Le problème du sac à dos

Présentation du problème (d'après http://interstices.info/jcms/c_19213/le-probleme-du-sac-a-dos)

L'énoncé de ce problème fameux est simple : *« Étant donné plusieurs objets possédant chacun un poids et une valeur et étant donné un poids maximum pour le sac, quels objets faut-il mettre dans le sac de manière à maximiser la valeur totale sans dépasser le poids maximal autorisé pour le sac ? »*

Toute formulation de ce problème commence par un énoncé des données. Dans notre cas, nous avons un sac à dos de poids maximal P et n objets. Pour chaque objet i, nous avons un poids p_i et une valeur v_i.

Pour quatre objets ($n = 4$) et un sac à dos d'un poids maximal de 30 kg ($P = 30$), nous avons par exemple les données suivantes :

Objets	1	2	3	4
v_i	7	4	3	3
p_i	13	12	8	10

Ensuite, il nous faut définir les variables qui représentent en quelque sorte les actions ou les décisions qui amèneront à trouver une solution. On définit la variable x_i associée à un objet i de la façon suivante : $x_i = 1$ si l'objet i est mis dans le sac, et $x_i = 0$ si l'objet i n'est pas mis dans le sac.

Dans notre exemple, une solution réalisable est de mettre tous les objets dans le sac à dos sauf le premier, nous avons donc : $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, et $x_4 = 1$.

Puis il faut définir les contraintes du problème. Ici, il n'y en a qu'une : la somme des poids de tous les objets dans le sac doit être inférieure ou égale au poids maximal du sac à dos.
Cela s'écrit ici $x_1.p_1 + x_2.p_2 + x_3.p_3 + x_4.p_4 \leq P$ et pour n objets :

$$\sum_{i=1}^{i=n} x_i.p_i \leq P$$

Pour vérifier que la contrainte est respectée dans notre exemple, il suffit de calculer cette somme : $0 \times 13 + 1 \times 12 + 1 \times 8 + 1 \times 10 = 30$, ce qui est bien inférieur ou égal à 30, donc la contrainte est respectée. Nous parlons alors de solution réalisable. Mais ce n'est pas nécessairement la meilleure solution.

Enfin, il faut exprimer la fonction qui traduit notre objectif : maximiser la valeur totale des objets dans le sac. Pour n objets, cela s'écrit :

$$\max \sum_{i=1}^{i=n} x_i.v_i$$

Dans notre exemple, la valeur totale contenue dans le sac est égale à 10. Cette solution n'est pas la meilleure, car il existe une autre solution de valeur plus grande que 10 : il faut prendre seulement les objets 1 et 2 qui donneront une valeur totale de 11. Il n'existe pas de meilleure solution que cette dernière, nous dirons alors que cette solution est optimale.

Méthodes de résolution
Il existe deux grandes catégories de méthodes de résolution de problèmes d'optimisation combinatoire : les méthodes exactes et les méthodes approchées. Les méthodes exactes permettent d'obtenir la solution optimale à chaque fois, mais le temps de calcul peut être long si le problème est compliqué à résoudre. Les méthodes approchées, encore appelées heuristiques, permettent d'obtenir rapidement une solution approchée, donc pas nécessairement optimale. Nous allons détailler un exemple d'algorithme de résolution de chaque catégorie.

Méthode approchée
Une méthode approchée a pour but de trouver une solution avec un bon compromis entre la qualité de la solution et le temps de calcul. Pour le problème du sac à dos, voici un exemple d'algorithme de ce type :

- calculer le rapport (v_i / p_i) pour chaque objet i ;
- trier tous les objets par ordre décroissant de cette valeur ;
- sélectionner les objets un à un dans l'ordre du tri et ajouter l'objet sélectionné dans le sac si le poids maximal reste respecté.

Déroulons cet algorithme sur notre exemple :

Première étape :

Objets	1	2	3	4
v_i / p_i	0,54	0,33	0,37	0,30

Deuxième étape :

Objets	1	3	2	4
v_i	7	3	4	3
p_i	13	8	12	10
v_i / p_i	0,54	0,37	0,33	0,30

Troisième étape :

Objet 1 : on le met dans le sac vide, le poids du sac est alors de 13 et inférieur à 30 ;
Objet 3 : on le met dans le sac car 13 + 8 = 21 est inférieur à 30 ;
Objet 2 : on ne le met pas dans le sac car le poids total (33) dépasserait 30.
Objet 4 : on ne le met pas dans le sac (poids total de 31).

La solution trouvée est donc de mettre les objets 1 et 3 dans le sac, ce qui donne une valeur de 10. Cette solution n'est pas optimale, puisqu'une solution avec une valeur totale de 11 existe. Néanmoins, cet algorithme reste rapide même si le nombre d'objets augmente considérablement.

Ce type d'algorithme est aussi appelé *algorithme glouton*, car il ne remet jamais en cause une décision prise auparavant. Ici, lorsque l'objet 2 ne peut pas entrer dans le sac, l'algorithme n'essaie pas d'enlever l'objet 3 du sac pour y mettre l'objet 2 à sa place.