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

Roxane Delepeyrat, Le Thanh Dung Nguyen

1er novembre 2015

Sommaire

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

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 :

$$minf(x) = \sum_{j=1}^{N} c_j xj \tag{1}$$

$$s.c. \sum_{j=1}^{N} a_{i,j} x_j \leq bi \quad i = 1, \ldots, M \tag{2}$$

$$x_j \in 0, 1 \tag{3}$$

$$\tag{4}$$

avec $x_j = 1$ si le projet est accepté. On suppose ici que :
- c_j et b_i sont des entiers positifs
- $A^j = (a_{i,j})_{i=1,\ldots,M}$ est différent du vecteur nul $\forall j \in 1, \ldots, N$.

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 ← x_0
while tps ≤ tpsmax do
  k ← 1
  while k ≤ k_max do
    {Étape de perturbation}
    Générer une solution x' aléatoirement dans le voisinage V_k(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 "valeur", puis le sac-à-dos est rempli en commençant par les projets de plus grandes valeurs. Ici, la valeur d'un projet est calculé de la manière suivante : $valeur_j = c_j / sum_{i=1,\cdots,M} a_{i,j}$. Voici le pseudo-code de l'algorithme glouton utilisé :

Soit valeur le vecteur des valeurs
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 $j = 1, \cdot, N$ **do**
  $valeur[j] = \frac{c_j}{\sum_{i=1,\cdots,M} a_{i,j}}$
end for
{Remplissage du sac-à-dos}
for chaque projet de la valeur la plus éléveé à la valeur la 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

L'heuristique utilise différents voisinages pour se déplacer dans l'univers de solutions. Les deux premiers voisinages choisis utilisent la distance de Hamming. Si une solution du problème est modélisée par un vecteur binaire où la i-ème coordonnée vaut 1 si le projet est retenu et 0 sinon, alors, si x est une solution du problème,le voisinage de distance 1 $V_1(x)$ est l'ensemble des vecteurs obtenus en complémentant un bit du vecteur x et le voisinage de distance 2 est l'ensemble des vecteurs obtenus en complémentant deux bits du vecteur x. Les deux voisinages sont donc inclus l'un dans l'autre et peuvent contenir des solutions irréalisables. Chaines d'échanges ?

5 Description de différentes recherches locales utilisées

L'heuristique utilise la distance de Hamming de 1 et de 2. Il a donc été logique d'implémenter comme algorithme de recherche locale des algorithmes de montée dans les voisinages descrits plus haut. Voici le pseudo-code de l'algorithme de montée :

Soit x une solution du problème admissible ou non. Si x n'est pas admissible alors sa valeur $f(x)$ vaut 0.
$x_{max} \leftarrow x$
for chaque bit i de la solution **do**
  Générer x' en complémentant le i-ème bit de x.
  Evaluer si x' est admissible ou non
  if x' est admissible **then**
    Evaluer la valeur de x', $f(x')$
    if $f(x') > f(x)$ **then**
      $x_{max} \leftarrow x'$
    end if
  end if
end for

6 Description des différents tests effectués

mama