

# Méthodes d'optimisation

**BUT Info 2e année**

Florent Foucaud  
Dipayan Chakraborty, Malika More, Adrien Wohrer



**IUT CLERMONT AUVERGNE**

Aurillac - Clermont-Ferrand - Le Puy-en-Velay  
Montluçon - Moulins - Vichy

2022-2023

# Heuristiques et méta-heuristiques

## Définition (Algorithme heuristique)

Algorithme qui calcule une solution d'un problème **sans garantie de qualité**.

Exemple : algorithme glouton pour la couverture par sommets (voir cours précédent)

# Heuristiques et méta-heuristiques

## Définition (Algorithme heuristique)

Algorithme qui calcule une solution d'un problème **sans garantie de qualité**.

Exemple : algorithme glouton pour la couverture par sommets (voir cours précédent)

## Définition (Méta-heuristique)

Algorithme heuristique **générique**, applicable à un grand nombre de problèmes.

# Heuristiques et méta-heuristiques

## Définition (Algorithme heuristique)

Algorithme qui calcule une solution d'un problème **sans garantie de qualité**.

Exemple : algorithme glouton pour la couverture par sommets (voir cours précédent)

## Définition (Méta-heuristique)

Algorithme heuristique **générique**, applicable à un grand nombre de problèmes.

Deux grandes familles de méta-heuristiques sont utilisées en Optimisation :

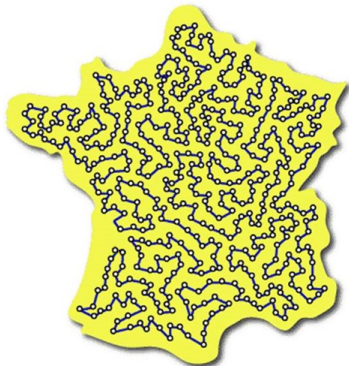
- Méta-heuristiques de **recherche locale** : descente de gradient, recuit simulé, méthode tabou...
- Méta-heuristiques **à population** : algorithmes génétiques, colonies de fourmis, essais de particules...

## Exemple de problème : le voyageur de commerce

**Entrée** : un ensemble de  $n$  villes et les distances entre toute paire de villes

**Sortie** : une tournée (permutation des  $n$  villes) la plus courte possible

(en anglais : Travelling Salesman Problem, TSP)



Un problème très célèbre depuis le début du 20e siècle, et très difficile à résoudre !

# Méta-heuristiques de recherche locale

## Méthode de la descente de gradient :

On parcourt l'espace de solutions par des transformations successives jusqu'à trouver un optimum local.

# Méta-heuristiques de recherche locale

## Méthode de la descente de gradient :

On parcourt l'espace de solutions par des transformations successives jusqu'à trouver un optimum local.



# Méta-heuristiques de recherche locale

## Méthode de la descente de gradient :

On parcourt l'espace de solutions par des transformations successives jusqu'à trouver un optimum local.



On définit une opération de transformation d'une solution en une autre, proche. Pour une solution  $S$ , soit  $N(S)$  l'ensemble des solutions "voisines".

## Algorithme de recherche locale par descente de gradient :

1. On génère une première solution  $S_0$

$$S \leftarrow S_0$$

#solution courante

2. Tant que  $N(S)$  contient une solution meilleure que  $S$  :

▶ choisir une nouvelle solution  $S'$  dans  $N(S)$

▶  $S \leftarrow S'$

3. Retourner  $S$



# Variations

## Algorithme de recherche locale par descente de gradient :

1. On génère une première solution  $S_0$

$$S \leftarrow S_0$$

#solution courante

2. Tant que  $N(S)$  contient une solution meilleure que  $S$  :
  - ▶ choisir une nouvelle solution  $S'$  dans  $N(S)$
  - ▶  $S \leftarrow S'$

3. Retourner  $S$

# Variations

## Algorithme de recherche locale par descente de gradient :

1. On génère une première solution  $S_0$   
 $S \leftarrow S_0$  #solution courante
2. Tant que  $N(S)$  contient une solution meilleure que  $S$  :
  - ▶ choisir une nouvelle solution  $S'$  dans  $N(S)$
  - ▶  $S \leftarrow S'$
3. Retourner  $S$

## Variantes :

# Variations

## Algorithme de recherche locale par descente de gradient :

1. On génère une première solution  $S_0$   
 $S \leftarrow S_0$  #solution courante
2. Tant que  $N(S)$  contient une solution meilleure que  $S$  :
  - ▶ choisir une nouvelle solution  $S'$  dans  $N(S)$
  - ▶  $S \leftarrow S'$
3. Retourner  $S$

## Variantes :

- Solution initiale  $S_0$  : aléatoire, algo glouton, solution triviale...

# Variations

## Algorithme de recherche locale par descente de gradient :

1. On génère une première solution  $S_0$   
 $S \leftarrow S_0$  #solution courante
2. Tant que  $N(S)$  contient une solution meilleure que  $S$  :
  - ▶ choisir une nouvelle solution  $S'$  dans  $N(S)$
  - ▶  $S \leftarrow S'$
3. Retourner  $S$

## Variantes :

- Solution initiale  $S_0$  : aléatoire, algo glouton, solution triviale...
- Sélection de la nouvelle solution : meilleur voisin, premier voisin améliorant, voisin aléatoire...

# Variations

## Algorithme de recherche locale par descente de gradient :

1. On génère une première solution  $S_0$

$$S \leftarrow S_0$$

#solution courante

2. Tant que  $N(S)$  contient une solution meilleure que  $S$  :
  - ▶ choisir une nouvelle solution  $S'$  dans  $N(S)$
  - ▶  $S \leftarrow S'$

3. Retourner  $S$

## Variantes :

- Solution initiale  $S_0$  : aléatoire, algo glouton, solution triviale...
- Sélection de la nouvelle solution : meilleur voisin, premier voisin améliorant, voisin aléatoire...

Passer à une solution non-améliorante pour quitter un mauvais optimum local (diversification).



# Variations

## Algorithme de recherche locale par descente de gradient :

1. On génère une première solution  $S_0$   
 $S \leftarrow S_0$  #solution courante
2. Tant que  $N(S)$  contient une solution meilleure que  $S$  :
  - ▶ choisir une nouvelle solution  $S'$  dans  $N(S)$
  - ▶  $S \leftarrow S'$
3. Retourner  $S$

## Variantes :

- Solution initiale  $S_0$  : aléatoire, algo glouton, solution triviale...
- Sélection de la nouvelle solution : meilleur voisin, premier voisin améliorant, voisin aléatoire...

Passer à une solution non-améliorante pour quitter un mauvais optimum local (diversification).

→ Variante du **recuit simulé** (1983) :

On définit une **température** qui décroît lentement au cours de l'algorithme. La **probabilité de diversification** dépend de la température.



Le recuit, un technique métallurgique



Scott Kirkpatrick



C. Daniel Gelatt



Mario P. Vecchi

# Méta-heuristiques à populations de solutions

**Idée générale** : on fait évoluer des ensembles de solutions (populations)

- **Algorithmes génétiques** (J. H. Holland, 1975)
  - ▶ Population initiale : ensemble de solutions
  - ▶ On “croise” des paires de solutions pour obtenir la population suivante
  - ▶ Des “mutations” apparaissent aléatoirement



# Méta-heuristiques à populations de solutions

**Idée générale** : on fait évoluer des ensembles de solutions (populations)

- **Algorithmes génétiques** (J. H. Holland, 1975)

- ▶ Population initiale : ensemble de solutions
- ▶ On “croise” des paires de solutions pour obtenir la population suivante
- ▶ Des “mutations” apparaissent aléatoirement



- **Algorithmes de colonies de fourmis** (M. Dorigo, 1992)

- ▶ Des “fourmis” se déplacent plus ou moins aléatoirement dans l’instance et laissent des phéromones
- ▶ Les fourmis suivantes vont là où il y a le plus de phéromones
- ▶ Les phéromones disparaissent après un certain temps



John H. Holland  
(1929-2015)



Marco Dorigo  
(1961-)