Recherche opérationnelle

DUT Info 2e année, parcours A

Méta-heuristiques

Florent Foucaud

IUT CLERMONT AUVERGNE

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

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 (vu précédemment)

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 (vu précédemment)

Définition (Méta-heuristique)

Algorithme heuristique générique, défini pour 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 (vu précédemment)

Définition (Méta-heuristique)

Algorithme heuristique générique, défini pour un grand nombre de problèmes.

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

- Méta-heuristiques de recherche locale : gradient, recuit simulé, tabou...
- Méta-heuristiques à population : algos génétiques, colonies de fourmis, essaims de particules...

Méta-heuristiques de recherche locale

Principe 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

Principe 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

Principe 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 meilleure 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 meilleure 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 meilleure 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 meilleure 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 meilleure 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).

Une variante probabiliste : le recuit simulé (1983)

Objectif : diversification pour éviter de tomber dans un mauvais optimum local

On définit une température qui décroit lentement au cours de l'algorithme.

La probabilité de diversification dépend de la température.

→ Au bout d'un certain temps, on arrête de diversifier.

Le recuit, une technique métallurgique

Scott Kirkpatrick

C. Daniel Gelatt, Jr

Mario P. Vecchi

Une variante probabiliste : le recuit simulé (1983)

Objectif : diversification pour éviter de tomber dans un mauvais optimum local

On définit une température qui décroit lentement au cours de l'algorithme.

La probabilité de diversification dépend de la température.
→ Au bout d'un certain temps, on arrête de diversifier.

Algorithme de recherche locale, version "recuit simulé" :

1. On génère une première solution S_0
   $S \leftarrow S_0$ #solution courante
   $i = 0$ #compteur du nombre d'itérations
   $T = 100$ #température initiale
   $k = 1000$ #nombre d'étapes
2. Tant que $i < k$:
   - $T = 0.99T$ #la température baisse
   - $i = i + 1$
   - choisir au hasard une nouvelle solution S' dans $N(S)$
   - si S' est meilleure que S OU $random(0, 1) < exp(-(|S| - |S'|)/T)$
     - $s \leftarrow S'$
3. Retourner S

Le recuit, une technique métallurgique

Scott Kirkpatrick

C. Daniel Gelatt, Jr

Mario P. Vecchi