Intelligence Artificielle

Algorithmes pour jeux

Emmanuel ADAM

Université de Valenciennes et du Hainaut-Cambrésis

UVHC/ISTV-LAMIH

Jeux à deux joueurs

Caractéristiques des jeux

- Les deux adversaires (O et H) jouent à tour de rôle,
- La situation globale du jeu est connue de chacun des joueurs,
- La chance n'intervient pas,
- Les jeux sont dits "à somme nulle" : les gains d'un joueur représentent exactement les pertes de l'autre joueur.

Algorithme MiniMax - NegaMax

Algorithme MiniMax - NegaMax

- Complexité : dimension de l'espace d'états
  - Morpion : facteur de branchement ≈ 3, nb de demi-coups $= 9$ au plus
    dimension $\approx 3^9 = 19683$
  - Echec : facteur de branchement ≈ 35, nb de demi-coups ≈ 30
    dimension $\approx 35^{30}$! ! !
- Nécessité d'une profondeur maximale de résolution
- Nécessité d'une fonction d'évaluation pour estimer les noeuds non feuilles
- Chaque joueur joue le coup de gain maximal pour lui,
  en sachant que, et en prenant en compte que, l'adversaire fera de même

Evaluation dans l'algorithme MiniMax - NegaMax

Evaluation

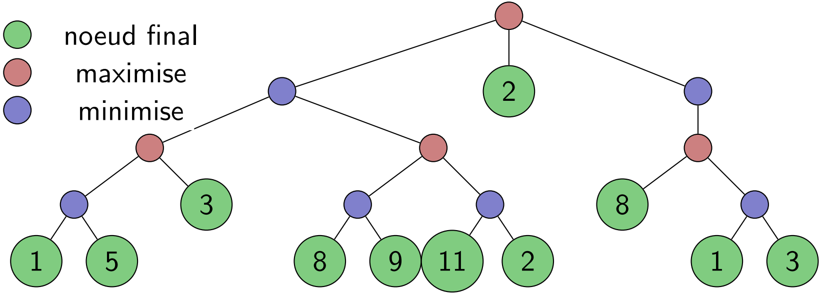
- Stratégie : profondeur limitée
- L'heuristique h évalue la qualité d'un noeud terminal (feuille ou de profondeur maximale)
- Comportement du joueur O_{rdi} :
  - O parcourt en profondeur et note les noeuds terminaux de niveau n d'une branche
  - O transmet les notes au noeud $n-1$ et sélectionne :
    - le noeud de valeur maximale si le trait appartient à O (étape maximisante)
    - le noeud de valeur minimale si le trait appartient à H (étape minimisante)
  - O transmet la note du noeud $n-1$ au noeud $n-2$ et note les autres noeuds de niveau $n-1$
  - O poursuit sa notation pour pouvoir choisir parmi les noeuds de niveau 1

Exemple de MiniMax 1/4

Evaluation

Arbre présentant l'espace d'états pour O.

- O cherche à maximiser ses gains
- sachant que H cherchera à lui les minimiser

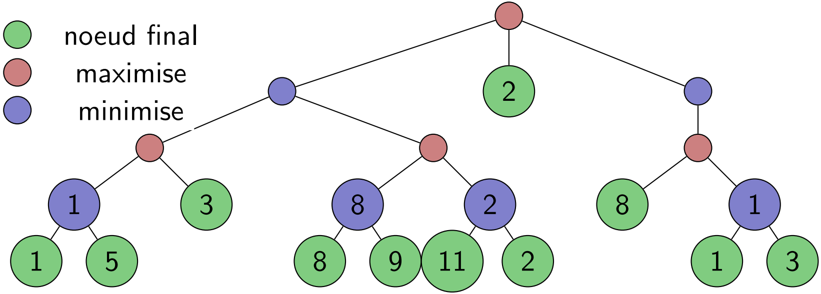


Exemple de MiniMax 2/4

Evaluation

Arbre présentant l'espace d'états pour O.

- O cherche à maximiser ses gains
- sachant que H cherchera à lui les minimiser

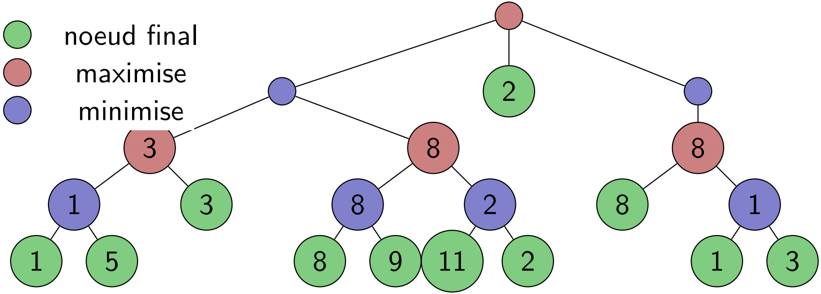


Exemple de MiniMax 3/4

Evaluation

Arbre présentant l'espace d'états pour O.

- O cherche à maximiser ses gains
- sachant que H cherchera à lui les minimiser

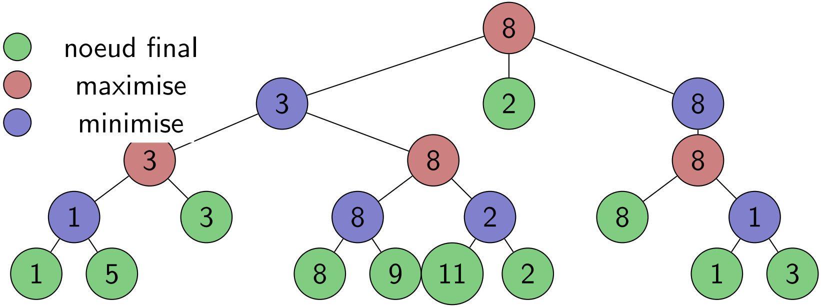


Exemple de MiniMax 4/4

Evaluation

Arbre présentant l'espace d'états pour O.

- O cherche à maximiser ses gains
- sachant que H cherchera à lui les minimiser



Exemple d'algorithme MiniMax I

```
procedure MINIMAX(s : situation)
    minimax; valSousArbre : entier ;

    if ESTFEUILLE(s) then
        RETOURNER(h(s))
    end if

    if ESTMAX(s) then
        minimax ← −∞
        for all s′ ∈ s.successeurs do
            valSousArbre ← MINIMAX(s')
            if (minimax < valSousArbre) then
                minimax ← valSousArbre
            end if
        end for
        RETOURNER(minimax)
    end if
```

Exemple d'algorithme MiniMax II

```
if ESTMIN(s) then
    minimax ← +∞
    for all s' ∈ s.successeurs do
        valSousArbre ← MINIMAX(s')
        if (minimax > valSousArbre) then
            minimax ← valSousArbre
        end if
    end for
    RETOURNER(minimax)
end if
end procedure
```

Exemple d'algorithme NegaMax I

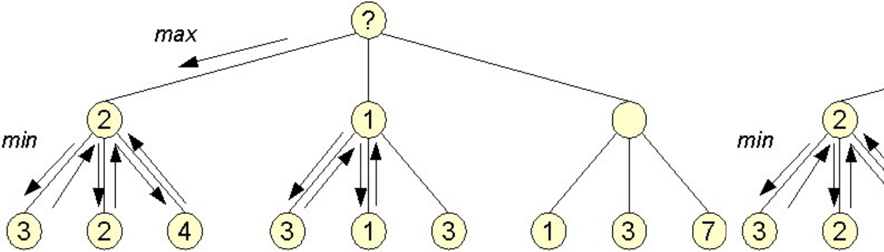
L'Algorithme du NegaMax est simplifié :
O prend le gain minimal de H en prenant l'oposé de son NegaMax et inversement

```
procedure NEGAMAX(s : situation)
    minimax; valSousArbre : entier ;
    if ESTFEUILLE(s) then
        RETOURNER(h(s))
    end if
    minimax ← −∞
    for all s′ ∈ s.successeurs do
        valSousArbre ← − NEGAMAX(s')
        if (minimax < valSousArbre) then
            minimax ← valSousArbre
        end if
    end for
    RETOURNER(minimax)
end procedure
```

Algorithme Alpha - Beta

Algorithme Alpha - Beta

- Problème du MiniMax : Exploration complète de l'arbre, de l'espace d'états
- Possibilité d'élaguer des branches en fonction des découvertes



Alpha - Beta

α et β

On distingue deux seuils, appelés α (pour les noeuds Min) et β (pour les noeuds Max) :

- le seuil α, pour un noeud Min s, est égal à la plus grande valeur (déjà déterminée) de tous les noeuds Max ancêtres de s. Si la valeur de s devient inférieure ou égale à α, l'exploration de sa descendance peut être arrêtée ;
- le seuil β, pour un noeud Max s, est égal à la plus petite valeur (déjà déterminée) de tous les noeuds Min ancêtres de s. Si la valeur de s devient supérieure ou égale à β, l'exploration de sa descendance peut être arrêtée.
- α et β sont initialisés réciproquement à $-\infty$ et $+\infty$.

Exemple d'algorithme $\alpha - \beta$ I

```
procedure AlphaBeta(s : situation, α : entier, β : entier)
    minimax; valSousArbre : entier ;
    if estFeuille(s) then retourner(h(s))
    end if

    if estMax(s) then
        minimax ← α
        for all s′ ∈ s.successeurs do
            valSousArbre ← AlphaBeta(s', minimax, β)
            if (minimax < valSousArbre) then
                minimax ← valSousArbre
            end if
            if (minimax ≥ β) then retourner(minimax)
            end if
        end for
        retourner(minimax)
    end if
```

Exemple d'algorithme $\alpha - \beta$ II

```
if ESTMAX(s) then
    minimax ← β
    for all  s' ∈ s.successeurs do
        valSousArbre ← ALPHABETA(s', α, minimax)
        if  (minimax > valSousArbre) then
            minimax ← valSousArbre
        end if
        if  (minimax ≤ α)  then RETOURNER(minimax)
        end if
    end for
    RETOURNER(minimax)
end if
end procedure
```

Exemple d'algorithme $\alpha - \beta$ simplifié

Voici l'algorithme $\alpha - \beta$ simplifié sur base du NegaMax :

```
procedure AlphaBetaNega(s : situation, α : entier, β : entier)
    if estFeuille(s) then
        retourner(h(s))
    else
        for all  s' ∈ s.successeurs do
            β ← max(α, −ALPHABETANEGA(s', −β, −α))
            if  α ≥ β then
                retourner(α)
            end if
            α ← max(α, β)
        end for
        retourner(α)
    end if
end procedure
```

Fonction d'évaluation

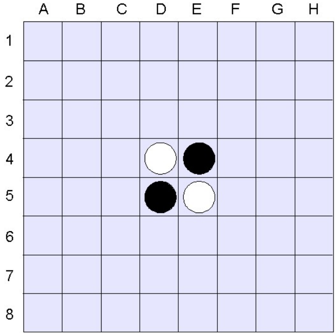
Heuristique

- Dans le cadre des jeux, l'heuristique dépend souvent des critères liés :
  - au matériel (jetons, ...) ;
  - à la mobilité (nombre de coups possibles) ;
  - à la position (des pièces, ...).
  - $h = p_{materiel} \times c_{materiel} + p_{mobilite} \times c_{mobilite} + p_{position} \times c_{position}$
- les pondérations peuvent évoluer en cours de partie

Exemple sur Othello

Othello

- Grille de 64 cases dont 4 sont occupées : le jeu prend fin au plus tard après 60 coups.



Heuristique pour Othello

Heuristique pour Othello

- Reprenons les 3 critères définissant l'heuristique pour un jeu, ici pour une couleur :
  - le matériel = nombre de pions de la couleur,
  - la mobilité = nombre de cases jouables par la couleur,
  - la valeur d'une position = somme des valeurs des cases occupées par la couleur.
- On utilise une grille évaluant la valeur d'une case, adaptée au jeu :

500	-150	30	10	10	30	-150	500
-150	-250	0	0	0	0	-250	-150
30	0	1	2	2	1	0	30
10	0	2	16	16	2	0	10
10	0	2	16	16	2	0	10
30	0	1	2	2	1	0	30
-150	-250	0	0	0	0	-250	-150
500	-150	30	10	10	30	-150	500

Stratégie pour Othello

Stratégie

- Evolution de l'importance (du poids) des 3 critères en cours de jeu
  - debut : mobilité et position favorisées ($p_{mobilite}$ au maximum et $p_{position}$ important)
  - milieu de partie : importance de conquérir bords et points : $p_{position}$ au maximum
  - fin de partie : importance du nombre de jetons de la bonne couleur : $p_{materiel}$ au maximum
- Evolution du facteur de branchement : plus faible vers la fin
  - donc possibilité d'augmenter la profondeur de calcul
- Possibilité d'évaluer le niveau du joueur adverse