Métaeuristique

Nilson Toula
Emmanuel Rochet

Contents

1. Introduction

2. Formulation du SPP

Selon Xavier Delorme, Xavier Gandibleux et Joaquin Rodriguez dans leur article (European Journal of Operational Research 153 (2004) 564–580):

”

Le problème de Set Packing est défini par le modèle mathématique suivant : Max sommei=1,. . . ,n c(i) x(i) sc sommei=1,. . . ,n t(i,j) x(i) ¡= 1, pour j=1,. . . ,m x(i)=0 ou 1

” Il s’agit d’une modélisation mathématique d’un problème qu’ils ont rencontré vis à vis de la gestion d’une jonction de voie ferré. Il fallait faire passer un maximum de trains sur une unique jonction sachant que les types de trains, leur longueur (donc le temps d’utilisation de la jonction) pouvaient varier (Transport de fret, Transport de passager, etc..) . Ainsi, chaque contrainte pourrait représenter un panel de train pouvant emprunter la jonction à une plage horaire donné, seul un train peut passer celle-ci à la fois. Il faudrait ainsi maximiser le nombre de trains passant sur la jonction lors d’une journée, tout en évitant les collisions et en respectant les distances réglementaire.

2.1. présentation SPP

3. Modélisation JuMP

4. Heuristique de Construction

L’heuristique de construction prend en variable:

- Le vecteur des coûts de chaque variable x de taille n - La matrice des contraintes de taille m*n - Le nombres de variable n - Le nombre de contraintes m

Et retourne: - Une solution x0 acceptable sous forme d’un tableau de taille n

Choix de cette signature: Nous pourrions nous contenter de donner en paramètre uniquement la matrice des contraintes ainsi que le tableau des coûts pour ensuite les récupérer via la taille de la matrice des contraintes, mais vu que ces variables(celles de la taille de la matrice) sont globales, nous avons supposer (peut être à tord) que l’on gagnait du temps si l’on ne recherchait pas la taille de la matrice des contraintes.

Déroulement de cette heuristique: Nous initialisons différents tableau qui symboliseront les lignes ainsi que les variables que nous allons étudier (sans avoir à interagir directement avec la matrice des contraintes)

Ensuite, nous entrons dans une boucle qui s’arrête une fois qu’on à traiter toutes les lignes de ou toutes les variables de notre problèmes (voir plus loin):

Dans la fonction Utilite: Nous calculons le nombre d’itération de chaque variable encore active dans chaque ligne encore active. Ensuite nous en déduisons utilité moyenne en divisant son coût par son nombre d’itération. Nous retournons un tableau comportant ces prix moyens.

Dans la fonction PosMax Nous sélectionnons ensuite dans ce tableau la position du premier candidat ayant le plus haut coût moyen. (ie: on récupère le i du premier meilleur candidat, sachant que les variables sont de la forme xi (avec i allant de 1 à n)) On retourne cette position.

Nous actualisons le tableau des solutions en passant à 1 la position du candidat choisit.

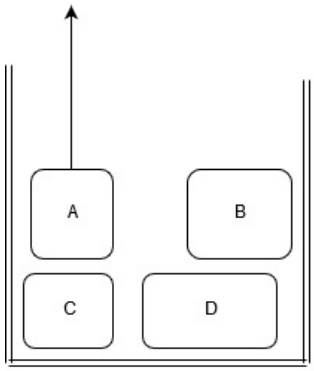
Dans la fonction Desactive! Pour un candidat choisit: Nous indiquons qu’il n’est plus utile d’étudier la ligne et les colonnes où apparaissent ce candidat. (ie: Nous mettons à jours les tableaux listant les colonnes étudiés et les lignes étudiées et passant respectivement à 1 sa position dans la liste des lignes actives: ”actif” et à 0 sa position dans la liste des variables étudié: ”varactive”)

Une fois cette boucle finie, nous indiquons dans la console la valeur de la solution trouvé

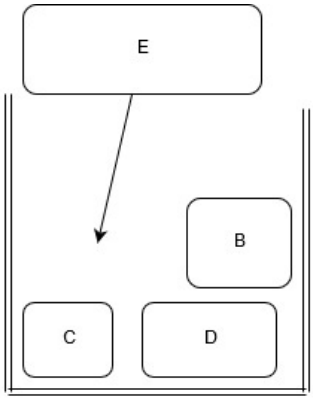
Enfin nous renvoyons le tableau de solution, la liste des lignes qui ont été désactivé ainsi que la valeur de la fonction objectif.

5. Plus profonde descente

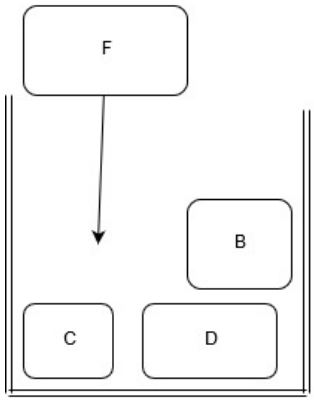
L'heuristique de plus profonde descente utilisée est une heuristique dite de "kp-exchange Cette heuristique de voisinage consiste à échanger une partie de la solution avec une autre solution proche en "échangeant" k variables contre p. Dans le problème de SPP, cela se traduit par la "sortie" du sac d'une ou plusieurs variables (k) contre la "remise" d'autres variables (p) dans le sac.
A chaque itération, on regarde la solution. Si elle est meilleur que la précédente, on la garde.



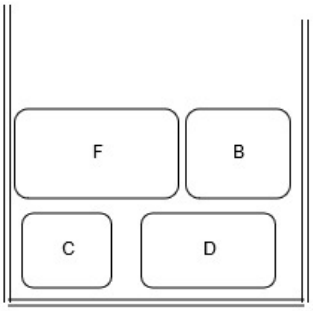
1) On retire A



2) On essaie de rentrer E mais E est incompatible donc on passe à la variable suivante



3) F peu rentrer, alors on le met



4) F remplit mieux que A, on garde alors cette solution

5.1. Implémentations

5.2. Optimisation

6. Résultats

7. Conclusion