Memory and Machine Learning
Paramater Estimation and Numerical Optimization
M1 SCA

Laure Buhry laure.buhry@loria.fr
LORIA bureau C042

Semester 2

Plan

1. Introduction to Optimization
2. Deterministic Methods
   - Gradient descent
   - Second Order Methods
3. Stochastic Optimization : some metaheuristics
   - Simulated Annealing
   - Evolutionary Algorithms
     - Genetic Algorithms and Evolutionary Strategies
     - Differential Evolution
     - Particle Swarm Optimization

Optimization problems

Any problem that requires a **parameter estimation**

Optimization problems

Any problem that requires a **parameter estimation**

Examples :

- Linearization from observations
- Reproduction of real observed data with a mathematical function
- OParameter optimization of a model (whatever the mathematical model, whatever the application domain)

Optimization : formal definition

Optimization

Minimization :

Given a function $f : A \to \mathbb{R}$ defined on a set A with values in $\mathbb{R}$ or in $\bar{\mathbb{R}} := \mathbb{R} \cup \{-\infty, +\infty\}$), the ***minimization of*** f or the ***optimization along the parameter*** x, is the search of an element x^* de A such that :

$$\forall x \in A, \qquad f(x^*) \leq f(x)$$

Remark : A can be of any dimension

Optimization : formal definition

Maximization :

One can transform a minimization problem into a maximization problem (and vice versa). Then, we look for a element x^* in A such that :

$$\forall x \in A, \qquad f(x^*) \geq f(x)$$

Optimization : in practice

Often, when this is possible, one transforms the problem on f to a problem on function g such that $\forall x\, g(x) \geq 0$ et $g(x^*) = 0$.
One thus want to minimize g.

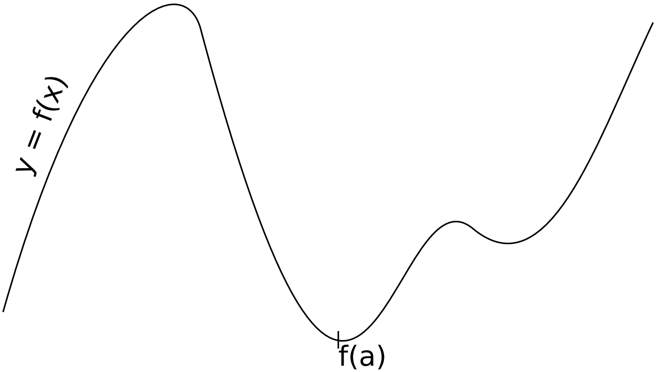
Some Other Definitions

f is called ***cost function***, ***objective function***, ***fitness function***, ***cost***, ***criteria***, or ***objective***.

Example in 2D

f : cost function

f reaches its global minimum in a

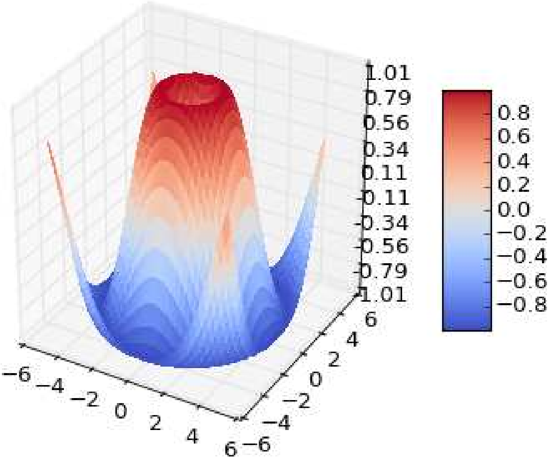


Minimizing f is equivalent to looking for a

Example in 2D

Any idea of a concrete example ?

Example in 3D



Optimization problems can be solved from a formal analytical point of view

but also with numerical methods

Optimization problems can be solved from a formal analytical point of view

but also with numerical methods

This course deals with **numerical optimization**

Two Main Classes of Methodes

- **Deterministic Methods**

  used for ***local search***

  requires the objective function to have specific properties.

- **Stochastic Methods**

  used for ***global search***, but can also be used for local search

Plan

1. Introduction to Optimization
2. Deterministic Methods
   - Gradient descent
   - Second Order Methods
3. Stochastic Optimization : some metaheuristics
   - Simulated Annealing
   - Evolutionary Algorithms
     - Genetic Algorithms and Evolutionary Strategies
     - Differential Evolution
     - Particle Swarm Optimization

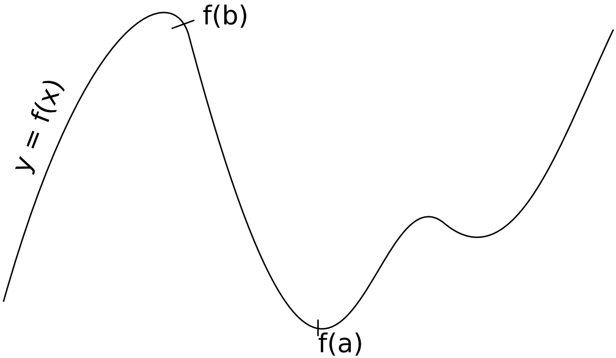
Plan

1. Introduction to Optimization
2. Deterministic Methods
   - Gradient descent
   - Second Order Methods
3. Stochastic Optimization : some metaheuristics
   - Simulated Annealing
   - Evolutionary Algorithms
     - Genetic Algorithms and Evolutionary Strategies
     - Differential Evolution
     - Particle Swarm Optimization

Gradient descent

One knows :

- f : cost function
- One point b
- that f is dérivable and one knows how to calculate its derivative.

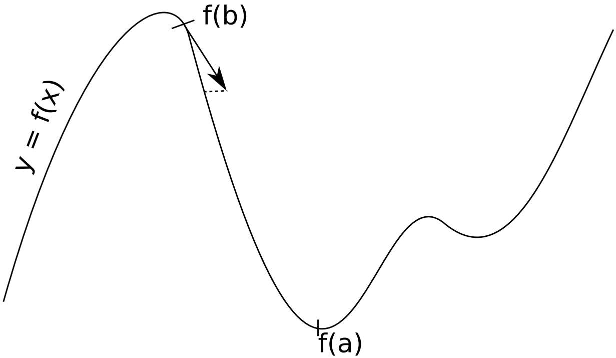


One wants to determine a.

Gradient Descent

One knows :

- f : cost function
- One point b
- that f is dérivable and one knows how to calculate its derivative.

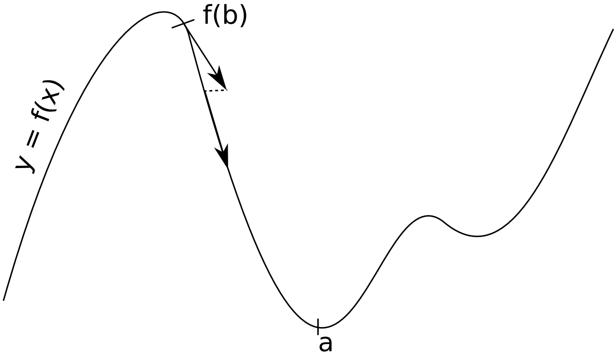


One wants to determine a.

Descente de gradient

One knows :

- f : cost function
- One point b
- that f is differentiable and one knows how to calculate its derivative.

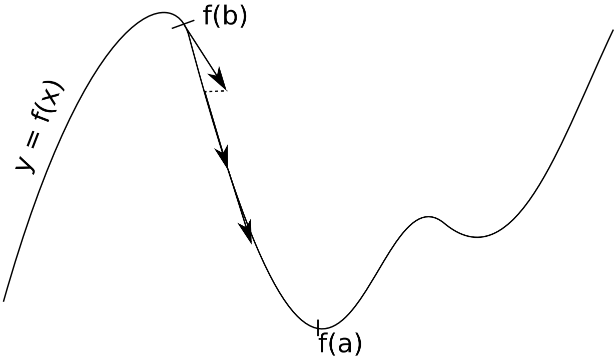


One wants to determine a.

Descente de gradient

One knows :

- f : cost function
- One point b
- that f is dérivable and one knows how to calculate its derivative.

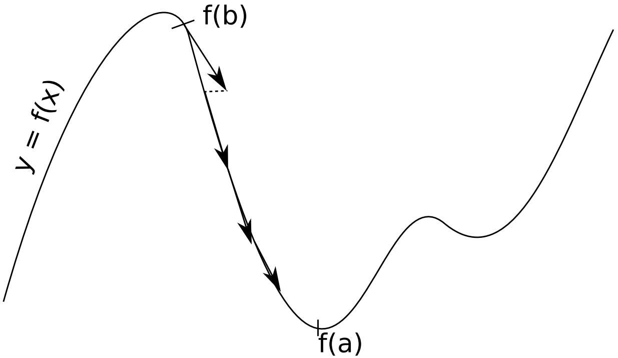


One wants to determine a.

Descente de gradient

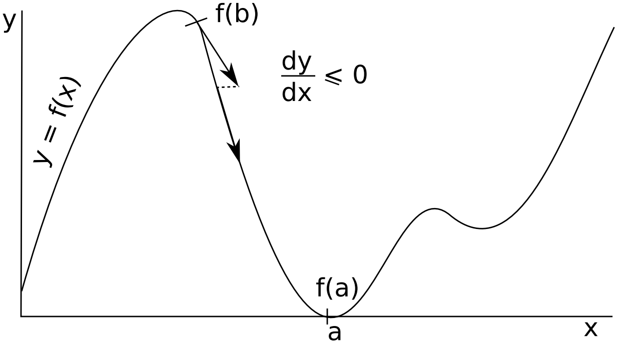
One knows :

- f : cost function
- One point b
- that f is dérivable and one knows how to calculate its derivative.



One wants to determine a.

Gradient Descent



One wants to go towards $x_1 = b - \alpha \frac{\partial(y)}{\partial(x)}$ beacause $f(x_1) \leq f(b)$.

Gradient Descent

The notion of gradient

The gradient ∇f is a vector that represents the variation of a function depending on the variation of its different parameters

In other words, if $\vec{i}$ is the unit vector,
$\vec{\nabla} f(x) = f'(x) \cdot \vec{i} = \frac{\partial f}{\partial x}(x) \cdot \vec{i}$

Gradient Descent

When to stop the descent ?

Gradient Descent

When to stop the descent ?

most of the time, when the gradient becomes very small

Gradient Descent : algorithm

Gradient Descent Algorithm (method of the steepest descent)

Gradient Descent : algorithm

Gradient Descent Algorithm (method of the steepest descent)

Let $x_0 \in \mathbb{A}$ be an inital point (or iterated point) (b in the example) and a threshold $\varepsilon \geqslant 0$. The algorithm defines a sequence $x_1, x_2, \ldots \in \mathbb{A}$, until the stop criterion is reached. One goes from x_k to x_{k+1} as follows :

1. Computation of $f'(x)$
2. Stop criterion : if $\|f'(x_k)\| \leqslant \varepsilon$ then stop else :
3. Coputation of the step $\alpha_k > 0$ with a linear search on f in x_k along $f'(x_k)$
4. $x_{k+1} = x_k - \alpha_k f'(x_k)$

Plan

1. Introduction to Optimization
2. Deterministic Methods
   - Gradient descent
   - Second Order Methods
3. Stochastic Optimization : some metaheuristics
   - Simulated Annealing
   - Evolutionary Algorithms
     - Genetic Algorithms and Evolutionary Strategies
     - Differential Evolution
     - Particle Swarm Optimization

Second Order Methods

One can accelerate the descent using information on the second derivative of f.

One has : $f(y) = f(x) + f'(x)(y-x) + 1/2f''(x)(y-x)^2 + r$
One then look for x^* such that
$f'(x)(x^* - x) + 1/2f''(x)(x^* - x)^2 = 0$, in other words,
$x^* = x - f'(x)/f''(x)$

The methods using this information are called local deterministic ***second order*** methods

Second Order Methods : Newton

Algorithm :

1. Computation of $f'(x)$ and $f''(x)$
2. Stop criterion : if $\|f'(x_k)/f''(x_k)\| \leqslant \varepsilon$, then stop, else :
3. $x_{k+1} = x_k - \alpha\, f'(x_k)/f''(x_k)$

Avdantages and Drawbacks of Deterministic Methods

- **Advantages**
  - Work well for ***continuous, differentiable*** and ***convex*** functions.
    $\Rightarrow$ local search

Avdantages and Drawbacks of Deterministic Methods

- **Advantages**
  - Work well for ***continuous, differentiable*** and ***convex*** functions.
    $\Rightarrow$ local search
  - ***Low complexity*** : low computation cost

Avdantages and Drawbacks of Deterministic Methods

- **Advantages**
  - Work well for ***continuous, differentiable*** and ***convex*** functions.
    $\Rightarrow$ local search
  - ***Low complexity*** : low computation cost
- **Drawbacks**
  - Does not work with non differentiable functions

Avdantages and Drawbacks of Deterministic Methods

- **Advantages**
  - Work well for ***continuous, differentiable*** and ***convex*** functions.
    $\Rightarrow$ local search
  - ***Low complexity*** : low computation cost
- **Drawbacks**
  - Does not work with non differentiable functions
  - Does not (generally) allow to find global extrema

Plan

1. Introduction to Optimization
2. Deterministic Methods
   - Gradient descent
   - Second Order Methods
3. Stochastic Optimization : some metaheuristics
   - Simulated Annealing
   - Evolutionary Algorithms
     - Genetic Algorithms and Evolutionary Strategies
     - Differential Evolution
     - Particle Swarm Optimization

Definition

Stochastic Optimization :

Definition

Stochastic Optimization :
optimizatiopn methods involving randomness in the optimum search.

Main Principles of Stochastic Optimization

Let F_{fit} be a cost function

1. Initialization : one randomly chooses one or several x_0 in the search space
2. While (stop criterion not reached)
    1. a strategy allows us to choose one or several x_i
    2. one compares $F_{fit}(x_i)$ to $F_{fit}(x_0)$ and one chooses the best (x_i or x_0)

Advantages and Drawbacks of Stochastic Methods

- **Advantages**
  - Works well for any type, non necessarily convex, of function (theoretically)

Advantages and Drawbacks of Stochastic Methods

- **Advantages**
  - Works well for any type, non necessarily convex, of function (theoretically)
  - Allows one to do ***global search*** (not only local)

Advantages and Drawbacks of Stochastic Methods

- **Advantages**
  - Works well for any type, non necessarily convex, of function (theoretically)
  - Allows one to do ***global search*** (not only local)
- **Drawbacks**
  - Do not guarantee a convergence

Advantages and Drawbacks of Stochastic Methods

- **Advantages**
  - Works well for any type, non necessarily convex, of function (theoretically)
  - Allows one to do ***global search*** (not only local)
- **Drawbacks**
  - Do not guarantee a convergence
  - Computational complexity

Definitions

Heuristic : $(\varepsilon\upsilon\rho\iota\sigma\kappa\omega)$

any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals. Where finding an optimal solution is impossible or impractical, heuristic methods can be used to speed up the process of finding a satisfactory solution.
Wikipedia Feb. 2017

Definitions

Heuristic : $(\varepsilon\upsilon\rho\iota\sigma\kappa\omega)$

any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals. Where finding an optimal solution is impossible or impractical, heuristic methods can be used to speed up the process of finding a satisfactory solution. Wikipedia Feb. 2017

Metaheuristic :

stochastic optimization algorithm used to solve complex problems. They are usually inspired from natural systems, from physics or biology. Many metaheuristics implement some form of stochastic optimization, so that the solution found is dependent on the set of random variables generated.

Context

One tries to minimize a function $f : D \subset \mathbb{R}^K \to \mathbb{R}, \quad K \in \mathbb{N}.$

Plan

1 Introduction to Optimization

2 Deterministic Methods
- Gradient descent
- Second Order Methods

3 Stochastic Optimization : some metaheuristics
- Simulated Annealing
- Evolutionary Algorithms
  - Genetic Algorithms and Evolutionary Strategies
  - Differential Evolution
  - Particle Swarm Optimization

Simulated Annealing : inspiration form metallurgy

The simulated annealing is inspired from annealing in metallurgy

Principle :

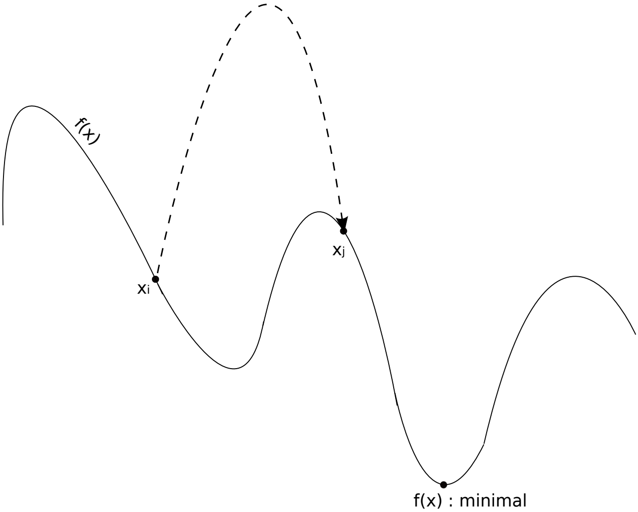
- improving the metal quality by reaching the minimal energy state corresponding to a stable chemical structure of the metal
- the metal is heated at high temperature (until liquid) then cooled in a controlled manner until it becomes solid again

Simulated annealing interprets slow cooling as a slow decrease in the probability of accepting worse solutions as it explores the solution space.

Simulated Annealing (Kirkpatrick 1983) : principle

Like in metallurgy :

- one wants our system to converge to a stable energy state, i.e. find the parameter x so that the function reaches its minimum value atteint son minimum
- one fixes a high value for the initial temperature parameter T, then we make it decrease at each iteration n or according to fixed stages. Then one randomly draw a parameter x_n that one compares to x_{n-1} ; one keeps it with a probability that depends on the temperature : when T is high, one authorize a "bad" x_n with the probability $e^{(-\Delta(F_{fit})/T)} > \mathscr{U}(0,1)$, and the more T is low, the lower the probability to keep a bad x_n, and then the more local the search is


f(x)
xj
xi
f(x) : minimal

Algorithm 1: Simulated Annealing Algorithm

Entrées: f : cost function, T_{init} : initial temperature, T_{fin} : final temperature, N_{iter} : number of iterations per stage of temperature, h : temperature function.

Sorties: x_{opt}, minimizing f.

Initialization : x ra,dom, $x_{opt} = x$, $f_{opt} = f(x_{opt})$, $T = T_{init}$, $k = 0$;

tant que $T > T_{fin}$ **faire**
- **tant que** $k < N_{iter}$ **faire**
  - Choose y in the neighborhood of x;
  - compute $\Delta f = f(y) - f(x)$;
  - **si** $\Delta f < 0$ **alors**
    - x=y;
    - **si** $f(x) < f(x_{opt})$ **alors**
      - $x_{opt} = x$;
    - $f_{opt} = f(x_{opt})$;
  - **sinon**
    - Draw randomly p in [0,1];
    - **si** $p \leq exp(-\Delta f / T)$ **alors**
      - x = y;
  - $k = k + 1$;
- $T = h(T)$;
- return x_{opt};

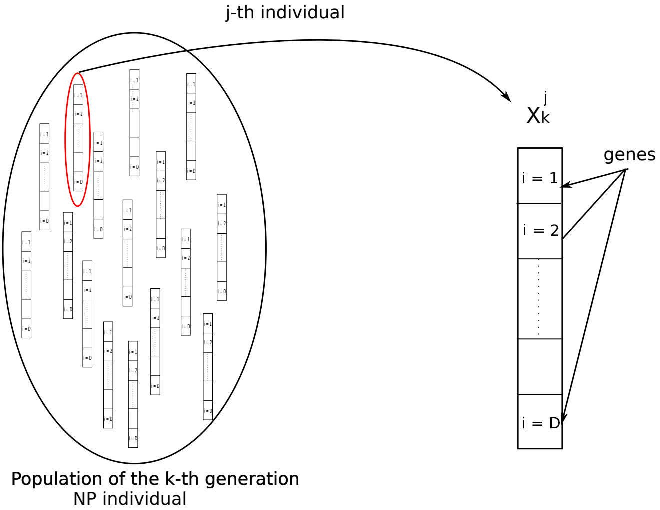
Simulated Annealing : choosing the metaparameters

How to choose T_{init}, T_{fin}, h, N_{iter} and the length of the temperature stages ?

Plan

1. Introduction to Optimization
2. Deterministic Methods
   - Gradient descent
   - Second Order Methods
3. Stochastic Optimization : some metaheuristics
   - Simulated Annealing
   - Evolutionary Algorithms
     - Genetic Algorithms and Evolutionary Strategies
     - Differential Evolution
     - Particle Swarm Optimization

Vocabulary of Evolutionary Algorithms



Two Coding Types

- binary
  gene coded by 0 ou 1 $\rightarrow$ genetic algorithmes (US)
- real
  gene coded by a real number $\rightarrow$ evolutionary strategies (Germany)

Two Coding Types

- binary
  gene coded by 0 ou 1 $\rightarrow$ genetic algorithmes (US)
- **real**
  gene coded by a real number $\rightarrow$ evolutionary strategies (Germany)

Principle

- Crossover

Principle

- Crossover
- Mutation

Principle

- Crossover
- Mutation
- Selection

Principle

- Crossover
- Mutation
- Selection
- Replacement

Crossover

Applied to the selection operator on a population P

It allows us to generate a population P' of $NP/2$ individuals by recombining the genes (parameters) of two parents with a probability p_c.

To complete the new population, one can keep half of the parents by randomly picking them in the initial population.

Mutation

To avoid degeneration : “get out” of local minima

Consists in changing a gene value with a probability p_m usually low ($p_m \in [0,001; 0,5]$)

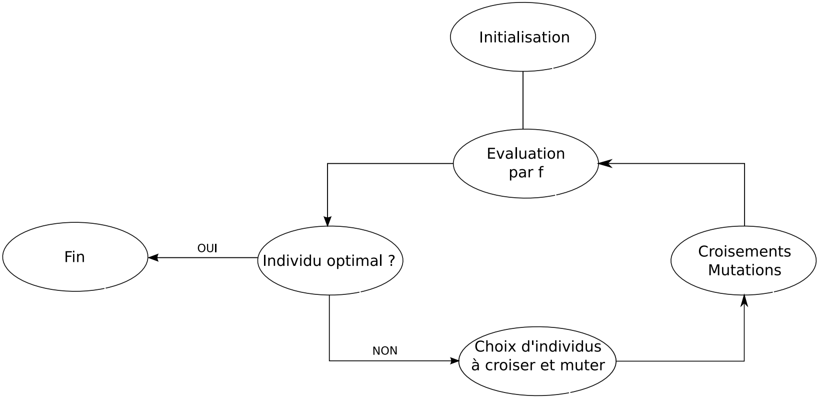
Tournament selection

Draw of two individuals. The one having the lowest cost function wins with a probability $p \in [0,5;1]$.

One repetes this process n times to get n individuals that will serve as parents.

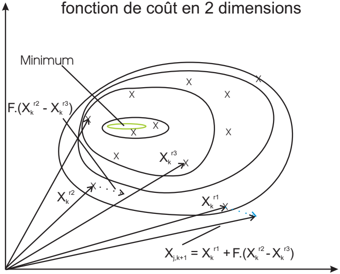
Genetic Algorithms and Evolutionary Strategy : how to choose the metaparameters ?

Evolutionary Strategy : summary



Differential Evolution Algorithm (ED)

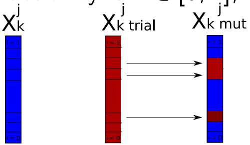
Processus de différenciation dans le cas d 'une fonction de coût en 2 dimensions



NP individuals

Differentiation : $\forall j = 1, \ldots, NP$,
$X^j_{k,trial} = X^{r_1}_k + F.(X^{r_2}_k - X^{r_3}_k)$,
F usually equal to $0, 5$.

Recombination : with a probability $CR \in [0; 1]$, usually $CR = 0, 9$.



Selection :

$$X^j_{k+1} = \begin{cases} X^j_{k,mut} & \text{if } F_{fit}(X^j_{k,mut}) \leq F_{fit}(X^j_k) \\ X^j_k & \text{otherwise} \end{cases}$$

Algorithm 2: Differential Evolution Algorithm (ED)

Entrées: NP : number of individuals, N_{iter} : number of iterations, F_{cost} : cost function, F : differentiation coefficient, CR : recombination constant.
Sorties: x_{opt}, minimizing F_{cost}.
Initialization : X radom uniform in D^{NP}, x_{opt}.;
pour k *de* 0 *à* $(N_{iter} - 1)$ **faire**

1) Differentiation :
the $r^{\text{ème}}$ new potential parameter vector, $X^r_{k,trial}$, is generated by adding $X^{r_1}_k$, randomly drawn amongst the individuals of the k^{th} generation, and $X^{r_2}_k$ et $X^{r_3}_k$, with $r_1 \neq r_2 \neq r_3$, thus :

$$\forall r = 1, \dots, NP, \quad X^r_{k,trial} = \min(\max(X^{r_1}_k + F.(X^{r_2}_k - X^{r_3}_k), X_{min}), X_{max}),$$

2) Recombination :
The r^{th} "mutant" individual of the k^{the} generation, $X^r_{k,mut}$, heritates the genes of $X^r_{k,trial}$ with a probability CR specific to each gene, i.e. one generate $u = \mathscr{U}(0,1)$ and :

$$\forall i = 1, \dots, K, \qquad \forall r = 1, .., NP,$$

$$X^r_{k,mut}(i) = \begin{cases} X^r_{k,trial}(i) & \text{if } u < CR \\ X^r_k(i) & \text{otherw.} \end{cases}$$

3) Selection :

$$X^r_{k+1} = \begin{cases} X^r_{k,mut} & \text{if } F_{\text{cost}}(X^r_{k,mut}) \leq F_{\text{cost}}(X^r_k) \\ X^r_k & \text{otherw.} \end{cases}$$

$x_{opt} = x \in \{X^1_k, X^2_k, ..., X^{NP}_k\}$ t.q. $\forall X^i_k \in \{X^1_k, X^2_k, ..., X^{NP}_k\}, F_{\text{cost}}(x) \leq F(X^i_k)$;

Particle Swarm Optimization (R. Eberhart et J. Kennedy 1995)

inspired from animal swarms (bird's fly)

Particle Swarm Optimization (R. Eberhart et J. Kennedy 1995)

inspired from animal swarms (bird's fly)

Main Principle

Each iteration makes individuals (particles) move along three components :

- its current speed $v(t)$
- the best particle up to now, $p(t)$
- the best particle of the generation, created at time t, $g(t)$
  The equations ruling the movement are thus :

$$v(t+1) = \omega v(t) + \phi_1(p(t) - X(t)) + \phi_2(g(t) - x(t))$$
$$x(t+1) = x(t) + v(t+1)$$

ω inertia, ϕ_1 ϕ_2 metaparameters

Particle Swarm Optimization : choosing the metaparameters

ω, ϕ_1, ϕ_2 make the particle speed vary.

Usually, one chooses $\omega < 1$.

ω close to $1 =$ slow convergence, but exhaustive exploration of the search space

Algorithm 3: Particle Swarm Optimization.

Entrées: N_{iter} : number of iterations, F_{cost} : cost function, N : number of particles.
Sorties: x_{opt}, minimizing F_{cost}.
Initialization : $X = X_0$ random uniform in D^N, $v_i(0), i = 1, 2, ..., N$, random uniform, $x_{opt} = x \in \{X_t^1, X_t^2, ..., X_t^N\}$ t.q. $\forall X_t^i \in \{X_t^1, X_t^2, ..., X_t^N\}, F_{\text{cost}}(x) \leq X_t^i$, $p = x_{opt}$.;
pour *t from* 0 *to* $(N_{iter} - 1)$ **faire**
  pour *i from* 1 *to N* **faire**
    $V_t^i = \omega . V_t^i + \phi_1 . \mathscr{U}(0,1).(p(t) - x) + \phi_2 . \mathscr{U}(0,1).(g(t) - X_t^i)$;
    $X_t^i = X_t^i + V_t^i$
  $x_{opt} = x \in \{g(t), X_t^1, X_t^2, ..., X_t^N\}$ t.q. $\forall X_t^i \in \{g(t), X_t^1, X_t^2, ..., X_t^N\}, F_{\text{cost}}(x) \leq F(X_t^i)$;
  $g(t) = x_{opt}$;
return
$p(t) = x_{opt} = x \in \{X^1, X^2, ..., X^N\}$ s.t. $\forall X^i \in \{X^1, X^2, ..., X^N\}, F_{\text{cost}}(x) \leq F(X^i)$;