Global Optimization

E.FAUVET

eric.fauvet@u-bourgogne.fr

Summary

- Introduction
- Definitions
  - Optimization
  - Optimum
  - Problems in optimization
- Optimization methods:
  - Hill climbing
  - Simulated annealing
  - Genetic algorithm
  - Ant colony optimization
  - Particle swarm optimization
  - Tabu search
- Examples of works in the Le2i laboratory

Bibliography

- This presentation is based on the e-book of Thomas Weise. You can find it at the following address:
- http://www.it-weise.de/
- It contains more than 2000 references about optimization
- Explains several methods in detail
- The currently encountered applications are described for each method.

Introduction

- One of the most important principle in the nature:
- Physics, biology, economy
- Optimization is one of the oldest science.
- Part of numerical analysis and applied mathematics

- Introduction
- **Definitions**
- Optimization methods:
  - Hill climbing
  - Simulated annealing
  - Genetic algorithm
  - Ant colony optimization
  - Particle swarm optimization
- Examples of works in the Le2i laboratory

Definition of optimization

- Mathematical definition: The goal of global optimization is to find the best possible elements $x^\star$ from a set X according to a set of criteria $F = \{f_1, f_2, .., f_n\}$.
- These criteria are expressed as mathematical functions, called objective functions.
- Definition: (Objective Function).
- An objective function $f : X \rightarrow Y$ with $Y \subseteq R$ is a mathematical function which is subject to optimization.

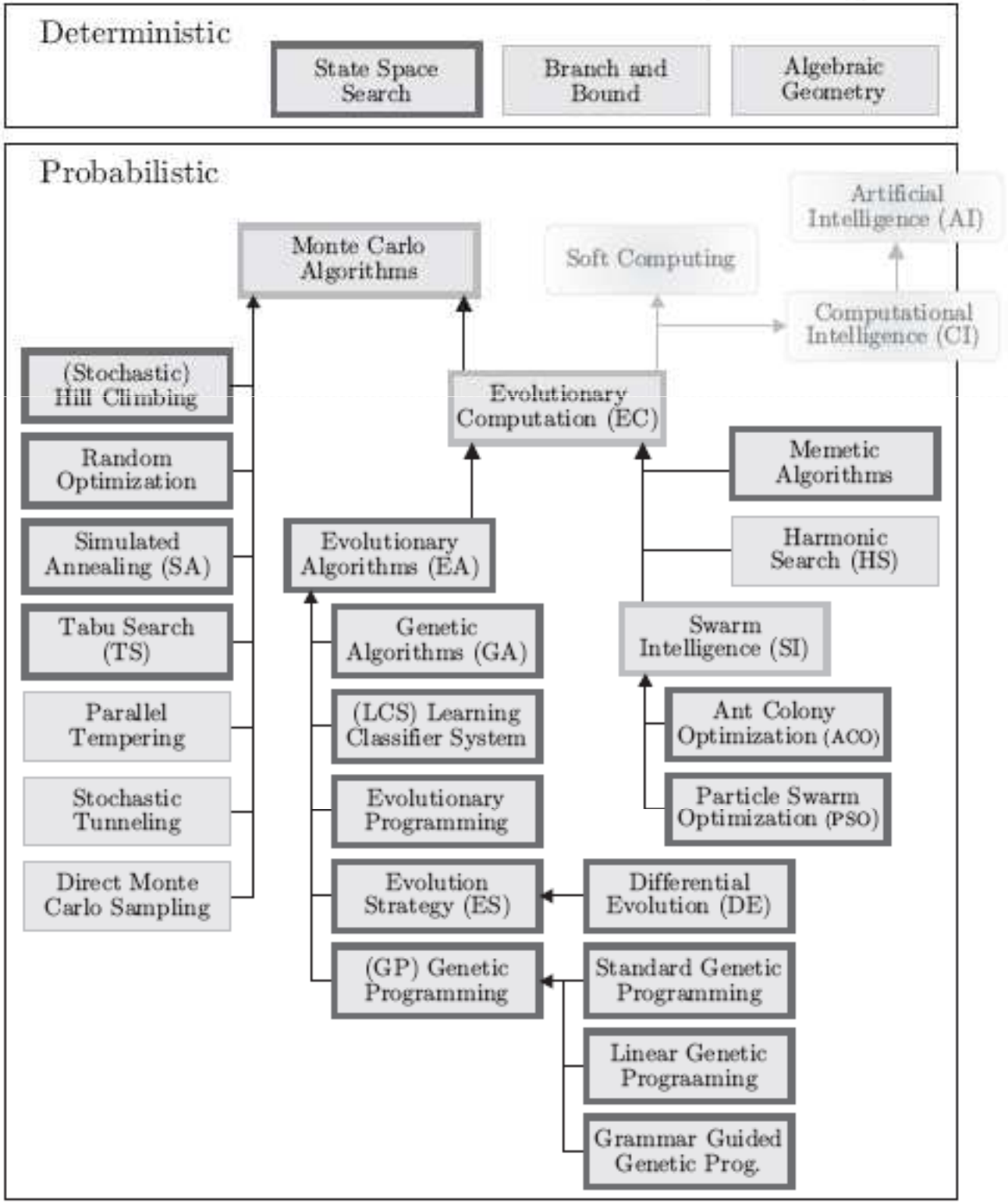
Classification of optimization algorithms

- Two classes of methods:
  - Deterministic: used if there exists a clear relation between the characteristics of the possible solutions and their utility for a given problem.
  - Probabilitic: used if the problem is too complicated
    - Beginning 60 years ago with the Monte-Carlo Algorithm
    - The result may not be the global optima
    - But a solution a little bit inferior to the best possible one is better than one which needs 10^{100} years to be found. . .

Heuristic

- A heuristic is a part of an optimization algorithm that uses the information currently gathered by the algorithm to help to decide which candidate solution should be tested next or how the next individual can be produced.
- Heuristics are usually class dependent problems.

definitions



Metaheuristic

- A metaheuristic is a heuristic method for solving a very general class of problems. It combines objective functions or heuristics in an abstract and hopefully efficient way, usually without utilizing deeper insight into their structure, by treating them as black-box-procedures.
- For example: Simulated annealing decides which solution candidate to be evaluated next according to the Boltzmann probability factor of atom configurations of solidifying metals in fusion.

Speed-accuracy compromise

- Speed and precision are conflicting objectives, at least in terms of probabilistic algorithms. A general rule of thumb is that you can gain improvements in accuracy of optimization only by investing more time. Scientists in the area of global optimization try to push this Pareto frontier further by inventing new approaches and enhancing or tweaking existing ones.

Time constraints

- **Online Optimization:** Online optimization problems are tasks that need to be solved quickly in a time span between ten milliseconds to a few minutes. In order to find a solution in this short time, optimality is normally traded in for speed gains.
- Examples: robot localization, load balancing, services composition for business processes, or updating a factory's machine job schedule after new orders come in.

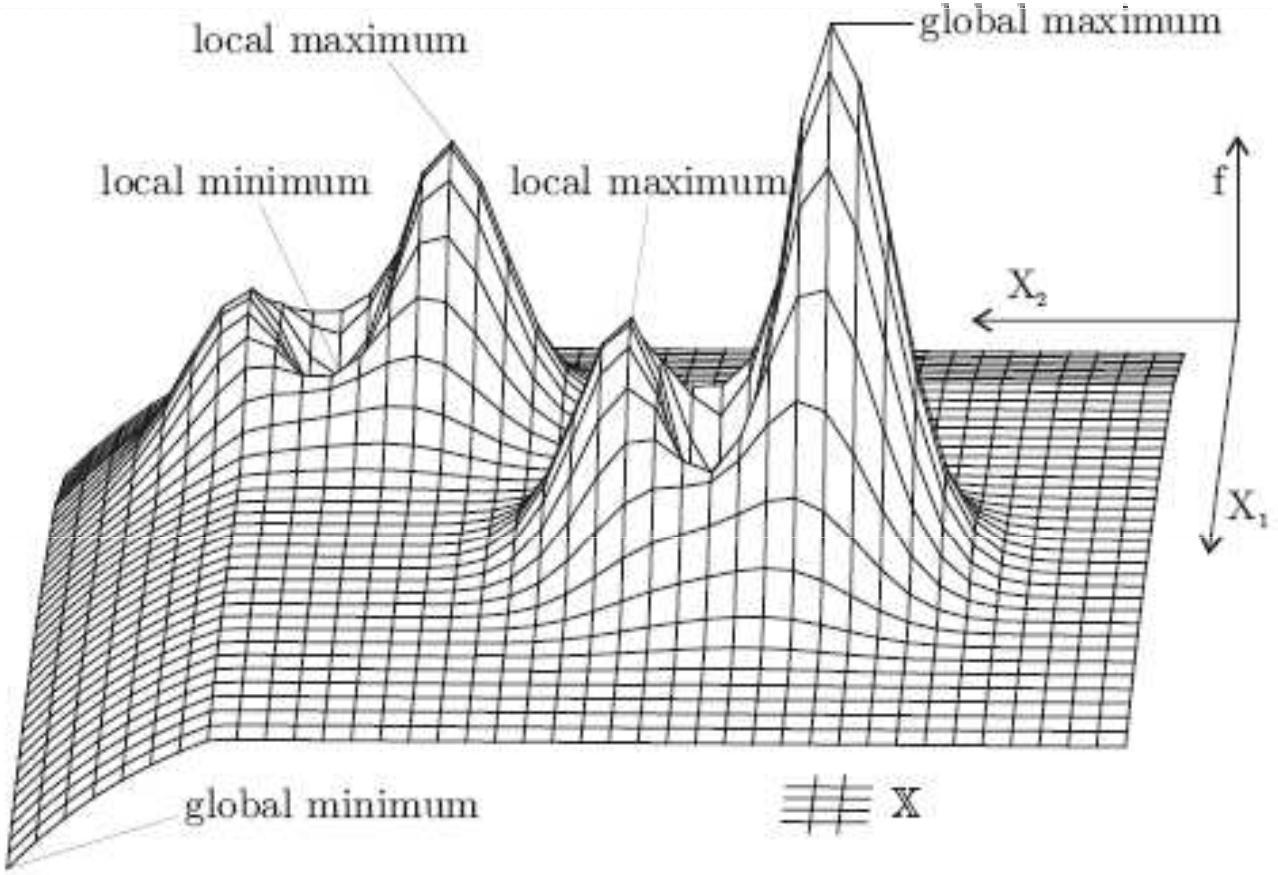
Time constraints

- **Offline Optimization:** In offline optimization problems, time is not so important and a user is willing to wait maybe even days if she can get an optimal or close-to-optimal result.
- Example : Such problems regard for example design optimization, data mining or creating long-term schedules for transportation crews. These optimization processes will usually be carried out only once in a long time.

Single objective functions

- In the case of optimizing a single criterion f, an optimum is either its maximum or minimum, depending on what we are looking for.
- In global optimization, it is a convention that optimization problems are most often defined as minimizations and if a criterion f is subject to maximization, we simply minimize its negation (–f).

Optima


local maximum
global maximum
local minimum
local maximum
f
X2
X1
global minimum
X

Definitions: What's an optimum

- Local Maximum: A (local) maximum $\hat{x}_l \in X$ of one (objective) function $f : X \to \mathbb{R}$ is an input element with $f\left(\hat{x}_l\right) \geq f\left(x\right)$ for all x neighboring $\hat{x}_1$.
- If $X \subseteq \mathbb{R}^n$, we can write:

  $$\forall \hat{x}_1 \quad \exists \varepsilon > 0 : f\left(\hat{x}_1\right) \geq f\left(x\right) \quad \forall x \in X, \left|x - \hat{x}_1\right| < \varepsilon$$

- Global Maximum: A global maximum $\hat{x} \in X$ of one (objective) function $f : X \to \mathbb{R}$ is an input element with

  $$f\left(\hat{x}\right) \geq f\left(x\right) \quad \forall x \in X$$

- Optimal Set: The optimal set $X^\star$ is the set that contains all optimal elements.

Multiple objective functions

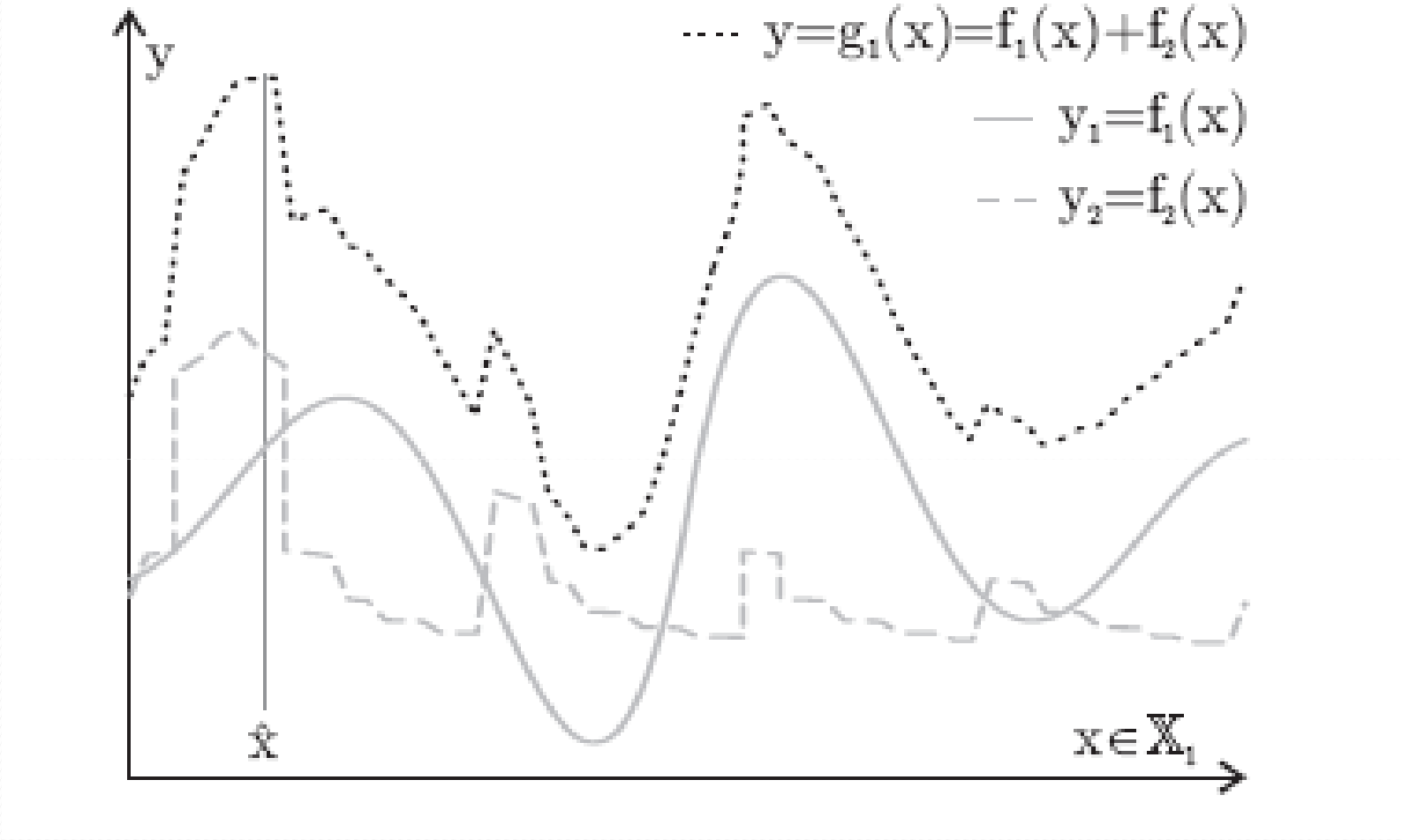
- Global optimization techniques are not just used for fonding the maxima or minima of single functions.
- They are rather applied to set F consisting of n objective functions f_i , each representing one criterion to be optimized

$$F = \{ f_i : X \rightarrow Y_i : 0 < i \leq n, Y_i \subseteq \mathbb{R} \}$$

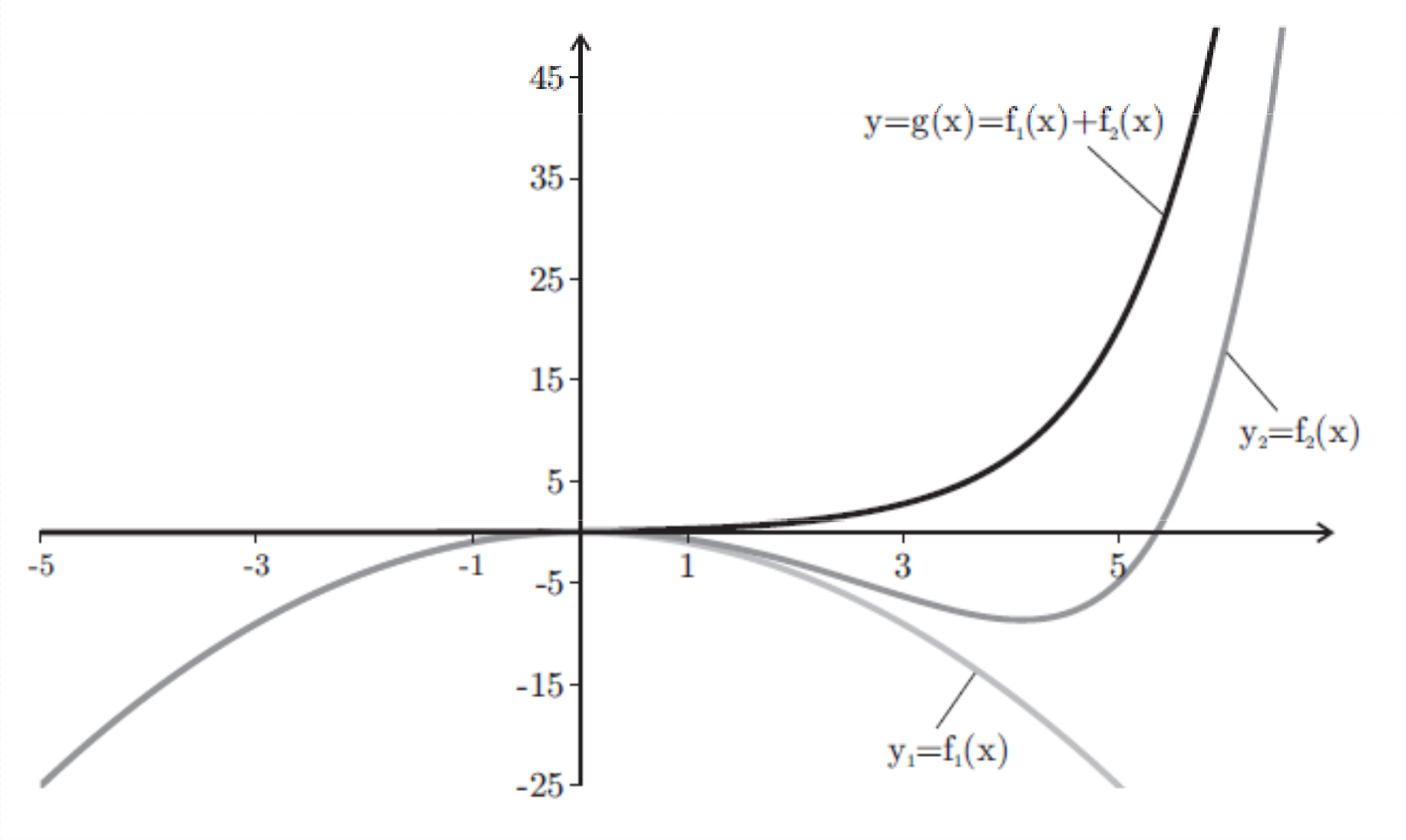
Examples

- Multi-objective optimization often means to compromise confliting goals.
- Taking into account the example of a factory the following objectives are subject to optimization:
  - Minimize time between an incoming order and the product delivery,
  - Maximize profit
  - Maximize product quality,
  - Minimize costs for advertinsing, personal…
  - Minimize negative impact on environment
- The mutual influences between objectives can become complicated and the solutions not always obvious.

Definition of an optimum



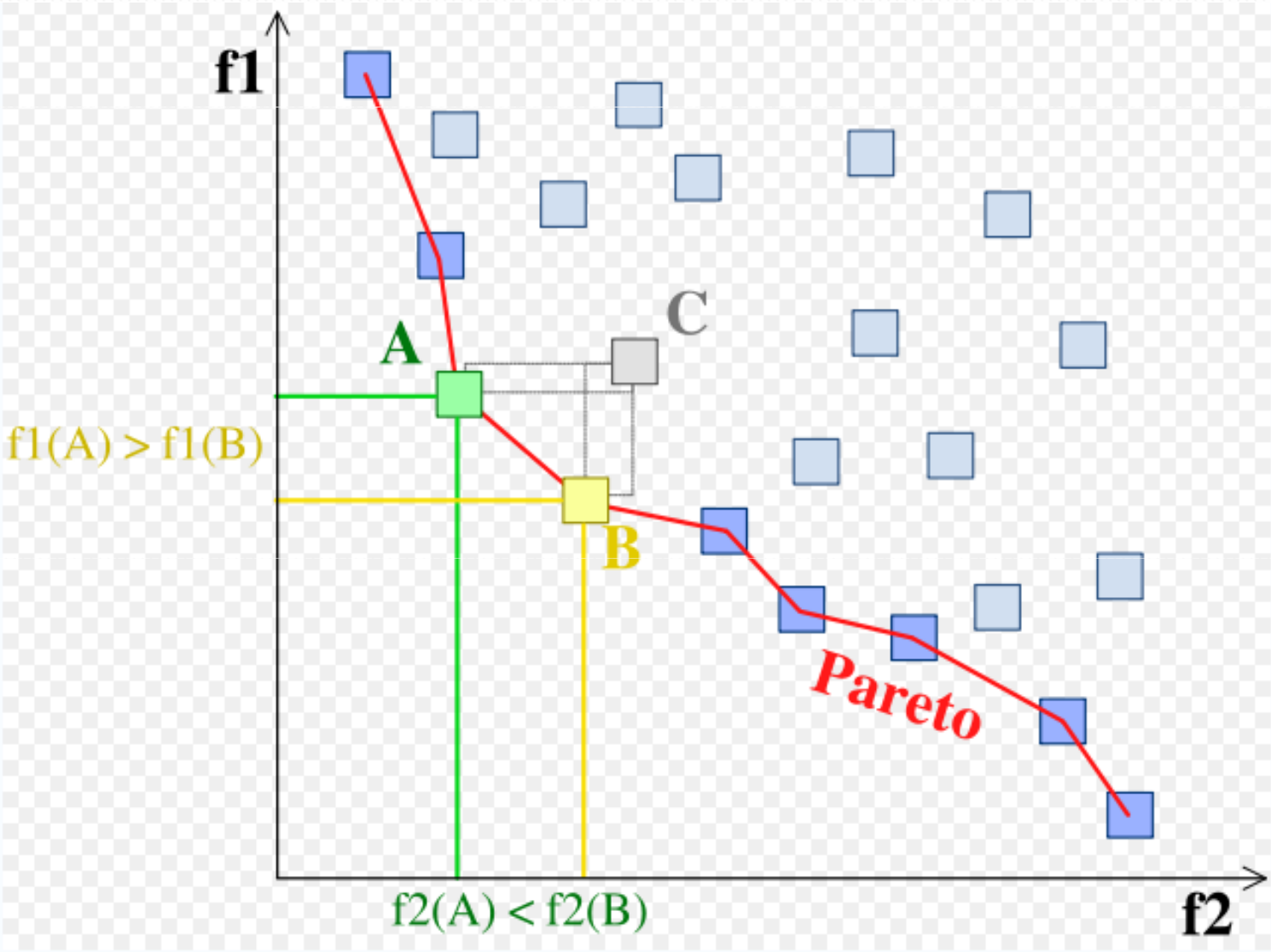
Problems with weighted sums



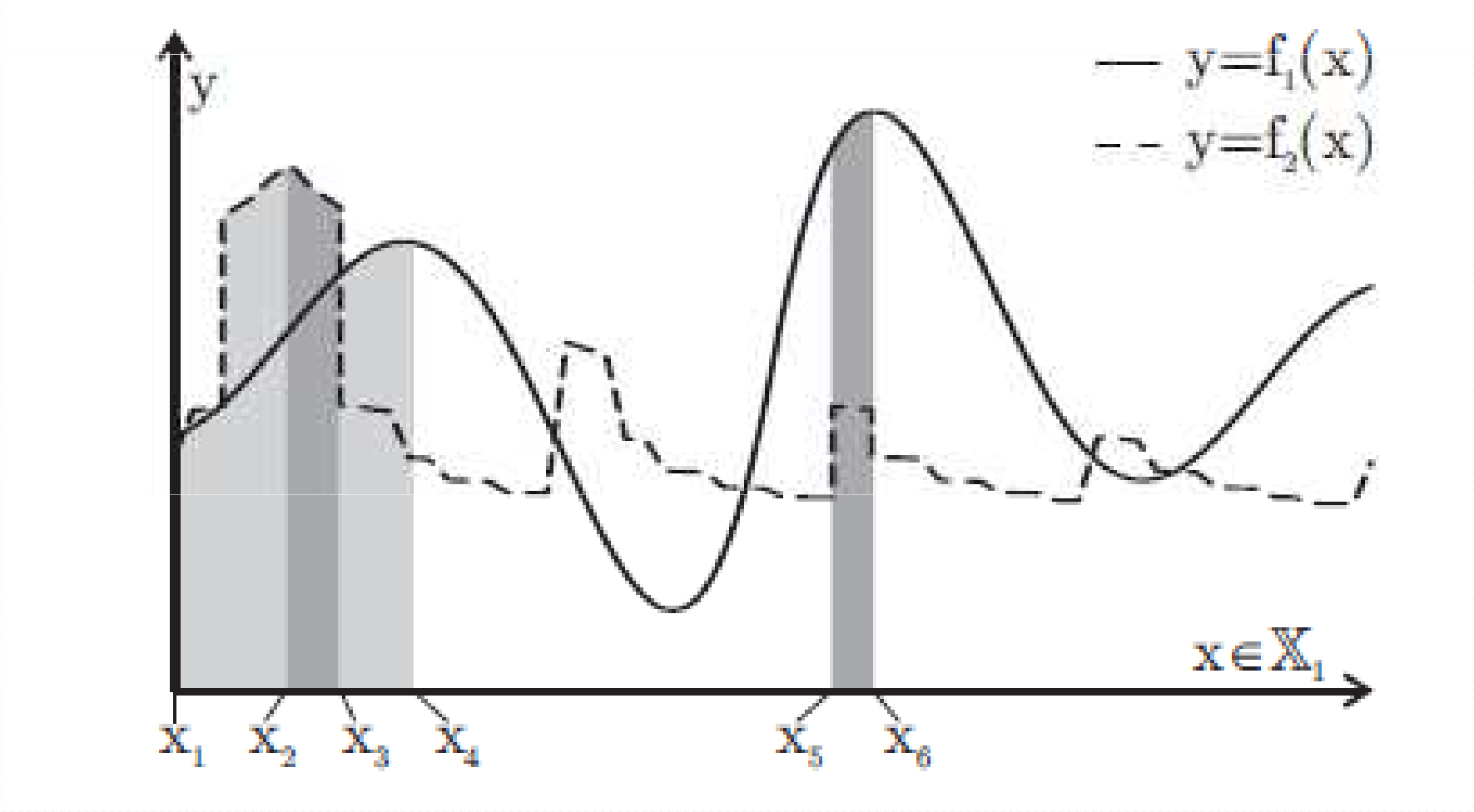
Wilfredo Pareto

- The **Pareto distribution**, named after the Italian economist Wilfredo Pareto, is a power law probability distribution that coincides with social, scientific, geophysical, actuarial, and many other types of observable phenomena.
- Pareto originally used this distribution to describe the allocation of wealth among individuals since it seemed to show rather well the way that a larger portion of the wealth of any society is owned by a smaller percentage of the people in that society. This idea is sometimes expressed more simply as the Pareto principle or the "80-20 rule" which says that 20% of the population controls 80% of the wealth.

Pareto optimization



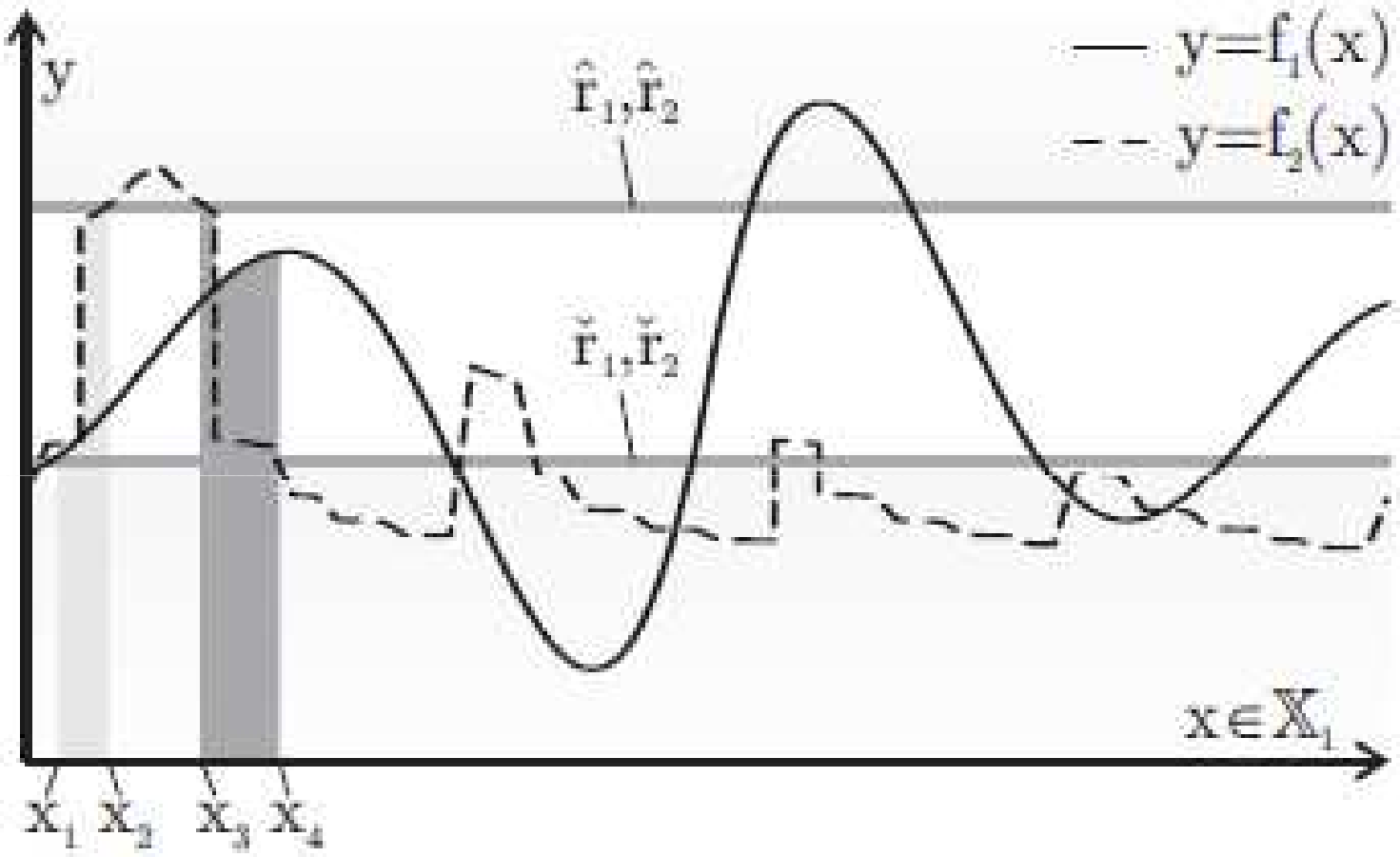
Pareto optimality on the example



problem

- The complete Pareto optimal set is often not the wanted result of an optimization algorithm.
- General inequality constraints can also be processed according to the method of inequalities, in this method, an area of interest is specified in form of a goal range for each objective function.
- This approach can be combined with Pareto optimisation.

Example


y
$\hat{r}_1, \hat{r}_2$
$\check{r}_1, \check{r}_2$
$y=f_1(x)$
$y=f_2(x)$
$x \in \mathbb{X}_1$
x_1
x_2
x_3
x_4

The structure of optimization

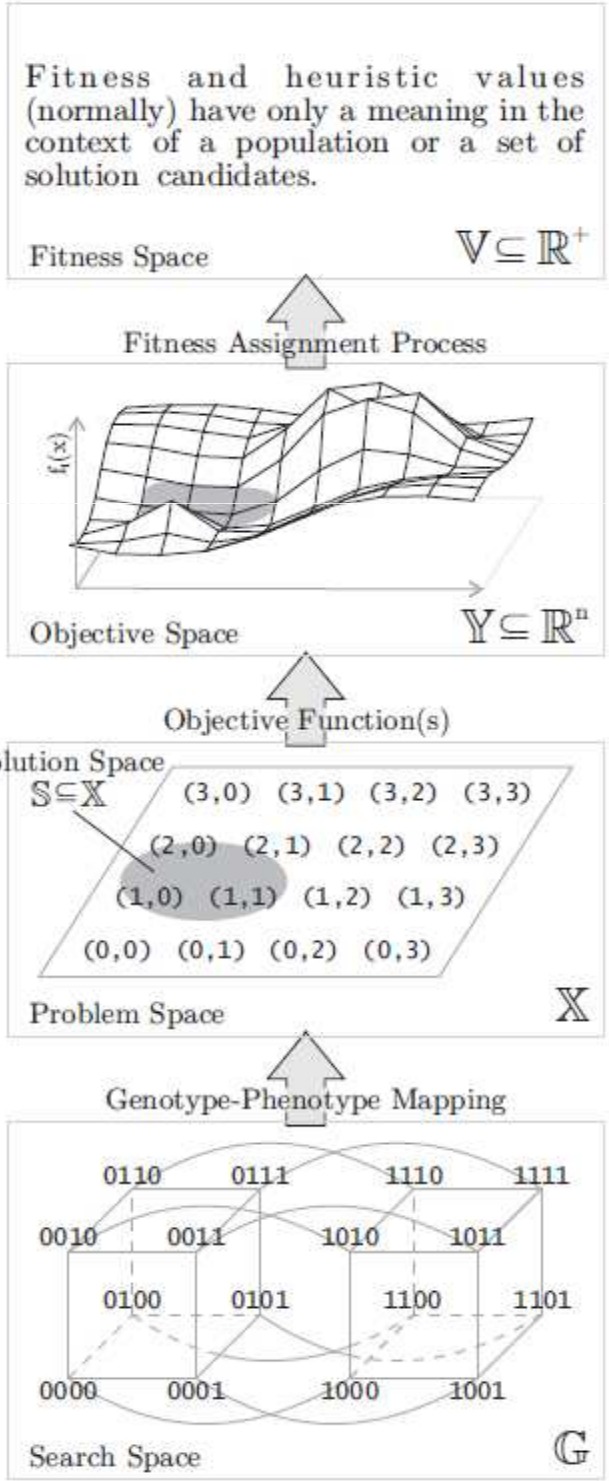
- Spaces, sets, and elements
- Fitness, landscapes and global optimization
- Gradient descent
- Other general features

Problem space, solution candidate

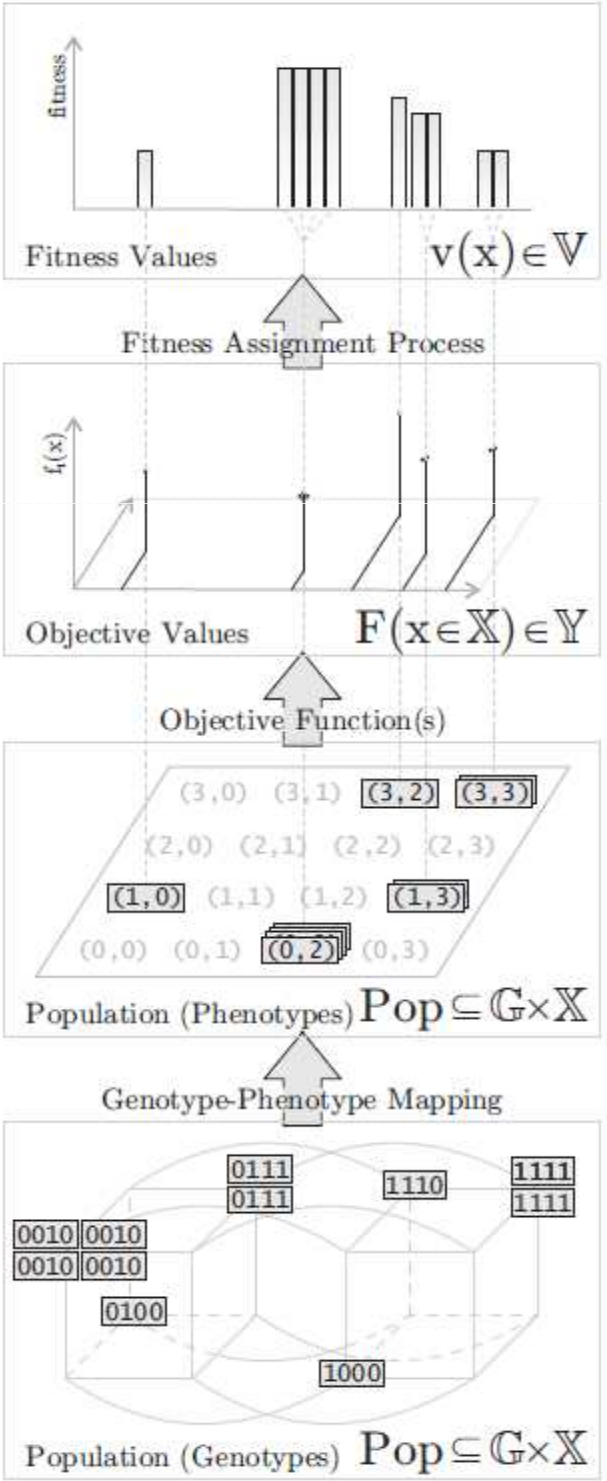
- The **problem space X** of an optimization problem is the set containing all elements x which could be its solution.
- A **candidate solution x** is an element of the problem space X of a certain optimization problem.
- We call the union of all solutions to an optimization problem its "**solution space**" **S**. This solution space contains (and can be equal to) the global optimal set $X^\star$. There may exist valid solutions $x \in S$ which are not elements of the $X^\star$ set.

Search space, …

- The **search space** G of an optimization problem is the set of elements g on which the search operations of an optimization algorithm will operate.
- The elements $g \in G$ of the search space G of a given problem are called the **genotypes**.
- The distinguishable units of information in a genotype that encode the phenotypical properties are called **genes**.
- An **allele** is a value of specific gene.
- The **locus** is the position where a specific gene can be found in a genotype.



The Involved Spaces



The Involved Sets/Elements

Search operations

- The result of a search operation is one element of the search space.
- We will encounter many different new operators for search. A unary search operation would, for example, be the mutation in genetic algorithms.
- Optimization processes are often initialized by creating random genotypes.
- Search operations often involve randomized numbers.
- A set Op of search operations searchOp is complete if and only if every point g1 in the search space G can be reached from every other point g2 ∈ G by applying only operations searchOp ∈ Op.

Connection between Search and Problem Space

- If the search space differs from the problem space, a translation between the two is further-more required. In our car example we would need to transform the binary strings processed by the genetic algorithm to objects which represent the corresponding car configurations and can be processed by the objective functions.

Fitness

- The fitness value $v(x) \in V$ of an element x of the problem space X corresponds to its utility as solution or its priority in the subsequent steps of the optimization process. The space spanned by all possible fitness values V is normally a subset of the positive real numbers $V \subseteq R+$.

Opitmization algorithm

- An optimization algorithm is characterized by
  - the way it assigns fitness to the individuals,
  - the ways it selects them for further investigation,
  - the way it applies the search operations, and the way it builds and treats its state information.
- Global optimization algorithms are optimization algorithms that employs measures that prevent convergence to local optima and increase the probability of finding a global optimum.

Gradient Descend

- A gradient of a scalar field $f : R^n \rightarrow R$ is a vector field which points into the direction of the greatest increase of the scalar field. It is denoted by ∇f or $\text{grad}(f)$.

Other general features

- There are some further common semantics and operations that are shared by most optimization algorithms. Many of them, for instance, start out by randomly creating some initial individuals which are then refined iteratively.
- Optimization processes which are not allowed to run infinitely have to find out when to terminate.
- In this section we define and discuss general abstractions for such commonalities.

Evaluation, iteration

- Evaluation: The value $\tau \in N_0$ denotes the number of solution candidates for which the set of objective functions F has been evaluated.
- An iteration refers to one round in a loop of an algorithm. It is one repetition of a specific sequence of instruction inside an algorithm.

Termination Criterion

- The termination criterion terminationCriterion() is a function with access to all the information accumulated by an optimization process, including the number of performed steps t, the objective values of the best individuals, and the time elapsed since the start of the process. With terminationCriterion(), the optimizers determine when they have to halt.
- When the termination criterion function terminationCriterion() $\in$ {true, false} evaluates to true, the optimization process will stop and return its results.

Examples of temination criterion

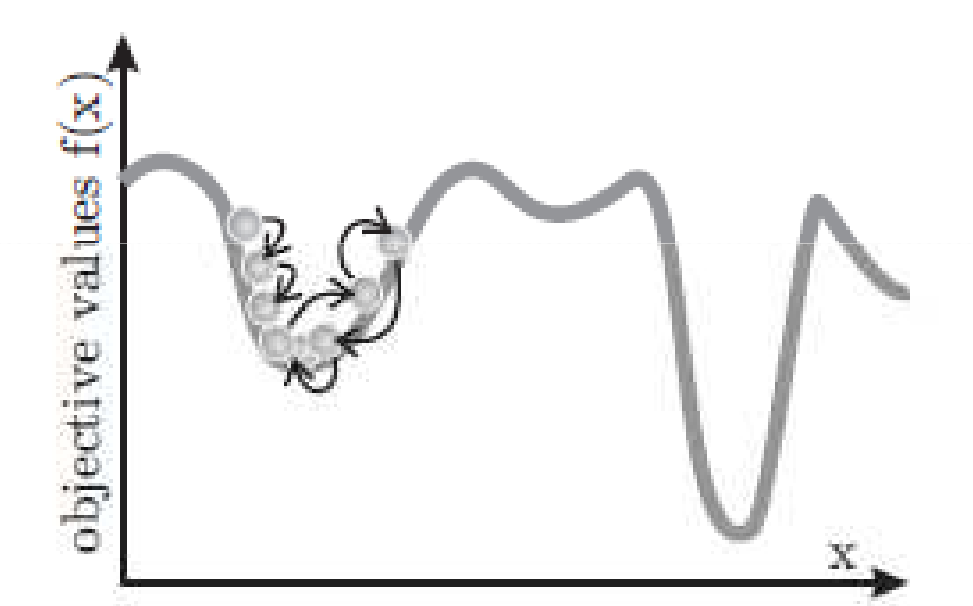
- Maximum computation time
- Number of iterations
- No more improvement during a number of iterations
- Sufficently good solution found
- Or a combination of the criteria above

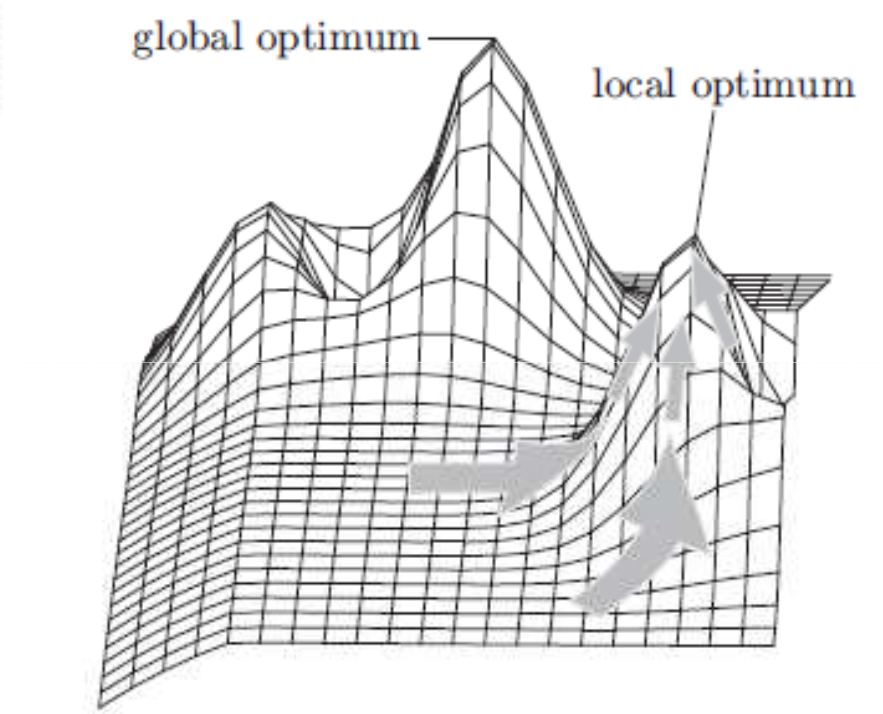
Problems in optimization

- Premature convergence
- Ruggedness, weak causality
- Deceptiveness
- Neutrality and redundancy
- Epitasis
- Noise and robustness

Premature convergence

- Convergence
- Premature convergence





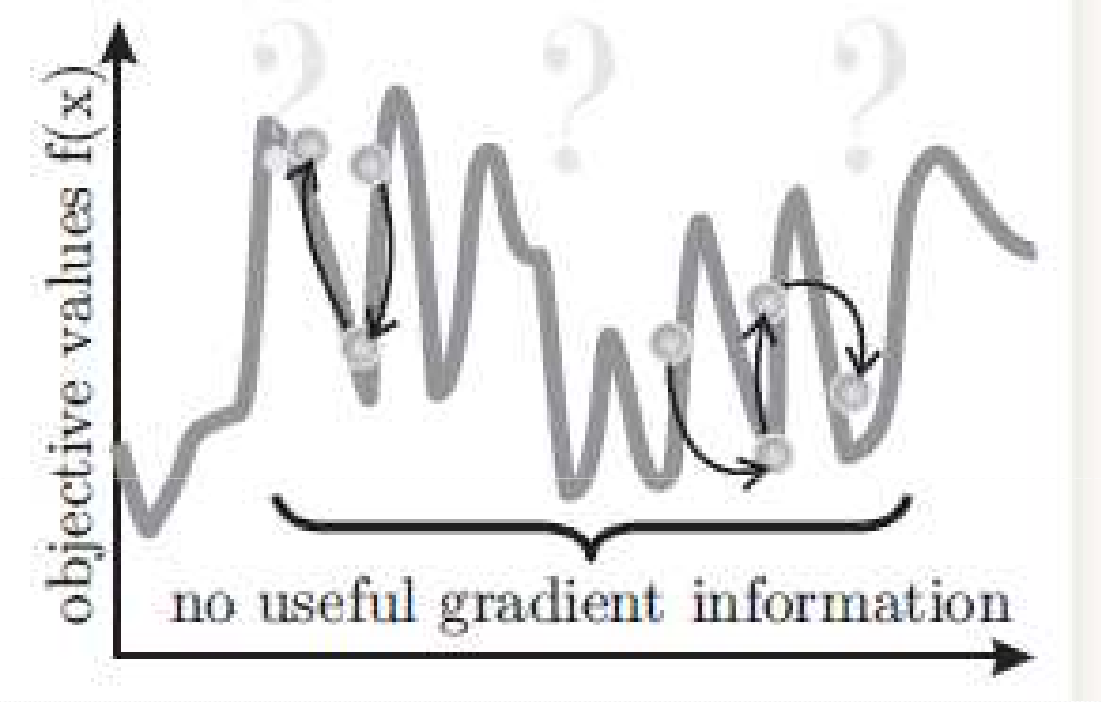
One Cause: Loss of Diversity

- In population-based global optimization algorithms, maintaining a set of diverse solution candidates is very important too.
- Losing diversity means approaching a state where all the solution candidates under investigation are similar to each other.
- Another term for this state is convergence.
- Preserving diversity is directly linked with maintaining a good balance between exploitation and exploration

Exploration, exploitation

- **Exploration** means finding new points in areas of the search space which have not been investigated before.
- **Exploration** is a metaphor for search operations which find totally new and maybe better solution structures.
- **Exploitation**, on the other hand, is the process of improving and combining the traits of the currently known solution(s).

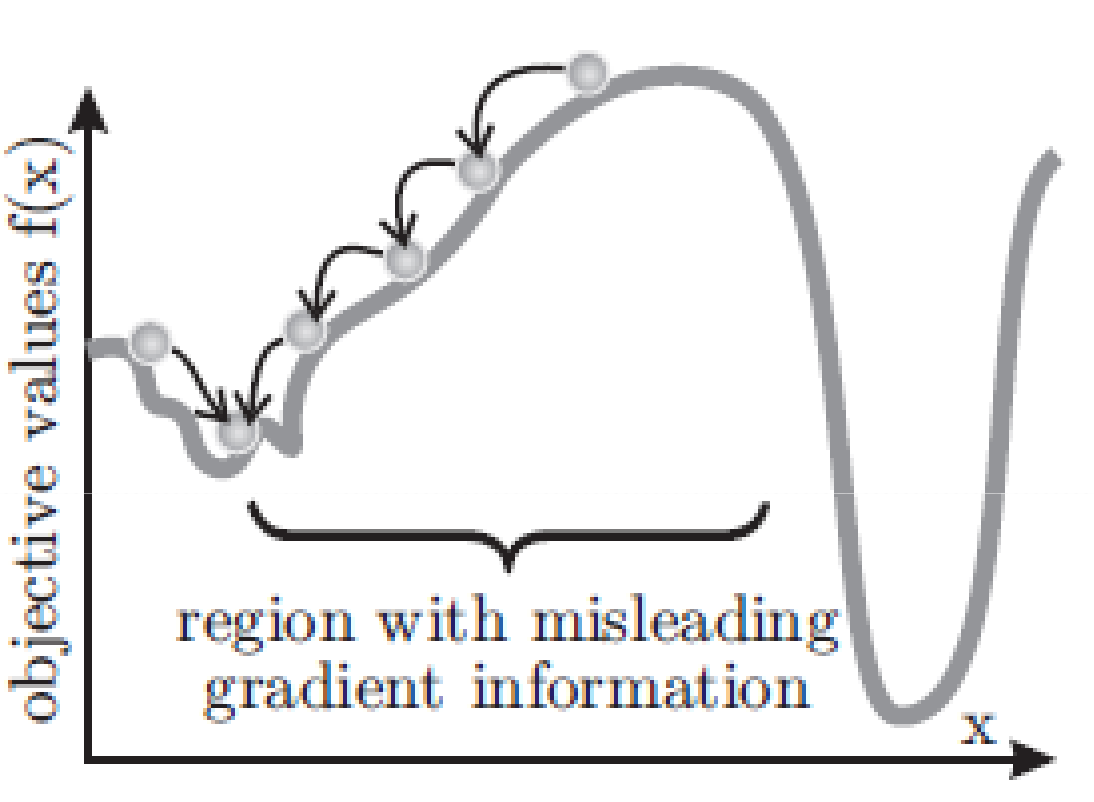
Ruggedness and Weak Causality



One Cause: Weak causality

- Strong causality (locality) means that small changes in an object also lead to small changes in its behavior

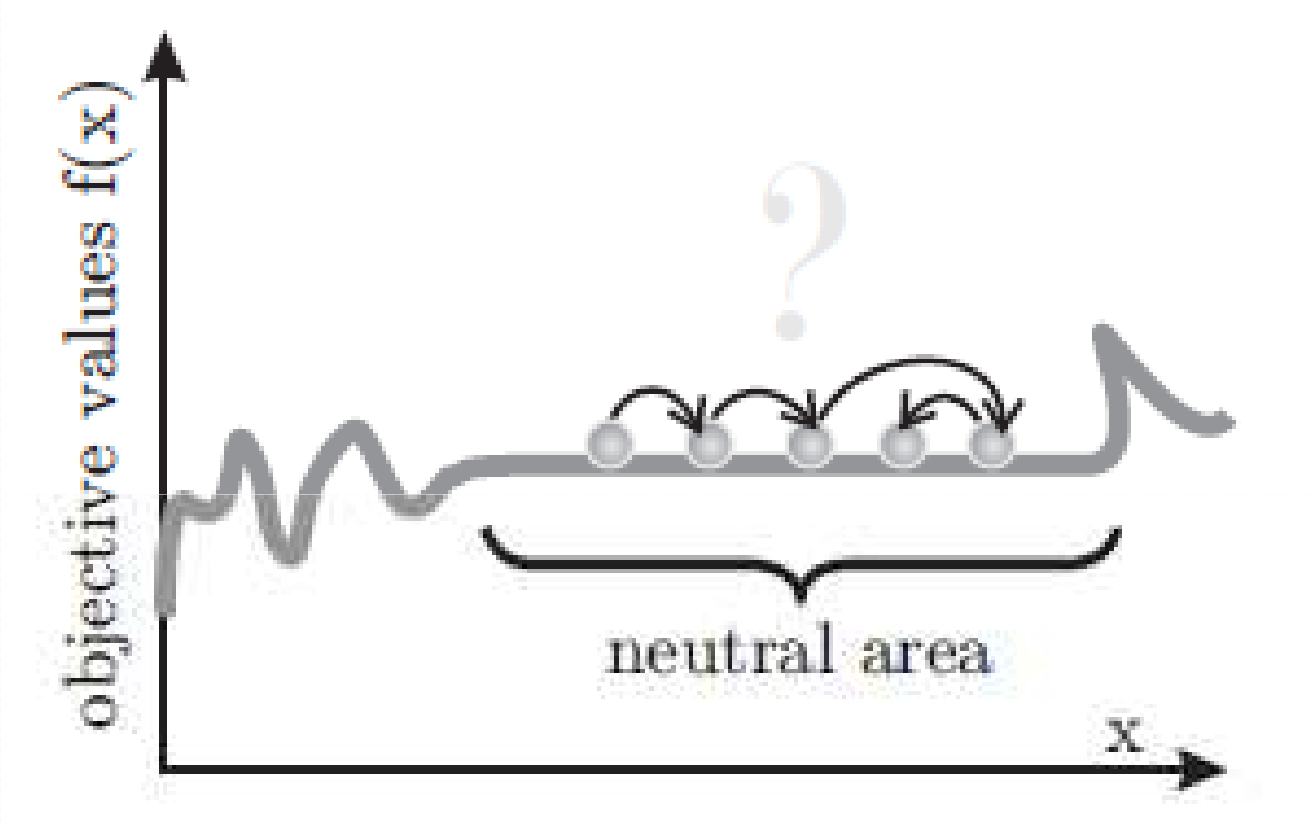
Deceptiveness



Deceptiveness

- Especially annoying fitness landscapes show deceptiveness (or deceptivity).
- The gradient of deceptive objective functions leads the optimizer away from the optima, as illustrated before.

Neutrality



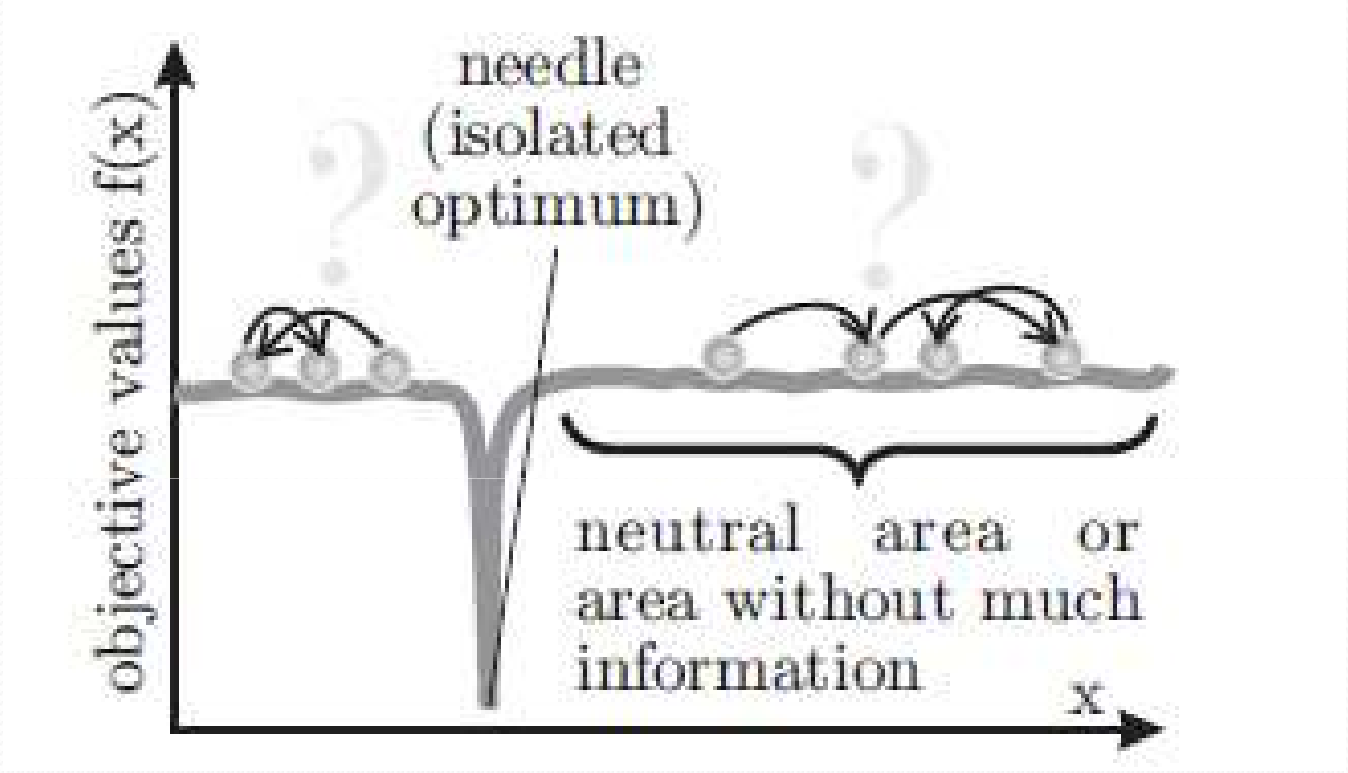
Neutrality, evolvability

- We consider the outcome of the application of a search operation to an element of the search space as neutral if it yields no change in the objective values.
- An optimizer then cannot find any gradient information and thus, no direction in which to proceed in a systematic manner. From its point of view, each search operation will yield identical individuals.
- The evolvability of an optimization process in its current state defines how likely the search operations will lead to solution candidates with new (and eventually, better) objectives values.

Needle-In-A-Haystack

- One of the worst cases of fitness landscapes is the needle-in-a-haystack problem where the optimum occurs as isolated spike in a plane.
- Such optimization problems are extremely hard to solve and the optimization processes often will converge prematurely or take very long to find the global optimum.

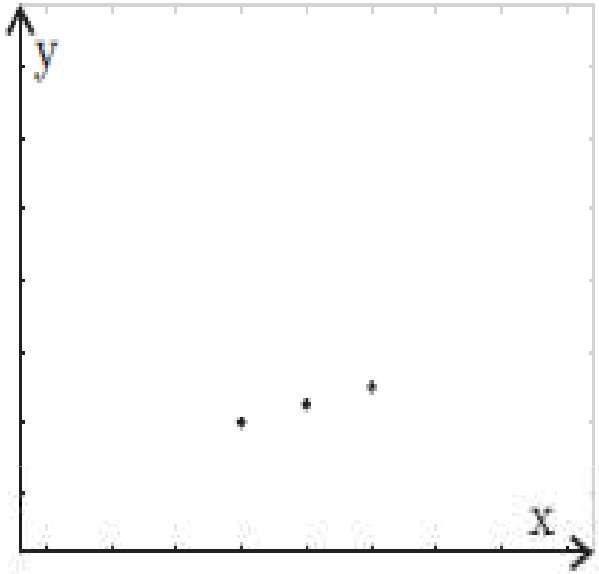
Needle-In-A-Haystack



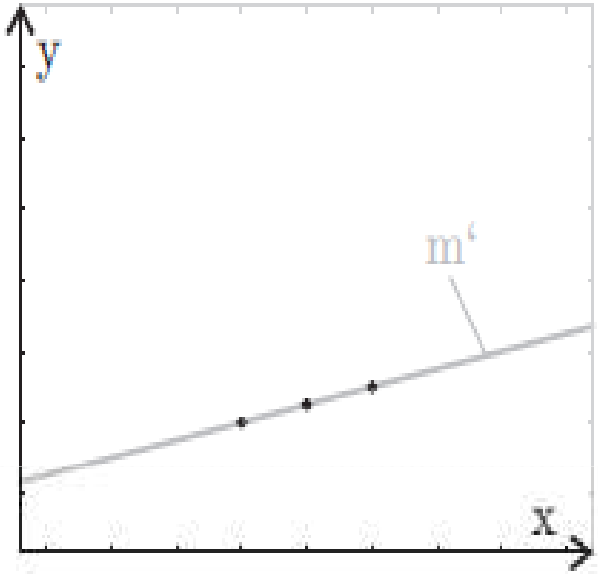
Overfitting

- Overfitting is the emergence of an overly complicated model (solution candidate) in an optimization process resulting from the effort to provide the best results for as much of the available training data as possible.
- The phenomenon of overfitting is best known and can often be encountered in the field of artificial neural networks or in curve fitting

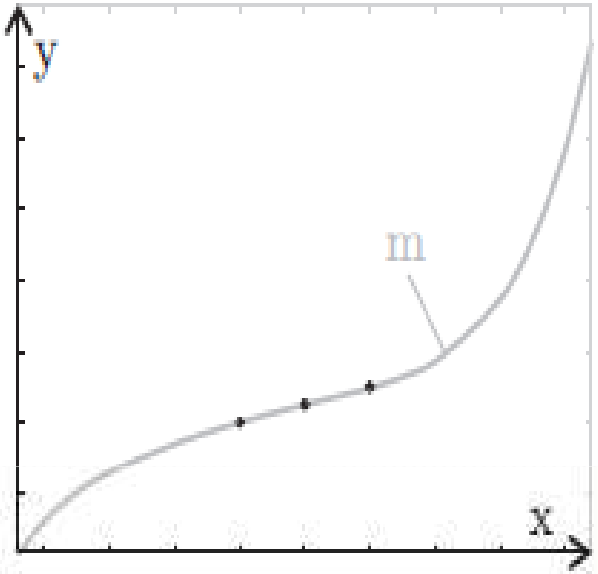
Overfitting due to complexity



Three sample points of f_1.



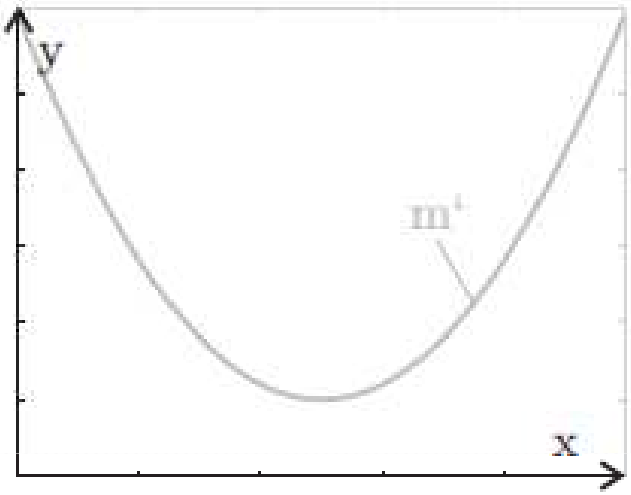
$m' \equiv f_1(x) = x.$



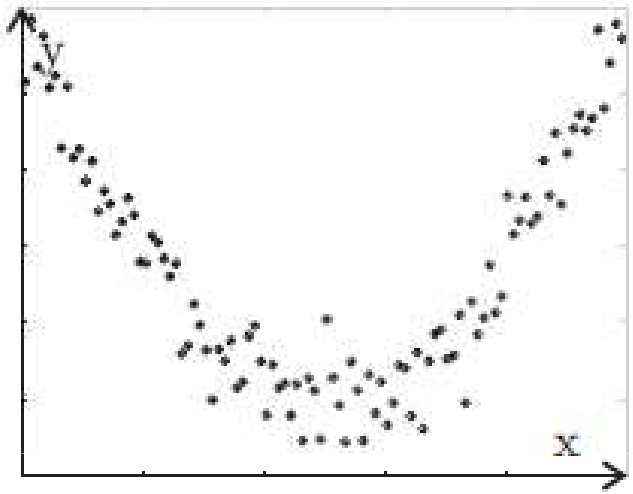
$m \equiv f_2(x).$

Overfitting

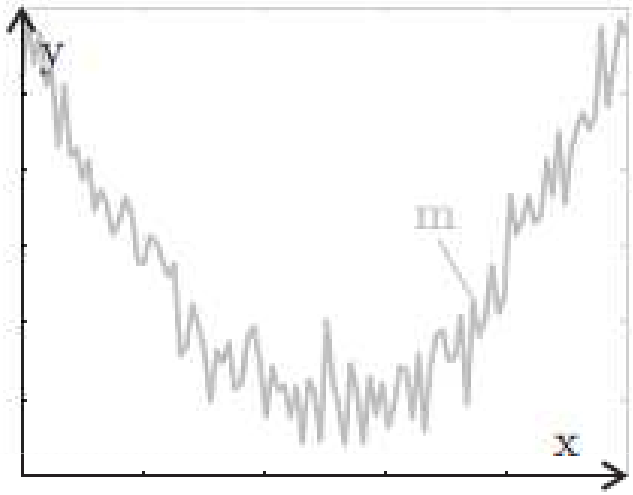
- A very common cause for overfitting is noise in the sample data.



The original physical process



The measurement



the overfitted result

- Introduction
- Definitions
- **Optimization methods:**
  - **Hill climbing**
  - Simulated annealing
  - Genetic algorithm
  - Ant colony optimization
  - Particle swarm optimization
  - Tabu search
- Examples of works in the Le2i laboratory

Hill Climbing

- Hill climbing is a very old and simple search and optimization algorithm for single objective functions f.
- In principle, hill climbing algorithms perform a loop in which the currently known best solution individual $p^\star$ is used to produce one offspring pnew. If this new individual is better than its parent, it replaces it. Then, the cycle starts all over again.
- In this sense, it is similar to an evolutionary algorithm with a population size ps of 1.
- The search space G and the problem space X are most often the same in hill climbing.

Hill Climbing

- The major problem of hill climbing is premature convergence, it gets easily stuck on a local optimum. It always uses the best known solution candidate $x^\star$ to find new points in the problem space X.
- Hill climbing utilizes a unary reproduction operation similar to mutation in evolutionary algorithms.
- It should be noted that hill climbing can be implemented in a deterministic manner if the neighbor sets in search space G, which here most often equals the problem space X, are always finite and can be iterated over.

Algorithms: hill climbing

- **Input:** f: the objective function subject to minization
- **Data:** pnew: the new element created
- **Data:** p*: the (currently) best individual
- **Output:** x*: the best element found
- 1 **begin**
- 2 | p*.g ←– create() // *Implicitly: p⋆.x ←– gpm(p*.g)*
- 3 | **while** terminationCriterion() do
- 4 | pnew.g ←– mutate(p* .g) // *Implicitly: pnew.x ←– gpm(pnew.g)*
- 5 | **if** f(pnew.x) < f(p*.x) **then** p* ←– pnew
- 6 | **return** p* .x
- 7 **end**

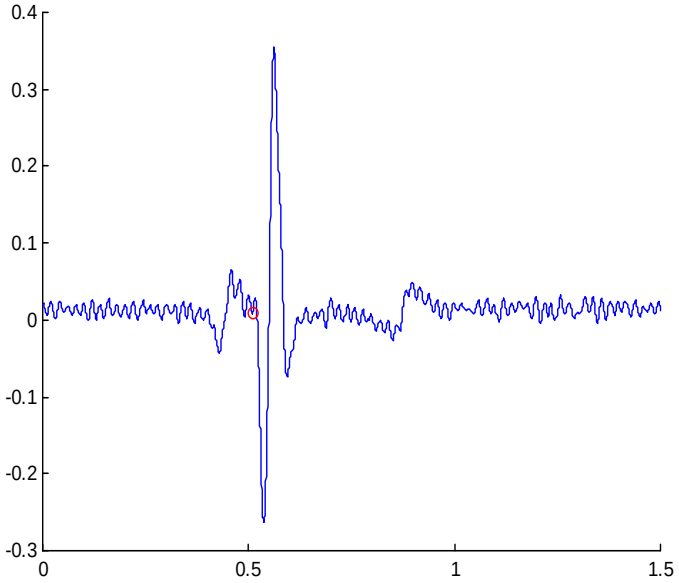
Matlab program

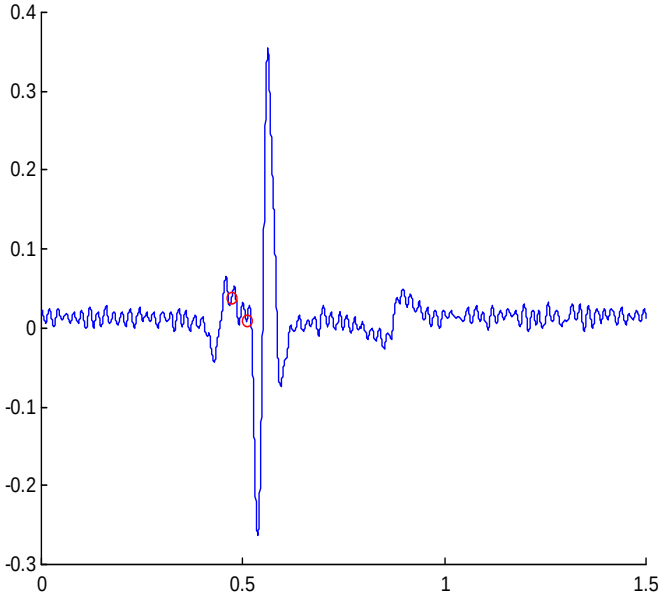
```
% on entre le vecteur v
% on sort la valeur de x qui conduit au maximum de v
function [bestx]=hillclimber(x,y)
t=length(x);
% initialisation de xbestx
bestindice=fix(t*rand)+1;
bestx=x(bestindice);
% critère d'arrêt y>0.9
 while y(bestindice)<0.9
% nouvelle valeur de x choisie au hasard dans le vecteur
indicenew=fix(t*rand)+1;
xnew=x(indicenew);
% calcul de y=f(x) correspondant
ynew=y(indicenew);
```

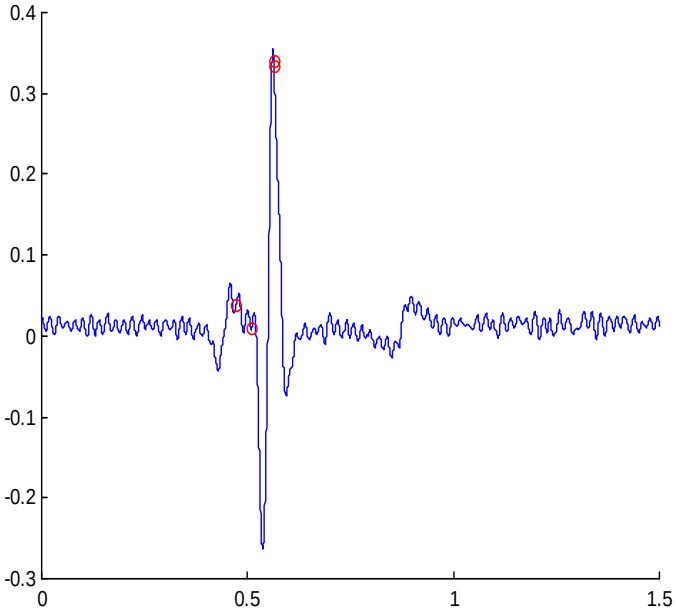
Matlab program -2

```
% Si le nouvel y  est mieux que le premier on le remplace
if ynew>y(bestindice)
   bestindice=indicenew;
   end
bestx=x(bestindice);
 end
return
```

Hill Climbing: Results on ECG







Algorithms: hill climbing

- We can easily extend hill climbing algorithms with a support for multi-objective optimization by using some of the methods of evolutionary algorithms.
- This extended approach will then return a set $X^\star$ of the best solutions found instead of a single individual $x^\star$ The set of currently known best individuals Arc may contain more than one element.
- Therefore, we employ a selection scheme in order to determine which of these individuals should be used as parent for the next offspring in the multi-objective hill climbing algorithm.

Problems

- Both versions of the algorithm are still very likely to get stuck on local optima. Hill climbing in this form is a local search rather than global optimization algorithm.
- By making a few slight modifications to the algorithm however, it can become a valuable global optimization technique:
- A tabu-list which stores the elements recently evaluated can be added. By preventing the algorithm from visiting them again, a better exploration of the problem space X can be enforced.

Problems

- Another way of preventing premature convergence is to not always transcend to the better solution candidate in each step. Simulated Annealing introduces a heuristic based on the physical model the cooling down molten metal to decide whether a superior offspring should replace its parent or not.
- Randomly restarting the search after so-and-so many steps is a crude but efficient method to explore wide ranges of the problem space with hill climbing.
- Using a reproduction scheme that not necessarily generates solution candidates directly neighboring $x\star$, as done in Random Optimization, an optimization approach may prove even more efficient

- Introduction
- Definitions
- Optimization methods:
  - Hill climbing
  - **Simulated annealing**
  - Genetic algorithm
  - Ant colony optimization
  - Particle swarm optimization
  - Tabu search
- Examples of works in the Le2i laboratory

Introduction

- In 1953, Metropolis developed a Monte Carlo method for “calculating the properties of any substance which may be considered as composed of interacting individual molecules”. With this so-called “Metropolis” procedure stemming from statistical mechanics, the manner in which metal crystals reconfigure and reach equilibria in the process of annealing can be simulated.
- This inspired Kirkpatrick to develop the Simulated Annealing1 (SA) algorithm for global optimization in the early 1980s and to apply it to various combinatorial optimization problems.

Introduction

- Independently, ˇ Cern´y employed a similar approach to the travelling salesman problem.
- Simulated Annealing is an optimization method that can be applied to arbitrary search and problem spaces.
- Like simple hill climbing algorithms, Simulated Annealing only needs a single initial individual as starting point and a unary search operation

Algorithms: Simulated annealing

- In metallurgy and material science, annealing is a heat treatment of material with the goal of altering its properties such as hardness.
- Metal crystals have small defects, dislocations of ions which weaken the overall structure.
- By heating the metal, the energy of the ions and, thus, their diffusion rate is increased. Then, the dislocations can be destroyed and the structure of the crystal is reformed as the material cools down and approaches its equilibrium state.
- When annealing metal, the initial temperature must not be too low and the cooling must be done sufficiently slowly so as to avoid the system getting stuck in a meta-stable, non-crystalline, state representing a local minimum of energy.

Introduction

- In physics, each set of positions of all atoms of a system pos is weighted by its Boltzmann probability
- factor $e^{\frac{-E(pos)}{kT}}$ where E(pos) is the energy of the
- configuration pos, T is the temperature measured in Kelvin, and k is the Boltzmann's constant

$$k = 1.380650524\ 10^{-23}\ J / K$$

Introduction

- The Metropolis procedure was an exact copy of this physical process which could be used to simulate a collection of atoms in thermodynamic equilibrium at a given temperature.
- A new nearby geometry pos_{i+1} was generated as a random displacement from the current geometry pos_i of an atom in each iteration.
- The energy of the resulting new geometry is computed and E, the energetic difference between the current and the new geometry, was determined.

Introduction

$$\Delta(E) = E(pos_{i+1}) - E(pos_i)$$

$$p(\Delta(E)) = \begin{cases} e^{\frac{-\Delta(E)}{kT}} \; if \; \Delta(E) > 0 \\ 1 \; otherwise \end{cases}$$

- Thus, if the new nearby geometry has a lower energy level, the transition is accepted.
- Otherwise, a uniformly distributed random number r=randomu() ∈ [0, 1] is drawn and the step will only be accepted in the simulation if it is less or equal the Boltzmann probability factor, r ≤ P(E).

- At high temperatures T, this factor is very close to 1, leading to the acceptance of many uphill steps. As the temperature falls, the proportion of steps accepted which would increase the energy level decreases. Now the system will not escape local regions anymore and (hopefully) comes to a rest in the global minimum at temperature T = 0K.
- The abstraction of this method in order to allow arbitrary problem spaces is straightforward – the energy computation $E(pos_i)$ is replaced by an objective function f or even by the result v of a fitness assignment process.
- Without loss of generality, we reuse the definitions from evolutionary algorithms for the search operations and set Op = {create,mutate}.

Algorithms: Simulated annealing

- **Input: f:** the objective function to be minimized
- **Data:** pnew: the newly generated individual
- **Data:** pcur: the point currently investigated in problem space
- **Data:** p*: the best individual found so far
- **Data:** T: the temperature of the system which is decreased over time
- **Data:** t: the current time index
- **Data:** DE: the energy difference of the xnew and xcur
- **Output:** x*: the best element found
- 01 **begin**
- 02 | pnew.g ← create() // *Implicitly: pnew.x ← gpm(pnew.g)*
- 03 | pcur ← pnew
- 04 | p* ← pnew
- 05 | t ← 0

Algorithms: Simulated annealing

```
06 | while terminationCriterion() do
07 |    | DE ← f(pnew.x) – f(pcur.x)
08 |    | if DE ≤ 0 then
09 |    |          | pcur ← pnew
10 |    |          | if f(pcur.x) < f(p*.x) then p* ← pcur
11 |    |else
12 |    |          | T ←– getTemperature(t)
13 |    |          | if random () <  then pcur ← pnew
14 |    | pnew.g ← mutate(pcur.g)    // Implicitly: pnew.x ← gpm(pnew.g)
15 |    | t ← t + 1
16 | return p*.x
17 end
```

Matlab program

```
function [bestx,bestindice]=SA(x,y)
l=length(x);
k=0.01; % plus adapté à sin(x) que 1.38*10^-23;
inew=fix(l*rand)+1          % new point;
icur=fix(l*rand)+1          % current point;
ibest=icur                  % starting best indice;
temp=800                    % initial temperature;
t=0                         % current time;
while temp>=1               % temperature loop;
```

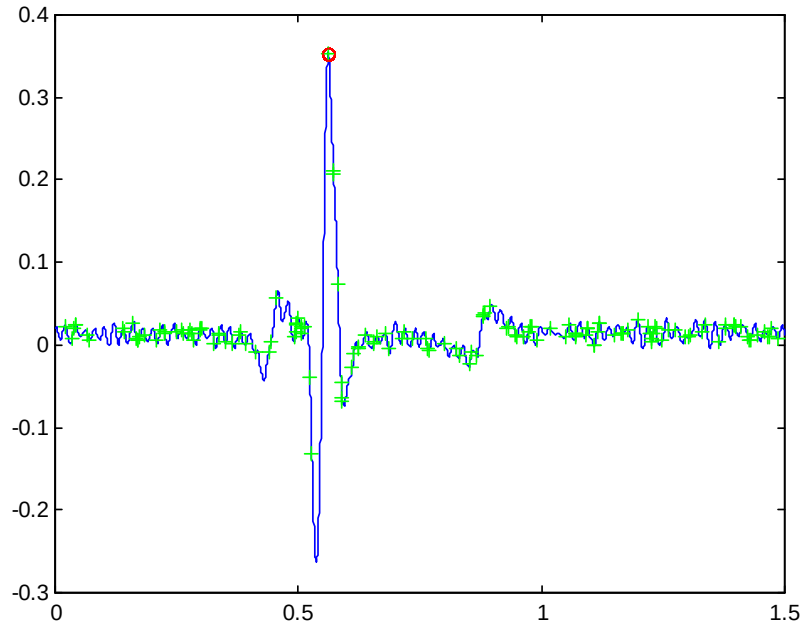
Matlab program part-2

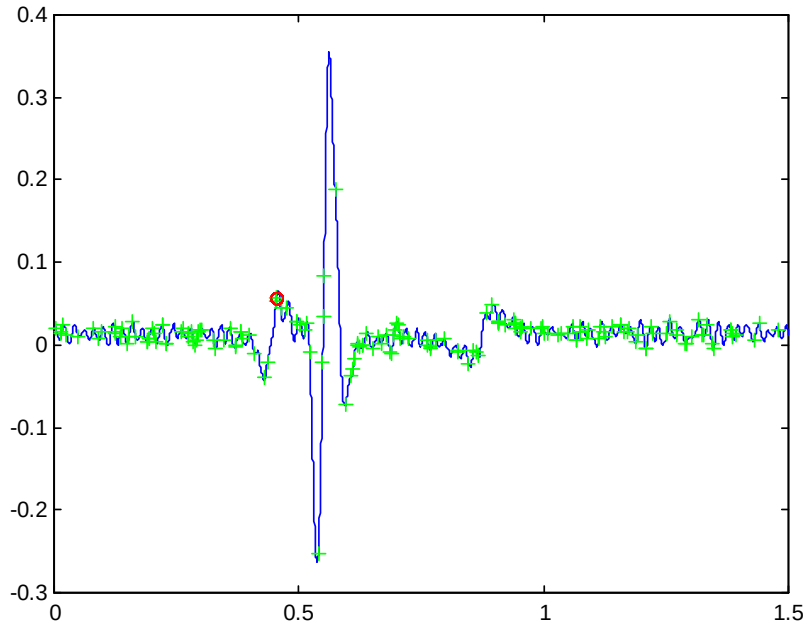
```
for n=1:10 % n calculs par valeur de température
deltaE=y(inew)-y(icur)
    if deltaE<=0 % on accepte une baisse de l'énergie
    icur=inew
                if y(icur)<y(ibest)
      ibest=icur
                end
                 else
    u=rand
    exp(-deltaE/(k*temp))
                if u<exp(-deltaE/(k*temp))
```

Matlab program part-3

```
% on peut accepter une augmentation de l'énergie avec une probabilité fonction de la température
        icur=inew
                    end
% nouveau point choisi au hasard
                    end
 t=t+1
        inew=fix(l*rand)+1
end
temp=temp/2;
% décroissance température
end
bestx=x(ibest)
bestindice=ibest
```

Application to ECG





- Global maximum found and global maximum not found

Temperature scheduling

- It has been shown that Simulated Annealing algorithms with **appropriate cooling strategies** will asymptotically converge to the global optimum.
- The temperature schedule defines how the temperature in Simulated Annealing is decreased.
- As already mentioned, this has major influence on whether the Simulated Annealing algorithm will succeeded, on whether how long it will take to find the global optimum, and on whether or not it will degenerate to simulated quenching.

Temperature scheduling

- There exists a wide range of methods to determine this temperature schedule.
- M1: Reduce T to (1 – ε)T after every m iterations, where the exact values of 0 < ε < 1 and m > 0 are determined by experiment.
- M2: Grant a total of K iterations, and reduce T after
- every m steps to a value T = $Tstart.\left(1-\frac{t}{k}\right)^{\alpha}$
- where t is the inted of the current iteration and α is a constant; large values of α will spend more iterations at lower temperature.

Temperature scheduling

- M3: After every m moves, set T to β times $Ec=f(x_{cur})-f(x\star)$, where β is an experimentally determined constant, $f(x_{cur})$ is the objective value of the currently examined solution candidate x_{cur}, and $f(x\star)$ is the objective value of the best phenotype $x\star$ found so far.
- Since Ec may be 0, we limit the temperature change to a maximum of $T * \gamma$ with $0 < \gamma < 1$.
- If we let the temperature sink fast, we will lose the property of guaranteed convergence.
- In order to avoid getting stuck at local optima, we can then apply random restarting, which already has been discussed in the context of hill climbing algorithm.

Multiobjective SA

- Again, we want to combine this algorithm with multiobjective optimization and also enable it to return a set of optimal solutions. This can be done even simpler than in multi-objective hill climbing.
- Basically, we just need to replace the single objective function f with the fitness values v computed by a fitness assignment process on basis of the set of currently known best solutions (Arc), the currently investigated individual (p_{cur}), and the newly created points in the search space (p_{new}).

- Introduction
- Definitions
- Optimization methods:
  - Hill climbing
  - Simulated annealing
  - **Genetic algorithm**
  - Ant colony optimization
  - Particle swarm optimization
  - Tabu search
- Examples of works in the Le2i laboratory

Genetic algorithm

- Evolutionary algorithms (EAs) are generic, population-based, metaheuristic optimization algorithms that use biology-inspired mechanisms like mutation, crossover, natural selection, and survival of the fittest in order to refine a set of solution candidates iteratively.
- The advantage of evolutionary algorithms compared to other optimization methods is their “black box” character that makes only few assumptions about the objective functions. Furthermore, the definition of objective functions usually requires lesser insight to the structure of the problem space than the manual construction of an admissible heuristic.
- EAs therefore perform consistently well in many different problem categories.

Introduction

- In 1859, Darwin published his book “On the Origin of Species” in which he identified the principles of natural selection and survival of the fittest as driving forces behind the biological evolution. His theory can be condensed into ten observations and deductions:
- 1. The individuals of a species posses great fertility and produce more offspring than can grow into adulthood.
- 2. Under the absence of external influences (like natural disasters, human beings, etc.), the population size of a species roughly remains constant.
- 3. Again, if no external influences occur, the food resources are limited but stable over time.

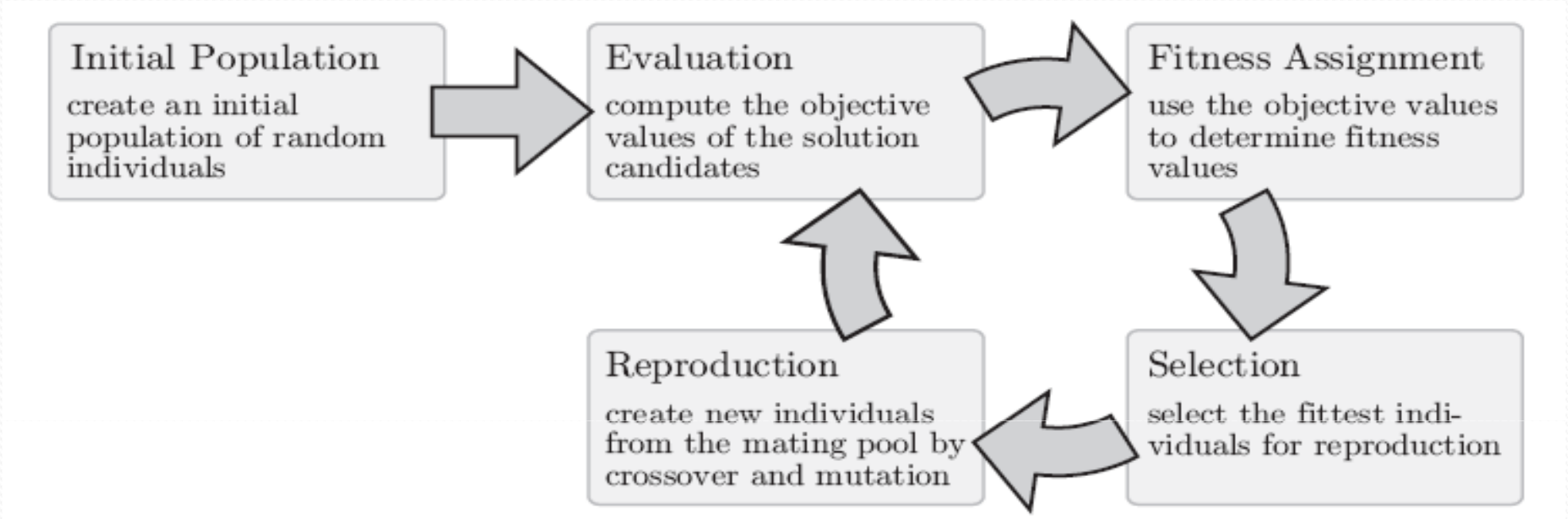
Introduction

- 4. Since the individuals compete for these limited resources, a struggle for survival ensues.
- 5. Especially in sexual reproducing species, no two individuals are equal.
- 6. Some of the variations between the individuals will affect their fitness and hence, their ability to survive.
- 7. A good fraction of these variations are inheritable.
- 8. Individuals less fit are less likely to reproduce, whereas the fittest individuals will survive and produce offspring more probably.

Introduction

- 9. Individuals that survive and reproduce will likely pass on their traits to their offspring.
- 10. A species will slowly change and adapt more and more to a given environment during this process which may finally even result in new species.

Introduction



All evolutionary algorithms proceed in principle according to this scheme

Introduction

- 1. Initially, a population Pop of individuals p with a random genome p.g is created.
- 2. The values of the objective functions $f \in F$ are computed for each solution candidate p.x in Pop. This evaluation may incorporate complicated simulations and calculations.
- 3. With the objective functions, the utility of the different features of the solution candidates have been determined and a fitness value v(p.x) can now be assigned to each of them. This fitness assignment process can, for instance, incorporate a prevalence comparator function cmpF which uses the objective values to create an order amongst the individuals.

Introduction

- 4. A subsequent selection process filters out the solution candidates with bad fitness and allows those with good fitness to enter the mating pool with a higher probability.
- 5. In the reproduction phase, offspring is created by varying or combining the genotypes (which are called reproduction operations in the context of EAs).
- 6. If the terminationCriterion() is met, the evolution stops here. Otherwise, the algorithm continues at step 2.

Algorithm

- John Holland proposed the first GA.
- This « canonic genetic algorithm » is decribed in the following part.
- Chromosoms are binary chains so the operators are easyly understable.
- The matlab program was published by CAO and WU under the title « teaching GA using Matlab ».

$$pop = \begin{pmatrix} chaine\ binaire\ 1 & x_1 & f(x_1) \\ chaine\ binaire\ 2 & x_2 & f(x_2) \\ . & . & . \\ . & . & . \\ . & . & . \\ chaine\ binaire\ \ popsize & x_{popsize} & f\left(x_{popsize}\right) \\ 010011000 & x & f(x) \end{pmatrix}$$

The table contains the binary value of x, real value of x, the fitness value of x
This table represents the initial population and will be modified at each generation loop.

CGA

```
% CGA (J.Holland 1975) *
clear all
hold on
% initialisation des paramètres *
stringlength=8;     % chromosom length
popsize=6;          % population size
borneinf=0;         % borne inf of search space
bornesup=1;         % borne sup of search space
pm=0.5;             % mutation probability
pc=0.5;             % probabilité de crossover
N=2^stringlength; % nombre de points à afficher
```

CGA

- % initialisation de la population *
- oldpop=initialise(popsize,stringlength,borneinf, bornesup)
- % 20 generations loop
- finalgen=20;
- for gen=1:1:finalgen;
- sprintf('generation numéro',gen)
- % selection génération courante *
- newpop=selection(oldpop,stringlength,popsize)

CGA

```
% crossover génération courante *
if rand<pc
        x=round(rand*(popsize-1))+1;
        y=round(rand*(popsize-1))+1;
        while x==y
          y=round(rand*(popsize-1))+1;
        end
        parent1=newpop(x,:);
        parent2=newpop(y,:);
[child1,child2]=croisement(parent1,parent2,stringlength,bornein
f,bornesup);
     newpop(x,:)=child1;
     newpop(y,:)=child2;
 end
```

CGA

- % mutation génération courante *
- for x=1:popsize
- if rand<pm
- parent=newpop(x,:);
- newpop(x,:)=mutation(parent,stringlength,borne inf,bornesup);
- end
- end
- % calcul de la fitness génération courante *
- oldpop=newpop

CGA

- %affichage des points *
- for h=borneinf:(bornesup-borneinf)/N:bornesup
- plot(h,fun(h),'k.-')
- end
- for x=1:popsize
- plot(oldpop(x,stringlength+1),oldpop(x,stringlength+2),'ro')
- end
- pause
- end

Operator: fitness calculation

- **% définition de la fonction fitness**
- **function[y]=fitness(x);**
- **y=sin(10*x)^2/(1+x);**
- **Return**

- **In this case, we try only to obtain the maximum of the function and the function is known.**
- **Single objective optimisation.**

Operators: initialisation

$$pop = \begin{pmatrix} chaine\ binaire\ 1 & x_1 & f(x_1) \\ chaine\ binaire\ 2 & x_2 & f(x_2) \\ . & . & . \\ . & . & . \\ . & . & . \\ chaine\ binaire\ \ popsize & x_{popsize} & f\left(x_{popsize}\right) \end{pmatrix}$$

$$010011000 \qquad x \qquad f(x)$$

Operators: initialisation

- % initialisation de la population *
- function[pop,res,res2]=initialise(popsize,stringlength,borneinf,bornesup);
- % tirage de popsize nombres binaires *
- pop=round(rand(popsize,stringlength+2))
- % calcul de la partie binaire
- res=2.^((size(pop(:,1:stringlength),2)-1):-1:0);
- puissance2mat=res;

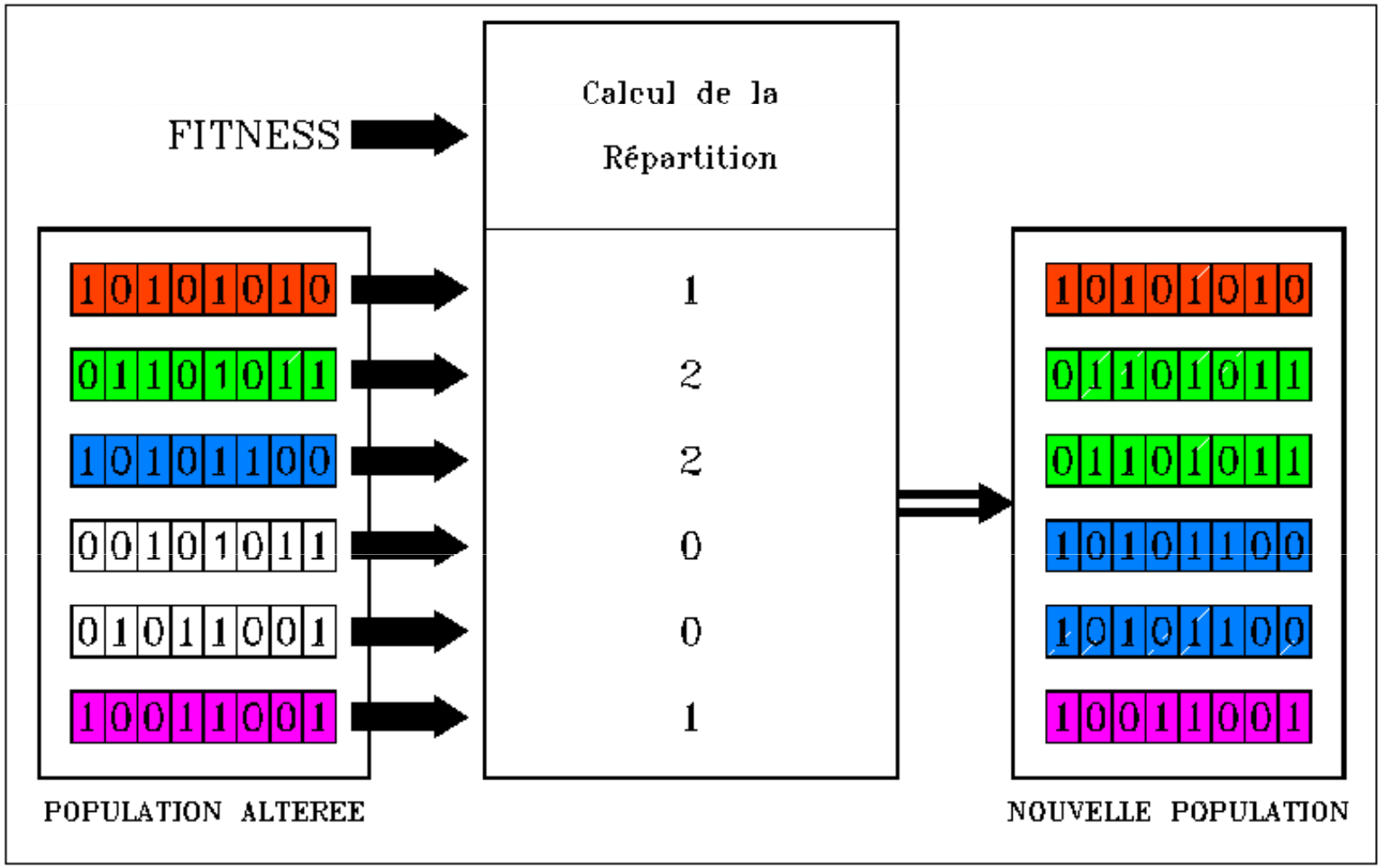
Operators:initialisation

- % uniformisation de taille avec la matrice binaire
- for p=1:1:popsize-1;
- puissance2mat=[puissance2mat;res];
- end
- res2=puissance2mat.*pop(:,1:stringlength);
- % conversion en réel de colonne lengthsize+1 *
- for p=1:1:popsize;
- pop(p,stringlength+1)=sum(res2(p,:))*(bornesup-borneinf)/(2.^stringlength-1)+borneinf;
- end

Operators: initialisation

- **% calcul de la fitness pour la colonne lengthsize+2**
  for p=1:1:popsize;
- **pop(p,stringlength+2)=fitness(pop(p,stringlength +1));**
- **End**
- **% retourne la matrice population avec code binaire, puis valeur réelle, puis fitness ***
- **return**

Operators: selection



Operators: selection

- Roulette wheel selection: fitness proportionate selection, introduced by Holland
- Tournament selection: k elements are picked from the population Pop and compared with each other in a tournament. The winner of this competition will then enter mating pool Mate.
- Ordered selection: the probability of an individual to be selected is proportional to (a power of) its position (rank) in the sorted list of all individuals in the population.

Operators: selection

- The elitist approach consist in keeping the best chromosom in the next generation.
- The convergence speed of the algorithm increase.

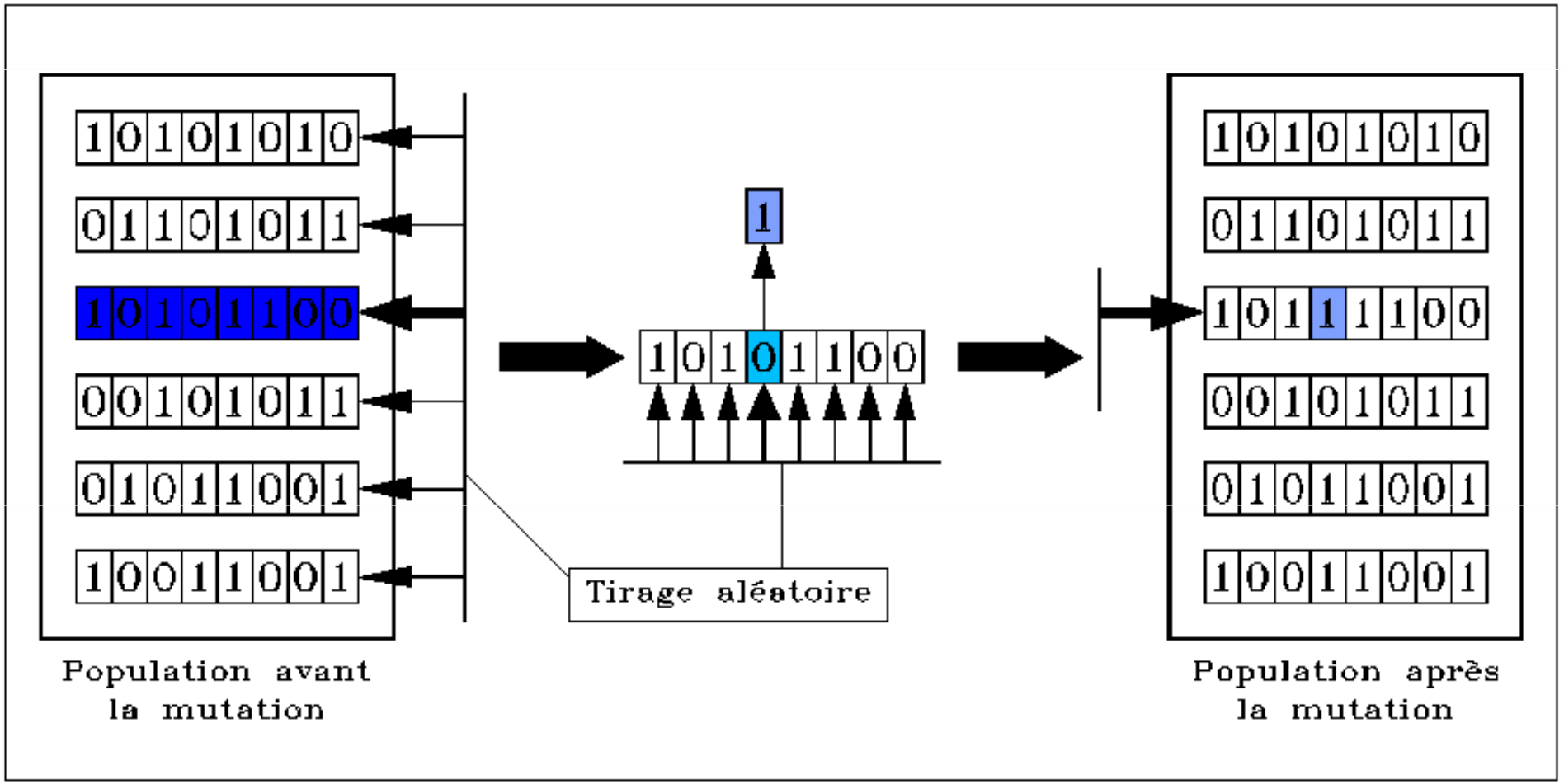
Operator: selection

```
% sélection par roulette *
function[newpop]=selection(oldpop,stringlength,popsize);
totalfit=sum(oldpop(:,stringlength+2));
prob=oldpop(:,stringlength+2)/totalfit;
prob=cumsum(prob);
rns=sort(rand(popsize,1));
fitin=1;newin=1;
while newin<=popsize
  if rns(newin)<prob(fitin);
    newpop(newin,:)=oldpop(fitin,:);
    newin=newin+1;
  else
    fitin=fitin+1;
  end
end
return
```

Operators: mutation

- preserving the diversity of the solution candidates.
- Necessary for a good exploration (of the solution space)
- Probability: pm
- In the nature, few possibilities of mutation are encountered.

Operators: mutation



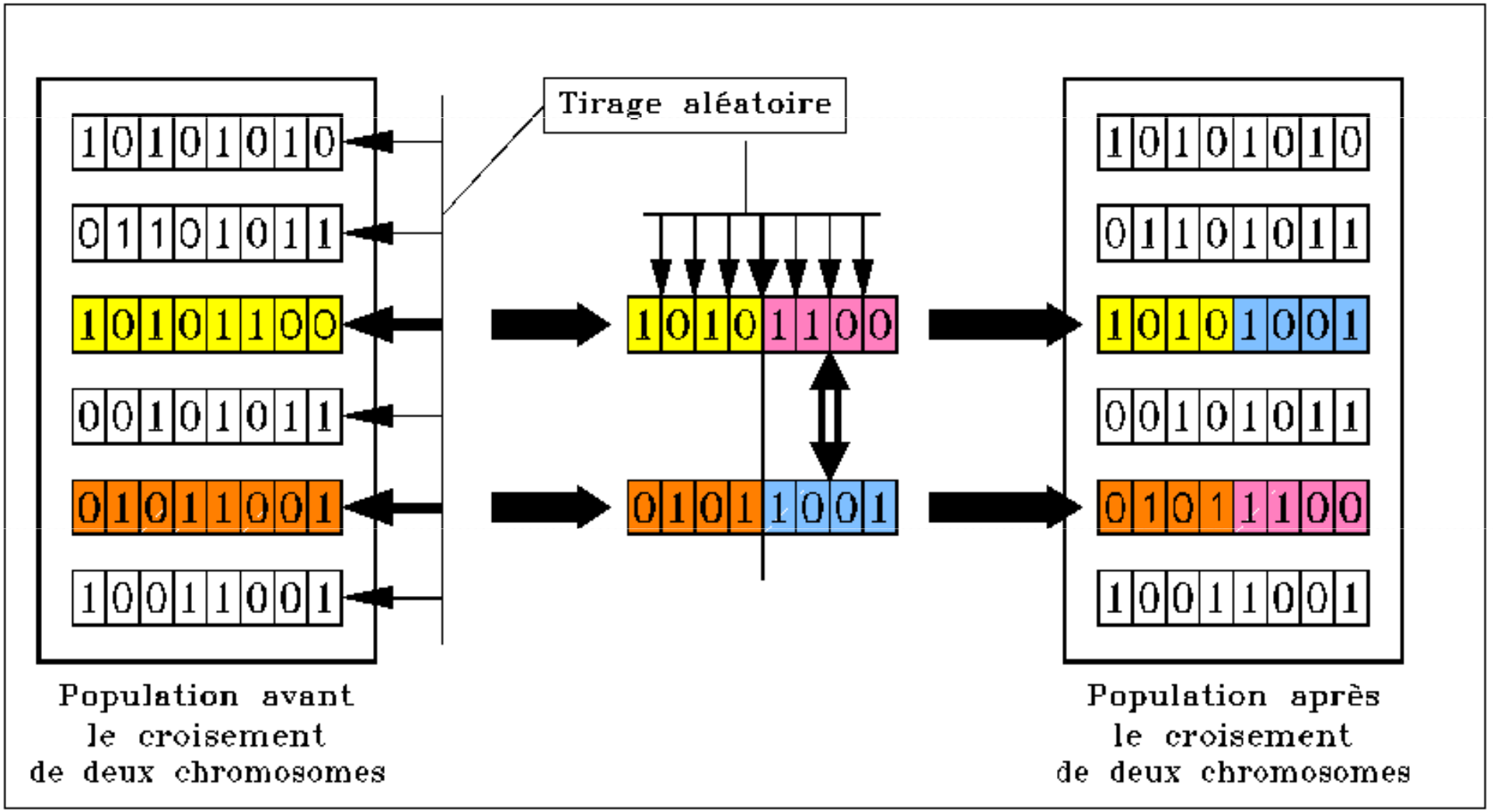
Operators: mutation

- **function[child]=mutation(parent,stringlength,borneinf, bornesup);**
- **% choix du point de mutation mpoint ***
- **mpoint=round(rand*(stringlength-1))+1;**
- **child=parent;**
- **child(mpoint)=abs(parent(mpoint)-1);**
- **% conversion en réel de colonne lengthsize+1 ***
- **child(:,stringlength+1)=sum(2.^(size(child(:,1:stringlength),2)-1:-1:0).*child(:,1:stringlength)) *(bornesup-borneinf)/(2.^stringlength-1)+borneinf;**
- **% calcul de la fitness pour la colonne lengthsize+2**
  child(:,stringlength+2)=fun(child(:,stringlength+1));
- **return**

Operators: crossover

- Amongst all evolutionary algorithms, genetic algorithms have a recombination operation which is closest to the natural paragon.
- The next figure outlines the recombination of two stringchromosomes, the so-called crossover, which is performed by swapping parts of two genotypes.
- Crossover contributes to exploration and exploitation.

Operators: crossover



Operators: crossover

- Single-point Crossover
- Two-points Crossover
- Multi-points Crossover

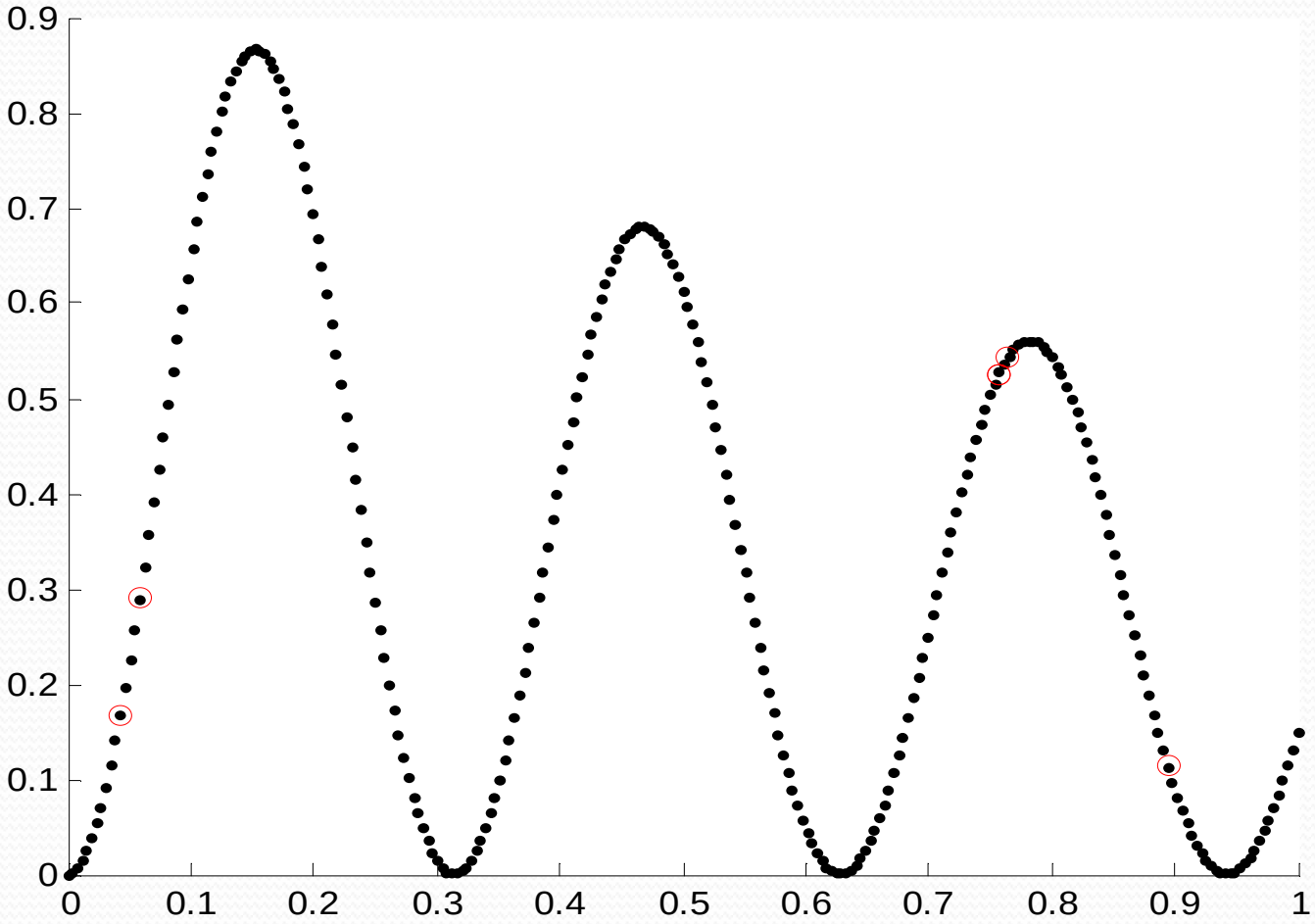
Operator: crossover

- **function[child1,child2]=croisement(parent1,parent2,stringlength,borneinf,bornesup);**
- **% choix du point de croisement cpoint**
- **cpoint=round(rand*(stringlength-2))+1;**
- **% Calcul des enfants**
- **child1=[parent1(:,1:cpoint) parent2(:,cpoint+1:stringlength)];**
- **child2=[parent2(:,1:cpoint) parent1(:,cpoint+1:stringlength)];**

Operators: crossover

- **% conversion en réel de colonne lengthsize+1**
- **child1(:,stringlength+1)=sum(2.^(size(child1(:,1:stringlength),2)-1:-1:0).*child1(:,1:stringlength))*(bornesup-borneinf)/(2.^stringlength-1)+borneinf;**
- **child2(:,stringlength+1)=sum(2.^(size(child2(:,1:stringlength),2)-1:-1:0).*child2(:,1:stringlength))*(bornesup-borneinf)/(2.^stringlength-1)+borneinf;**
- **% calcul de la fitness pour la colonne lengthsize+2**
- **child1(:,stringlength+2)=fun(child1(:,stringlength+1));**
- **child2(:,stringlength+2)=fun(child2(:,stringlength+1));**
- **return**

Example



pop =

1	0	1	0	0	0	1	1	0.6392	0.0072
0	0	1	1	0	1	1	0	0.2118	0.6021
1	1	0	0	0	0	0	1	0.7569	0.5241
0	0	0	0	1	1	1	1	0.0588	0.2908
1	1	1	1	0	0	1	1	0.9529	0.0056
1	1	1	0	0	0	0	0	0.8784	0.1901

newpop =

1	1	0	0	0	0	0	1	0.7569	0.5241
1	1	0	0	0	0	0	1	0.7569	0.5241
1	1	0	0	0	0	0	1	0.7569	0.5241
0	0	0	0	1	1	1	1	0.0588	0.2908
0	0	0	0	1	1	1	1	0.0588	0.2908
1	1	1	0	0	0	0	0	0.8784	0.1901

newpopaltérée =

1	1	0	0	0	0	0	1	0.7569	0.5241
1	1	0	0	0	0	1	1	0.7647	0.5427
1	1	0	0	0	0	0	1	0.7569	0.5241
0	0	0	0	1	0	1	1	0.0431	0.1676
0	0	0	0	1	1	1	1	0.0588	0.2908
1	1	1	0	0	1	0	0	0.8941	0.1141

- Introduction
- Definitions
- Optimization methods:
  - Hill climbing
  - Simulated annealing
  - Genetic algorithm
  - **Ant colony optimization**
  - Particle swarm optimization
- Examples of works in the Le2i laboratory

ACO

- Ant colony optimisation
- Marco Dorigo et al.
- Algorithm for problems that can be reduced to finding optimal paths in graphs in 1996.
- Based on the metaphor of ants seeking food.
- An ant will leave the anthill and begin to wander into a random direction.
- While the little insect paces around, it lays a trail of pheromone.
- Thus, after the ant has found some food, it can track its way back.

ACO

- By doing so, it distributes another layer of pheromone on the path.
- An ant that senses the pheromone will follow its trail with a certain probability.
- Each ant that finds the food will excrete some pheromone on the path.
- By time, the pheromone density of the path will increase and more and more ants will follow it to the food and back.
- The higher the pheromone density, the more likely will an ant stay on a trail.

ACO

- However, the pheromones vaporize after some time.
- If all the food is collected, they will no longer be renewed and the path will disappear after a while.
- Now, the ants will head to new, random locations.

- Introduction
- Definitions
- Optimization methods:
  - Hill climbing
  - Simulated annealing
  - Genetic algorithm
  - Ant colony optimization
  - **Particle swarm optimization**
- Examples of works in the Le2i laboratory

PSO

- Particle Swarm Optimization, developed by Eberhart and Kennedy (1995)
- a form of swarm intelligence in which the behavior of a biological social system like a flock of birds or a school of fish is simulated.
- When a swarm looks for food, its individuals will spread in the environment and move around independently.
- Each individual has a degree of freedom or randomness in its movements which enables it to find food accumulations.

PSO

- So, sooner or later, one of them will find something digestible and, being social, announces this to its neighbors.
- These can then approach the source of food, too.
- Each individual has a degree of freedom or randomness in its movements which enables it to find new food accumulations.
- The velocity vector of an individual p determines in which direction the search will continue and if it has an explorative (high velocity) or an exploitive (low velocity) character.

TS

- Tabu Search has been developed by Glover in the mid 1980s.
- The word “tabu” stems from Polynesia and describes a sacred place or object.
- Things that are tabu must be left alone and may not be visited or touched.
- Tabu Search extends hillclimbing by this concept – it declares solution candidates which have already been visited as tabu.
- they must not be visited again and the optimization process is less likely to get stuck on a local optimum.

- Introduction
- Definitions
- Optimization methods:
  - Hill climbing
  - Simulated annealing
  - Genetic algorithm
  - Ant colony optimization
  - Particle swarm optimization
  - **Tabu search**
- Examples of works in the Le2i laboratory

TS

- **Input:** f: the objective function subject to minization
- **Input:** n: the maximum length of the tabu list (n > 0)
- **Data:** pnew: the new element created
- **Data:** p⋆: the (currently) best individual
- **Data:** tabu: the tabu list
- **Output:** x⋆: the best element found

TS

- **01** begin
- **02** | p⋆ ← create()
- **03** | tabu ← createList(1, p⋆)
- **04** | while terminationCriterion() do
- **05** | |pnew ← mutate(p⋆)
- **06** | | if searchItem(pnew. tabu) < 0 then
- **07** | | | if f(pnew) < f(p⋆) then p⋆ ← pnew
- **08** | | | if len(tabu) ≥ n then tabu ← deleteListItem(tabu, 0)
- **09** | | | tabu ← addListItem(tabu, pnew)
- **10** | return p⋆.x
- **11** end

- Introduction
- Definitions
- Optimization methods:
  - Hill climbing
  - Simulated annealing
  - Genetic algorithm
  - Ant colony optimization
  - Particle swarm optimization
  - Tabu search
- **Examples of works in the Le2i laboratory**

Le2i works

- Xavier Vernassier mémoire CNAM 1998
- Sandra Bouchard (thèse 1999)
- Youssef Bokhabrine (Master 2006)
- Sophie Voisin thèse 2008:

Youssef Bokhabrine

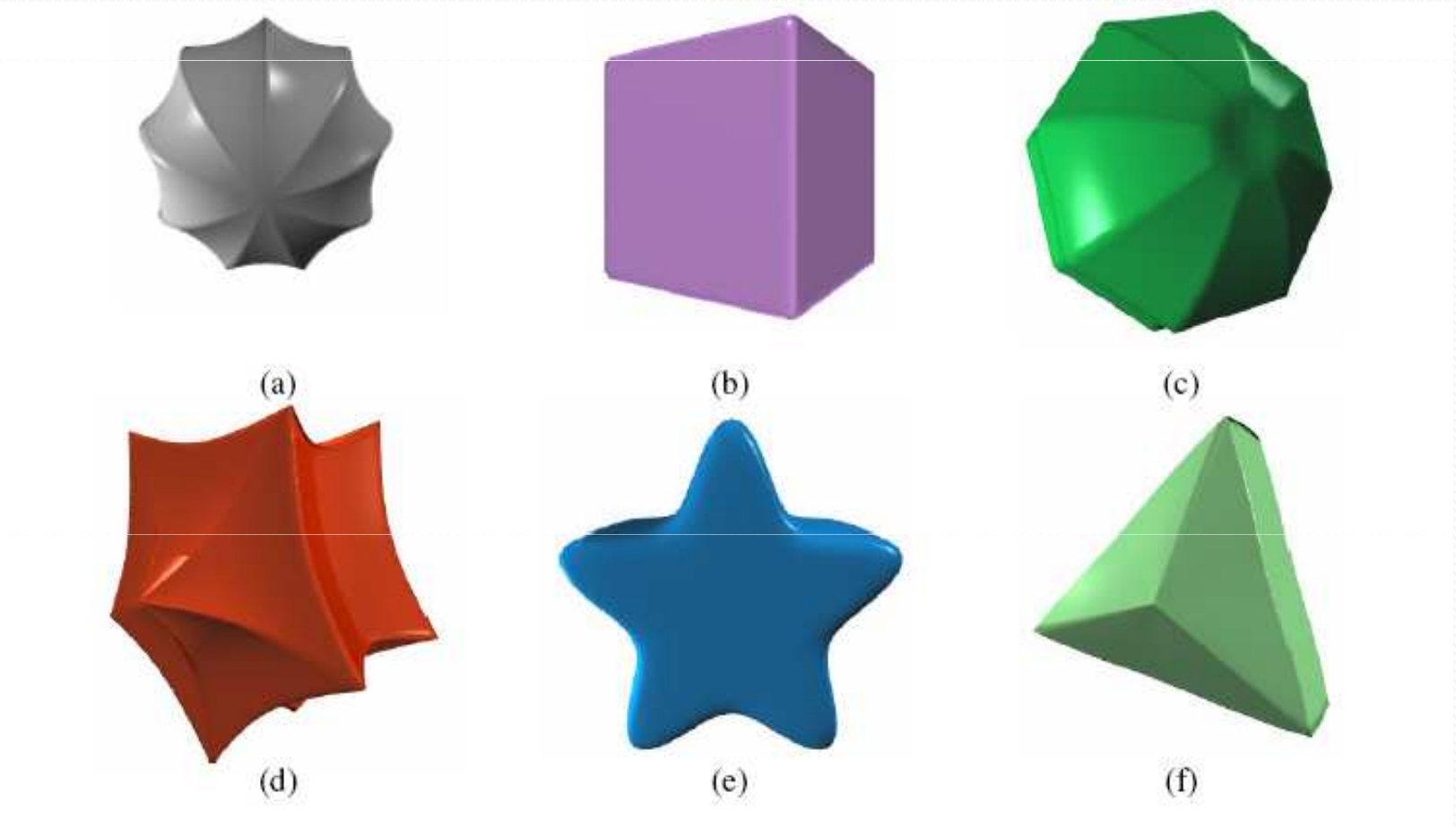
$$\begin{pmatrix} x(\theta,\phi) \\ y(\theta,\phi) \\ z(\theta,\phi) \end{pmatrix} = \begin{pmatrix} r_1(\theta) r_2(\phi) \cos\theta \cos\phi \\ r_1(\theta) r_2(\phi) \sin\theta \cos\phi \\ r_2(\phi) \sin\phi \end{pmatrix} \qquad (1)$$

avec
$$r(\theta) = \frac{1}{\sqrt[n_1]{\left|\frac{1}{a}\cos\left(\frac{m\theta}{4}\right)\right|^{n_2} + \left|\frac{1}{b}\sin\left(\frac{m\theta}{4}\right)\right|^{n_3}}} \qquad (2)$$

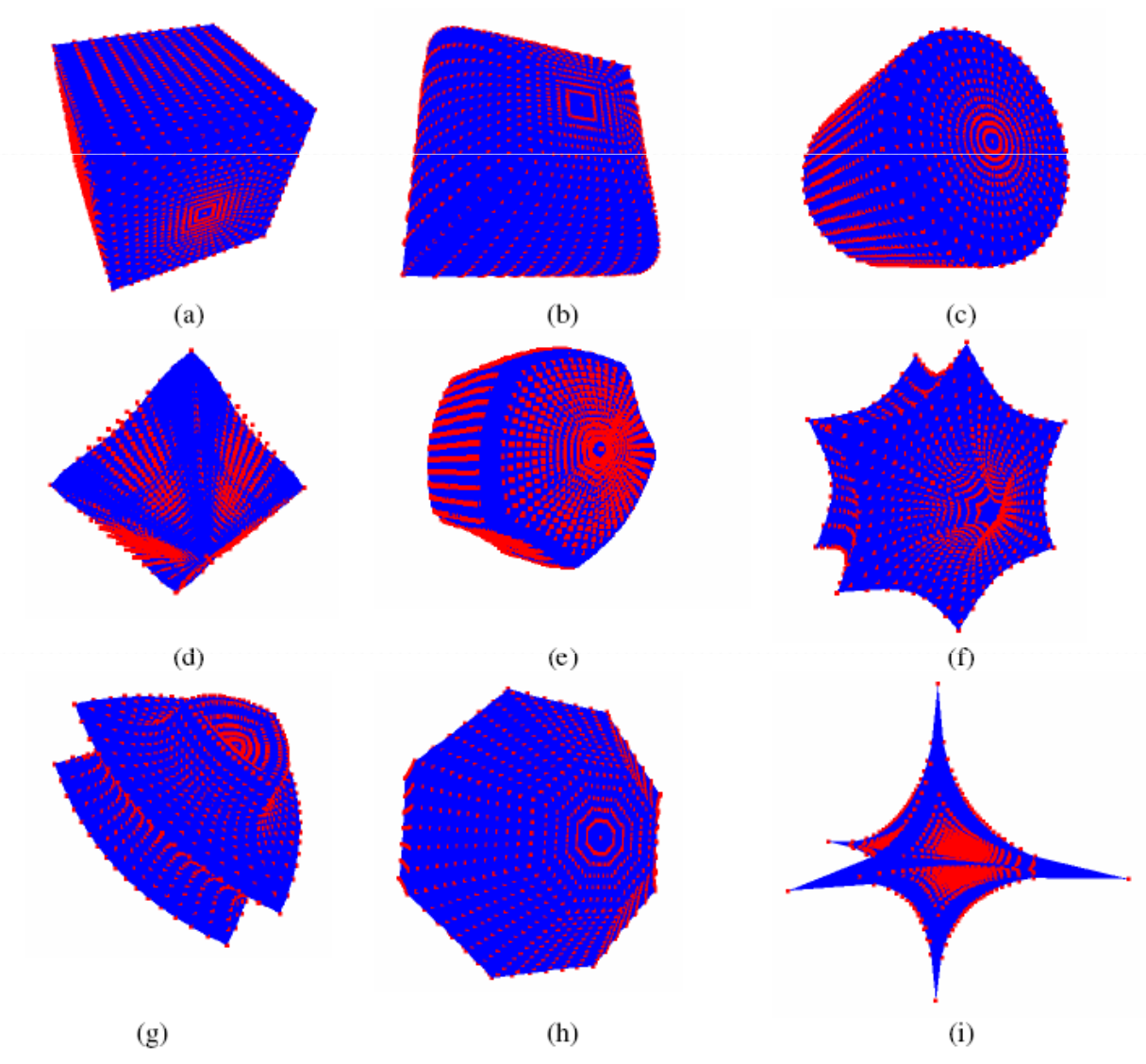
avec a, b et $n_i \in R^+$, et $m \in R_*^+$

Les paramètres a > 0 et b > 0 contrôlent les dimensions du super polygone, m ≥ 0 correspond au nombre de symétries et les n_i représentent les coefficients de forme.

Supershapes



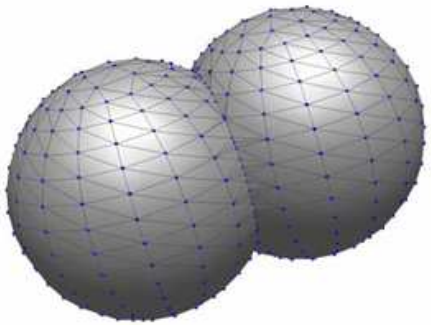
Fitting of the points cloud with a supershape



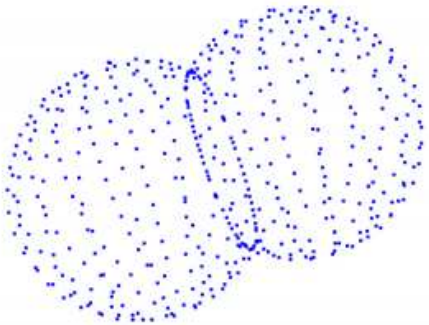
Results

figure	n_1	n_2	n_3	N_1	N_2	N_3	Erreur par méthode déterministe	Erreur par algorithme génétique
8.(a)	50/1680	50/1677	50/1637	50/196	50/196	50/196	0.04	0.008
8.(b)	50/1895	50/1845	50/1886	2/1918	2/3	2/16	0.05	0.005
8.(c)	2/1968	2/1.5	2/3	50/50	50/50	50/50	0.001	4.11e -7
8.(d)	1/0.6	1 /1.3	1/1.2	1/1.9	1/0.4	1/1.2	1.35	0.76
8.(e)	100/661	25/177	100/712	5/1330	5/693	5/754	3.54	0.24
8.(f)	100/76.2	100/76.2	100/76.2	100/76.2	100/76.2	100/76.2	0.02	0.0007
8.(g)	100/90	100/90	100/90	100/90	100/90	100/90	0.04	5.4e -5
8.(h)	1000/1086	250/268	250/288	1000/360	250/90	250/90	0.11	0.0008
8.(i)	0.5/0.508	0.5/0.508	0.5/0.508	0.5/0.508	0.5/0.508	0.5/0.508	32.5	2.017

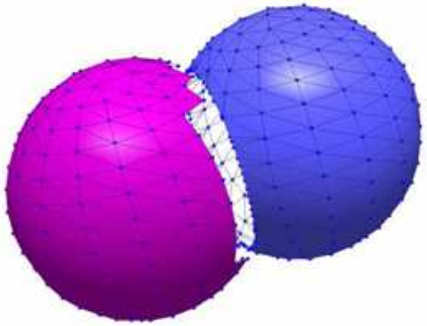
Assembly of forms



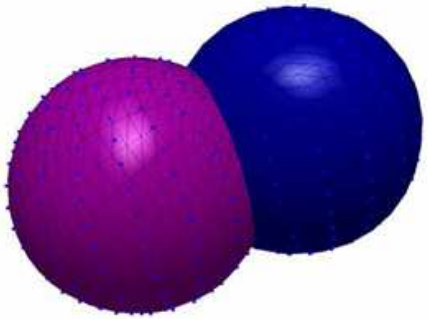
Maillage 3D original,



nuage de points réel,

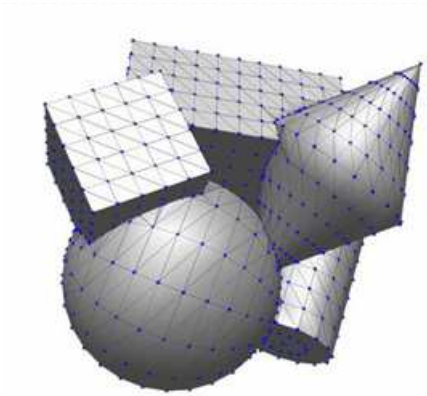


segmentation,

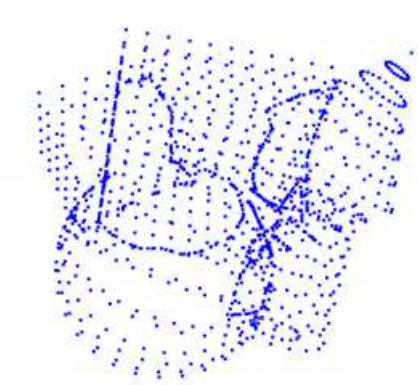


reconstruction

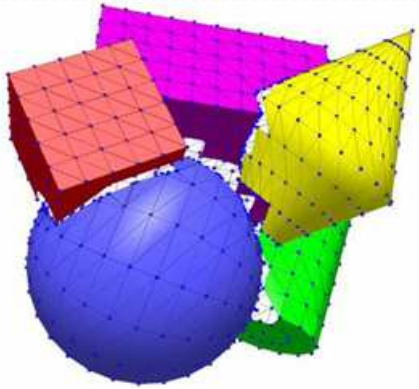
Assembly of forms



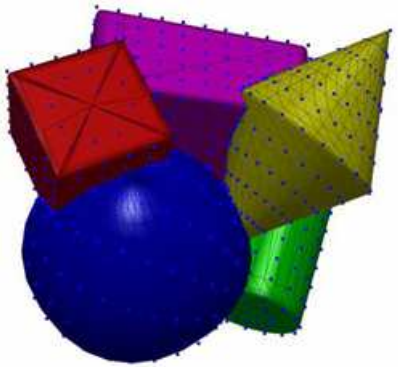
Maillage 3D original,



nuage de points réel,



segmentation,



reconstruction