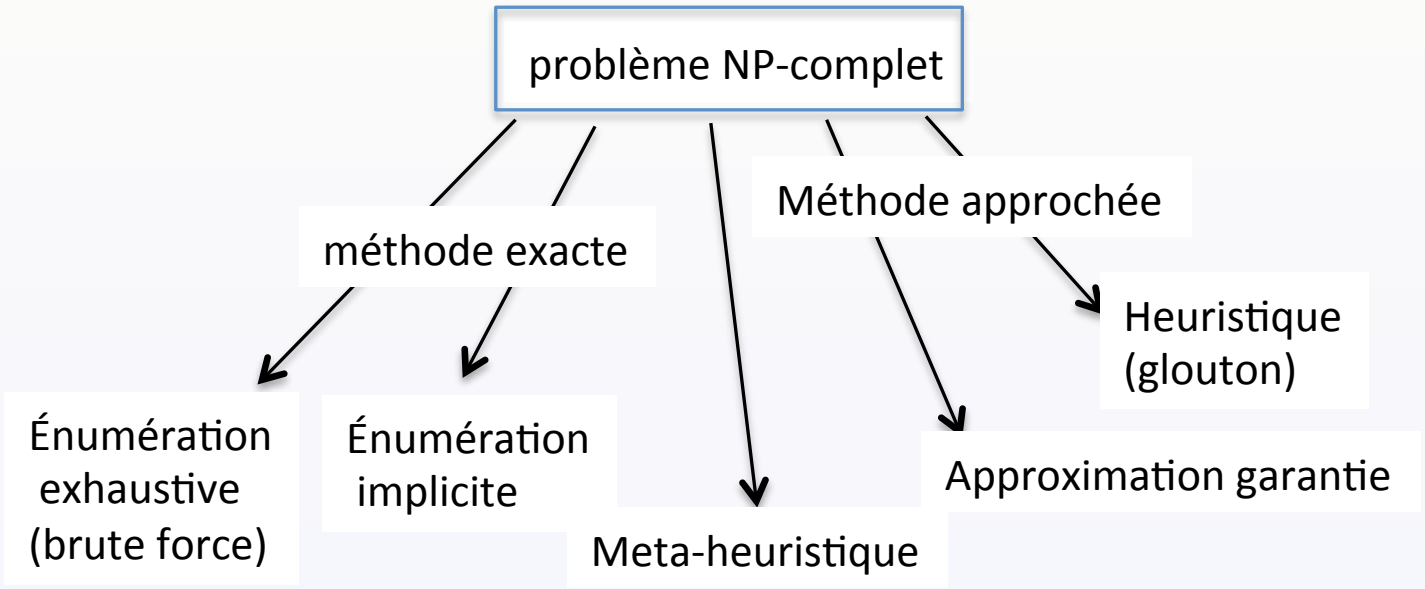
Algorithmique avancée
Branch and Bound

Denis TRYSTRAM
Notes de cours ENSIMAG Alternants 2A

sept. 2021

Différentes manières de résoudre un problème



Nous verrons dans la suite du cours ce que sont ces problèmes NP-complets et comment les caractériser...

Présentation du problème

On cherche à résoudre des problèmes *difficiles* par méthodes exactes.

- On ne peut pas espérer concevoir une solution en temps polynomial pour obtenir une solution exacte à ce type de problèmes.
- Par contre, nous allons montrer comment construire des solutions efficaces qui font beaucoup mieux que l'énumération exhaustive de toutes les solutions possibles.
  Bien sûr, ces stratégies restent exponentielles.

C'est le cas de la méthode de séparation/évaluation (plus connues par son nom en anglais : *Branch-and-Bound*).

B-and-B

On considère le problème général suivant[1] :

- minimiser $f(x)$ pour $x \in \Omega$ subset of IR^n
- sous un ensemble de contraintes

Une exploration exhaustive systématique de toutes les solutions possibles est impossible pour des tailles raisonnables de problèmes.

[1] Cette formulation englobe de très nombreux problèmes

Principe

Il est possible de limiter l'explosion combinatoire en introduisant un système de bornes dans le parcours des configurations possibles du problème.

Définition

Un algorithme Branch-and-Bound consiste à **énumérer systématiquement toutes les solutions candidates, où de larges ensembles de configurations inutiles sont éliminées en utilisant des bornes inférieures et supérieures** (estimations de la fonction objectif que l'on cherche à optimiser).

Concrétement

Une stratégie Branch-and-Bound divise le problème en sous-problèmes.

- Supposons que l'on dispose d'une méthode qui fournit une borne inférieure du coût de n'importe quelle solution dans un sous-ensemble donné.
- Si la meilleure solution trouvée jusqu'à présent a un coût plus petit que cette borne, il est inutile d'explorer les solutions de ce sous-ensemble.

BandB pour un problème de min

BandB repose sur deux procédures qui calculent (efficacement) une borne supérieure et inférieure de la valeur optimale de l'objectif dans chaque région.

- Une borne supérieure peut être déterminée par l'évaluation de n'importe quelle configuration dans la région considérée (par une heuristique glouton ou par optimisation locale).
- Une borne inférieure de la solution dans la région considérée est en général fournie par relaxation (ou par dualité).

Etape 1.

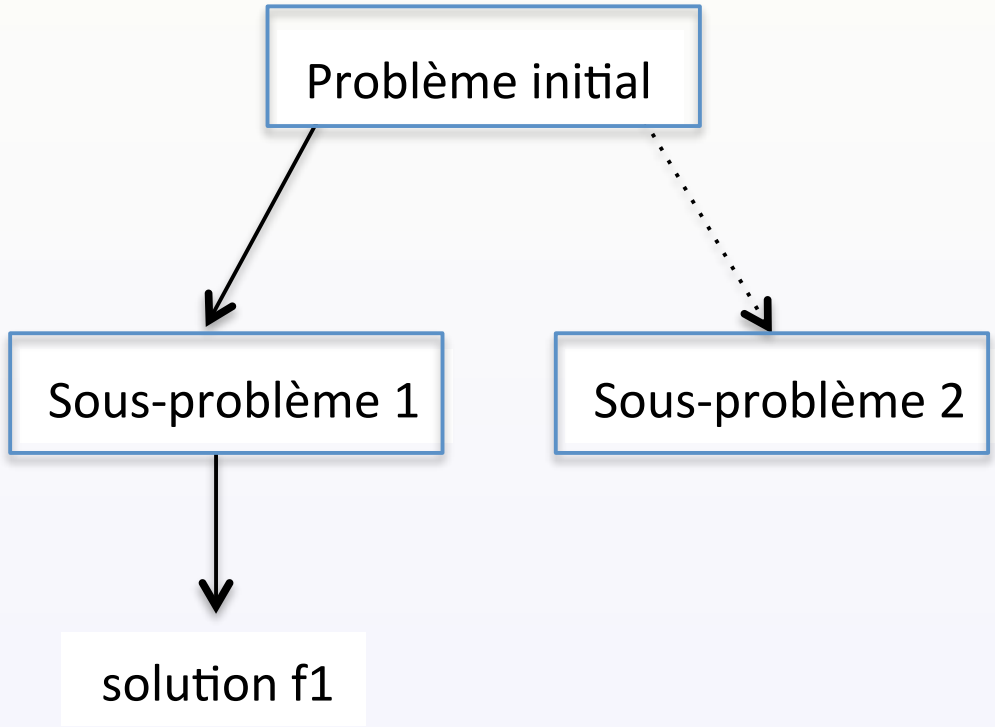


Figure: On explore une cascade de sous-problèmes jusqu'à l'obtention d'une solution réalisable (ici, $f1$).

Etape 1. Visualisation des valeurs de l'objectif.

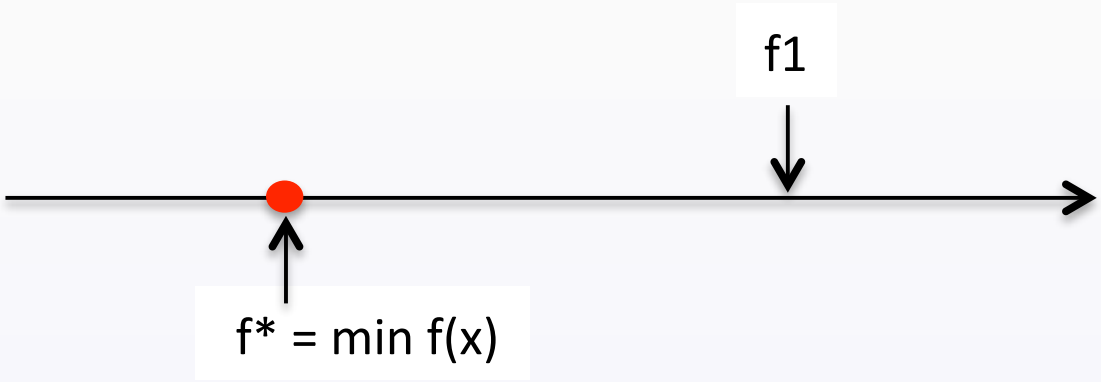


Figure: La première solution fournit une borne supérieure de l'optimal f^*.

Etape 2.

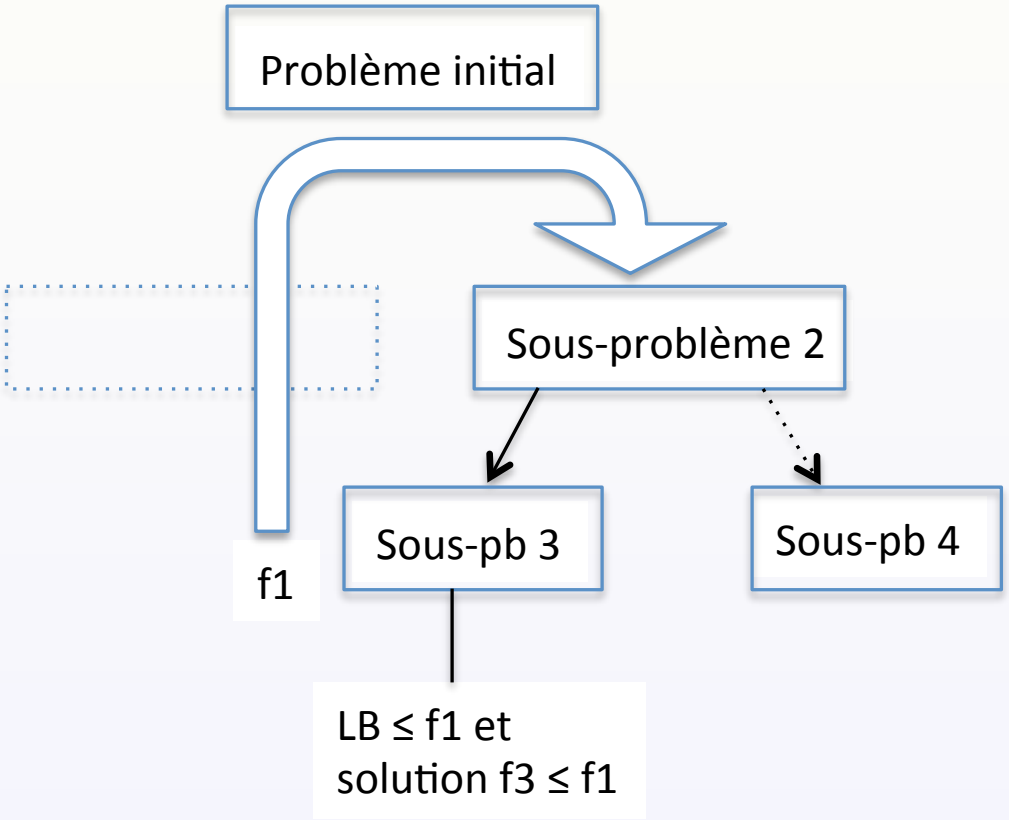


Figure: Exploration du sous-problème 3 : on continue à explorer l'arbre des configurations en cascade de sous-problèmes jusqu'à l'obtention d'une solution réalisable.

Visualisation de l'objectif.

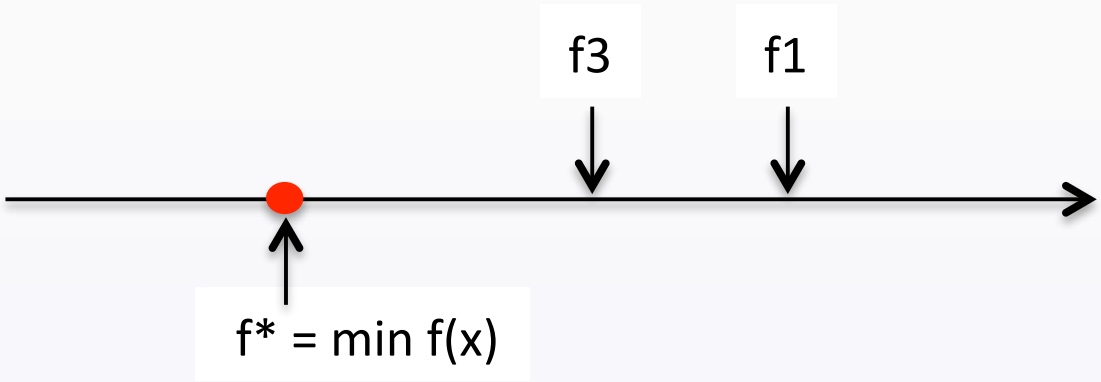


Figure: La solution $f3$ obtenue à l'étape 2 améliore la borne supérieure $f1$.

Etape 3.

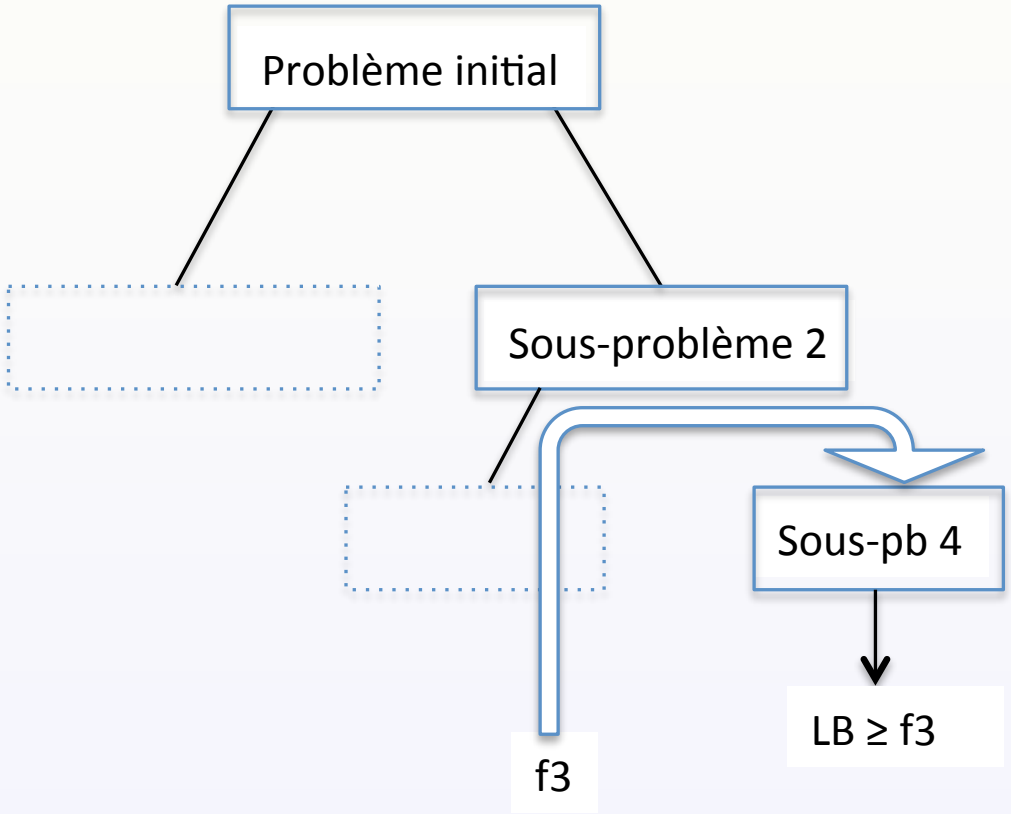


Figure: L'évaluation de la borne inférieure du sous-problème 4 montre que toutes les solutions seront moins bonnes que la solution déjà trouvée. On arrête l'exploration dans cette branche.

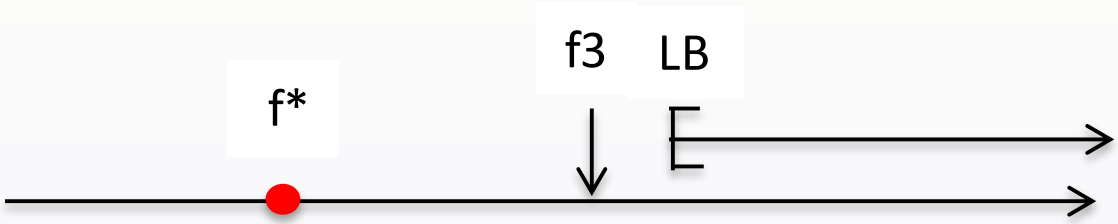


Figure: On arrête l'exploration dans cette branche car la borne inférieure du sous-problème est plus grande que la solution courante $f3$.

En résumé

Pour mettre en œuvre un BandB avec minimum, on a besoin de :

- Une stratégie de découpe en sous-problèmes (séparation).
- Une heuristique pour initialiser la borne supérieure.
- Une procédure (rapide) de calcul d'une borne inférieure (en général, non réalisable).

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 : $K = 8$

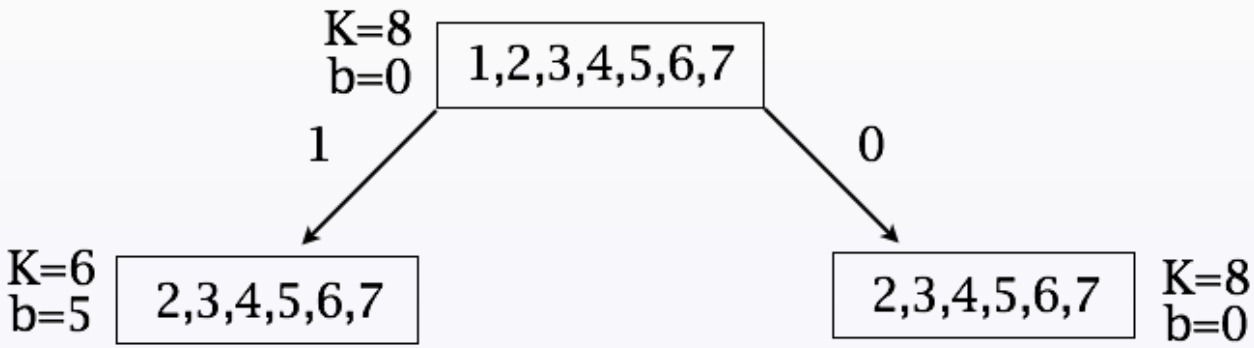
index	1	2	3	4	5	6	7
	(2,5)	(3,8)	(6,11)	(4,9)	(5,12)	(8,17)	(4,8)

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



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

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 on remplissait le reste avec le meilleur bénéfice possible restant.

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

Densité

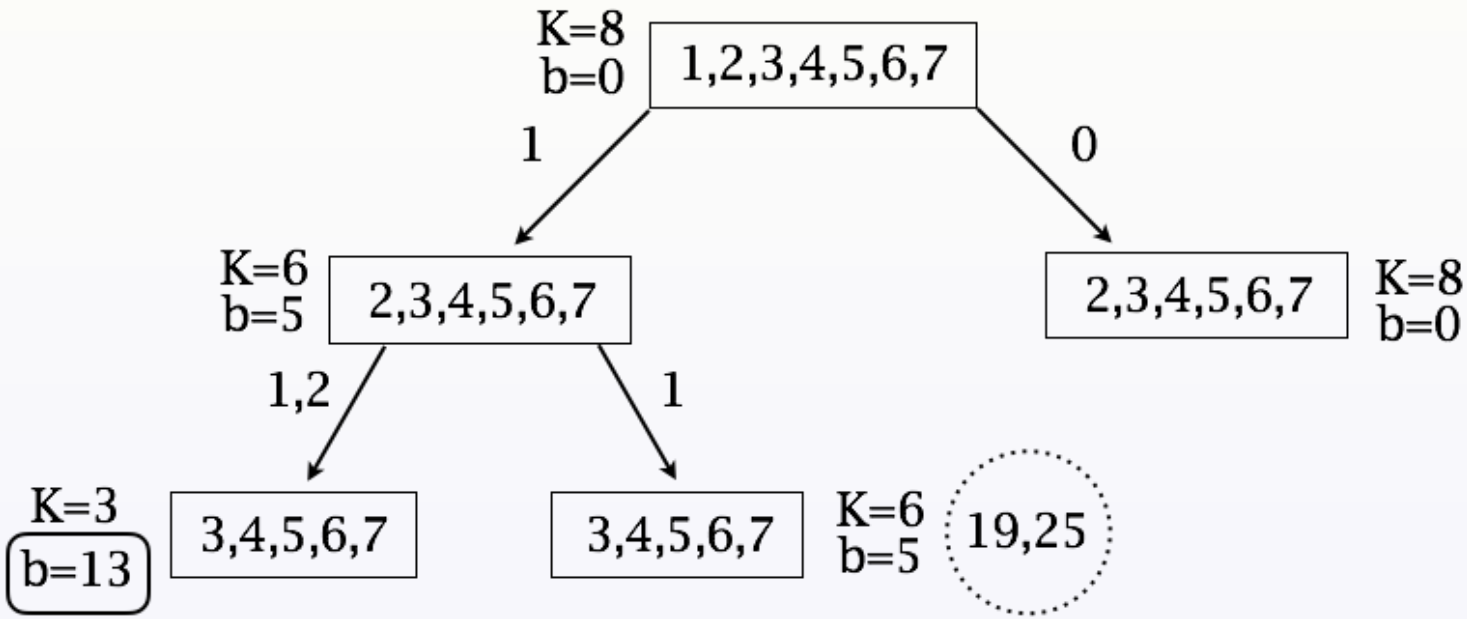
Sur l'instance de l'exemple,

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 une place de 2.

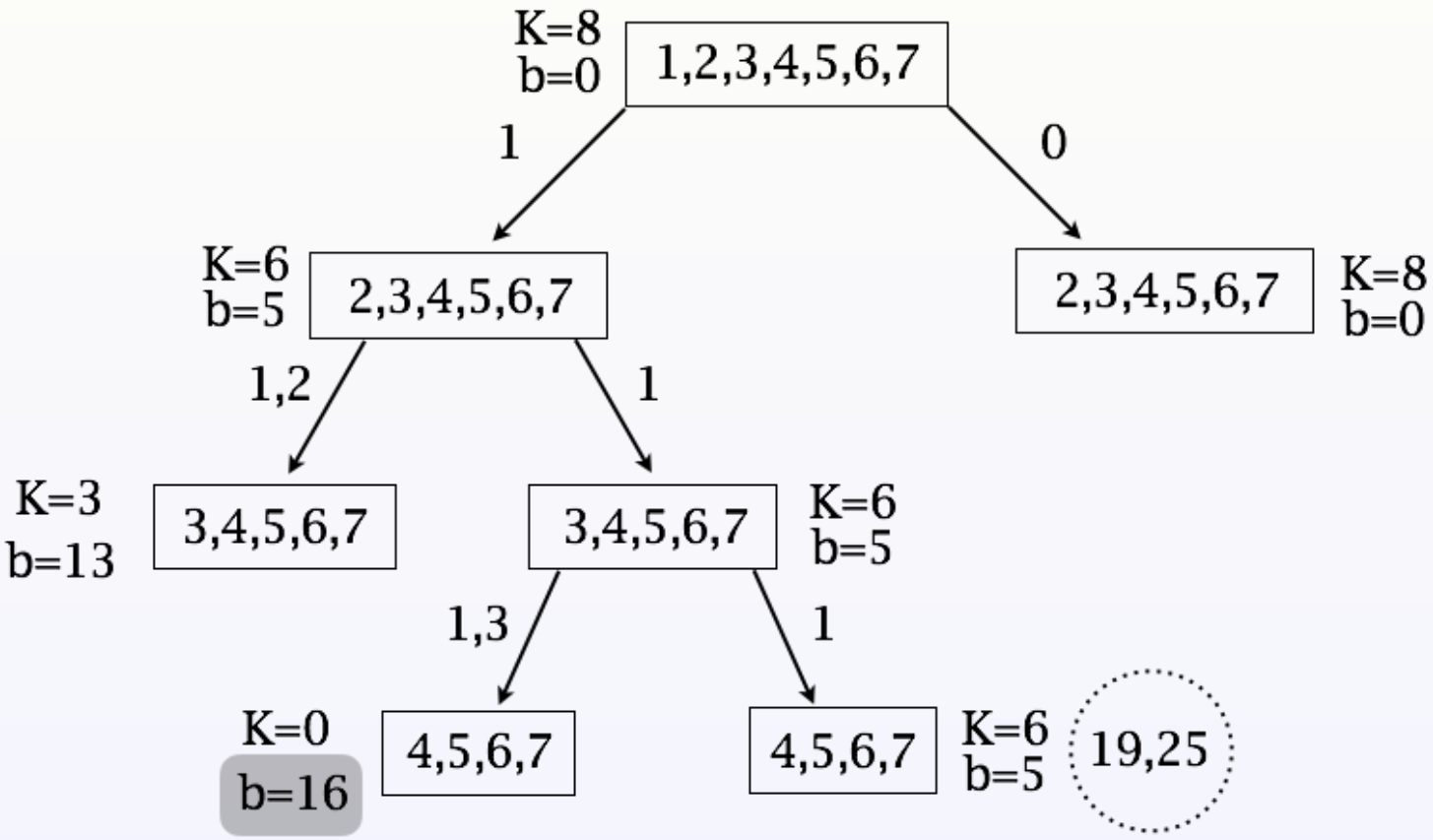
Aucun objet ne rentre, on a donc terminé,
on a ainsi obtenu une borne inférieure de la solution cherchée.

Evaluation

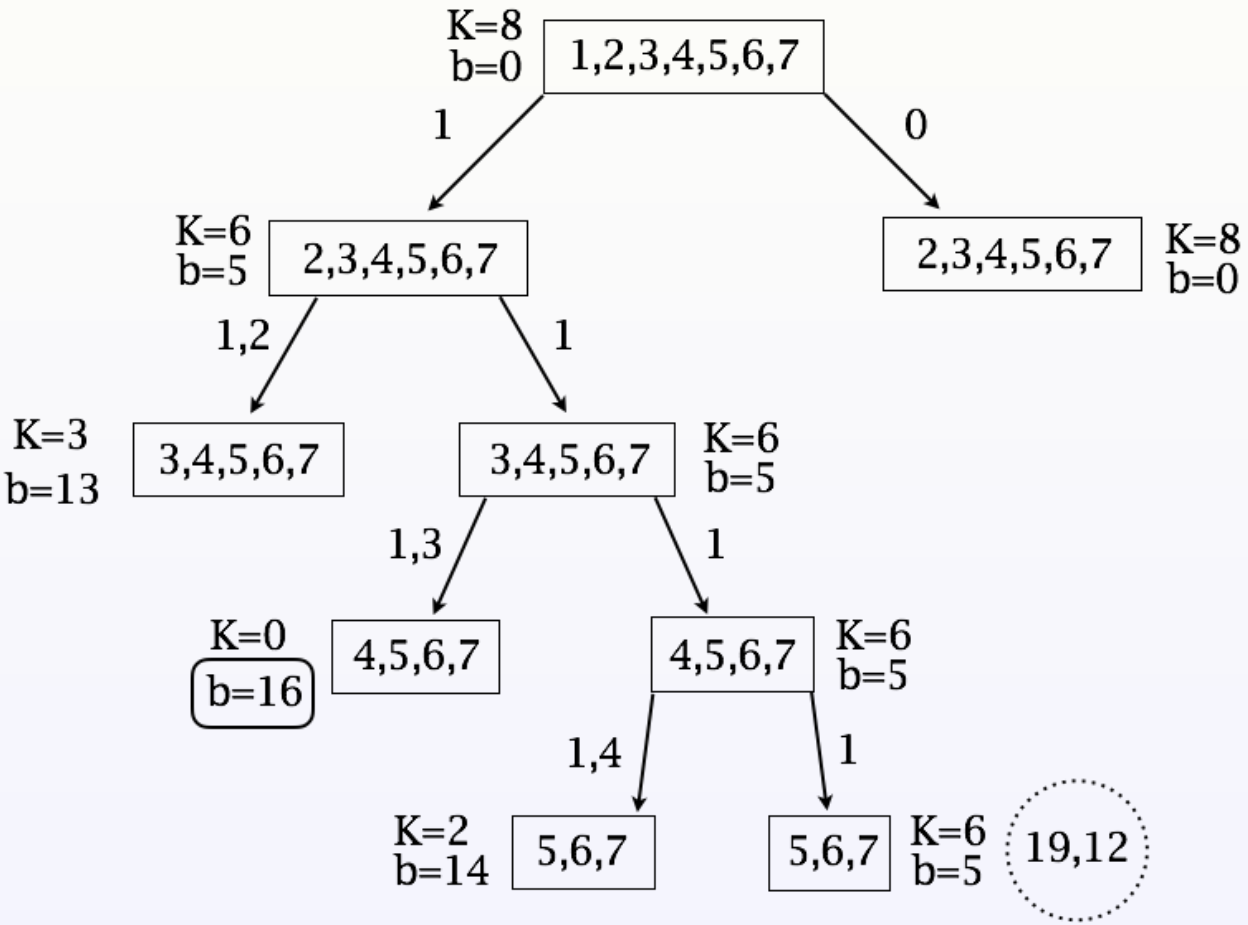
- Si l'objet n'est pas mis dans le sac avec le premier, le bénéfice reste de 5 et il reste une place de 6.
- La meilleure chose que l'on puisse faire est de mettre l'objet de plus grand bénéfice qui rentre (ici, le cinquième) et compléter avec une partie de l'ojet suivant de façon à saturer le sac.

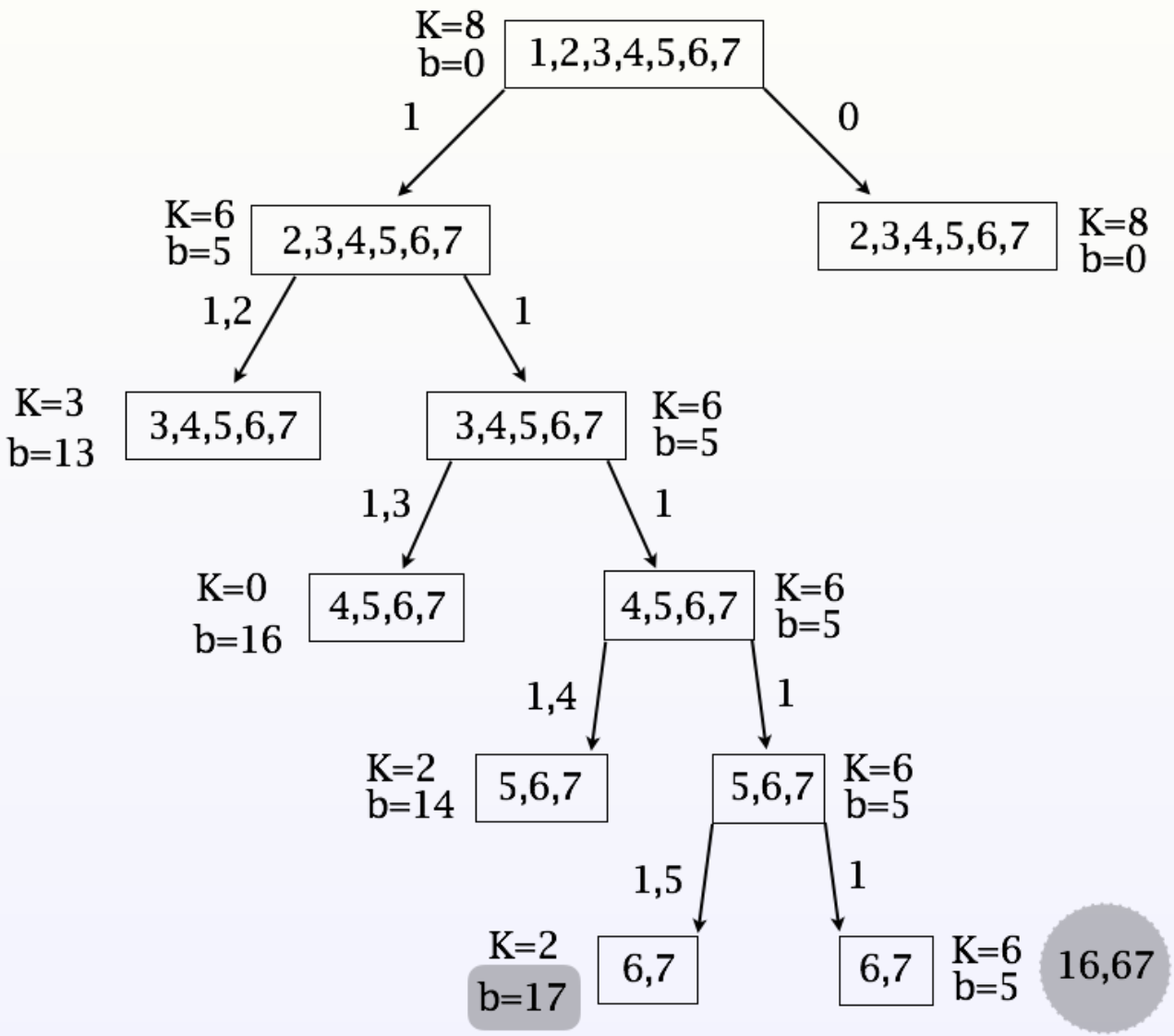
On obtient ainsi $19,25$.
Comme c'est supérieur à la solution courante 13, on ne peut rien dire de plus...

- On continue le parcours en considérant le troisième objet



On améliore la solution courante :
Objets 1 et 3 pour un bénéfice de 16.

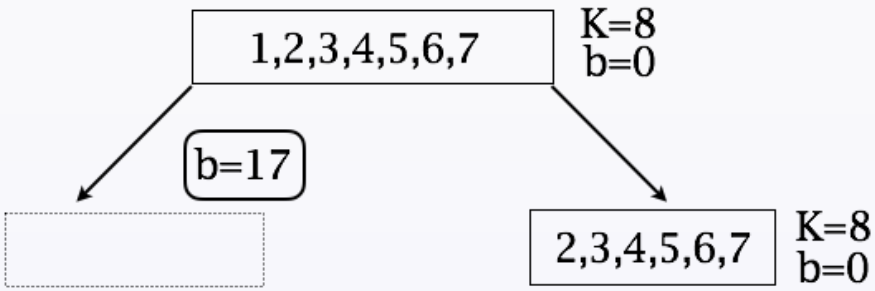

K=8
b=0
1,2,3,4,5,6,7
1
0
K=6
b=5
2,3,4,5,6,7
2,3,4,5,6,7
K=8
b=0
1,2
1
K=3
b=13
3,4,5,6,7
3,4,5,6,7
K=6
b=5
1,3
1
K=0
b=16
4,5,6,7
4,5,6,7
K=6
b=5
1,4
1
K=2
b=14
5,6,7
5,6,7
K=6
b=5
19,12


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
16,67

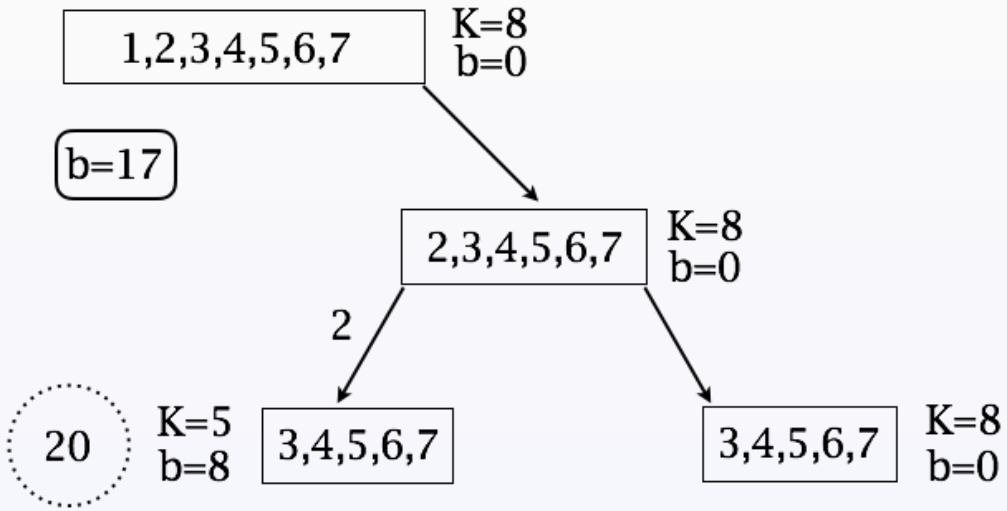
Coupe (borne supérieure)

- Lorsque l'on place l'objet 5, on obtient un bénéfice de 17 et il reste une place de 2 dans le sac.
  On améliore encore la borne inférieure (on se rapproche de l'optimum).
- S'il n'est pas mis dans le sac, on peut compléter le sac par une partie de l'objet 6.
- La borne ainsi obtenue est plus petite que la solution courante.
  On peut élaguer toutes les explorations restantes !

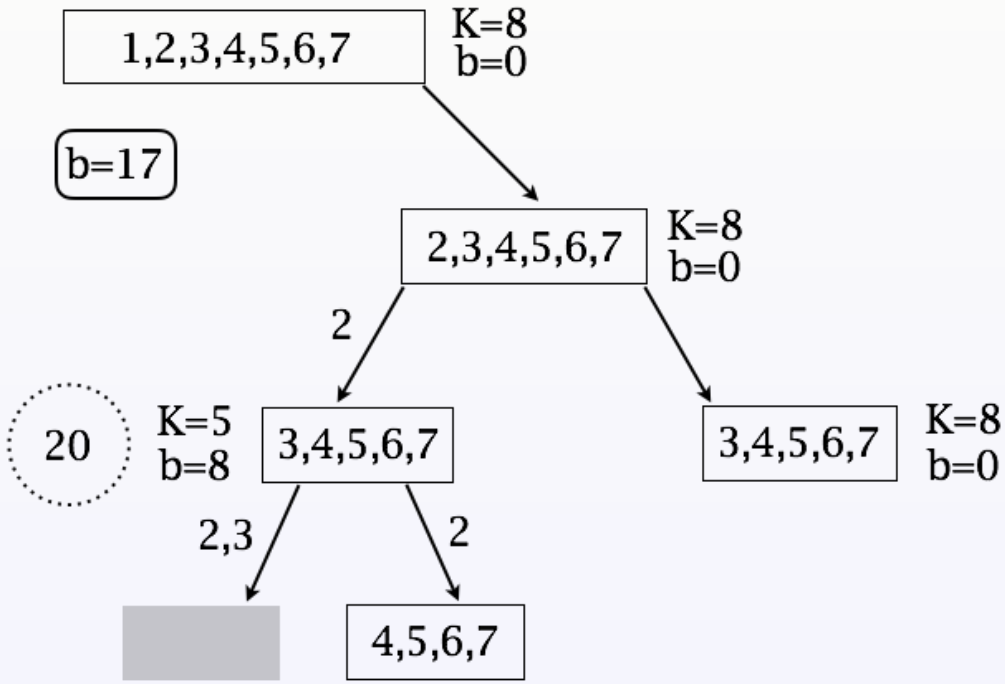
Explorons l'autre branche...



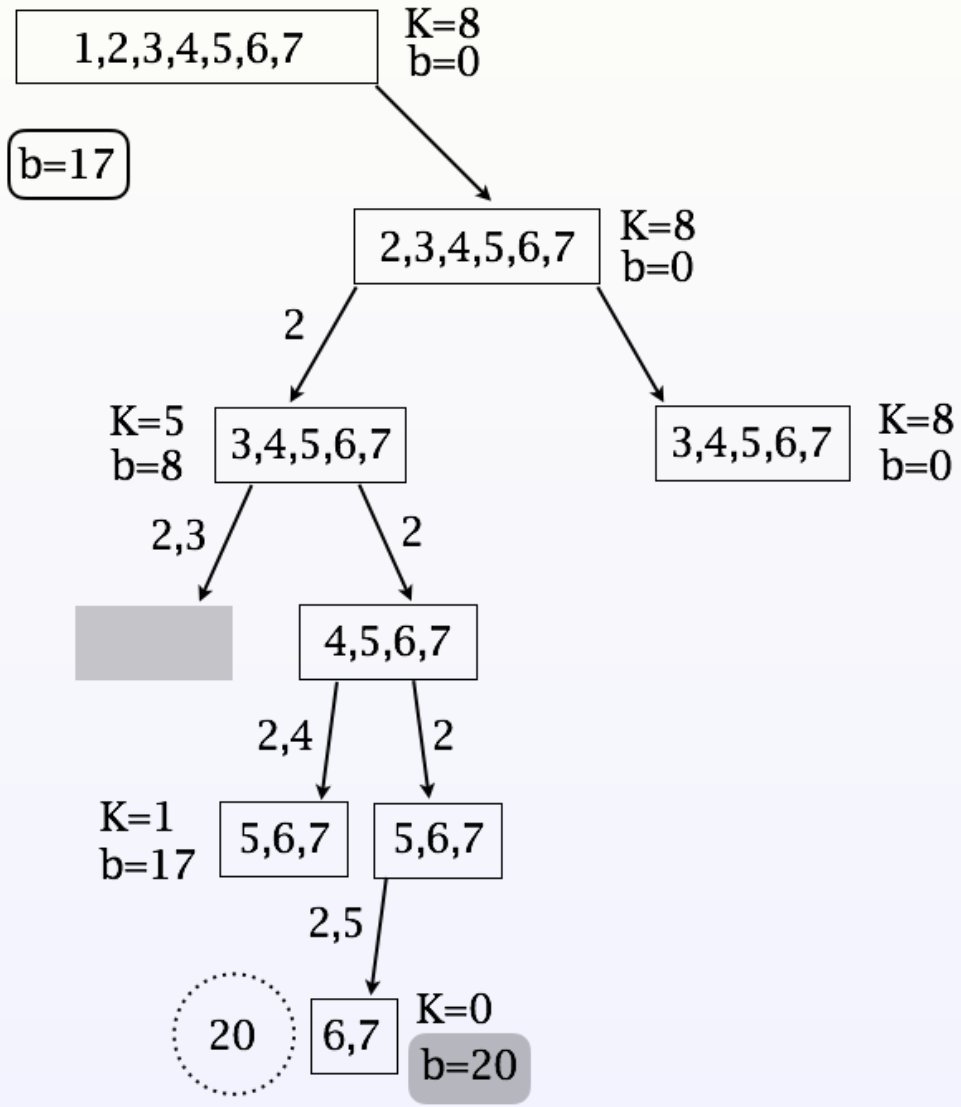
Explorons l'autre branche...



Explorons l'autre branche...

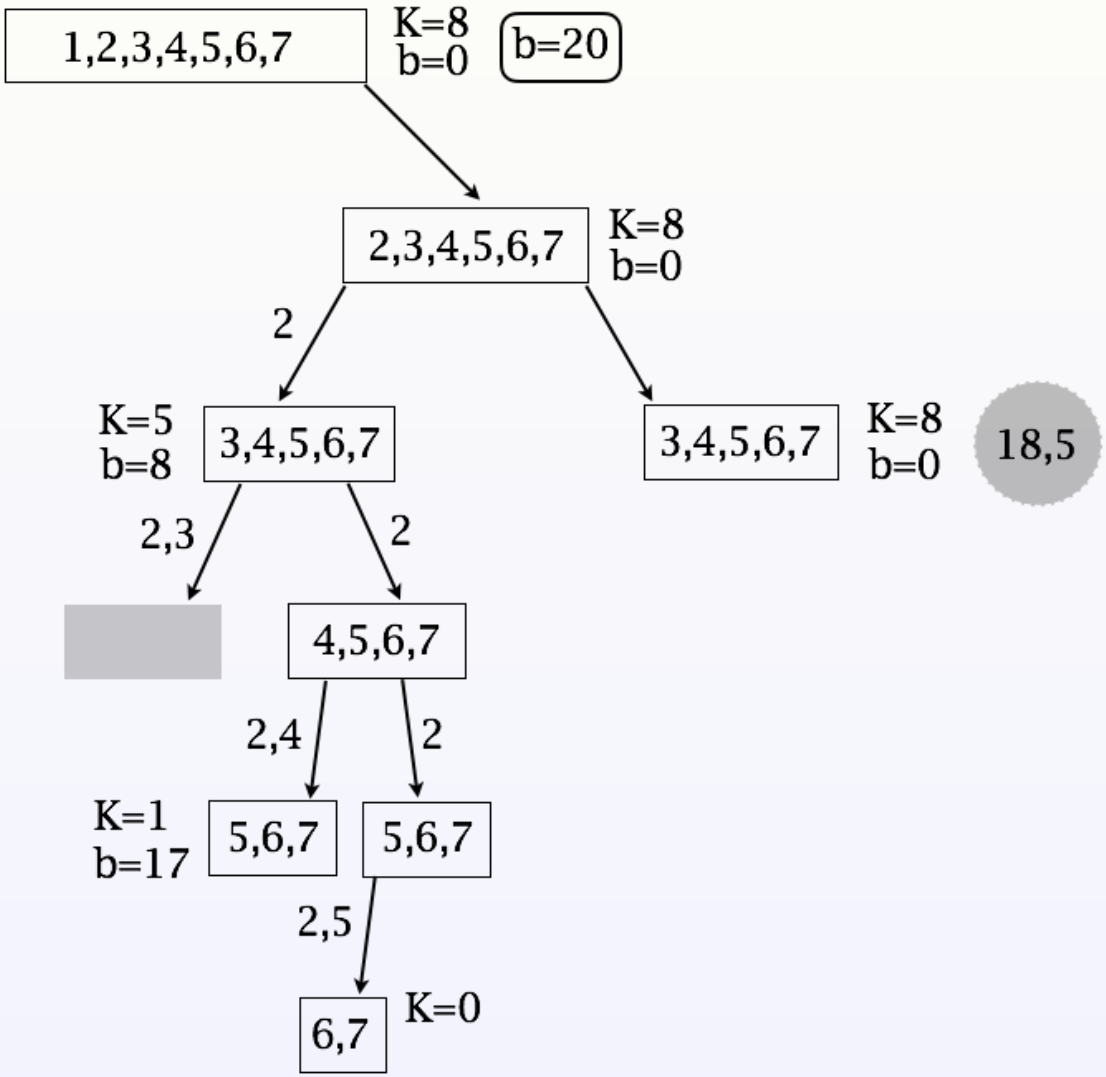


Explorons l'autre branche...



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

Explorons la dernière branche...



- 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[2].

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

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