Variantes de la Recherche à Voisinage Variable avec Décomposition pour le Sac-à-Dos Multidimensionnel

Saïd Hanafi[1], Jasmina Lazić[2], Nenad Mladenović[2], Christophe Wilbaut[1] et Igor Crévits[1]

[1] LAMIH, Université de Valenciennes, F 59313 Valenciennes Cedex9, France
{said.hanafi,christophe.wilbaut,igor.crevits}@univ-valenciennes.fr
[2] Brunel University, West London UB8 3PH, UK
{Jasmina.Lazic,Nenad.Mladenovic}@brunel.ac.uk

Mots-Clés : *Méthode Hybride, Sac à dos, Variable Neighbourhood Decomposition Search.*

Nous proposons une nouvelle approche hybride pour résoudre le problème du sac à dos multidimensionnel en variables 0–1 (MKP pour *Multidimensional Knapsack Problem*) dans laquelle nous améliorons dynamiquement les bornes inférieure et supérieure sur la valeur optimale du problème en utilisant la recherche à voisinage variable avec décomposition (VNDS pour *Variable Neighbourhood Decomposition Search*). Le MKP est un problème d'allocation de ressources très connu qui peut être formulé par :

$$(\text{MKP})\ \max\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \forall i \in M = \{1, 2, \ldots, m\}, x_j \in \{0, 1\}, \forall j \in N = \{1, 2, \ldots, n\}\}$$

avec n le nombre d'objets et m le nombre de contraintes. Toutes les données a_{ij}, b_i, c_j sont supposées entières et positives. Le lecteur peut se référer à [4] pour une revue de la littérature sur le MKP.
Avant de décrire notre approche, nous introduisons la notion de problème réduit qui est obtenu à partir du problème original en fixant un sous-ensemble de variables à une valeur donnée. Soit $x^0 \in \{0, 1\}^n$ un vecteur binaire et $J \subseteq N$ un sous-ensemble quelconque de variables, le problème réduit associé à x^0 et J est défini par :

$$\text{MKP}(x^0, J)\ \max\{c^T x : Ax \leq b, x_j \in \{0, 1\}, \forall j \in N, x_j = x_j^0, \forall j \in J\}$$

Nous définissons également la distance partielle entre deux solutions x et y sur un sous-ensemble $J \subseteq N$ par :

$$\delta(J, x, y) = \sum_{j \in J} \mid x_j - y_j \mid.$$

VNDS est un schéma à deux niveaux basé sur VNS (*Variable Neighbourhood Search*) et sur la décomposition du problème [2]. L'application de VNDS au MKP exploite l'idée mentionnée entre autre dans [3] qui stipule que fixer les variables ayant la même valeur dans la solution courante et dans la solution de la relaxation en continu peut conduire à obtenir une solution réalisable de bonne qualité. Cependant notre procédure est plus flexible, permettant la fixation d'autres variables ou au contraire la libération de certaines variables, en suivant le principe général de VNDS. De plus nous incorporons un processus visant à améliorer dynamiquement les bornes sur la valeur optimale du problème à l'aide de contraintes supplémentaires : une sur l'objectif, $c^{\mathrm{T}} x \geq c^{\mathrm{T}} x^* + \varepsilon$ qui est utilisée dans le sous-problème courant (x^* étant la meilleure solution courante et ε étant fixé à 1 puisque les données sont entières), et une autre visant à éliminer l'ensemble des solutions partielles (i.e., dans $\{0, 1, *\}^n$) déjà visitées. Nous proposons plusieurs variantes de VNDS utilisant différentes stratégies d'exploration de l'espace de recherche. Les étapes principales d'une version de VNDS pour le MKP sont données dans la Figure 1.

```
VNDS-MKP(MKP, d, t_max, t_sub, t_vnd, k_max)
 1  Soit x une solution réalisable de MKP.
 2  x* = x, t_start = cpuTime( ), t = 0, Q = MKP.
 3  tant que (t < t_max)
 4      Soit x̄ une solution optimale de LP(Q).
 5      Calculer δ_j =| x_j − x̄_j | pour j ∈ N, et ordonner
        les variables x_j, j ∈ N de sorte que δ_1 ≤ δ_2 ≤ ... ≤ δ_n.
 6      n_d =| {j ∈ N | δ_j ≠ 0} |, k_step = [n_d/d], k = n − k_step ;
 7      tant que (t < t_max) et (k > 0)
 8        x' = MIPSOLVE(Q(x, {1, 2, ..., k}), t_sub, x) ;
 9        Enrichir Q avec une pseudo-coupe : Q = (Q | δ(N, x*, x) > 1) ;
10        si (c^T x' > c^T x) alors
11            x = VND-MKP(Q, t_vnd, k_max, x') ;
12        x* = x ;
13        Ajouter à Q une coupe sur l'objectif :
              Q = (Q | c^T x ≥ c^T x* + 1) ;
14        break ;
15      sinon
16        si (k − k_step > n − n_d) alors
              k_step = max{[k/2], 1} ;
17        k = k − k_step ;
18        t_end = cpuTime( ), t = t_end − t_start ;
19      fin si
20    fin tq
21  fin tq
22  retourner x*.
```

FIGURE 1 – une implémentation de VNDS pour le MKP.

Les paramètres en entrée de l'algorithme sont : MKP, une instance du problème ; d, le nombre de variables libres à chaque itération ; t_{max} le temps total autorisé ; t_{sub} le temps de résolution d'un sous-problème ; t_{vnd} le temps pour une recherche locale (type VND), et k_{max} le nombre maximal de voisinages à explorer. A chaque itération nous résolvons la relaxation en continu du problème courant LP(Q), puis nous réordonnons les variables selon la distance $\delta_j =\mid x_j - \overline{x}_j \mid$ entre la solution de la relaxation $(\overline{x}_j)_{j\in N}$ et la solution courante $(x_j)_{j\in N}$. Ensuite nous résolvons la séquence de sous-problèmes $Q(x, \{1, \ldots, k\})$ pour différentes valeurs de k jusqu'à améliorer la solution courante. L'ajout de la pseudo-coupe $\delta(N, x^*, x) > 1$ permet d'éviter le retour à une solution préalablement visitée. Lorsqu'une amélioration se produit, une recherche locale est activée sur l'espace de recherche initial, et la coupe sur l'objectif est ajoutée. Notons que des stratégies de décomposition similaires récentes peuvent être recensées dans la littérature pour résoudre des problèmes en variables mixtes [1, 5].

Nous avons validé notre approche sur deux ensembles d'instances de la littérature disponibles sur Internet [1] [2]. Le premier est composé de 90 instances avec $n = 500$ et $m = 5, 10, 30$. Le second est composé de 18 instances avec n allant de 100 à 2500 et m de 15 à 100. Les résultats, qui seront donnés en détails lors de la conférence, montrent un bon comportement global des différentes variantes qui convergent rapidement vers des solutions très proches des meilleures solutions connues. En particulier, nos heuristiques obtiennent 53 meilleures bornes inférieures du premier groupe en moins d'une heure (pour un gap moyen inférieur à 0.01%), et 14 meilleures bornes inférieures du second groupe (pour un gap moyen inférieur à 0.001%). Ces résultats encourageants nous poussent à explorer de nouvelles pistes pour améliorer le comportement de notre approche pour le MKP en essayant d'intégrer d'autres éléments tels que la mémoire adaptative pour mieux guider le processus de recherche.

Références

[1] F. Glover. Adaptive memory projection methods for integer programming. Metaheuristic Optimization Via Memory and Evolution : Tabu Search and Scatter Search. Kluwer Academic Publishers, 2005.

[2] P. Hansen, N. Mladenović et D. Perez-Britos. Variable Neighborhood Decomposition Search. *Journal of Heuristics*, 7(4):335–350, 2001.

[3] J. Lazić, S. Hanafi, N. Mladenović et D. Urosević. Variable Neighbourhood Decomposition Search for 0–1 Mixed Integer Programs Accepté pour publication dans *Computers & Operations Research*, 2009.

[4] H. Kellerer, U. Pferschy et D. Pisinger. Knapsack Problems. Springer, 2004.

[5] C. Wilbaut et S. Hanafi. New convergent heuristics for 0–1 mixed integer programming. *European Journal of Operational Research*, 195:62–74, 2009.

1. http ://people.brunel.ac.uk/∼mastjjb/jeb/info.html
2. http ://hces.bus.olemiss.edu/tools.html