Algorithmes de recherche locale

Recherche Opérationnelle et Optimisation
Master 1

Sébastien Verel
verel@lisic.univ-littoral.fr
http://www-lisic.univ-littoral.fr/~verel

Université du Littoral Côte d'Opale
Laboratoire LISIC
Equipe CAMOME

Plan

1 Introduction

2 Recherche locales basées sur le gradient

Optimization

Inputs

- Search space : Set of all feasible solutions,

$$\mathcal{X}$$

- Objective function : Quality criterium

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

Goal

Find the best solution according to the criterium

$$x^\star = \operatorname{argmax} f$$

But, sometime, the set of all best solutions, good approximation of the best solution, good 'robust' solution...

Contexte

Black box Scenario

We have only $\{(x_0, f(x_0)), (x_1, f(x_1)), ...\}$ given by an "oracle"
No information is either not available or needed on the definition of objective function

- Objective function given by a computation, or a simulation
- Objective function can be irregular, non differentiable, non continous, etc.

Typologie des problèmes

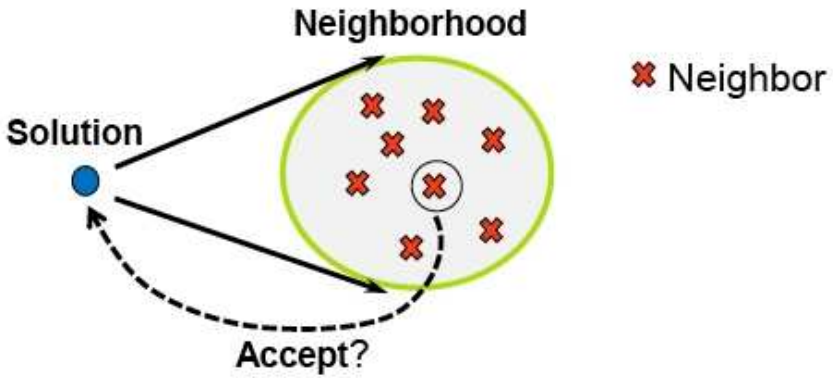
- Espace de recherche très large dont les variables sont discrètes (cas NP-complet) : optimisation combinatoire
- Espace de recherche dont les variables sont continues : optimisation numérique

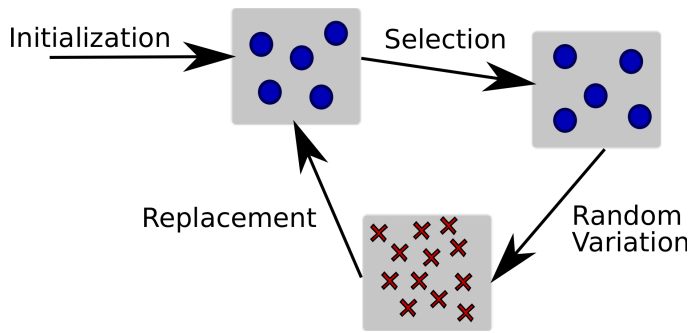
Search algorithms

Principle

Enumeration of the search space

- A lot of ways to enumerate the search space
- Using random sampling : Monte Carlo technics
- Local search technics :



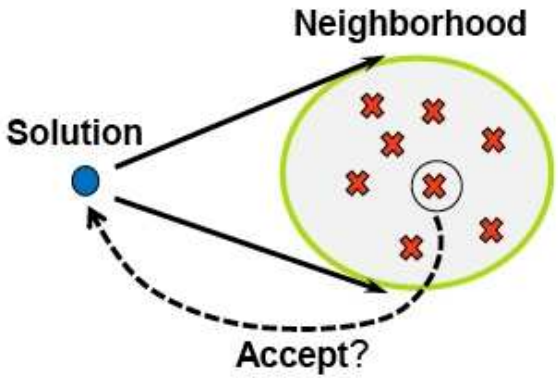


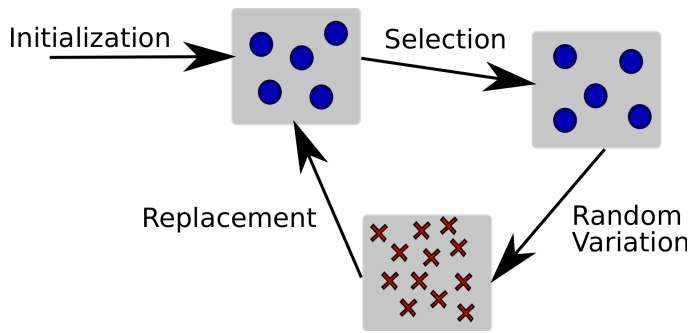
Search algorithms

Principle

Enumeration of the search space

- A lot of ways to enumerate the search space
- Using random sampling : Monte Carlo technics
- Local search technics :





- If objective function f has no propertie : random search
- If not...

Retour à MAX-SAT

Comment résoudre ce genre de problèmes ?

$$(x_1 \vee x_2) \wedge (\bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_3})$$

$$(x_4 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (\bar{x_2} \vee x_1 \vee x_5) \wedge (\bar{x_2} \vee x_5 \vee x_4) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4})$$

Retour à MAX-SAT

mathématiquement ou ...

- Exhaustivement $n = 20$, $n = 100$, ...
- Aléatoirement
- Construction de solution (plus tard)
- Méthodes exactes (plus tard)

ou ...

Heuristiques

Heuristique

Algorithme de résolution dont la conception repose sur l"'expérience" du concepteur.

Heuristiques

Heuristique

Algorithme de résolution dont la conception repose sur l"'expérience" du concepteur.

Souvent :

- Pas de garantie d'obtenir une solution optimale

On désire toutefois :

- Le plus souvent possible une solution proche de l'optimalité
- Le moins souvent possible un mauvaise solution (différent !)
- Une complexité "raisonnable"
- De la simplicité d'implémentation (code light en version de base...)

Metaheuristiques

Peu probable qu'un algorithme puisse résoudre tout problème

Métaheuristique

Ensemble d'heuristiques :

- regroupe des heuristiques dépendant de paramètres
- décrit une méthode de conception d'heuristique

de
un aveu d'impuissance
à
des techniques performantes d'optimisation difficile

Metaheuristisques de recherche locale

Algorithmes à **population de solutions**

- Algorithmes Evolutionnaires (EA) : Holland 1975 et même avant
- Algorithmes d'essaims particulaires (PSO) : R. Ebenhart et J. Kennedy 1995.
- Algorithmes de fourmis (ACO) : Bonabeau 1999

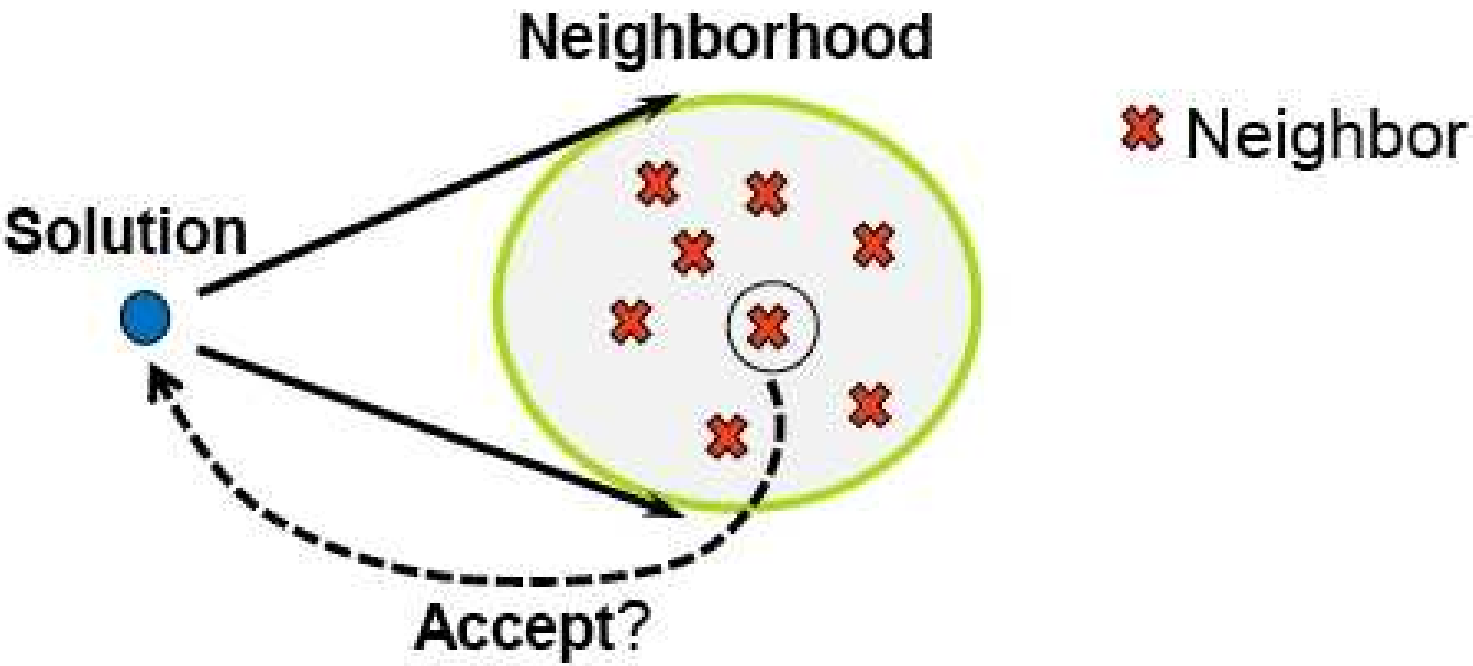
Metaheuristisques de recherche locale

Algorithmes à **solution unique**

- (Recherches aléatoire),
- Algorithmes de descente : Hill-Climber (HC), première-descente
- Recuit Simulé (SA) : Kirkpatrick *et al* 1983,
- Recherche Tabou (TS) : Glover 1986 - 89 -90,
- Iterated Local Search

Stochatic algorithms with unique solution (Local Search)

- $\mathcal{S}$ set of solutions (search space)
- $f : \mathcal{S} \rightarrow \mathbb{R}$ objective function
- $\mathcal{V}(s)$ set of neighbor's solutions of s



Recherche Locale (LS)

- $\mathcal{S}$ ensemble des solutions (espace de recherche),
- $f : S \rightarrow \mathbb{R}$ fonction objectif à maximiser (ou coût à minimiser)
- $\mathcal{V}(s)$ ensemble des solutions voisines de s

Algorithme d'une Recherche Locale

Choisir solution initiale $s \in \mathcal{S}$
repeat
  choisir $s' \in \mathcal{V}(s)$
  if accept(s, s') **then**
    $s \leftarrow s'$
  end if
until critère d'arrêt non verifié

Un exemple très simple : OneMax ou Sac à dos sans contrainte

Fonction d'évaluation du One Max

Pour toute chaîne binaire de longueur N, $x \in \{0,1\}^N$

$$f(x) = \sum_{i=1}^{N} x_i$$

Par exemple,

$$\text{pour } x = 01101,\ f(x) = 3$$

- Taille de l'espace de recherche $\sharp\mathcal{S} =$

Un exemple très simple : OneMax ou Sac à dos sans contrainte

Fonction d'évaluation du One Max

Pour toute chaîne binaire de longueur N, $x \in \{0,1\}^N$

$$f(x) = \sum_{i=1}^{N} x_i$$

Par exemple,

$$\text{pour } x = 01101,\ f(x) = 3$$

- Taille de l'espace de recherche $\sharp\mathcal{S} = 2^N$

Un exemple très simple : OneMax ou Sac à dos sans contrainte

Fonction d'évaluation du One Max

Pour toute chaîne binaire de longueur N, $x \in \{0,1\}^N$

$$f(x) = \sum_{i=1}^{N} x_i$$

Par exemple,

$$\text{pour } x = 01101,\ f(x) = 3$$

- Taille de l'espace de recherche $\sharp\mathcal{S} = 2^N$

Exercice

- Coder, dans le langage que vous voulez, la fonction d'évaluation.
- Tester en affichant une solution et la valeur de la fonction

Black-Box

Contexte black-box

Oublier la fonction à maximiser !

Pour chaque solution x, on connait seulement $f(x)$.
Pas la manière dont est calculée f.

Black-Box

Contexte black-box

Oublier la fonction à maximiser !

Pour chaque solution x, on connait seulement $f(x)$.
Pas la manière dont est calculée f.

A la limite, ici, dans le cadre de l'exercice, on peut admettre connaitre la valeur du maximum.

Résolution d'un problème d'optimisation combinatoire

Inputs

- Espace de recherche :

$$\mathcal{X} = \{0,1\}^n$$

- Function objectif :

$$f = \text{oneMax}$$

Goal

Find the best solution according to the criterium

$$x^\star = \operatorname{argmax} f$$

Recherche Locale (LS)

- $\mathcal{S}$ ensemble des solutions (espace de recherche),
- $f : S \rightarrow \mathbb{R}$ fonction objectif à maximiser (ou coût à minimiser)
- $\mathcal{V}(s)$ ensemble des solutions voisines de s

Algorithme d'une Recherche Locale

Choisir solution initiale $s \in \mathcal{S}$
repeat
  choisir $s' \in \mathcal{V}(s)$
  if accept(s, s') **then**
    $s \leftarrow s'$
  end if
until critère d'arrêt non verifié

Recherche aléatoire sur l'espace de recherche

Algorithme de Recherche Aléatoire (sur l'espace de recherche)

Choisir solution initiale $s \in \mathcal{S}$ aléatoirement uniformément sur $\mathcal{S}$.
repeat
  choisir aléatoirement $s' \in \mathcal{S}$
  $s \leftarrow s'$
until critère d'arrêt non verifié

- Voisinage $\mathcal{V} = \mathcal{S}$
- 1 seule itération, ou mieux aucune itération
- accept$(s, s') = \mathit{true}$

Recherche aléatoire sur l'espace de recherche

Algorithme de Recherche Aléatoire (sur l'espace de recherche)

Choisir solution initiale $s \in \mathcal{S}$ aléatoirement uniformément sur $\mathcal{S}$.
repeat
  choisir aléatoirement $s' \in \mathcal{S}$
  $s \leftarrow s'$
until critère d'arrêt non verifié

Exercice

- Coder la recherche aléatoire
- Evaluer les performances de la recherche aléatoire en calculant les statistiques de la qualité des solutions obtenues sur un grand d'exécutions de la recherche.

Recherche Locale Aléatoire (marche aléatoire)

Heuristique d'**exploration** maximale

Recherche locale aléatoire
Marche aléatoire

Choisir solution initiale $s \in \mathcal{S}$
Evaluer s avec f
repeat
  choisir $s' \in \mathcal{V}(s)$ aléatoirement
  Evaluer s' avec f
  $s \leftarrow s'$
until Nbr d'éval. $\leq$ maxNbEval

- Algorithme inutilisable en pratique
- Algorithme de comparaison
- Opérateur local de base de nombreuses métaheuristiques

Voisinage des chaines binaires

Distance de Hamming

Nombre de différence entre 2 chaînes.

Voisinage de $x \in \{0,1\}^N$

Voisinage des chaines binaires

Distance de Hamming

Nombre de différence entre 2 chaînes.

Voisinage de $x \in \{0,1\}^N$

$\mathcal{V}(x)$: ensemble des chaînes binaires à une distance 1 de x.
"On modifie 1 seul bit"

Voisinage des chaines binaires

Distance de Hamming

Nombre de différence entre 2 chaînes.

Voisinage de $x \in \{0,1\}^N$

$\mathcal{V}(x)$: ensemble des chaînes binaires à une distance 1 de x.
"On modifie 1 seul bit"

Pour $x = 01101$, $\mathcal{V}(x) = \{$

Voisinage des chaines binaires

Distance de Hamming

Nombre de différence entre 2 chaînes.

Voisinage de $x \in \{0,1\}^N$

$\mathcal{V}(x)$: ensemble des chaînes binaires à une distance 1 de x.
"On modifie 1 seul bit"

$$\text{Pour } x = 01101,\ \mathcal{V}(x) = \{\ \begin{array}{l} 01100, \\ 01111, \\ 01001, \\ 00101, \\ 11101 \end{array}\ \}$$

Voisinage des chaines binaires

Distance de Hamming

Nombre de différence entre 2 chaînes.

Voisinage de $x \in \{0,1\}^N$

$\mathcal{V}(x)$: ensemble des chaînes binaires à une distance 1 de x.
"On modifie 1 seul bit"

$$\text{Pour } x = 01101,\ \mathcal{V}(x) = \{\ \begin{matrix} 01100, \\ 01111, \\ 01001, \\ 00101, \\ 11101 \end{matrix}\ \}$$

- Taille du voisinage d'une chaîne binaire de longueur :

Voisinage des chaines binaires

Distance de Hamming

Nombre de différence entre 2 chaînes.

Voisinage de $x \in \{0,1\}^N$

$\mathcal{V}(x)$: ensemble des chaînes binaires à une distance 1 de x.
"On modifie 1 seul bit"

$$\text{Pour } x = 01101,\ \mathcal{V}(x) = \{\ \begin{matrix} 01100, \\ 01111, \\ 01001, \\ 00101, \\ 11101 \end{matrix}\ \}$$

- Taille du voisinage d'une chaîne binaire de longueur : N

Marche aléatoire

Exercice

- Coder la recherche locale aléatoire (marche aléatoire)
- Afficher le graphique de la dynamique de recherche : nb d'évaluations *vs.* valeur de la fonction

Hill-Climber (HC) (ou steepest-descent)

Heuristique d'**exploitation** maximale.

Hill Climber (best-improvement)

Choisir solution initiale $s \in \mathcal{S}$
Evaluer s avec f
repeat
  Choisir $s' \in \mathcal{V}(s)$ telle que $f(s')$ est maximale
  if s' strictement meilleur que s **then**
    $s \leftarrow s'$
  end if
until s optimum local

- Algorithme de comparaison
- Opérateur local de base de métaheuristique

Une variante : first-improvement

Hill-climber First-improvement

Choisir solution initiale $s \in \mathcal{S}$
Evaluer s avec f
repeat
  Choisir $s^{'} \in \mathcal{V}(s)$ aléatoirement
  Evaluer $s^{'}$ avec f
  if $f(s) \leq f(s^{'})$ **then**
    $s \leftarrow s^{'}$
  end if
until s optimum local OU nbr d'éval. $\leq$ maxNbEval

Une variante : first-improvement

Hill-climber First-improvement

Choisir solution initiale $s \in \mathcal{S}$
Evaluer s avec f
repeat
  Choisir $s^{'} \in \mathcal{V}(s)$ aléatoirement
  Evaluer $s^{'}$ avec f
  if $f(s) \leq f(s^{'})$ **then**
    $s \leftarrow s^{'}$
  end if
until s optimum local OU nbr d'éval. $\leq$ maxNbEval

Quelle est l'avantage de cet algorithme par rapport au Hill-Climber Best-improvement ?

Un exemple très simple

Exercice

- Coder les recherches Hill-Climber best-improvement et Hill-Climber first-improvement
- Evaluer les performances (en nombre d'évaluation) de ces recherches et comparer les.