Introduction to optimization and machine learning

Lesson 2 : basis of optimization

Sébastien Verel

Laboratoire d'Informatique, Signal et Image de la Côte d'opale (LISIC)
Université du Littoral Côte d'Opale, Calais, France
http://www-lisic.univ-littoral.fr/~verel/

April, 2021

General outline

- Introduction to optimization problems
- Introduction to machine learning
- Fundamentals of optimization methods
- Fundamentals of machine learning methods
- Practice of some algorithms using python

Outline of the day

- Numerical optimization :
  - White-box scenario : gradient descents
  - Black-box scenario : evolutionary algorithms
- Combinatorial optimization :
  - White-box scenario : tree search (branch & bound)
  - Black-box scenario : local search

Single-objective optimization

Definition

An optimization problem is a couple $(\mathcal{X}, f)$ with :

- Search space : set of candidate solutions

$$\mathcal{X}$$

- Objective fonction : quality criteria (or non-quality)

$$f : \mathcal{X} \to \mathbb{R}$$

Solve an optimization problem (minimization)

$$\mathcal{X}^\star = \text{argmin}_{\mathcal{X}} \ f$$

or find an approximation of $\mathcal{X}^\star$.

White-box optimization

Objective function f for $x \in \mathbb{R}^d$,

$$f(x) = \frac{x_2^3 e^{-0.4x_1}}{\sum_k e^{x_k}}$$

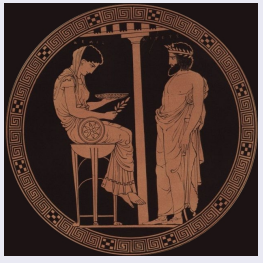
White-box optimization definition

Analytic expression of the objective function f is known

In this case, usually the objective fonction is :

- continuous, and differentiable (if we are lucky)

Black-box optimization

$x \longrightarrow$  $\longrightarrow f(x)$

No information on the objective function definition f

Objective fonction :

- can be irregular, non continuous, non differentiable . . .
- given by a computation or a simulation

Typology of optimization problems

Classification according to decision variables

- **Combinatorial optimisation** :
  search space is discrete (sometime finite) : NP-hard
- **Numerical optimization** :
  search space is subset of $\mathbb{R}^d$
- **Others** :
  discrete and numerical, program, morphology, topology, etc.

Classification according to information

- **White-box optimisation** :
  Some useful properties are known
- **Black-box optimization** :
  A minimum of *a priori* information is used
  Computation time can be expensive (simulator, in vivo, etc.)
- **Grey-box optimization** : in between

Numerical optimization

Definition : numerical optimization problem

An numerical optimization problem is a couple $(\mathcal{X}, f)$ with :

- Search space : set of candidate solutions

$$\mathcal{X} \text{ connected subset of } \mathbb{R}^d$$

- Objective fonction : quality criteria, minimization

$$f : \mathbb{R}^d \to \mathbb{R}$$

Minimization algorithms in white-box scenario

Bibliography

Daniele A. Di Pietro, Université de Montpellier,
Cours master 1, optimisation numérique :
`http://www.math.univ-montp2.fr/~di-pietro/Teaching.html`

Sebastian Ruder, An overview of gradient descent optimization algorithms, arXiv, 2017.
`http://sebastianruder.com/optimizing-gradient-descent/index.html`

Descent direction

Definition : descent direction

Let be $\mathcal{X} \subset \mathbb{R}^n$, $f : \mathcal{X} \to \mathbb{R}$, and $x \in \mathcal{X}$.

$w \in \mathcal{X} \setminus \{0\}$ is a descent direction in x when :
it exists a real number $\sigma_0 > 0$ such that :

$$\forall \sigma \in [0, \sigma_0], \quad f(x + \sigma\ w) \leqslant f(x)$$

Descent direction

Definition : descent direction

Let be $\mathcal{X} \subset \mathbb{R}^n$, $f : \mathcal{X} \to \mathbb{R}$, and $x \in \mathcal{X}$.

$w \in \mathcal{X} \setminus \{0\}$ is a descent direction in x when :
it exists a real number $\sigma_0 > 0$ such that :

$$\forall \sigma \in [0, \sigma_0], \quad f(x + \sigma\ w) \leqslant f(x)$$

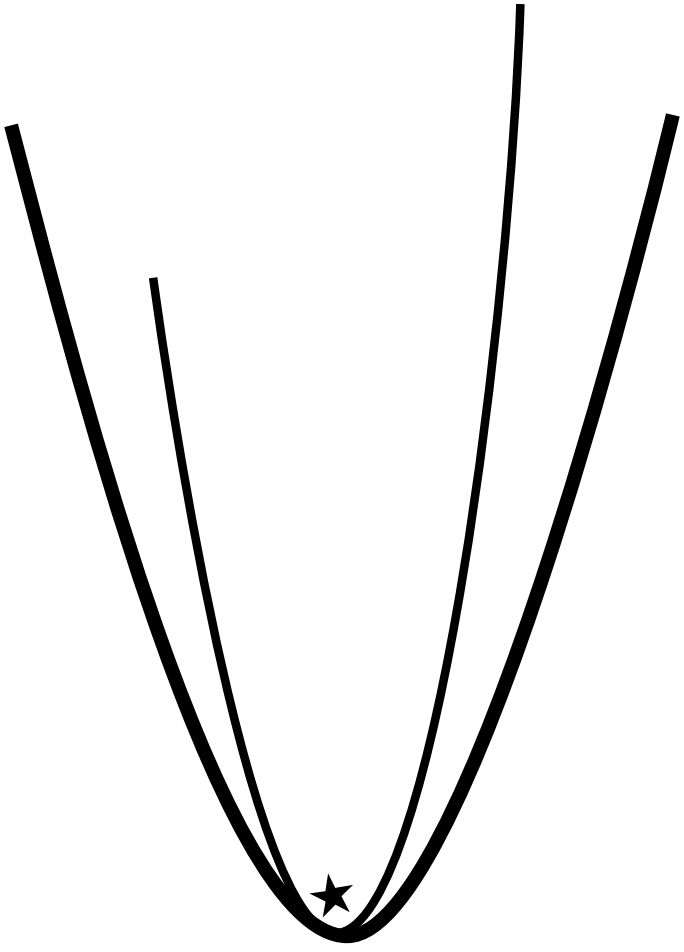
Definition : strict descent direction

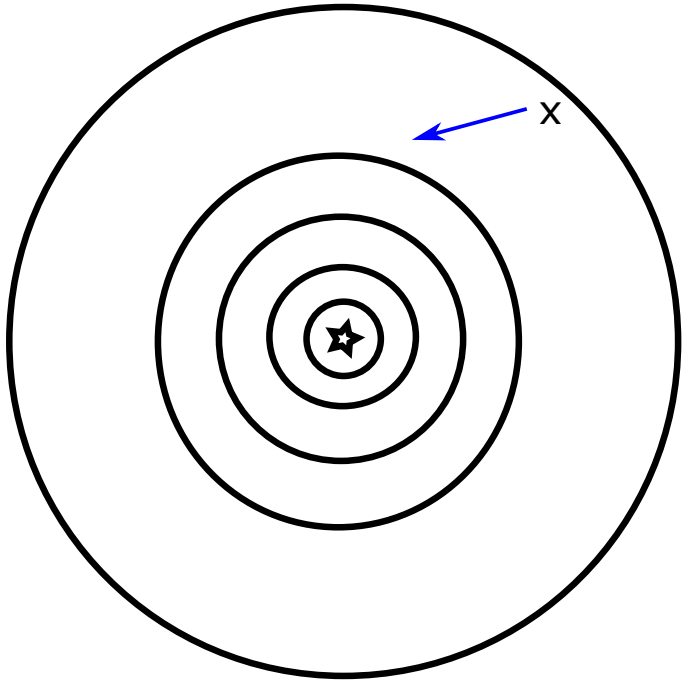
Let be $\mathcal{X} \subset \mathbb{R}^n$, $f : \mathcal{X} \to \mathbb{R}$, and $x \in \mathcal{X}$.

$w \in \mathcal{X} \setminus \{0\}$ is strict descent direction in x when :
it exists a real number $\sigma_0 > 0$ such that :

$$\forall \sigma \in [0, \sigma_0], \quad f(x + \sigma\ w) < f(x)$$

Descent direction





Descent algorithm

Descent Algorithm

Choose an initial solution $x \in \mathcal{X}$

repeat

Find a strict descent direction in x : $w \in \mathcal{X} \setminus \{0\}$

Choose a real number $\sigma > 0$

$x \leftarrow x + \sigma\ w$

until stopping criterium is false

Descent algorithm

Descent Algorithm

Choose an initial solution $x \in \mathcal{X}$

repeat

Find a strict descent direction in x : $w \in \mathcal{X} \setminus \{0\}$

Choose a real number $\sigma > 0$

$x \leftarrow x + \sigma\ w$

until stopping criterium is false

Open questions :

- How to choose the descent direction w as a function of x ?
- How to choose the step size σ ?
- How to define the stopping criterium ?

Gradient direction

Intuitions

from physic,

the **gradient** show the speed vector of the trajectory (surface),

i.e. the direction, the way, and the amplitude of the speed vector of the surface.

Formal definition

If f is differentiable in $x \in \mathbb{R}^d$,

the gradient of f in x is equal to :

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \dots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

Gradient : first example

$$f(x) = 2 + 4x_1 + x_2 + 2x_1^2 + 2x_1x_2 + x_1^2x_2$$

Gradient : first example

$$f(x) = 2 + 4x_1 + x_2 + 2x_1^2 + 2x_1x_2 + x_1^2x_2$$

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix}$$

with :

$$\frac{\partial f}{\partial x_1} = 4 + 4x_1 + 2x_2 + 2x_1x_2$$

$$\frac{\partial f}{\partial x_2} = 1 + 2x_1 + x_1^2$$

Gradient : Mean Square Error example

Multiple linear regression on data $\{(x_i, y_i) \mid i \in \{1, \ldots, n\}\}$ with linear model $m_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

Mean Square Error function :

$$f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (m_\beta(x_i) - y_i)^2$$

$$f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} - y_i)^2$$

Gradient : Mean Square Error example

Multiple linear regression on data $\{(x_i, y_i) \mid i \in \{1, \ldots, n\}\}$ with linear model $m_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

Mean Square Error function :

$$f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (m_\beta(x_i) - y_i)^2$$

$$f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} - y_i)^2$$

$$\nabla f(\beta) = \begin{pmatrix} \frac{\partial f}{\partial \beta_0} \\ \frac{\partial f}{\partial \beta_1} \\ \frac{\partial f}{\partial \beta_2} \end{pmatrix}$$

Gradient : Mean Square Error example

Multiple linear regression on data $\{(x_i, y_i) \mid i \in \{1, \ldots, n\}\}$ with linear model $m_\beta(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$

Mean Square Error function :

$$f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (m_\beta(x_i) - y_i)^2$$

$$f(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} - y_i)^2$$

with :

$$\frac{\partial f}{\partial \beta_0} = \frac{1}{n} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} - y_i)$$

$$\frac{\partial f}{\partial \beta_1} = \frac{1}{n} \sum_{i=1}^{n} x_{i,1} (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} - y_i)$$

$$\frac{\partial f}{\partial \beta_2} = \frac{1}{n} \sum_{i=1}^{n} x_{i,2} (\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} - y_i)$$

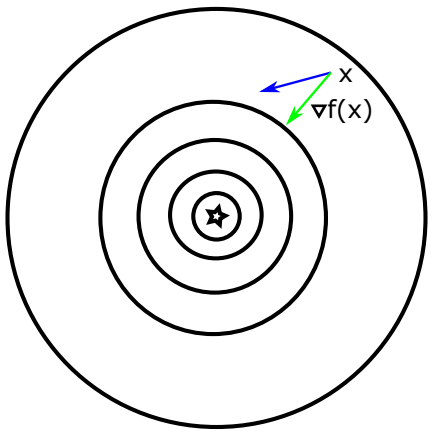
Gradient, and descent directions

Result

Let be f a continuously differentiable function on open set with $x \in \mathbb{R}$. The notation . is the scalar production on $\mathbb{R}^d$.

If $w \in \mathcal{X} \setminus \{0\}$ is a descent direction,
then $\nabla f(x).w \leqslant 0$
gradient vector is at the opposite direction of descent direction.

If $\nabla f(x) \neq 0$,
then $w = -\nabla f(x)$ is strict descent direction in x.



Method of gradient descent

Algorithm of gradient descent

Choose initial solution $x \in \mathcal{X}$

repeat

$w \leftarrow -\nabla f(x)$

Choose a real number $\sigma > 0$

$x \leftarrow x + \sigma\ w$

until stopping criterium is false

Method of gradient descent

Algorithm of gradient descent

Choose initial solution $x \in \mathcal{X}$

repeat

$w \leftarrow -\nabla f(x)$

Choose a real number $\sigma > 0$

$x \leftarrow x + \sigma\ w$

until stopping criterium is false

Open questions :

- How to choose the step size σ ?
- How to define the stopping criterium ?

Gradient descent with fix step size

Algorithm of gradient descent with fix step size

Choose a step size $\sigma \in \mathbb{R}^+$
Choose initial solution $x \in \mathcal{X}$
repeat
$w \leftarrow -\nabla f(x)$
$x \leftarrow x + \sigma\ w$
until stopping criterium is false

From the code `exo1.ipynb` (jupyter notebook),

- Define the gradient function of the two examples
- Code the gradient descent algorithm
- Test the gradient descent on the two examples

Gradient descent with fix step size

Algorithm of gradient descent with fix step size

Choose a step size $\sigma \in \mathbb{R}^+$
Choose initial solution $x \in \mathcal{X}$
repeat
  $w \leftarrow -\nabla f(x)$
  $x \leftarrow x + \sigma\ w$
until stopping criterium is false

Open questions :

- How to choose the step size σ ?
- How to define the stopping criterium ?

Newton Method (1669)

Newton algorithm (dimension 1)

Choose initial solution $x \in \mathcal{X}$

repeat

$w \leftarrow \frac{-1}{f''(x)} f'(x)$

$x \leftarrow x + w$

until stopping criterium is false

Newton algorithm (dimension n)

Choose initial solution $x \in \mathcal{X}$

repeat

$w \leftarrow -[H(f)(x)]^{-1} \nabla f(x)$

$x \leftarrow x + w$

until stopping criterium is false

H is the Hersian matrix (matrix of second order partial derivatives)

Variants and improvements of gradient descent

- (Batch) gradient descent :
  $\nabla f(\theta) = \mathbb{E}_{j \in 1 \dots p}[\frac{\partial f}{\partial \theta}(\theta; x^{(j)}, y^{(j)})]\,; \quad \theta \leftarrow \theta + \sigma\ \nabla f(\theta)$
- Stochastic gradient descent :
  $\nabla f(\theta; j) = \frac{\partial f}{\partial \theta}(\theta; x^{(j)}, y^{(j)})\,;$
  $\forall j$ rnd order, $\theta \leftarrow \theta + \sigma\ \nabla f(\theta; j)$
- Momentum gradient descent :
  $v_t = \gamma v_{t-1} + \sigma \nabla f(\theta)\,; \quad \theta \leftarrow \theta - v_t$
- Nesterov accelerated gradient descent (NAG) :
  $v_t = \gamma v_{t-1} + \sigma \nabla f(\theta - \gamma v_{t-1})\,; \quad \theta \leftarrow \theta - v_t$
- Adagrad gradient descent :
  $g_{t,i} = \nabla_i f(\theta)\,; \ \ G_{t,ii} = \sum_{t' \leqslant t} g_{t',i}^2\,; \ \ \forall i,\ \theta_i \leftarrow \theta_i - \frac{\sigma}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i}$
- AdaDelta gradient descent :
  $E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2\,;$
  $E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma) \Delta\theta_t^2\,;$
  $\Delta\theta_t = -\frac{\sqrt{E[\Delta\theta^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} g_t\,; \ \ \theta_t \leftarrow \theta_{t-1} + \Delta\theta_t$

Variants and improvements of gradient descent

- Adam gradient descent :
  $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$; $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$;
  $\hat{m}_t = m_t/(1-\beta_1^t)$; $\hat{v}_t = v_t/(1-\beta_2^t)$; $\theta_{t+1} = \theta_t - \frac{\sigma}{\sqrt{v_t}+\epsilon}\hat{m}_t$
- AdaMax gradient descent :
  $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$; $v_t = \max(\beta_2 v_{t-1}, |g_t|)$;
  $\hat{m}_t = m_t/(1-\beta_1^t)$; $\theta_{t+1} = \theta_t - \frac{\sigma}{v_t}\hat{m}_t$
- Nadam gradient descent :
  $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$; $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$;
  $\hat{m}_t = m_t/(1-\beta_1^t)$; $\hat{v}_t = v_t/(1-\beta_2^t)$;
  $\theta_{t+1} = \theta_t - \frac{\sigma}{\sqrt{v_t}+\epsilon}(\beta_1 \hat{m}_t + \frac{(1-\beta_1) g_t}{1-\beta_1^t})$

Black-box optimization scenario

$x \longrightarrow$  $\longrightarrow f(x)$

No information on the objective function definition f

Objective fonction :

- can be irregular, non continuous, non differentiable . . .
- given by a computation or a simulation

Optimization methods

- Bayesian optimization :
  Jonas Mockus, 1970 - 1980, well-known Kriging method
  0. Function is represented as random function
  1. Assume a prior of the behavior of function
  2. Sample some x
  3. Update the post-perior the distribution
- Evolutionary algorithm :
  Bio-inspired algorithms
  (genetic algorithm, evo. strategy, genetic prog., etc.)
  ES : Ingo Rechenberg, Hans-Paul Schwefel, early 1960

Introduction to evolution strategy

Bibliography :

Summer school on artificial evolution

Anne Auger, june 2012 :
`https://sites.google.com/site/ecoleea2012/programme`

Evolutionary algorithm : evolution strategy

Evolutionary algorithm

Choose initial population Parents *i.e. set of solutions*

repeat

Genitors = select(Parents)

Children = random variation (Genitors)

Parents = replacement(Genitors, Parents)

until stopping criterium is false

Evolution strategy (continuous optimization)

Initialize distribution parameter θ

repeat

Sample population $(x_1, \ldots, x_\lambda)$ using distribution $P(x|\theta)$

Evaluate $(x_1, \ldots, x_\lambda)$ on f

Update parameter $\theta = F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda))$

until stopping criterium is false

$(1+1)$-Evolution Strategy

Basic idea

Parameter $\theta = m$:

current position of the best known candidate solution

Iterate :

1. Sample one solution "around" m
2. If better, update parameter m

$(1+1)$-Evolution Strategy

Basic idea

Parameter $\theta = m$:

current position of the best known candidate solution

Iterate :

1. Sample one solution "around" m
2. If better, update parameter m

Basic version of $(1+1)$-Evolution Strategy

Choose initial mean $m \in \mathbb{R}^d$
repeat
  $x' \leftarrow Norm(m, \sigma^2)$
  if $f(x')$ is better than $f(m)$ **then**
    $m \leftarrow x'$
  end if
until stopping criterium is false

$(1+1)$-Evolution Strategy

Basic version of $(1+1)$-Evolution Strategy

Choose initial mean $m \in \mathbb{R}^d$
repeat
  $x' \leftarrow Norm(m, \sigma^2)$
  if $f(x')$ is better than $f(m)$ **then**
    $m \leftarrow x'$
  end if
until stopping criterium is false

From the previous jupyter notebook,

- Code the basic $(1+1)$-ES
- Test the code on the 2 examples with different step sizes

$(1+1)$-Evolution Strategy

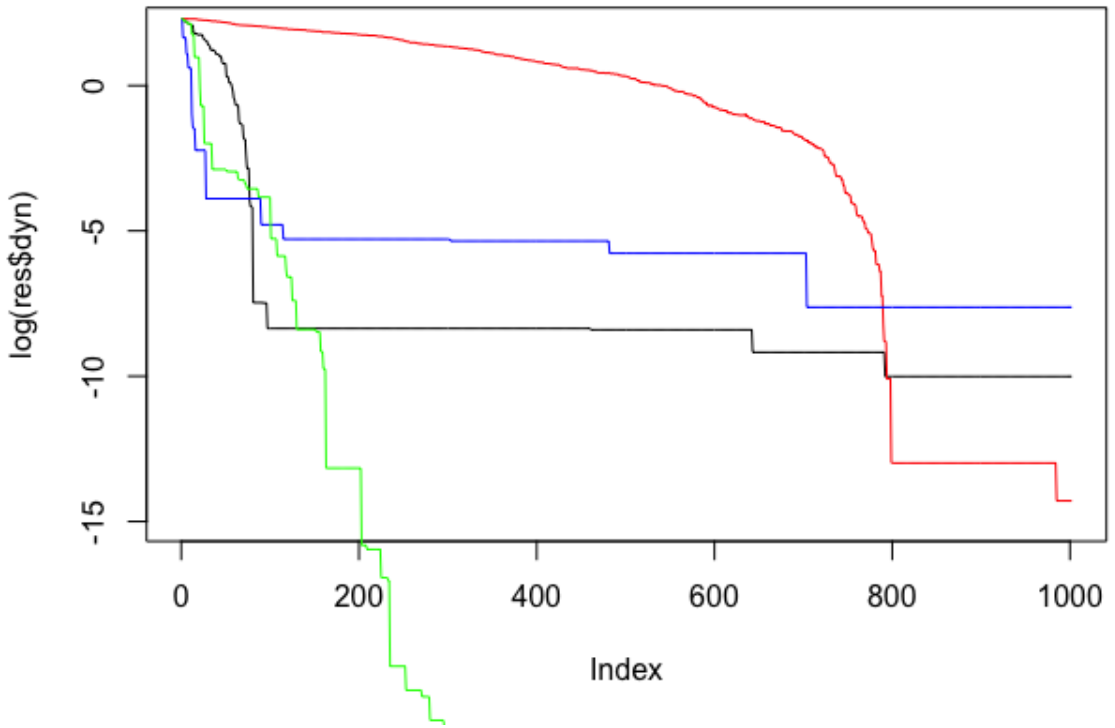
$(1+1)$-ES

Choose randomly initial mean $m \in \mathbb{R}^d$
repeat
  $x' \leftarrow Norm(m, \sigma.C)$
  if $f(x')$ is better than $f(m)$ **then**
    $m \leftarrow x'$
  end if
until stopping criterium is false

Parameters of the algorithm :
  $\sigma \in \mathbb{R}$: step size
  Matrice $C \in \mathbb{R}^{d \times d}$: covariance matrix

Open questions :
  How to choose the step size ?
  How to choose the covariance matrix ?

Search dynamic according to step size



- black : $\sigma = 0.1$
- red : $\sigma = 0.01$
- blue : $\sigma = 0.5$
- green : adaptive σ

$(1+1)$-Evolution Strategy with One-fifth success rule

$(1+1)$-Evolution Strategy with $1/5$ success rule

Choose randomly initial solution $m \in \mathbb{R}^n$
repeat
  $x' \leftarrow m + \sigma\, \mathcal{N}(0, C)$
  if x' is better than m **then**
    $m \leftarrow x'$
    $\sigma \leftarrow \sigma \times exp(1/3)$
  else
    $\sigma \leftarrow \sigma / exp(1/3)^{1/4}$
  end if
until stopping criterium is false

From the previous jupyter notebook,

- Code the basic $(1+1)$-ES with $1/5$ success rule
- Compare the results with fix step size

Larger populations : $(\mu/\mu, \lambda)$-Evolution Strategy

$(\mu/\mu, \lambda)$-ES

Choose randomly initial mean $m \in \mathbb{R}^n$

repeat

  for $i \in \{1 \dots \lambda\}$ **do**

    $x_i^{'} \leftarrow m + \sigma\, \mathcal{N}(0, C)$

    Evaluate $x_i^{'}$ with f

  end for

  Select the μ best solutions from $\{x_1^{'}, \dots, x_\lambda^{'}\}$

  Let be $x_{:j}$ those solutions ranking by increasing order of f :

$$f(x_{:1}) \leq \dots \leq f(x_{:\mu})$$

  $m \leftarrow \sum_{j=1}^{\mu} w_j x_{:j}^{'}$

until stopping criterium is false

avec $\hat{w}_i = \log(\mu + 0.5) - \log(i)$ et $w_i = \hat{w}_i / \sum_{i=1}^{\mu} \hat{w}_i$

Advanced Evolution Strategy : CMA-ES

The CMA-ES

Input: $m \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ

Initialize: $\mathbf{C} = \mathbf{I}$, and $p_c = \mathbf{0}$, $p_\sigma = \mathbf{0}$,

Set: $c_c \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \leq 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$, and $w_{i=1\ldots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \approx 0.3\,\lambda$

While not terminate

$$\boldsymbol{x}_i = \boldsymbol{m} + \sigma \boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \quad \text{for } i = 1, \ldots, \lambda \qquad \text{sampling}$$

$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda} = \boldsymbol{m} + \sigma \boldsymbol{y}_w \quad \text{where } \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \qquad \text{update mean}$$

$$p_c \leftarrow (1 - c_c)\, p_c + \mathbb{1}_{\{\|p_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_c)^2}\, \sqrt{\mu_w}\, \boldsymbol{y}_w \qquad \text{cumulation for } \mathbf{C}$$

$$p_\sigma \leftarrow (1 - c_\sigma)\, p_\sigma + \sqrt{1 - (1 - c_\sigma)^2}\, \sqrt{\mu_w}\, \mathbf{C}^{-\frac{1}{2}} \boldsymbol{y}_w \qquad \text{cumulation for } \sigma$$

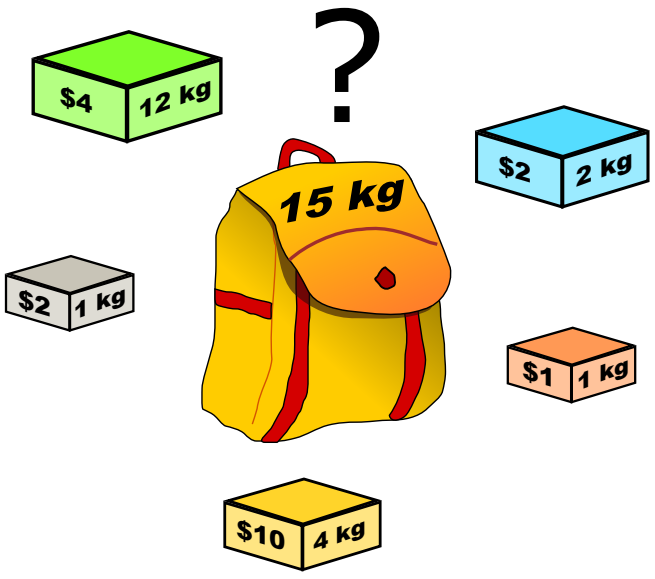
$$\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\, \mathbf{C} + c_1\, p_c p_c^{\mathrm{T}} + c_\mu \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}} \qquad \text{update } \mathbf{C}$$

$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{\mathrm{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right) \qquad \text{update of } \sigma$$

Not covered on this slide: termination, restarts, useful output, boundaries and encoding

Example of White-box optimization : Knapsack problem

How to pack the most interesting objects with respect to capacity ?



source : wikipedia

Example of White-box optimization : Knapsack problem

Knapsack problem :

For n objects, profits $(p_i)_{i\in\{1\dots n\}}$, weights $(w_i)_{i\in\{1\dots n\}}$

binary representation : object i is in $x_i = 1$, object out $x_i = 0$.

Maximize

$$P(x) = \sum_{i=1}^{n} p_i x_i$$

such that :

$$W(x) = \sum_{i=1}^{n} w_i x_i \leqslant C$$

$$\forall i \in \{1 \dots n\}, x_i \in \{0,1\}^n$$

Complexity : NP-hard problem

How many candidate solutions as function of objects n ?

To be compare to the number of atoms in the universe...

Complexity : NP-hard problem

How many candidate solutions as function of objects n ?

To be compare to the number of atoms in the universe...

How long for a full enumeration of the search space for $n = 1000$?

Suppose that we can check one single solution in $1\mu s$.

Search algorithms

Principle

(implicite) **enumeration of a subset of the search space**

- Many ways to enumerate the search space
  - *Exact methods* :
    pros : Methods able to find the optimal solution
    cons : Computational complexity
    ex : $A^\star$, Branch&Bound ...
  - *Random sampling* :
    pros : Computational complexity
    cons : No guarantee to find the optimal solution
    ex : Monte Carlo, approximation with guarantee, bayesian optimization, local search, evolutionary algorithms, ...

Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.

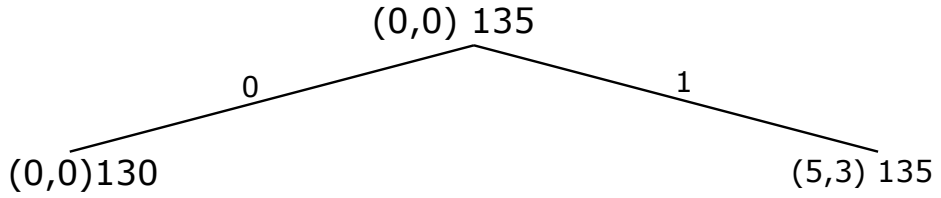
(0,0) 135
(P,W) Upper Bound

Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.

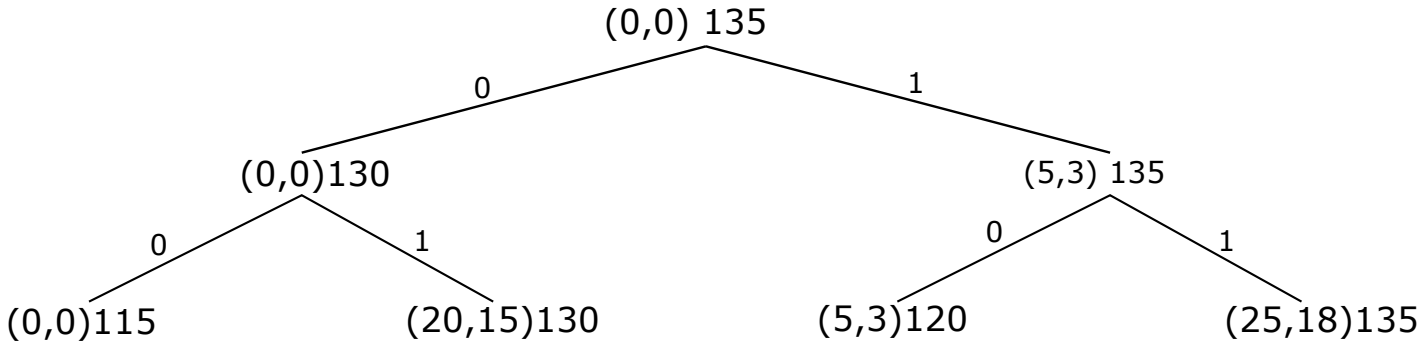


Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.

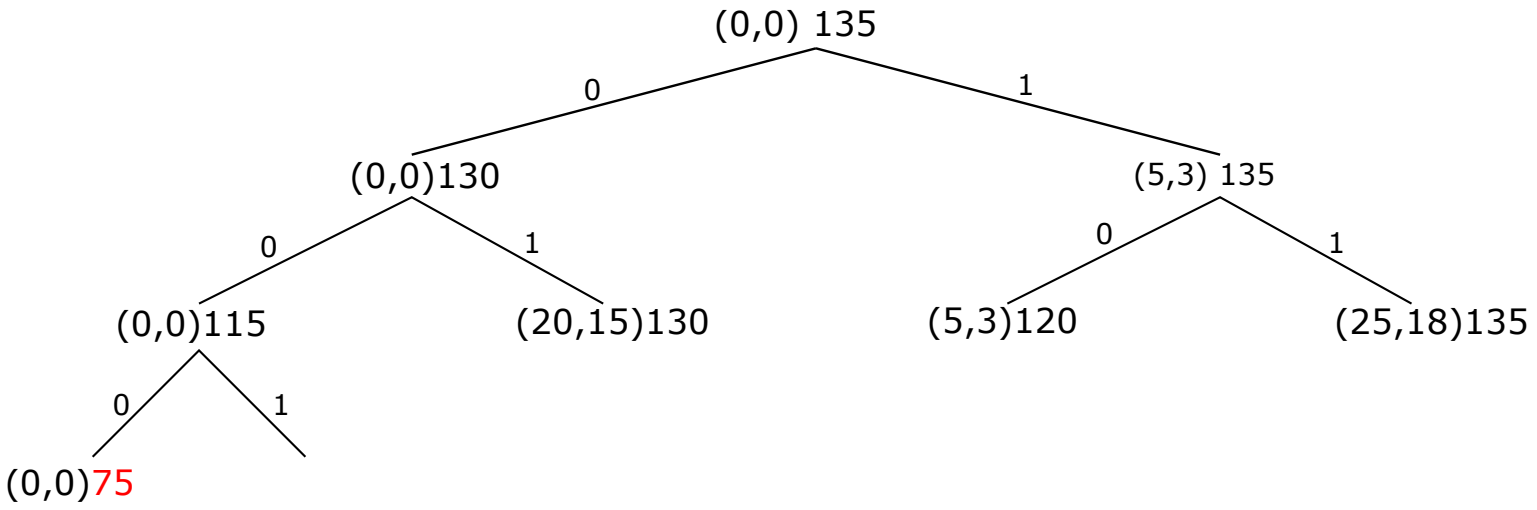


Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.

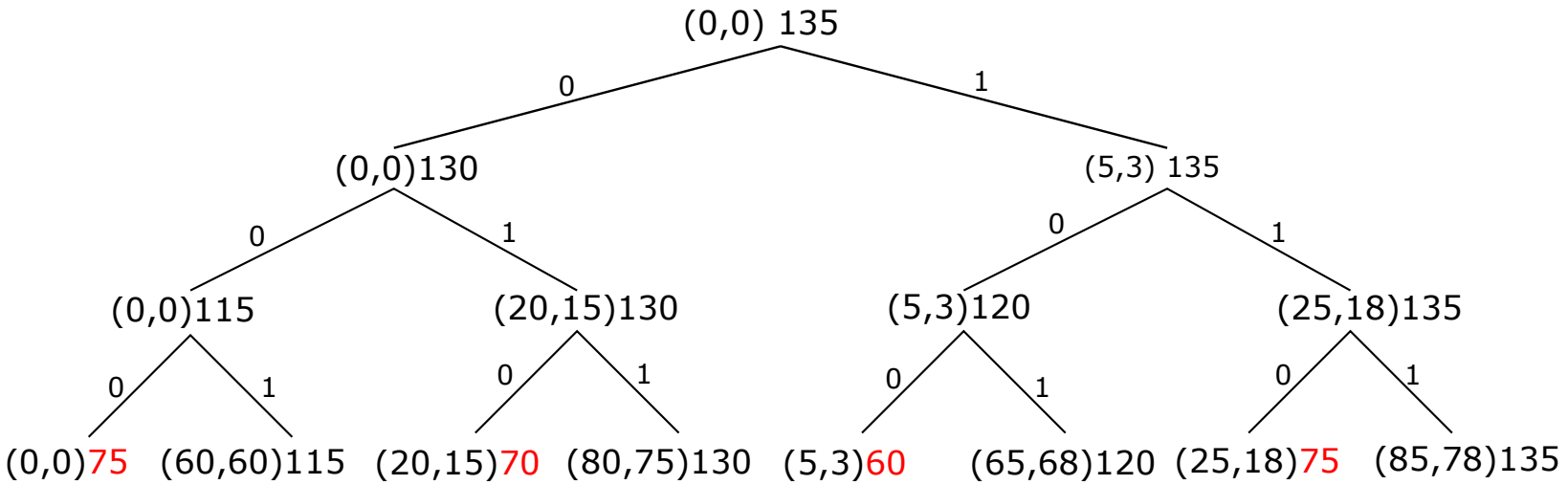


Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.

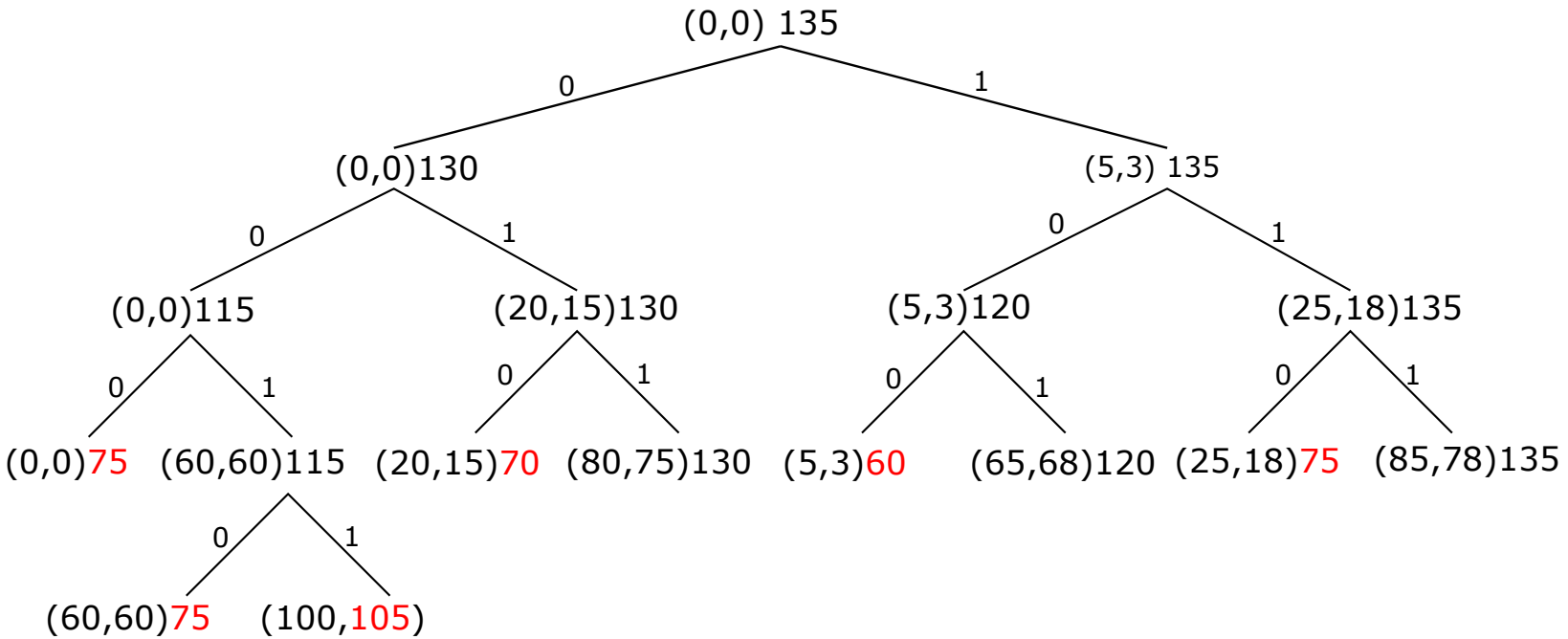


Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.

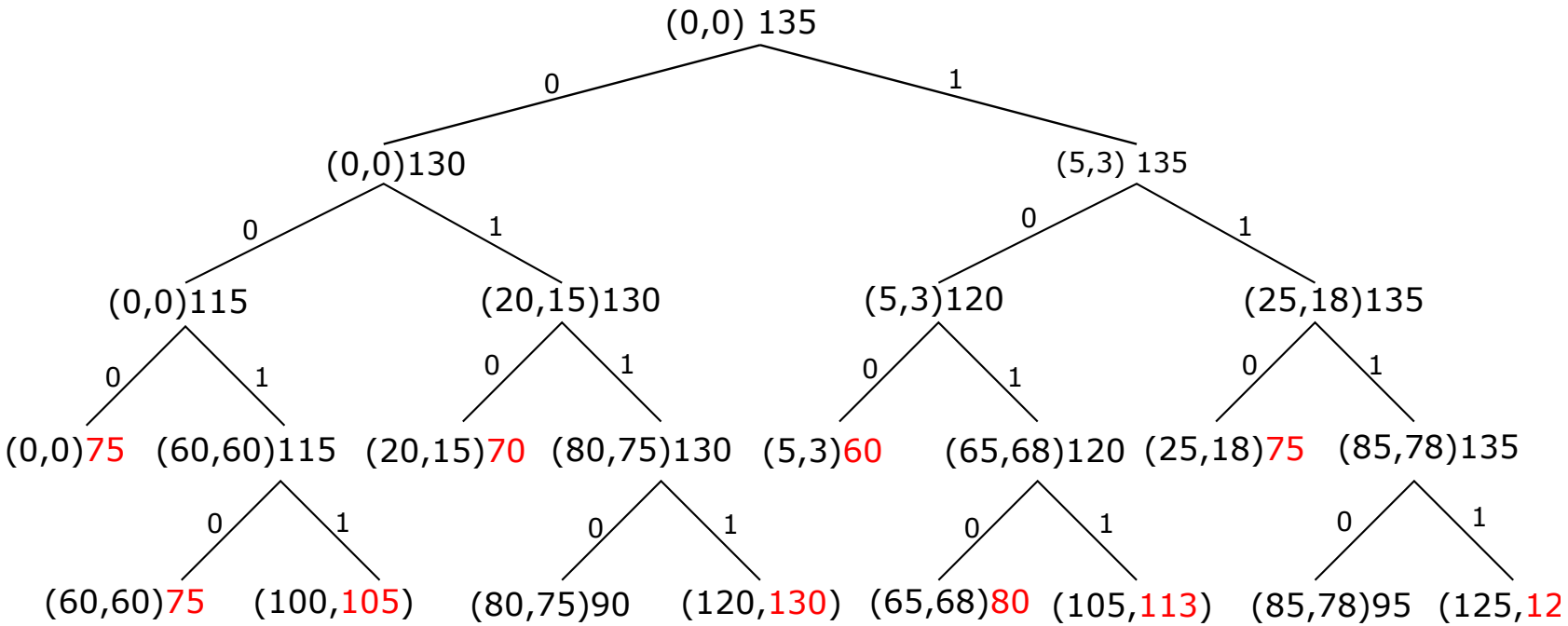


Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.

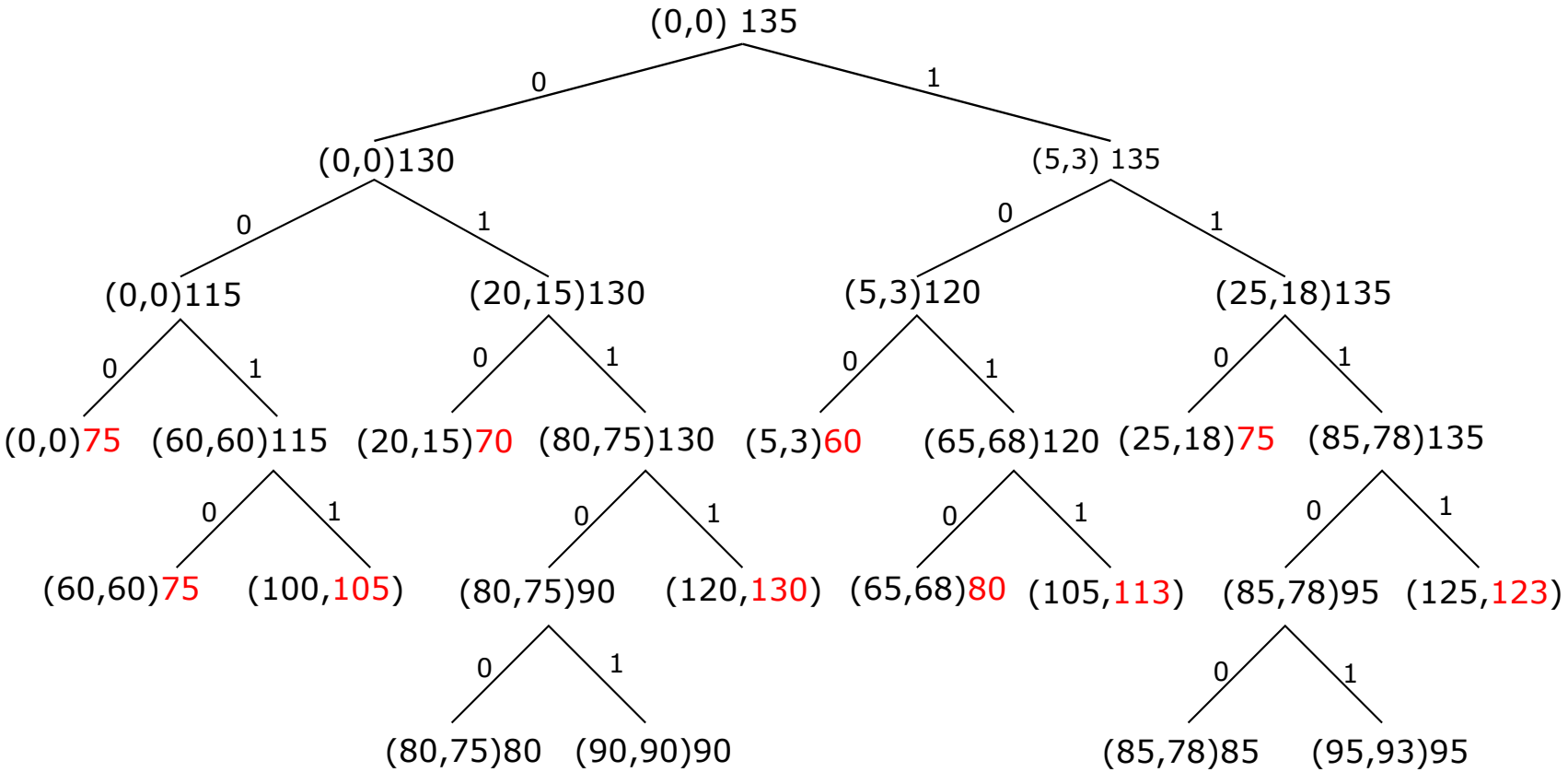


Branch and Bound algorithm

Maximal capacity $C = 100$ with objects ranked by the ratio p_i/w_i :

i	1	2	3	4	5
p_i	5	20	60	40	10
w_i	3	15	60	45	15

With a greedy algorithm : $x = 11100$ with $P = 85$, and $W = 78$.



Branch and Bound algorithm

Ailsa Land and Alison Doig, 1960

Keys elements

- Branch :
  Use a strategy to select the branching node
- Bound :
  Use the white-box model to compute a sharp upper bound
- Initialization :
  Use an efficient heuristic to find a good initial solution

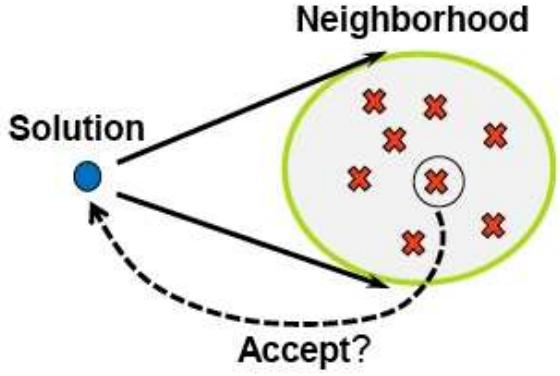
Search algorithms

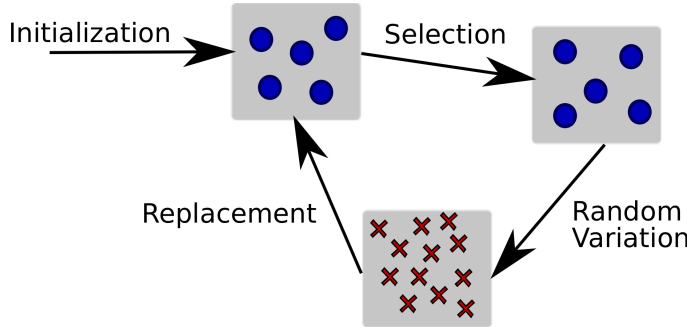
Principle

(implicite) **enumeration of a subset of the search space**

- Many ways to enumerate the search space
  - *Exact methods* : $A^\star$, Branch&Bound ...
  - *Random sampling* : Monte Carlo, approximation with guarantee, bayesian optimization, ...

Local search / Evolutionary algorithms





Play the game

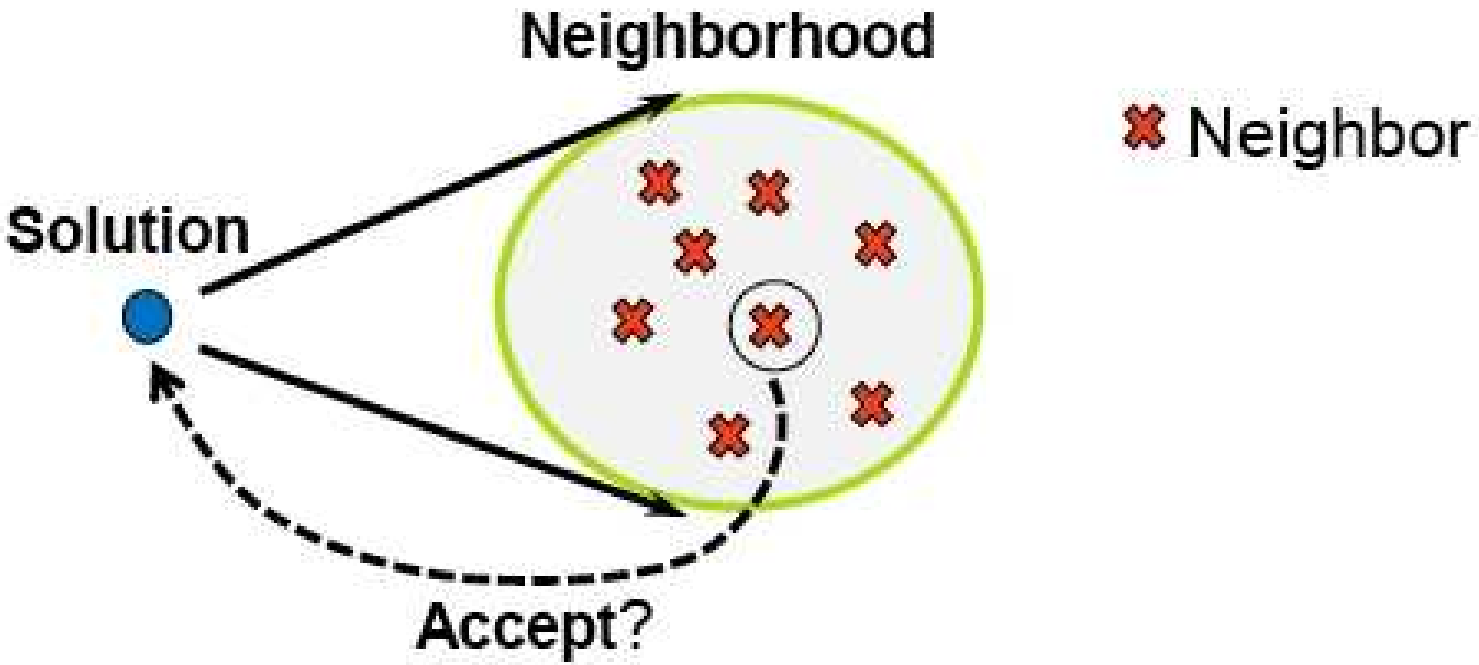
Serious game

homepage → research → *"A game to understand fitness landscape"*

`https://www-lisic.univ-littoral.fr/~verel/RESEARCH/notSeriousGame/index.html`

Stochastic algorithms with a single solution (Local Search)

- X set of candidate solutions (the search space)
- $f : X \to \mathbb{R}$ objective function
- $\mathcal{N}(x)$ set of neighboring solutions from x



Local search algorithm

- X set of candidate solutions (the search space)
- $f : X \rightarrow \mathbb{R}$ objective function
- $\mathcal{N}(x)$ set of neighboring solutions from x

Local Search algorithm

Choose initial solution $x \in \mathcal{X}$
repeat
  choose $x' \in \mathcal{N}(x)$
  if accept(x, x') **then**
    $x \leftarrow x'$
  end if
until stopping criterium

Hill-Climber Best Improvement (ou steepest-descent)

Hill Climber (best-improvement)

Choose initial solution $x \in \mathcal{X}$
Evaluate x with f
repeat
  Choose $x' \in \mathcal{N}(x)$ such that $f(x')$ is maximal
  if x' strictly better than x **then**
    $x \leftarrow x'$
  end if
until x is a optimum local

Greedy algorithm :
at each step of the search, from all available neighbor's solutions the algorithm selects the best one.

Drawback : stop in local optima

Exercice : design an Hill-Climbing algorithm

How to deal with constraints ?

- Remove/avoid non feasible solutions
- Repair mechanism of non feasible solutions
- Penalty methods : a penalty when the contraint is not satisfied
- Use an heuristic to generate only feasible solutions

With Knapsack problem, linear penalty method :

$$f(x) = \begin{cases} P(x) & \text{si } W(x) \leqslant C \\ P(x) - \beta \times (W(x) - C) & \text{si } W(x) > C \end{cases}$$

From the code `exo2.ipynb`,

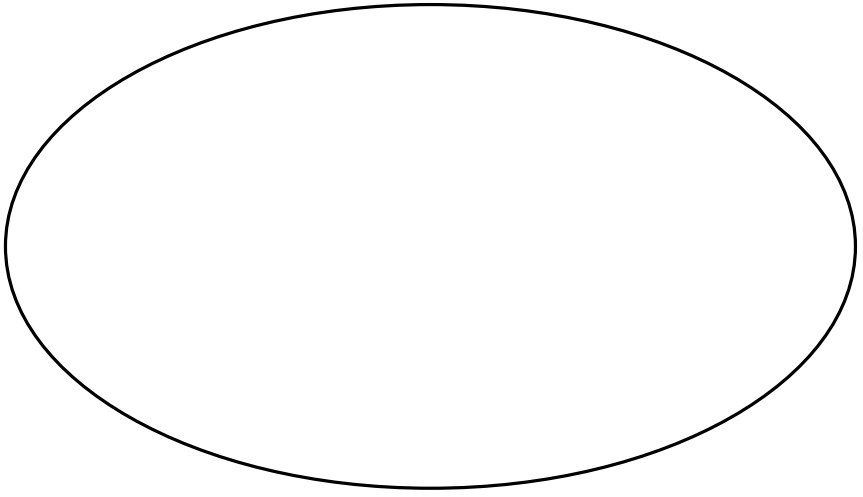
- Define the Hill-Climber algorithm with best improvement pivot rule

Main idea behind local search strategy

Why using a local search strategy based on neighborhood ?

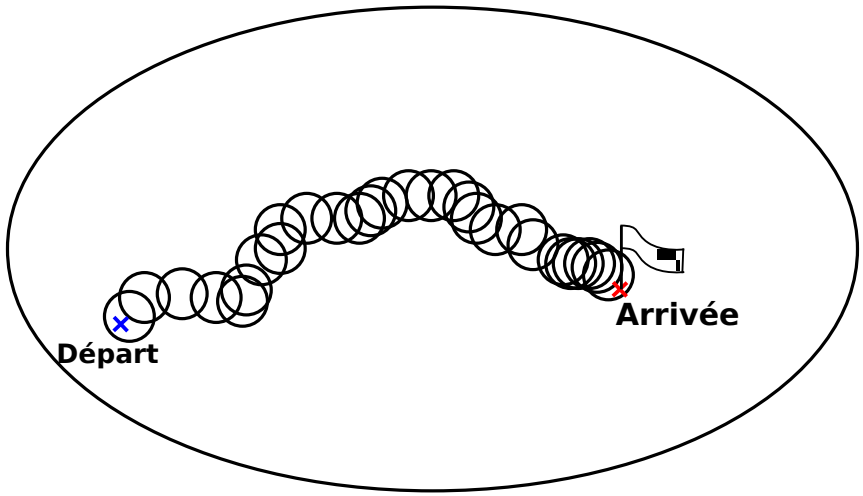
Main idea behind local search strategy

Why using a local search strategy based on neighborhood ?



Main idea behind local search strategy

Why using a local search strategy based on neighborhood ?

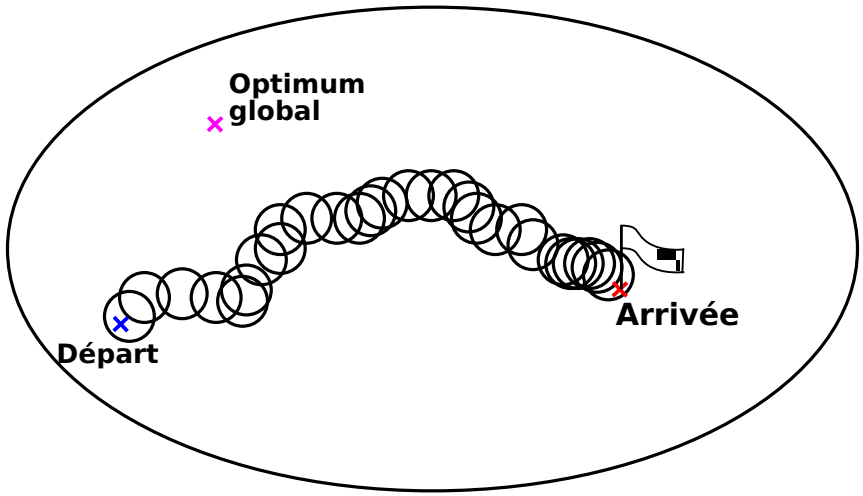


Reduce a search on large set
to a sequence of searches on small sets

- Pros : reduce the complexity

Main idea behind local search strategy

Why using a local search strategy based on neighborhood ?



Reduce a search on large set
to a sequence of searches on small sets

- Pros : reduce the complexity
- Cons : stop in sub-optimal solution

Local et global optimum

Local optimum

Let be $(X, f, \mathcal{N})$, f to maximize.

$x^\star$ is a local optimum iff for all $x \in \mathcal{N}(x^\star)$, $f(x) \leq f(x^\star)$

Strict local optimum

Let be $(X, f, \mathcal{N})$, f to maximize.

$x^\star$ is a strict local optimum iff for all $x \in \mathcal{N}(x^\star)$, $f(x) < f(x^\star)$

Global optimum

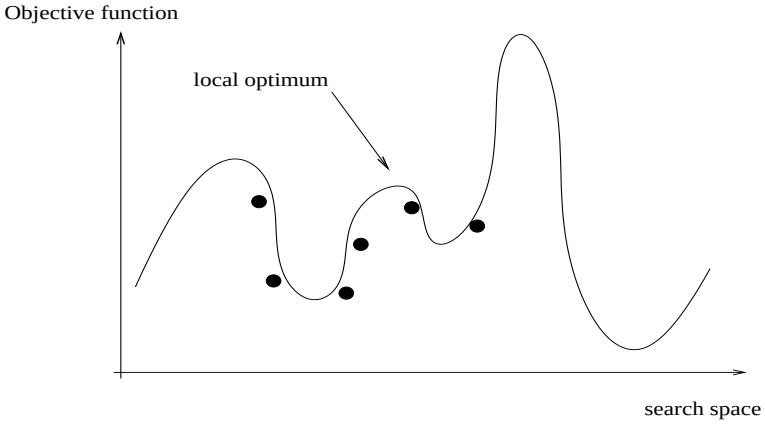
Let be $(X, f, \mathcal{N})$, f to maximize.

$x^\star$ is a global optimum iff for all $x \in X$, $f(x) \leq f(x^\star)$

Metaheuristics

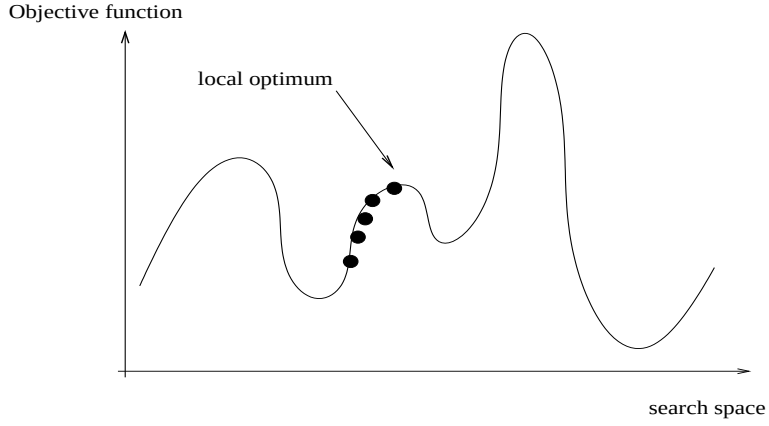
Random search / Hill Climbing

Random search



maximal exploration,
diversification

Hill-climbing



maximal exploitation,
intensification

Tradeoff between Exploration / Exploitation

Escape from local optima, etc.

$\Longrightarrow$ simulated annealing, tabu search, Iterated Local Search

Advanced local search : metaheuristics

- **Simulated Annealing (SA)** : Kirkpatrick *et al* (IBM 1983),
  Probability to select non-improving neighbors
- **Tabu Search (TS)** : F. GLover 1986,
  Use memory to not search again on the same solutions
- **Iterated Local Search (ILS)** :
  Restart strategy from local optima (perturbation)
- **Variable Neighborhood Search (VNS)** :
  Use a family of neighborhoods
- **Local Search with Evolutionary Algorithms** :
  Population + crossover + local search
- **Local search with exact methods** :
  For ex. exact method on subspaces
- **Local Search with machine learning techniques** :
  ML to guide the search on LS