Résolution du sac à dos multi-objets avec la Programmation Dynamique

Max z=8.x1 + 9.x2 + 6.x3
s.c. 4.x1 + 3.x2 + 2.x3 <= 11
x1, x2, x3 entier positif

Ici, les paramètres du problème sont d'une part le poids P du sac (limité à 11) et les **nombres i de types d'objets** (i dans {1, 2, 3}, le nombre de chaque type d'objet est donné par les variables xi dans la formulation ci-dessus).

ATTENTION : i est le nombre de premiers types d'objets et non le nombre d'objets. Nous pouvons avoir deux premiers types d'objets (i=2) et 5 objets (par exemple x1=2 de type 1 et x2=3 de type 2).

D'où l'idée d'une fonction f(P,i) qui indique l'utilité maximale du sac de poids P avec i premiers types d'objets (peu importe la composition du sac). Avec cette définition, le **principe de sous-optimalité** indique que f(P,i-1) est l'utilité maximale pour tout sac de poids P avec i-1 premiers types d'objets ; et que f(P-pi,i) est l'utilité maximale **pour tout** sac de poids (P-pi) avec i premiers types d'objets, où pi est le poids d'une unité du ième type d'objet.

L'utilité maximale d'un sac de poids P avec i premiers type d'objet, i.e. f(P,i), est donc :

- Soit le **meilleur sac de poids P sans objet de type i et sans la possibilité d'ajouter un objet de type i** (c'est le meilleur sac avec seulement les autres i-1 premiers types d'objets). L'utilité de ce sac est donnée par f(P,i-1). Autrement dit, le sac est déjà plein avec les i-1 premiers types d'objet et f(P,i) = f(P,i-1) (nombre venant de la colonne P);
- Soit le **meilleur sac (avec éventuellement les i premiers types d'objets) ayant encore la possibilité d'ajouter un objet de type i** avec son poids pi. Ce sac a donc seulement un poids de P-pi et son utilité est f(P-pi,i). A ce sac, on peut donc lui ajouter un objet de type i avec son utilité ui et nous avons alors f(P,i)=f(P-pi,i)+ui (nombre venant de la ligne i).

D'où la définition de récurrence :

$$f(P,i) = \max \{ f(P, i-1), f(P-pi, i) + ui \}.$$

Autrement dit, nous prenons le meilleur des deux meilleurs sacs ! Chaque sac étant le meilleur sur sa dimension, P ou i.

Donc nous obtenons le tableau d'utilité f(P,i) suivant :

3	0	0	6	9	12	15	18	21	24	27	30	33
2	0	0	0	0	9	9	9	18	18	18	27	27
1	0	0	0	0	8	8	8	8	16	16	16	16
i / P	0	1	2	3	4	5	6	7	8	9	10	11

Dans ce tableau chaque case est donnée par les coordonnée P et i.

Le calcul de la case (P=7, i=3) (en orange), l'utilité f(7,3) a la valeur 21 qui vient du max{18, 15+6}.

En effet, f(7,2) qui vaut 18 fournit l'utilité maximale du sac à poids égal 7 avec les 2 premiers types d'objets. Le f(7-2,3) qui vaut 15 donne l'utilité maximale du sac à poids 5 avec les trois types d'objets (peu importe la composition des objets, mais leur poids est de 5 en tout).

Or, f(7-2,3) est la meilleure utilité que nous pouvons obtenir avec 3 types d'objets, mais avec un poids égale à 7-2 où 2 est le poids d'une unité du 3ème type d'objet. Donc f(7-2,3)+6, avec 6 l'utilité apportée par une unité supplémentaire du 3ème type d'objet, donne la meilleure utilité que l'on peut obtenir d'un sac de poids 5 avec une unité d'objet de type 3 en plus.

Le max entre f(7,2) (meilleure utilité d'un sac de poids 7 avec 2 types d'objets) et f(7-2, 3)+6 (meilleure utilité d'un sac de poids 5 avec 3 types d'objets plus un objet de type 3 en plus), donne donc la meilleure utilité que nous pouvons obtenir pour f(7,3).

Un autre exemple avec le type 2 (en mauve), f(9,2)=max{ f(9,1)=16, f(9-3,2)+9=18 }=18.
Un 3ème exemple avec le type 3 (en jaune), f(11,3)=max{ f(11,2)=27, f(11-2,3)+6=33 }=33.

Remarque sur la complexité de l'algorithme et du problème ;

Les utilités f(P, i) peuvent être calculées en O(i.P), où i est le nombre de type d'objets et P le poids max. du sac. Donc si la taille du problème est le nombre i de type d'objets (ou encore le nombre de variables), et P le paramètre du problème, alors **le calcul de f(P, i) a une complexité pseudo-polynomiale** : polynomiale en la taille i du problème ; mais pseudo, car le polynôme en i dépend d'un paramètre P qui lui-même dépend de l'instance du problème (on peut avoir un sac de poids max à 11 comme un sac de poids max à 200 ; cette valeur est connue au début de la résolution). On voit bien que suivant la valeur de P, on a un tableau plus ou moins grand, et donc plus ou moins de calculs à effectuer pour obtenir les utilités.

Or, le problème du **sac à dos multi-objets est NP-difficile**, i.e. on ne sait pas à ce jour s'il existe ou pas un algorithme de complexité temporelle polynomiale (alors qu'il existe un de complexité temporelle exponentielle) pour le résoudre. Néanmoins, là, nous avons exhibé un algorithme de complexité temporelle pseudo-polynomiale. Dans ce cas, le problème du **sac à dos multi-objets est qualifié de NP-difficile au sens faible**.