CORRECTION DU CONTRÔLE D'INTELLIGENCE ARTIFICIELLE 13/03/2017

EXERCICE 1 : ALGORITHME A* (9 PT)

1.1 Comme on cherche à **minimiser le nombre total d'actions** réalisées pour arriver à la situation terminale T,

a. chaque action a le même coût : $\boldsymbol{k(u, v) = 1 \quad \forall v \in S(u)}$

b. la fonction $\boldsymbol{g(u)}$ comptabilise le nombre d'actions déjà réalisées depuis u_0 pour arriver en u.

$$g(u_i)= \sum_{j=1}^{j=i} k(u_{j-1}, u_j)$$

1.2 Propriétés de h

a. $h(T)=0$ (aucun bloc périphérique n'est mal placé) donc **h est coïncidente**

b. Soit une action a qui permet de passer de u à v ; entre u et v on ne peut placer correctement qu'un bloc au plus, puisque le nouveau bloc central n'est pas comptabilisé (seule compte le nouveau bloc périphérique).

On étudie 3 cas de figure pour la variation de h entre 2 situations successives u et v

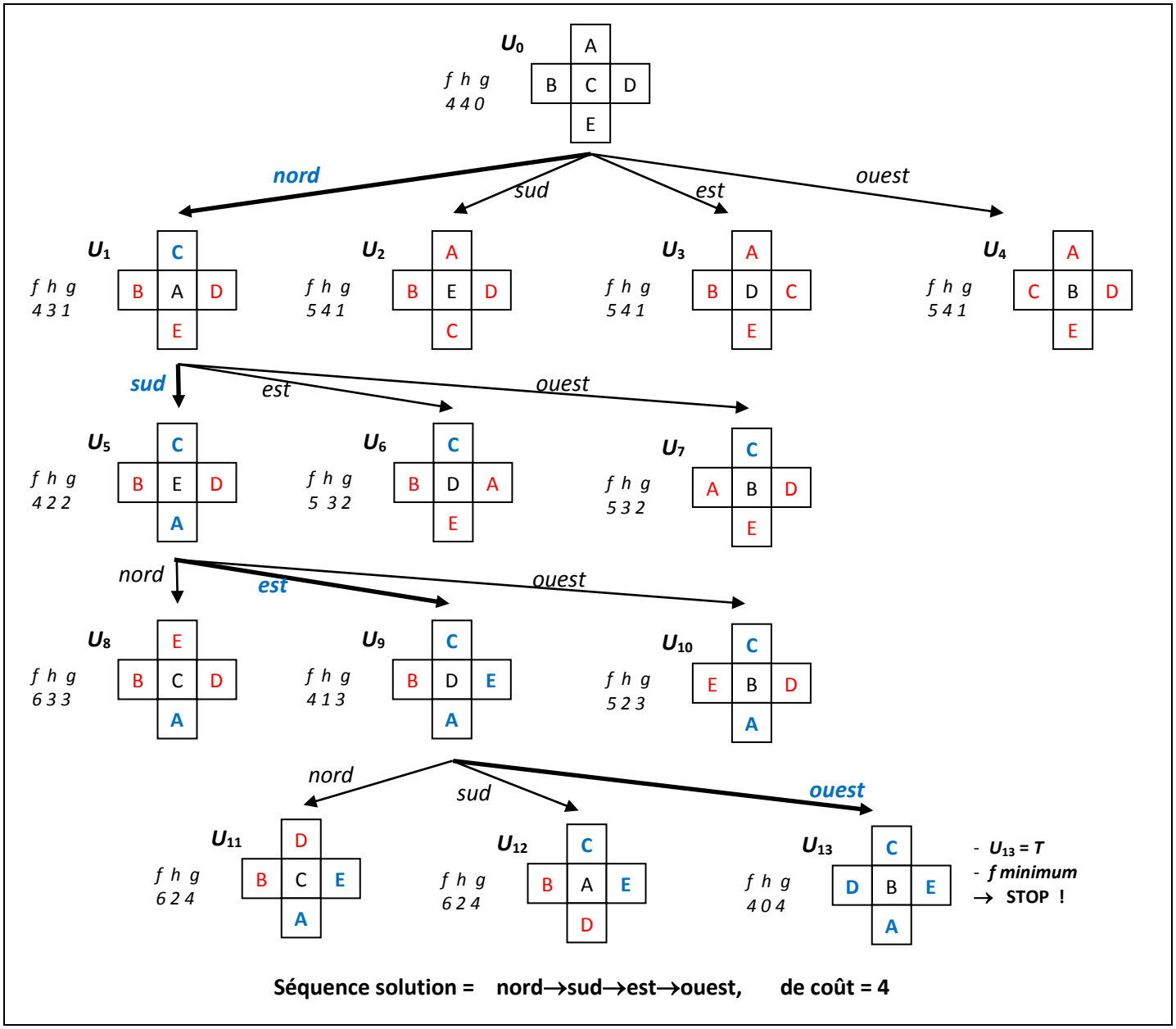
- Cas d'une augmentation $h(v)>h(u)$
  La seule façon d'augmenter h est de remplacer un bloc périphérique mal placé par le bon : $\boldsymbol{h(v)-h(u)=1}$
- Cas de la stabilité : $h(v)=h(u)$
  Dans ce cas un bloc mal placé est remplacé par un nouveau bloc également mal placé : $\boldsymbol{h(v)-h(u)=0}$
- Cas d'une diminution : $h(v)<h(u)$
  La seule façon de diminuer h est de remplacer un bloc périphérique bien placé par un autre : $\boldsymbol{h(v)-h(u)=1}$

Dans les 3 cas la variation de h est bornée : $\boldsymbol{|h(v)-h(u)| \leq 1}$.
Or on sait que $\underline{k}(u, v) = 1$; on en conclue donc que : $\boldsymbol{h(v)-h(u) \leq k(u,v)}$, c-à-d que **$h$ est monotone**.

c. D'après a. et b., et le théorème démontré en cours (*si une heuristique est à la fois coïncidente et monotone alors elle est minorante*), on a démontré que : **h est minorante**.

On peut également montrer que $\boldsymbol{h}$ est minorante directement :

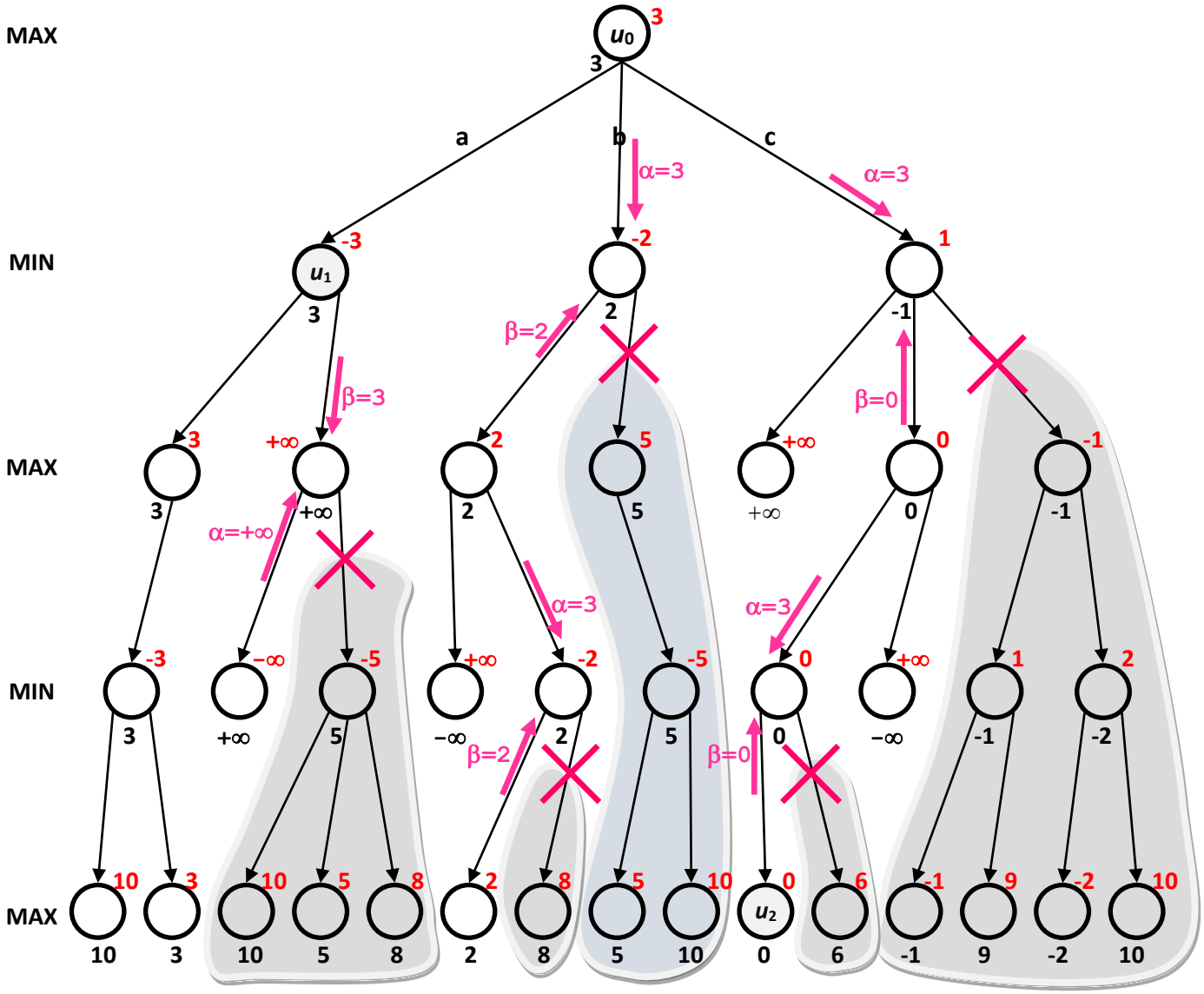
- **pour chaque bloc périphérique mal placé**, il faut 1 action (parfois 2) pour le faire parvenir à sa bonne place => **il faut au minimum une action**.
- **au cours d'une seule action élémentaire, on ne peut placer correctement qu'un seul bloc périphérique à la fois** (puisque pour cela on effectue une permutation avec le bloc situé au centre et non avec un autre bloc périphérique mal placé).
- Donc pour n blocs périphériques mal placés, il faut réaliser au minimum n actions pour les placer correctement. Le nombre de blocs périphériques mal placés est bien un minorant du nombre d'actions à effectuer pour parvenir à la solution. **L'heuristique est donc minorante**.



Séquence solution = nord→sud→est→ouest, de coût = 4

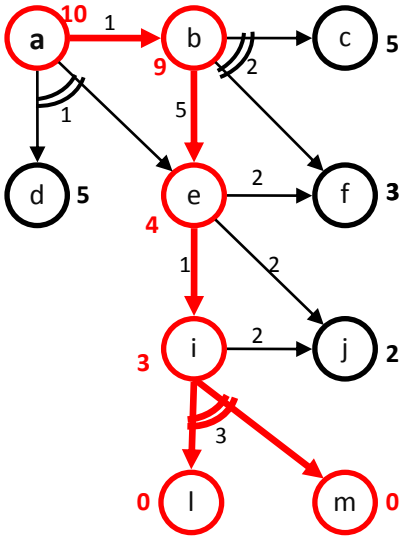
2.1 Il faut jouer a (voir ci-dessous l'évaluation minmax.
2.2 En rouge , l'évaluation negamax.
2.3 Les coupures alphabeta sont représentées (X) et les régions en grisé sont inexplorées

Annexe 2 : Arbre minmax – négamax – alpha-beta

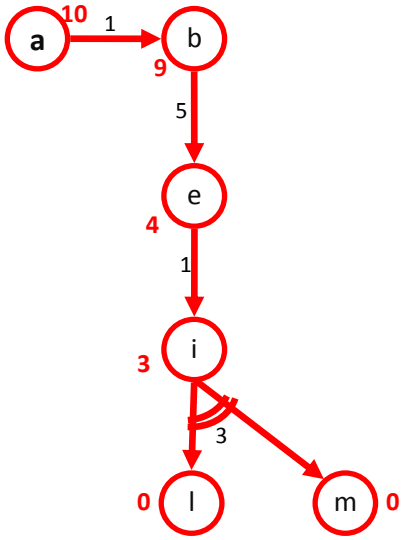


2.4 L'égalité des situations u_2=u_1 change-t-elle le résultat ?
Non. Explication :
La situation ***u*₁ a été développée sur 3 coups** alors que ***u*₂ n'a pas été développée** car elle est située à la profondeur maximale. En principe, pour une même situation, la valeur minmax établie par une exploration est plus fiable qu'une estimation heuristique. Mais ici le joueur n'est pas le même : les situations issues de ***u*₁** envisagent des coups pour le 2ème joueur alors que l'arbre qui serait développé sous ***u*₂** envisagerait des coups possibles pour le 1er joueur. On ne peut pas remplacer $h(\boldsymbol{u_2})$ par la valeur minmax($\boldsymbol{u_1}$) ; le résultat n'est pas changé : **a reste le meilleur coup en *u*₀**.
Même si le joueur avait été le même dans les 2 situations, la valeur 3 serait remontée de 3 niveaux et aurait autorisé la dernière coupure alphabeta (située à droite) : α=3, β=3, donc $\alpha \geq \beta$; comme ***u*₁** est un nœud de type max, même dans ce cas, ***a*** demeurerait le meilleur coup en ***u*₀**.

U	a	b	C	d	e	f	g	h	i	j	k	l	m	n
***h*(*U*)**	10	9	5	5	4	3	0	4	3	2	0	0	0	0



Graphe développé par AO*



Sous-graphe solution

Coût du sous-graphe solution = 10