

Projet Métaheuristique : Application de la VNS au problème de sac-à-dos multidimensionnel

Roxane DELEPEYRAT, Le Thanh Dung NGUYEN

22 octobre 2015

Sommaire

1	Description du problème de sac-à-dos multidimensionnel	2
2	Description de la recherche à voisinages variables (VNS)	2
3	Algorithme glouton pour générer une solution initiale	3
4	Description des différents voisinages utilisés	3
5	Description de différentes recherches locales utilisées	3
6	Description des différents tests effectués	3

Introduction

Nous avons travaillé sur le problème du sac-à-dos multidimensionnel en appliquant un métaheuristique de recherche à voisinages variables (VNS).

1 Description du problème de sac-à-dos multidimensionnel

Ce problème modélise un problème d'allocation de ressources. On se donne :

- N projets et M ressources
- b_i la quantité de ressource i disponible ($i = 1, \dots, M$)
- c_j le profit associé au projet j ($j = 1, \dots, N$)
- $a_{i,j}$, la quantité de ressource i nécessaire à la réalisation du projet j ($i = 1, \dots, M$ et $j = 1, \dots, N$)

Le problème consiste à sélectionner un sous-ensemble des projets maximisant le gain total dans la limite des ressources disponibles. Il peut être modélisé de cette manière :

$$\min f(x) = \sum_{j=1}^N c_j x_j \quad (1)$$

$$s.c. \sum_{j=1}^N a_{i,j} x_j \leq b_i \quad i = 1, \dots, M \quad (2)$$

$$x_j \in 0, 1 \quad (3)$$

$$(4)$$

avec $x_j = 1$ si le projet est accepté. On suppose ici que :

- c_j et b_i sont des entiers positifs
- $A^j = (a_{i,j})_{i=1,\dots,M}$ est différent du vecteur nul $\forall j \in 1, \dots, N$.

2 Description de la recherche à voisinages variables (VNS)

Le principe de la VNS est dépasser les optimaux locaux en changeant de voisinages. L'algorithme part d'une solution réalisable, crée une solution perturbée dans le voisinage de cette dernière puis effectue une recherche locale dans le voisinage de la solution perturbée. Si la solution trouvée par la recherche locale est meilleure que celles trouvées jusqu'à présent, alors l'algorithme change de voisinage et recommence. Si une solution est optimale pour tous les voisinages explorés par l'heuristique ou si le temps alloué est dépassé alors l'algorithme s'arrête et renvoie la meilleure solution réalisable rencontrée. Voici le pseudo-code décrivant la VNS :

```
{début}
tps ← 0
x ← x0
while tps ≤ tpsmax do
  k ← 1
  while k ≤ kmax do
    {Étape de perturbation}
    Générer une solution x' aléatoirement dans le voisinage Vk(x)
    {Étape de recherche locale}
    Appliquer une heuristique de recherche locale à partir de x'
    Soit x'' la solution trouvée par la recherche locale.
    {Étape de déplacement}
    if f(x'') > f(x) then
      x ← x''
      k ← 1
    else
      k ← k + 1
    end if
  end while
  tps ← Cputtime()
end while
Retourner x
{Fin}
```

Pour appliquer la VNS, il faut donc décider :

- quels voisinages vont être utilisés pour générer une solution perturbée.
- quelle méthode de recherche locale utiliser pour chaque voisinage exploré
- quelle solution prendre comme solution de départ et comment la générer.

3 Algorithme glouton pour générer une solution initiale

Pour générer une solution initiale, l'algorithme glouton s'inspire de celui pour le problème du sac-à-dos unidimensionnel. Les projets sont d'abord classés par "prix", puis le sac-à-dos est rempli en commençant par les projets de plus grandes valeurs. Ici, le prix d'un projet est calculé de la manière suivante : $prix_j = c_j / \sum_{i=1, \dots, M} a_{i,j}$. Voici le pseudo-code de l'algorithme glouton utilisé :

Soit $prix$ le vecteur des prix

La solution initiale n'accepte aucun projet

Soit $reste$ un vecteur de taille M tel que $reste[i]$ soit égal à la quantité de ressource i restante après avoir accepter les projets de x

{Classement des projets par valeur}

for $j = 1, \dots, N$ **do**

$$prix[j] = \frac{c_j}{\sum_{i=1, \dots, M} a_{i,j}}$$

end for

{Remplissage du sac-à-dos}

for chaque projet plus prix le plus élevé au prix le plus faible **do**

if le projet peut être ajouté à la solution en respectant les contraintes **then**

 Ajouter le projet à la solution

end if

end for

Retourner la solution

4 Description des différents voisinages utilisés

5 Description de différentes recherches locales utilisées

6 Description des différents tests effectués