

Hard optimization (multimodal, adversarial, noisy)

<http://www.lri.fr/~teytaud/hardopt.odp>

<http://www.lri.fr/~teytaud/hardopt.pdf>

(or Quentin's web page)

Acknowledgments: koalas, dinosaurs, mathematicians.

Olivier Teytaud

Knowledge also from A. Auger, M. Schoenauer.



***The next slide is the most important
of all.***

Olivier Teytaud

Inria Tao, en visite dans la belle ville de Liège



**In case of trouble,
Interrupt me.**

Olivier Teytaud

Inria Tao, en visite dans la belle ville de Liège



**In case of trouble,
Interrupt me.**

Further discussion needed:

- R82A, Montefiore institute
- olivier.teytaud@inria.fr
- or after the lessons (the 25th, not the 18th)

Olivier Teytaud

Inria Tao, en visite dans la belle ville de Liège

Rather than complex algorithms,
we'll see some concepts and some simple
building blocks for defining algorithms.

Warm starts

Coevolution

Niching

Clearing

Multimodal optimization

Restarts (sequential / parallel)

Koalas and Eucalyptus

Dinosaurs (and other mass extinction)

Fictitious Play

EXP3

Robust optimization

Noisy optimization

Nash equilibria

Surrogate models

Maximum uncertainty

Monte-Carlo estimate

Quasi-Monte-Carlo

Van Der Corput

Halton

Scrambling

EGO

I. Multimodal optimization

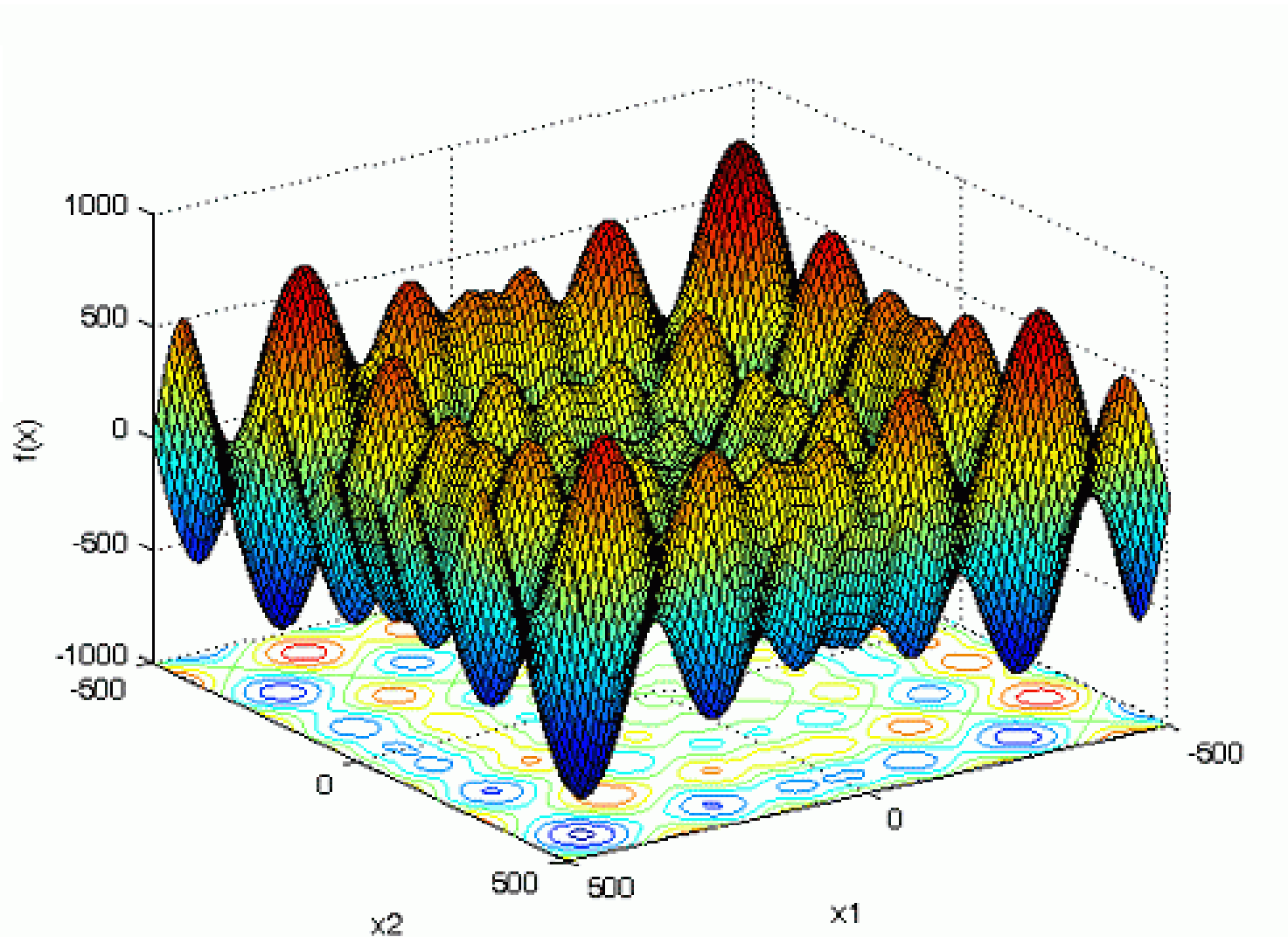
II. Adversarial / robust cases

III. Noisy cases

e.g. Schwefel's function

$$f(\vec{x}) = \sum_{j=1}^d x_j \sin(\sqrt{|x_j|})$$

$$x_j \in [-500, 500],$$
$$\vec{x}^* = (s, s, \dots, s), s = 420.9687,$$
$$f(\vec{x}^*) = -418.9829 d$$



(thanks to
Irafm)

What is multimodal optimization ?

- finding one global optimum ?
(not getting stuck in a local minimum)
- finding all local minima ?
- finding all global minima ?

Classical methodologies:



1. Restarts

2. ``Real'' diversity mechanisms

Restarts:

While (time left >0)

{

run your favorite algorithm

}

(requires halting criterion + randomness – why ?)

Parallel restarts

Concurrently, p times:

{

run your favorite algorithm

}

==> still needs randomization

Parallel restarts

Concurrently, p times:

{

run your favorite algorithm

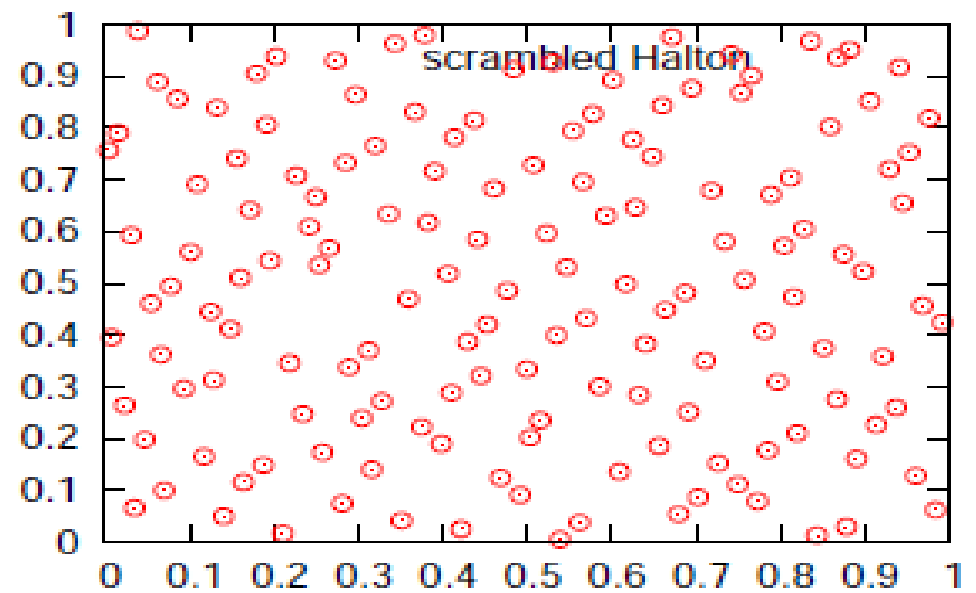
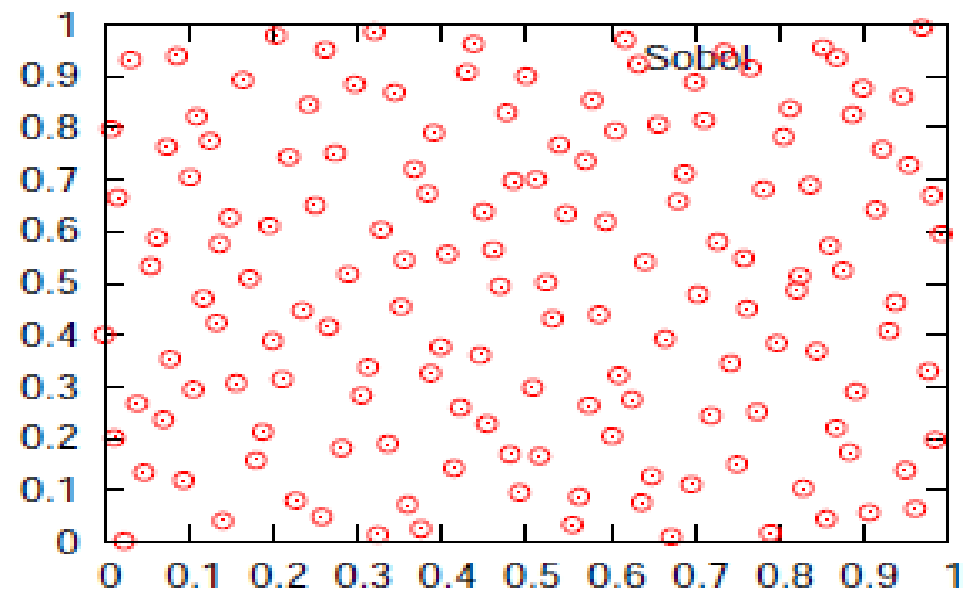
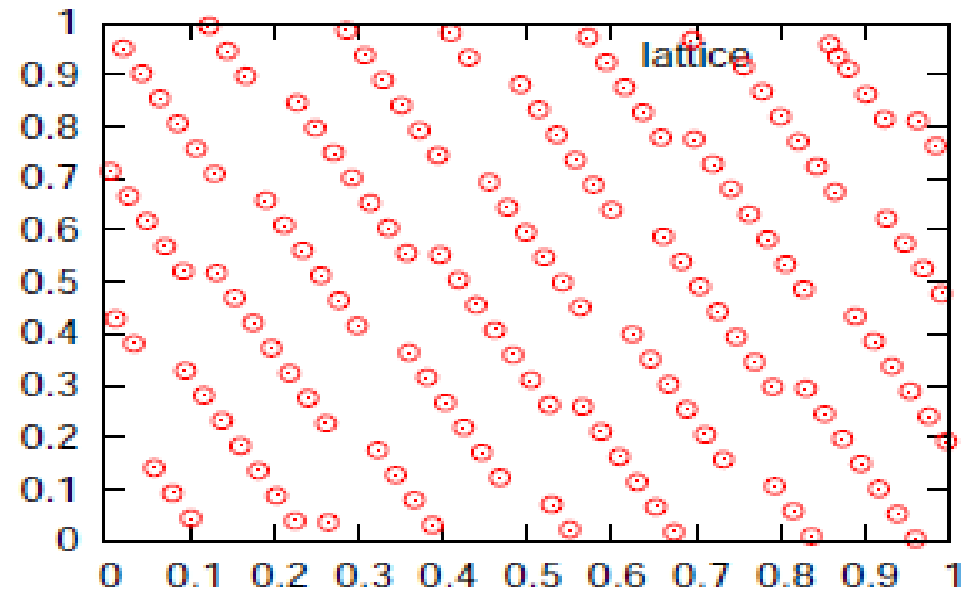
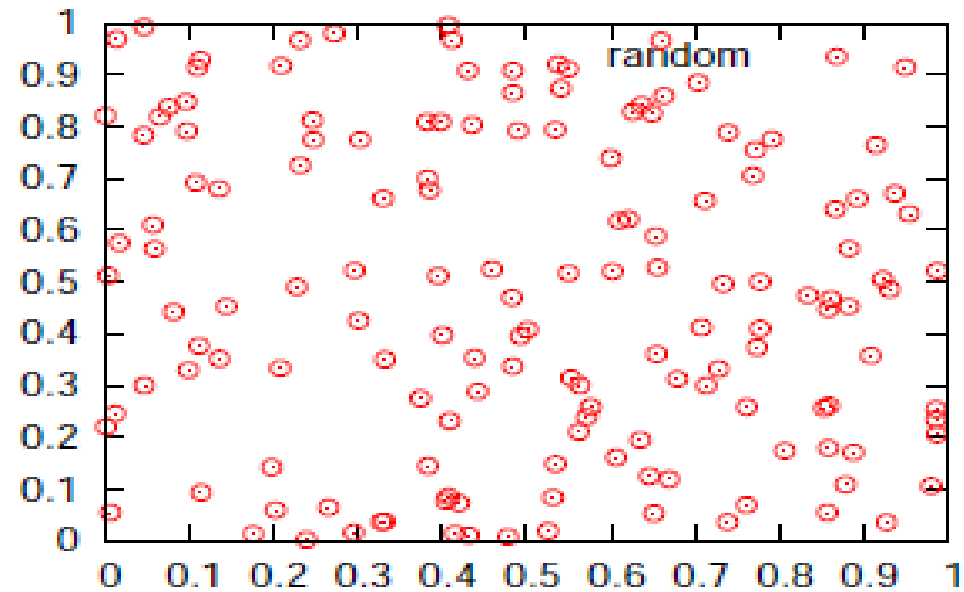
}

==> needs randomization

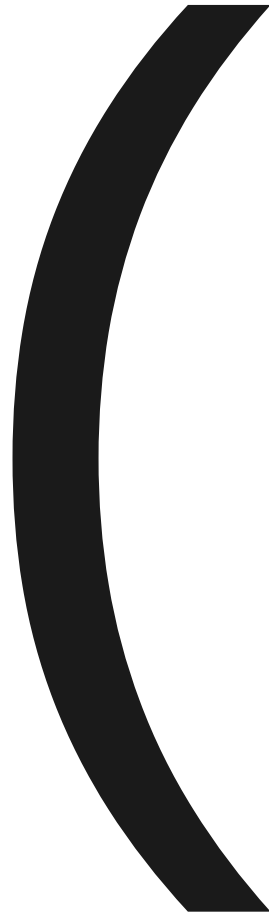
(at least in the initialization)

Random restarts

Quasi-random restarts



Let's open a parenthesis...



The Van Der Corput sequence:

quasi-random numbers in dimension 1.

How should I write the n^{th} point ?

(p =integer, parameter of the algorithm)

write n in basis p : $n = p_k p_{k-1} \dots p_1$,

i.e. $n = \sum_{i=1}^k p_i p^i$ with $p_i \in [[0, p - 1]]$

$x_{n,p} = 0.p_1 p_2 \dots p_k$ in basis p .

A (simple) Scrambled

Van Der Corput sequence:

$$x_{n,p} = 0.\pi(p_1)\pi(p_2)\dots\pi(p_k)$$

π is some permutation of $0, 1, \dots, p - 1$

$$\pi(0) = 0$$

Halton sequence:

= generalization to dimension d

Choose p_1, \dots, p_d

$$x_n = (x_{n,p_1}, x_{n,p_2}, \dots, x_{n,p_d}) \in [0, 1]^d$$

All p_i 's must be different (why?).

Typically, the first prime numbers (why?).

Scrambled Halton sequence:

= generalization to dimension d with scrambling

Choose p_1, \dots, p_d

$$x_n = (x_{n,p_1}, x_{n,p_2}, \dots, x_{n,p_d}) \in [0, 1]^d$$

All p_i 's must be different (why?).

Typically, the first prime numbers (why?).

Let's close the parenthesis...



(thank you Asterix)

Consider an ES (an evolution strategy = an evolutionary algorithm in continuous domains).

The restarts are parametrized by an initial state x and an initial step-size σ .

Which hypothesis on x and σ do we need, so that the restarts of the ES will find all local optima ?

The assumptions are:

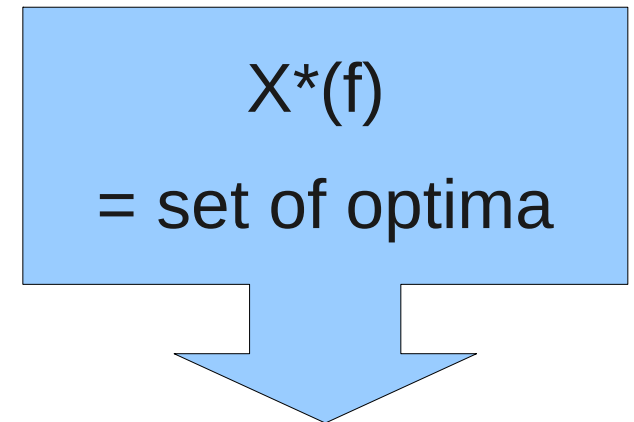
- *ES* ensures **convergence**, i.e.

$$\forall f \in \mathcal{F}, \forall (x, \sigma), ES(x, \sigma, f) \in X^*(f)$$

- Moreover, there is a **locality property**:

$$\forall x^* \in X^*(f), \exists (\epsilon, \sigma_0) > 0 \text{ such that } \|x - x^*\| < \epsilon \\ \wedge 0 < \sigma < \sigma_0 \Rightarrow ES(x, \sigma, f) = x^* .$$

If I start close enough to an optimum, with
small enough step-size, I find it.



Restart (RS) algorithm:

Given $S = (x_i, \sigma_i)_{i \in \{1, \dots, N\}} \in (D \times]0, \infty[)^N$,

$$x'_i(f) = ES(x_i, \sigma_i, f).$$

Given $S = (x_i, \sigma_i)_{i \in \{1, \dots, N\}} \in (D \times]0, \infty[)^N$,

$$x'_i(f) = ES(x_i, \sigma_i, f).$$

RS is **consistent** for \mathcal{F} if

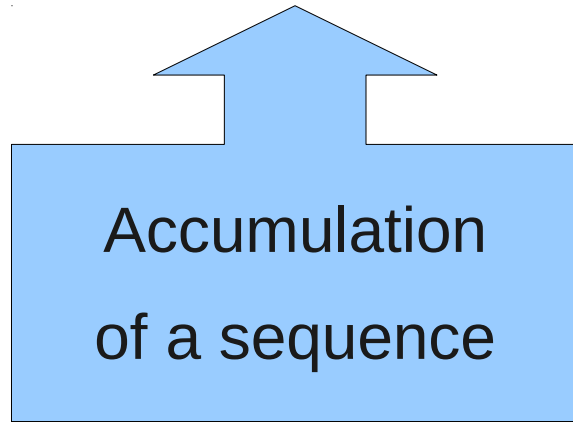
$$\forall f \in \mathcal{F}, X^*(f) = \{x'_i(f); i \in \{1, 2, \dots\}\}.$$

So, how to choose the parameters ?

(initial points and step-sizes) ?

What do you suggest ?

$$\text{Acc } S = \bigcap_{n \geq 1} \overline{\{s_{n+1}, s_{n+2}, \dots\}}.$$



$$\text{Acc } S = \bigcap_{n \geq 1} \overline{\{s_{n+1}, s_{n+2}, \dots\}}.$$

THEOREM:

1. If $\forall f \in \mathcal{F}, X^*(f) \times \{0\} \subset \text{Acc } S$, then RS is consistent for \mathcal{F} ;

$$\text{Acc } S = \bigcap_{n \geq 1} \overline{\{s_{n+1}, s_{n+2}, \dots\}}.$$

THEOREM:

1. *If $\forall f \in \mathcal{F}, X^*(f) \times \{0\} \subset \text{Acc } S$, then RS is consistent for \mathcal{F} ;*
2. *if $X \times \{0\} \subset \text{Acc } S$, then RS is consistent for all \mathcal{F} .*

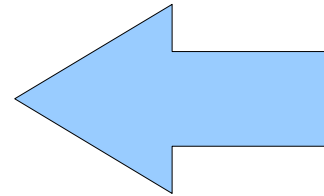
Ok, step-sizes should accumulate around 0,
and the x_i 's should be dense.

$X(i)$ = uniform random = ok

and

$\sigma(i) = \exp(\text{Gaussian})$

are ok.



Or something decreasing;
But not a constant.

(Why ?)

Should I do something more sophisticated ?

I want a small number of restarts.

**A criterion: dispersion (some maths
should be developed around why...).**

$$D(x, n) = \sup_{x \in [0, 1]^d} \inf_{i \in [1, n]} \|x - x_i\|.$$

$O\left((1/\epsilon)^d\right)$ for Halton or Hammersley point sets

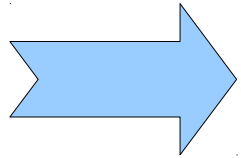
$\Omega\left((1/\epsilon)^d \log(1/\epsilon)\right)$ for random points

\implies small difference

\implies the key point is more the decreasing (or small) step-size

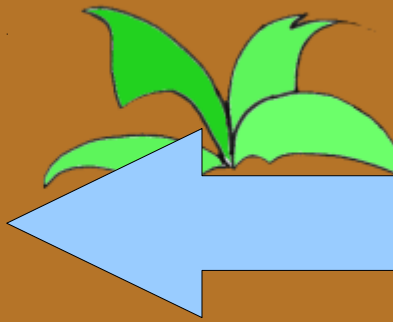
Classical methodologies:

1. Restarts



2. ``Real" diversity mechanisms

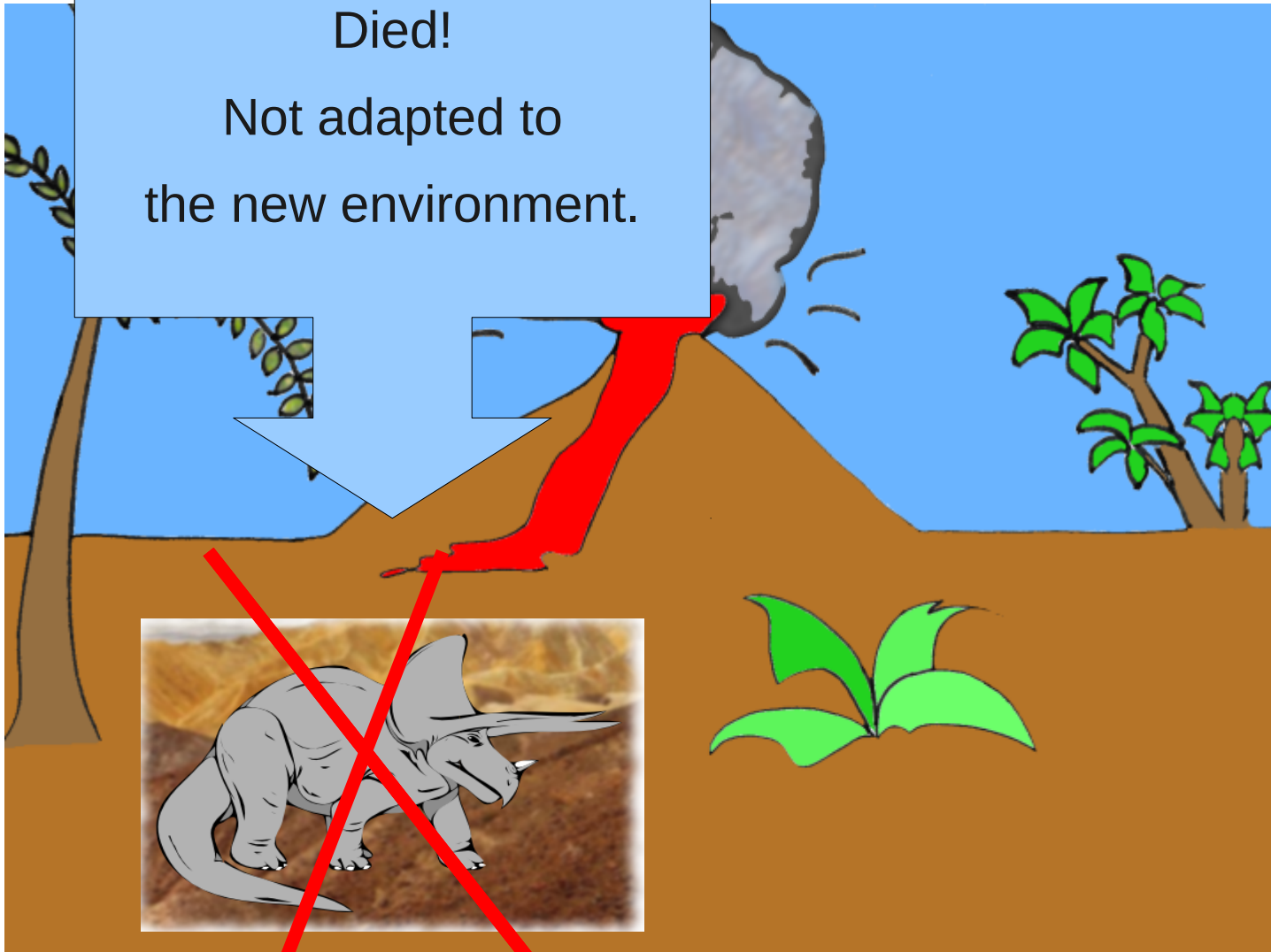
~ -7×10^7 years ago



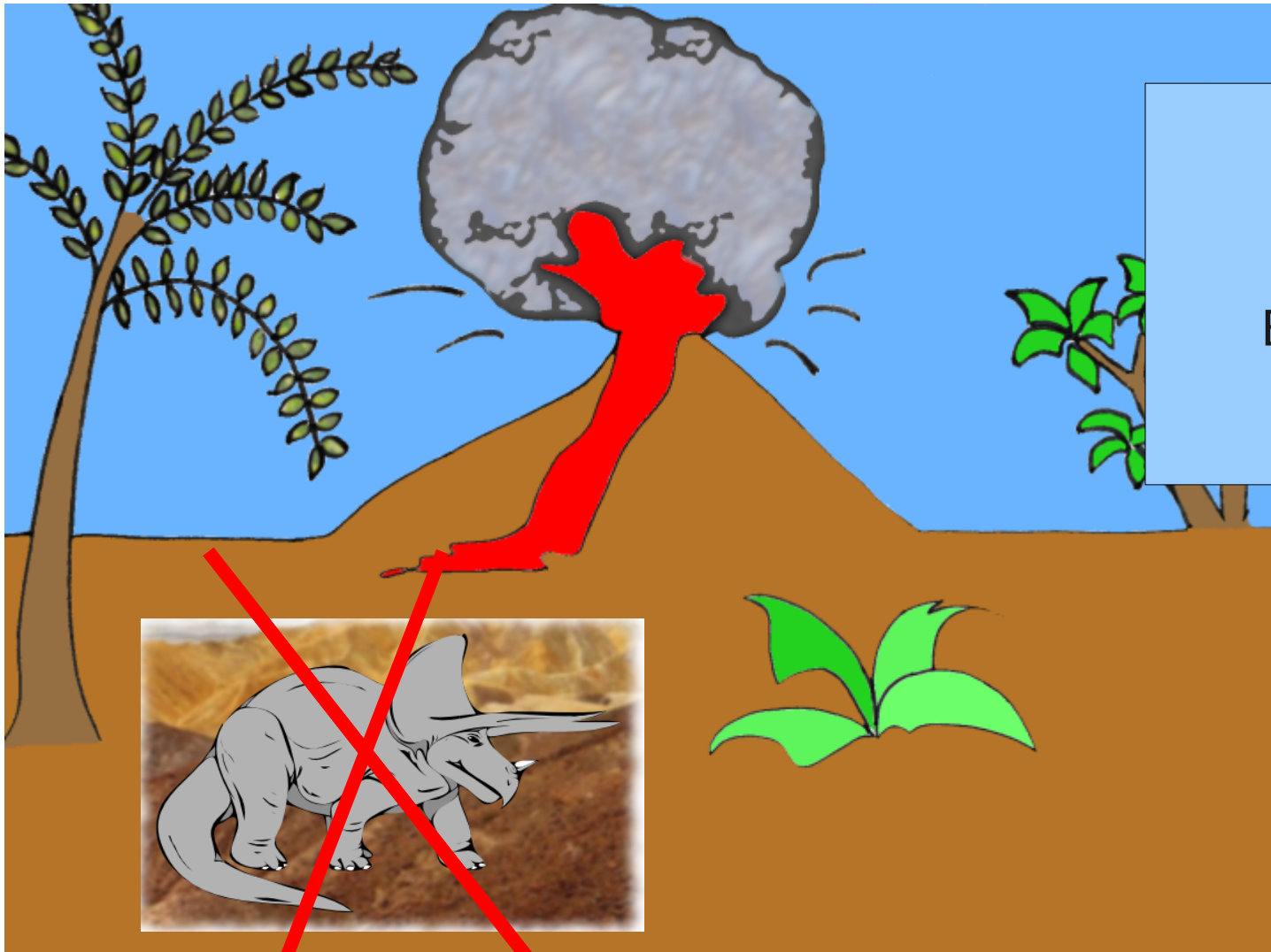
Highly optimized.
Hundreds of millions of
years on earth.

~ -7×10^7 years ago

Died!
Not adapted to
the new environment.



~ -7×10^7 years ago



Survived thanks
to a niching.
Better worst-case
analysis.





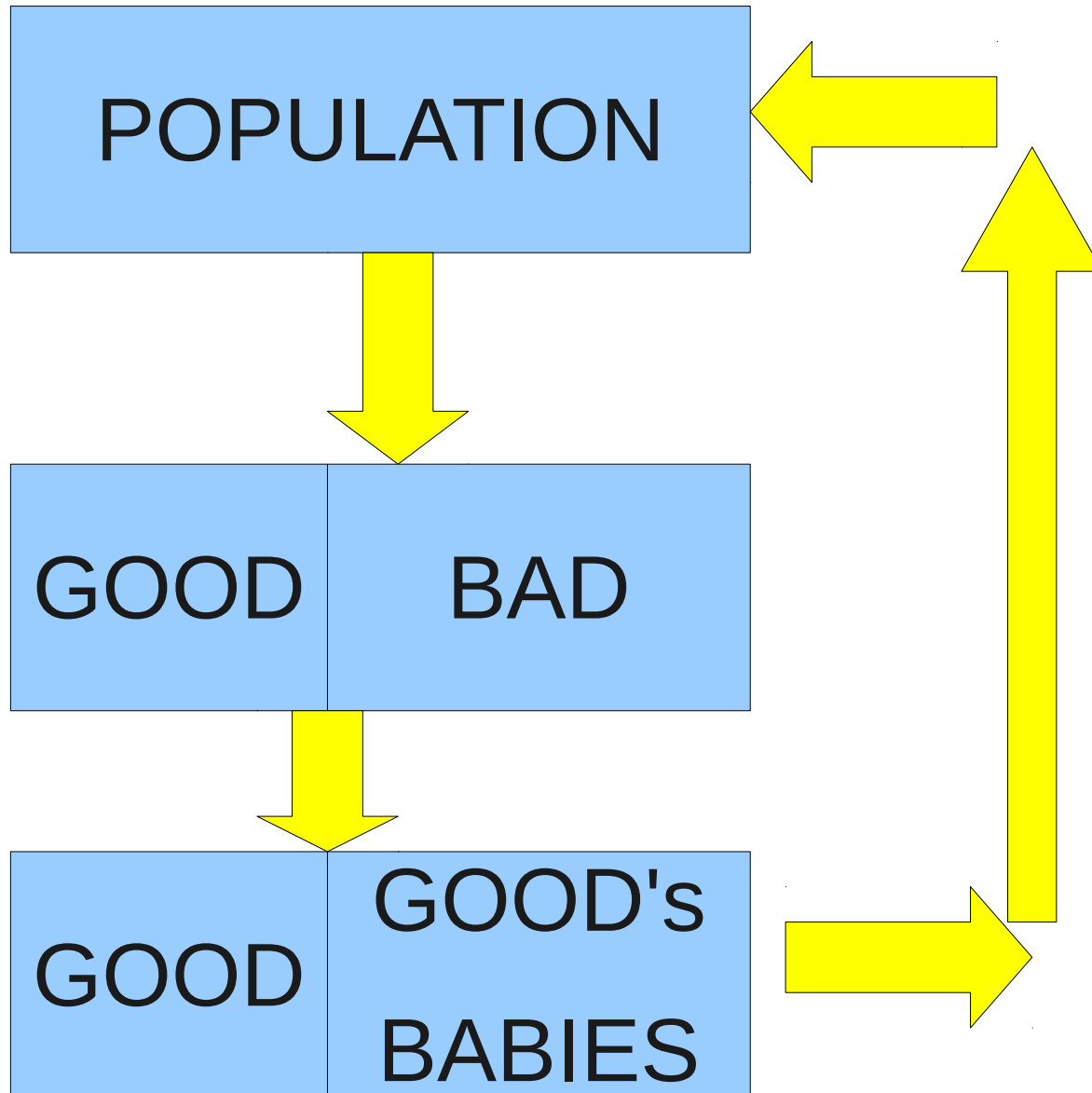
Eating Eucalyptus is dangerous for Health.

Except for him.

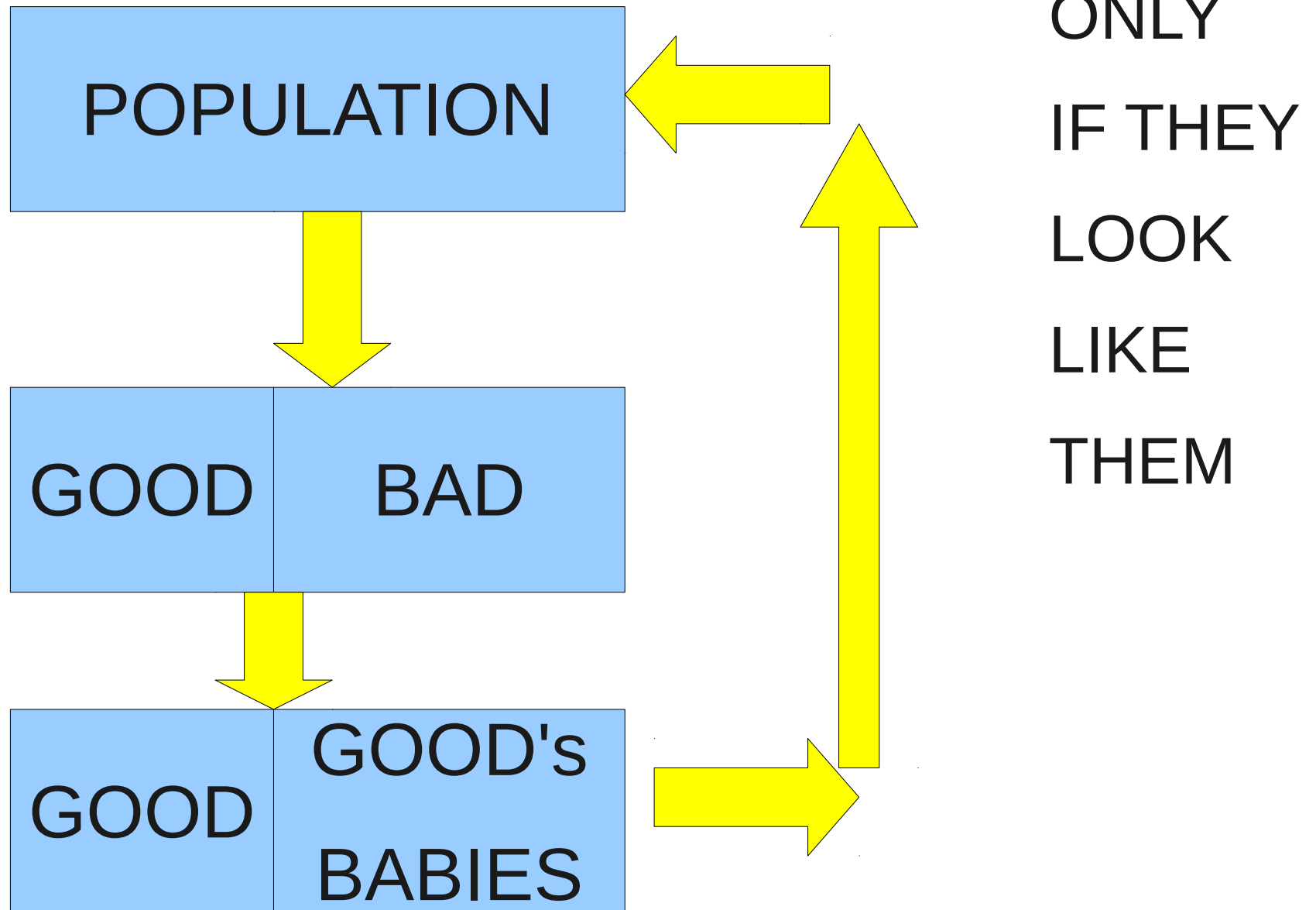
==> This is an example of Niching.

==> Niching is good for diversity.

USUAL SCHEME: WINNERS KILL LOOSERS



NEW SCHEME: WINNERS KILL LOOSERS



CLEARING ALGORITHM

Input: population

Output: smaller population

1) Sort the individuals (the best first)

2) Loop on individuals; for each of them (if alive)

kill individuals which are

(i) worse

(ii) at distance $<$ some constant

(...but the first K murders are sometimes canceled)

3) Keep at most μ individuals (the best)

I. Multimodal optimization

II. Adversarial / robust cases

a) What does it mean ?

b) Nash's stuff: computation

c) Robustness stuff: computation

III. Noisy cases

What is adversarial / robust optimization ?

I want to find $\operatorname{argmin} f$.

But I don't know exactly f

I just know that f is in a family of functions parameterized by Θ .

I can just compute $f(x, \theta)$, for a given θ .

What is adversarial / robust optimization ?

I want to find $\operatorname{argmin} f$.

But I don't know exactly f

I just know that f is in a family of functions parameterized by Θ .

I can just compute $f(x, \theta)$, for a given θ .

Criteria:

$\operatorname{Argmin}_X \mathbb{E} f(x, \theta)$.

$\theta \in \Theta$.

(average case; requires a probability distribution)

Or

$\operatorname{Argmin}_X \sup_{\theta \in \Theta} f(x, \theta)$.

$\theta \in \Theta$

\Leftarrow this is adversarial / robust.

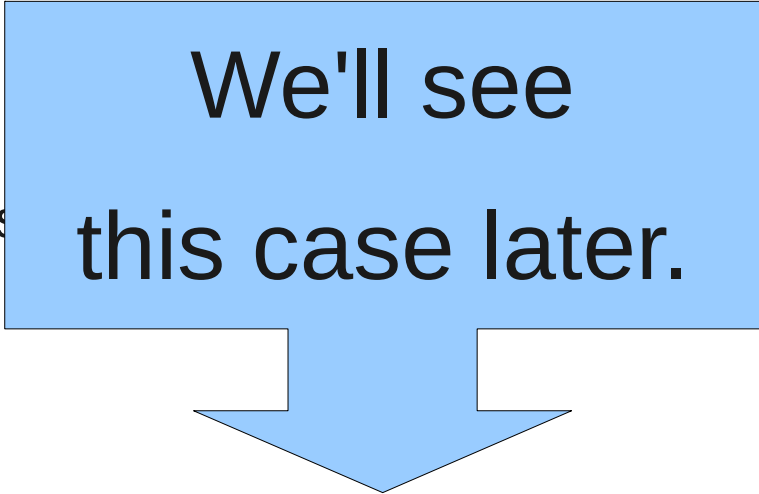
What is adversarial / robust optimization ?

I want to find $\operatorname{argmin} f$.

But I don't know exactly f

I just know that f is in a family of functions

I can just compute $f(x, \theta)$, for a given θ .



We'll see
this case later.

Criteria:

$\operatorname{Argmin}_{X, \theta} E f(x, \theta)$.

X, θ .

(average case; requires a
probability distribution)

Or

$\operatorname{Argmin}_{X, \theta} \sup f(x, \theta)$.

X, θ

\Leftarrow this is adversarial / robust.



Let's see this.



x

VS



$y (= \theta)$

Let's have a high-level look at all this stuff.

We have seen two cases:

$\inf_x \sup_y f(x,y)$ (best power plant for
worst earthquake ?)

$\inf_x \mathbb{E}_y f(x,y)$ (worst earthquake for
best power plant ?)

Does it really cover all possible models ?

Let's have a high-level look at all this stuff.

We have seen two

$\inf_x \sup_y f(x,y)$

$x \quad y$

$\inf_x \exists y f(x,y)$

$x \quad y$

A bit pessimistic.

This is the performance if y is chosen by an adversary who:

- knows us
- and has infinite computational resources

Does it really cover all cases ?

Let's have a high-level look at all this stuff.

We have seen two

$$\inf_x \sup_y f(x,y)$$

x y

$$\inf_x \mathbb{E}_y f(x,y)$$

x y

Does it really cover all cases ?

A bit pessimistic.

This is the performance if y is chosen by an adversary who:

- knows us (or can do many trials)
- and has infinite computational resources

Good for nuclear power plants.

Not good for economy, games...

Let's have a high-level look at all this stuff.

We have seen two

$\inf \sup_{x, y} f(x, y)$

x, y

$\inf \exists_{x, y} f(x, y)$

x, y

Does it really cover all cases ?

A bit pessimistic.

This is the performance if y is chosen by an adversary who:

~~- knows us~~

- and has infinite computational resources

Good for nuclear power plants.

Not good for economy, games...

Inf sup $f(x,y)$

$x \ y$

Replaced by sup inf $f(x,y)$?

$y \ x$

Inf sup $f(x,y)$

$x \quad y$

Replaced by sup inf $f(x,y)$?

$y \quad x$

Means that we know y ! No more robustness !

Who plays first ? x or y ?

Inf sup $f(x,y)$

$x \quad y$

Replaced by sup inf $f(x,y)$?

$y \quad x$

Means that we know y ! No more robustness !

Who plays first ? x or y ?

\implies simultaneous actions (Von Neumann)

\implies choose a distribution and not a single point!

$\inf_{x \in X} \sup_{y \in Y} f(x,y)$ is not equal to $\sup_{y \in Y} \inf_{x \in X} f(x,y)$

...but with finite domains:

$\inf_{x \in L(x)} \sup_{y \in L(y)} f(x,y)$ is equal to $\sup_{y \in L(y)} \inf_{x \in L(x)} f(x,y)$



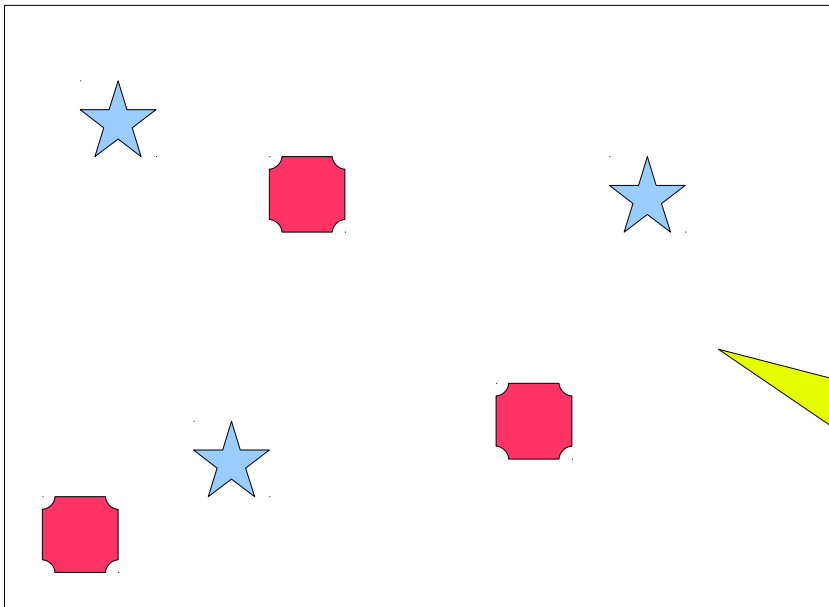
Von Neumann

Decision = positionning my distribution points.
Reward = size of Voronoi cells.

Simple model:

Each company makes his positionning
am morning for the whole day.

But choosing among strategies with
observations leads to nearly the same thing.



**Newspapers distribution,
or Quick vs MacDonalDs
(if customers ==> nearest)**

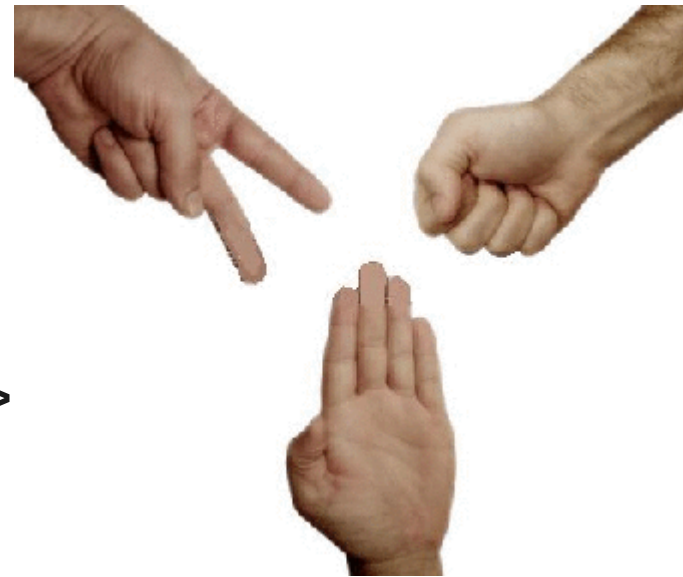
$\inf_x \sup_y f(x,y)$ is not equal to $\sup_y \inf_x f(x,y)$

E.g.: Rock-paper-scissors

Inf sup = I win

Sup inf = I loose

And $\sup \inf \sup \inf \sup \inf \dots \implies$



But equality in

Chess, draughts, ... (turn-based, full-information)

$\inf_x \sup_y f(x,y)$ is not equal to $\sup_y \inf_x f(x,y)$

...but with finite domains:

$\inf_{L(x)} \sup_{L(y)} f(x,y)$ is equal to $\sup_{L(y)} \inf_{L(x)} f(x,y)$

...and is equal to $\inf_x \sup_{L(y)} f(x,y)$

\implies The opponent chooses a randomized strategy without knowing what we choose.

$\inf_x \sup_y f(x,y)$ is not equal to $\sup_y \inf_x f(x,y)$

...but with finite domains:

$\inf_{L(x)} \sup_{L(y)} f(x,y)$ is equal to $\sup_{L(y)} \inf_{L(x)} f(x,y)$

...and is equal to $\inf_x \sup_{L(y)} f(x,y)$

==> The opponent chooses a randomized strategy without knowing what we choose.

==> Good model for average performance against non-informed opponents.

Let's summarize.

Nash: best distribution against worst

distribution = good model if opponent can't
know us, if criterion = average perf.

Best against worst: good in many games, and
good for robust industrial design

Be creative: $\inf \sup E$ is often reasonable

(e.g. \sup on opponent, E on climate...)

Let's summarize.

Nash: best distribution against worst

distribution good model if opponent can't

know μ = average perf.

B Nash: many games, and

Adversarial bidding (economy). sign

Choices on markets. reasonable

B Military strategy. on climate...)
Games.

Let's summarize.

Nash: best distribution against worst

distribution = good model if opponent can't
know us, if criterion = average perf.

Best against worst: good in many games, and
trial design

In repeated games, you can
Predict your opponent's decisions.

E.g.: Rock Paper Scissors.

s often reasonable

ment, E on climate...)

Let's summarize.

Nash: best distribution against worst

distribution = good mod if opponent can't

know us, if criterion = large perf.

Best a

Remarks on this “worst-case” analysis:

good

- if you're stronger (e.g. computer against humans in chess), take into account human's weakness
- in Poker, you will earn more money by playing against weak Opponents

Be cre

(

$\inf_x \sup_y f(x,y)$ is not equal to $\sup_y \inf_x f(x,y)$

...but with finite domains:

$\inf_{L(x)} \sup_{L(y)} f(x,y)$ is equal to $\sup_{L(y)} \inf_{L(x)} f(x,y)$

...and is equal to $\inf_x \sup_{L(y)} f(x,y)$

$x \in L(y)$

$L(x), L(y)$ is a

Nash equilibrium;

$L(x), L(y)$ not necessarily unique.

I. Multimodal optimization

II. Adversarial / robust cases

a) What does it mean ?

b) Nash's stuff: computation

- fictitious play

- EXP3

- coevolution

c) Robustness stuff: computation

III. Noisy cases

FIRST CASE:

everything finite (X and Θ are in finite domains).

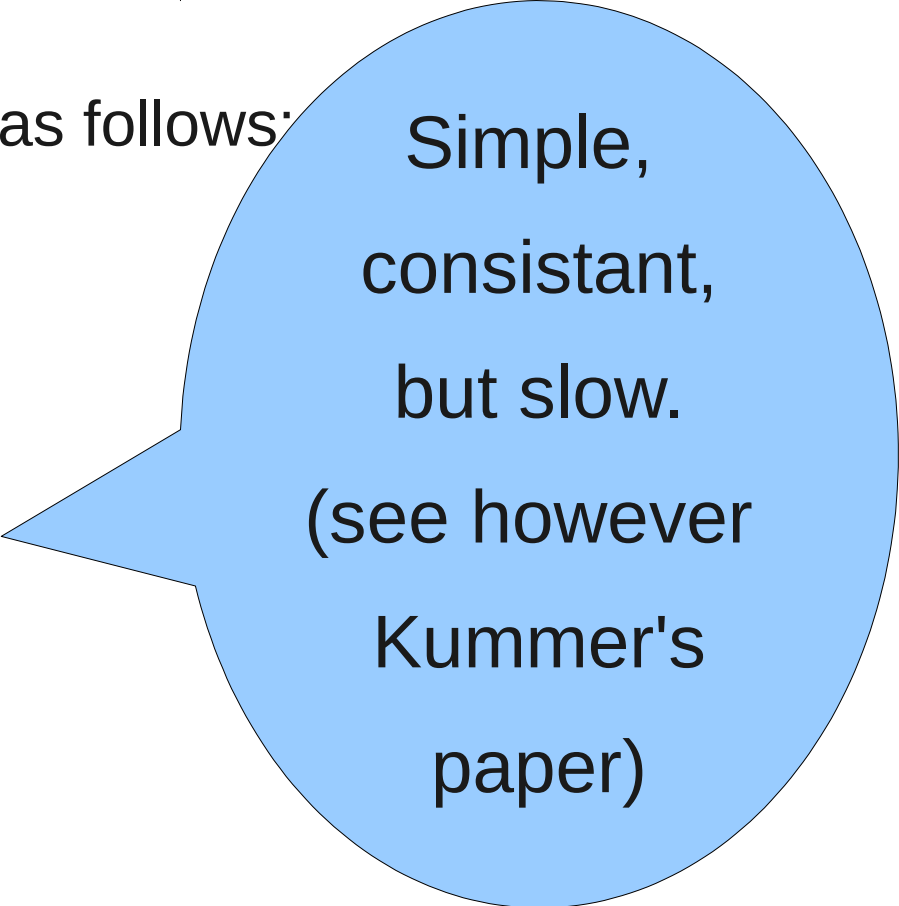
==> Fictitious Play (Brown, Robinson)

==> sequence of simulated games as follows:

$(x(n), y(n))$ defined as follows:

$$x(n+1) = \operatorname{argmin}_x \sum_{i < n+1} f(x, y(i))$$

$$y(n+1) = \operatorname{argmax}_Y \sum_{i < n+1} f(x(i), y)$$



Simple,
consistent,
but slow.
(see however
Kummer's
paper)

Much faster: EXP3, or INF (more complicated, see Audibert)

EXP3 (Exploration-Exploitation with Exponential weights):

Parameter: $\eta \in (0, 1/K]$.

Let p_1 be the uniform distribution over $\{1, \dots, K\}$.

For each round $t = 1, 2, \dots$,

- (1) Draw an arm I_t from the probability distribution p_t .
- (2) Compute the estimated gain for each arm: $\tilde{g}_{i,t} = \frac{g_{i,t}}{p_{i,t}} \mathbb{1}_{I_t=i}$ and update the estimated cumulative gain:
 $\tilde{G}_{i,t} = \sum_{s=1}^t \tilde{g}_{i,s}$.
- (3) Compute the new probability distribution over the arms:

$$p_{i,t+1} = (1 - K\eta) \frac{\exp\left(\eta \tilde{G}_{i,t}\right)}{\sum_{k=1}^K \exp\left(\eta \tilde{G}_{k,t}\right)} + \eta.$$

Complexity of EXP3 within logarithmic terms:

$$K / \epsilon^2$$

for reaching precision ϵ .

Complexity of EXP3 within logarithmic terms:

$$K / \epsilon^2$$

for reaching precision ϵ .

\implies we don't even read all the matrix of
the $f(x,y)$! (of size K^2)

Complexity of EXP3 within logarithmic terms:

$$K / \epsilon^2$$

for reaching precision ϵ .

==> we don't even read all the matrix of
the $f(x,y)$! (of size K^2)

==> provably better than all deterministic
algorithms

Complexity of EXP3 within logarithmic terms:

$$K / \epsilon^2$$

for reaching precision ϵ .

==> we don't even read all the matrix of
the $f(x,y)$! (of size K^2)

==> provably better than all deterministic
algorithms

==> requires random + RAM

Complexity of EXP3 within logarithmic terms:

$$K / \epsilon^2$$

for reaching precision ϵ .

Ok, it's great, but complexity K is still far too big.

For Chess, $\log_{10}(K) \approx 43$.

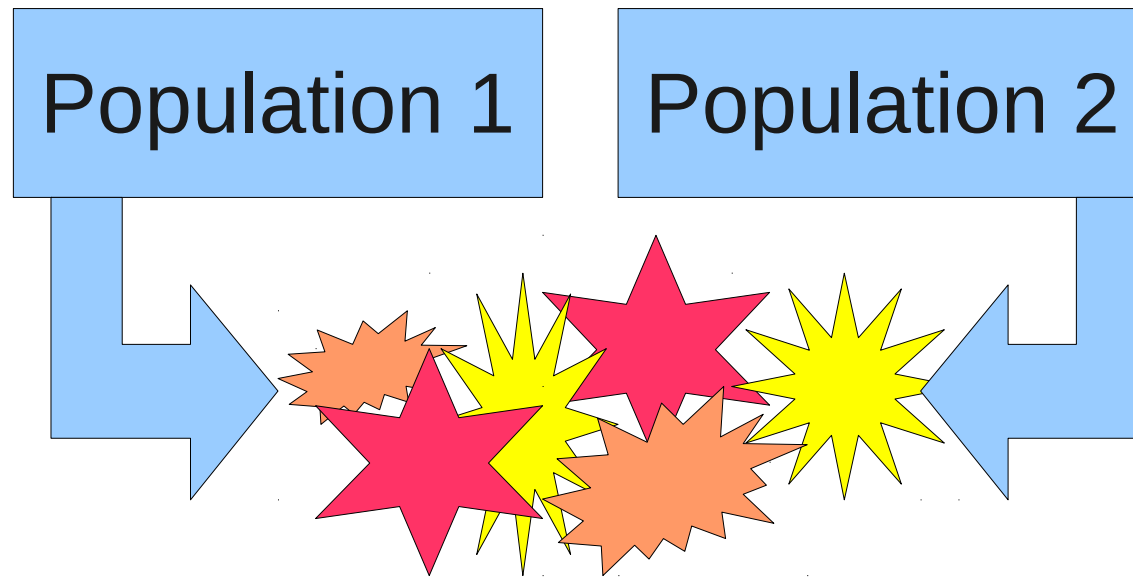
For the game of Go, $\log_{10}(K) \approx 170$.

Coevolution for K huge:

Population 1

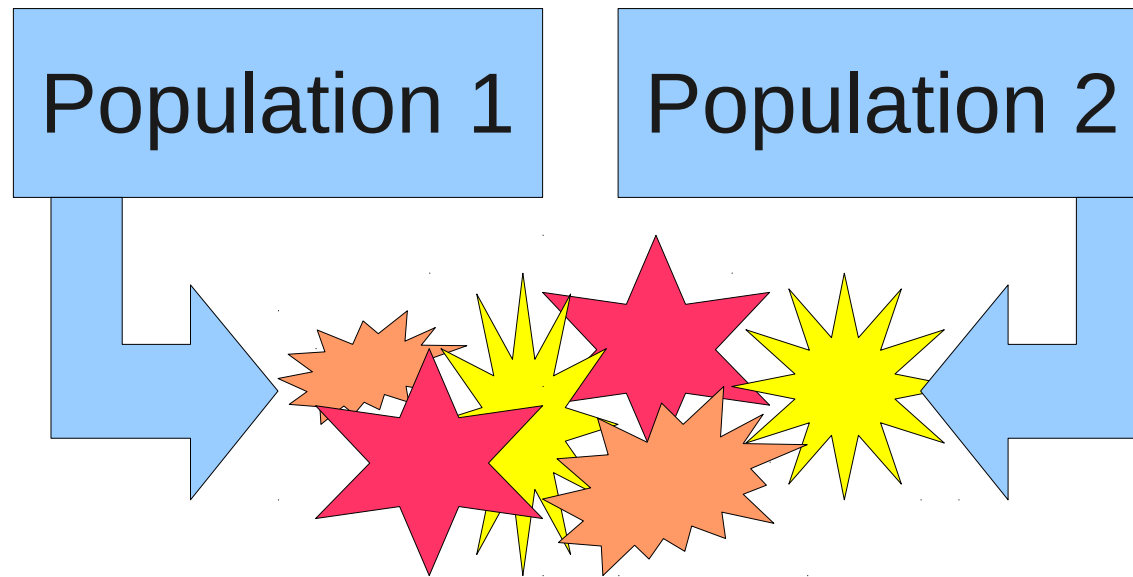
Population 2

Coevolution for K huge:



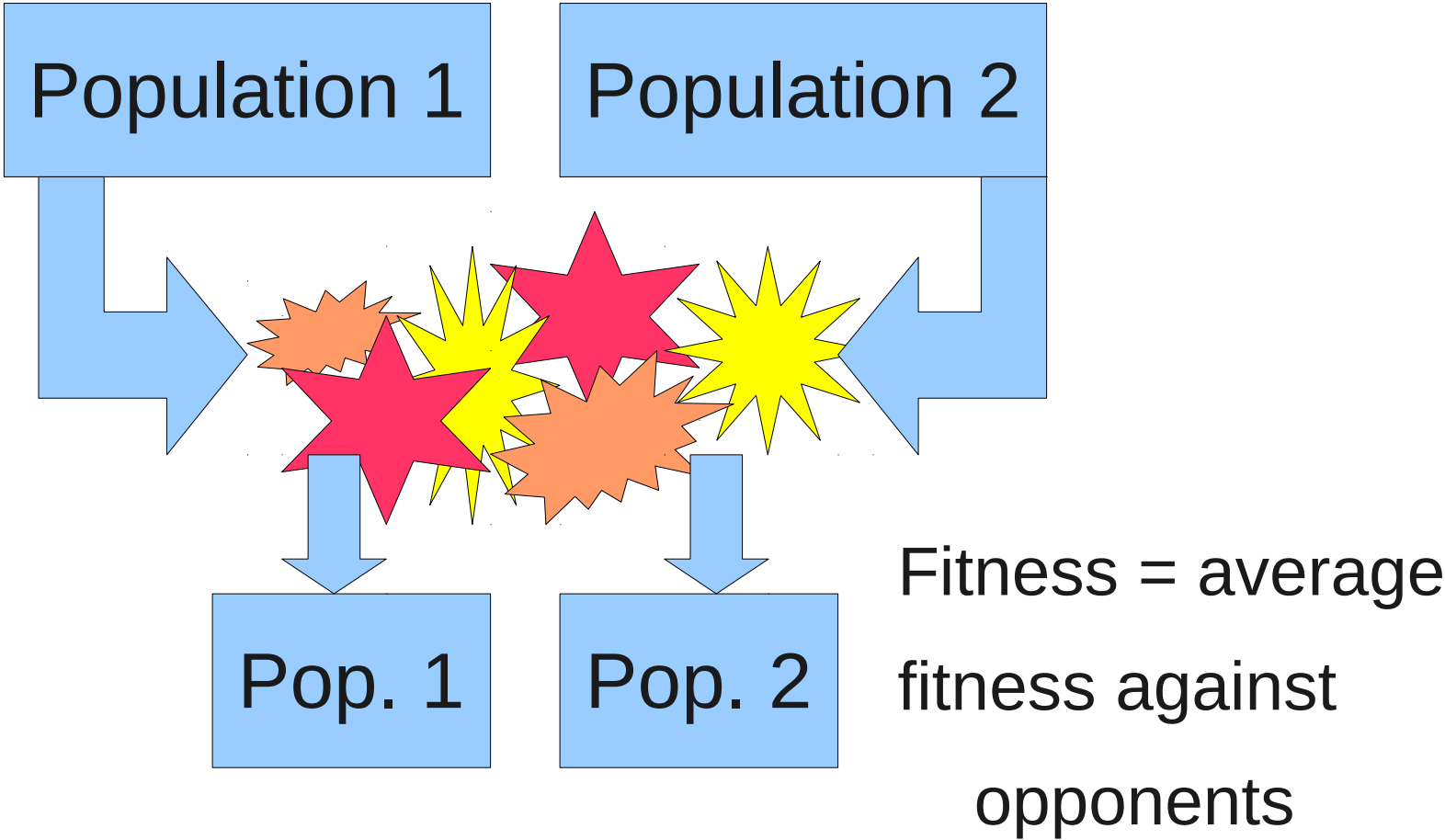
(play games)

Coevolution for K huge:

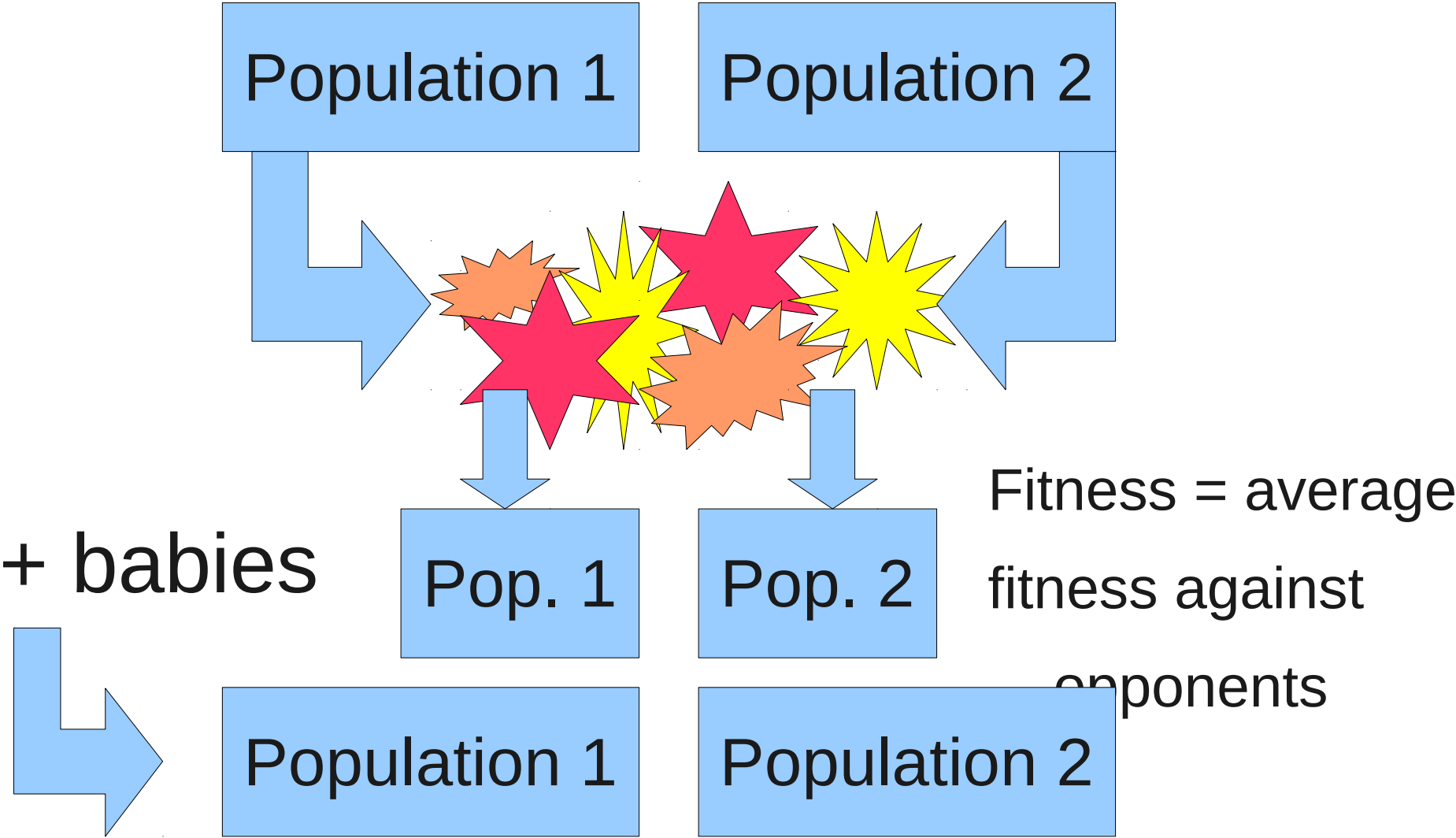


Fitness = average
fitness against
opponents

Coevolution for K huge:



Coevolution for K huge:



RED QUEEN EFFECT:

1. Rock vs Scissor
2. Rock vs Paper
3. Scissor vs Paper
4. Scissor vs Rock
5. Paper vs Rock
6. Paper vs Scissor
7. Rock vs Scissor = 1

RED QUEEN EFFECT:

1. Rock vs Scissor

2. Rock vs Paper

....

6. Paper vs Scissor

7. Rock vs Scissor = 1

==> Population with archive (e.g.

the best of each past iteration – looks like FP)

Reasonably good in very hard cases
but plenty of more specialized algorithms for
various cases:

- alpha-beta
- (fitted)-Q-learning
- TD(lambda)
- MCTS
- ...

I. Multimodal optimization

II. Adversarial / robust cases

a) What does it mean ?

b) Nash's stuff: computation

c) Robustness stuff: computation

III. Noisy cases

c) Robustness stuff: computation

\implies we look for $\operatorname{argmin}_x \sup_y f(x,y)$

\implies classical optimization for x

\implies classical optimization for y with
restart

Naive solution for robust optimization

Procedure Fitness(x)

$y = \text{maximize } f(x, \cdot)$

Return $f(x, y)$

Mimize fitness

Warm start for robust optimization

==> much faster

Procedure Fitness(x)

Static $y=y_0$;

$y = \text{maximize } f(x,y)$

Return $f(x,y)$

Mimize fitness

Population-based warm start for
robust optimization

Procedure Fitness(x)

Static $P=P_0$;

$P = \text{multimodal-maximize } f(x,P)$

Return $\max f(x,y)$

$y \text{ in } P$

Mimize fitness

Population-based warm start for robust optimization

$$f(x,y) = \sin(y) + \cos((x+y) / \pi) \quad x,y \text{ in } [-10,10]$$

==> global maxima (in y) moves quickly (==> looks hard)

==> but local maxima (in y) move slowly as a function of x

==> warm start very efficient

Population-based warm start for robust optimization

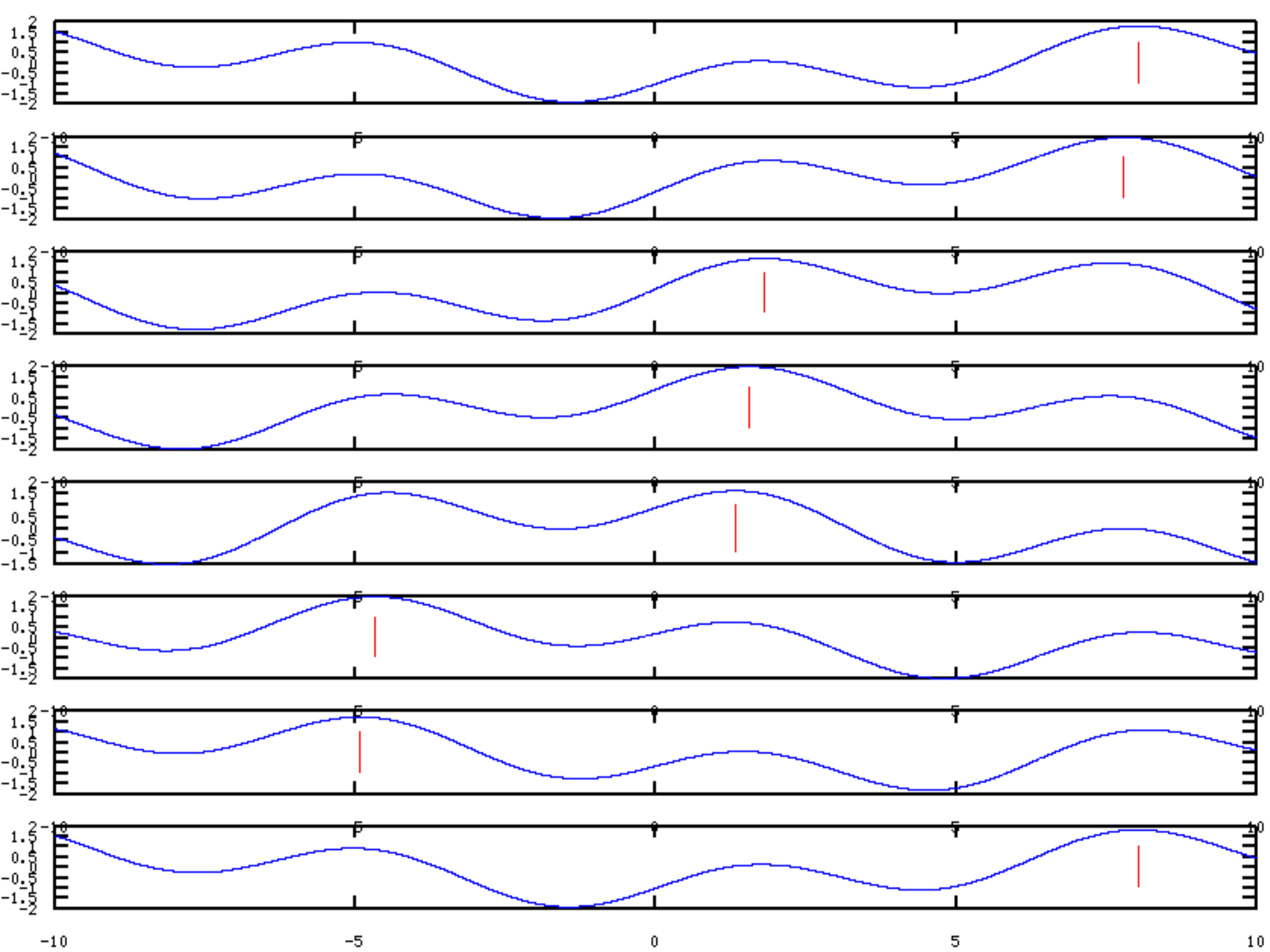
$$f(x,y) = \sin(y) + \cos((x+y) / \pi) \quad x,y \text{ in } [-10,10]$$

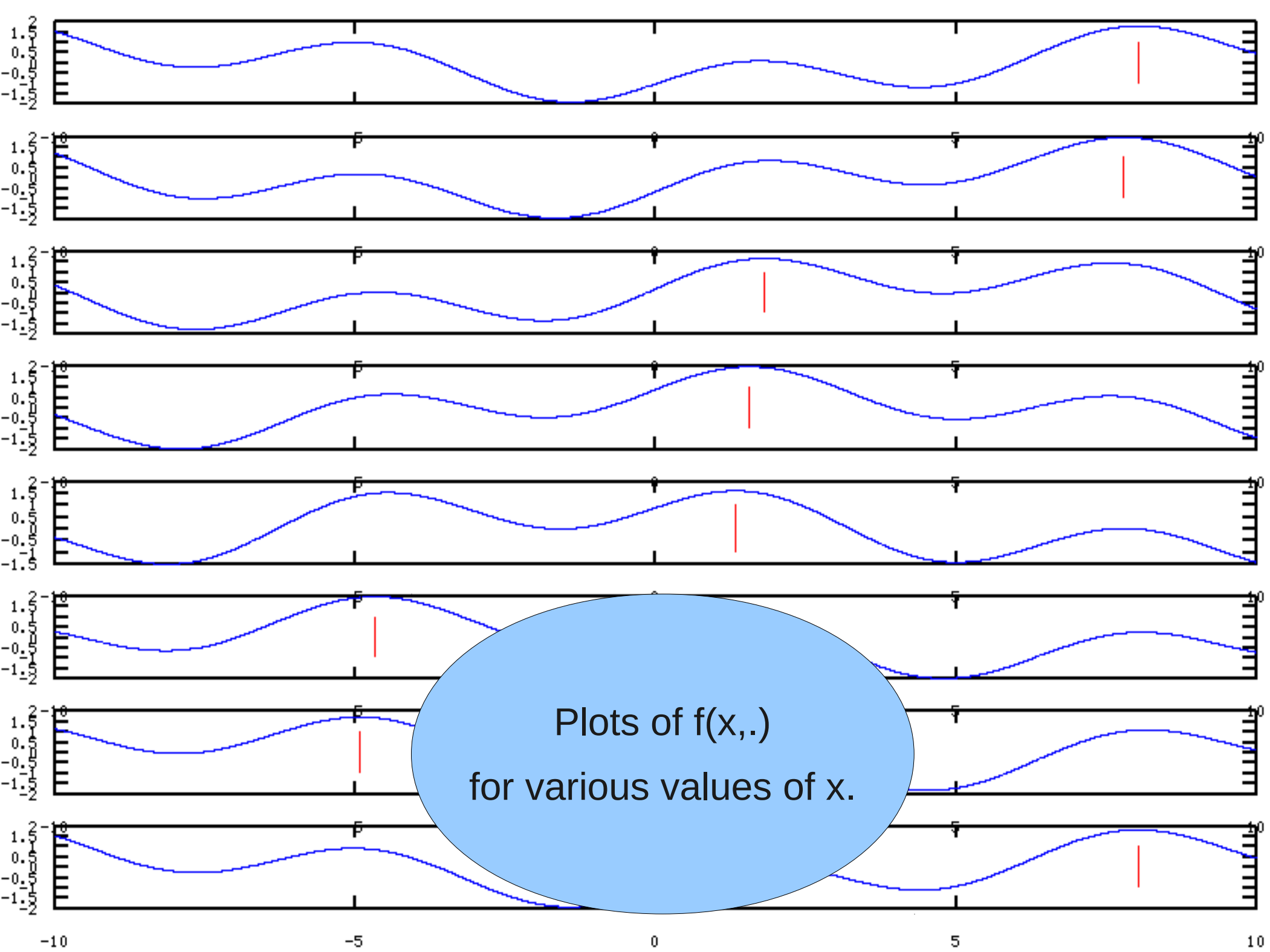
==> global maxima (in y) moves quickly (==> looks hard)

==> but local maxima (in y) move slowly as a function of x

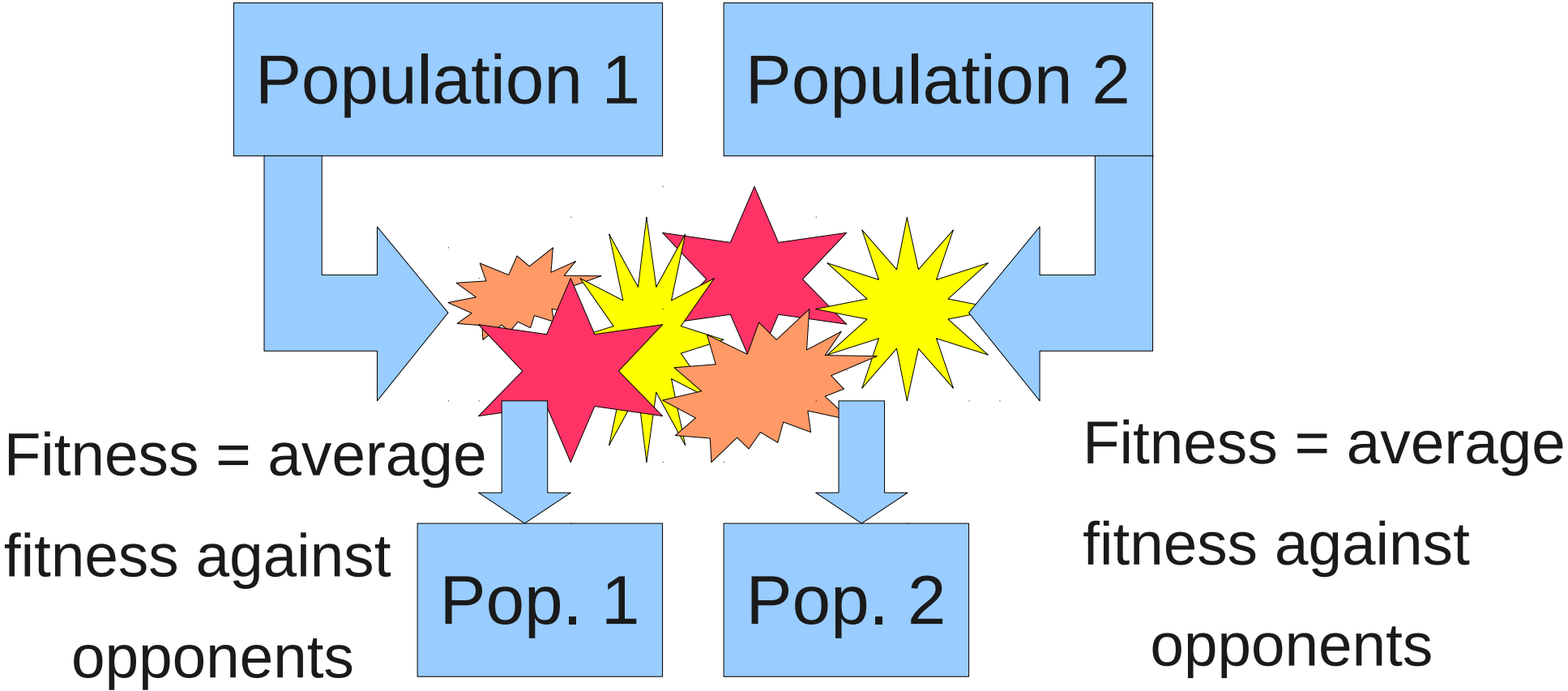
==> warm start very efficient

==> interestingly, we just rediscovered coevolution :-)

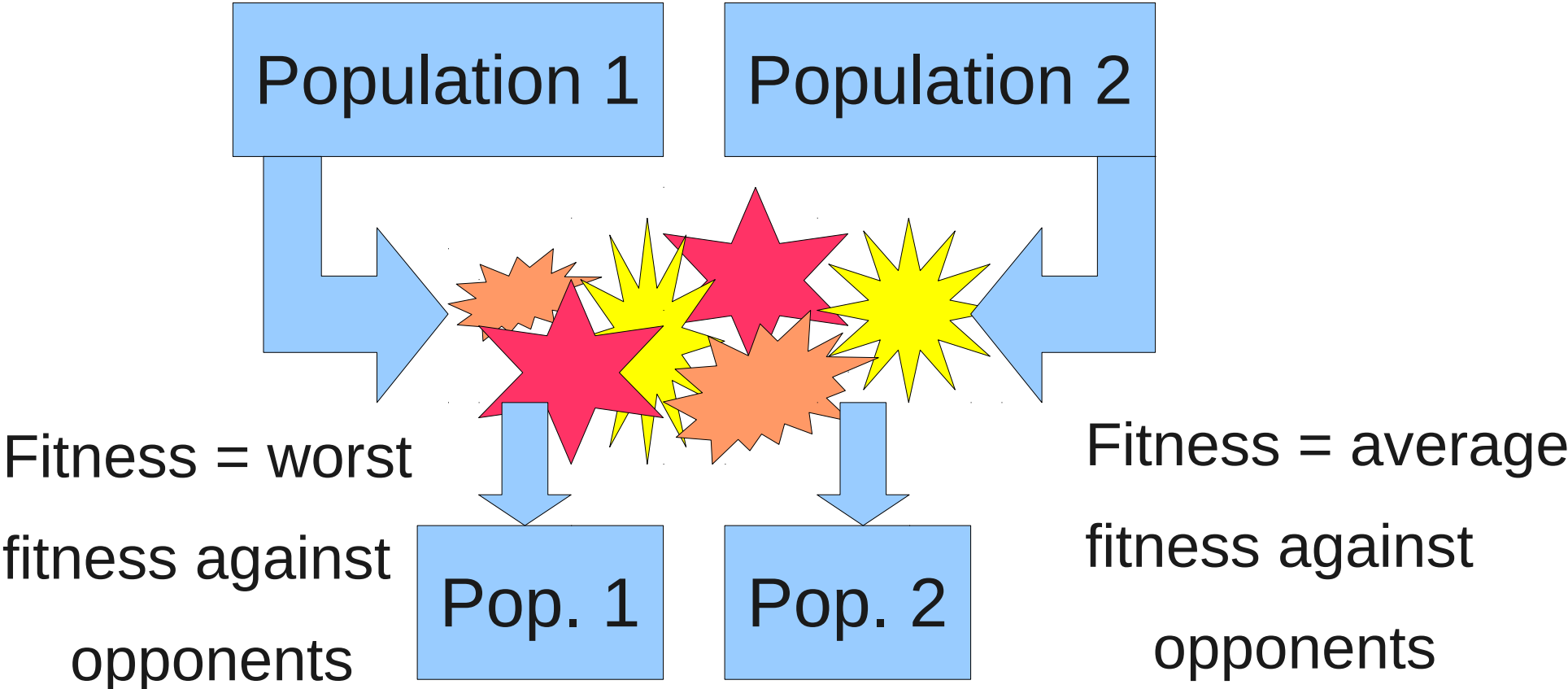




Coevolution for Nash:



Coevolution for inf sup:



I. Multimodal optimization

II. Adversarial / robust cases

III. Noisy cases

Arg min $E f(x,y)$

x y

1) Monte-Carlo estimates

2) Races: choosing the precision

3) Using Machine Learning

Monte-Carlo estimates

$$E f(x,y) = 1/n \sum f(x, y_i) + \text{error}$$

Error scaling as $1/\text{squareRoot}(n)$

==> how to be faster ?

==> how to choose n ? (later)

Faster Monte-Carlo evaluation:

- more points in critical areas (prop. to standard deviation), (+rescaling!)
- evaluating $f-g$, where g has known expectation and is “close” to f .
- evaluating $(f + f \circ h)/2$
where h is some symmetry

Faster Monte-Carlo evaluation:

- more points in critical areas (prop. to standard deviation), (+rescaling!)
- evaluating f - g , where g has known expectation and f is the target function
- evaluating $(f + f \circ h)/2$ where h is some symmetry

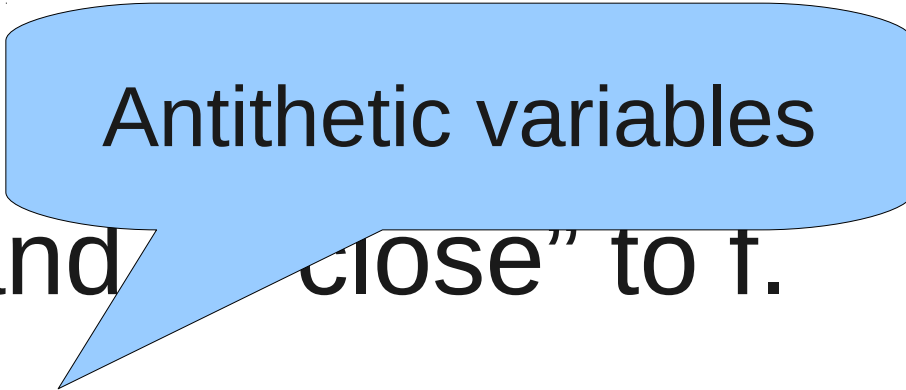
Importance sampling

Faster Monte-Carlo evaluation:

- more points in critical region (rescaling!)
standard deviation
- evaluating $f-g$, where g has known expectation and is “close” to f .
- evaluating $(f + f \circ h)/2$
where h is some symmetry

Control variable

Faster Monte-Carlo evaluation:

- more points in critical areas (prop. to standard deviation), (+rescaling!)
- evaluating $f-g$, where g is an "antithetic" expectation and f is "close" to f .

- evaluating $(f + f \circ h)/2$
where h is some symmetry

And quasi-Monte-Carlo evaluation ?

$\log(n)^d / n$ instead of $\text{std} / \text{sqrt}(n)$

==> much better for n really large (if some smoothness)

==> much worse for d large

Arg min $E f(x,y)$

1) Monte-Carlo estimates

2) Races: choosing the precision

3) Using Machine Learning

Bernstein's inequality (see Audibert et al):

with probability at least $1 - \delta$

$$|\bar{X}_t - \mu| \leq \bar{\sigma}_t \sqrt{\frac{2 \log(3/\delta)}{t}} + \frac{3R \log(3/\delta)}{t},$$

Where: $\bar{\sigma}_t^2 = \frac{1}{t} \sum_{i=1}^t (X_i - \bar{X}_t)^2.$

R = range of the X_i 's

$$\bar{X}_t = 1/t \sum_{i=1}^t X_i.$$

Comparison-based optimization with noise:
how to find the best individuals by a race ?

While (I want to go on)

{

I evaluate once each arm.

I compute a lower and upper bound for all
non-discarded individuals

(by Bernstein's bound).

I discard individuals which are excluded
by the bounds.

}

Trouble:

What if two individuals have the same fitness value ?

==> infinite run !

Tricks for avoiding that at iteration N:

- limiting number of evals by some function of N
(e.g. $\exp(N)$ if a log-linear convergence is expected w.r.t number of iterations)
- limiting precision by some function of sigma
(e.g. $\text{precision} = \sqrt{\text{sigma}} / \log(1/\text{sigma})$)

Arg min $E f(x,y)$

1) Monte-Carlo estimates

2) Races: choosing the precision

3) Using Machine Learning

USING MACHINE LEARNING

$$x_{n+1} = \text{Opt}(x_1, f(x_1), \dots, x_n, f(x_n)).$$

Statistical tools: $f'(x)$ = approximation

$$(x, x_1, f(x_1), x_2, f(x_2), \dots, x_n, f(x_n))$$

$$y(n+1) = f'(x(n+1))$$

e.g. f' = quadratic function closest to f on the $x(i)$'s.

and $x(n+1) = \text{random}$ or $x(n+1) = \text{maxUncertainty}$

or $x(n+1) = \text{argmin } E f$ or $x(n+1) = \text{argmax } E \max(0, \text{bestSoFar} - E f(x))$

USING MACH

Kriging, RBF, SVM,
Gaussian Processes,
Or $f'(x)$ = maximum likelihood
on a parametric noisy model

$$x_{n+1} = \text{Op}$$

)

Statistical tools: $f'(x)$ = parametric approximation

$$(x, x_1, f(x_1), x_2, f(x_2), \dots, x_n, f(x_n))$$

$$y(n+1) = f'(x(n+1))$$

e.g. f' = logistic quadratic function closest to f on the $x(i)$'s.

and $x(n+1) = \text{random}$ or $x(n+1) = \text{maxUncertaintyRed}$

or $x(n+1) = \text{argmin } f'$ or $x(n+1) = \text{argmax } E \max(0, \text{bestSoFar} - f(y))$

USING MACHINE LEARNING

$$x_{n+1} = \text{Opt}(x_1, f(x_1), \dots, x_n, f(x_n)).$$

Very fast if good model
(e.g. quadratic model + noise)

Statistical tools: $f'(x) = a$

Also termed
“expected improvement”

(1)

on closest to f of the $x(i)$'s.

and $x(n+1) = \text{random}$ or $x(n+1) = \text{maxUncertaintyRed}$
or $x(n+1) = \text{argmin } f'$ or $x(n+1) = \text{argmax } E \max(0, \text{bestSoFar} - E f(x))$

So some tools for noisy optimization:

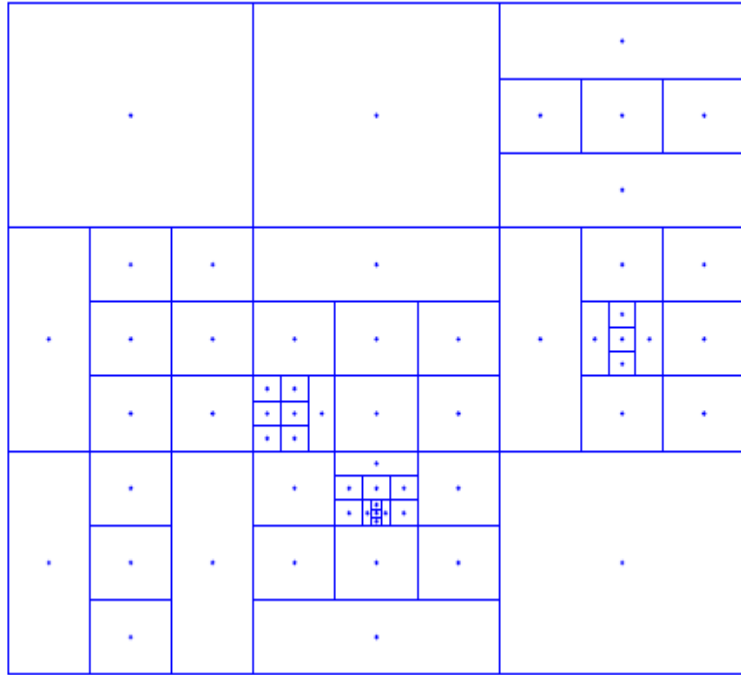
1. introducing races + tricks

2. using models equipped

with uncertainty

(2. more difficult, but

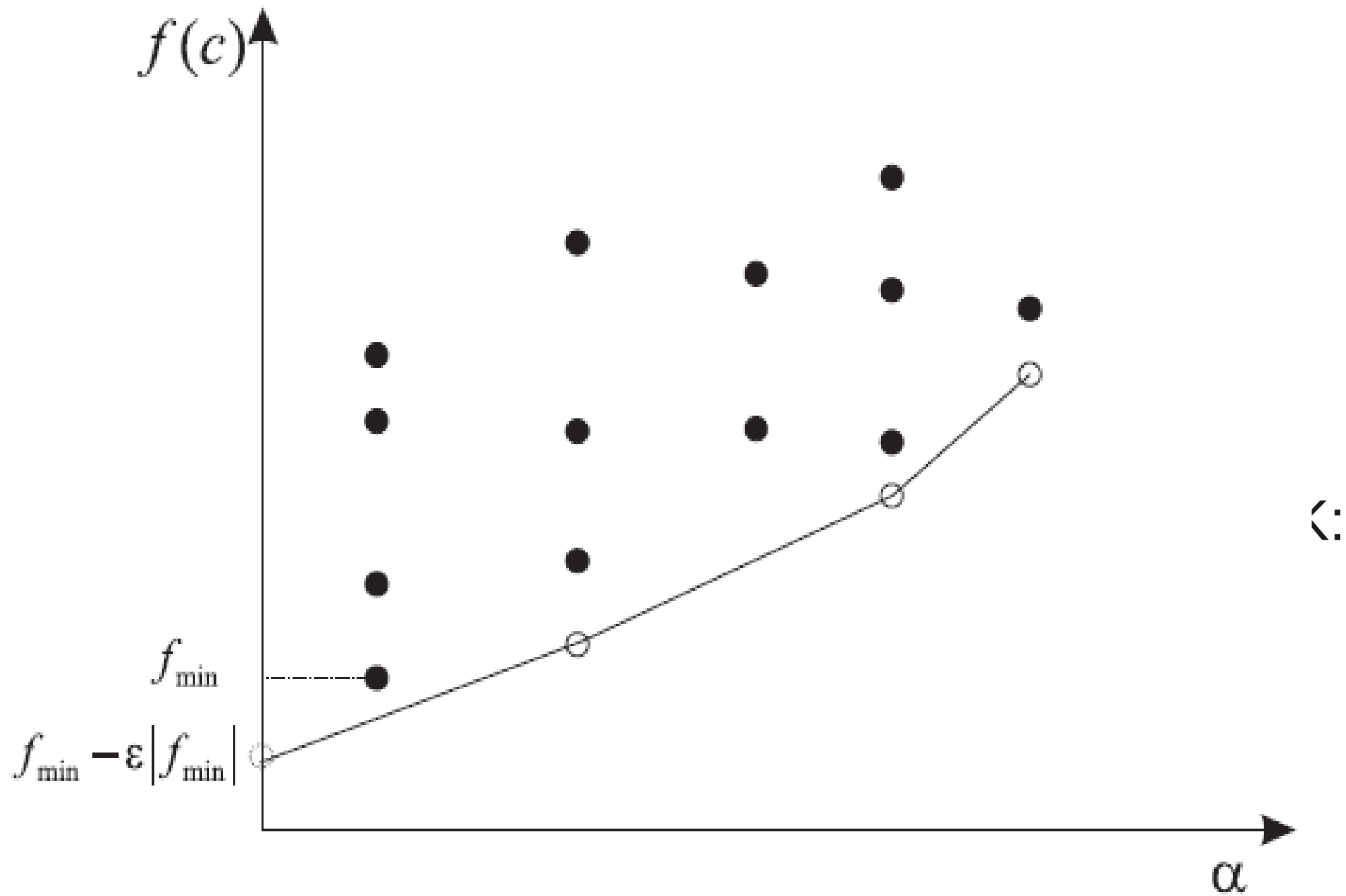
sometimes much more efficient)



A rectangle j of size $\alpha(j)$ is splitted if for some K :

$$f(c_j) - \tilde{K} \alpha_j \leq f(c_i) - \tilde{K} \alpha_i, \forall i \in \mathcal{H},$$

$$f(c_j) - \tilde{K} \alpha_j \leq f_{\min} - \varepsilon |f_{\min}|.$$



Warm starts

Coevolution

Niching

Clearing

Multimodal optimization

Restarts (sequential / parallel)

Koalas and Eucalyptus

Dinosaurs (and other mass extinction)

Fictitious Play

EXP3

EGO

Robust optimization

Noisy optimization

Nash equilibria

Surrogate models

Maximum uncertainty

Monte-Carlo estimate

Quasi-Monte-Carlo

Van Der Corput

Halton

Scrambling

Antithetic variables, control

variables, importance sampling.

EXERCISES (to be done at home, deadline given soon):

- 1. Implement a (1+1)-ES and test it on the sphere function $f(x)=\|x\|^2$ in dimension 3. Plot the convergence.**

Nb: choose a scale so that you can check the convergence rate conveniently.

- 2. Implement a pattern search method (PSM) and test it on the sphere function, also in dimension 3 (choose the same scale as above).**

3. We will now consider a varying dimension;

- $x(n,d)$ will be the n th visited point in dimension d for the (1+1)-ES, and**
- $x'(n,d)$ will be the n th visited point in dimension d for the PSM.**

For both implementations, choose e.g. $n=5000$, and plot $x(n,d)$ and $x'(n,d)$ as a function of d . Choose a convenient scaling as a function of d so that you can see the effect of the dimension.

4. What happens if you consider a quadratic positive definite ill-conditioned function instead of the sphere function ?

E.g.: $f(x) = \text{sum of } 10^i x_i^2$

= $10x(1)^2 + 100x(2)^2 + 1000x(3)^2$ in dimension 3; test in Dimension 50.

5. (without implementing) Suggest a solution for the problem in Ex. 4.