## EXERCICE 1 : ALGORITHME A* (10 PT)

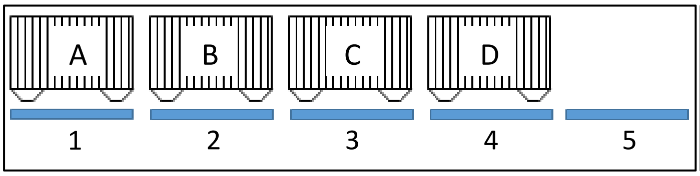


*figure 1 : la situation initiale* $u_0$

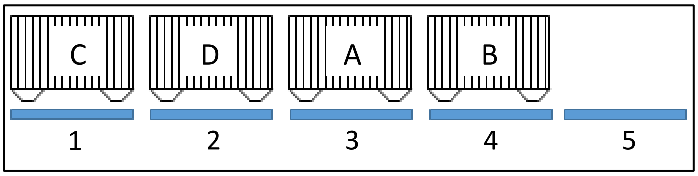


*figure 2 : la situation finale* $t$

1.2 Rappel (vu en cours) : par définition la fonction $g$ évalue la **somme des coûts des actions déjà effectuées depuis la situation initiale $u_0$ pour parvenir à la situation courante**. Soit $(u_0, u_1, u_2, \dots, u_{i-1}, u_i)$ le chemin suivi pour aller de $u_0$ à $u_i$ après $i$ actions ; on peut définir $g(u_i)$
- de façon récursive :

$$g(u_0) = 0 \quad \text{et} \quad \forall i > 0 \; g(u_i) = g(u_{i-1}) + k(u_{i-1}, u_i)$$

- ou bien de façon directe :

$$g(u_0) = 0 \quad \text{et} \quad \forall i > 0 \; g(u_i) = \sum_{j=1}^{j=i} k(u_{j-1}, u_j)$$

Dans ce problème, le coût d'une action(ou *transfert*), $k(u_{j-1}, u_j)$, est homogène à une quantité de travail, car égal au produit du poids du container déplacé, $c_j$, par la distance parcourue :

$$k(u_{j-1}, u_j) = poids(c_j).\Delta(c_j, u_{j-1}, u_j)$$

$\Delta(c_j, u_{j-1}, u_j)$ est la distance parcourue par le container $c_j$, déplacé lors du passage de $u_{j-1}$ à $u_j$ :

$$\Delta(c_j, u_{j-1}, u_j) = |pos(c_j, u_j) - pos(c_j, u_{j-1})|$$

avec $pos(c, u)$ = position de $c$ dans la situation $u$ ($pos(c, u) \in 1..5$)

1.3 Choix de la fonction $h$

Par définition, la fonction $h$ est une estimation de la **somme des coûts des transferts restant à effectuer** pour aller de situation courante $u$ à la situation terminale $t$. Cette fonction doit être choisie avec de bonnes propriétés pour que *A** soit efficace et optimal : il faudra démontrer que $h$ est coïncidente, monotone et minorante.
Le coût total d'une solution est définie par :

$$f(u) = g(u) + h(u)$$

Intervenant dans une somme, il est nécessaire que les fonctions $g$ et $h$ soient des **quantités homogènes**, exprimées dans la même unité (ex : nb d'actions, somme dépensée totale, travail effectué total, temps passé total …). Comme $g$ mesure une quantité homogène à un travail (produit d'une force par une distance), il faut aussi que $h$ soit homogène à un travail. On peut par exemple mesurer le coût pour déplacer chaque container depuis sa position actuelle vers sa position désirée, comme s'il était seul sur le quai de chargement. On fait ensuite la somme de ces coûts sur l'ensemble des containers :

$$h(u) = \sum_{c \in \{A,B,C,D\}} poids(c).\Delta(c, u, t)$$

*Proposition 1.3a* **La fonction $h$ est coïncidente**

*Démonstration (triviale)*

Par définition, en $t$ les 4 containers sont bien placés. Donc $\Delta(c, t, t) = 0 \; \forall c$

$$h(t) = \sum_{c \in \{A,B,C,D\}} poids(c).\Delta(c, t, t) = 0$$

*Proposition 1.3b* **La fonction $h$ est monotone**

*Démonstration*

Soit $u$ et $v$ deux situations successives entre lesquelles un seul container a changé de place.

$$h(u) = \sum_{c \in \{A,B,C,D\}} poids(c).\Delta(c,u,t)$$

$$h(v) = \sum_{c \in \{A,B,C,D\}} poids(c).\Delta(c,v,t)$$

d'où

$$h(u) - h(v) = \sum_{c \in \{A,B,C,D\}} poids(c).\left(\Delta(c,u,t) - \Delta(c,v,t)\right)$$

Comme $v$ est une situation qui résulte de $u$ après déplacement **d'un seul container** parmi {*A*, *B*, *C*, *D*}, trois termes sur quatre sont nuls ; soit $x$ le seul container qui a été déplacé entre $u$ et $v$ ; $h(u) - h(v)$ ne dépend que du déplacement de $x$ :

$$h(u) - h(v) = poids(x).\left(\Delta(x,u,t) - \Delta(x,v,t)\right)$$

Il faut donc démontrer que :

$$h(u) - h(v) \leq k(u,v)$$

⇔

$$poids(x).\left(\Delta(x,u,t) - \Delta(x,v,t)\right) \leq poids(x).\Delta(x,u,v)$$

⇔

$$\Delta(x,u,t) - \Delta(x,v,t) \leq \Delta(x,u,v)$$

⇔

$$|pos(x,u) - pos(x,t)| - |pos(x,v) - pos(x,t)| \leq |pos(x,u) - pos(x,v)|$$

En ajoutant le terme nul $-pos(x,t) + pos(x,t)$ au second membre

⇔

$$|pos(x,u) - pos(x,t)| - |pos(x,v) - pos(x,t)| \leq |(pos(x,u) - pos(x,t)) + (pos(x,t) - pos(x,v))|$$

En posant **:** $a = pos(x,u) - pos(x,t)$ et $b = pos(x,t) - pos(x,v)$

- 

$$|a| - |-b| \leq |a + b|$$

Comme $|-b| = |b|$

$$|a| - |b| \leq |a + b|$$

Ceci est facile à démontrer car :

$$|a| - |b| \leq \big||a| - |b|\big|$$

Et d'après l'inégalité triangulaire de la valeur absolue :

$$\big||a| - |b|\big| \leq |a + b|$$

Donc

$$|a| - |b| \leq |a + b|$$

CQFD Donc est $h(u)$ monotone.

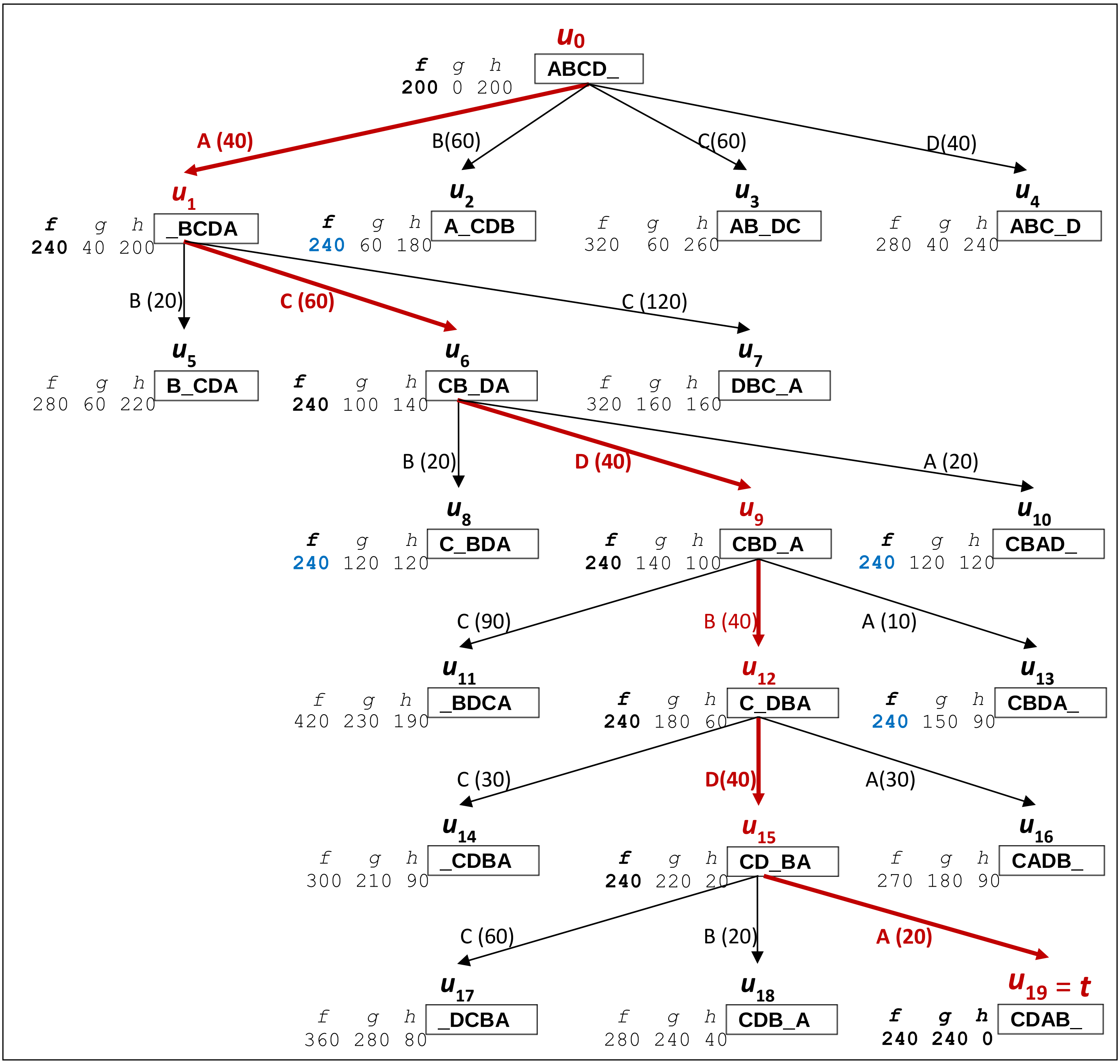
*Proposition 1.3c* $h(u)$ est minorante.

*Démonstration* (triviale) : d'après la propriété donnée en cours :

$h$ est coïncidente et $h$ est monotone $\Rightarrow$ $h$ est minorante.

*Autre démonstration (informelle)*

Pour résoudre ce problème il faut au moins un transfert par container $c$ mal placé (car un seul transfert ne déplace qu'un container à la fois). Si $c$ était seul sur le quai, alors un transfert direct coûterait $poids(c).\Delta(c,u,t)$. Mais en réalité, la place finale de $c$ n'est pas nécessairement libre dans la situation $u$ ; si elle ne l'est pas, il faudra un transfert supplémentaire et donc le coût pour bien placer $c$ augmentera. Donc $h(u)$ est minorante.

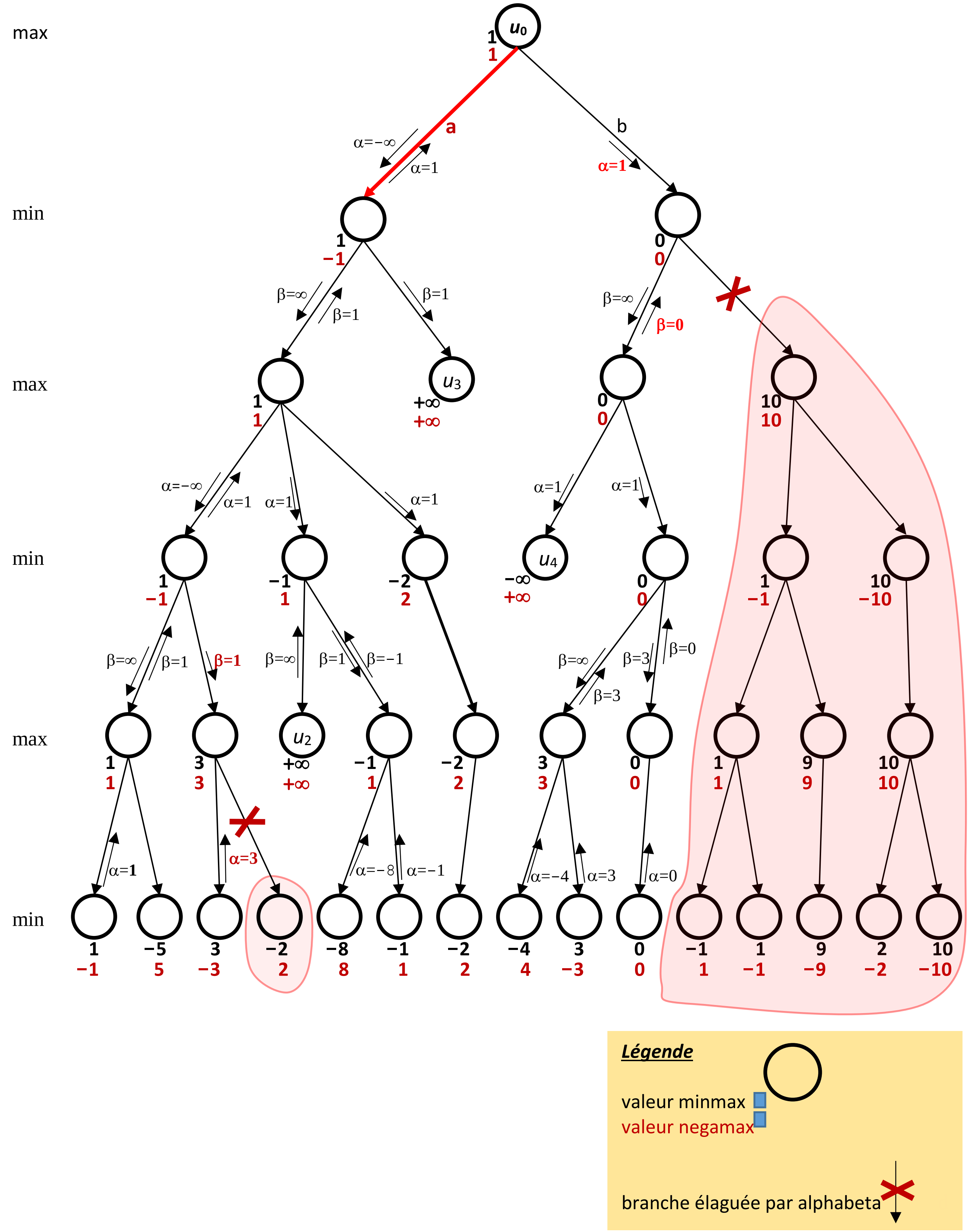



La solution trouvée est (en termes de containers déplacés) : **A→C→D→B→D→A et coûte 240**

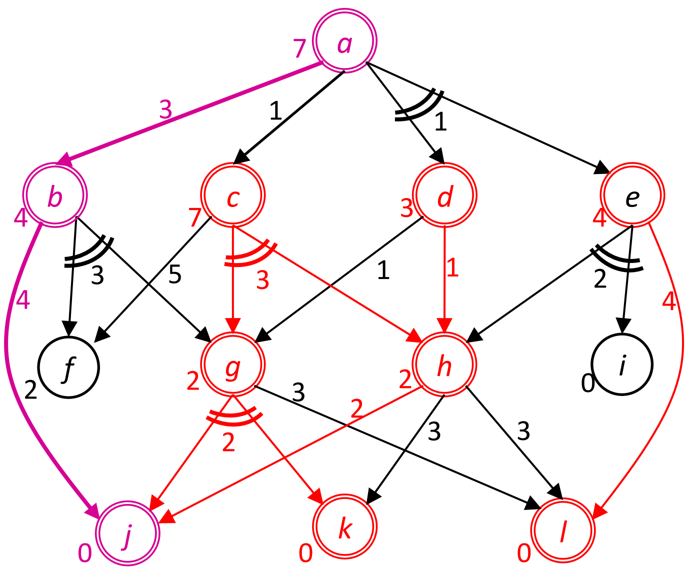
*Remarques*

- Plusieurs nœuds non développés ($u_2$, $u_8$, $u_{10}$, $u_{13}$) ont une valeur $f$=240 ; il est très probable que ce problème ait plusieurs solutions optimales (de coût total équivalent).
- $u_0$ est le seul état où $f(u) < f^*(u)$. Dans tous les états suivants de la solution, $h$ et $f$ sont des estimations parfaites. Ceci est lié au fait que l'utilisation de l'emplacement 5 dans lequel on est obligé initialement de placer un container fait perdre 1 coup (car l'emplacement 5 n'est pas utilisé dans la situation finale), mais ce coup est nécessaire pour libérer un des 4 autres emplacements utiles. Ensuite, à partir de $u_1$, on peut toujours faire un transfert utile, soit en occupant l'emplacement libre courant par le container adéquat, soit en l'utilisant pour rapprocher un des containers mal placés de sa position finale.
- Contrairement au critère du nombre minimal d'actions, le critère qui a été choisi dans ce problème (somme pondérée minimale des distances de déplacement) permet d'envisager le placement correct d'un container en plusieurs fois ; par exemple le container D se déplace de 4 vers 2 utilise 2 transferts, de 4 en 3, puis plus tard de 3 en 2.

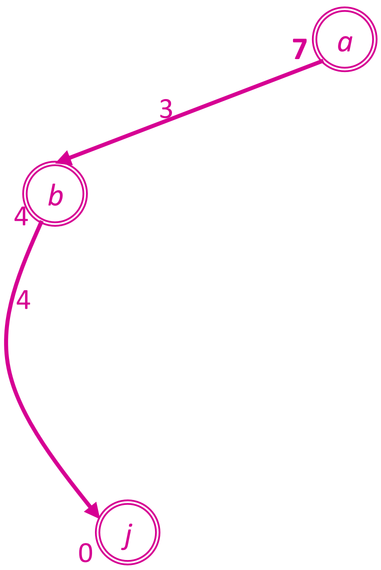
## EXERCICE 2 : ALGORITHME MINMAX – NEGAMAX – ALPHA-BETA



## EXERCICE 3 : ALGORITHME AO*




Nœuds développés et coût associé en fin d'algorithme




Sous-graphe solution