

Subregion definition

Subregions P_i are essentially defined by the centers c_i of the subregions : P_i is the set of points closer to c_i than to other centers. P_i are Voronoi cells.



Dynamic partitioning

The partitioning is updated by moving the centers to the best point in their subregion :

current = current center
new = point added to P_i at the last iteration and not on boundary of P_i

```
if current is infeasible then
   if new is less infeasible then move to new
elseif current is feasible then
   if new is feasible & has better f then move
to new
end
```

Property : agents will stabilize at local optima.

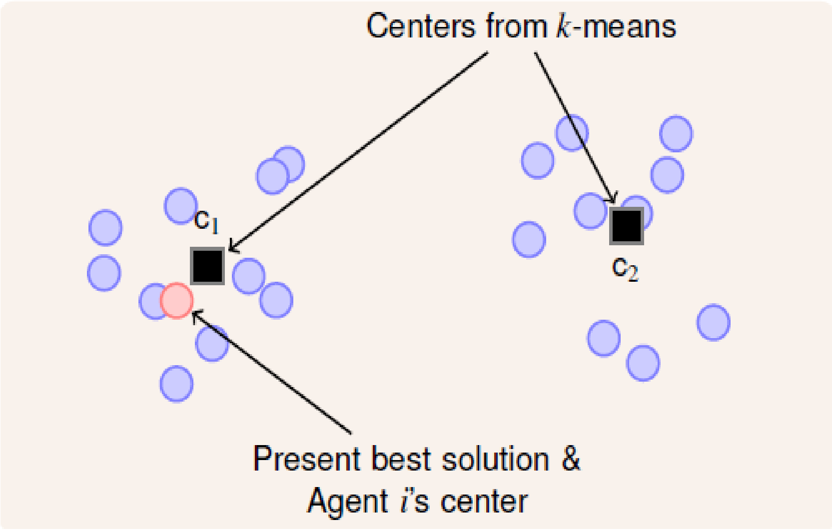
Agent deletion and creation

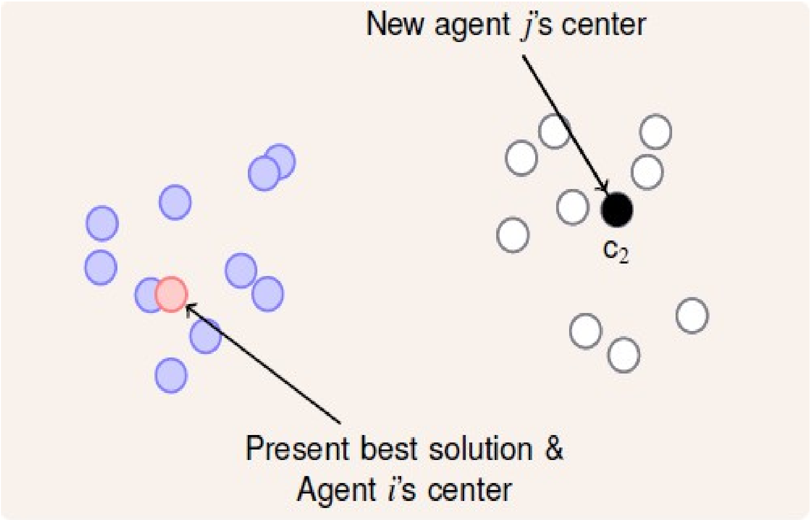
Deletion

If two agent centers are getting too close to each other, delete the worst.

Creation

Principle : the existence of 2 clusters in a subregion is a sign of at least 2 basins of attraction → split the subregion by creating a new agent.
Implementation : K-means + check on inter vs. intra class inertia + move centers at data points.





2D example with disconnected feasible regions

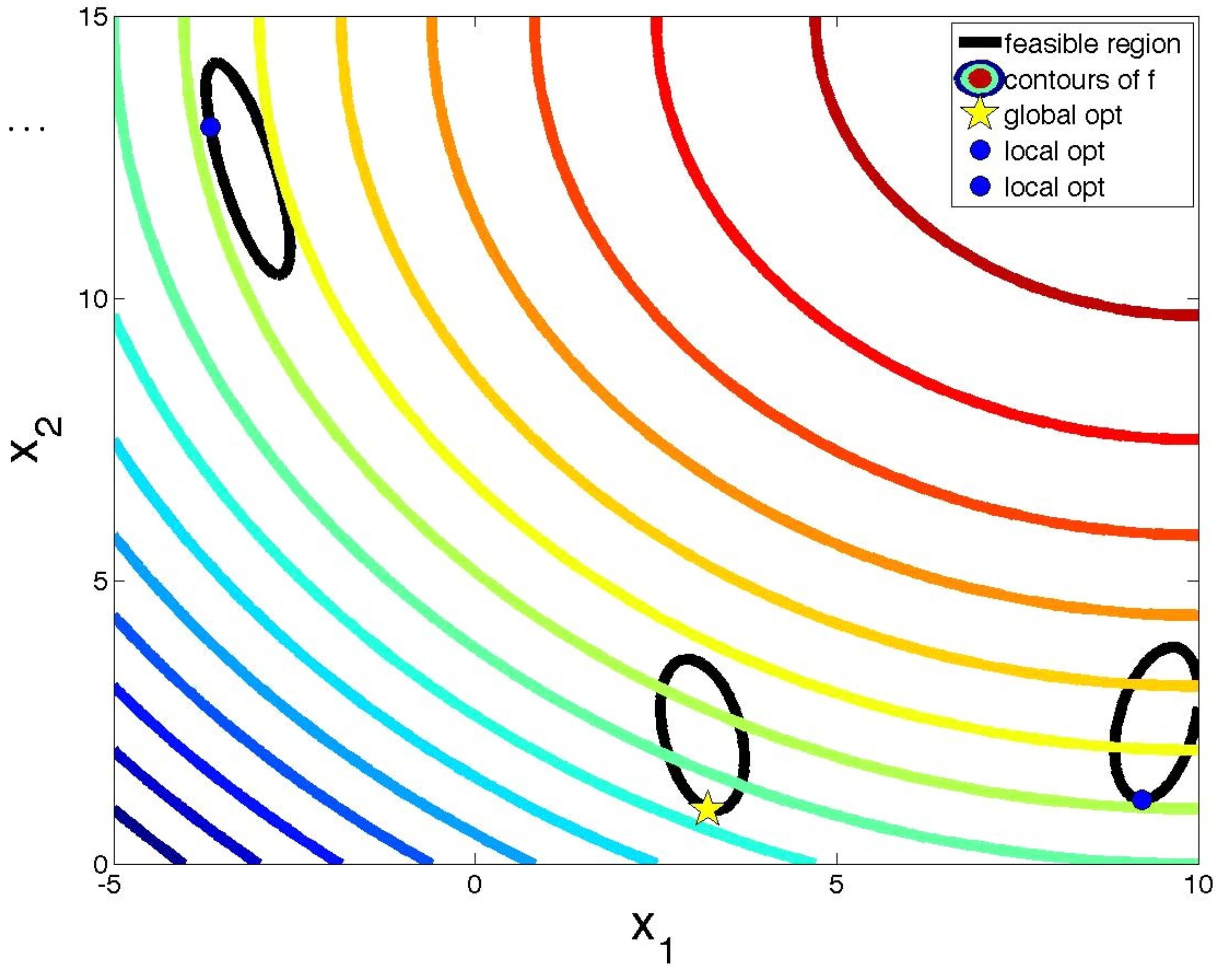
$$\underset{x}{\text{minimize}} \quad f(x) = -(x_1 - 10)^2 - (x_2 - 15)^2$$

$$\text{subject to} \quad g(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right) + \cdots$$

$$10\left(1 - \frac{1}{8\pi}\right) cos(x_1) + 10 - 2 \leq 0$$

$$-5 \leq x_1 \leq 10$$

$$0 \leq x_2 \leq 15$$

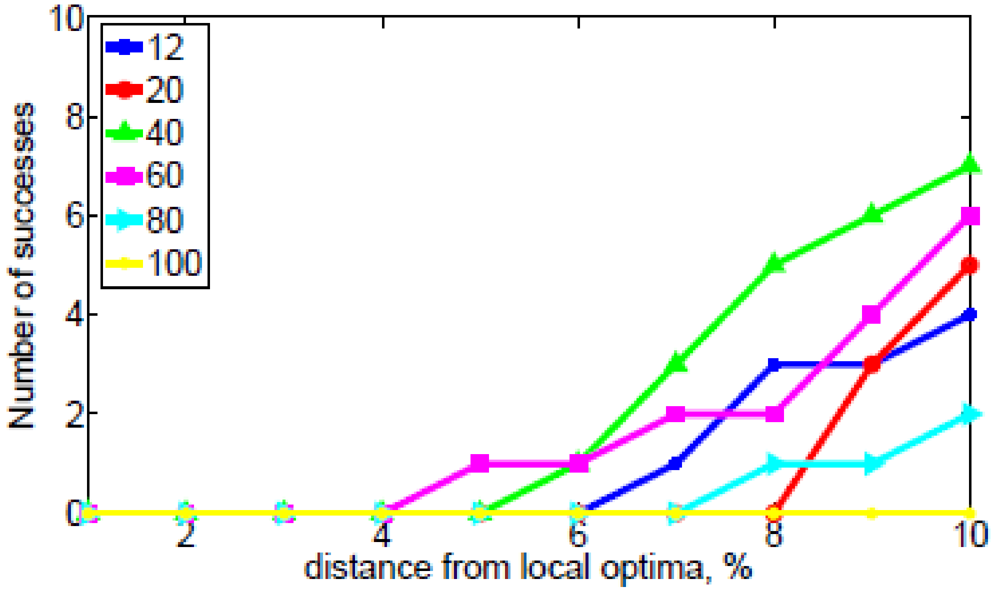


Both *f* and *g* are considered expensive → approximated with surrogates.

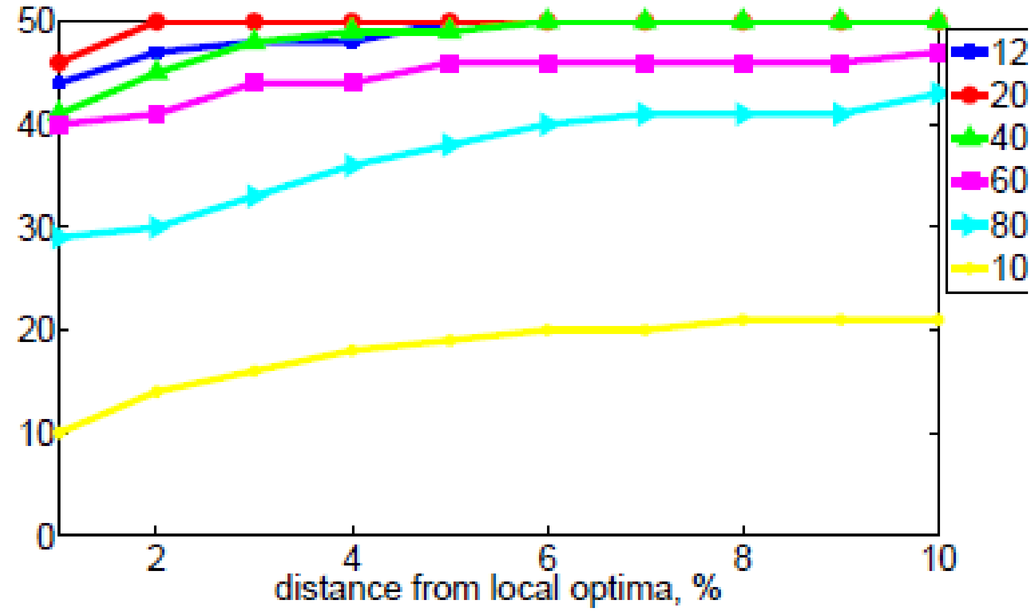
from Sasena, Papalembros, Goovaerts, Global optimization of problems with disconnected feasible regions via surrogate modeling, 9th AIAA/ISSMO symposium on Multidisciplinary Analysis and Optimization, AIAA-2002-5573.

Success at finding all local optima

1 agent



System of agents

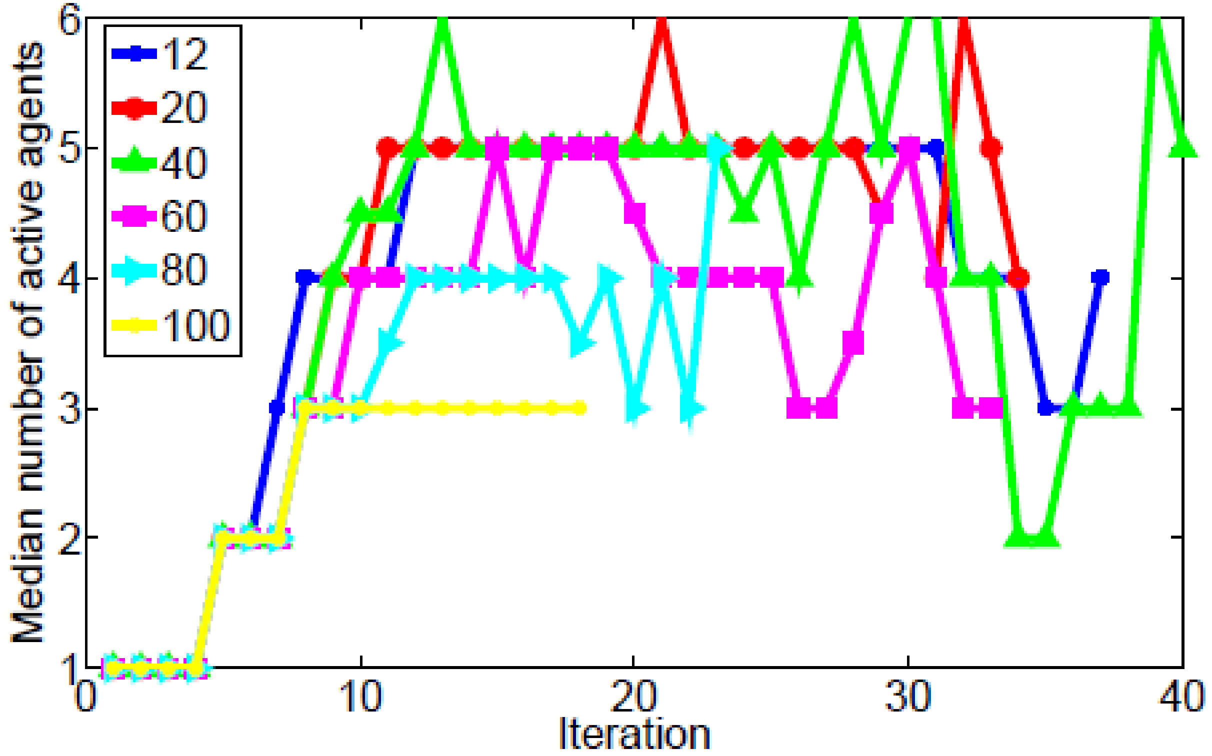


Different curves (colors) = size of initial DOE.
Fair comparisons : size of initial DOE + sum$_{\text{iterations}}$ nb. agents = 132 (constant).
Each curve is the median of 50 repetitions.

- **The partitioning is more efficient at finding all optima than repeated local searches + exploration.**

- **The algorithm benefits from small initial DOEs.**

Agent dynamics : number of agents

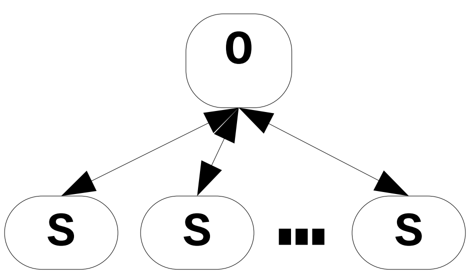


The median number of agents is between 3 and 5.

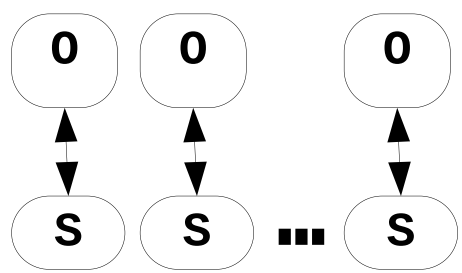
Note : there are 3 local optima.

Concluding remarks

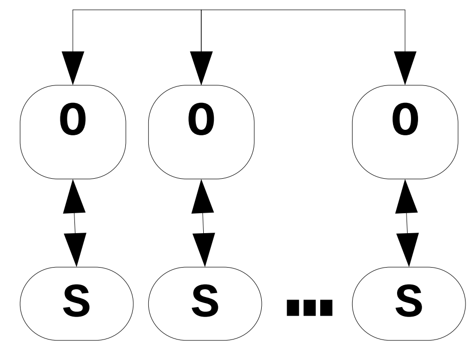
We have shown examples of parallelized global optimization algorithms that are adapted to expensive functions because they use surrogates. They follow the three patterns :



master-slave
$EI^{\mu,\lambda}$



embarrassingly //
different surrogates
(cov. functions) for EI



// + coordination
dynamic partitioning
of the search space

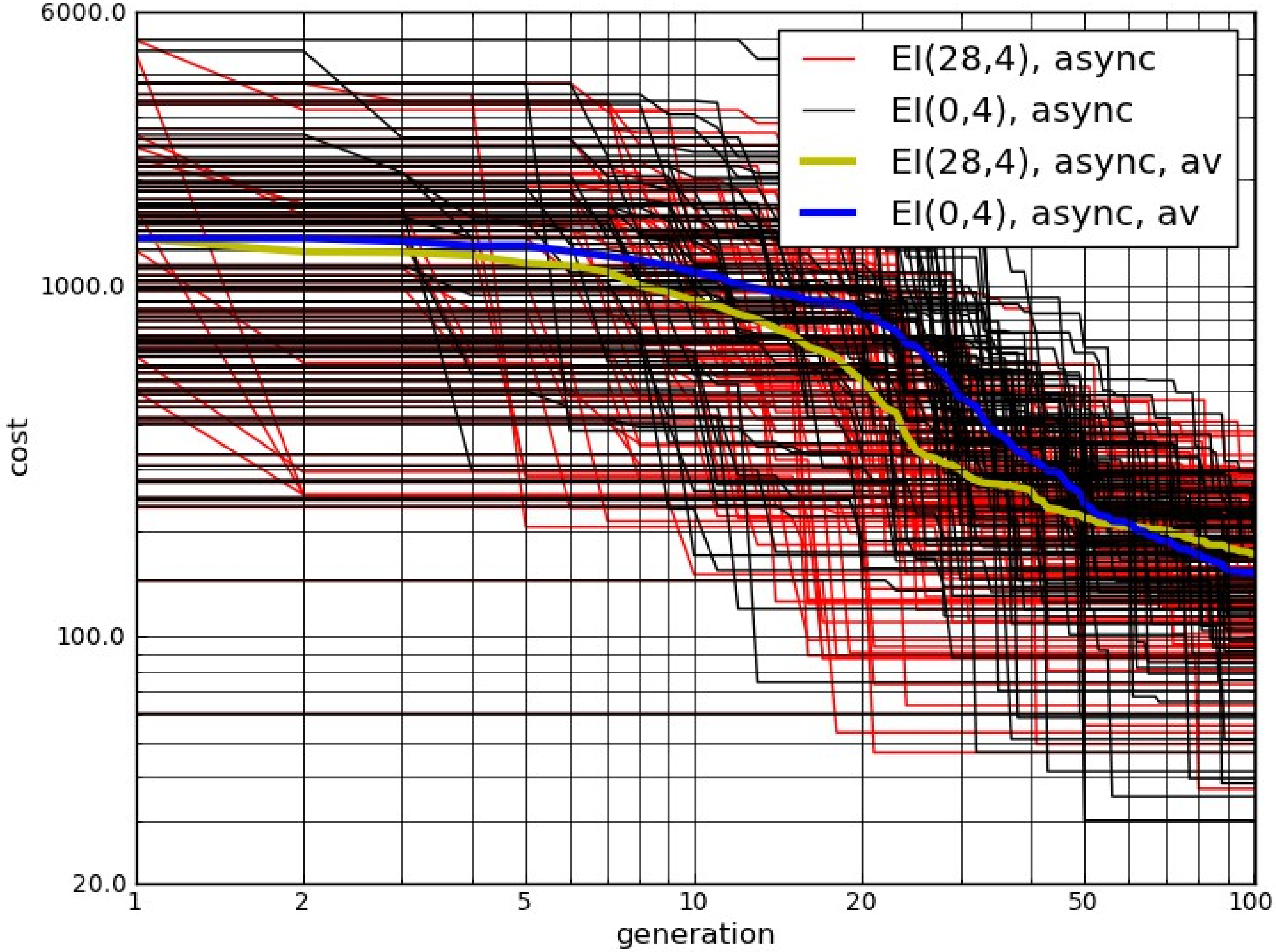
Perspective : better understand what strategy is the best considering

- a function landscape,
- a computational cost / budget,
- a computing infrastructure.

Additional slides

Effect of the μ busy points in $EI^{\mu,\lambda}$

100 runs, $EI^{0,4}$ asynchronous vs. $EI^{28,4}$ asynchronous, Rosenbrock function in 6D



W.r.t. $EI^{0,4}$, $EI^{28,4}$ better avoids sending duplicates to the nodes for evaluation.