Programmation en nombre entiers

Rafaël Lopez

LRI

Plan

- PL et PNE : différences au point de vue algorithmique
- Classification des algorithmes
- Le piège des optima locaux

Programme linéaire et programme en nombres entiers

- Pour les programmes linéaires continus, il existe des algorithmes généraux (ne dépendent pas du problème) et efficaces (capables de résoudre des problèmes de grande taille). Les exemples sont : le simplexe, et les méthodes de points intérieurs.
- Pour les programmes en nombre entiers, il n'existe pas d'algorithme à la fois général et efficace. De plus, la théorie suggère qu'il n'en existera jamais.

Résoudre un problème en nombres entiers

- Il existe de manières de classer les algorithmes de résolution pour les PNE
  - A caractère général : peut résoudre n'importe quel PNE, mais est potentiellement inefficace.
  - A caractère spécifique : étudié pour un type de problème particulier, mais potentiellement efficace.
- Le deuxième classement possible :
  - optimal : l'algorithme est certain de trouver la solution optimale.
  - heuristique : l'algorithme trouve une bonne solution, que l'on espère proche (ou égale) à l'optimum.
- Il en découle donc quatre “catégories” à considérer.

Algorithmes généraux à solution optimale

- Énumération exhaustive
- Branch and bound
- Difficulté : le nombre de solutions à énumérer devient rapidement trop grand.
  - Pour 4 variables binaires: $2^4 = 16$ solutions possibles
  - Pour 100 variables binaires : $2^{100} = 10^{30}$ solutions possibles!
- Les techniques de B&B permettent de limiter le nombre de solutions à explorer, mais le temps de calcul reste conséquent.

Algorithmes généraux à heuristique

- En pratique, cela consiste à terminer un des algorithmes précédents de manière anticipée (après un temps ou un nombre d'itérations fixé à l'avance).

Algorithmes particuliers à solution exacte

- Montée duale (dual ascent)
- Relaxation Lagrangienne et :
  - optimisation de sous-gradient ; or
  - Ajustement de multiplicateur
- Reposent sur les principes des méthodes précédentes (type B&B)
- Utilisent souvent la PL via une relaxation linéaire
- Utilisent souvent la structure des contraintes du problème à résoudre
- Ces algorithmes sont efficace jusqu'à certaines tailles, après quoi ils deviennent inefficaces (l'effort à fournir pour trouver une solution optimale devient exponentielle avec la taille du problème).

Algorithmes particuliers à heuristique

- en ce qui concerne ces méthodes, il existe un certain nombre d'approches génériques (dans le sens où l'approche est générique, mais l'implémentation doit être adaptée au problème considéré).
  - glouton
  - échanges
  - heuristiques basées sur les bornes(ex : heuristiques lagrangiennes)
  - recherche tabou
  - recuit simulé
  - heuristiques à base de populations (ex algorithmes génétiques)

Pourquoi faire des heuristiques ?

- Il se peut que l'on ne soit pas intéressé par la solution optimale:
  - taille trop importante
  - temps nécessaire pour trouver l'optimum ne justifiant pas le coût en temps/ressources
- Les heuristiques sont en général plus « simples », et moins gourmandes en temps/ressources qu'un algorithme exact.
- Exemple : assignation de tâches.

Optimum local et optimum global

- Cas d'algorithmes gloutons
- Illustration sur une fonction quelconque à une variable
- D'où nécessité de sortir des optima locaux.
- Exemples :
  - Recherche tabou
  - Recuit simulé