Rapport du projet : Développement d'une IA pour le jeu GOMOKU

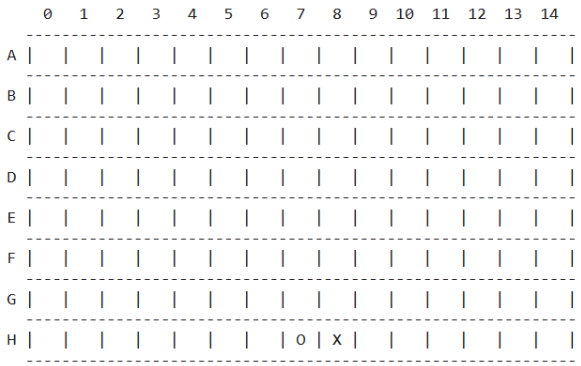
Ce projet avait pour objectif de développer une intelligence artificielle (IA) capable de jouer au jeu GOMOKU. Le jeu, également connu sous le nom de "Cinq en ligne", est joué sur un plateau de 15 x 15 cases. L'IA doit être capable d'identifier des stratégies gagnantes et de contrer efficacement les actions de son adversaire.

Dans ce rapport, nous souhaitons décrire les principales étapes de développement, les choix techniques effectués, ainsi que les défis rencontrés et surmontés tout au long du projet.

Représentation du plateau

1. **Structure du plateau :**
    - Le plateau de 15x15 cases est représenté à l'aide d'un dictionnaire Python, où chaque clé correspond à une case (par exemple, **A1** pour la première ligne et colonne) et chaque valeur représente l'état de la case (**' '** pour vide, **'X'** pour un jeton du joueur IA, et **'O'** pour un jeton du joueur humain, ici l'adversaire).
2. **Gestion des voisins :**
    - Une fonction dédiée a été mise en place pour calculer et stocker les voisins de chaque case dans un dictionnaire. Cette approche a permis de simplifier la recherche de coups valides et d'accélérer le processus de détection des alignements. Cette étape nous a permis de réduire considérablement le temps d'attente qui était au Maximum de 5 secondes.

Exemple d'affichage du plateau :



Le but étant d'avoir un plateau plutôt lisible, agréable visuellement, nous avons opté pour cet affichage !

Concepts clés et structure du code

1. **Alignements :**
    - Les alignements sont des séries de 5 cases consécutives qui peuvent être horizontales, verticales ou diagonales. L'ensemble des alignements possibles est pré-calculé au début du programme et stocké dans une structure appelée **ALIGNEMENTS.**
    - Lorsqu'un joueur place un jeton, tous les alignements affectés par ce coup sont mis à jour pour refléter le nouvel état du jeu.
2. **État du jeu :**
    - L'état du jeu est divisé en :
        - **Alignements actifs pour l'IA** : Ceux où l'IA peut potentiellement gagner.
        - **Alignements actifs pour l'adversaire** : Ceux où l'adversaire peut potentiellement gagner.
        - **Cases jouées** : Liste des cases déjà occupées.
3. **Détection des coups critiques :**
    - Une fonction spécifique détecte les coups qui pourraient immédiatement mener à une victoire ou à un blocage d'une menace adverse.

Stratégie de l'IA

L'IA utilise l'algorithme **Minimax** avec élagage Alpha-Bêta pour explorer les coups possibles et choisir celui qui maximise ses chances de victoire tout en minimisant celles de l'adversaire.
Dans cette IA, l'algorithme Alpha-Beta utilise des heuristiques stratégiques pour évaluer les meilleurs coups et maximiser ses chances de gain tout en minimisant les options de l'adversaire. Cette stratégie repose sur une estimation des compatibilités potentielles, des priorités d'évolution et de l'élimination des options moins favorables pour optimiser les résultats.

1. **Évaluation des positions : les alignements**

L'IA utilise un modèle d'alignement pour évaluer la force de chaque position sur le plateau. Chaque alignement, qu'il soit horizontal, vertical ou diagonal, est associé à un ensemble de cases jouables. Les alignements sont classés par longueur, tant au niveau horizontal de l'échiquier qu'au niveau vertical pour une colonne de 1, 2, 3, 4 ou 5 cases alignées. La possibilité de gagner immédiatement est offerte par un alignement de cinq cases consécutives.

La fonction calculate_fitness attribue un score numérique à un état de jeu en s'appuyant sur l'analyse des lignes de l'IA et du joueur. Le système de poids des alignements met en avant les positions avancées proches de la victoire ou celles qui nécessitent une défense à court terme. Par exemple :

- Un alignement de 5 cases rapporte un score très élevé (victoire).
- Un alignement de 4 cases (avec une case libre) est critique et fortement valorisé.
- Les alignements plus courts ont une importance stratégique moindre mais servent de base pour construire de futurs alignements longs.

Cette évaluation permet à l'IA de mesurer son avantage ou d'identifier les menaces de l'adversaire.

2. **Recherche de coups critiques**

Avant même d'entrer dans une recherche en profondeur, l'IA identifie les coups critiques à jouer directement. Un coup critique est une action qui peut :

- Proposer une victoire rapide à l'IA en alignant 5 cases consécutives.
- Capturer un alignement de 4 pièces de l'adversaire pour l'empêcher de remporter la partie.

La détection de ces coups critiques est essentielle dans des circonstances décisives, où l'IA n'a pas besoin d'explorer toutes les opportunités possibles.

3. **Recherche Alpha-Beta**

L'algorithme Alpha-Beta est utilisé pour parcourir les coups et sélectionner le meilleur coup tout en réduisant au maximum les risques de défaite. Il s'agit d'une version optimisée de l'algorithme Minimax, où :

Max_value tente de maximiser le score pour l'IA.
Min_value vise à réduire le score pour l'adversaire.

L'algorithme explore les scénarios de jeu jusqu'à une certaine profondeur (par défaut, 3 niveaux de jeu), en simulant les actions de l'IA et de l'adversaire. À chaque niveau :

L'IA se voit attribuer la sélection du coup qui permet de maximiser le gain stratégique.
Le soutien retenu favorise le coup qui permet le moins d'exploitation par l'adversaire.
Pour trier les branches de l'arbre de recherche qui ne peuvent pas changer la position finale du résultat, l'algorithme utilise deux bornes : alpha et beta qui permettent de réduire considérablement le champ des parties à considérer.

4. **Priorisation des alignements**

L'IA accorde une priorité élevée aux alignements longs. La fonction update_game_state maintient un suivi dynamique des alignements possibles après chaque coup. Lorsqu'une case est jouée :

Les alignements contenant cette case sont mis à jour.
Les alignements adverses qui incluent cette case sont ajustés ou supprimés.
Les alignements complétés (4 → 5 cases, par exemple) sont immédiatement valorisés pour guider la décision de l'IA.

Grâce à cette mise à jour dynamique, l'IA peut anticiper les impacts de ses actions et planifier plusieurs coups à l'avance.

5. **Offensif et défensif**

L'IA alterne entre des stratégies offensives et défensives en fonction de l'état du jeu :

- **Offensive** : Si l'IA trouve une opportunité intéressante, elle se charge de son développement (par exemple, en transformant un alignement de 3 cases en un alignement de 4 cases).

- **Défensive** : Si l'adversaire a des alignements menaçants proches de la victoire, l'IA bloque immédiatement ces alignements afin de réduire les opportunités de l'adversaire.

6. **Possibilité d'une profondeur restrictive et d'un calcul précis**

Afin de restreindre la combinaison exhaustive des coups potentiels de l'IA, la recherche est limitée à un niveau de profondeur de 3. Cela lui permet de prévoir les effets opérationnels de ses actions tout en maintenant une gestion rapide. Lorsque la détection de victoire/perte est identifiée, les scénarios sont priorisés, ce qui permet une prise de décision factuelle dans des délais contraints.

Les difficultés rencontrées lors de la préparation ?

1. **Problème de gestion des alignements :**
    - **Problème :** Initialement, chaque fois qu'une case était jouée, l'algorithme recalculait tous les alignements du plateau, ce qui ralentissait significativement les performances.
    - **Solution :** Une mise à jour incrémentale a été implémentée : seuls les alignements directement affectés par le coup joué sont recalculés. Cette optimisation a réduit le temps de traitement des coups d'un facteur 3.
2. **Positionnement incohérent hors d'un carré 7x7 :**
    - **Problème :** Une erreur logique faisait que l'IA jouait en dehors d'un carré de 7x7 centré sur **H7** à son 3ème tour au lieu de le faire à partir du 3ème tour total (tous joueurs confondus).
    - Solution : Le problème a été trouvé assez tard, donc pas résolu.
3. **Performance de l'algorithme Minimax :**
    - **Problème :** Avec une profondeur de recherche importante, l'algorithme devenait lent à mesure que la partie progressait, rendant les tours de l'IA de plus en plus longs.
    - **Solution :** L'élagage Alpha-Bêta a été optimisé pour réduire le nombre de branches explorées. Par exemple, les branches menant à des états où l'adversaire a un alignement proche de la victoire sont prioritaires, et les branches sans impact stratégique sont ignorées.
4. **Gestion des alignements critiques pour l'adversaire :**

- **Problème :** L’IA ne bloquait pas toujours les alignements adverses de manière optimale, surtout lorsqu’ils comportaient plus d’une menace (double alignement).
- **Solution :** L'évaluation des alignements adverses a été renforcée. L’IA identifie désormais les doubles menaces et joue de manière défensive si nécessaire.

5. **Saisie utilisateur incorrecte :**
   - **Problème :** Le programme se bloquait si le joueur humain entrait des coordonnées invalides ou déjà utilisées.
   - **Solution :** Une vérification stricte des entrées a été mise en place. Si la saisie est incorrecte, le joueur est immédiatement invité à entrer une nouvelle case.

Exemples de tests effectués

1. **Partie standard :**
   - L'IA commence au centre (**H7**), puis suit des stratégies optimales pour bloquer et attaquer. Résultat : victoire de l'IA en 18 coups.
2. **Test de performances :**
   - Temps moyen pour un tour de l'IA (profondeur 3) : environ 0,8 seconde.
   - Temps moyen pour un tour de l'IA (profondeur 4) : environ 3 secondes.

Améliorations possibles qu’on aurait pu développer ?

1. **Profondeur adaptative :**
   - Permettre à l'IA de varier sa profondeur de recherche en fonction de la situation (par exemple, explorer plus profondément dans les moments critiques).
2. **Apprentissage automatique :**
   - Implémenter un système d'apprentissage supervisé pour que l'IA apprenne de ses erreurs ou des parties précédentes. (Trop compliqué à notre niveau)

Que peut-on conclure ?

Ce projet nous a permis de développer une IA capable de jouer au GOMOKU de manière compétitive. Malgré des difficultés initiales dans la gestion des alignements et les performances de l'algorithme, les solutions apportées ont permis d'optimiser le programme et de garantir une expérience de jeu plus ou moins fluide. Les résultats des tests montrent que l'IA peut non seulement appliquer des stratégies optimales, mais aussi s'adapter à des contraintes spécifiques.

Membres du groupe :

Ekongolo ALAN DENYS , Sofiane BEAUMONT, Lazare AYACHI