Chapitre 2.

Algorithmes de recherche pour les jeux

2.1 Introduction

Il est possible de représenter le déroulement d'un jeu (comme exemple le jeu d'échecs), par un arbre où les nœuds correspondent aux différentes configurations du jeu (échiquier).
Par alternance, les nœuds d'un même niveau sont associés à un joueur. Les nœuds du niveau suivant sont alors associés au second joueur (voir figure 1).

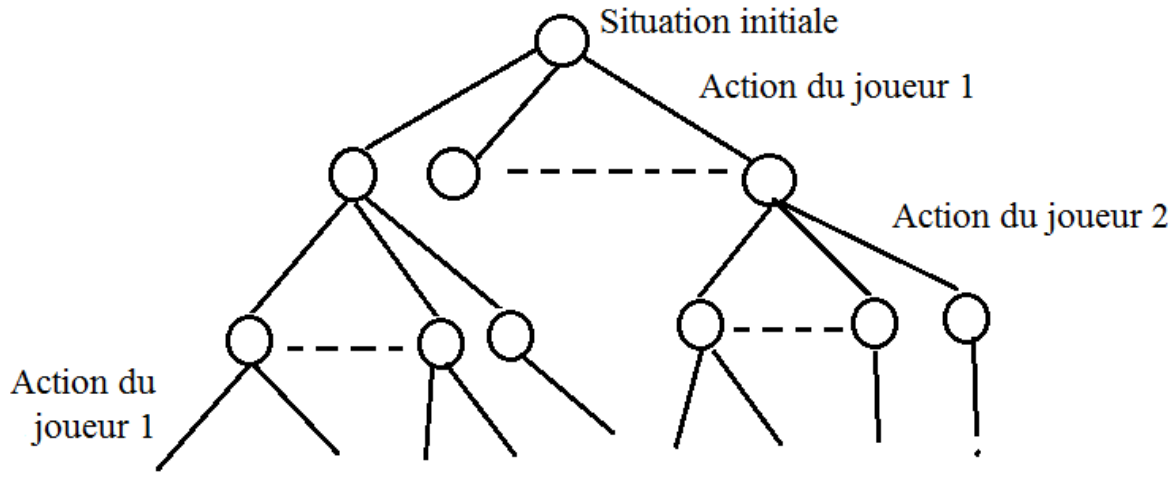


Figure 1. Représentation d'un jeu par un arbre

Pour chacun des joueurs, s'il veut choisir une action qui lui permettra de gagner, il doit explorer tout l'arbre (au maximum). Pour le jeu d'échecs, et pour une profondeur de 100, le nombre de nœuds visités, sachant que le nombre d'actions possibles à chaque étape est de 16, est de 16^{100}.
Il est donc impératif d'appliquer un algorithme intelligent pour espérer trouver une stratégie gagnante dans un temps raisonnable.

2.2 Procédure MinMax

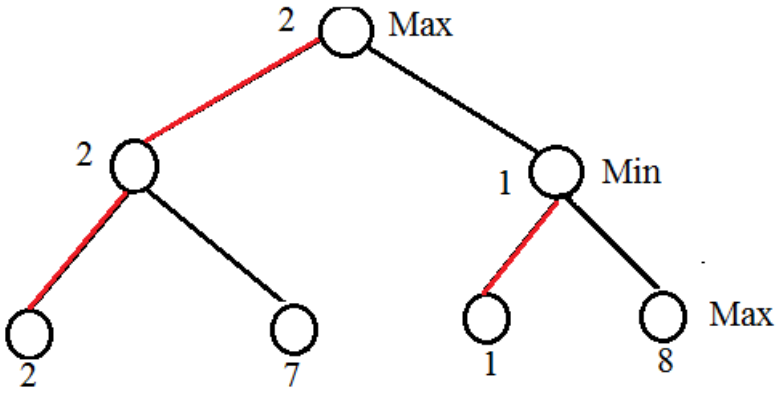


Figure 2. Exemple d'arbre de jeu avec Min Max

2 joueurs : J1 (jetons de couleur blanche), J2 (de couleur Noir).
J1 va récupérer les jetons N, J2 …. B
Coût du jeu : Gain : fonction qui va Compter les jetons de couleur Noir.
J1 commence : maximiser le nombre de jetons noir (récupérés par J1)
J2 minimiser le nombre de jetons Noir (récupérés par J1)

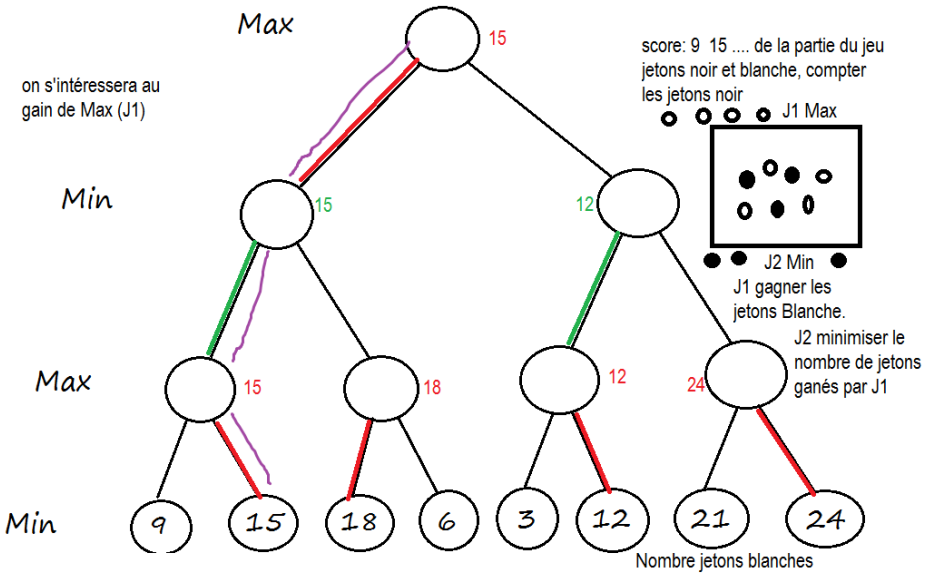
On parlera de deux joueurs : Max Min.
Qui va commencer le jeu ?
Peu importe : Max Min (J1 J2)
Max, Min : Notation
Un joueur va maximiser un gain (minimiser celui de l'adversaire) et l'autre va le minimiser (maximiser le sien)

On peut représenter deux joueurs comme étant Max et Min où chacun va essayer d'optimiser le gain qui est donné par rapport au joueur Max.
Max va choisir les actions qui maximisent son gain alors que le joueur Min va choisir les actions qui permettront de minimiser le gain de Max.
Pour choisir l'action à opérer, Max doit explorer l'arbre est descendre jusqu'aux feuilles pour trouver quel chemin mènera au gain maximum en faisant l'hypothèse que Min tentera de minimiser ce gain à chacune de ses actions.

Cette façon d'opérer est appelée procédure MinMax et donc consiste à :

- Si le nœud est une feuille alors associer la valeur du nœud au gain du joueur se trouvant à ce niveau (Max ou Min).
- Si le nœud considéré est un nœud Max, alors appliquer la procédure MinMax à ses successeurs et affecter à ce nœud un gain égal au maximum des valeurs retournées par les nœuds fils.
- Si le nœud considéré est un nœud Min, alors appliquer la procédure MinMax à ses successeurs et affecter à ce nœud un gain égal au minimum des valeurs retournées par les nœuds fils.

L'écriture algorithmique est donnée comme suit :
Pour un nœud Max, la procédure suivante permet de retourner le max des valeurs de ses successeurs.



int Max(n)
{
-Déterminer les successeurs de n, soient n_1, n_2, .., n_d
// On va appliquer les actions du possibles, et produire un nouvel état pour chque action ce qui engendre l'ensemble des successeurs.
If(d==0) then return V(n) valeur fournie par l'heuristique choisie pour le nœud feuille
Else maximum=-∞
For(i=1 to d)
{
t=Min(n_i) // appel de la fonction min pour chacun des succ
If(t> maximum) then maximum =t;
}
return m
}

Pour un nœud Min, la procédure suivante permet de retourner le min des valeurs de ses successeurs.

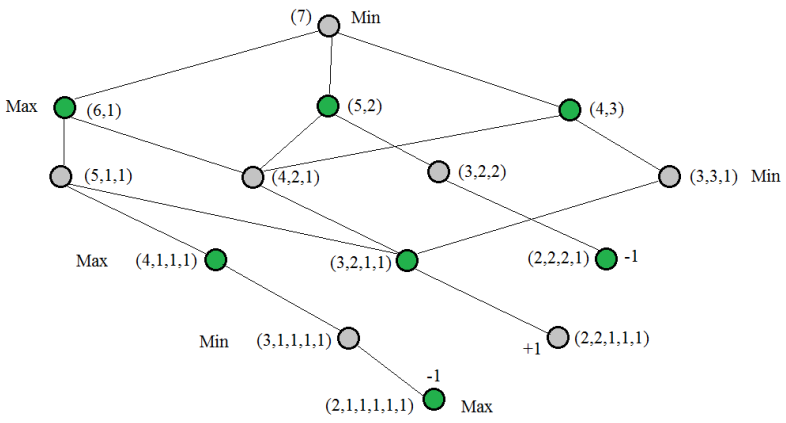
Int Min(n)
{
-Déterminer les successeurs de n, soient n_1, n_2, .., n_d
If(d==0) Then return V(n)/ valeur fournie par l'heuristique choisie pour le nœud feuille
Else minimum=+∞
For(i=1 to d)

```
                {
                t=Max(n_i) // appel à max pour tous les succ
                If(t< minimum) then minimum =t;
                }
return minimum
}
```

Exemple 1 : **Jeu de Grundy** :

Il s'agit de partager une pile de jetons de deux piles de hauteurs différentes. Le joueur qui est dans l'incapacité de jouer perd la partie.
Appelons les deux joueurs Max et Min, et supposons que Min joue en premier.
Le gain +1 est associé au nœud si Max gagne et -1 s'il perd.
Donnez l'arbre de jeu et donnez la stratégie pour que Min (premier joueur) gagne.
(Voir la solution sur la figure 2).

Exemple 2 : **Jeu de Fan-Tan** :
Considérons le jeu Fan-Tan à deux joueurs. On dispose de deux rangées d'allumettes contenant 1 et 3 allumettes.
Chaque joueur peut prendre d'une rangée autant d'allumettes qu'il le souhaite à la fois. Le joueur qui prend en dernier gagne.
Donnez l'arbre de jeu et donnez la stratégie pour que Max (premier joueur) gagne.
(Voir la solution sur la figure 3)



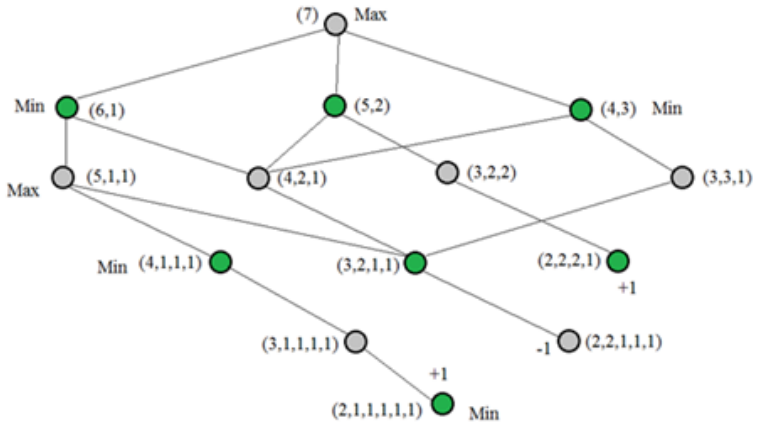


Figure 2. Arbre de jeu pour un état initial (Min ou Min, 7 jetons)

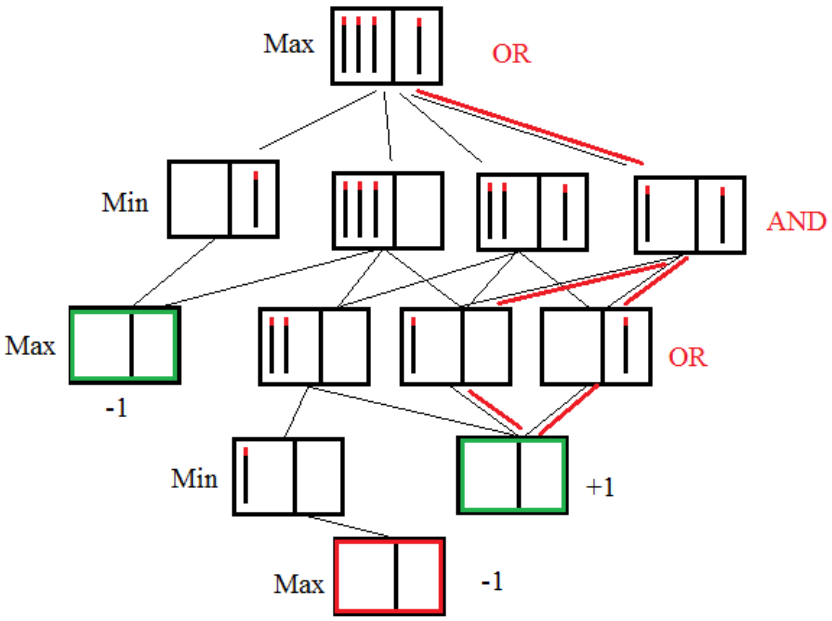


Figure 3. Arbre de jeu

2.3 Procédure NEGAMAX

Au lieu d'utiliser ces deux fonctions, on peut utiliser une seule NEGAMAX qui traite tous les nœuds de la même façon, on sélectionne toujours le max des opposés. Les valeurs des nœuds terminaux correspondants à Min sont initialement inversées comme illustré par les exemples des figures 4 et 5.

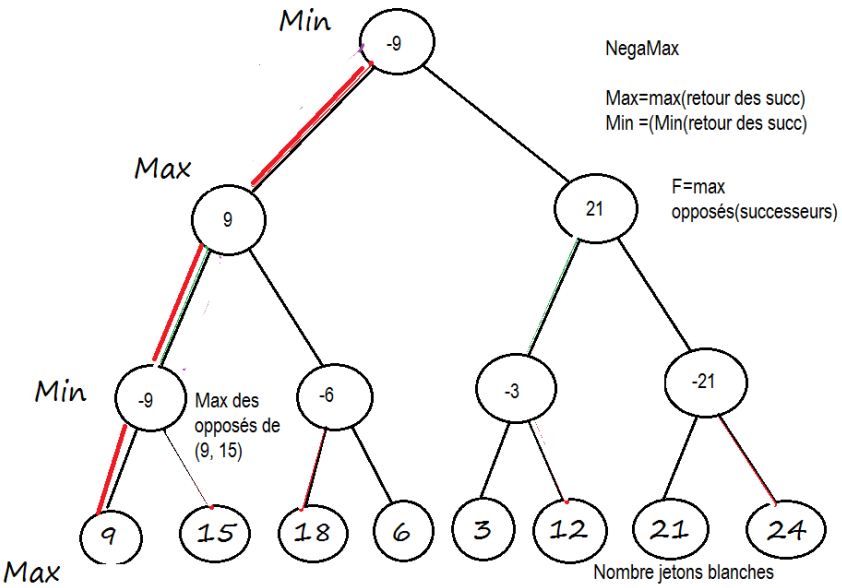


Figure 4.

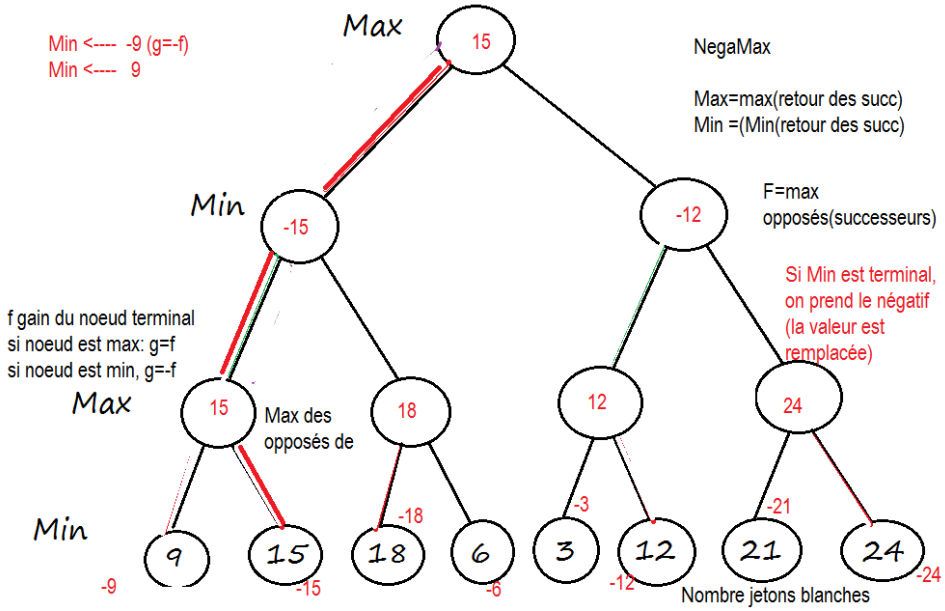


Figure 5.

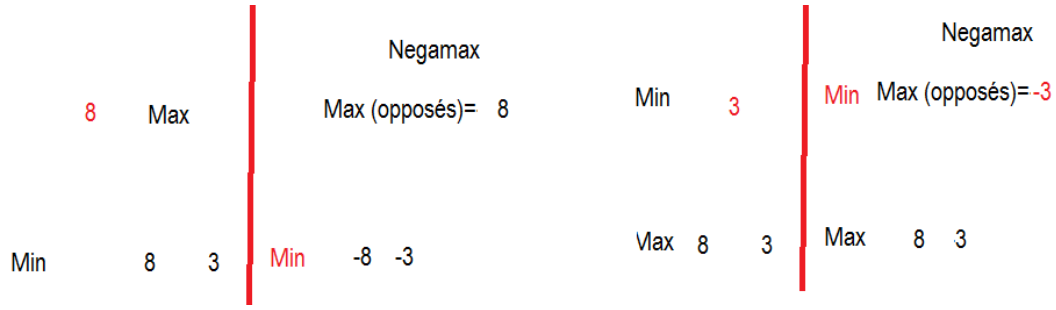
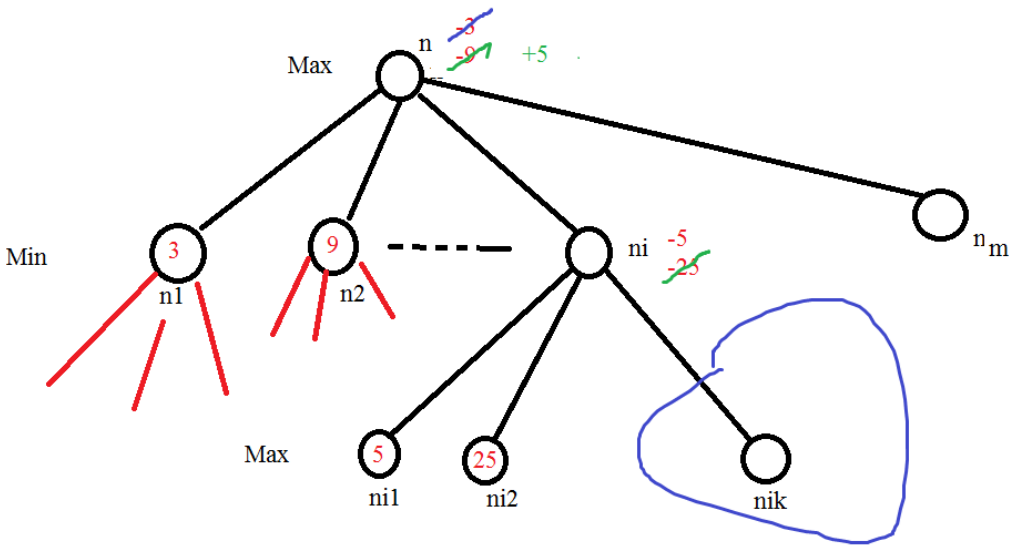
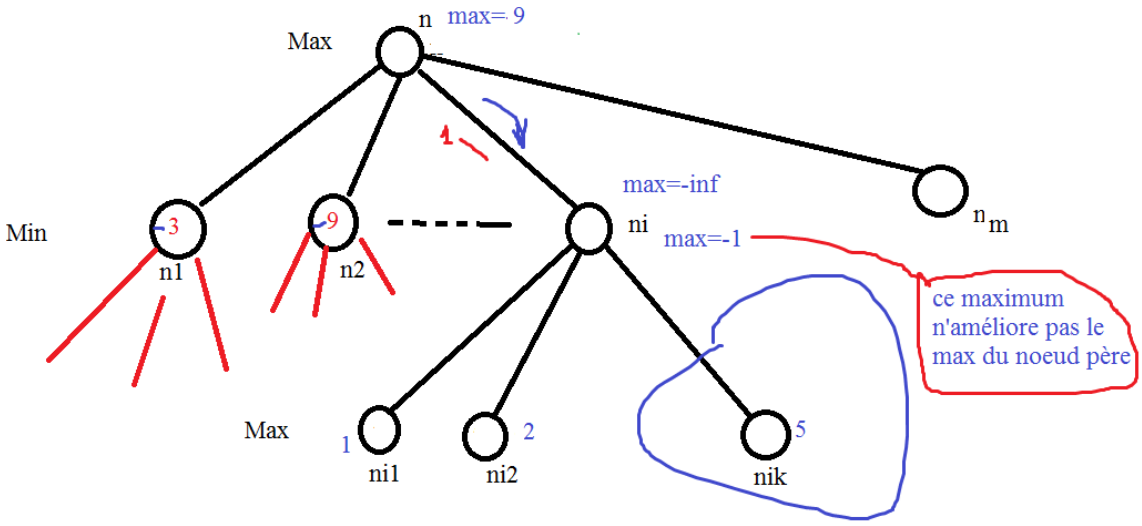


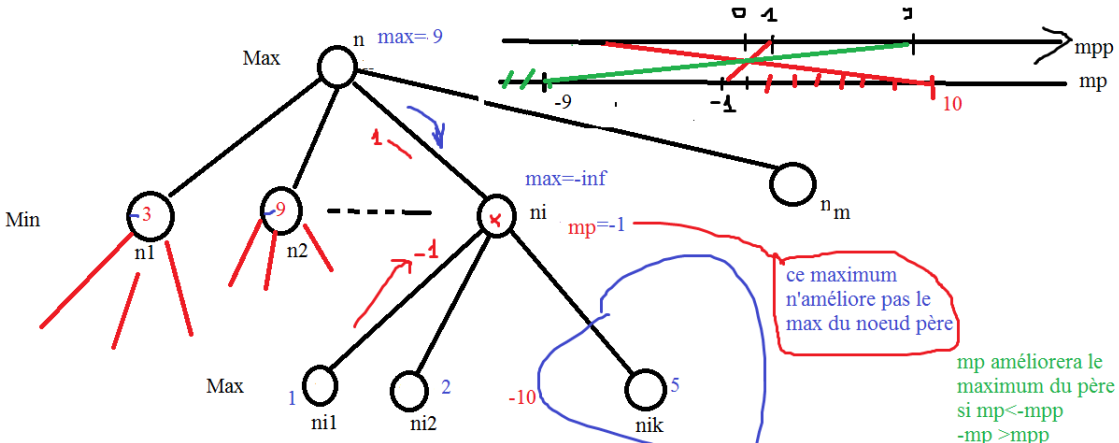
Figure 6. Equivalence entre les procédures minmax et Negamax

Int NEGAMAX(n)
{
-Déterminer les successeurs de n, soient n_1, n_2, .., n_d
If(d==0) then return V(n) / valeur fournie par l'heuristique choisie pour le nœud feuille
Else m=-∞
For(i=1 to d)
{
t=-NEGAMAX(n_i)
If(t>m) then m=t;
}
return m
}

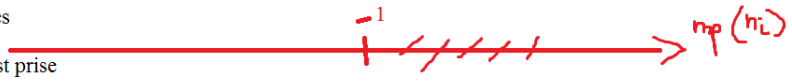
2.4 **Borne supérieure** beta (β)

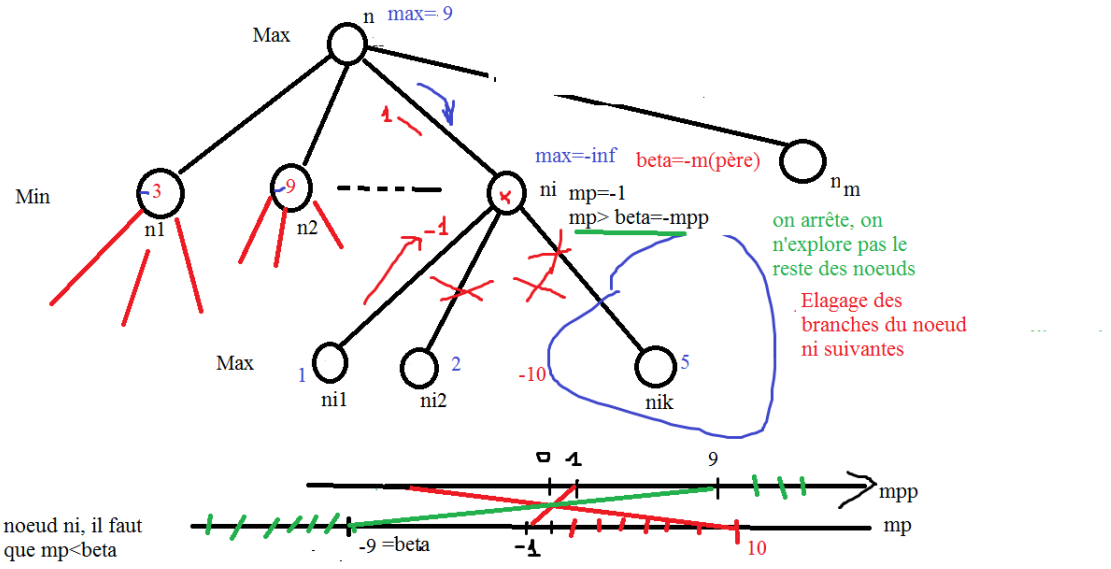






Après appel à negamax du premier succ, la valeur -1 est retournée. Toute valeur à retourner par les autres successeurs doit être supérieure à -1, sinon, elle n'est prise en compte





noeud ni, il faut que mp<beta

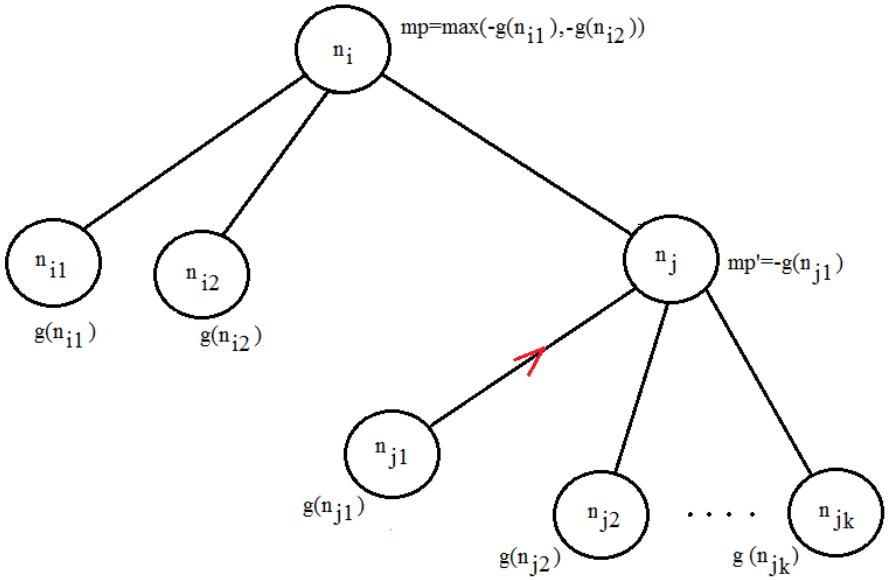


Figure 4. Exemple d'un arbre de jeu

Sur la figure 4, le maximum provisoire associé au nœud n_i est calculé en utilisant les nœuds n_{i1} , n_{i2} comme étant le maximum de $-g(n_{i1})$ et de $-g(n_{i2})$.
L'exploration du nœud fils n_j aura un effet sur le maximum provisoire si l'évaluation du nœud n_j donne une valeur qui mp' qui vérifie $-mp' > mp$ et donc $mp' < -mp$.

Pour que mp soit mis à jour, il faudra que l'évaluation du nœud fils n_j donne un valeur mp' qui doit vérifier : $-mp' > mp$ ce qui est équivalent à $mp' < -mp$.

Si on note $\beta = -mp$, alors si $mp' > \beta$ alors il n'est pas utile de continuer à évaluer les nœuds fils de n_j, car les valeurs retournées $-g(n_{jl})$ seront retenues que si elles sont supérieures à mp'. $-g(n_{jl}) > mp'$ et comme $' > \beta$ **,** ces nœuds vont mettre à jour mp' mais qui reste supérieur à β.

Conclusion : On évaluera le maximum provisoire du nœud n_j tant qu'il est inférieur à β. On arrêtera dès qu'il dépasse β. Les nœuds restants ne seront pas explorés et l'évaluation de nœud ni ne changera pas. On parle alors d'élagage β (voir figure 5).

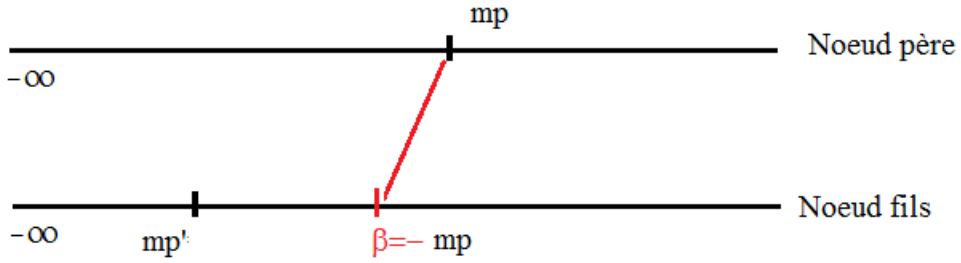
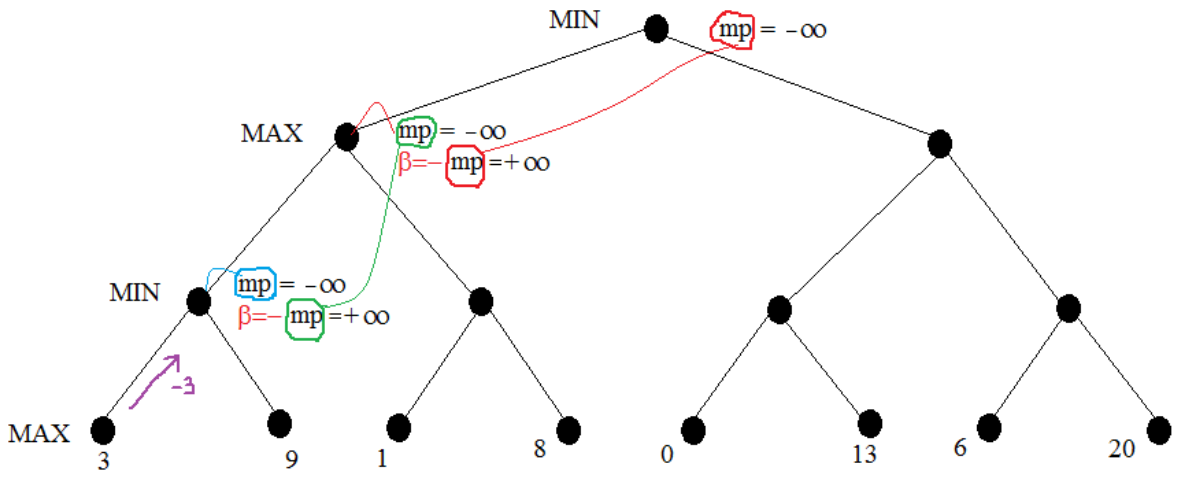


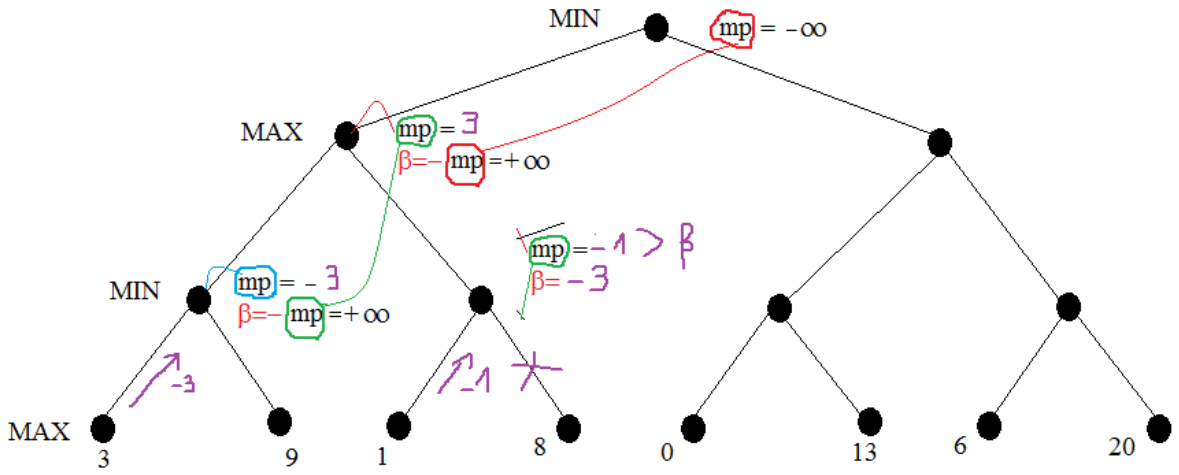
Figure 5. Evolution de mp' relativement à beta

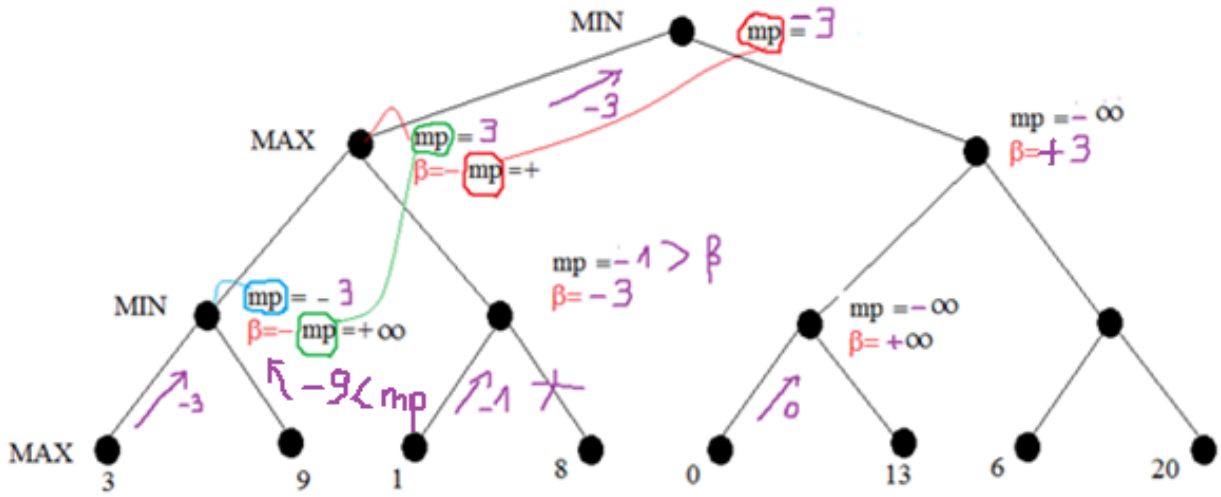
L'utilisation de la borne supérieure beta (β) se fait selon la procédure suivante:

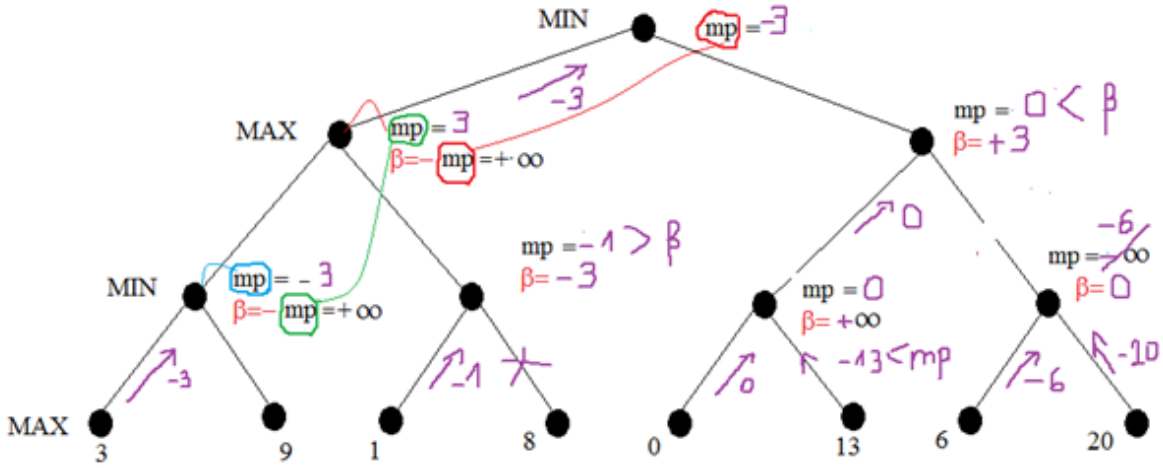
Int NEGAMAX(n, β)
{
-Déterminer les successeurs de n, soient n_1, n_2, .., n_d
If(d==0) then return V(n) / valeur fournie par l'heuristique choisie pour le nœud feuille
Else m=-∞; i=1;
While((i≤d)&&(m<β))
{
t=-NEGAMAX(n_i, -m)
If(t>m) then m=t;
i++ ;
}
return m
}

Exemple :




MIN
mp = -∞
MAX
mp = 3
β=- mp =+∞
MIN
mp = - 3
β=- mp =+∞
mp =-1 > β
β= -3
MAX
3
9
1
8
0
13
6
20


MIN
mp =3
MAX
mp = 3
β=- mp =+
mp =- ∞
β=+3
MIN
mp = - 3
β=- mp =+∞
mp =-1 > β
β= -3
mp =- ∞
β= +∞
MAX
3
9
1
8
0
13
6
20


MIN
mp =-3
MAX
mp = 3
β=- mp =+∞
mp = 0 < β
β=+3
MIN
mp = - 3
β=- mp =+∞
mp =-1 > β
β= -3
mp = 0
β= +∞
mp =- ∞
β= 0
MAX
3
9
1
8
0
13
6
20

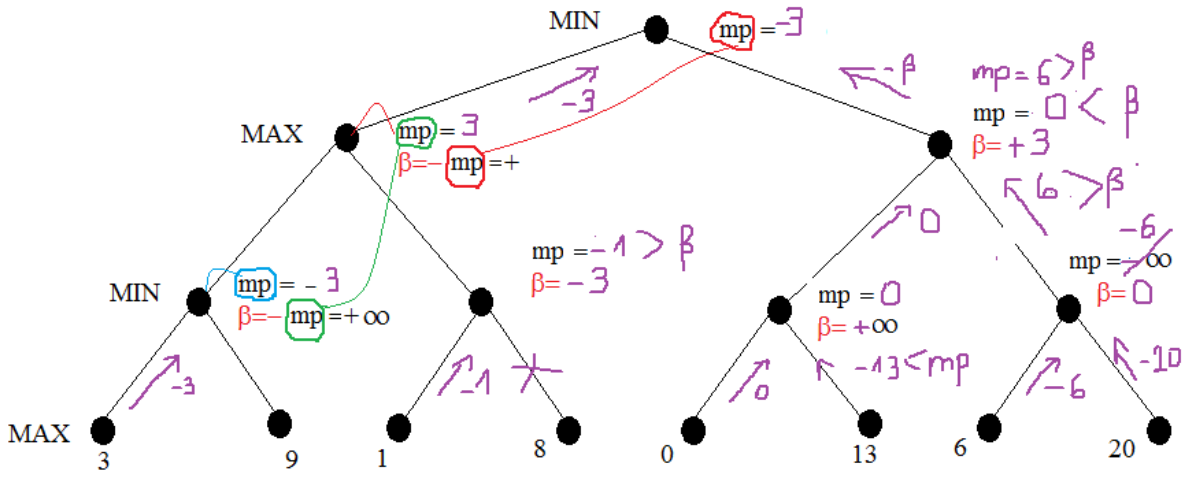
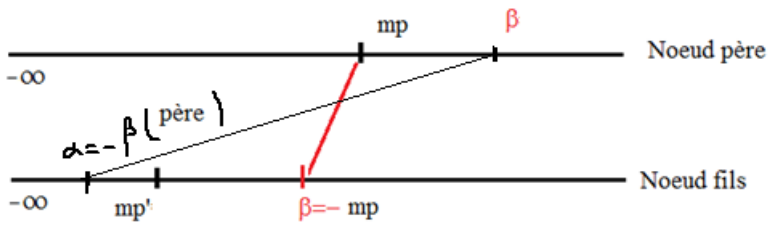


Figure 6. Déroulement Negamax avec élagage beta

2.5 Borne inférieure beta (α)

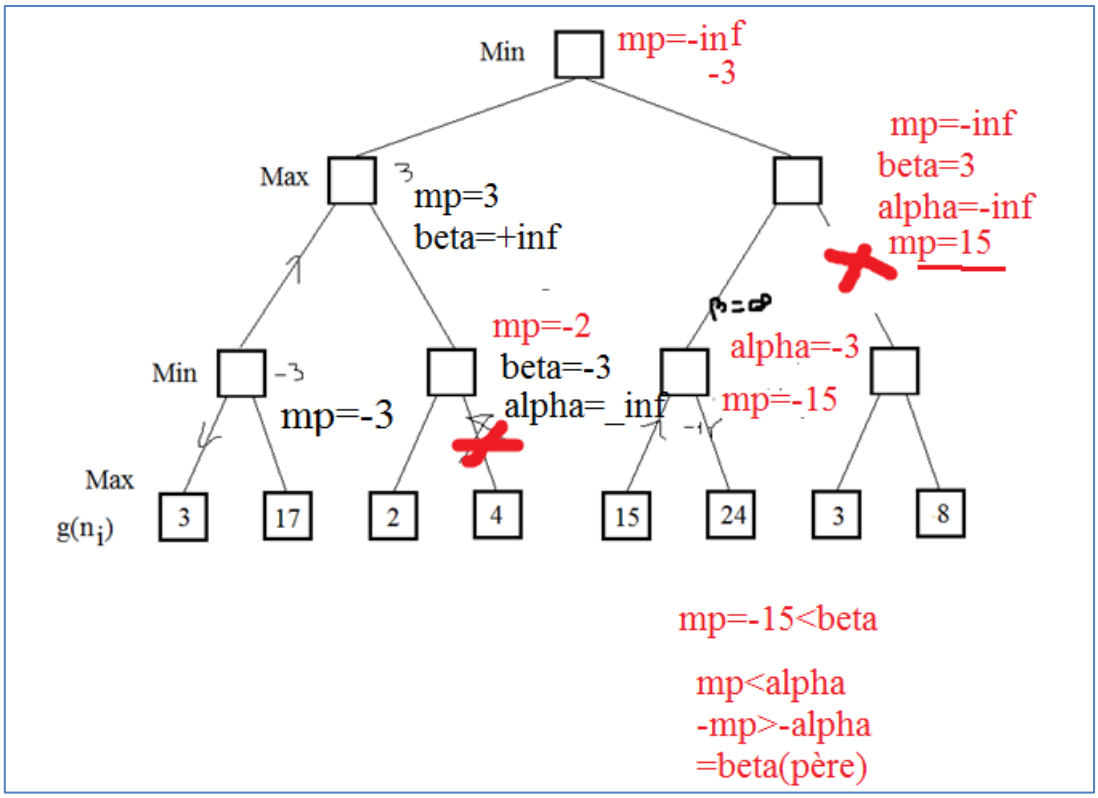
L'utilisation de la borne inférieure alpha (α) se fait selon la procédure suivante.
L'appel du nœud racine se fait avec : **α=-∞,β=+∞**

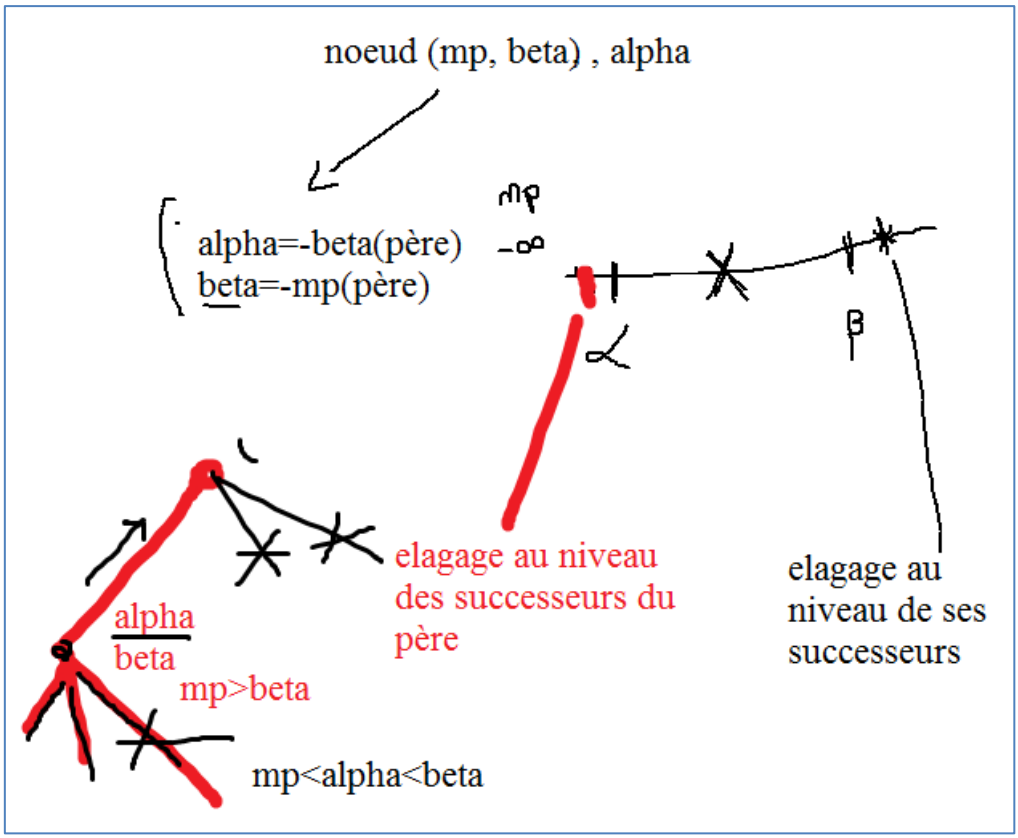


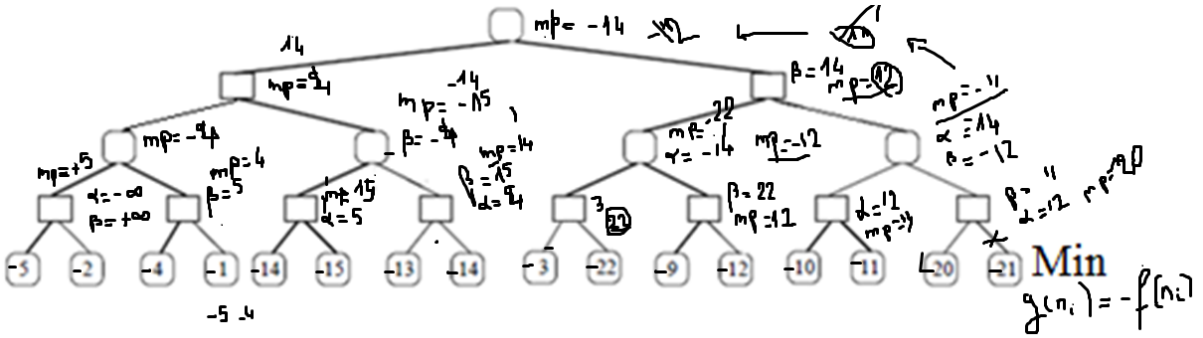
int NEGAMAX(n, α,β)
{
-Déterminer les successeurs de n, soient n_1, n_2, .., n_d
If(d==0) then return V(n) / valeur fournie par l'heuristique choisie pour le nœud feuille
Else m=α; i=1;
While((i≤d)&&(m<β))
{
t=-NEGAMAX(n_i, -β, -m)
If(t>m) then m=t;
i++ ;
}
return m
}

Résultat :

Si *mp* d'un nœud est inférieur à α de ce nœud, alors il y aura élagage des nœuds fils suivants du nœud père comme le montre la figure suivante.


Min
mp=-inf
-3
Max
3
mp=3
beta=+inf
mp=-inf
beta=3
alpha=-inf
mp=15
Min
-3
mp=-3
mp=-2
beta=-3
alpha=_inf
alpha=-3
mp=-15
Max
g(n_i)
3
17
2
4
15
24
3
8
mp=-15<beta
mp<alpha
-mp>-alpha
=beta(père)


noeud (mp, beta) , alpha
alpha=-beta(père)
beta=-mp(père)
mp
-∞
α
β
elagage au niveau des successeurs du père
elagage au niveau de ses successeurs
alpha
beta
mp>beta
mp<alpha<beta


Min