

Chapitre 6: Meta-heuristiques sur une solution (version avec réponses) Algorithmes de recherche locale

F. Bannay

(d'après les supports de J. Mengin et P. Muller,
et le livre (Talbi 2009))



2020-2021

Les deux méthodes itératives vues dans la partie B

Recherche locale

F. Bannay

Introduction

Iter/Constr

Exactes/Incompl.

Quand ?

Algo R. Locale

Optima locaux

Conclusion

Références

Méthode **itérative** : on part d'une solution complètement décrite et on la fait évoluer vers une meilleure.

■ Algo de Ford-Fulkerson :

- on part d'un flot compatible

Repeat

- marquage: chaîne augmentante
↪ nouveau flot compatible de valeur supérieure

Until on ne peut plus augmenter le flot

■ Algo du Simplexe :

- on part d'une base réalisable

Repeat

- une variable sort et une variable rentre
↪ nouvelle base réalisable avec meilleur gain

Until on ne peut plus augmenter le gain

Les deux méthodes itératives vues dans la partie B

Recherche locale

F. Bannay

Introduction

Iter/Constr

Exactes/Incompl.

Quand ?

Algo R. Locale

Optima locaux

Conclusion

Références

Méthode **itérative** : on part d'une solution complètement décrite et on la fait évoluer vers une meilleure.

■ Algo de Ford-Fulkerson :

- on part d'un flot compatible

Repeat

- marquage: chaîne augmentante
↪ nouveau flot compatible de valeur supérieure

Until on ne peut plus augmenter le flot

■ Algo du Simplexe :

- on part d'une base réalisable

Repeat

- une variable sort et une variable rentre
↪ nouvelle base réalisable avec meilleur gain

Until on ne peut plus augmenter le gain

Les méthodes constructives

Recherche locale

F. Bannay

Introduction

Iter/Constr

Exactes/Incompl.

Quand ?

Algo R. Locale

Optima locaux

Conclusion

Références

- Méthode **constructive** : on part d'une solution vide et on affecte les variables une à une jusqu'à obtenir une solution complètement décrite.
- Algo de Kruskal (arbre couvrant de poids min)
 - on part d'un ensemble vide d'arêtes

Repeat

 - sélectionner une arête de poids minimum
 - l'ajouter si pas de cycle sinon la jeter

Until graphe connexe

Méthodes Exactes vs Incomplètes

Recherche locale

F. Bannay

Introduction

Iter/Constr

Exactes/Incompl.

Quand ?

Algo R. Locale

Optima locaux

Conclusion

Références

- Méthode exacte :
 - si solution optimale existe alors l'algo la trouve.
 - exploration systématique de l'espace de recherche
- Pour de nombreux problèmes : pas de méthode exacte efficace.
- Problème d'explosion combinatoire
 - ex : voyageur de commerce avec n villes
 - ↪ $n!$ séquences possibles
- Idée :
 - méthodes **incomplètes** **suffisamment** efficaces dans certains cas
 - ↪ les **meta-heuristiques**.

Méthodes Exactes vs Incomplètes (suite)

Recherche locale

F. Bannay

Introduction

Iter/Constr

Exactes/Incompl.

Quand ?

Algo R. Locale

Optima locaux

Conclusion

Références

Méthodes exactes	Méthodes Incomplètes
<ul style="list-style-type: none">■ Programmation dynamique■ Branch and bound/ cut / price ; A* (heuristique), IDA*■ Programmation par contraintes ; Programmation Linéaire ; Flots■ ...	<ul style="list-style-type: none">■ Algo d'approximation : garantie écart solution optimale $\leq \varepsilon$■ Algo Heuristiques : spécifiques / meta-heuristiques (une solution/ une population)■ ...

Quand utiliser des meta-heuristiques ?

Recherche locale

F. Bannay

Introduction

Iter/Constr

Exactes/Incompl.

Quand ?

Algo R. Locale

Optima locaux

Conclusion

Références

■ Complexité du problème et taille des instances

Taille instance Complexité pb	Petite	Grande
Faible	exacte	exacte ou incomplète (temps réel)
Forte	exacte (peut suffire) ou incomplète	incomplète

- Structure des instances : des structures spécifiques peuvent être résolues par des méthodes exactes
- Moralité :
 - 1 étudier complexité
 - 2 réduire à un problème connu
 - 3 estimer faisabilité selon taille instance

Meta-heuristiques sur une solution = recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Algorithme générique de **recherche locale** :

- on part d'une solution s

Repeat

- choisir un voisin s' de s
- si s' meilleure_que s alors $s \leftarrow s'$

Until s suffisamment_bon ou time-out

solution : affectation de toutes les variables

voisin : solution obtenue par "petite" modification ("locale")

meilleure_que/suffisamment_bon : selon critère(s) à optimiser et tolérance (acceptable)

essai : séquence de choix de voisins

time-out : nombre max de voisins choisis atteint ou plus de voisins

Meta-heuristiques sur une solution = recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Algorithme générique de **recherche locale** :

- on part d'une **solution** s

Repeat

- choisir un voisin s' de s
- si s' meilleure_que s alors $s \leftarrow s'$

Until s suffisamment_bon ou time-out

solution : affectation de toutes les variables

voisin : solution obtenue par "petite" modification ("locale")

meilleure_que/suffisamment_bon : selon critère(s) à optimiser et tolérance (acceptable)

essai : séquence de choix de voisins

time-out : nombre max de voisins choisis atteint ou plus de voisins

Meta-heuristiques sur une solution = recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Algorithme générique de **recherche locale** :

- on part d'une solution s

Repeat

- choisir un **voisin** s' de s
- si s' meilleure_que s alors $s \leftarrow s'$

Until s suffisamment_bon ou time-out

solution : affectation de toutes les variables

voisin : solution obtenue par "petite" modification ("locale")

meilleure_que/suffisamment_bon : selon critère(s) à optimiser et tolérance (acceptable)

essai : séquence de choix de voisins

time-out : nombre max de voisins choisis atteint ou plus de voisins

Meta-heuristiques sur une solution = recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Algorithme générique de **recherche locale** :

- on part d'une solution s

Repeat

- choisir un voisin s' de s
- si s' **meilleure_que** s alors $s \leftarrow s'$

Until s **suffisamment_bon** ou time-out

solution : affectation de toutes les variables

voisin : solution obtenue par “petite” modification (“locale”)

meilleure_que/suffisamment_bon : selon critère(s) à optimiser et tolérance (acceptable)

essai : séquence de choix de voisins

time-out : nombre max de voisins choisis atteint ou plus de voisins

Meta-heuristiques sur une solution = recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Algorithme générique de **recherche locale** :

- on part d'une solution s

Repeat

- choisir un voisin s' de s
- si **acceptable_voisin**(s', s) alors $s \leftarrow s'$

Until **acceptable_solution**(s) ou time-out

solution : affectation de toutes les variables

voisin : solution obtenue par "petite" modification ("locale")

meilleure_que/suffisamment_bon : selon critère(s) à optimiser et tolérance (acceptable)

essai : séquence de choix de voisins

time-out : nombre max de voisins choisis atteint ou plus de voisins

Meta-heuristiques sur une solution = recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Algorithme générique de **recherche locale** :

- on part d'une solution s

Repeat

- choisir un voisin s' de s
- si `acceptable_voisin(s' , s)` alors $s \leftarrow s'$

Until `acceptable_solution(s)` ou **time-out**

solution : affectation de toutes les variables

voisin : solution obtenue par “petite” modification (“locale”)

meilleure_que/suffisamment_bon : selon critère(s) à optimiser et tolérance (acceptable)

essai : séquence de choix de voisins

time-out : nombre max de voisins choisis atteint ou plus de voisins

Meta-heuristiques sur une solution = recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Algorithme générique de **recherche locale** :

- on part d'une solution s
- $\text{nb_choisis}=0$

Repeat

- choisir un voisin s' de s
- si $\text{acceptable_voisin}(s',s)$ alors $s \leftarrow s'$
- $\text{nb_choisis}++$

Until $\text{acceptable_solution}(s)$ ou $\text{nb_choisis}=\text{MAX}$ ou $\text{vois}(s)=\emptyset$

solution : affectation de toutes les variables

voisin : solution obtenue par “petite” modification (“locale”)

meilleure_que/suffisamment_bon : selon critère(s) à optimiser et tolérance (acceptable)

essai : séquence de choix de voisins

time-out : nombre max de voisins choisis atteint ou plus de voisins

Les paramètres de la recherche locale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

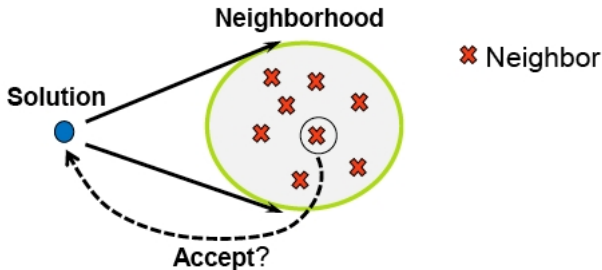
Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références



Paramètres à définir :

- représentation de la solution,
- choix d'une solution initiale
- choix d'un voisin,
- fonction d'évaluation de la solution (`acceptable_solution`)
- fonction d'évaluation d'un voisin (`acceptable_voisin`)
- voisinage = la génération des voisins

Exercice 1 : “solution”

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

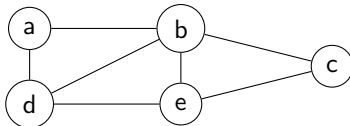
Hill-Climbing

Optima locaux

Conclusion

Références

- On cherche une chaîne binaire T de taille N telle que
$$\max \sum_{i=1}^N \sum_{j=1}^N (-1)^{3i+j} \cdot (2)^{i+7j} T[i] \cdot T[j]$$
- Proposez une représentation d’une “sol. réalisable” de ce pb.
- coloration de graphe t.q. somme des couleurs est min



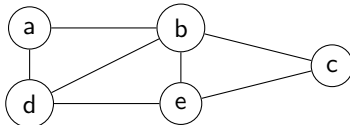
- Proposez une représentation d’une “sol. réalisable” de ce pb.
- voyageur de commerce avec N villes et une matrice $N \times N$ des distances
- Proposez une représentation d’une “sol. réalisable” de ce pb.

Exercice 1 : “solution” (réponses)

- On cherche une chaîne binaire T de taille N telle que $\max \sum_{i=1}^N \sum_{j=1}^N (-1)^{3i+j} \cdot (2)^{i+7j} T[i] \cdot T[j]$

→ “sol. réalisable” de ce pb : **un vecteur de N bits (0 ou 1).**

- coloration de graphe t.q. somme des couleurs est min



→ “sol. réalisable” de ce pb : **un vecteur de 5 chiffres entre 1 et 5 tels que le premier correspond à la couleur de a, le deuxième celle de b etc. et deux sommets adjacents n’ont pas la même couleur.**

- voyageur de commerce avec N villes et une matrice $N \times N$ des distances

→ “sol. réalisable” de ce pb : **un vecteur de N chiffres de 1 un N tel que chaque chiffre n’est présent qu’une seule fois.**

Choix d'un voisin

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Les choix de voisins couramment utilisés :

rapide calculer un seul voisin (n'améliore pas forcément)

plus long énumérer tous les voisins, garder le meilleur

intermédiaire générer échantillon de voisins, garder le meilleur

premier qui améliore calculer le premier voisin qui améliore (pire cas : explore tous les voisins)

amélioration aléatoire sélectionner aléatoirement un voisin parmi ceux qui améliorent

Illustration de voisinage (1)

Voisinage de solutions binaires : distance de Hamming

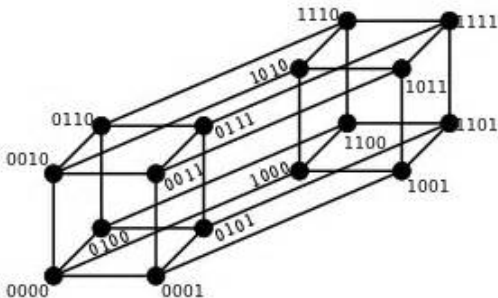
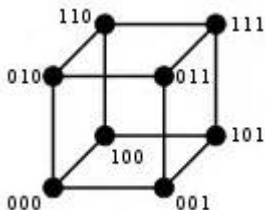


Illustration de voisinage (2)

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Voisinage de la solution C dans espace de dimension 2 : disque.
(Dans \mathbb{R}^n c'est une boule de dimension n).

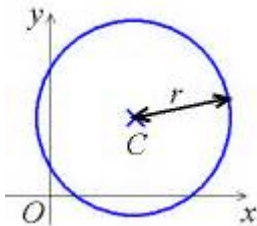
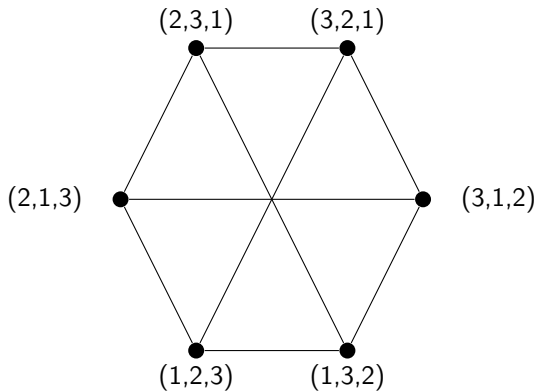


Illustration de voisinage (3)

Voisinage pour un problème de permutation de taille 3 (échange de 2 éléments)



Exercice 2 : voisins

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

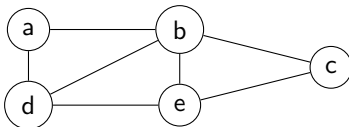
Hill-Climbing

Optima locaux

Conclusion

Références

- On appelle voisin d'une chaîne binaire, la chaîne obtenue en inversant un bit de cette chaîne. Combien de voisins à la chaîne 0101110 ?
- Coloration de :

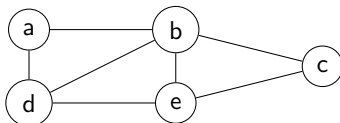


On propose de définir comme voisin d'une coloration le vecteur identique au précédent dans lequel on duplique un des chiffres en le recopiant à une autre position et on garde tous les autres, le vecteur obtenu est considéré comme voisin seulement s'il vérifie la propriété de coloration. Combien de voisins à la solution 12345 ?

- On propose de définir comme voisin d'une solution d'un voyageur de commerce à 6 villes, la solution résultant en échangeant deux villes du parcours. Combien de voisins a la solution 123456 ?

Exercice 2 : voisins (réponse)

- On appelle voisin d'une chaîne binaire, la chaîne obtenue en inversant un bit de cette chaîne. Combien de voisins à la chaîne 0101110? **7**
- Coloration de :



On propose de définir comme voisin d'une coloration le vecteur identique au précédent dans lequel on duplique un des chiffres en le recopiant à une autre position et on garde tous les autres, le vecteur obtenu est considéré comme voisin seulement s'il vérifie la propriété de coloration. Combien de voisins à la solution 12345? **6 : 12145, 12341, 32345, 12335, 12445, 52345**

- On propose de définir comme voisin d'une solution d'un voyageur de commerce à 6 villes, la solution résultant en échangeant deux villes du parcours. Combien de voisins a la solution 123456? **5+4+3+2+1=15**

Exemple de recherche locale : le hill-climbing

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Hill-climbing (ou *descente en gradient* si min recherché)

- choisir_un_voisin : choix aléatoire voisin
- acceptable_voisin(s', s) : si s' meilleur que s

méthode opportuniste

Hill-climbing version glouton

- choisir_un_voisin : choix d'un voisin qui améliore s
- acceptable_voisin(s', s) : toujours

[...]

Hill-Climbing Glouton

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

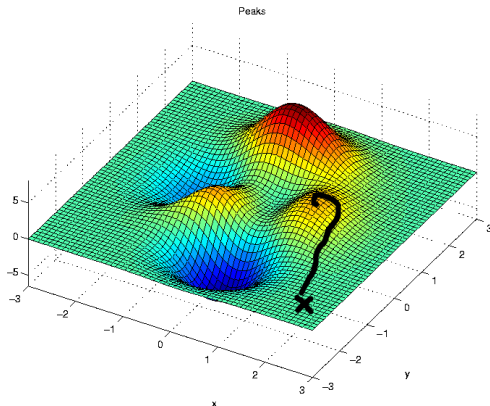
Hill-Climbing

Optima locaux

Conclusion

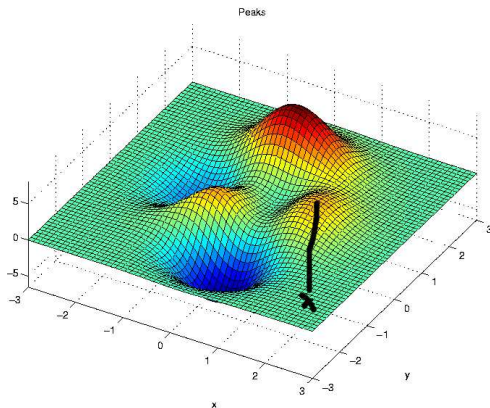
Références

En montant de façon à améliorer (hill-climbing glouton)



Steepest Hill-Climbing

En montant la plus forte pente (steepest hill-climbing) : meilleur voisin



Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

Hill-Climbing

Optima locaux

Conclusion

Références

Exercice 3 : steepest hill-climbing

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Paramètres

Solutions

Voisinage

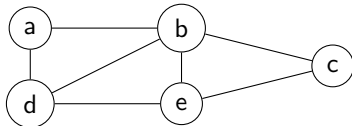
Hill-Climbing

Optima locaux

Conclusion

Références

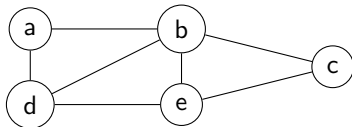
Coloration de somme minimale pour :



- On propose de définir comme voisin d'une coloration le vecteur identique au précédent dans lequel on duplique un des chiffres en le recopiant à une autre position et on garde tous les autres, le vecteur obtenu est considéré comme voisin seulement s'il vérifie la propriété de coloration.
- Décrivez les solutions successives parcourues par un steepest Hill Climbing depuis la solution 12345.

Exercice 3 : steepest hill-climbing (réponse)

Coloration de somme minimale pour :



- On propose de définir comme voisin d'une coloration le vecteur identique au précédent dans lequel on duplique un des chiffres en le recopiant à une autre position et on garde tous les autres, le vecteur obtenu est considéré comme voisin seulement s'il vérifie la propriété de coloration.
- Les solutions successives parcourues par un steepest Hill Climbing depuis la solution 12345 sont :
 - **12345 puis on choisit le meilleur parmi (12145, 12341, 32345, 12335, 12445, 52345) : 12341 puis on choisit le meilleur parmi (12331, 32341, 12441) : 12331 puis stop plus de voisin.**

Optimum local/global

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Relaxation

Conclusion

Références

Un **optimum local** est une solution dont tous les voisins sont moins bons.

- Problème important : un **optimum local** n'est pas nécessairement un **optimum global**
 - un optimum local en Programmation Linéaire ou pour les flots est un optimum global
 - ce n'est pas le cas pour les problèmes d'optimisation combinatoire en général
 - la recherche locale peut se retrouver coincée dans un optimum local (tous ses voisins sont moins bons que lui)
- moins de chance d'en rencontrer si voisinage **grand**
- mais grand voisinage = modifs complexes et plus de voisins (à l'extrême : voisinage = tout l'espace de recherche...)

Illustration dans le cas de fonctions continues : 1D

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Relaxation

Conclusion

Références

Descente en gradient (hill-climbing glouton)

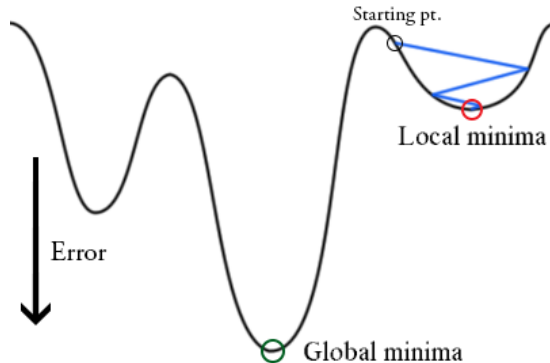


Illustration dans le cas de fonctions continues : 2D

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

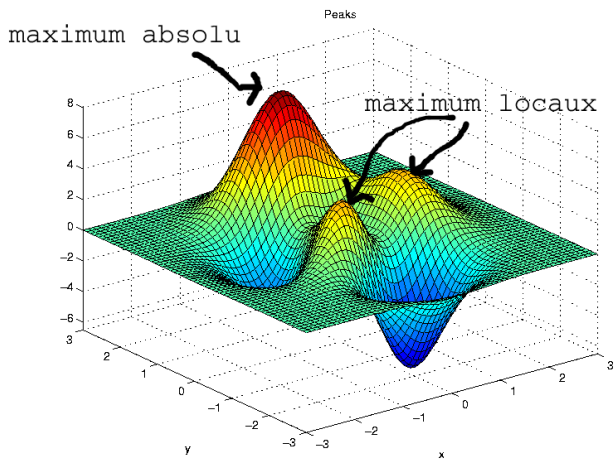
R. locale itérée

Voisins pires

Relaxation

Conclusion

Références



Exercice 4 : Optimum local ou global ?

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

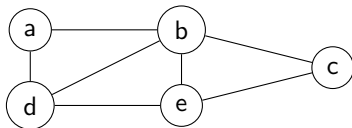
Voisins pires

Relaxation

Conclusion

Références

Coloration de somme minimale pour :



- La solution 12331 est-elle un optimum global ? sinon proposez le.

Exercice 4 : Optimum local ou global (réponse)

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

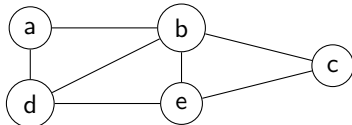
Voisins pires

Relaxation

Conclusion

Références

Coloration de somme minimale pour :



- La solution 12331 est-elle un optimum global ? sinon proposez le.
- **c'est un optimum local, l'optimum global est 13221**

Échapper aux optima locaux

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Relaxation

Conclusion

Références

Quatre famille d'approches :

- 1 Réitérer depuis différentes solutions initiales :
 - recherche locale avec redémarrage
 - recherche locale itérée
 - GRASP
 - ...
- 2 Accepter des voisins moins bons :
 - recherche tabou
 - recuit simulé
- 3 Changer le voisinage pendant la recherche
 - recherche à voisinage variable
- 4 Changer la fonction objectif ou les contraintes :
 - recherche locale guidée
 - stratégie de lissage (smoothing)
 - méthodes bruitées (noisy methods)

Échapper par redémarrage

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Redémarrage

GRASP

Voisins pires

Relaxation

Conclusion

Références

Recherche locale **avec redémarrages** : succession d'essais

```
Repeat /* faire un essai */
```

- choisir **nouvelle solution** initiale s
- nb_choisis=0

(#1)

```
Repeat
```

- choisir un voisin s' de s
- si acceptable_voisin(s, s') alors $s \leftarrow s'$
- nb_choisis ++

```
Until acceptable_solution(s) ou nb_choisis=MAX ou vois(s)=∅
```

```
Until acceptable_solution(s) ou time-out(recherche locale)
```

à l'étape (#1) : génération de solutions aléatoires

↔ ne conduira pas forcément au même optimum local.

time-out(recherche locale) : fonction du nombre d'essais.

Échapper par redémarrage

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Redémarrage

GRASP

Voisins pires

Relaxation

Conclusion

Références

Recherche locale **avec redémarrages** : succession d'essais

- **nb_essais=0**

Repeat /* faire un essai */

- choisir nouvelle solution initiale s

(#1)

- nb_choisis=0

Repeat

- choisir un voisin s' de s

- si acceptable_voisin(s, s') alors $s \leftarrow s'$

- nb_choisis ++

Until acceptable_solution(s) ou nb_choisis=MAX ou vois(s)= \emptyset

- **nb_essais ++**

Until acceptable_solution(s) ou **nb_essais=MAX2**

à l'étape (#1) : génération de solutions aléatoires

↔ ne conduira pas forcément au même optimum local.

time-out(recherche locale) : fonction du nombre d'essais.

Recherche locale itérée : GRASP

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Redémarrage

GRASP

Voisins pires

Relaxation

Conclusion

Références

Le GRASP (Greedy Randomized Adaptive Search Procedure) est basé sur un algorithme constructif puis une recherche locale.

- l'algorithme constructif doit pouvoir être randomisé pour générer différentes solutions initiales
- on lance ensuite une recherche locale depuis une solution initiale
- et on répète
- itérations indépendantes : pas de mémoire de recherche.

La lutte contre les optima locaux : mémoire

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Tabou

Recuit simulé

Relaxation

Conclusion

Références

Mémoire à court terme = **liste tabou**

- Pour sortir d'un optimum local : autoriser le passage par une solution moins optimale.
- le meilleur voisin remplace s même s'il est moins bon
- pb : on risque de retomber immédiatement après dans le même optimum local
- mémoriser à l'aide d'une liste tabou les k solutions récemment explorées, où ça n'est pas la peine de retourner.
- en pratique : mémoriser les k derniers mouvements (moins coûteux mais parfois moins précis, en réalité : stocker l'inverse du mouvement)
- détermination de la taille de la liste par expérimentation

Algorithme d'une recherche tabou

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Tabou

Recuit simulé

Relaxation

Conclusion

Références

```
•  $s \leftarrow s_0$                                 /* solution initiale*/  
• Tabou  $\leftarrow []$   
Repeat  
    • choisir meilleur voisin non Tabou  $s'$  de  $s$   
    • Tabou  $\leftarrow$  Tabou +  $\{s\}$   
    •  $s \leftarrow s'$   
Until condition d'arrêt
```

choisir_voisin(s) : meilleur voisin non Tabou

acceptable_voisin(s', s) : toujours

condition d'arrêt ■ soit limite en temps ou en itérations,
 ■ soit condition de non amélioration de la solution.

liste Tabou : implémentée en FIFO de taille k

Améliorations du Tabou

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Tabou

Recuit simulé

Relaxation

Conclusion

Références

- **Critères d'aspiration** : conditions permettant de lever le tabou
 - stockage des mouvements dans Tabou peut interdire des solutions intéressantes.
 - ↪ test à ajouter pour autoriser à considérer des solutions dans les voisins non tabous (ex : un mouvement qui génère une solution meilleure que tout ce qu'on a trouvé)
- Intensification : **mémoire à moyen-terme** qui stocke **l'élite** (meilleures solutions rencontrées durant la recherche)
 - ↪ biaiser la recherche pour favoriser les solutions ayant des attributs commun avec l'élite
- Diversification : **mémoire à long-terme** qui stocke les solutions visitées
 - ↪ explorer des zones non visitées.

Algorithme d'une recherche tabou amélioré

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Tabou

Recuit simulé

Relaxation

Conclusion

Références

Ajout des mémoires à moyen et long-terme et des critères d'aspiration.

```
•  $s \leftarrow s_0$                                 /* solution initiale*/  
• Tabou  $\leftarrow []$   
• Moyen-Terme  $\leftarrow []$   
• Long-Terme  $\leftarrow []$   
Repeat  
    • choisir_voisin_aspi(Tabou)  $s'$  de  $s$   
    • Tabou  $\leftarrow$  Tabou +  $\{s\}$   
    •  $s \leftarrow s'$   
    • mettre à jour Moyen-Terme et Long-Terme  
    • If critère_intensification Then intensification  
    • If critère_diversification Then diversification  
Until condition d'arrêt
```

choisir_voisin_aspi(s) : meilleur voisin non Tabou ou aspirable

Le recuit simulé

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Tabou

Recuit simulé

Relaxation

Conclusion

Références

- Méthode inspirée de la métallurgie : recuire un métal pour atteindre un état stable.
- Énergie E du système = fonction à minimiser
- Température T = paramètre contrôlant la diversification
 - ↪ T élevée/basse : instabilité/stabilité
 - ↪ on part de T élevée pour que chaque mouvement ait une probabilité d'acceptation de plus de 50% (mouvement très chaotique)
- on refroidit lentement par plateaux (réduction du bruit jusqu'à ne plus accepter de solutions plus mauvaises que s).

Le recuit simulé : Algorithme

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Tabou

Recuit simulé

Relaxation

Conclusion

Références

```
•  $s \leftarrow s_0$  solution initiale
•  $T \leftarrow T_{MAX}$  température de départ
Repeat
  Repeat /* à une température fixée*/
    •  $s' \leftarrow$  voisin de  $s$ 
    • calculer la variation d'énergie  $\Delta_E = f(s') - f(s)$ 
    • If  $\Delta_E < 0$  Then acceptable_voisin( $s', s$ )  $\leftarrow 1$ 
      Else acceptable_voisin( $s', s$ )  $\leftarrow e^{\frac{-\Delta_E}{T}}$ 
    • If acceptable_voisin( $s', s$ ) Then  $s \leftarrow s'$ 
  Until condition d'équilibre
  • diminuer température:  $T \leftarrow \lambda T$  avec  $\lambda < 1$ 
Until critère d'arrêt (ex:  $T < T_{min}$ )
```

$\text{acceptable_voisin}(s', s)$ = probabilité d'accepter s' depuis s

Illustration du recuit simulé

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

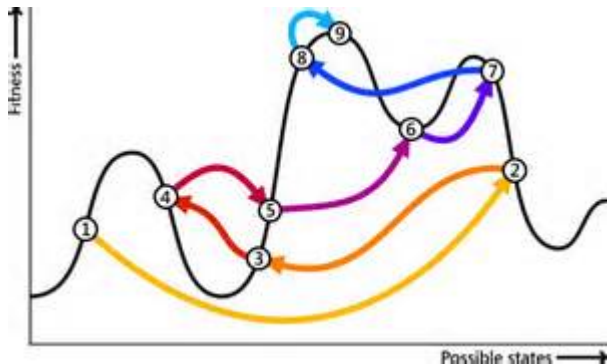
Tabou

Recuit simulé

Relaxation

Conclusion

Références



Réglage des paramètres

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Tabou

Recuit simulé

Relaxation

Conclusion

Références

- valeur de $T_{initiale}$: si la valeur T_{init} permet d'accepter $x\%$ des configurations de départ, alors on la garde sinon on la double et on recommence (x au moins plus de 50, certains auteurs : 80).
- la durée des paliers (longueur du plateau) : le plus simple est de le borner par le nombre de configurations atteignables en un mouvement élémentaire. là encore subtilités possibles
- la décroissance de la température : le plus simple est facteur constant, de l'ordre de 0.9 / 0.95 (décroissance lente). pour raffiner, le réglage se joue entre ce paramètre (+/- rapide) et le précédent (plateau +/- long).
- condition d'arrêt : la température mini ; soit 0 mais pas très efficace, soit quand améliorations très petites (très empirique).

Relaxation du problème : le lissage

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Relaxation

Le lissage

Bruit

Conclusion

Références

- transformer le problème en cachant des optima locaux
- trouver l'optimum s dans le paysage des solutions le plus lisse
- faire une recherche locale à partir de s dans le problème initial
- ou faire une recherche locale dans des variantes de moins en moins lissées du pb initial

Illustration du smoothing

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Relaxation

Le lissage

Bruit

Conclusion

Références

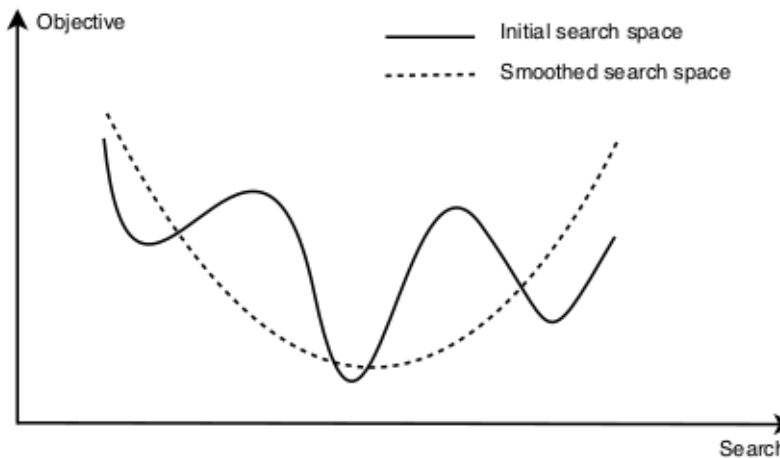


Illustration du smoothing (2)

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

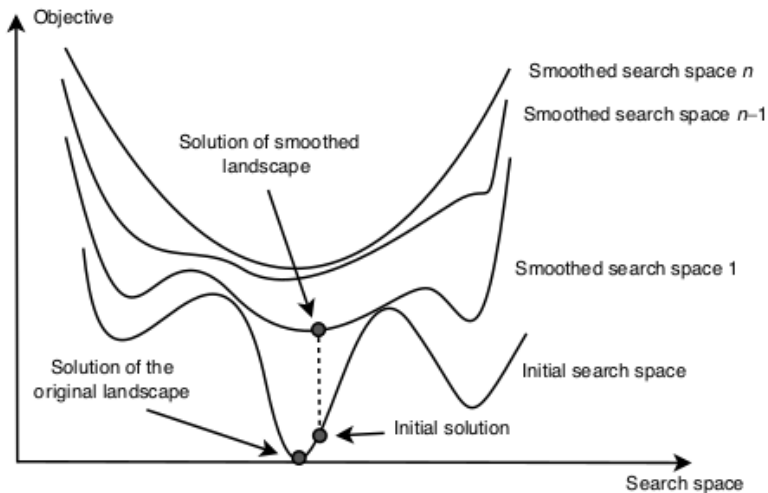
Relaxation

Le lissage

Bruit

Conclusion

Références



Méthodes bruitées

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Optimum
local/global

R. locale itérée

Voisins pires

Relaxation

Le lissage

Bruit

Conclusion

Références

Pour éviter d'être coincé dans un optimum local, on peut ajouter du **bruit** à la fonction objectif ou au choix du mouvement.

- le bruit est choisi aléatoirement dans l'intervalle $[-r, +r]$
- à chaque itération le bruit est réduit (l'intervalle est réduit)

Principe du bruit sur voisinage :

- avec une proba p , faire un mouvement aléatoire.
- avec une proba $1 - p$, suivre la méthode originale.

Reste le problème : comment régler p ?

Conclusion Chapitre 6 :

Les méthodes locales en question(s)

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Conclusion

Références

Comportement général :

- 1 une majorité de mouvements améliorent la solution courante.
- 2 le nombre d'améliorations diminue.
- 3 améliorations terminées : optimum, peut-être local.

Les questions à se poser :

- quand faut-il s'arrêter : faut-il être opportuniste ou gourmand ?
- comment ajuster les paramètres nombre d'essais/nombre de mouvements ?
- comment comparer les performances de deux méthodes différentes ? (qualité de la solution vs. temps consommé)

Avantages/Inconvénients des recherches locales

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Conclusion

Références

Avantages :

- Méthodes rapides (et temps paramétrable)
- Faciles à implémenter
- Donnent souvent de bonnes solutions
- Fonctionnement intuitif

Inconvénients :

- Manque de modélisation mathématiques (chaque cas est différent : il faut adapter la recherche à chaque problème particulier)
- Difficiles à paramétrer (= bricolage)
- Aucune évaluation de la distance à l'optimum (pas une approximation, trouve au pire un optimum local qui n'a rien à voir)
- **video** <https://www.youtube.com/watch?v=SC5CX8drAtU>

Conclusion Générale

Recherche locale

F. Bannay

Introduction

Algo R. Locale

Optima locaux

Conclusion

Références

- plein d'autres Structures de données intéressantes (kd-trees, dynamic trees...)
- les algorithmes de flot et de programmation linéaire sont toujours l'objet d'optimisations algorithmiques
- plein d'autres méthodes incomplètes (A^* incomplet, algorithmes génétiques, colonies de fourmi, essaim de particules, méthodes hybrides globales/locales...)
- des outils :
 - Structure de données :
<https://people.ksp.sk/~kuko/gnarley-trees>
 - Flots (MatLab, Pseudoflow solver (Chandran and Hochbaum 2007 <http://riot.ieor.berkeley.edu/Applications/Pseudoflow/maxflow.html>)...
 - Programmation Linéaire :
<http://www.zweigmedia.com/RealWorld/simplex.html>
 - Meta-heuristiques : outils non commerciaux (ParadisEO, ECj, oMetah) et commerciaux (CPLEX, Mathematica)



Ahuja, R., Magnanti, T., Orlin, J., and Reddy, M. (1995).
Applications of Network Optimization, volume 7, chapter 1.
Elsevier Science B.V.



Alj, A. and Faure, R. (1990).
Guide de la recherche opérationnelle : Les applications.
Masson.



Busacker, R. and Gowen, P. (1961).
A procedure for determining minimal-cost network flow patterns.
Technical Report ORO-15, Operational Research Office, John Hopkins
University.



Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2009).
Introduction to Algorithms.

MIT Press.



Dinic, E. A. (1970).

Algorithm for solution of a problem of maximum flow in a network with power estimation.

Soviet Math. Doklady, 11 :1277–1280.



Edmonds, J. and Karp, R. M. (1972).

Theoretical improvements in algorithmic efficiency for network flow problems.

Journal of the ACM, 19(2) :248–264.

doi :10.1145/321694.321699.



Faure, R., Lemaire, B., and Picouleau, C. (2000).

Précis de Recherche Opérationnelle.

Dunod, Paris.



Ford, L. and Fulkerson, D. (1955).

A simplex algorithm finding maximal networks flows and an application to the hitchcock problem.

Rand Report Rand Corporation.



Fournier, J. (2011).

Théorie des graphes et applications : Avec exercices et problèmes.
Collection Informatique. Hermes Science Publications.



Gondran, M. and Minoux, M. (2009).

Graphes et Algorithmes (4e ed.).
Lavoisier.



Hochbaum, D. S. (2004).

Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today.
Management Science, 50(6) :709–723.



Klein, M. (1967).

A primal method for minimal cost flows with applications to the assignment and transportation problems.
Management Science, 14 :205–220.



Kleinberg, J. and Tardos, E. (2005).

Algorithm Design.

Addison Wesley.



Maurin, H. (1967).

Programmation linéaire appliquée.

Éditions Technip.

374 pages.



Orlin, J. B. (2013).

Max flows in $o(nm)$ time, or better.

In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, STOC '13*, pages 765–774, New York, NY, USA. ACM.



Papadimitriou, C. and Steiglitz, K. (2013).

Combinatorial Optimization.

Dover Publications Inc.



Roy, B. (1969).

Algèbre moderne et théorie des graphes.

Dunod, Paris.



Talbi, E.-G. (2009).

Metaheuristics : from design to implementation.

Wiley.



Wikipédia (2007).

Recherche opérationnelle.

Wikipédia, l'encyclopédie libre, page

http://fr.wikipedia.org/w/index.php?title=Recherche_op%C3%A9rationnelle&oldid=17837831.

[En ligne ; Page disponible le 14-août-2007].