RUNSER Lucas
KOHLER Jarod

2025-2026
2A IR

Création d'une IA pour le jeu de puissance 4 utilisant l'algorithme α–β

ensisa
École nationale supérieure d'ingénieurs sud Alsace

UNIVERSITÉ HAUTE-ALSACE

Sommaire

1. Introduction

L'objectif de ce projet est d'implémenter une intelligence artificielle capable de jouer au Puissance 4. Le jeu étant déjà fonctionnel pour deux joueurs humains, le travail consiste à intégrer une IA dans une architecture existante, sans remettre en cause le fonctionnement global de l'application.

L'algorithme utilisé repose sur la recherche adversariale, plus précisément sur l'algorithme α–β, qui est une optimisation du Minimax. L'IA doit être capable d'anticiper plusieurs coups à l'avance afin de choisir un coup pertinent, tout en conservant des temps de calcul raisonnables.

Un autre objectif important du projet est la mise en place d'une fonction d'évaluation permettant d'attribuer un score à une grille non terminale. Dans cette fonction d'évaluation nous allons utiliser diverses heuristiques que l'on aura codées de manière arbitraire et qui serviront de "guide" à l'ia afin d'influencer ses choix et d'optimiser ses chances de victoires.

2. Algorithme α–β

L'intelligence artificielle repose sur l'algorithme α–β, qui est une optimisation du Minimax. Il permet d'explorer l'arbre des coups possibles tout en réduisant le nombre d'états évalués grâce à l'élagage.

Le principe consiste à alterner des niveaux maximisants (IA) et minimisants (adversaire), en supposant que chaque joueur joue de manière optimale. Les valeurs α et β représentent respectivement les meilleures bornes connues pour chaque joueur. Lorsqu'une branche ne peut plus influencer le résultat final, elle est ignorée.

L'implémentation utilisée s'appuie sur celle réalisée en TD sur le jeu du Morpion, adaptée au Puissance 4. Les principales différences concernent le facteur de branchement plus élevé, ainsi que les conditions de victoire spécifiques au jeu (alignement de quatre pions).

La profondeur de recherche est fixée par le niveau de l'IA choisie, afin de trouver un compromis entre qualité de jeu et temps de calcul.

3. Implémentation et intégration

L’algorithme α–β est intégré directement dans le déroulement du jeu existant. Lorsqu’un tour doit être joué par l’IA, l’état courant de la grille est utilisé comme racine de l’arbre de recherche.

Pour chaque coup valide, une simulation est effectuée jusqu’à atteindre soit un état terminal (victoire, défaite, match nul), soit la profondeur maximale. Les méthodes déjà présentes dans le code sont utilisées pour :

- générer les coups possibles
- appliquer un coup sur la grille
- détecter les fins de partie

Cette approche permet de ne pas dupliquer la logique du jeu et de garantir que l’IA respecte strictement les règles existantes. L’algorithme se contente de prendre des décisions à partir des outils fournis par l’architecture initiale.

4. Stratégie d’évaluation des grilles incomplètes

On a conçu une IA qui se base sur plusieurs heuristiques pour jouer. Dans le jeu du puissance 4 le but est de former des lignes, colonnes et diagonales de 4 pions d’affilés pour gagner. On a donc codé la première heuristique `heuristique_column_line_value` qui se base sur le nombre de puissance 4 qu’il est possible de faire à partir d’une case, on se sert de cette info afin d’assigner un poids arbitraire à chaque ligne et colonne. Cela encourage donc l’IA à jouer proche du centre en priorité.

Voici un tableau dénombrant le nombre de puissance 4 possible via chaque case d’un tableau de jeu :

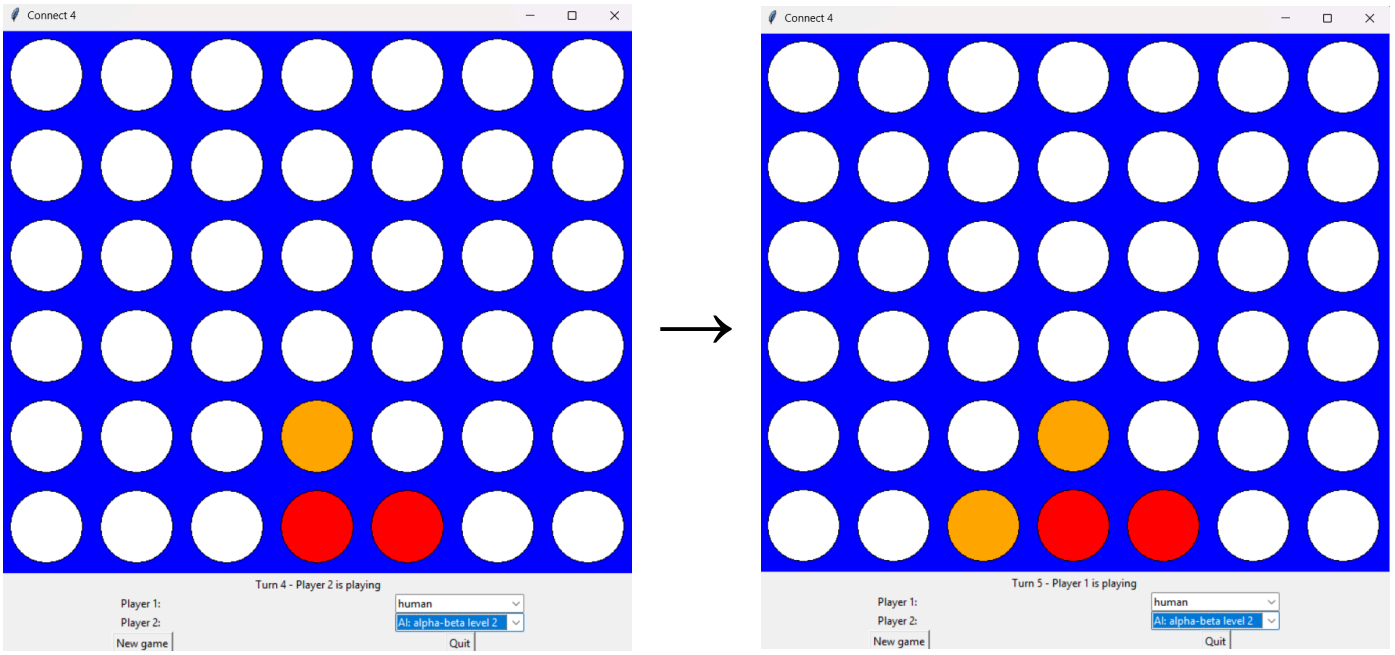
3	4	5	7	5	4	3
4	6	8	10	8	6	4
5	8	11	13	11	8	5
5	8	11	13	11	8	5
4	6	8	10	8	6	4
3	4	5	7	5	4	3

Nous avons aussi créé deux heuristiques appelées :

- `heuristique_2_aligner`
- `heuristique_3_aligner`

Ces deux heuristiques servent à la gestion de la défensivité et l'agressivité de l'IA. L'IA sera récompensée si son prochain coup permet d'aligner 2 ou 3 pions, ce qui lui permet sur le long terme de créer des situations qui piègent l'adversaire.

La partie défensive s'explique dans le sens ou l'IA va être récompensée si son prochain coup bloque un potentiel alignement de 2 ou 3 pions adverse. En pratique, l'IA aura tendance à jamais laisser l'adversaire aligner plus de 2 pions d'affilés. Voici un exemple qui illustre ce que fait l'heuristique :



Et enfin nous avons l'heuristique `heuristique_defaite_victoire` qui permet simplement de forcer l'IA à se défendre en cas de défaite imminente, et de gagner en cas d'opportunité de victoire immédiate.

L'IA détecte aussi le nombre de case vide dans le tableau et à partir de ça va déterminer si elle se situe plutôt dans le early, mid ou late game. Ces 3 états permettent de donner un poids différent aux heuristiques en fonction de l'état d'avancement de la partie.

Par exemple, en fin de partie la notion de "case du milieu représente un avantage" devient obsolète car la partie défense/attaque est bien plus importante pour gagner, d'autant plus que les cases du milieu sont probablement déjà prises d'ici la fin de la partie.

Le choix des poids était arbitraire, nous avons simplement ajouté un multiplicateur au moment de la prise en compte d'un score calculé par une heuristique :

```
# early game
if filled <= 10:
    score += heuristique_column_line_value(self, player, opponent) * 4
    score += heuristique_3_aligner(self, player, opponent) * 3
    score += heuristique_2_aligner(self, player, opponent)

# mid game
elif filled <= 30:
    score += heuristique_column_line_value(self, player, opponent)
    score += heuristique_3_aligner(self, player, opponent) * 3

# late game
else:
    score += heuristique_3_aligner(self, player, opponent)

return score
```

5. Résultats et limites

Pour des profondeurs de recherche allant jusqu'à 5 ou 6, l'IA joue de manière stable et cohérente face à un joueur humain. Lors des tests, elle ne se limite pas à des réactions immédiates, mais choisit des coups qui améliorent sa position globale sur la grille, aussi bien pour attaquer que pour se défendre.

Au-delà d'une profondeur de 6, les temps de calcul deviennent trop importants pour permettre une partie fluide. Malgré l'utilisation de l'algorithme α–β, le facteur de branchement élevé du Puissance 4 limite fortement la profondeur exploitable en pratique.

Les performances observées restent donc dépendantes du compromis entre profondeur de recherche et temps de calcul, ainsi que de la qualité de l'évaluation des grilles non terminales.

Grâce à l'introduction du multiprocessing, certaines branches de l'arbre de recherche peuvent être évaluées en parallèle, ce qui permet de conserver des temps de calcul acceptables jusqu'à une profondeur 7.

À partir du niveau 8, le temps de calcul augmente fortement, et devient très important pour des profondeurs supérieures, en raison de la croissance exponentielle de l'arbre de recherche et des limites matérielles de la machine utilisée. Par exemple, pour une profondeur 10, le temps nécessaire pour calculer un coup dépasse les 10 minutes.

6. Conclusion

Ce projet a permis d'implémenter une intelligence artificielle basée sur l'algorithme α–β pour le jeu du Puissance 4. L'IA s'intègre correctement dans l'architecture existante et est capable de jouer de manière compétitive face à un joueur humain ou une autre IA.

Les résultats obtenus montrent que la qualité de jeu dépend principalement de la profondeur de recherche et de la stratégie d'évaluation des grilles incomplètes. Ce projet met ainsi en évidence le rôle central de la fonction d'évaluation dans les performances réelles d'une IA de type recherche adversariale ainsi que son lien avec la profondeur de recherche.

Un être humain n'a pas en temps normal sans entraînement les capacités cognitives nécessaires pour visualiser une partie avec beaucoup de coups à l'avance. Nous n'avons pas réussi à vaincre l'IA de profondeur 6, on peut donc conclure que la qualité de la méthode eval() est bonne et que les choix des heuristiques étaient judicieux.