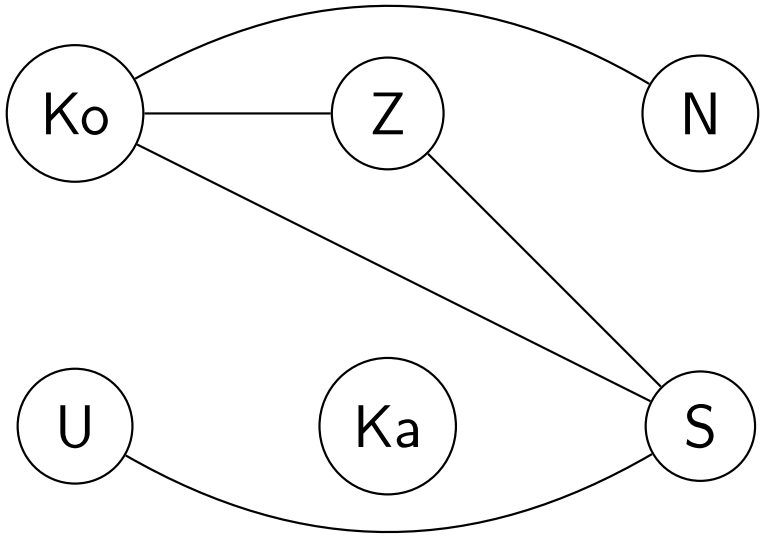
Chapitre 4 : Séparation et Évaluation

ENSIIE - Module de Recherche Opérationnelle

Dimitri Watel (dimitri.watel@ensiie.fr)

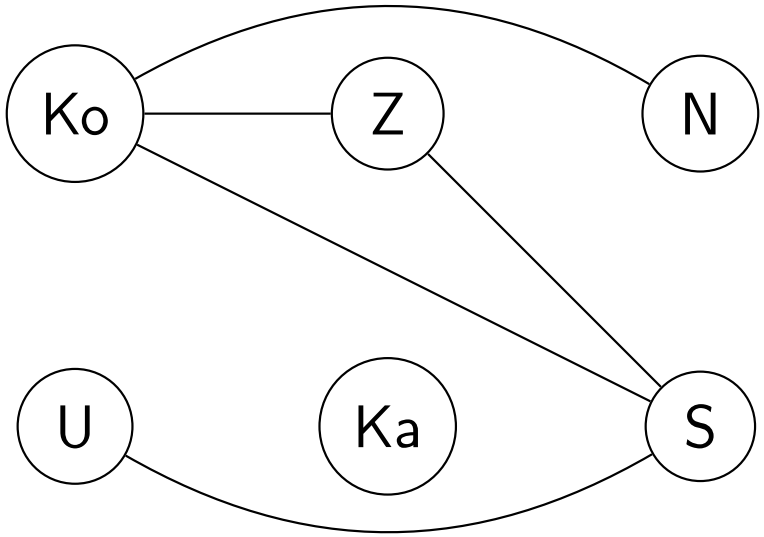
2025

Le problème de Stable maximum



Combien de personnes peut-on mettre dans l'équipage sans conflit ?

Le problème de Stable maximum



Combien de personnes peut-on mettre dans l'équipage sans conflit ?

64 possibilités

Ko	*Z*	*N*	*U*	*Ka*	*S*	Valide ?	Poids
×	×	×	×	×	×	N	–
×	×	×	×	×		N	–
×	×	×	×		×	N	–
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

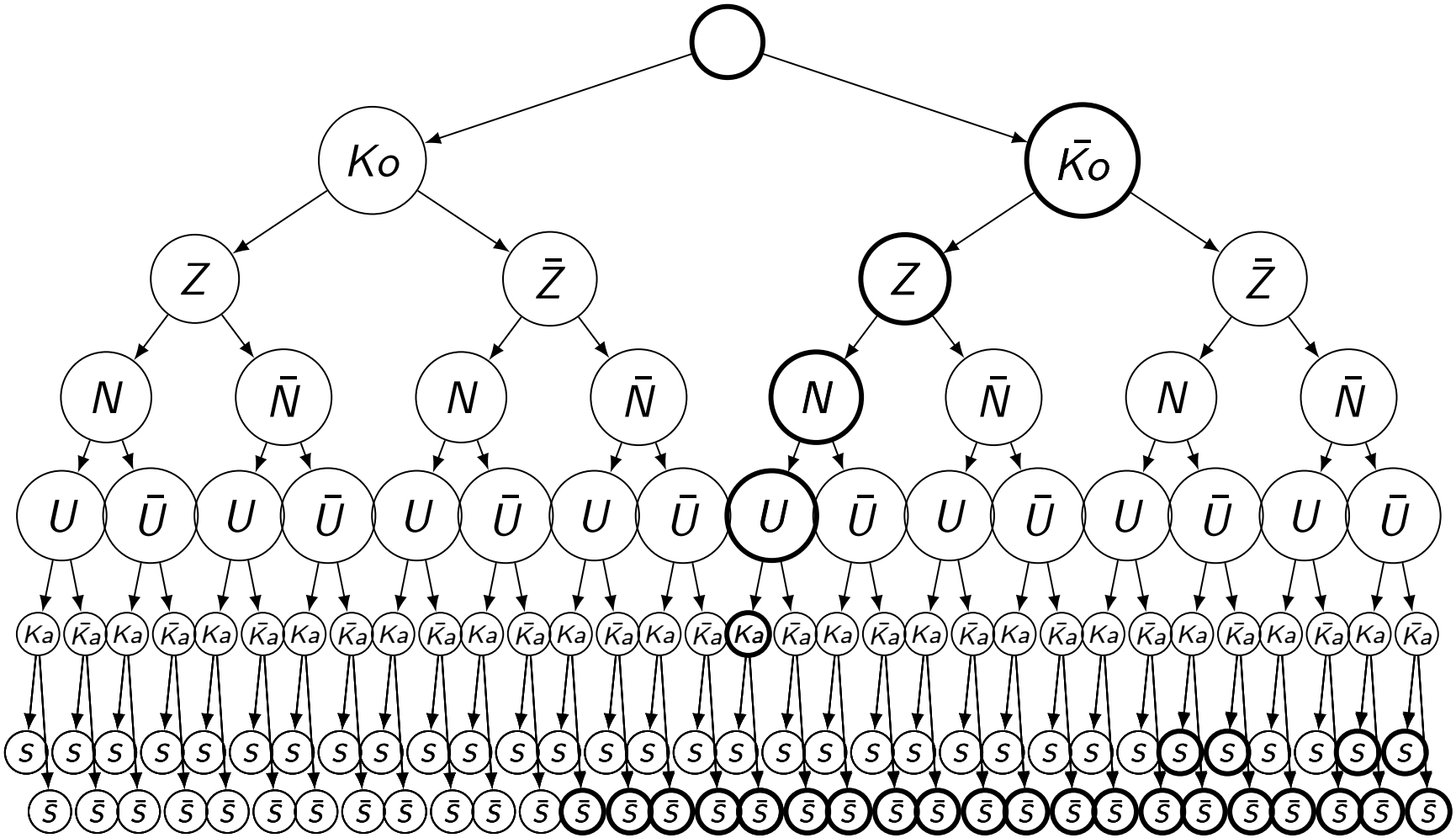
Le problème de Stable maximum

64 possibilités, triées par taille décroissante

Ko	Z	N	U	Ka	S	Valide ?	Poids
×	×	×	×	×	×	N	–
×	×	×	×	×		N	–
×	×	×	×		×	N	–
×	×	×		×	×	N	–
×	×		×	×	×	N	–
×		×	×	×	×	N	–
	×	×	×	×	×	N	–
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	×	×	×	×		V	4

Solution optimale : Z, N, U, Ka

Représentation arborescente : séparation



Représentation arborescente : evaluation

Si une solution contient *Ko*, quelle valeur maximale puis-je espérer atteindre ?

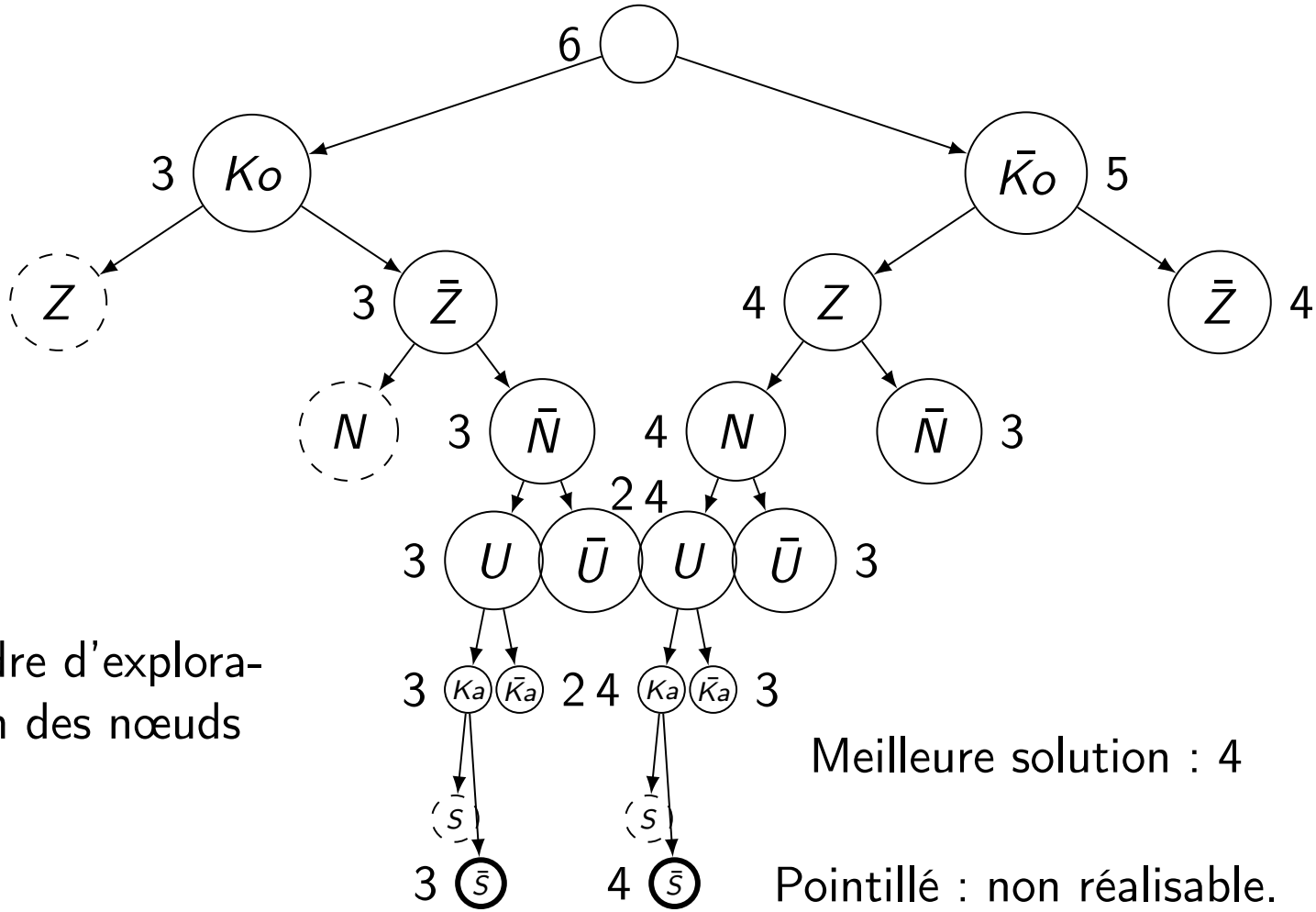
3, car *Ko* empêche *Z*, *N* et *S*.

Si j'ai déjà une solution de valeur 3 ou plus, inutile d'explorer plus loin le cas où *Ko* est dans la solution.

Représentation arborescente : evaluation

Si une solution contient $\bar{Ko}$ et Z, quelle valeur maximale puis-je espérer atteindre ?
4, car Z empêche S et $\bar{Ko}$ empêche Ko.
Si j'ai déjà une solution de valeur 4 ou plus, inutile d'explorer plus loin ce cas.

Exploration en profondeur à gauche

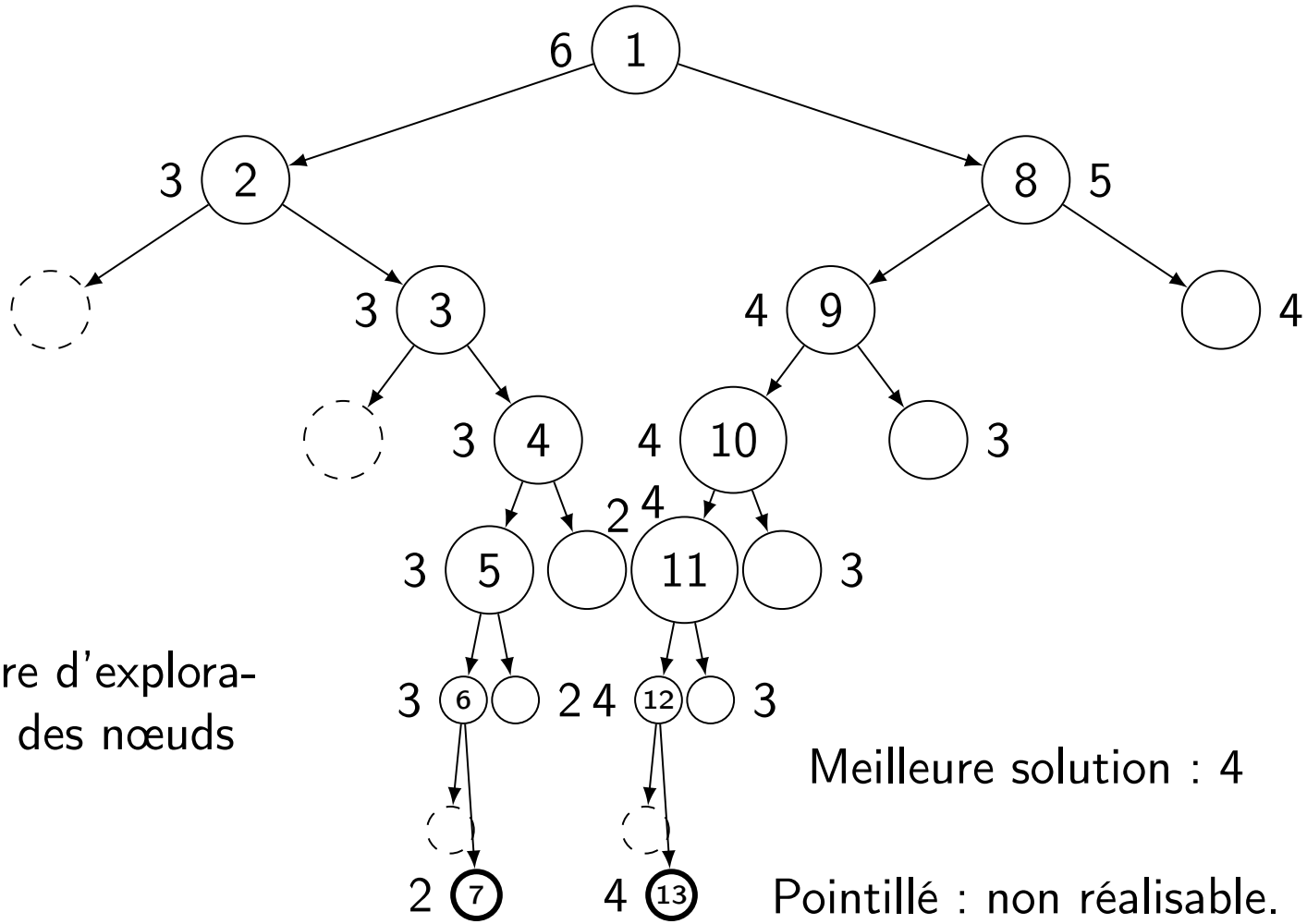


Ordre d'exploration des nœuds

Meilleure solution : 4

Pointillé : non réalisable.

Exploration en profondeur à gauche

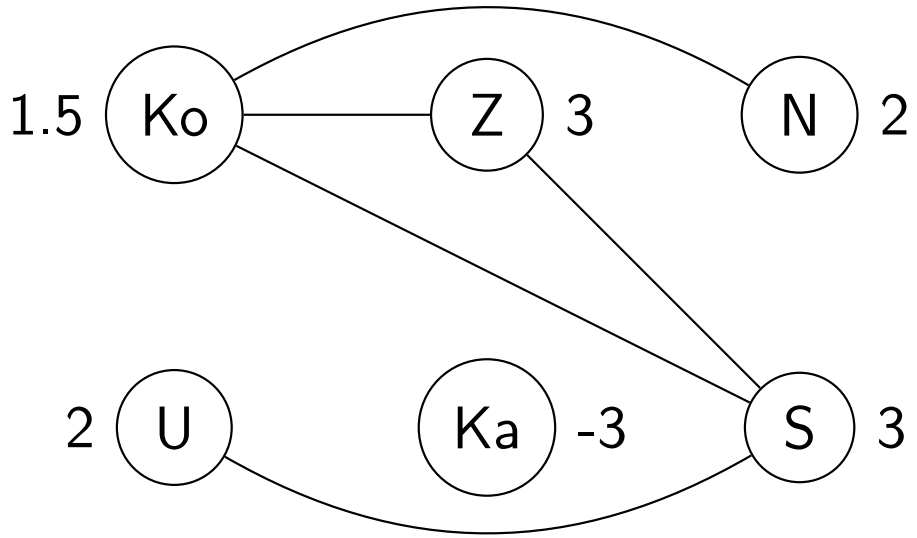


Ordre d'exploration des nœuds

Meilleure solution : 4

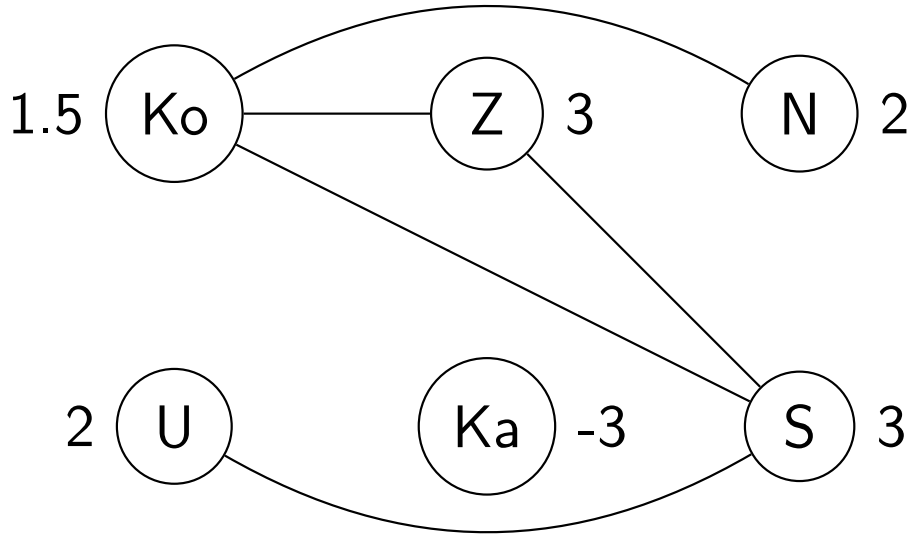
Pointillé : non réalisable.

Sélection de poids maximum



Chaque arête coûte -1, chaque nœud rapporte sa valeur. Quels nœuds choisir pour maximiser le score ?

Sélection de poids maximum



Chaque arête coûte -1, chaque nœud rapporte sa valeur. Quels nœuds choisir pour maximiser le score ? 64 possibilités

Ko	*Z*	*N*	*U*	*Ka*	*S*	Poids
×	×	×	×	×	×	3.5
×	×	×	×	×		3.5
×	×	×	×		×	6.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮

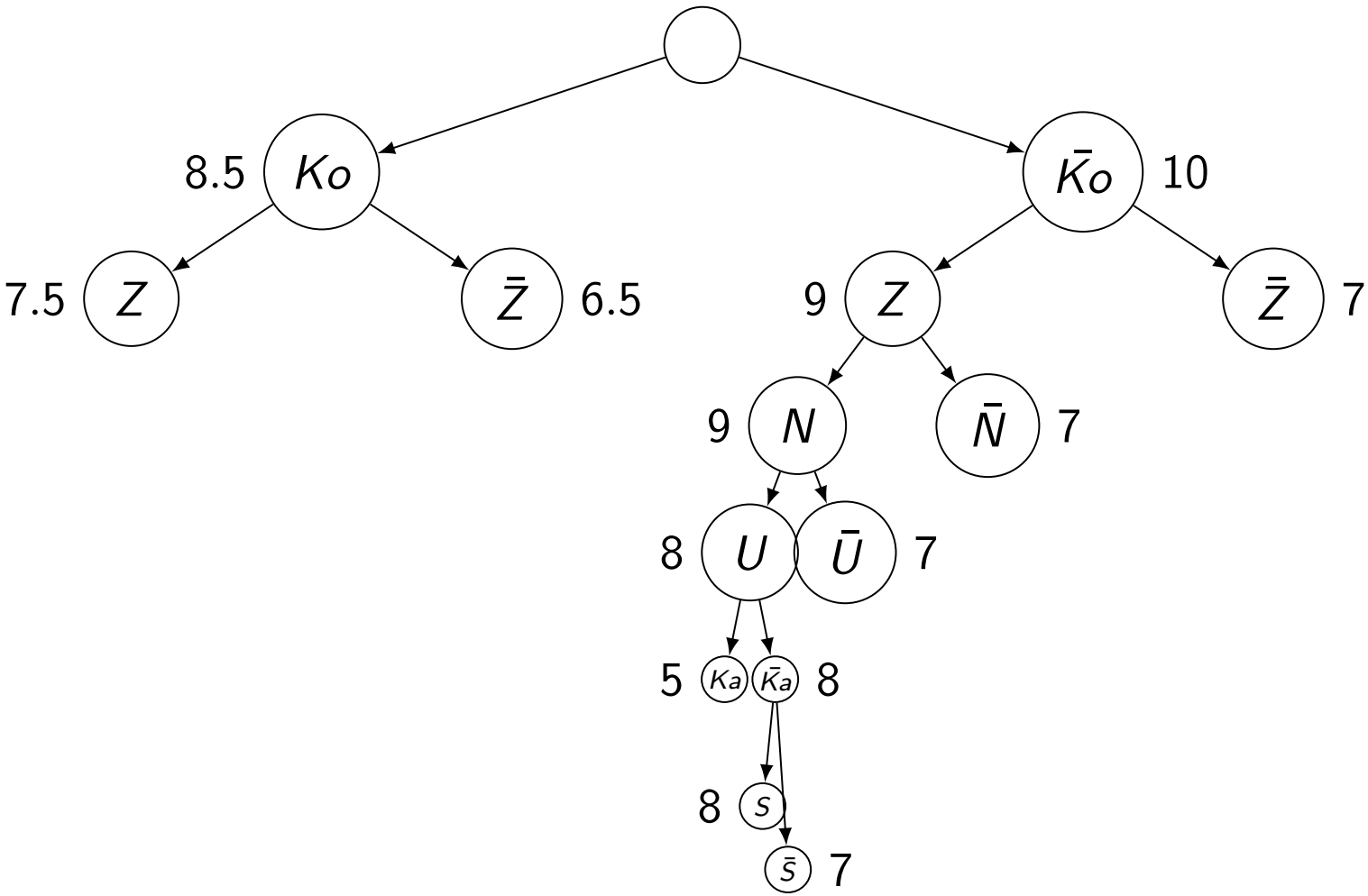
Représentation arborescente : evaluation

Si une solution contient *Ko*, quelle valeur maximale puis-je espérer atteindre ?
8.5, car

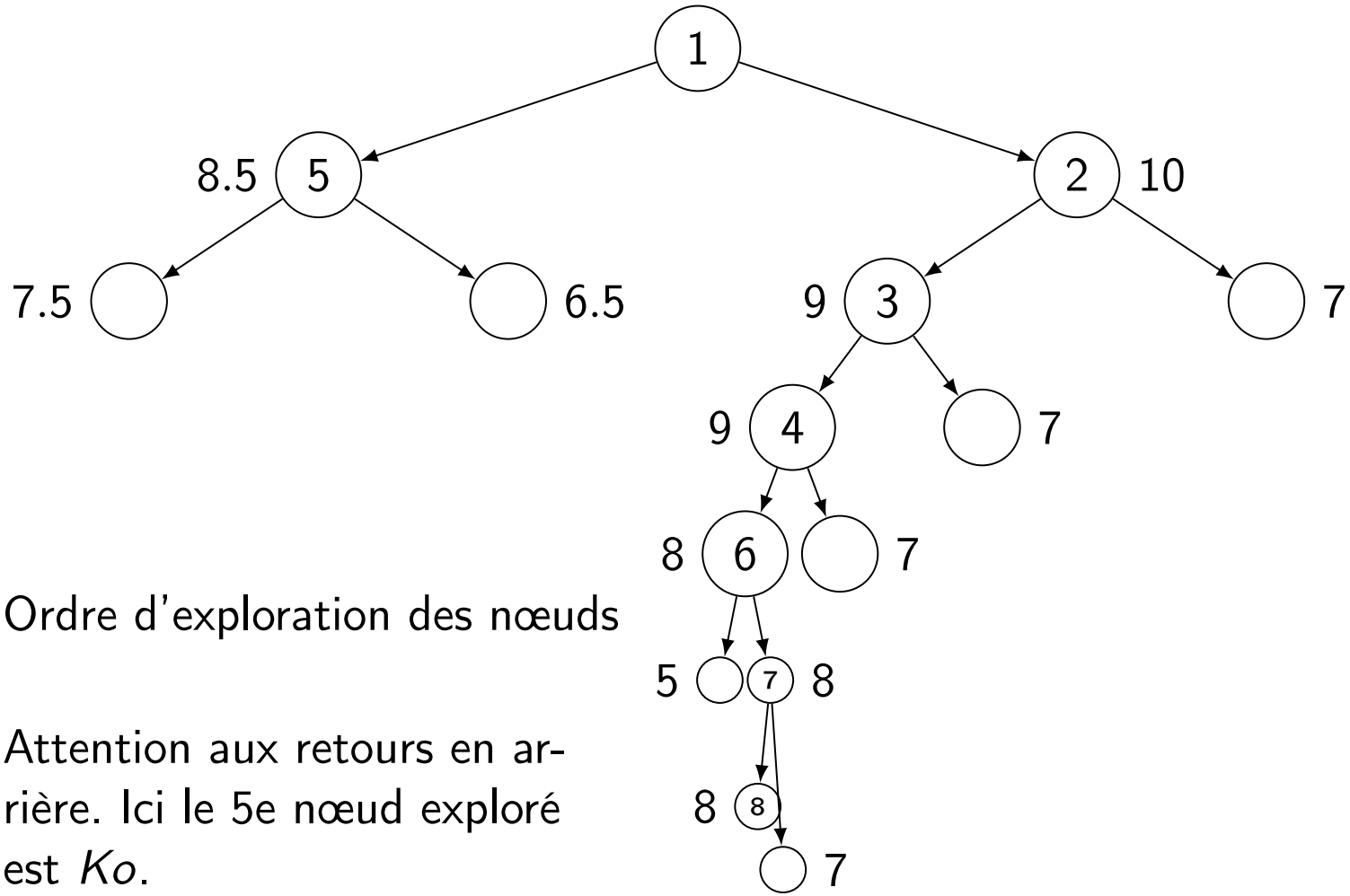
- *Ko* rapporte 1.5
- *Z* rapporte au plus 2 au lieu de 3
- *N* rapporte au plus 1 au lieu de 2
- *U* rapporte au plus 2
- *Ka* rapporte au plus 0
- *S* rapporte au plus 2 au lieu de 3

Si j'ai déjà une solution de valeur 8.5 ou plus, inutile d'explorer plus loin ce cas.

Exploration du meilleur d'abord



Exploration du meilleur d'abord



Ordre d'exploration des nœuds

Attention aux retours en arrière. Ici le 5e nœud exploré est *Ko*.

Autres exemples

Au tableau

Principe

Définition

La méthode par *séparation et évaluation* (ou Branch and bound) est une approche de résolution ; une manière de concevoir un algorithme pour résoudre un problème. Ce n'est pas un algorithme.

Pour résoudre un problème avec la méthode de séparation et d'évaluation, on a besoin de

- couper le problème en sous-problèmes (généralement en fixant une variable) ;
- décider comment explorer l'arbre des sous-problèmes ;
- savoir borner la valeur optimale des sous-problèmes.

Séparation en sous-problèmes

Séparation en sous-problèmes

Soit un problème d'optimisation Π dont l'ensemble des solution est $\xi(\Pi)$. Une séparation $S(\Pi)$ en sous-problèmes est un ensemble de problèmes d'optimisations $(\Pi_1, \Pi_2, \ldots \Pi_p)$ tels que

$$\bigcup_{j=1}^{p} \xi(\Pi_j) = \xi(\Pi)$$

(Exemple au tableau)

Borner la solution optimale d'un sous-problème

Borner la solution optimale d'un sous-problème

Soit un problème de **minimisation** Π. On note $\omega(s)$ la valeur d'une solution réalisable s de $\xi(\Pi)$. Soit s^* une solution optimale de Π (minimisant $\omega(s^*)$).
Une borne $B(\Pi)$ est un nombre **rapide à calculer** et vérifiant

$$B(\Pi) \leq \omega(s^*)$$

Remarque : Si Π est un problème de maximisation,

$$B(\Pi) \geq \omega(s^*)$$

(Exemple au tableau)

Explorer l'arbre des sous-problèmes

Explorer un nœud

Explorer un nœud Π signifie calculer $S(\Pi)$ et $B(\pi)$ pour tout $\pi \in S(\Pi)$.

Explorer l'arbre des sous-problèmes

Exploration en profondeur

On parcours l'arbre en profondeur à gauche (ou à droite) : on explore toujours le nœud non exploré le plus profond et le plus à gauche. L'avantage est de trouver rapidement des solutions réalisables.

Exploration du meilleur

On explore toujours le nœud non exploré ayant la meilleure borne (la plus basse pour un problème de minimisation, la plus haute sinon). On augmente les chances de tomber d'abord sur une solution proche de l'optimal.

On peut, bien entendu, ne pas se limiter à ces méthodes.

Relation avec la programmation dynamique

Algorithme : exploration en profondeur

Algorithm 1 $BB(\Pi, uB)$, $uB = \infty$ par défaut

Entrées: un problème de **minimisation** Π, et un réel $uB \geq \omega(s^*)$
Sorties: Le poids $\omega(s^*)$ d'une solution optimale s^* de Π

1: **Si** Π ne peut être séparé **Alors Renvoyer** Le poids $\omega(s^*)$ d'une solution optimale s^* de Π
2: Explore Π
3: **Pour** $\Pi_j \in S(\Pi)$ tel que $\xi(\Pi_j) \neq \emptyset$ **Faire**
4: $\quad lb \leftarrow B(\Pi_j)$
5: $\quad$ **Si** $lb < ub$ **Alors**
6: $\quad\quad \omega \leftarrow BB(\Pi_j, ub)$
7: $\quad\quad ub \leftarrow \min(ub, \omega)$
8: **Renvoyer** uB

Algorithme : exploration du meilleur

Algorithm 2 $BB(\Pi)$

Entrées: un problème de **minimisation** Π

Sorties: Le poids $\omega(s^*)$ d'une solution optimale s^* de Π

1: $L \leftarrow [\Pi]$; $uB \leftarrow +\infty$
2: **Tant que** $L \neq \emptyset$ **Faire**
3: $\quad \pi \leftarrow \arg\min_{\pi \in L}(B(\pi))$; Remove π from L
4: $\quad$ **Si** $B(\pi) \geq ub$ **Alors**
5: $\quad\quad$ **Break**
6: $\quad$ **Si** π ne peut être séparé **Alors**
7: $\quad\quad ub = \min(ub,$ Poids $\omega(s^*)$ d'une sol opt s^* de $\pi)$
8: $\quad$ **Sinon**
9: $\quad\quad$ Explore Π
10: $\quad\quad$ Add all elements of $S(\Pi)$ to L
11: **Renvoyer** uB