|  |
| --- |
|  |
| Projet minimax, alpha-beta et MCTS |
| Projet d’IA |

|  |
| --- |
| Guillaume Grandy et Mathis Poncet-Crouzet  23/05/2024 |

**Table des matières**

[Minimax VS Alpha-Beta 2](#_Toc167970450)

[En termes de nœuds crées 2](#_Toc167970451)

[En termes de temps d’exécution 3](#_Toc167970452)

[Performances agent contre agent 4](#_Toc167970453)

[Alpha Beta / Min-Max 4](#_Toc167970454)

[Les fonctions d’évaluations mis en place 4](#_Toc167970455)

[Test IA contre adversaire multiple 5](#_Toc167970456)

# Minimax VS Alpha-Beta

Les tests ci-dessous sont effectuées sur un plateau de 6x6 cases, soit 36 cases vides initialement. Chaque agent a une profondeur de 3. Pour les tests, nous avons d’abord fait une partie en faisant commencé minimax, puis on a ensuite interverti en faisant commencer alpha-beta. Cela nous a permis d’avoir le nombre d’états par tour pour les 2 joueurs.

## En termes de nœuds crées

La courbe bleu et l’axe des ordonnées à droites et le nombre de cases vides par tour. C’est évidemment une fonction qui décroit car à chaque tour 2 pions sont ajoutés avec comme équation de droite

On voit que l’élagage de branches ajoutées dans Alpha Beta permet de limiter une exploration trop profonde et donc d’éviter la création de nouveaux états à explorer. Cela évite en plus de lancer une évaluation sur une branche qui ne nécessitait aucune évaluation.

La différence de nombres d’états créés entre les 2 algos sur chaque tour est conséquente. Rien que sur le 1 er tour, on a entre 35 000 et 40 000 nœuds en moins qui ne sont pas créés pour l’algorithme alpha-beta.

## En termes de temps d’exécution

Pour le temps d’exécution, on voit bien qu’Alpha-Beta est l’algorithme le plus rapide. En effet, beaucoup moins d’états sont explorés => le nombre d’appel de la fonction d’évaluations est donc forcément beaucoup moins important dans l’algorithme AlphaBeta. Etant donnée que l’on fait un parcours dans un tableau 2 dimensions, c’est la partie de l’algorithme qui peut faire perdre le plus de temps.

# Performances agent contre agent

## Alpha Beta / Min-Max

Afin de gagner du temps, nous avons estimer que les performances d’alpha-beta et min-max étaient similaires. En effet, le principe est le même pour les 2 algorithmes et ils donnent tout 2 le même résultat. Les seules différences sont que l’algorithme alpha-beta est plus optimisée et permet d’obtenir un résultat avec un temps d’exécution nettement inférieur à min-max (voir cette partie En termes de temps d’exécution) grâce au principe d’élagage de branches effectués par Alpha Beta.

### Les fonctions d’évaluations mis en place

|  |  |
| --- | --- |
| Nom | Description |
| Eval2 | * si on a un alignement de pions du même type qui peut être gagnant et pour les pions opposants |
| Eval3 | * si on a un alignement de pions du même type qui peut être gagnant et pour les pions opposants * si on a un alignement de pions du même type qui est gagnant et pour les pions opposants |

Sur la fonction d’évaluation 2, **eval2**, on s’est rendu compte que l’IA ne prenait pas en compte les récompenses immédiates. Voici un exemple ci-dessous pour illustrer notre propos.

Une image contenant cercle, Graphique

Description générée automatiquement Est le pion qu’on joue, et Une image contenant conception

Description générée automatiquement avec une confiance moyenne est le pion adverse. Sur cette série de coup, Une image contenant cercle, Graphique

Description générée automatiquement est le pion qui commence. Chaque grille est une étape de la partie jouée par les 2 joueurs avec Une image contenant cercle, Graphique

Description générée automatiquement qui commence.

Une image contenant ligne, Tracé, diagramme, capture d’écran

Description générée automatiquement

On voit à l’étape 4 que Une image contenant conception

Description générée automatiquement avec une confiance moyenne préfère continuer sa diagonale commencée à l’étape 3 au lieu de bloquer Une image contenant cercle, Graphique

Description générée automatiquement qui va gagner au prochain tour en alignant 4 pions Une image contenant cercle, Graphique

Description générée automatiquement

Cela nous a amené à mettre en place une nouvelle fonction d’évaluation 3, **eval3**. On a décidé de rajouter un score très élevé si on a la possibilité de gagner immédiatement et un malus > à ce score afin que l’IA évite au maximum de choisir des actions menant à une victoire immédiate de l’adversaire.

Une image contenant cercle, Graphique

Description générée automatiquement Est le pion qu’on joue, et Une image contenant conception

Description générée automatiquement avec une confiance moyenne est le pion adverse. Sur cette série de coup, Une image contenant cercle, Graphique

Description générée automatiquement est le pion qui commence. Chaque grille est une étape de la partie jouée par les 2 joueurs avec Une image contenant cercle, Graphique

Description générée automatiquement qui commence.

Une image contenant ligne, capture d’écran, Tracé, Caractère coloré

Description générée automatiquement

On voit qu’à l’étape 4, cette IA prend en compte le malus attribué à un état qui fait que l’adversaire peut gagner. Elle choisit donc de bloquer le prochain coup gagnant du joueur adverse.

### Test IA contre adversaire multiple

Les essais ont été fait sur un tableau de taille 6x6 avec une taille de victoire de 4 pions alignés.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Joueur 1** | **Depth Joueur 1** | **Type Joueur 2** | **Depth Joueur 2** | **Nombre partie** | **WinRate Joueur 1** | **WinRate Joueur 2** | **Egalité** | **Evaluation** |
| Alpha Beta | 1 | Alpha Beta | 1 | 10 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 1 | Alpha Beta | 2 | 10 | 0% | 100% | 0% | eval3 |
| Alpha Beta | 1 | Alpha Beta | 3 | 10 | 0% | 100% | 0% | eval3 |
| Alpha Beta | 2 | Alpha Beta | 1 | 10 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 2 | Alpha Beta | 2 | 10 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 2 | Alpha Beta | 3 | 10 | 0% | 0% | 100% | eval3 |
| Alpha Beta | 3 | Alpha Beta | 1 | 10 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 3 | Alpha Beta | 2 | 10 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 3 | Alpha Beta | 3 | 10 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 1 | Coup aleatoire | Non applicable | 100 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 2 | Coup aleatoire | Non applicable | 100 | 100% | 0% | 0% | eval3 |
| Alpha Beta | 3 | Coup aleatoire | Non applicable | 100 | 100% | 0% | 0% | eval3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Joueur 1** | **Depth Joueur 1** | **Type Joueur 2** | **Depth Joueur 2** | **Nombre partie** | **WinRate Joueur 1** | **WinRate Joueur 2** | **Egalité** | **Evaluation** |
| Alpha Beta | 1 | Alpha Beta | 1 | 10 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 1 | Alpha Beta | 2 | 10 | 0% | 100% | 0% | eval2 |
| Alpha Beta | 1 | Alpha Beta | 3 | 10 | 0% | 100% | 0% | eval2 |
| Alpha Beta | 2 | Alpha Beta | 1 | 10 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 2 | Alpha Beta | 2 | 10 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 2 | Alpha Beta | 3 | 10 | 0% | 0% | 100% | eval2 |
| Alpha Beta | 3 | Alpha Beta | 1 | 10 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 3 | Alpha Beta | 2 | 10 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 3 | Alpha Beta | 3 | 10 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 1 | Coup aleatoire | Non applicable | 100 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 2 | Coup aleatoire | Non applicable | 100 | 100% | 0% | 0% | eval2 |
| Alpha Beta | 3 | Coup aleatoire | Non applicable | 100 | 100% | 0% | 0% | eval2 |

On remarque plusieurs choses :

Si la profondeur du joueur 1 et 2 sont identiques, le joueur 1 gagne car commencé en premier est un net avantage.

Si la profondeur du joueur 1 est inférieur à celle du joueur 2, le joueur 2 gagne ou les deux sont égalités