École Nationale Supérieure d'Ingénieurs
Sud-Alsace

Spécialité Informatique et Réseaux

Rapport de Projet

Conception d'une IA pour le jeu Puissance 4 avec l'algorithme Alpha-Beta

Auteurs :
Ozkosar Enes
Lidalt Omar

Enseignant :
M. Weber

Janvier 2026

Table des matières

1. Introduction

Le Puissance 4 est un jeu de stratégie combinatoire à information parfaite, il n'y a aucun hasard et tout est visible. Bien que le premier joueur possède une stratégie gagnante, la taille de l'arbre des possibilités (environ $4,5 \times 10^{13}$ positions) rend impossible une résolution brute en temps réel sur un ordinateur personnel.

L'objectif de ce projet est de concevoir une Intelligence Artificielle capable de jouer de manière compétitive contre un humain ou une autre IA. Pour relever ce défi, nous avons implémenté l'algorithme **Minimax** optimisé par la technique de l'élagage **Alpha-Beta**, couplé à une fonction d'évaluation heuristique stratégique.

2. Architecture logicielle

L'application a été développée en Python en suivant le paradigme de la Programmation Orientée Objet. Elle s'articule autour de deux entités principales qui séparent la logique du jeu de son affichage.

2.1. La classe Board (Le Modèle)

Cette classe représente le "cerveau" du jeu. Elle est indépendante de l'interface graphique et contient toute la logique des règles du Puissance 4 :

- **La Grille :** Stockée sous forme d'un tableau numérique (via la bibliothèque `numpy`) pour optimiser les accès mémoire.
- **La Gestion des coups :** Elle vérifie si une colonne est pleine et génère la liste des coups valides pour l'IA.
- **L'Analyse :** C'est elle qui contient la fonction de détection de victoire et la fonction d'évaluation heuristique utilisée par l'IA.

2.2. La classe Connect4 (Le Contrôleur/Vue)

Cette classe gère le déroulement de la partie et l'interaction avec l'utilisateur :

- Elle gère l'alternance des tours.
- Elle met à jour l'affichage graphique via la bibliothèque `Tkinter`.
- Elle capture les clics de la souris pour les coups du joueur humain.

2.3. Justification de l'approche séquentielle

Nous n'avons pas utilisé le multiprocessing pour l'algorithme alpha–bêta car l'efficacité de cet algorithme repose sur le partage dynamique des bornes α et β entre les différentes branches de l'arbre de recherche.

Or, dans un contexte multiprocessus, chaque processus possède sa propre mémoire et ne peut pas partager efficacement ces bornes avec les autres sans introduire des mécanismes de synchronisation complexes et coûteux. En conséquence, l'exploration parallèle des coups au niveau racine empêcherait les élagages entre branches, réduisant fortement l'intérêt de l'alpha–bêta et se rapprochant d'un simple minimax parallèle.

Nous avons donc privilégié une implémentation séquentielle, plus cohérente avec son principe de fonctionnement et suffisante pour les profondeurs de recherche visées dans ce projet.

3. L'Algorithme Alpha-Beta

3.1. Principe du Minimax

L'algorithme repose sur la théorie des jeux à somme nulle. Il simule l'arbre des coups futurs en alternant deux rôles distincts :

- **Le Joueur MAX :** Son objectif est de maximiser le score final. Il choisira toujours la branche qui lui rapporte le plus de points.
- **Le Joueur MIN :** L'algorithme part du principe que l'adversaire joue parfaitement. Son but est de minimiser le score de l'IA.

Cette alternance permet à l'IA de prévoir les meilleures parades de l'adversaire et de ne pas tomber dans des pièges simples.

3.2. Optimisation : Élagage et Tri des Coups

Parcourir tout l'arbre est trop lent. Nous utilisons l'élagage Alpha-Beta pour "couper" les branches inutiles. Si l'IA trouve un coup catastrophique dans une branche, elle arrête immédiatement de l'explorer, car elle sait déjà qu'elle ne choisira jamais cette option (ou que l'adversaire ne nous laissera jamais arriver à une bonne option derrière).

Pour rendre cet élagage plus efficace, nous avons optimisé l'ordre de recherche. Au lieu de tester les colonnes de gauche à droite (0 à 6), notre algorithme teste d'abord la **colonne centrale**, puis s'écarte vers les bords. Comme le centre est souvent le meilleur coup, l'algorithme trouve une bonne solution très vite et peut ainsi éliminer (élaguer) les autres possibilités beaucoup plus rapidement.

4. Stratégie et Heuristique

Puisque l'ordinateur ne peut pas calculer jusqu'à la fin de la partie (profondeur max de 42 coups), il doit s'arrêter avant (par exemple à 4 ou 6 coups d'avance) et "noter" la grille. C'est le rôle de la fonction d'évaluation.

4.1. Méthode d'Analyse : La Fenêtre Glissante

Pour évaluer une grille, notre algorithme utilise une technique de balayage. Une "fenêtre" virtuelle de 4 cases parcourt l'intégralité du plateau :

1. Horizontalement (lignes).
2. Verticalement (colonnes).
3. En diagonale (montante et descendante).

Pour chaque fenêtre de 4 cases, on regarde combien de pions nous possédons et combien l'adversaire en possède.

4.2. Pondération et Stratégie Défensive

Chaque configuration rapporte un certain nombre de points. Voici le barème que nous avons défini :

Configuration	**Score**	**Signification**
4 pions alignés (IA)	+100 000	**Victoire.** Priorité absolue.
3 pions Adversaire + 1 vide	**−120**	**Danger Critique.** Il faut bloquer immédiatement.
3 pions IA + 1 vide	+100	**Attaque.** Bonne opportunité de gagner.
2 pions IA + 2 vides	+10	**Construction.** Développement du jeu.
Pion au centre	+3	**Contrôle.** Bonus positionnel.

TABLE 1 – Table des poids stratégiques

Pourquoi ce choix ? Nous avons volontairement donné un poids plus important à la défense (**−120**) qu'à l'attaque (+100). Cela rend notre IA **prudente**. Si elle a le choix entre "tenter de gagner" (mais risquer de perdre au tour suivant) et "bloquer l'adversaire", le calcul mathématique ($100 - 120 = -20$) lui indiquera que le résultat est négatif. Elle choisira donc systématiquement la sécurité (le blocage).

5. Conclusion

Ce projet nous a permis de mettre en pratique les concepts fondamentaux de l'Intelligence Artificielle. L'implémentation de l'algorithme Alpha-Beta, renforcée par une heuristique privilégiant la défense et une optimisation positionnelle (contrôle du centre), a abouti à une IA robuste.

Elle est capable de calculer plusieurs coups à l'avance sans ralentir le jeu, offrant ainsi une opposition crédible et difficile à battre pour un joueur humain occasionnel.