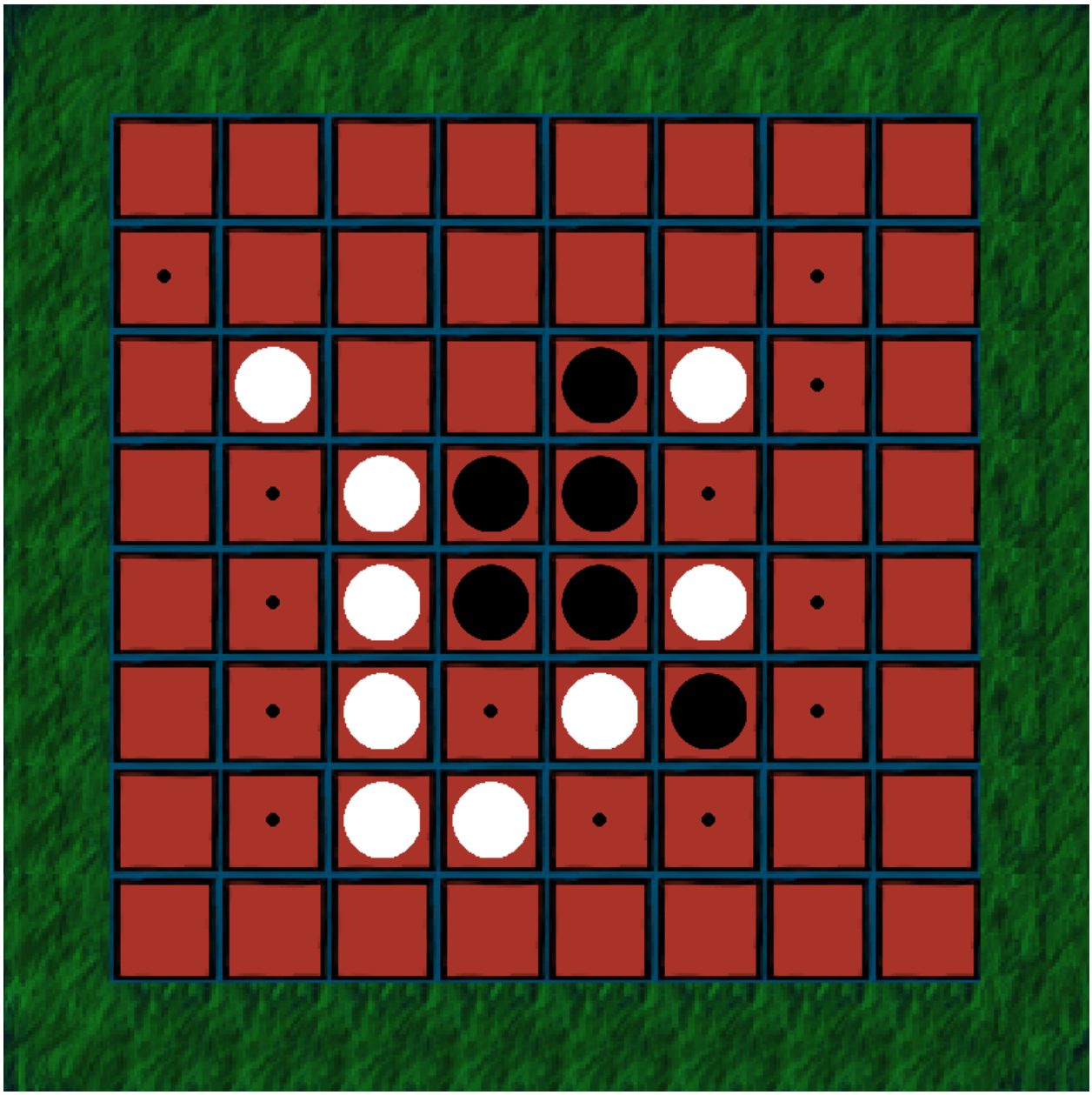
Intelligence artificielle

Rapport de projet IA : Othello/Reversi

Maily CIAVALDINI, Samuel LAVALLEE



Sommaire

1. Introduction

Dans le cadre de l'Unité d'Enseignement Intelligence Artificielle pour le semestre 6 du parcours Informatique et Applications de l'Université Paris Cité, nous avons été proposés de faire un projet qui consiste à programmer un jeu à deux joueurs, ayant la possibilité de faire jouer un joueur humain contre un joueur artificiel. L'objectif est aussi d'ajouter le choix entre au moins 3 difficultés différentes d'IAs. Nous avons choisi de programmer une IA dans le jeu nommé Othello, aussi connu sous le nom de Reversi.

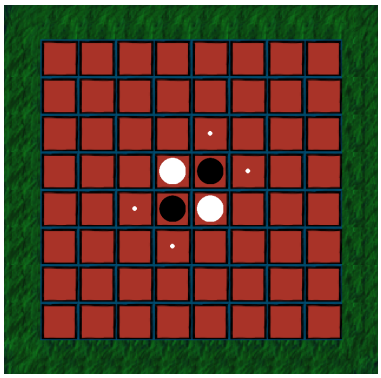
Ce document a pour but de présenter notre projet, en expliquant certains points importants et en se focalisant sur le détail de l'implémentation des IAs.

2. Présentation et règles du jeu

Le jeu de Reversi, également connu sous le nom d'Othello, est un jeu de société pour deux joueurs, généralement considéré comme un jeu de stratégie.

L'objectif du jeu est de posséder plus de pièces que l'adversaire à la fin de la partie, lorsque le plateau est rempli de pièces ou si l'un des joueurs ne peut plus poser de pièces.

Le plateau de jeu est constitué d'une grille de 8x8 cases, avec des pions noirs et blancs disposés de manière alternée sur les 4 cases centrales du plateau au début de la partie comme nous le montre l'illustration ci dessous :



Les joueurs jouent à tour de rôle en posant un pion de leur couleur sur le plateau, dans une case vide adjacente à un pion de l'adversaire. Les pions de l'adversaire qui se trouvent entre le nouveau pion et les pions déjà posés du joueur sont retournés pour devenir de la couleur du joueur. Le joueur peut alors capturer un ou plusieurs pions de l'adversaire dans une ligne droite dans n'importe quelle direction (horizontale, verticale ou diagonale). Le joueur est obligé de jouer dans une case qui va permettre de retourner des pions adverses. Si un joueur ne peut jouer aucun coup permettant de retourner au moins un pion, alors il se doit de passer le tour et laisser son adversaire jouer.

Le jeu se poursuit ainsi jusqu'à ce que le plateau soit rempli de pions ou qu'aucun des joueurs ne puisse jouer. À la fin de la partie, le joueur avec le plus de pions de sa couleur sur le plateau est déclaré vainqueur. Si les deux joueurs ont le même nombre de pions, cela amène à un match nul.

3. Implémentation des Intelligences Artificielles

3.1. Objectifs

L'objectif principal du projet est d'implémenter des intelligences artificielles intéressantes, et surtout de comprendre les notions vues en cours grâce à cette application concrète. La base de ce projet est donc l'utilisation de l'algorithme minimax. Il a aussi été demandé d'implémenter l'élagage alpha-beta, qui va donc optimiser la recherche du minimax, et réduire le temps d'attente. Après avoir implémenté la base de l'IA, il a fallu ajouter différents niveaux. La stratégie à adopter était de réfléchir à comment modifier les fonctions d'évaluation utilisées par l'algorithme minimax, afin d'améliorer les prises de décisions. Une autre façon était aussi de modifier la profondeur de recherche, de sorte à ce que l'IA cherche plus où moins loin (à l'avance), ce qui donne un avantage plus conséquent lorsque celle-ci est plus élevée. Nous allons ainsi détailler chaque étape qui a permis la réalisation des différentes IAs maintenant fonctionnelles.

3.2. Fonctions d'évaluation

Tout d'abord, pour que l'algorithme minimax fonctionne, elle doit se baser sur les résultats de chaque feuille des nœuds parcourus.

La fonction d'évaluation est généralement utilisée dans les algorithmes de recherche de jeux. Elle sert à évaluer la qualité d'un état de jeu donné. Cette fonction prend en entrée l'état de jeu actuel et retourne une valeur numérique qui représente la qualité de l'état pour le joueur actuel. Cette valeur est souvent basée sur des caractéristiques du jeu telles que le nombre de pièces sur le plateau, la position des pièces, les zones contrôlées, etc.

3.2.1. Première approche

Dans notre jeu, nous avons tout d'abord établi une fonction d'évaluation très simple : il s'agit de compter le nombre de pions présents dans l'état du jeu actuel. Un score sera alors calculé, après avoir additionné d'un point chaque pion contrôlé par le joueur actuel et soustrait d'un point pour ceux contrôlés par l'opposant. Cette évaluation étant très

basique, fonctionne bien lorsque l'IA se bat contre un joueur débutant. Néanmoins, elle dépend fortement de la profondeur de recherche.

3.2.2. Deuxième approche

Étant donné la simplicité de la première approche, nous avons voulu améliorer l'intelligence de l'IA. Il s'agissait maintenant de comprendre différentes stratégies de victoire propres au jeu Othello. En effet, l'une des meilleures stratégies dans ce jeu est de contrôler les coins, mais aussi les côtés du plateau. Nous avons donc opté pour l'ajout de bonus plus ou moins grands lorsque le joueur contrôle un coin ou un côté. De la même façon, nous avons fait en sorte de diminuer les points du joueur lorsque celui-ci contrôle une case qui risque de donner le contrôle à l'adversaire, donc les cases adjacentes.

De plus, il était aussi intéressant de prendre en compte la mobilité de l'adversaire, un critère important sur Othello. Nous avons donc procédé de la même façon en ajoutant des bonus pour le nombre de coups possibles.

C'est donc de cette façon que notre IA de niveau difficile a été implémentée, étant assez robuste pour qu'un joueur expérimenté ne puisse pas le vaincre.

3.3. Minimax

L'algorithme minimax est un algorithme de recherche de l'arbre de jeu qui explore toutes les possibilités jusqu'à une certaine profondeur, puis évalue les feuilles de l'arbre (les états finaux) en utilisant une fonction d'évaluation. L'objectif est de maximiser le score du joueur qui joue en premier (le joueur qui appelle la fonction minimax), tout en minimisant le score de l'adversaire.

La profondeur de recherche choisie va très fortement impacter la rapidité et la qualité de la recherche. Concernant la rapidité, plus la profondeur est élevée, plus elle va prendre du temps. En contrepartie, l'algorithme trouvera une solution encore meilleure. Dans notre jeu et durant des tests, l'algorithme minimax que nous avons implémenté est extrêmement lent au-delà d'une profondeur de 3. De ce fait, nous avons rapidement voulu optimiser la recherche avec l'élagage alpha-beta.

3.4. Élagage alpha-beta

L'élagage alpha-beta est une amélioration de l'algorithme minimax pour accélérer la recherche en éliminant certaines branches de l'arbre de jeu qui ne contribuent pas à la décision finale.

L'algorithme utilise deux valeurs, alpha et beta, pour élaguer les branches qui ne sont pas utiles à la recherche. À chaque étape de la recherche, l'algorithme compare la valeur alpha, qui représente la meilleure valeur trouvée pour le joueur maximisant jusqu'à présent, à la valeur beta, qui représente la meilleure valeur trouvée pour le joueur minimisant. Si une valeur est trouvée qui est meilleure que la valeur beta, alors la recherche peut être interrompue car l'adversaire ne choisira jamais cette branche. De même, si une valeur est trouvée qui est pire que la valeur alpha, alors cette branche peut également être élaguée car le joueur maximisant ne la choisira jamais.

En élaguant les branches, l'élagage alpha-beta permet de réduire considérablement le nombre de nœuds évalués, accélérant ainsi la recherche tout en conservant la même qualité de décision que l'algorithme minimax. De ce fait, nous avons pu continuer nos tests avec des profondeurs au-delà de 5. Cependant, la recherche pouvait prendre beaucoup de temps même avec cette optimisation, notamment lorsque nous testions la profondeur au-delà de 8.

3.5. Choix des niveaux de difficulté des IAs

Pour résumer, nos IAs sont toutes basées sur l'algorithme minimax avec l'élagage alpha-beta. La première IA que nous avons choisie est le niveau "facile", qui utilise le critère de nombre de pions sur le plateau en tant que fonction d'évaluation. Pour la deuxième difficulté "moyen", nous avons voulu mettre en valeur l'importance de la profondeur de recherche de l'algorithme. Nous avons donc choisi la même IA que le niveau facile, mais avec une profondeur plus élevée. Enfin, le dernier niveau que nous avons choisi est le niveau "difficile". De la même façon, sa recherche se base sur le même algorithme, mais avec une fonction d'évaluation beaucoup plus poussée. Les critères mis en avant sont, comme dit précédemment, les coins, les côtés ainsi que la mobilité.

Ainsi, nous avons choisi nos trois niveaux respectifs, se basant donc sur la profondeur, et sur l'évaluation du plateau.

3.6. Améliorations

Des améliorations peuvent être faites pour notre projet notamment pour les différentes fonctions d'évaluation que nous avons implémentées. Par exemple, nous pourrions implémenter les différentes techniques et stratégies plus poussées utilisées par les professionnels, qui connaissent des stratégies qui ont été étudiées cas par cas. Bien sûr, il serait tout aussi intéressant d'observer les résultats d'une profondeur plus élevée avec l'IA de niveau difficile.

Concernant la base de notre IA, elle se repose sur l'algorithme minimax et l'élagage alpha-beta qui est fonctionnelle, mais il semblerait que la recherche soit toujours aussi lente, même avec l'élagage. Il s'agirait alors de faire appel à des notions plus poussées d'optimisation.

4. Résultats des Tournois

Nous disposons de trois niveaux d'intelligence artificielle (IA) dans notre programme : Facile, Moyen et Difficile. En outre, nous avons également inclus un mode "Aléatoire" où l'ordinateur place les pions de manière aléatoire.

Lors de nos tests, nous avons effectué de nombreux tournois, mais les résultats ont toujours été les mêmes : l'IA la plus forte a remporté toutes les parties, avec un taux de victoire de 100%. L'IA Facile ne parvient jamais à battre l'IA Difficile, quel que soit le nombre d'essais effectués. Cette observation est également valable pour les confrontations entre Facile et Moyen, ainsi que Moyen et Difficile.

Voici quelques résultats des matchs effectués :

4.1. IA Facile vs IA Moyen

Joueur **blanc** : niveau **Facile**

Joueur **noir** : niveau **Moyen**

Pions blancs : 14 | Pions noirs : 50

Victoire **noir**, niveau **Moyen**

4.2. IA Facile vs IA Difficile

Joueur **blanc** : niveau **Facile**

Joueur **noir** : niveau **Difficile**

Pions blancs : 16| Pions noirs : 48

Victoire noir, niveau **Difficile**

4.3. IA Difficile vs IA Moyen

Joueur **blanc** : niveau **Difficile**

Joueur **noir** : niveau **Moyen**

Pions blancs : 42| Pions noirs : 22

Victoire blanc, niveau **Difficile**

4.4. Remarques intéressantes

Pour donner plus de sens à la différence des IAs, nous avons lancé des tests de matchs contre l'agent aveugle. Ce genre de test est surtout intéressant lorsqu'il s'agit de faire une différence d'IA ayant une profondeur différente. Nous n'avons malheureusement pas pu automatiser ces tests pour avoir des statistiques avancées, mais nous avons facilement pu en ressortir des conclusions, bien que ces tests aient été effectués manuellement.

Les tests ont été réalisés en faisant 10 matchs contre l'agent aveugle pour chaque niveau de difficulté.

IA Facile : 1 match nul et 1 défaite / 10

IA Moyen : 1 défaite / 10

IA Difficile : 0 défaite

Nous observons donc que l'IA utilisée pour les niveaux Facile et Moyen n'est pas très intelligente. En effet, il se peut que l'agent aveugle prenne, sans le vouloir, un ou plusieurs coins, que l'algorithme n'a pas considéré, se focalisant uniquement sur le nombre total de pions. Sachant que la seule différence entre le niveau Facile et le niveau Moyen est la profondeur, ces résultats sont cohérents. Cela indique que ce ne sont pas des joueurs artificiels très intelligents.

En revanche, l'IA Difficile a été très satisfaisante, prouvant que nous avons réussi à développer une IA complètement fonctionnelle et beaucoup plus intéressante.

5. Conclusion

Pour conclure, nous avons été très satisfaits des résultats de notre projet. Nous avons pu appliquer des notions apprises durant le cours d'IA, ce qui nous a aidé à comprendre en profondeur ce sujet en particulier. En plus du domaine de l'Intelligence Artificielle, nous avons pu nous entraîner sur le langage Python, un langage très important que nous avons tous les deux voulu exercer une fois de plus. Ce projet a été une découverte très accueillante, et constitue une expérience qui nous aidera fortement pour le futur.