Chapitre 5

Problèmes de remplissage

Problèmes de Remplissage 1/10

- **Problème de sac à dos** (Knapsack Problem KP)

Modélisez une situation analogue au remplissage d'un sac à dos, ne pouvant dépasser la capacité maximale du sac **W**, avec un ensemble donné de n produits ayant chacun un poids **wi** et une valeur **bi.** Maximisez la valeur totale des objets mis dans le sac à dos.

1) Modélisez le problème sous forme d'un problème linéaire.

2) Résoudre le problème.

Problèmes de Remplissage 2/10

- **Modélisation du problème (**Knapsack 0-1)

Variable de décision

xj = 1 si le produit est transporté

0 sinon

Fonction objectif

Maximiser le bénéfice : $\sum_{i=1}^{n} xj$ biavec i ∈ { 0,…,n}

Contraintes :

$\sum_{i=1}^{n} xj$ wi≤ W // ne pas dépasser la capacité du sac à dos .

Problèmes de Remplissage 3/10

Résolution avec programmation dynamique

Principes :

- ✓ L'objectif est de diviser le sac de capacité k en k sous-sacs dont la capacité allant de zero à k.
- ✓ La méthode commence par la construction d'un tableau V donnant les bénéfices possible pour chaque sous-sac
- ✓ Ordonner les produits par volume croissant
- ✓ Chaque case du tableau représente le bénéfice maximum possible **V(i,w)** pour les i premiers objets avec une capacité w.

Problèmes de Remplissage 4/10

- **Résolution avec programmation dynamique**

Les sous-sacs à dos ordonnés selon leur capacité

L'ensemble des produits que le sac peut contenir allant de 1 à n. les produits sont ordonnés dans l'ordre croissant de leurs poids

	1	2	…	…	…	W
1						
…						
n						

Problèmes de Remplissage 5/10

- **Algorithme programmation dynamique**

```
// Remplir le tableau de bénéfice
        For w=0 to W      // initialisation
            V [0,w]=0
        For i=0 to n
            V [i,0]=0
        For  i= 1 to n
            For w=1 to W
                if (wi <= w)
                    if (bi + V [i-1,w-wi] > V[i-1,w])
                        V[i,w]=V[i-1,w-wi] +bi
                    Else  V[i,w]=V[i-1,w]
                Else   V[i,w]=V[i-1,w]
            End For
        End For
```

Problèmes de Remplissage 6/10

- **Algorithme programmation dynamique**

// Pour déterminer les produits qui sont dans le sac à dos on procède de la manière suivante

Soient i=n et k=W

if (V [i,k] ≠ V[i-1,k]) // Entre l'itération i-1 et i , le produit a été ajouté

i = i-1 , k= k-wi // la capacité restante après l'ajout du produit

Else i =i-1

Problèmes de Remplissage 7/10

Exercice d'application :

Cherchez à à résoudre le problème suivant par programmation dynamique :

- Ensemble d'élément n=4
- i(wi, bi) ➔ 1(2,3)

  2(3,4)

  3(4,5)

  4(5,6)
- Capacité totale W=5

Problèmes de Remplissage 8/10

Solution :

i\ wi	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

➔ Le bénéfice maximale est de **V(4,5) = 7**

Problèmes de Remplissage 9/10

- **Calcul du bénéfice** (pour i=1)

  Pour w=1

  wi =2 ➔ w-wi = -1

  V (1,1) = V(0,1) =0

  Pour w =2

  wi =2 ➔ w-wi =0

  V (0,0)+3 = 3> V(0,2)=0

  Donc V (1,2)= 3+V(1,1) =3

  Pour w=3

  V (1,3) = 3

Problèmes de Remplissage 10/10

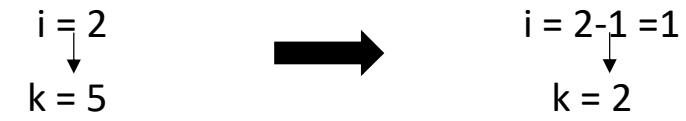
- **Déterminons les produits qui sont dans le sac :**

On a V (4,5) = V(3, 5) ➔ { 4 } ∉ S

V (3,5) = V(2,5) ➔ { 3 } ∉ S

V (2,5) ≠ V(1,5) ➔ { 2 } ∈ S

Donc itération actuelle nouvelle itération



Finalement **S = {1 ,2}**