Algorithmique avancée
Sac-à-Dos par BandB

Denis TRYSTRAM
Notes de cours ENSIMAG Alternants 2A

Sept. 2023

Détail de la méthode sur un exemple

On considère le problème du Sac-à-Dos (KnapSack)

- **Entrées** :
  n couples de nombres entiers (w_i, b_i) — poids, bénéfices
  Un entier K (capacité du sac)
- **Question** Remplir le sac S en maximisant le bénéfice total c'est-à-dire, sélectionner des objets dans S tels que $\Sigma_{i \in S} w_i \leq K$

Une instance de KnapSack

- Attention : le problème ici est un MAX

Capacité du sac : $K = 8$

index	1	2	3	4	5	6	7
	(2,5)	(3,8)	(6,11)	(4,9)	(5,12)	(8,17)	(4,8)

Analyse de complexité

Le problème du Sac-à-Dos est NP-complet.
La preuve est simple par une réduction à partir du problème 2Partition qui est NP-complet :

2Partition.
Instance : n entiers n_i et un entier pair $S = \Sigma_{1 \leq i \leq n} n_i$.
Question : Existe-t-il une partition des entiers en deux sous-ensembles A_1 et A_2 telle que $\Sigma_{i \in A_1} n_i = \Sigma_{i \in A_2} n_i$?

- La vérification d'une solution de KnapSack se fait en $\mathcal{O}(n)$ en comparant la somme des poids et bénéfices des objets du sac W et à K, il est donc dans $\mathcal{NP}$.

- La vérification d'une solution de KnapSack se fait en $\mathcal{O}(n)$ en comparant la somme des poids et bénéfices des objets du sac W et à K, il est donc dans $\mathcal{NP}$.

- La réduction τ est la suivante :
  On transforme une instance $\mathcal{I}$ de 2Partition en une instance de KnapSack où $w_i = b_i = n_i$ et $W = K = \frac{S}{2}$.

  On vérifie que cette instance est positive *ssi* $\tau(\mathcal{I})$ l'est.

Preuve

- L'implication est évidente par construction.
- Pour la réciproque, on considère une instance positive d'une transformée par τ.

  Comme le problème détermine un sous-ensemble d'objets A tel que $\Sigma_{i \in A} b_i \geq \frac{S}{2}$ et $\Sigma_{i \in A} w_i \leq \frac{S}{2}$, on a bien une partition en A et son complémentaire tel que $\Sigma_{i \in A} n_i = \frac{S}{2}$.

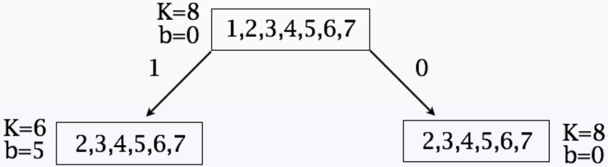
Mise en oeuvre du BandB

- Stratégie de découpe :

  On parcourt toutes les configurations objet par objet.
  On branche en considérant que l'objet est sélectionné ou non.

  On choisit les objets dans l'ordre initial, mais tout autre choix serait possible.

On part du premier objet $(2, 5)$.



S'il est placé dans le sac, le bénéfice est 5 et il reste $K = 8 - 2 = 6$.

Borne pour les coupes

Une borne supérieure *naturelle* d'un sac-à-dos partiellement rempli est son bénéfice plus le bénéfice restant que l'on obtiendrait si l'on remplissait le reste avec le meilleur bénéfice possible restant. Cette politique est optimale, mais pas toujours réalisable.

- Pour calculer une telle borne,
  on introduit la densité de l'objet i comme $\lambda_i = \frac{b_i}{w_i}$

Densité

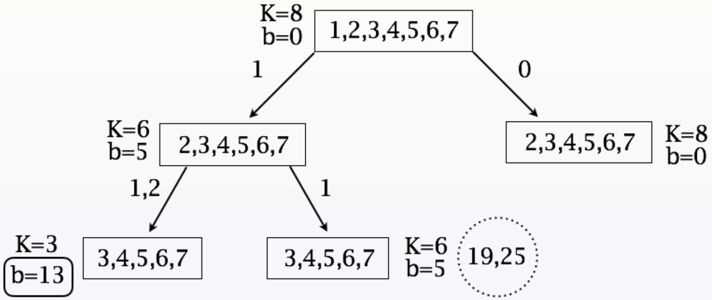
Sur l'instance de l'exemple,

index	1	2	3	4	5	6	7
	(2,5)	(3,8)	(6,11)	(4,9)	(5,12)	(8,17)	(4,8)
densité	2,50	2,67	1,83	2,25	2,40	2,12	2,00

On trie les objets par densité décroissante

index	2	1	5	4	6	7	3
	(3,8)	(2,5)	(5,12)	(4,9)	(8,17)	(4,8)	(6,11)

- On continue le parcours en considérant le second objet :



- Si l'objet est mis dans le sac avec le premier, on obtient un bénéfice de $5 + 8 = 13$ et il reste un poids de 3.

Aucun objet ne rentre, on a donc terminé,
on a ainsi obtenu une borne inférieure de la solution cherchée.

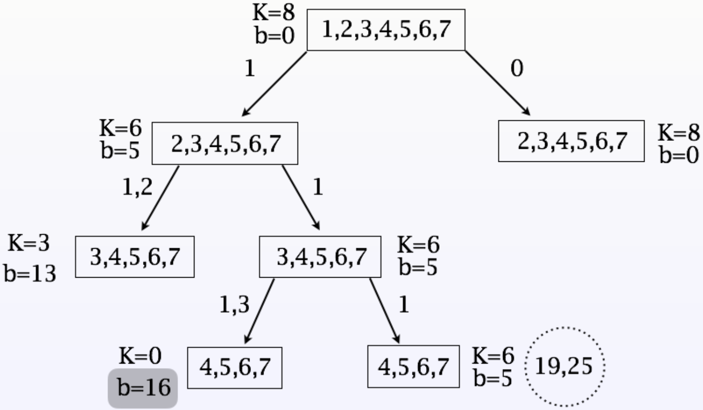
Evaluation de l'autre branche

- Si l'objet n'est pas mis dans le sac avec le premier, le bénéfice reste de 5 et il reste un poids de 6
- La meilleure chose que l'on puisse faire est de mettre l'objet de plus grand bénéfice qui rentre (ici, le cinquième) et compléter avec une partie de l'objet suivant de façon à saturer le sac.

On obtient ainsi $19,25$

Comme c'est supérieur à la solution courante 13, on ne peut rien dire de plus...

- On continue le parcours en considérant le troisième objet



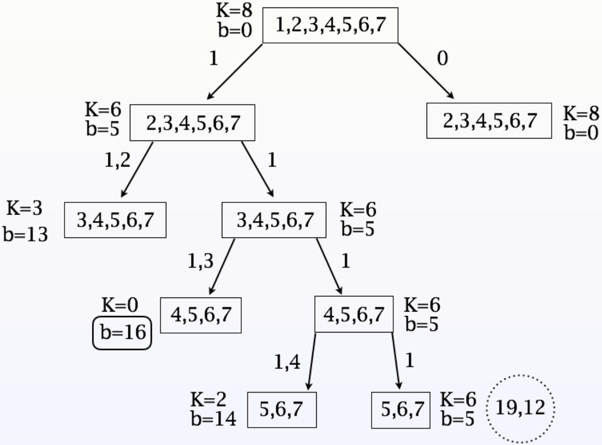
On améliore la solution courante :

- Les objets 1 et 3 sont sélectionnés pour un bénéfice de $5 + 11 = 16$

Inutile de développer le sous-arbre car aucun autre objet ne peut plus rentrer dans le sac.

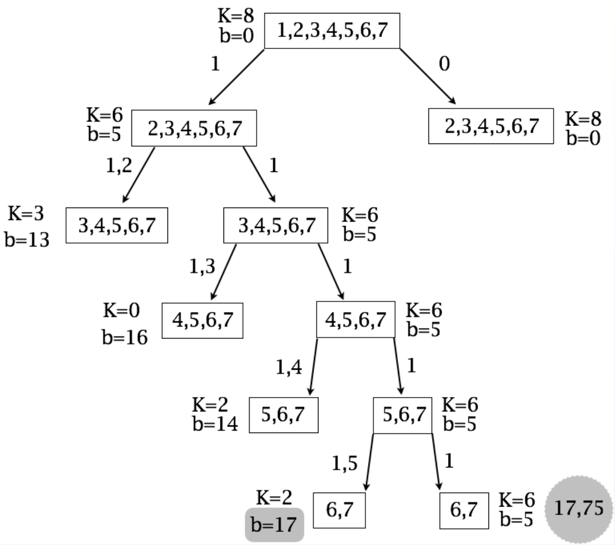
- On continue à développer l'autre branche de l'arbre en considérant maintenant l'objet 4.

La borne supérieure n'est pas changée.


K=8
b=0
1,2,3,4,5,6,7
1
0
K=6
b=5
2,3,4,5,6,7
2,3,4,5,6,7
K=8
b=0
1,2
1
K=3
b=13
3,4,5,6,7
3,4,5,6,7
K=6
b=5
1,3
1
K=0
b=16
4,5,6,7
4,5,6,7
K=6
b=5
1,4
1
K=2
b=14
5,6,7
5,6,7
K=6
b=5
19,12

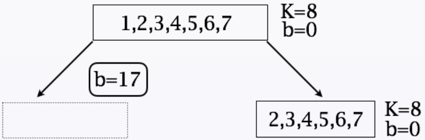
On améliore la solution courante :

- **Sous arbre gauche :**
  les objets 1 et 4 sont sélectionnés
  il reste un poids 2 et le bénéfice est de 14 qui est inférieur à solution inférieure précédente de 16
  Aucun autre objet ne peut plus rentré dans le sac.
- **Sous-arbre droit :**
  La borne supérieure s'améliore à $19,12$ mais reste toujours plus grande que le bénéfice courant
  On continue avec les objets restants.

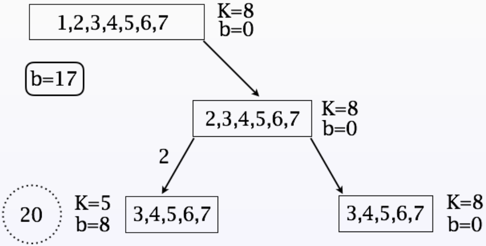

K=8
b=0
1,2,3,4,5,6,7
1
0
K=6
b=5
2,3,4,5,6,7
2,3,4,5,6,7
K=8
b=0
1,2
1
K=3
b=13
3,4,5,6,7
3,4,5,6,7
K=6
b=5
1,3
1
K=0
b=16
4,5,6,7
4,5,6,7
K=6
b=5
1,4
1
K=2
b=14
5,6,7
5,6,7
K=6
b=5
1,5
1
K=2
b=17
6,7
6,7
K=6
b=5
17,75

- Lorsque l'on place l'objet 5, on obtient un bénéfice de 17 et il reste un poids de 2 dans le sac.
  On améliore encore la borne inférieure (on se rapproche de l'optimum).
- Sur l'autre branche (s'il n'est pas mis dans le sac), on peut compléter le sac par une partie de l'objet 6.
  Le bénéfice devient : $5 + 6 \times \frac{17}{8} = 17,75$

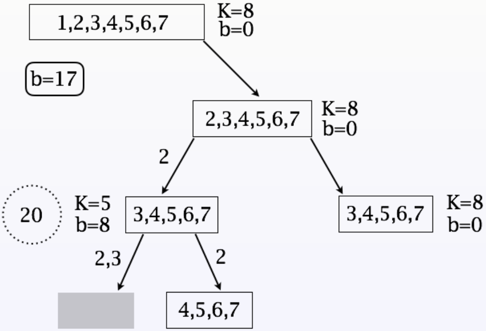
Explorons l'autre branche...



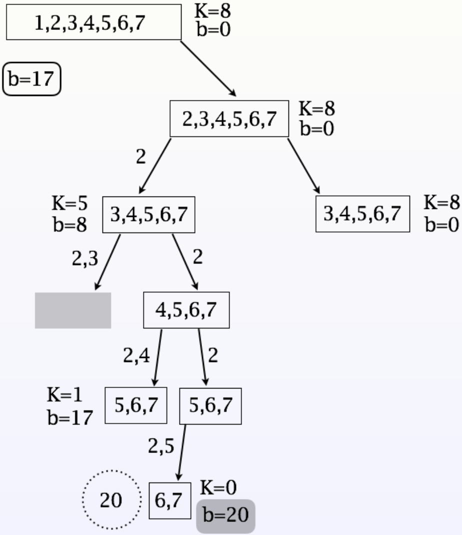
Explorons l'autre branche...



Explorons l'autre branche...

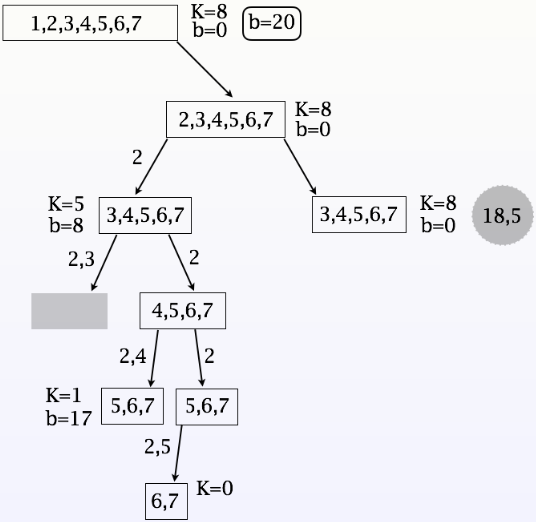


Explorons l'autre branche...



- Dans la configuration trouvée, la borne sup et la borne inf sont les mêmes,
  on ne peut trouver mieux *localement*...

Explorons la dernière branche...



Coupe

- L'heuristique d'évaluation de la borne supérieure permet d'arrêter l'exploration.
- La valeur optimale de la solution est $b = 20$

Bilan

Nous avons procédé à une énumération implicite des solutions possibles.

Beaucoup d'explorations ont été évitées par une évaluation efficace d'une borne[1].

Cette évaluation est un algorithme en $\Theta(n)$.

[1] Leur nombre dépend bien entendu de l'instance