Chapitre 3.

Algorithmes de recherche pour les jeux

3.1 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).

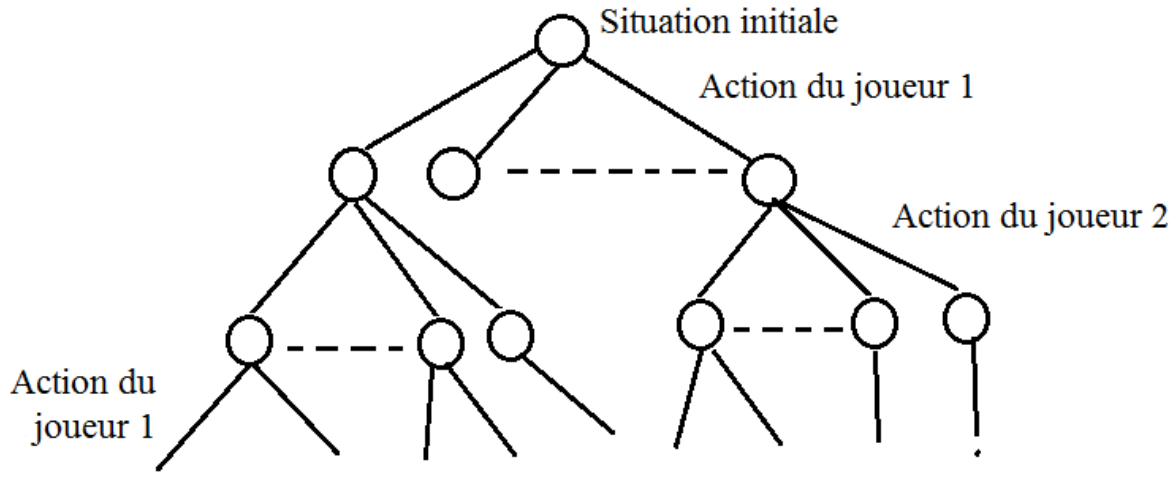

Figure 1. Représentation d'un jeu par un arbre

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.

Procédure MinMax

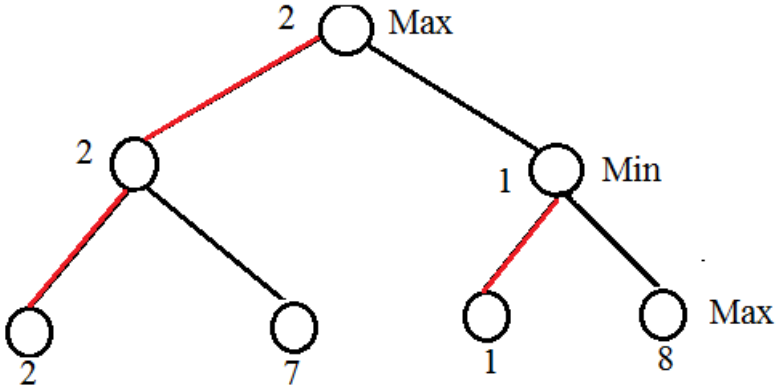

Figure 2. Exemple d'arbre de jeu avec Min Max

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 n1, n2, .., nd
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=Min(ni)
                  If(t>m) then m=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 n1, n2, .., nd
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=Max(ni)
              If(t<m) then m=t;
              }
return m
}
```

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.

```
Int NEGAMAX(n)
{
-Déterminer les successeurs de n, soient n1, n2, .., nd
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(ni)
                If(t>m) then m=t;
                }
return m
}
```

Exemple :

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.
Ci-après l'arbre de jeu. Le gain +1 est associé au nœud si Max gagne et -1 s'il perd.

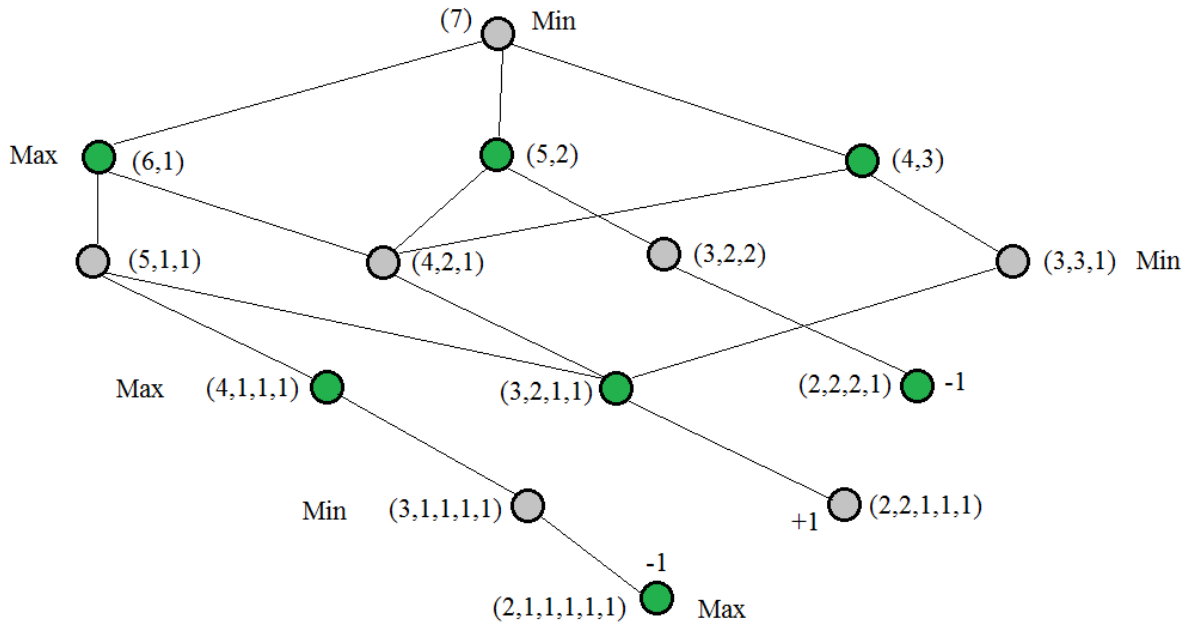

Figure 3. Arbre de jeu pour un état initial (Min, 7 jetons)

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

```
Int NEGAMAX(n, β)
{
-Déterminer les successeurs de n, soient n1, n2, .., nd
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(ni, -m)
            If(t>m) then m=t;
            i++ ;
            }
return m
}
```

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 n1, n2, .., nd
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(ni, -β, -m)
            If(t>m) then m=t;
            i++ ;
            }
return m
}
```