Université de Versailles Saint-Quentin-en-Yvelines
Master 1 : Calcul Haute Performance, Simulation

Nom : Lougani
Prénom : Faouzi
Numéro étudiant : 22003152
Objet : Projet d'Optimisation et recherche opérationnelle

Rapport : Problème du sac à dos

Introduction :

Le problème du sac à dos est un problème modélisant une situation analogue au remplissage d'un sac à dos avec des objets dont chacun possède un poids noté **P** et une valeur notée **V** en maximisant la valeur totale, sans dépasser le poids maximum du sac.

Résolution du problème du sac à dos :

Le problème de sac à dos est un problème d'optimisation combinatoire.Il y a deux types de méthodes de résolution pour ces problèmes connues sous le nom de méthodes exactes et méthodes approchées, pour notre projet on se basera sur le premier type (méthode exactes) en choisissant la méthode de séparation et évaluation (branch and bound en anglais) afin de résoudre le problème du sac à dos.

Branch and bound :

Afin d'implémenter cet algorithme,j'ai commencé par dérouler à la main la méthode de séparation et évaluation sur notre problème du sac à dos pour un cas simple.

On commence par définir :

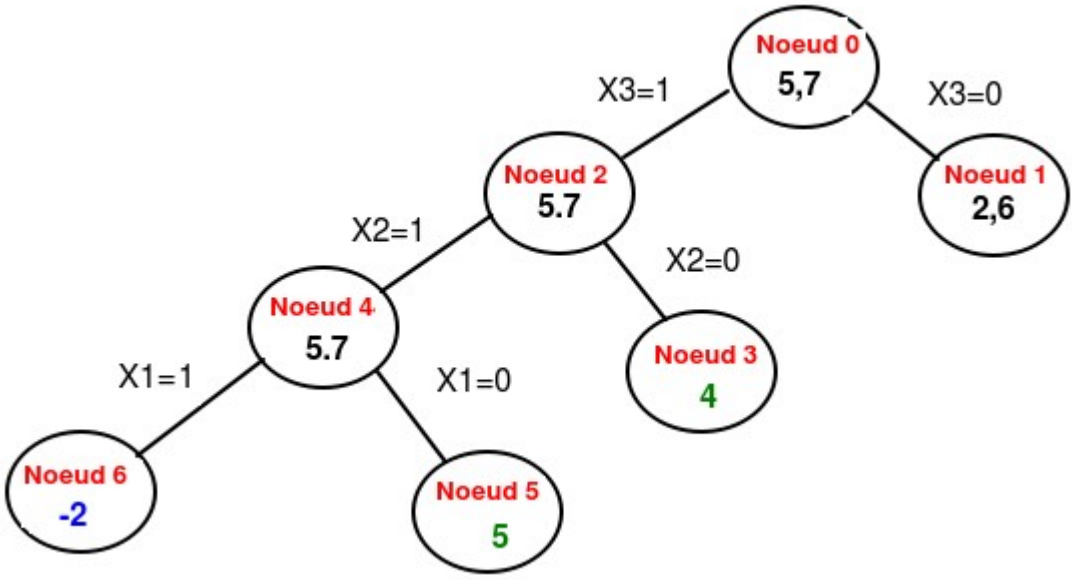
- Capacité du sac :**8**
- Nombre d'objets :**N=3**

On remplie notre tableau avec les valeurs et poids des objets respectivement .

	Obj1	Obj2	Obj3
Valeur	1	2	3
Poids	6	3	1
Valeur /Poids	1/6	2/3	3/1

Séparation : la première étape est la séparation,ce principe se base sur l'application des critères de branchement en associant à chaque objet i, une variable de décision binaire **Xi** qui vaut **1** si l'objet est sélectionné et qui vaut **0** sinon.

On aura le schéma suivant après avoir appliquer en plus l'évaluation vue en cours c'est à dire :



Nœud 0 :

- On sélectionne l'objet 3 ,donc le poids restant 7 (8-1), la valeur est 3.
- On sélectionne l'objet 2 ,donc le poids restant 5 (7-2) , la valeur est 2.
- On sélectionne 5/6 de l'objet 1 ,la valeur est 5/6

la valeur totale sera alors 5,7

Nœud 1 :

- On sélectionne l'objet 2,donc le poids restant 5 la valeur est 2.
- On sélectionne 4/6 de l'objet 1 ,la valeur est 4/6

la valeur totale sera alors 2,6

Nœud 2:

- *On sélectionne l'objet 3 ,donc le poids restant 7 (8-1), la valeur est 3.*
- On sélectionne l'objet 2 ,donc le poids restant 4 (7-3) , la valeur est 2.
- On sélectionne 4/6 de l'objet 1 ,la valeur est 4/6

la valeur totale sera alors 5,7

Nœud 3:

- On sélectionne l'objet 3 ,donc le poids restant 7 (8-1), la valeur est 3.
- On sélectionne l'objet 1,donc le poids restant 6 (7-1), la valeur est 1.

la valeur totale sera alors 4 .

Nœud 4:

- *On sélectionne l'objet 3 ,donc le poids restant 7 (8-1), la valeur est 3.*
- On sélectionne l'objet 2 ,donc le poids restant 4 (7-3) , la valeur est 2.
- On sélectionne 4/6 de l'objet 1 ,la valeur est 4/6

la valeur totale sera alors 5,7

Nœud 5:

- *On sélectionne l'objet 3 ,donc le poids restant 7 (8-1), la valeur est 3.*
- On sélectionne l'objet 2 ,donc le poids restant 4 (7-3) , la valeur est 2.

la valeur totale sera alors 5 .

Nœud 6:

- On sélectionne l'objet 3 ,donc le poids restant 7 (8-1), la valeur est 3.
- On sélectionne l'objet 2 ,donc le poids restant 4 (7-3) , la valeur est 2.

Si on veut sélectionner l'objet 1 dont le poids =6 on aura un poids négatif (4-6=-2 a dépassé la capacité du sac) ce qui n'existe pas, d'où on déduit qu'on a atteint la condition d'arrêt .

Donc la meilleur solution réalisable est alors de prendre **l'objet 2 et l'objet 3** avec une valeur totale maximale qui est 5.

Implémentation :

Choix du langage de programmation :

Une petite recherche sur la relation entre la complexité des méthodes de résolution exactes et le choix du langage de programmation me conduit a opter pour le **C** et non pas C++,car non seulement les bibliothèques en C++ (STL , math …) vont mettre plus de temps et aussi si on a plusieurs nœuds on risque d'avoir des limites de chargement mémoire.

Déterminer la valeur maximale ?

Avoir la valeur maximale signifie de calculer le max des valeurs totales pour des solutions réalisables.

Pour avoir une valeur totale d'une solution réalisable ,on doit savoir les objets mis dans le sac ,car selon le principe dévaluation séparation (appelée aussi PSE) afin de savoir la décision qui apportera la valeur maximale a chaque nœud on doit savoir les valeurs des nœuds suivants ,et ce processus se répète tout au long de la profondeur de notre arbre .

Ce processus nous conduit a définir une fonction **valeurs_max()** qui prends en paramètres : les poids et valeurs des objets ,le numéro de l'objet et la capacité du sac .

J'ai opté pour la récursivité car coté performance dans les parcours d'arbre on est beaucoup mieux qu'on itératif et aussi en analysant le principe de notre algorithme,

- A chaque fois on ajoute la valeur de l'objet choisi et et on définie une nouvelle capacité (la capacité précédente-le poids de l'objet choisi) puis on résout le problème (récursivité).
- Sinon on prend juste les objets déjà présents dans le sac (la capacité précédente)
- On résout le problème en choisissant la valeur maximale des deux cas.

La fonction nous permet de prendre la décision qui nous apporte la valeur maximale si :

- Ne pas sélectionner l'objet **k** avec (**k<=N**) et **N= nombre d'objets** revient à résoudre le problème avec **N-1** objets .
- Sélectionner l'objet **k** ,donc dans ce cas on aura alors la valeur de l'objet **k** ainsi que le max pour le problème de **k-1** objets.(ce qui revient a appliquer la séparation et évaluation).

Déterminer les articles à mettre dans le sac ?

Une fois on a les valeurs maximales , on doit déterminer les articles à mettre dans le sac ,pour cela j'ai déclaré la fonction la fonction **selection() q**ui prends en paramètres : les poids et valeurs des objets ,la capacité du sac et une variable déterminant si l'objet sera mit dans le sac ou non.

Cette fonction vérifie d'abord si l'objet n'est pas encore sélectionné,aussi que le poids de l'objet n'a pas dépassé la capacité du sac ,puis en déterminant l'objet qui a le maximum de **valeur/poids** parmi les objet **,**si ce dernier n'est pas déjà dans le sac , on le sélectionne alors.

Conclusion :

Pour trouver la solution optimale, et être certain qu'il n'y a pas mieux, il faut utiliser une méthode exacte, qui demande un temps de calcul beaucoup plus long (si le problème est difficile à résoudre). Il n'existe pas une méthode exacte universellement plus rapide que toutes les autres. Chaque problème possède des méthodes mieux adaptées que d'autres.

Sources :

- Support de cours **Optimisation et recherche opérationnelle** M1-CHPS-UVSQ
- https://www.slideshare.net/chbassem/problme-de-sac-dos
- https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_sac_%C3%A0_dos
- https://devdocs.io/c/

LOUGANI FAOUZI