Modélisation et Simulation : Optimisations

Gaëtan Hello

Université d'Evry Val d'Essonne
UFR Sciences et Technologies
gaetan.hello@ufrst.univ-evry.fr

L1 SPI

1. Optimisation continue
   - Principes élémentaires de l'optimisation continue
   - Applications à des fonctions-objectif de $\mathbb{R} \to \mathbb{R}$
   - Méthodes de résolution numérique

2. Optimisation combinatoire
   - Exemple : "problème du voyageur de commerce"
   - Complexité des algorithmes
   - Heuristiques

1. Optimisation continue
   - Principes élémentaires de l'optimisation continue
   - Applications à des fonctions-objectif de $\mathbb{R} \to \mathbb{R}$
   - Méthodes de résolution numérique

2. Optimisation combinatoire

Vocabulaire

De nombreux problèmes en sciences physiques et de l'ingénieur peuvent s'envisager au travers de la recherche d'un état assurant la maximisation ou la minimisation d'une quantité d'intérêt appropriée. Le vocabulaire suivant est alors employé :

- *fonction-objectif* $f : \mathbb{R}^n \to \mathbb{R}$:
  il s'agit de la quantité d'intérêt scalaire pertinente pour le problème envisagé (ex : puissance, rendement, masse, rigidité ...),
- *paramètres d'optimisation* $\underline{x} \in \mathbb{R}^n$:
  ce sont les n variables scalaires influençant f (ex : coordonnées, épaisseurs, modules de matériaux, ...),
- *domaine de conception* D :
  espace auquel appartiennent les paramètres d'optimisation,
- *minimisation/maximisation* :
  l'optimisation consiste en pratique à déterminer une valeur de $\underline{x}$ conduisant à minimiser/maximiser f sur D.

Ecriture formelle du problème d'optimisation

Sans perte de généralité, on considérera désormais que le problème d'optimisation consiste en un problème de minimisation (maximiser f revenant à minimiser $-f$).

Le problème de minimisation revient alors à déterminer $\underline{x}^*$ tel que :

$$\underline{x}^* = \underset{\underline{x} \in D}{\operatorname{argmin}}\ f(\underline{x})$$

($\underline{x}^*$ est la valeur de l'argument $\underline{x} \in D$ conduisant à minimiser $f(\underline{x})$)

Exemples

- parmi les triangles isocèles de périmètre p fixé, lequel à la plus grande aire ?
- une ferme est située au centre d'un champ de blé circulaire (rayon R). D'un point de vue économique, le gain surfacique du champ est constant et vaut a tandis que le coût surfacique pour ramener cette production à la ferme évolue linéairement en $b \cdot r$ (plus la récolte est loin de la ferme et plus il est onéreux de l'y ramener). Quelles sont les dimensions de a et b ? Quel est le rayon optimal R^* du champ maximisant le bénéfice ?

Condition nécessaire d'optimalité en 1D

La satisfaction de la CNO pour une fonction $f : \mathbb{R} \to \mathbb{R}$ s'écrit simplement :

$$\frac{df}{dx}(x^*) = 0$$

Si f est convexe au voisinage de x^* alors x^* est un minimum local. Si l'information sur la convexité n'est pas disponible, pour garantir que x^* est un minimum local il suffit de montrer que :

- $\frac{d^2 f}{dx^2}(x^*) > 0$ (pente croissante en x^* - valeurs propres de la matrice hessienne positives pour $f : \mathbb{R}^n \to \mathbb{R}$)
- ou que $\exists\ \varepsilon_0 > 0,\ \forall \varepsilon \leq \varepsilon_0,\ f(x^* \pm \varepsilon) > f(x^*)$ (positivité de la matrice hessienne au voisinage de x^* pour $f : \mathbb{R}^n \to \mathbb{R}$).

Exemple 1

Un ressort sans masse de raideur k et de longueur à vide L_0 est accroché au plafond. Un poids de masse m est ensuite accroché à l'extrémité libre du ressort. Déterminer la position d'équilibre du système soumis au champ de pesanteur terrestre d'accélération g (pour simplifier les calculs, on pourra considérer le vecteur de base $\vec{e}_x$ selon la verticale et du haut vers le bas).

Exemple 1

Solution par application du principe fondamental de la statique ou du principe de minimisation de l'énergie potentielle totale (énergie potentielle de pesanteur de la masse + énergie de déformation élastique du ressort) :

$$x^* = L_0 + \frac{m \cdot g}{k}$$

Remarque : une procédure d'analyse dimensionnelle permettrait également de dériver ce résultat.

Exemple 2

Deux ressorts sans masse de raideurs k_1, k_2 et de longueur à vide L_{01}, L_{02} sont reliés à l'une de leurs extrémités respectives. Les deux extrémités restantes sont ensuite accrochées de sorte que la distance entre celles-ci soit L. On fait l'hypothèse que les deux ressorts restent parfaitement alignés. Déterminer la position d'équilibre du système.

Exemple 2

Solution par application du principe fondamental de la statique ou du principe de minimisation de l'énergie potentielle totale (énergies de déformation élastique des deux ressorts) :

$$x^* = \frac{k_1 \cdot L_{01} + k_2 \cdot (L - L_{02})}{k_1 + k_2}$$

Exemple 3

Une tige sans masse indéformable de longueur L est liée à un bâti au moyen d'un ressort de torsion de raideur C et d'angle à vide θ_0. A sa seconde extrémité est accroché un poids de masse m. Ce système est soumis à l'accélération de la pesanteur terrestre d'intensité g. En supposant que le mouvement a lieu dans le plan $(\vec{e_x}, \vec{e_y})$ (par exemple avec $\vec{e_x}$ vertical de haut en bas), exprimer la position d'équilibre du système.

Exemple 3

Solution par application du principe fondamental de la statique (somme des moments nuls) ou du principe de minimisation de l'énergie potentielle totale (énergie potentielle de pesanteur de la masse + énergie de déformation élastique du ressort) :

$$C \cdot (\theta^* - \theta_0) + m \cdot g \cdot L \cdot \sin(\theta^*) = 0$$

Il s'agit désormais de pouvoir résoudre une équation non-linéaire de la forme $f(\theta) = 0$ (où f est la dérivée de la fonction-objectif) ! ! !

1. Optimisation continue
   - Principes élémentaires de l'optimisation continue
   - Applications à des fonctions-objectif de $\mathbb{R} \to \mathbb{R}$
   - Méthodes de résolution numérique

2. Optimisation combinatoire

Méthode de la dichotomie : principe

La méthode de la dichotomie a pour objectif de déterminer numériquement la solution d'un problème de la forme $f(x) = 0,\ f : \mathbb{R} \to \mathbb{R}$.

Il suffit pour cela de connaître un intervale initial $[a, b]$ où la fonction f change une unique fois de signe (choix par exemple guidé par l'espace de conception du problème d'optimisation sous-jacent).

La dichotomie va alors consister à remplacer sur chaque itération l'une des extrémités de l'intervalle par le milieu de celui-ci. Le choix de l'extrémité à remplacer doit conduire à un nouvel intervalle deux fois plus court où f change de signe.

Méthode de la dichotomie : algorithme

```
input : a, b, epsilon, fonction f(x)
fa=f(a), fb=f(b)
while b-a>epsilon
    c=(a+b)/2.
    fc=f(c)
    if fa*fc<=0 #changement de signe sur [a;c]
        b=c
        fb=fc
    else #changement de signe sur [c;b]
        a=c
        fa=fc
    endif
endwhile
```

Méthode de la dichotomie $f(\theta) = C \cdot (\theta - \theta_0) + m \cdot g \cdot L \cdot \sin(\theta)$

paramètres (unités SI) :

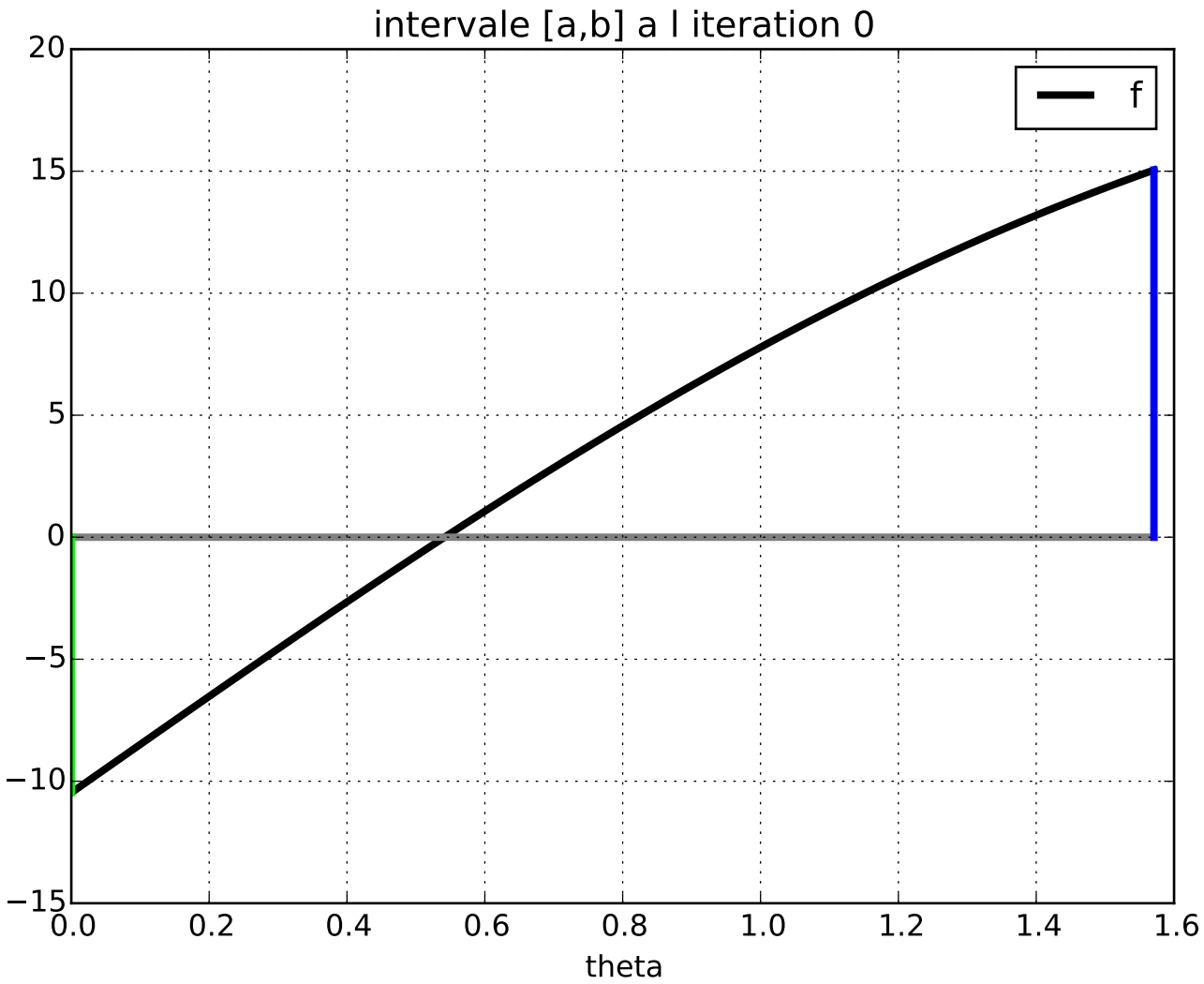
$m = 1$

$g = 9.81$

$L = 1$

$C = 10$

$\theta_0 = \frac{\pi}{3}$

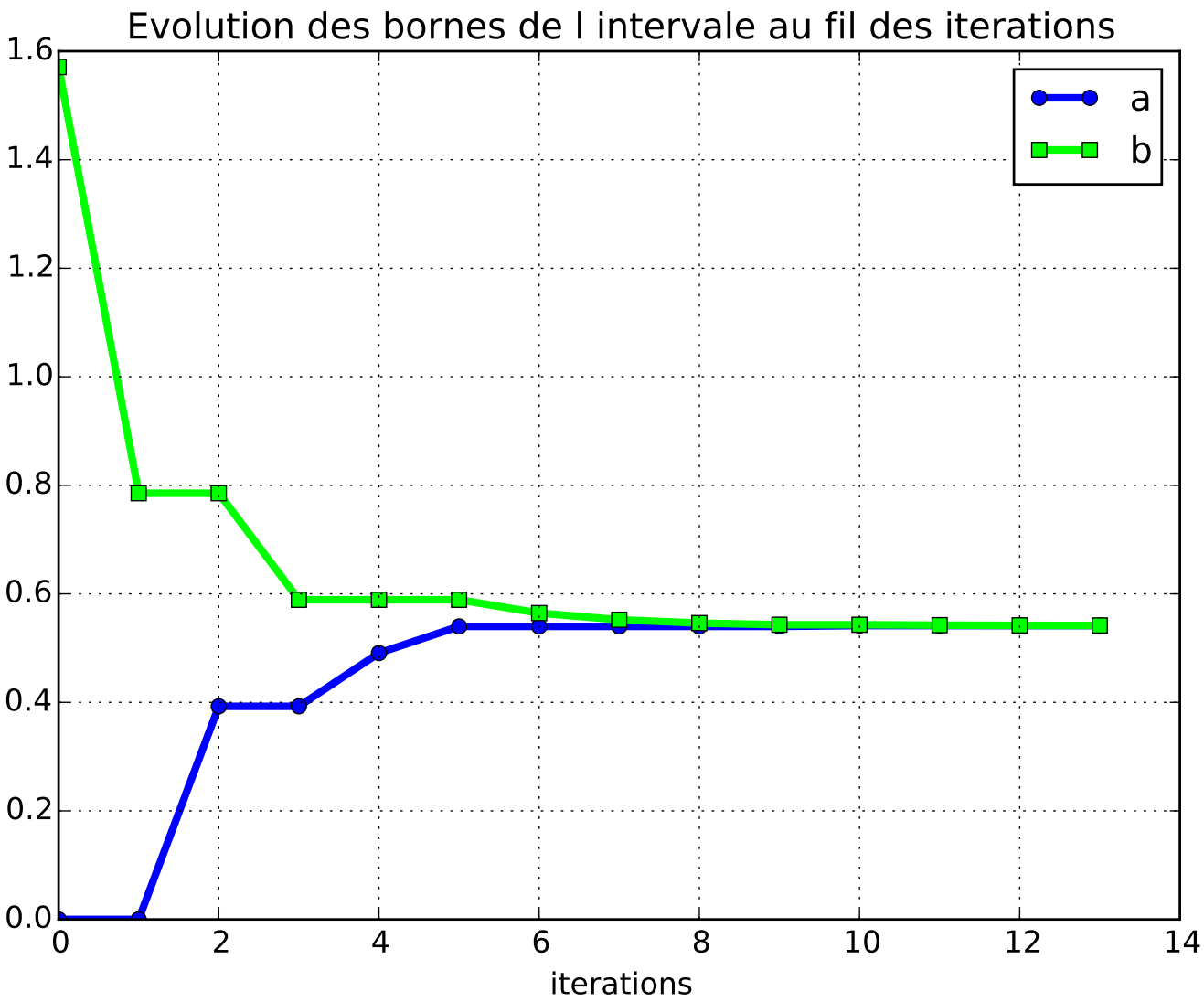


(animation visible avec Acrobat Reader seulement)

Méthode de la dichotomie $f(\theta) = C \cdot (\theta - \theta_0) + m \cdot g \cdot L \cdot \sin(\theta)$

paramètres (unités SI) :

$$m = 1$$
$$g = 9.81$$
$$L = 1$$
$$C = 10$$
$$\theta_0 = \frac{\pi}{3}$$



Méthode de la dichotomie $f(\theta) = C \cdot (\theta - \theta_0) + m \cdot g \cdot L \cdot \sin(\theta)$

paramètres (unités SI) :

$m = 1$

$g = 9.81$

$L = 1$

$C = 10$

$\theta_0 = \frac{\pi}{3}$

iter	a^i	b^i
0	0.0000000000000000e+00	1.5707963267948966e+00
1	0.0000000000000000e+00	7.8539816339744827e-01
2	3.9269908169872413e-01	7.8539816339744827e-01
3	3.9269908169872413e-01	5.8904862254808621e-01
4	4.9087385212340517e-01	5.8904862254808621e-01
5	5.3996123733574563e-01	5.8904862254808621e-01
6	5.3996123733574563e-01	5.6450492994191592e-01
7	5.3996123733574563e-01	5.5223308363883072e-01
8	5.3996123733574563e-01	5.4609716048728818e-01
9	5.3996123733574563e-01	5.4302919891151691e-01
10	5.4149521812363122e-01	5.4302919891151691e-01
11	5.4149521812363122e-01	5.4226220851757412e-01
12	5.4149521812363122e-01	5.4187871332060267e-01
13	5.4149521812363122e-01	5.4168696572211694e-01
14	5.4149521812363122e-01	5.4159109192287413e-01

Méthode de Newton-Raphson : principe

La méthode de Newton-Raphson a pour objectif de déterminer numériquement la solution d'un problème de la forme $f(x) = 0,\ f : \mathbb{R} \to \mathbb{R}$.

On se donne pour cela une estimation initiale raisonable x^0 de la solution x^* (guidé par des considérations physiques par exemple). La fonction f doit être dérivable dans l'intervalle de recherche.

La méthode de Newton-Raphson va alors consister à déduire x^{k+1} de x^k en approximant localement f par l'équation de sa courbe tangente en x^k et à déterminer en quelle valeur celle-ci s'annule.

Méthode de Newton-Raphson : algorithme

Equation de la courbe tangente en x^k :

$$y(x) = f'(x^k) \cdot (x - x^k) + f(x^k) \tag{1}$$

Cette droite coupe l'axe des x en x^{k+1} d'où :

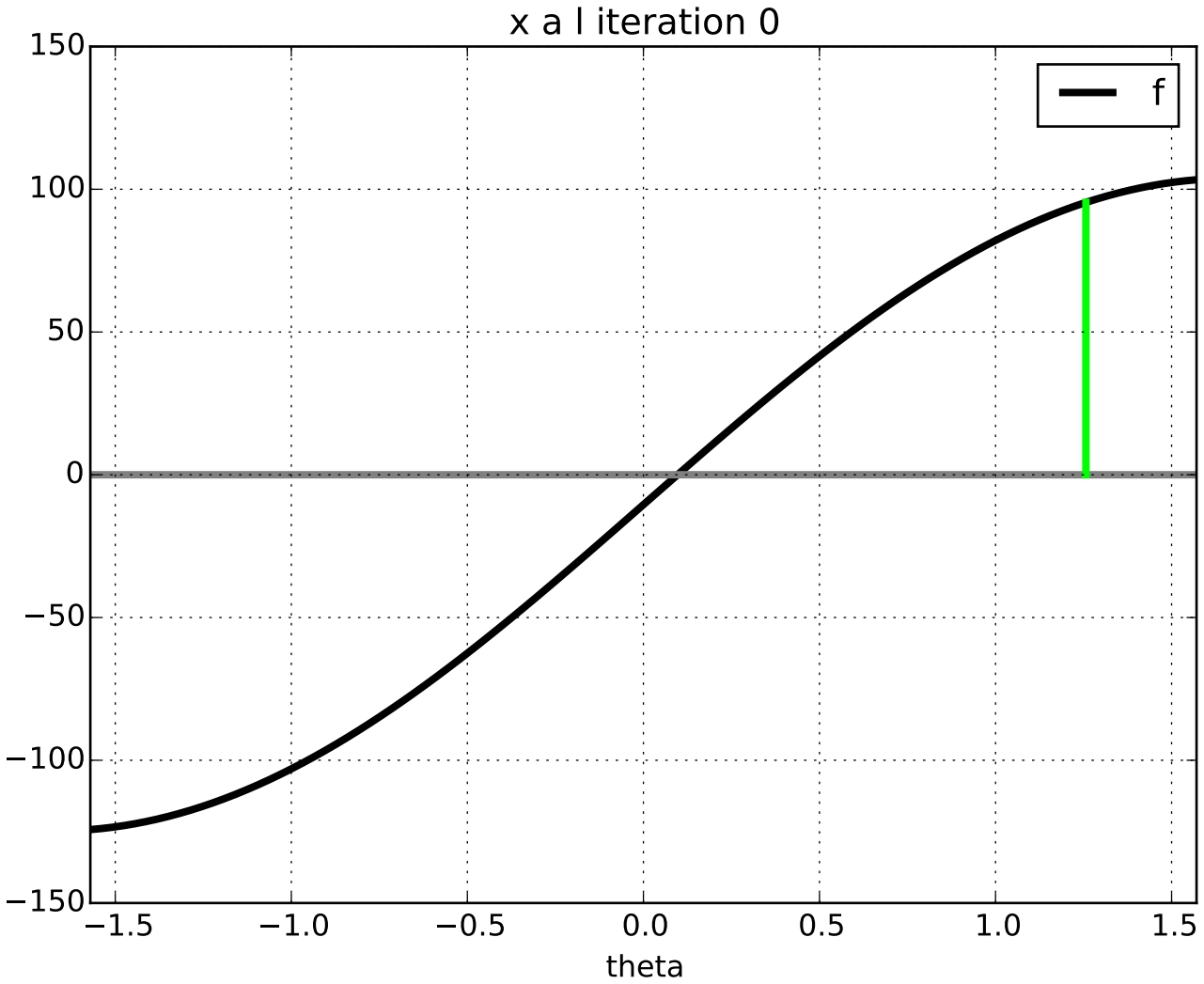
$$\begin{aligned} 0 &= f'(x^k) \cdot (x^{k+1} - x^k) + f(x^k) \\ x^{k+1} &= x^k - \frac{f(x^k)}{f'(x^k)} \end{aligned} \tag{2}$$

```
input : x, epsilon, fonction f(x), fonction f'(x)
while abs(f(x))>epsilon
    x=x-f(x)/f'(x)
endwhile
```

Méthode de Newton-Raphson $f(\theta) = C \cdot (\theta - \theta_0) + m \cdot g \cdot L \cdot \sin(\theta)$

paramètres (unités SI) :

$$m = 10$$
$$g = 9.81$$
$$L = 1$$
$$C = 10$$
$$\theta_0 = \frac{\pi}{3}$$

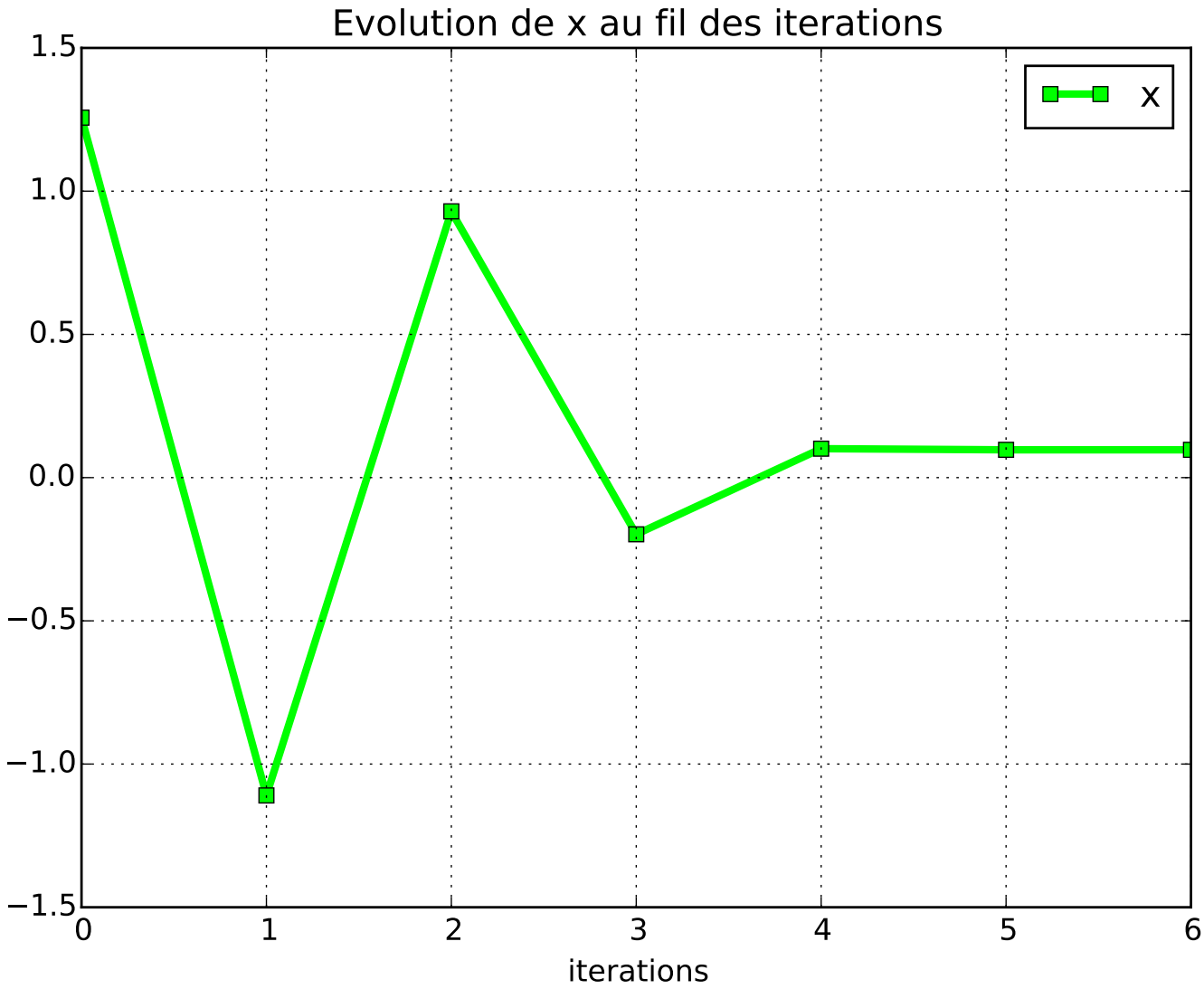


(animation visible avec Acrobat Reader seulement)

Méthode de Newton-Raphson $f(\theta) = C \cdot (\theta - \theta_0) + m \cdot g \cdot L \cdot \sin(\theta)$

paramètres (unités SI) :

$$m = 10$$
$$g = 9.81$$
$$L = 1$$
$$C = 10$$
$$\theta_0 = \frac{\pi}{3}$$



Méthode de Newton-Raphson $f(\theta) = C \cdot (\theta - \theta_0) + m \cdot g \cdot L \cdot \sin(\theta)$

paramètres (unités SI) :

$m = 10$

$g = 9.81$

$L = 1$

$C = 10$

$\theta_0 = \frac{\pi}{3}$

iter	x^i	$f(x^i)$
0	1.25663706143591724 6e+00	9.539303935094775966e+01
1	-1.10958056407686767 0e+00	-1.094175181512762691e+02
2	9.29578936855333104 5e-01	7.743802878037534754e+01
3	-1.97930225351429678 3e-01	-3.174169999901330641e+01
4	1.00999005765956884 1e-01	4.291806769509456387e-01
5	9.70103405813251412 6e-02	-7.764916048458303521e-05
6	9.70110619679912383 8e-02	-2.470912363605748396e-12
7	9.70110619680141922 4e-02	-1.776356839400250465e-15

1. Optimisation continue

2. Optimisation combinatoire
   - Exemple : "problème du voyageur de commerce"
   - Complexité des algorithmes
   - Heuristiques

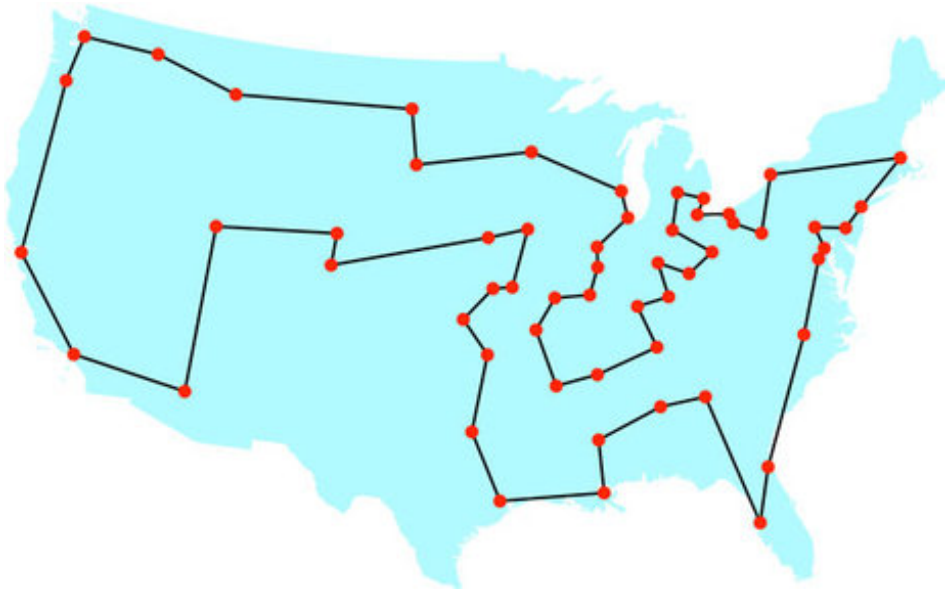
1. Optimisation continue

2. Optimisation combinatoire
   - Exemple : "problème du voyageur de commerce"
   - Complexité des algorithmes
   - Heuristiques

Problème du voyageur de commerce : principe

De manière générale, l'optimisation combinatoire consiste à trouver une solution optimale à un problème donné parmi un ensemble **fini** de solutions possibles.

Le problème du voyageur de commerce (Travelling Salesman Problem - TSP), qui consiste à déterminer le plus court trajet reliant n villes données, constitue un exemple typique de problème d'optimisation combinatoire.



Problème du voyageur de commerce : complexité

On s'intéresse désormais à un cas particulier du TSP où les points de départ et d'arrivée sont fixés et identiques (ex. : tournée d'un facteur).

La tournée comprend n sites distincts à visiter chacun une unique fois. L'ensemble de toutes les tournées est ainsi un ensemble fini qui comprend $n \cdot (n-1) \cdot (n-2) \cdot \ \ldots \ \cdot 1 = n!$ éléments. Parmi ces $n!$ possibilités, il en existe une optimale.

Un parcours exhaustif de l'ensemble des possibilités doit permettre de déterminer la solution optimale. Du fait de la complexité en $O(n!)$ de l'algorithme de parcours, il devient rapidement très onéreux, voire impossible en pratique, de calculer toutes les possibilités.

7 !	10 !	13 !
5.04e+3	3.6288e+6	6.2270e+009

Problème du voyageur de commerce : algorithme naïf

```
input : villes, fonction d(ville_i,ville_j)
dTotMin=10^12 #nombre plus grand que le plus grand trajet possible
for v1 dans villes
    dTot=0
    for v2 dans villes\{v1}
        dTot=dTot+d(v1,v2)
        for v3 dans villes\{v1,v2}
            dTot=dTot+d(v2,v3)
            ...
            for vn dans villes\{v1,v2,...,v(n-1)}
                dTot=dTot+d(v(n-1),vn)
                if dTot<dTotMin
                    dTotMin=dTot
                    trajetOptimum=(v1,v2,...,vn)
                endif
            endfor
            ...
         endfor
     endfor
endfor
```

Autres problèmes typiques en optimisation combinatoire

- "knapsack problem" : on considère un ensemble de n objets ayant comme caractéristiques un couple (v_i, m_i) (par exemple valeur et masse). Une personne souhaite emporter avec elle un sous-ensemble des objets dont la valeur cumulée soit maximale au regard d'une contrainte de masse maximale à ne pas dépasser.
- exemple de "minimum spanning tree" : on considère n villes distinctes dont on connait les distances les unes par rapport aux autres (graphe pondéré non-orienté). On souhaite alors construire le réseau routier le plus court permettant de relier les n villes (chaque ville pouvant accueillir plusieurs tronçons routiers).

Complexités algorithmiques

- complexité de Kolmogorov : il s'agit d'une mesure de la taille (au sens d'information) du plus petit programme permettant de réaliser un traitement.
- complexité en temps d'exécution : mesure du nombre d'opérations à effectuer pour que le programme réalise un traitement.

Complexités algorithmiques : complexité de Kolmogorov

Afficher : 01010101010101010101010101010101

Afficher : 10110100010111001001111111010000

Complexités algorithmiques : complexité de Kolmogorov

Afficher : 01010101010101010101010101010101

```
for i = 1 to 16
    afficher '0'
    afficher '1'
end
```

Afficher : 10110100010111001001111111010000

```
afficher '1'
afficher '0'
afficher '1'
afficher '1'
...
afficher '0'
```

Complexités algorithmiques : complexité en temps d'exécution

Exemple du produit matrice par vecteur :
$A \in M_{n\times n}(\mathbb{R}),\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^n$
$y = A \cdot x$

```
for i = 1 to n
    somme=0
    for j = 1 to n
        somme=somme+A[i,j]*x[j]
    endfor
    y[i]=somme
endfor
```

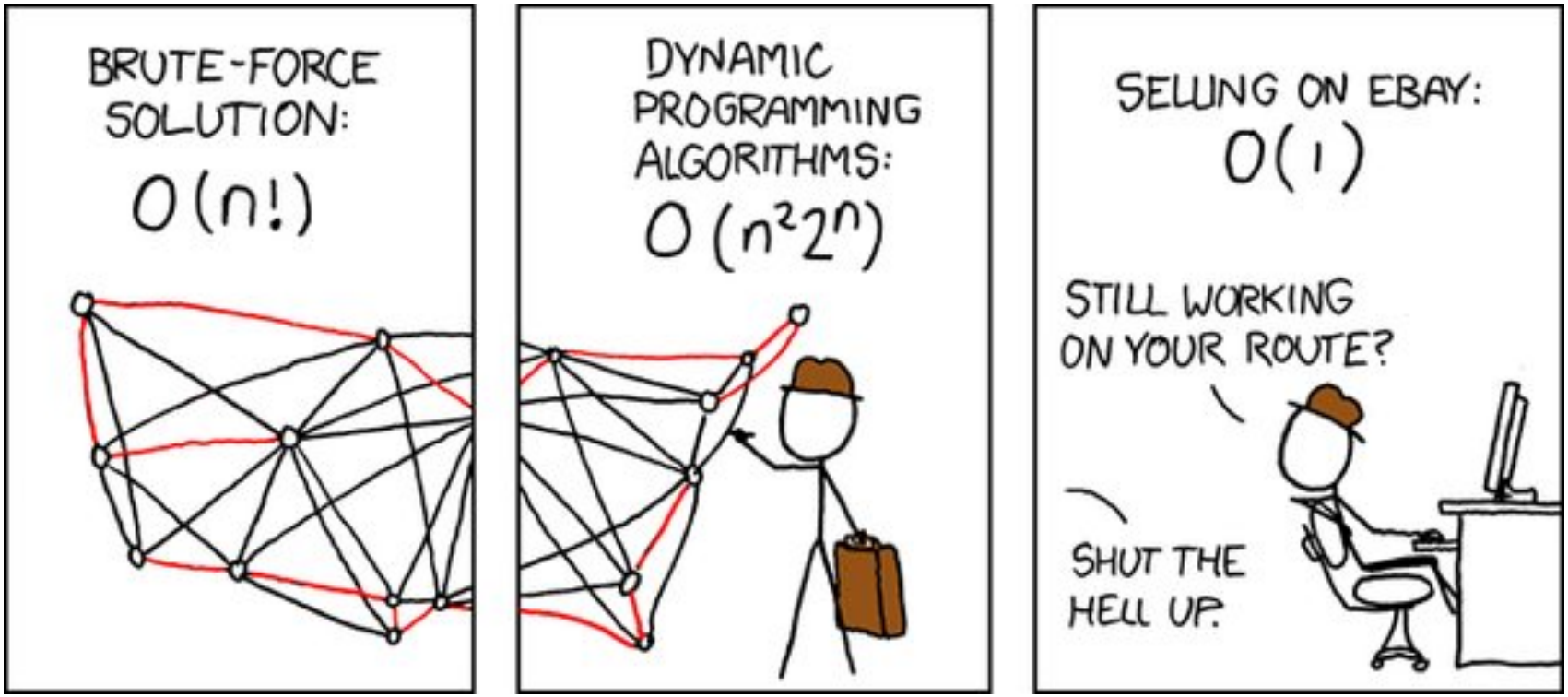
$\rightarrow n^2$ additions et multiplications

Complexités algorithmiques : complexité en temps d'exécution

temps :	type :	exemple :
$O(1)$	constante	afficher une lettre
$O(log(n))$	logarithmique	trouver la position d'un élément dans une liste ordonnée par dichotomie
$O(n)$	linéaire	trouver la position d'un élément dans une liste ordonnée par force brute
$O(n \cdot log(n))$	linéarithmique	classement par *heapsort*
$O(n^2)$	quadratique	produit matrice vecteur
$O(n^3)$	cubique	produit matrice matrice
$O(n!)$	factorielle	problème du voyageur de commerce

Exemples : 1

Problème du voyageur de commerce (source : xkcd.com)



Exemples : 2

Exercices : en se plaçant dans les cas le plus défavorables, déterminer les complexités pour :

- rechercher la valeur maximale dans une liste de nombres,
- trier par ordre croissant une liste de nombres.

1. Optimisation continue

2. Optimisation combinatoire
   - Exemple : "problème du voyageur de commerce"
   - Complexité des algorithmes
   - Heuristiques

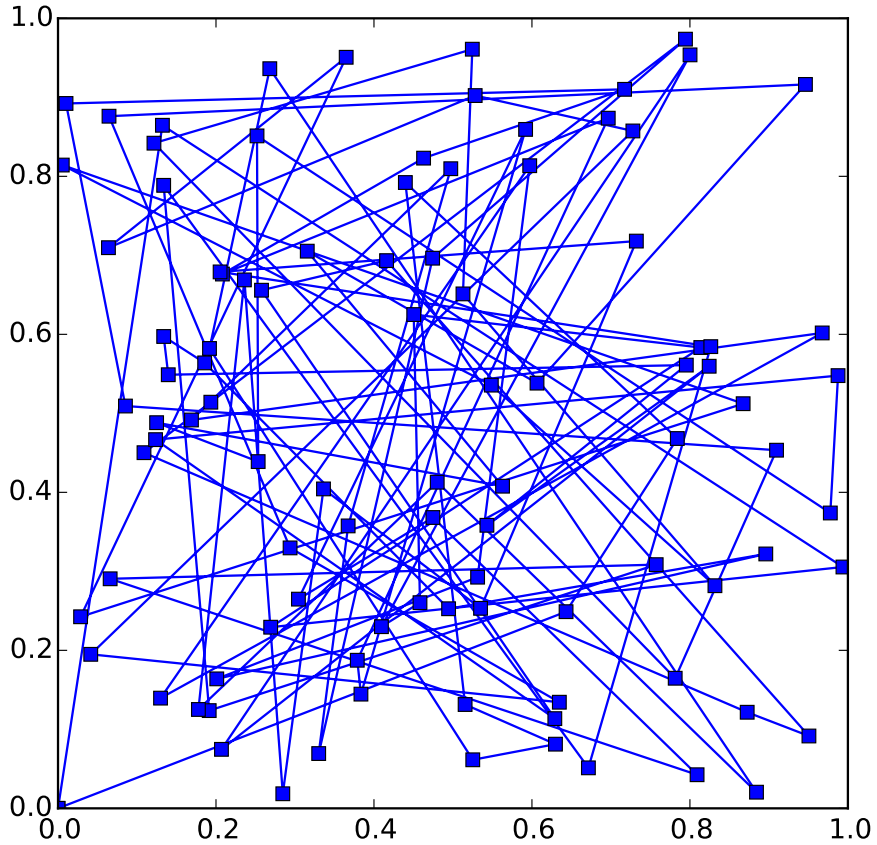
Notion d'heuristique

Etant donné un problème d'optimisation (continue ou combinatoire), une heuristique est dans ce cas une stratégie qui permet d'approcher la solution optimale avec un compromis qualité/coût plus favorable que l'algorithme de base s'il existe.

Pour le TSP, l'algorithme de recherche exhaustive est certain de trouver la meilleure solution mais son coût $O(n!)$ est prohibitif. Une(des) heuristique(s) doivent alors être employée(s) pour traiter des problèmes autrement non-calculables en temps raisonnable ($n = 10000$ villes).

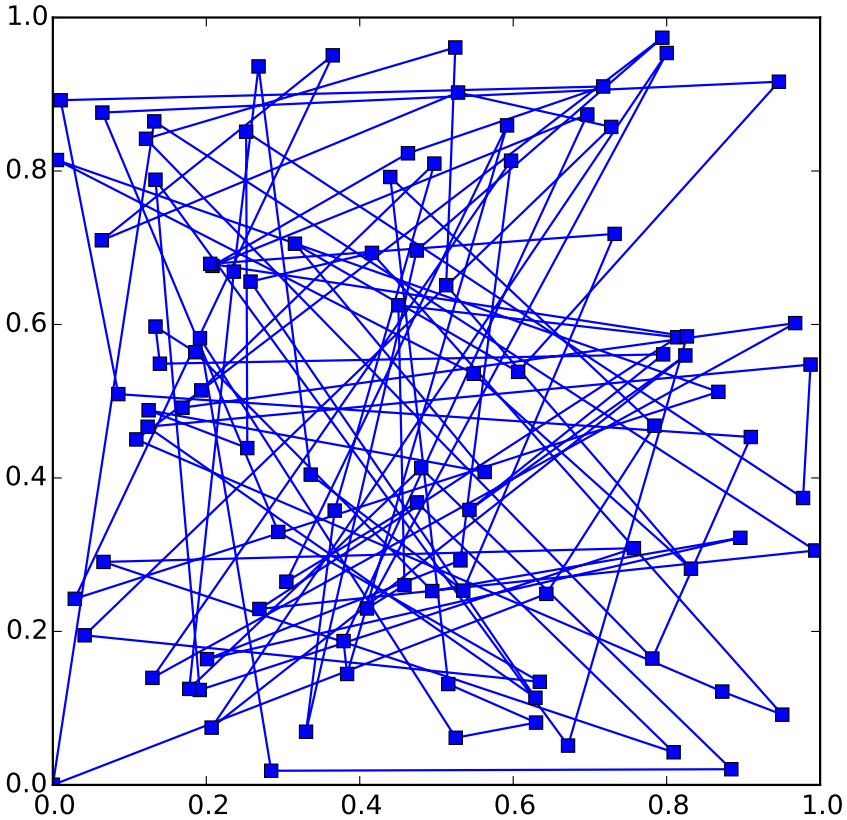
Heuristiques simples pour le TSP : 1

Processus itéré p fois :
Un couple aléatoire de villes est permuté, si cette permutation conduit à "réduire les distances", la permutation est conservée. ($n = 100$, $p = 10000$ dans l'exemple ci-dessous)



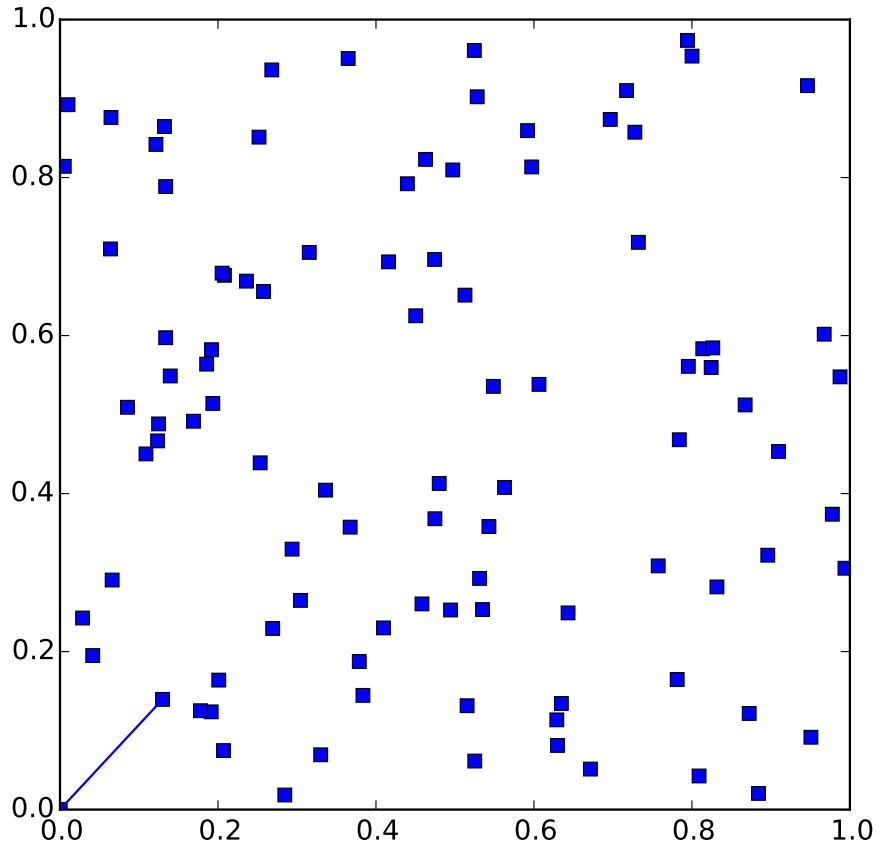
Heuristiques simples pour le TSP : 1

Processus itéré p fois :
Un couple aléatoire de villes est permuté, si cette permutation conduit à "réduire les distances", la permutation est conservée. ($n = 100$, $p = 10000$ dans l'exemple ci-dessous)



Heuristiques simples pour le TSP : 2

A partir d'une ville donnée, on choisit la suivante comme étant celle des villes restantes qui lui est la plus proche au sens de la norme euclidienne.



Heuristiques simples pour le TSP

algo	distance
aucun	53.17
h1	21.38
h2	9.88

Quel algorithme possède le meilleur rapport "qualité/prix" (coût lié au calcul des distances inter-villes) ?