Les méthodes de résolution approchées pour le Programmation en nombres entiers

Amélie Lambert

Cnam

ECE 2016-2017

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Rappel : Comment dérober le maximum ?

Un malfaiteur arrive à s'introduire à l'intérieur d'une banque

Rappel : Comment dérober le maximum ?

Un malfaiteur arrive à s'introduire à l'intérieur d'une banque

Il peut voler

- des barres d'or
- des liasses de billets

Problème : Son sac à dos a :

- un volume max : 32 litres
- une charge max : 20kg

Rappel : Comment dérober le maximum ?

Un malfaiteur arrive à s'introduire à l'intérieur d'une banque

Il peut voler

- des barres d'or
- des liasses de billets

Problème : Son sac à dos a :

- un volume max : 32 litres
- une charge max : 20kg

- une barre d'or : 300000 $, 8kg, 6litres
- un paquet de billets : 100000 $, 3kg, 6litres

Rappel : Comment dérober le maximum ?

Un malfaiteur arrive à s'introduire à l'intérieur d'une banque

Il peut voler

- des barres d'or
- des liasses de billets

Problème : Son sac à dos a :

- un volume max : 32 litres
- une charge max : 20kg

- une barre d'or : 300000 $, 8kg, 6litres
- un paquet de billets : 100000 $, 3kg, 6litres

$\Rightarrow$ **Modélisation du problème**

Modélisation du problème

Utilisation de la programmation linéaire en nombres entiers.

Variables :

- variable x_1 nombre de barres d'or (300000\$, 8 kg, 6 litres)
- variable x_2 nombre de paquets de billets (100000\$, 3 kg, 6 litres)

Modélisation du problème

Utilisation de la programmation linéaire en nombres entiers.

Variables :

- variable x_1 nombre de barres d'or (300000\$, 8 kg, 6 litres)
- variable x_2 nombre de paquets de billets (100000\$, 3 kg, 6 litres)

$$(PLNE) \begin{cases} \max \quad f(x_1, x_2) = 3x_1 + x_2 & \text{Maximiser le profit} \\ 8x_1 + 3x_2 \leq 20 & \text{Contrainte de charge} \\ 6x_1 + 6x_2 \leq 32 & \text{Contrainte de volume} \\ x_1, x_2 \in \mathbb{N} & \text{Contraintes d'intégrité} \end{cases}$$

Un algorithme de résolution approché

Algorithme :

Remplir le sac avec le maximum d'or et compléter avec des billets.

Un algorithme de résolution approché

Algorithme :

Remplir le sac avec le maximum d'or et compléter avec des billets.

Avec ces données le voleur a intérêt à dérober 2 barres d'or et 1 paquet de billets pour gagner 700000 $.

Un algorithme de résolution approché

Algorithme :

Remplir le sac avec le maximum d'or et compléter avec des billets.

Avec ces données le voleur a intérêt à dérober 2 barres d'or et 1 paquet de billets pour gagner 700000 $.

On trouve une solution, mais on ne sait pas si c'est la meilleure solution pour toute les instances de ce problème

$\Rightarrow$ notre algorithme est **Heuristique**

Un algorithme dit "heuristique"

Définition 1 : Une solution *réalisable* d'un (PLNE) est une solution x qui satisfait les contraintes du problème.

Un algorithme dit "heuristique"

Définition 1 : Une solution *réalisable* d'un (PLNE) est une solution x qui satisfait les contraintes du problème.

Définition 2 : Un algorithme de résolution *Heuristique* est un algorithme qui fournit une solution réalisable en un temps polynomial pour un problème $\mathcal{NP}$-difficile.

Comment dérober le maximum

Un malfaiteur arrive à s'introduire à l'intérieur d'une banque

Comment dérober le maximum

Un malfaiteur arrive à s'introduire à l'intérieur d'une banque

Caractéristiques :

- une barre d'or : 300000 $, 8kg, 6litres
- un paquet de billets : 100000 $, 3kg, 6litres

Solution heuristique : l'or d'abord

- 2 lingots + 1 liasse
- profit : $2 * 3 + 1 * 1 = 7$

Résolution approchée d'un problème

Résoudre un P(L)NE peut prendre longtemps !

Mais si on a besoin d'une solution rapidement ?

Idée : rechercher une "bonne" solution admissible (borne primale). Cette solution n'est pas nécessairement optimale, mais meilleure que la plupart des autres solutions admissibles !

Intérêt : recherche rapide

Ces solutions (et les algorithmes pour les rechercher) sont dit(e)s **approché(e)s, ou heuristiques**

Possibilité de validation par des bornes duales (et même recommandé !)

Algorithmes approchés

Algorithmes approchés particuliers (heuristiques)

- Méthodes par Séparation-Evaluation avortées, méthode gloutonne, arrondi de la solution optimale de la relaxation continue, algorithme par recherche locale, ...

Algorithmes approchés

Algorithmes approchés particuliers (heuristiques)

- Méthodes par Séparation-Evaluation avortées, méthode gloutonne, arrondi de la solution optimale de la relaxation continue, algorithme par recherche locale, ...

Méthodes génériques (métaheuristiques)

- Algorithmes mono-solution :
  - Variable Neighborhood Descent (VND), Variable Neighborhood Search (VNS), recherche tabou, recuit simulé, ...
- Algorithmes multi-solutions
  - Algorithmes génétiques, colonies de fourmis, ...

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Méthodes par Séparation-Evaluation avortées

Principe : exécuter partiellement une méthode par Séparation-Evaluation, et à l'arrêter quand un critère d'arrêt est vérifié :

- Temps limite : risque de ne pas avoir de solution réalisable (borne primale) !
- Ecart prédéfini entre borne primale / borne duale
- Critère mixte : temps limite si borne primale existe OU écart prédéfini entre borne primale/duale

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Heuristique par arrondi de la solution

Principe : mettre à 1 toutes les variables qui valent 1 dans la solution optimale de la RC (et à 0 les autres).

Heuristique par arrondi de la solution

Principe : mettre à 1 toutes les variables qui valent 1 dans la solution optimale de la RC (et à 0 les autres).

Fournit-il une bonne solution entière ?

Exemple : le sac à dos

$$(SAD) \begin{cases} \max \quad \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x \in \{0,1\}^n \end{cases}$$

Exemple : le sac à dos

$$(SAD) \begin{cases} \max \quad \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x \in \{0,1\}^n \end{cases}$$

Résolution de la relaxation continue :

1. Trier les objets dans l'ordre décroissant des ratios $\frac{c_i}{a_i}$
2. Charger le sac à dos dans cet ordre jusqu'à sa capacité maximale
   $\Longleftrightarrow$ mettre à 1 toutes les variables possible et couper la dernière

Exemple : le sac à dos

$$(SAD)\begin{cases} \max \quad \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x \in \{0,1\}^n \end{cases}$$

Résolution de la relaxation continue :

1. Trier les objets dans l'ordre décroissant des ratios $\frac{c_i}{a_i}$
2. Charger le sac à dos dans cet ordre jusqu'à sa capacité maximale
   $\Longleftrightarrow$ mettre à 1 toutes les variables possible et couper la dernière

Algorithme 1 : arrondir la solution continue pour obtenir une solution entière : $\Longrightarrow$ mettre à 1 toutes les variables qui valent 1 dans la solution optimale de la RC (et à 0 les autres).

Exemple : le sac à dos

Soit le sac à dos suivant :

$$(SAD) \begin{cases} \max & x + (b-1)y \\ & x + by \leq b \\ & x, y \in \{0,1\}^2 \end{cases}$$

Exemple : le sac à dos

Soit le sac à dos suivant :

$$(SAD) \begin{cases} \max & x + (b-1)y \\ & x + by \leq b \\ & x, y \in \{0,1\}^2 \end{cases}$$

- On prend $x = 1$ et $y = 0$, car $\frac{1}{1} > \frac{b-1}{b}$

Exemple : le sac à dos

Soit le sac à dos suivant :

$$(SAD) \begin{cases} \max & x + (b-1)y \\ & x + by \leq b \\ & x, y \in \{0,1\}^2 \end{cases}$$

- On prend $x = 1$ et $y = 0$, car $\frac{1}{1} > \frac{b-1}{b}$
- Valeur de la solution (entière) obtenue par l'algorithme $1 = 1$

Exemple : le sac à dos

Soit le sac à dos suivant :

$$(SAD) \begin{cases} \max & x + (b-1)y \\ & x + by \leq b \\ & x, y \in \{0,1\}^2 \end{cases}$$

- On prend $x = 1$ et $y = 0$, car $\frac{1}{1} > \frac{b-1}{b}$
- Valeur de la solution (entière) obtenue par l'algorithme $1 = 1$
- Mais, valeur de la solution (entière) optimale $= b - 1$!

Exemple : le sac à dos

Algorithme 2 : prendre la meilleure de 2 solutions suivantes :

- la solution de l'algorithme 1
- Choisir le 1er objet non inclus dans le SAD dans l'algorithme 1
  $\Longleftrightarrow$ le premier qui à une valeur < 1 , on note $j + 1$ son indice

Exemple : le sac à dos

Algorithme 2 : prendre la meilleure de 2 solutions suivantes :

- la solution de l'algorithme 1
- Choisir le 1er objet non inclus dans le SAD dans l'algorithme 1
  $\Longleftrightarrow$ le premier qui à une valeur < 1 , on note $j+1$ son indice
- Algorithme 1 : valeur $\sum_{i=1}^{j} c_i$
- Algorithme 2 : valeur c_{j+1}
- Solution approchée de valeur $\max\{\sum_{i=1}^{j} c_i, c_{j+1}\}$

Algorithme approché spécifique : le sac-à-dos (3/3)

Avec cette variante, on peut prouver que la valeur de la solution admissible calculée est au moins la moitié de la valeur d'une solution optimale !

Algorithme approché spécifique : le sac-à-dos (3/3)

Avec cette variante, on peut prouver que la valeur de la solution admissible calculée est au moins la moitié de la valeur d'une solution optimale !

Cela nous fournit une **garantie de performance** a priori, mais l'algorithme peut être bien meilleur en pratique

Exercice 1

appliquez ces deux algorithmes sur le problème suivant :

$$(SAD)\begin{cases} \max & x_1 + x_2 + 5x_3 + 3x_4 \\ & 3x_1 + 2x_2 + 4x_3 + 2x_4 \leq 5 \\ & x \in \{0,1\}^4 \end{cases}$$

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Heuristique par méthode gloutonne

Définition 3 : Un algorithme *glouton* fait, étape par étape, le choix d'un optimum local.

Heuristique par méthode gloutonne

Définition 3 : Un algorithme *glouton* fait, étape par étape, le choix d'un optimum local.

Le problème du sac à dos.

Approche gloutonne Trier les objets dans l'ordre décroissant des ratios $\frac{c_i}{a_i}$
Tant qu'il reste de la place dans le sac à dos faire

- Charger le sac à dos dans cet ordre jusqu'à sa capacité maximale
  $\Longleftrightarrow$ mettre à 1 toutes les variables possible

Exercice 2

Appliquez cet algorithme sur le problème suivant :

$$(SAD)\begin{cases} \max & x_1 + x_2 + 5x_3 + 3x_4 \\ & 3x_1 + 2x_2 + 4x_3 + 2x_4 \leq 5 \\ & x \in \{0,1\}^4 \end{cases}$$

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Le voisinage d'une solution

Définition 6 : Le *voisinage* d'une solution donnée x est l'ensemble de solutions réalisables du problème de départ qui sont obtenues à partir de x via une transformation élémentaire.

Le voisinage d'une solution

Définition 6 : Le *voisinage* d'une solution donnée x est l'ensemble de solutions réalisables du problème de départ qui sont obtenues à partir de x via une transformation élémentaire.

Observation 3 : On dit que ces solutions sont "proches" de x. La transformation élémentaire nécéssite d'être définie selon les besoins de l'algorithme. La taille d'un voisinage est variable et est au maximum égale au cardinal de l'ensemble des solutions réalisables.

Le voisinage d'une solution

Définition 6 : Le *voisinage* d'une solution donnée x est l'ensemble de solutions réalisables du problème de départ qui sont obtenues à partir de x via une transformation élémentaire.

Observation 3 : On dit que ces solutions sont "proches" de x. La transformation élémentaire nécéssite d'être définie selon les besoins de l'algorithme. La taille d'un voisinage est variable et est au maximum égale au cardinal de l'ensemble des solutions réalisables.

Définition 7 : Une solution est un *optimum local* si elle est meilleure que toutes les autres solutions d'un voisinage prédéfini

Le voisinage d'une solution

Définition 6 : Le *voisinage* d'une solution donnée x est l'ensemble de solutions réalisables du problème de départ qui sont obtenues à partir de x via une transformation élémentaire.

Observation 3 : On dit que ces solutions sont "proches" de x. La transformation élémentaire nécéssite d'être définie selon les besoins de l'algorithme. La taille d'un voisinage est variable et est au maximum égale au cardinal de l'ensemble des solutions réalisables.

Définition 7 : Une solution est un *optimum local* si elle est meilleure que toutes les autres solutions d'un voisinage prédéfini

Idée : rechercher une ou des solutions "localement optimales"

Voisinage et transformations élémentaires

Définir le voisinage correspond à définir la/les transformation(s) élémentaire(s) !

Voisinage et transformations élémentaires

Définir le voisinage correspond à définir la/les transformation(s) élémentaire(s) !

Une transformation élémentaire doit être suffisamment simple

Voisinage et transformations élémentaires

Définir le voisinage correspond à définir la/les transformation(s) élémentaire(s) !

Une transformation élémentaire doit être suffisamment simple

Sac à dos :
Echanger (au plus) k objets : cela correspond à modifier au moins une valeur dans le vecteur solution :

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \longrightarrow \begin{pmatrix} \mathbf{1} \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

La distance dite de "Hamming" correspond nombres de bits qui diffèrent.

Trouver un optimum local : par recherche locale (RL)

On note :

- f la fonction objectif que l'on minimise
- $V(x)$ un voisinage (voisins admissibles) de la sol. x

```
Recherche locale()
DEBUT
  Choisir une solution initiale x0 (obtenue par une méthode heuristique par
exemple)
  x → x0
  TANT QUE x n'est pas un minimum local  FAIRE
   Choisir x' ∈ V(x) tel que f(x') < f(x)
   x → x'
  FIN TANT QUE
FIN
```

Illustration du voisinage local

Remarques :

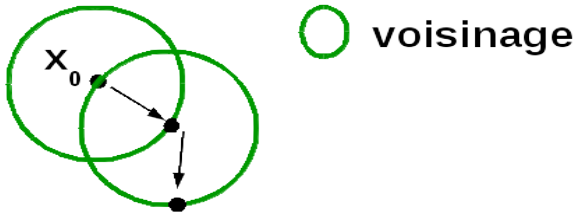
- La solution initiale x_0 peut être obtenue par une méthode heuristique quelconque
- Le critère d'arrêt est que le meilleur voisin n'améliore pas $\Rightarrow$ on a bien un optimum local pour le voisinage

Illustration du voisinage local

Remarques :

- La solution initiale x_0 peut être obtenue par une méthode heuristique quelconque
- Le critère d'arrêt est que le meilleur voisin n'améliore pas
  $\Rightarrow$ on a bien un optimum local pour le voisinage

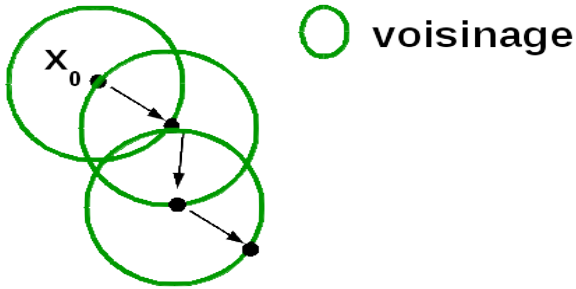
Illustration du voisinage local



Remarques :

- La solution initiale x_0 peut être obtenue par une méthode heuristique quelconque
- Le critère d'arrêt est que le meilleur voisin n'améliore pas
  $\Rightarrow$ on a bien un optimum local pour le voisinage

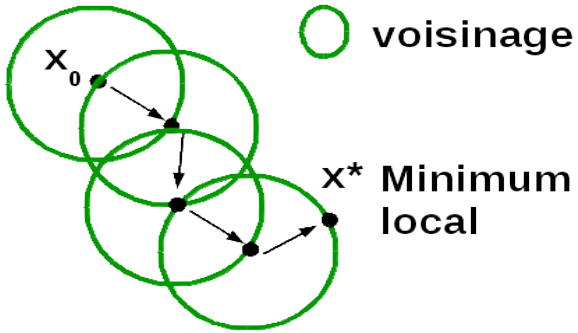
Illustration du voisinage local



Remarques :

- La solution initiale x_0 peut être obtenue par une méthode heuristique quelconque
- Le critère d'arrêt est que le meilleur voisin n'améliore pas
  $\Rightarrow$ on a bien un optimum local pour le voisinage

Illustration du voisinage local



Remarques :

- La solution initiale x_0 peut être obtenue par une méthode heuristique quelconque
- Le critère d'arrêt est que le meilleur voisin n'améliore pas
  $\Rightarrow$ on a bien un optimum local pour le voisinage

Exercice 3

En partant de la solution $x = (0, 0, 0, 0)$, appliquez cet algorithme sur le problème suivant :

$$(SAD) \begin{cases} \max & x_1 + x_2 + 5x_3 + 3x_4 \\ & 3x_1 + 2x_2 + 4x_3 + 2x_4 \leq 5 \\ & x \in \{0, 1\}^4 \end{cases}$$

Optimalité locale vs globale

Définition 8 : Une solution est un *optimum global* si elle est un optimum local quelque soit le voisinage, et donc en particulier si le voisinage correspond à l'ensemble des solutions réalisables.

Optimalité locale vs globale

Définition 8 : Une solution est un *optimum global* si elle est un optimum local quelque soit le voisinage, et donc en particulier si le voisinage correspond à l'ensemble des solutions réalisables.

Attention : Un optimal local n'est pas forcément un optimum global

Optimalité locale vs globale

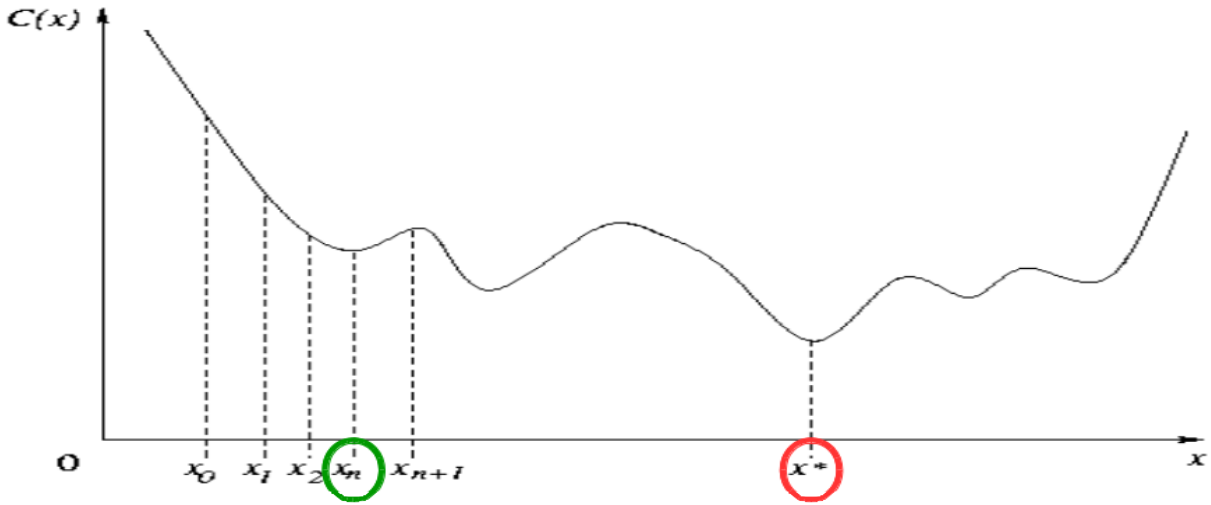
Intérêt de rechercher des optima locaux ?

- Il est possible de tomber sur l'optimum global
- Méthodes relativement efficaces pour le faire

Optimalité locale vs globale

Intérêt de rechercher des optima locaux ?

- Il est possible de tomber sur l'optimum global
- Méthodes relativement efficaces pour le faire

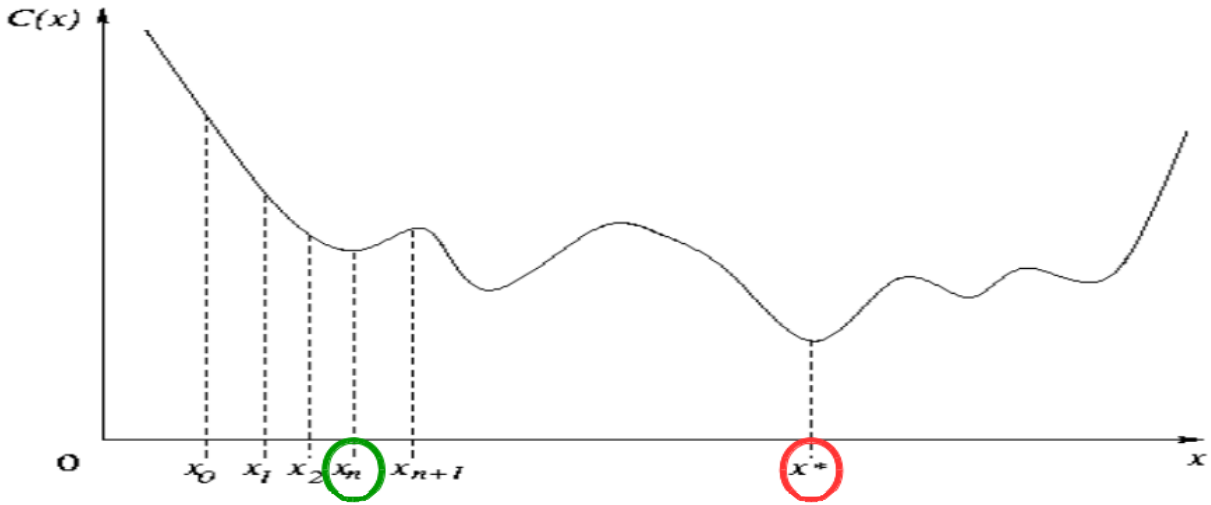


Piège à éviter : rester bloqué dans un optimum **local**, loin du **global**

Optimalité locale vs globale

Intérêt de rechercher des optima locaux ?

- Il est possible de tomber sur l'optimum global
- Méthodes relativement efficaces pour le faire



Piège à éviter : rester bloqué dans un optimum **local**, loin du **global**

Pour cela, il faut bien définir le voisinage ET réussir à s'extraire du voisinage d'un optimum local : utilisation des métaheuristiques.

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Les métaheuristiques

Définition 4 : Un algorithme de résolution *métaheuristique* est un algorithme heuristique "générique" qu'il faut adapter à chaque problème.

Les métaheuristiques

Définition 4 : Un algorithme de résolution *métaheuristique* est un algorithme heuristique "générique" qu'il faut adapter à chaque problème.

Il peut être basé sur plusieurs **principes communs :**

- Voisinage
- Recherche locale

Les métaheuristiques

Définition 4 : Un algorithme de résolution *métaheuristique* est un algorithme heuristique "générique" qu'il faut adapter à chaque problème.

Il peut être basé sur plusieurs **principes communs :**

- Voisinage
- Recherche locale

Et sur un **schéma commun :**

- Un algorithme générique (souvent) stochastique,
- Un ensemble de paramètres, qui permet d'adapter l'algorithme à chaque problème particulier.

Les métaheuristiques

Définition 4 : Un algorithme de résolution *métaheuristique* est un algorithme heuristique "générique" qu'il faut adapter à chaque problème.

Il peut être basé sur plusieurs **principes communs :**

- Voisinage
- Recherche locale

Et sur un **schéma commun :**

- Un algorithme générique (souvent) stochastique,
- Un ensemble de paramètres, qui permet d'adapter l'algorithme à chaque problème particulier.

Rôle : l'idée est d'utiliser les métaheuristique pour s'extraire d'un optimum local obtenu par une méthode heuristique.

Comment s'extraire d'un optimum local ?

Algorithmes mono-solutions :

1. En faisant se basant sur plusieurs voisinages :
   - *Variable Neihborhood Descent (VND) :* definition de plusieurs structures de voisinages, pour tenter d'atteindre une solution meilleure.

Comment s'extraire d'un optimum local ?

Algorithmes mono-solutions :

1. En faisant se basant sur plusieurs voisinages :
   - *Variable Neihborhood Descent (VND) :* definition de plusieurs structures de voisinages, pour tenter d'atteindre une solution meilleure.

2. S'autoriser à dégrader la valeur de la fonction objectif :
   - *Variable Neihborhood Search (VNS) :* du VND avec la possiblité de dégrader la meilleure solution courante.
   - *Recherche tabou :* Si, après "apprentissage", aucun autre mouvement n'est possible.
   - *Recuit simulé :* Avec une certaine probabilité.

Comment s'extraire d'un optimum local ?

Algorithmes mono-solutions :

1. En faisant se basant sur plusieurs voisinages :
   - *Variable Neihborhood Descent (VND) :* definition de plusieurs structures de voisinages, pour tenter d'atteindre une solution meilleure.

2. S'autoriser à dégrader la valeur de la fonction objectif :
   - *Variable Neihborhood Search (VNS) :* du VND avec la possiblité de dégrader la meilleure solution courante.
   - *Recherche tabou :* Si, après "apprentissage", aucun autre mouvement n'est possible.
   - *Recuit simulé :* Avec une certaine probabilité.

Algorithmes multi-solutions : Faire évoluer plusieurs solutions en parallèle (et donc explorer simultanément plusieurs zones) : Algorithmes distribués/évolutifs (génétiques, etc.)

Quelle métaheuristique choisir ?

Quelle est la plus efficace ? :

- Aucune, et toutes à la fois : cela dépend du problème !
- En "moyenne", toutes ont la même efficacité : seul résultat théorique réellement applicable

Quelle métaheuristique choisir ?

Quelle est la plus efficace ? :

- Aucune, et toutes à la fois : cela dépend du problème !
- En "moyenne", toutes ont la même efficacité : seul résultat théorique réellement applicable

Comment en choisir une, pour un problème donné ?

- Pas de règle absolue pour : choisir une métaheuristique ou régler ses paramètres
- Expérience (savoir-faire) et/ou expérimentations (comparaison expérimentale) !

Mise en oeuvre de métaheuristiques

Le réglage des paramètres varie en fonction de l'adaptation au problème :

- Définir une solution initiale
- Définir une structure de voisinage
- Définir les paramètres spécifiques

Mise en oeuvre de métaheuristiques

Le réglage des paramètres varie en fonction de l'adaptation au problème :

- Définir une solution initiale
- Définir une structure de voisinage
- Définir les paramètres spécifiques

Bon usage : comparer bornes primales et duales !

Mise en oeuvre de métaheuristiques

Le réglage des paramètres varie en fonction de l'adaptation au problème :

- Définir une solution initiale
- Définir une structure de voisinage
- Définir les paramètres spécifiques

Bon usage : comparer bornes primales et duales !

Critère d'arrêt :

- Temps limite
- Ecart de la valeur de la sol. courante à la borne duale
- Combinaison de ces deux critères

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Variable Neihborhood Descent (VND)

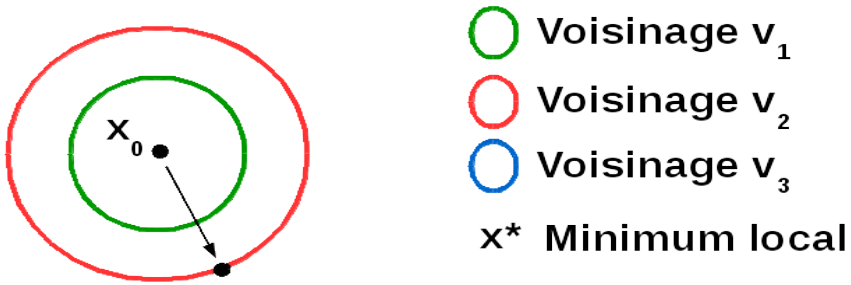
Principe :

1. On définit k voisinages v_k tels que $|v_1|<|v_2|< \ldots <|v_k|$.
2. On cherche une meilleure solution que x_0 dans v_1, pui dans v_2, etc ...
3. Lorsqu'une telle solution x_{best} est trouvée, on tente de l'améliorer d'abord dans le voisinage v_1, puis dans v_2, etc...
4. On s'arrête lorsqu'on de peut plus améliorer x_{best}.

Variable Neihborhood Descent (VND)

```
VND()
DEBUT
  Choisir une solution initiale x_0
  x_best → x_0
  TANT QUE x_best peut être amélioré FAIRE
   POUR i allant de 1 à k FAIRE
    POUR TOUT x' ∈ v_i(x) FAIRE
     SI f(x') < f(x_best) ALORS
       x_best → x'
       arrêter la boucle POUR
       arrêter la boucle POUR
     FIN SI
  FIN POUR
  FIN TANT POUR
  FIN TANT QUE
FIN
```

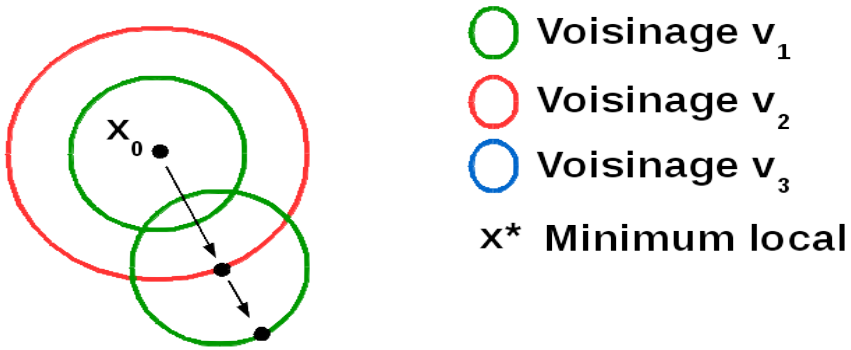
Illustration



Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

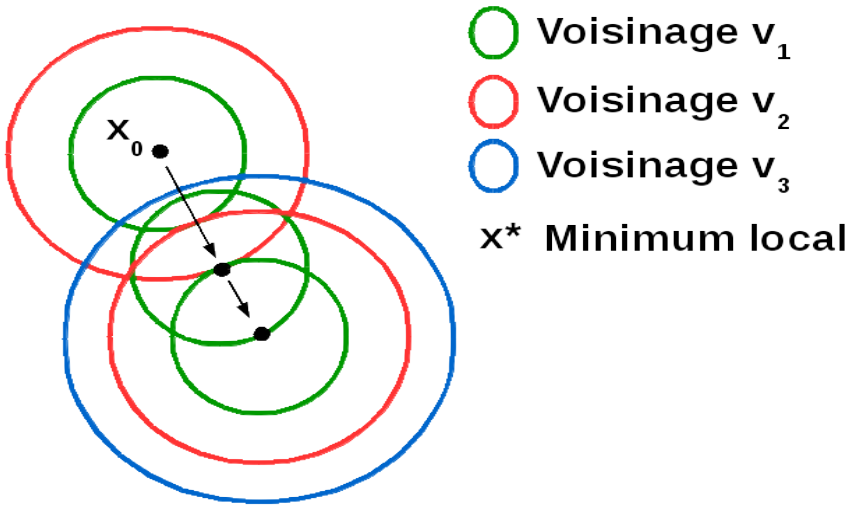
Illustration



Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

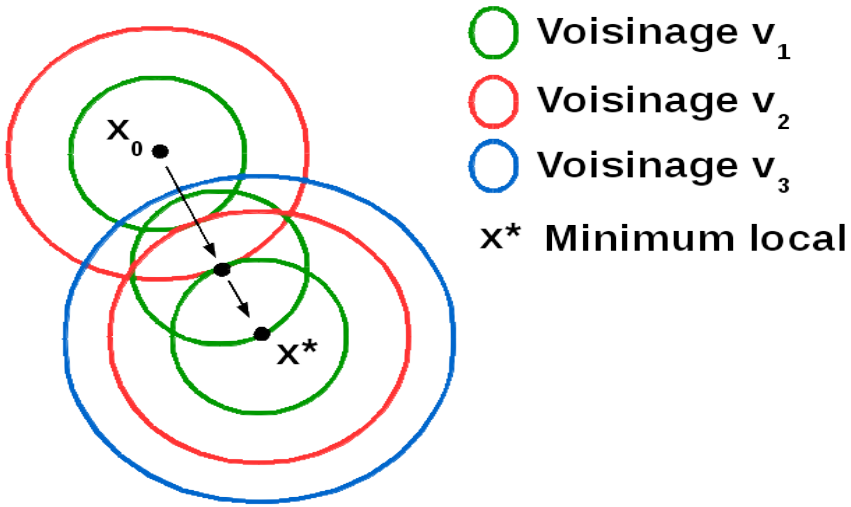
Illustration



Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

Illustration



Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Variable Neihborhood Search (VNS)

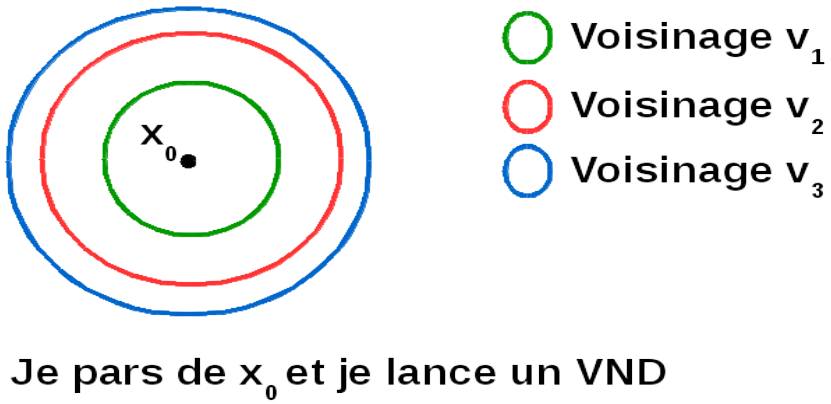
Principe :

1. On définit k voisinages v_k tels que $|v_1|<|v_2|< \ldots <|v_k|$.
2. On cherche une meilleure solution que x_0. Pour cela on va appliquer VND.
3. Si une telle solution x_{best} est trouvée, on tente de l'améliorer d'abord dans le voisinage v_1, puis dans v_2, ..
4. Sinon on est bloqué sur un optimum local, on tire au hasard une solution qui va devenir la solution à améliorer.
5. On s'arrête lorsqu'on de peut plus améliorer x_{best}.

Variable Neihborhood Search (VNS)

VNS()
DEBUT
  Choisir une solution initiale x_0
  $x_{\text{best}} \rightarrow x_0$
  TANT QUE x_{best} peut être amélioré FAIRE
   POUR i allant de 1 à k FAIRE
    Générer un point x' au hasard dans le i^{eme} voisinage de x ($x' \in v_i(x)$)
    Appliquer VND en partant de x', on note x'' l'optimum local trouvé.
    SI $f(x'') < f(x_{\text{best}})$ ALORS
     $x_{\text{best}} \rightarrow x''$
     i $\rightarrow$ 1
    FIN SI
   FIN POUR
  FIN TANT QUE
FIN

Illustration

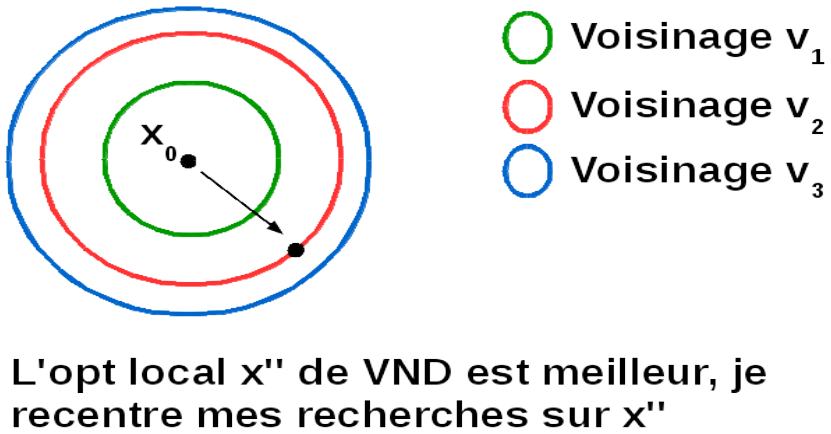


Je pars de x_0 et je lance un VND

Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

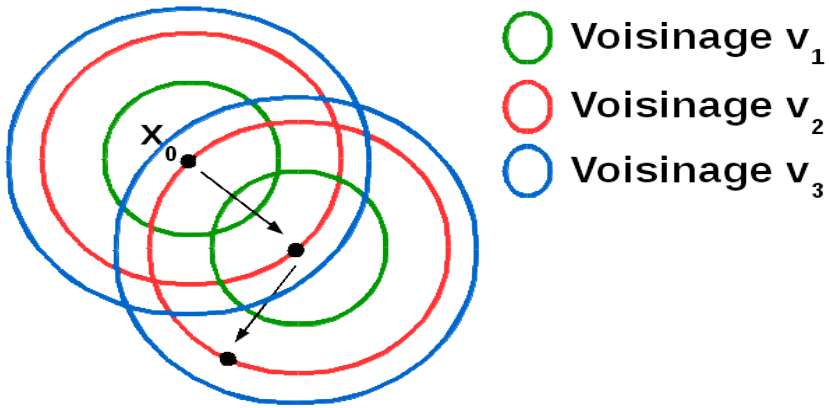
Illustration



Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

Illustration

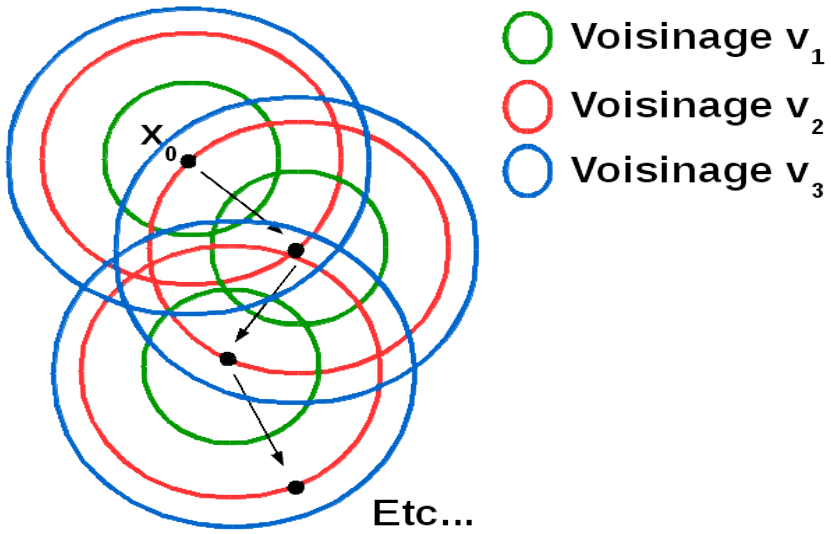


Pas d'opt local meilleur par VND, je genère une solution x' dans le vois. v_2

Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

Illustration



Remarques :

- Il n'y a pas forcement $v_1 \in v_2 \in \ldots \in v_k$
- Cet algorithme nécéssite donc la définition de plusieurs structures de voisinage.
- Le critère d'arrêt est de ne trouver aucune solution améliorante dans les k voisinages.

Exercice 4

En partant de la solution $x = (0, 1, 0, 1)$ et pour $k = 3$, appliquez cet algorithme sur le problème suivant :

$$(SAD) \begin{cases} \max & x_1 + x_2 + 5x_3 + 3x_4 \\ & 3x_1 + 2x_2 + 4x_3 + 2x_4 \leq 5 \\ & x \in \{0, 1\}^4 \end{cases}$$

1. Introduction et définition

2. Les algorithmes approchés : heuristiques
   - Heuristique par Séparation-Evaluation avortée
   - Heuristique par arrondi de la solution
   - Heuristique par méthode gloutonne
   - Heuristique par recherche locale

3. Les métaheuristique
   - La métaheuristique Variable Neihborhood Descent (VND)
   - La métaheuristique Variable Neihborhood Search (VNS)
   - La métaheuristique Tabou

Recherche tabou (RT) - Tabu search (1/4)

Principe : associer un mécanisme de mémoire à une recherche locale classique

Recherche tabou (RT) - Tabu search (1/4)

Principe : associer un mécanisme de mémoire à une recherche locale classique

Plus précisément, on construit une liste $\mathcal{T}$ (liste tabou) contenant les p dernières sol. visitées : p dépend du pb et des instances considéré(es)

Recherche tabou (RT) - Tabu search (1/4)

Principe : associer un mécanisme de mémoire à une recherche locale classique

Plus précisément, on construit une liste $\mathcal{T}$ (liste tabou) contenant les p dernières sol. visitées : p dépend du pb et des instances considéré(es)

A partir de la solution courante, on parcourt le voisinage, et on se déplace vers le meilleur voisin, tout en modifiant la liste tabou au besoin

Recherche tabou (RT) - Tabu search (2/4)

RT()
DEBUT
  Choisir une solution initiale x
  $x_{\text{best}} \rightarrow x$
  $\mathcal{T} = \emptyset$
  TANT QUE x_{best} peut être amélioré FAIRE
    Appliquer une recherche locale en partant x sur le voisinage de x moins les éléments $\mathcal{T}$. Soit x' la solution obtenue
    $\mathcal{T} \rightarrow \mathcal{T} \cup x'$ (éliminer si besoin le plus ancien élément de $\mathcal{T}$).
    SI $f(x') < f(x_{\text{best}})$ ALORS
      $x_{\text{best}} \rightarrow x'$
    FIN SI
  FIN TANT QUE
FIN

Recherche tabou (RT) - Tabu search (3/4)

Paramètre (choix concepteur) : taille p de la liste tabou $\mathcal{T}$, qui est gérée comme une file (FIFO) (typiquement, $p \in [5, 10]$)

Recherche tabou (RT) - Tabu search (3/4)

Paramètre (choix concepteur) : taille p de la liste tabou $\mathcal{T}$, qui est gérée comme une file (FIFO) (typiquement, $p \in [5, 10]$)

En pratique, on stocke les transformations dans $\mathcal{T}$, et non les solutions complètes $\Rightarrow$ Efficacité (temps, espace) vs perte d'information

Recherche tabou (RT) - Tabu search (3/4)

Paramètre (choix concepteur) : taille p de la liste tabou $\mathcal{T}$, qui est gérée comme une file (FIFO) (typiquement, $p \in [5, 10]$)

En pratique, on stocke les transformations dans $\mathcal{T}$, et non les solutions complètes $\Rightarrow$ Efficacité (temps, espace) vs perte d'information

Capacité de la RT à s'extraire des min. locaux ?

- Si $\min_{x' \in V(x)} f(x') > f(x)$, on "remonte" (on effectue une transformation qui accroît la fonction objectif f)
- Ce mécanisme est susceptible de permettre d'échapper à un minimum local

Recherche tabou (RT) - Tabu search (4/4)

Conditions d'arrêt :

Recherche tabou (RT) - Tabu search (4/4)

Conditions d'arrêt :

- Conditions déjà évoquées
- $V(x) = \emptyset$ (rare)
- Nb maximum d'itérations sans améliorations

Exercice 5

En partant de la solution $x = (0, 1, 0, 1)$ et en considérant un voisinage avec une distance de hamming de 1 et une liste taboue de taille 3, appliquez cet algorithme sur le problème suivant :

$$(SAD) \begin{cases} \max & x_1 + x_2 + 5x_3 + 3x_4 \\ & 3x_1 + 2x_2 + 4x_3 + 2x_4 \leq 5 \\ & x \in \{0, 1\}^4 \end{cases}$$

Avantages des métaheuristiques

Souvent efficaces en pratique

Avantages des métaheuristiques

Souvent efficaces en pratique

Pas d'algorithme spécifique à concevoir pour chaque problème

Avantages des métaheuristiques

Souvent efficaces en pratique

Pas d'algorithme spécifique à concevoir pour chaque problème

N'utilisent pas la formulation P(L)NE, donc indépendantes d'une éventuelle non linéarité.

Avantages des métaheuristiques

Souvent efficaces en pratique

Pas d'algorithme spécifique à concevoir pour chaque problème

N'utilisent pas la formulation P(L)NE, donc indépendantes d'une éventuelle non linéarité.

La diversité des métaheuristiques existantes offre une certaine souplesse, car si une métaheuristique est inefficace sur un problème donné, on peut en essayer d'autres, avec une mise en oeuvre rapide.

Avantages des métaheuristiques

Souvent efficaces en pratique

Pas d'algorithme spécifique à concevoir pour chaque problème

N'utilisent pas la formulation P(L)NE, donc indépendantes d'une éventuelle non linéarité.

La diversité des métaheuristiques existantes offre une certaine souplesse, car si une métaheuristique est inefficace sur un problème donné, on peut en essayer d'autres, avec une mise en oeuvre rapide.

Peuvent être utilisées avant une méthode par Séparation-Evaluation

Inconvénients des métaheuristiques :

Garantie de performance a posteriori (la plupart du temps, pas d'analyse théorique possible)

Inconvénients des métaheuristiques :

Garantie de performance a posteriori (la plupart du temps, pas d'analyse théorique possible)

Nécessité de régler des paramètres spécifiques aux pb par des expérimentations (pas d'anticipation).

Inconvénients des métaheuristiques :

Garantie de performance a posteriori (la plupart du temps, pas d'analyse théorique possible)

Nécessité de régler des paramètres spécifiques aux pb par des expérimentations (pas d'anticipation).

Comme pour les algorithmes approchés spécifiques, difficiles (impossibles ?) à utiliser si trouver une borne primale (solution initiale) est difficile pour le problème considéré.