Université Paris Cité

Rapport de projet d'IA : le Gomoku

Intelligence Artificielle L3

Anaïs Kadic
Athénaïs Gravil

Enseignante : Mme Elise Bonzon

Contents

1 Introduction

1.1 Présentation du jeu Gomoku

Gomoku, aussi appelé "Five in a Row", est un jeu de stratégie qui se joue à deux, sur un plateau de Go.

Malgré la simplicité apparente de ses règles, Gomoku se distingue par sa complexité stratégique, ce qui en fait un sujet d'étude intéressant pour les stratèges et les amateurs de jeux de réflexion.

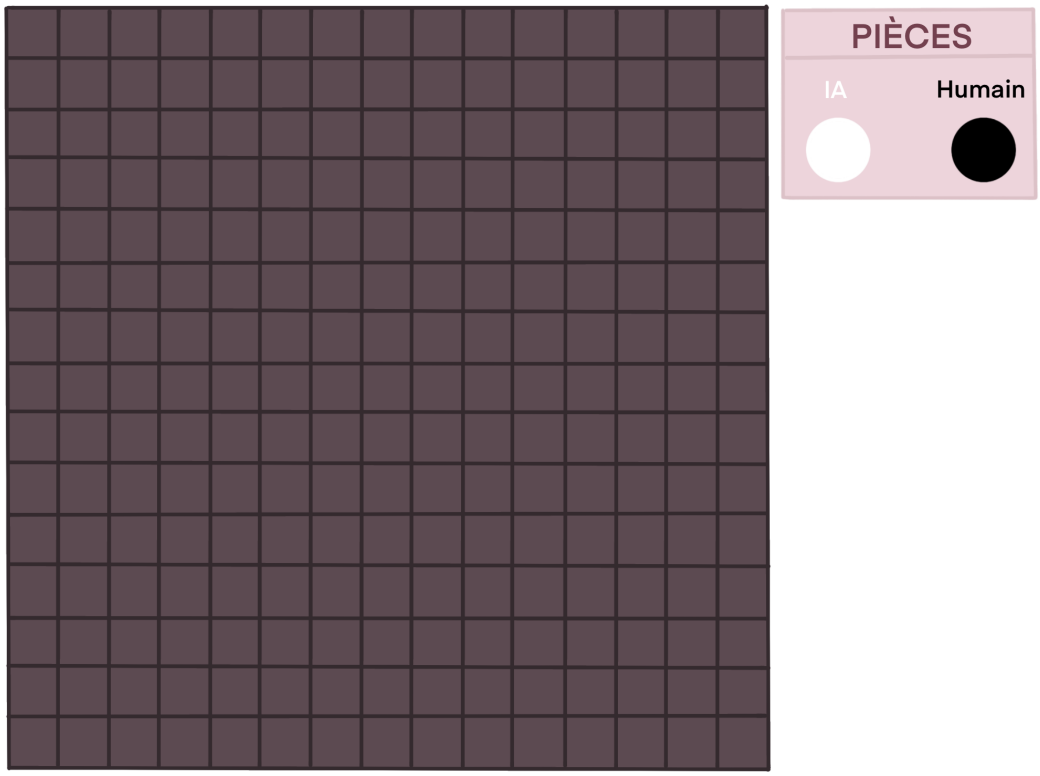


Figure 1: Plateau de jeu et pièces du Gomoku

Règles du jeu:

Gomoku se joue sur un plateau de Go standard, constitué d'un quadrillage de 15x15 intersections. Voici les règles principales :

- ***Initialisation de la partie :*** **Le joueur qui a les pierres noires commence toujours.**
- ***Placement des pierres :*** **Les deux joueurs placent, tour à tour, une pierre de leur couleur (noir ou blanc) sur une intersection vide du plateau.**
- ***Objectif :*** **Le but est d'aligner cinq pierres de sa couleur, avant l'adversaire, que ce soit à l'horizontale, à la verticale ou bien en diagonale.**
- ***Fin de partie :*** **La partie se termine dès qu'un joueur a aligné cinq de ses pierres, il est alors déclaré vainqueur. En revanche, si toutes les intersections sont remplies sans qu'aucun des deux joueurs n'aient gagné, la partie est déclarée nulle.**

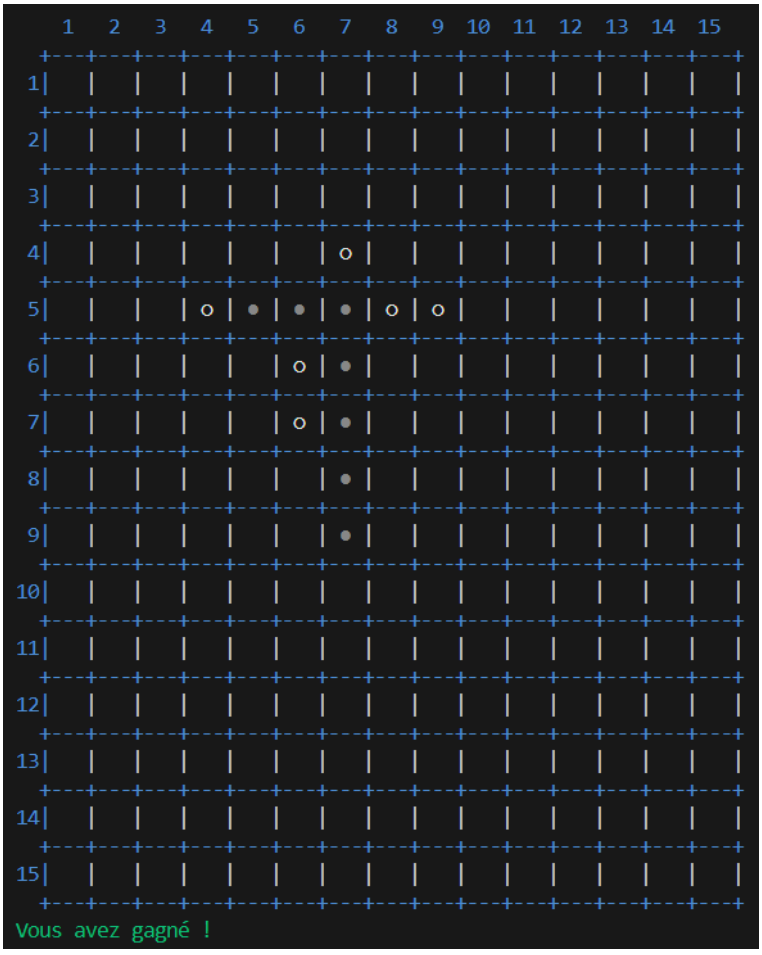


Figure 2: Plateau avec une partie finie

1.2 Choix du language de programmation

Nous avons choisi Python pour réaliser ce projet car grâce à son efficacité et sa clarté, il offre une syntaxe intuitive et facilite la programmation des algorithmes.
De plus, ses bibliothèques incluent des modules pour la gestion de structures de données. C'est donc un langage idéal pour le développement d'applications, en intelligence artificielle.

1.3 Objectif du projet

L'objectif de ce projet est d'implémenter une intelligence artificielle, pour le jeu Gomoku, capable de jouer contre un joueur humain. Ce jeu doit comporter différents niveaux de difficulté. Nous avons suivi les étapes suivantes :

Étape	Description
1	Développer un moteur de jeu Gomoku fonctionnel pour permettre des parties entre deux joueurs humains.
2	Implémenter l'algorithme Minimax pour permettre le jeu contre l'ordinateur.
3	Intégrer l'élagage $\alpha\beta$ à l'algorithme Minimax pour améliorer les performances de l'IA, en réduisant le nombre de nœuds évalués.
4	Créer plusieurs niveaux de difficulté pour l'IA, en ajustant la profondeur de recherche et les fonctions d'évaluation utilisées par l'algorithme Minimax.
5	Organiser des tournois entre les différentes configurations de l'IA pour évaluer leur stratégie en situation de jeu.

2 Architecture du système

2.1 Description de la structure du code

Nous avons organisé le projet à l'aide de modules, en utilisant la programmation orientée objet (POO).
Nous avons structuré les répertoires et les fichiers du projet de la manière suivante :

- GOMOKU/ – **Racine du projet.**
  - /joueur – **Modules liés aux joueurs.**
    - `joueur.py` – **Définit la classe de base pour un joueur.**
    - `joueur_ia.py` – **Définit la classe pour les joueurs IA, implémentant des stratégies spécifiques.**
  - /plateau – **Module pour gérer le plateau de jeu.**
    - `plateau.py` – **Gère l'état du plateau de jeu et la vérification des règles.**
  - /strategie – **Modules qui implémentent les algorithmes de décision pour l'IA.**
    - `evaluation.py` – **Fonctions d'évaluation utilisées par l'IA.**
    - `minmax.py` – **Implémente l'algorithme Minimax.**
    - `transposition_table.py` – **Optimise Minimax avec une table de transposition.**
  - /tournoi – **Modules qui gèrent les tournois entre les différentes IAs.**
    - `gestionnaire_tournoi.py` – **Planification et enregistrement des résultats des matchs.**
    - `simulateur_partie.py` – **Simule les parties entre deux IAs.**
  - `main.py` – **Script principal qui lance le jeu et gère les interactions utilisateurs.**

3 Optimisation des décisions de l'IA

3.1 Algorithme Minimax

Principe de base de l'algorithme Minimax

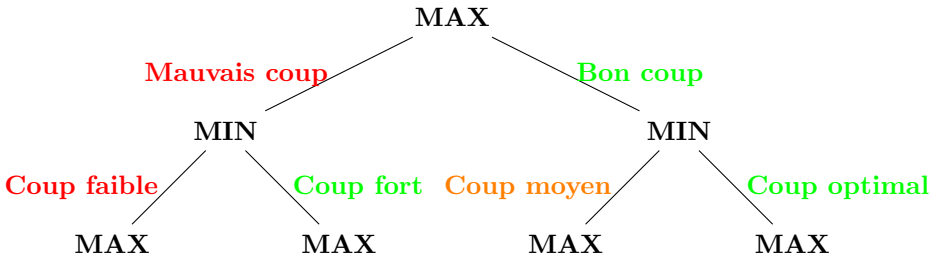
L'algorithme Minimax est une méthode fondamentale en théorie des jeux. Cet algorithme est utilisé pour les jeux à deux joueurs à somme nulle, où chaque avantage obtenu par un joueur représente une perte directe pour l'autre.

Concept de Minimax **: Le principe de Minimax repose sur la stratégie où chaque joueur essaie de minimiser la perte maximale possible, en anticipant les pires scénarios et en maximisant son propre gain. En d'autres termes, MAX essaie d'atteindre le score le plus élevé possible à chaque coup, tandis que MIN s'efforce de réduire ce score**

Voici comment cela fonctionne :

1. **Joueurs : Les joueurs sont identifiés comme MAX, qui cherche à maximiser son score, et MIN, qui tente de minimiser le score de MAX.**
2. **Arbre de décision : Le jeu est représenté par un arbre où chaque nœud est un état du jeu et chaque niveau un tour de jeu.**
3. **Choix stratégiques : À chaque niveau, le joueur concerné sélectionne le meilleur coup selon sa stratégie, maximisant ou minimisant le score, en fonction de son rôle.**

Visualisation des stratégies



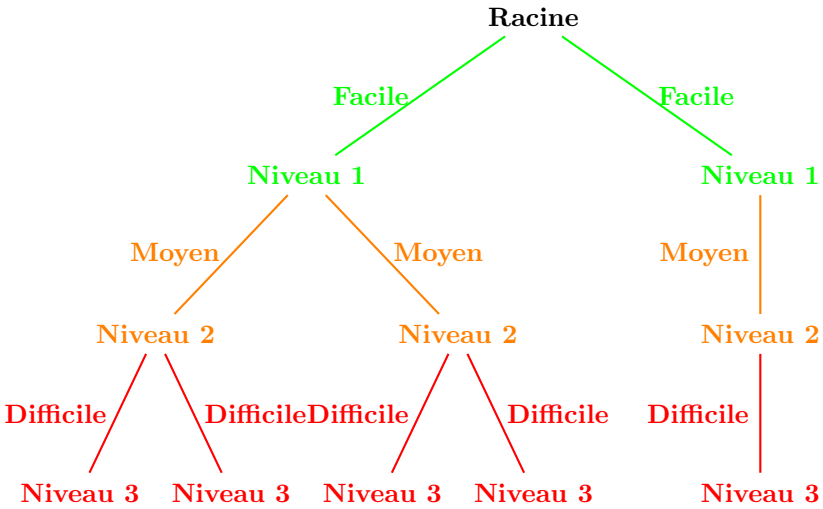
Application dans Gomoku

L'intégration de l'algorithme Minimax dans Gomoku permet à l'IA d'anticiper les coups adverses et d'optimiser ses propres stratégies.

Adaptation à Gomoku

Dans Gomoku, Minimax est utilisé pour évaluer les conséquences de chaque coup potentiel en attaque et en défense. La taille du plateau de Gomoku (15x15) et l'alignement de cinq pierres, d'une même couleur, complexifient l'application de l'algorithme.

- `Profondeur de recherche variable` **: En fonction du niveau de difficulté sélectionné, la profondeur de recherche de l'algorithme Minimax est ajustée. Pour les niveaux plus difficiles, l'IA explore plus profondément dans l'arbre de jeu.**

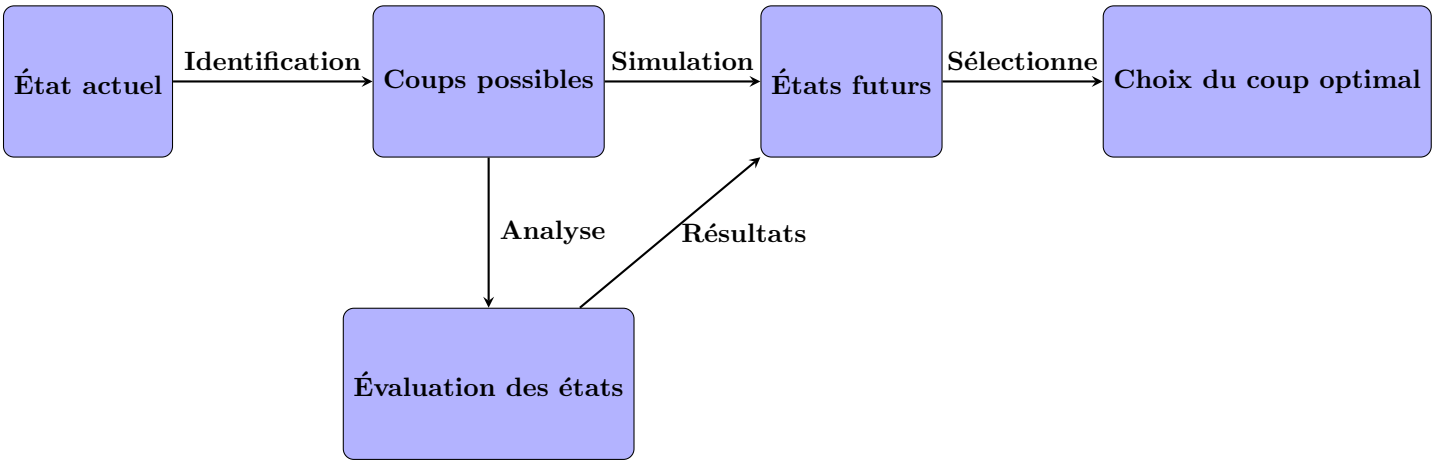


- `Utilisation de la table de transposition` **: Une table de transposition est utlisée pour optimiser la performance de Minimax, en mémorisant et réutilisant les évaluations des configurations déjà calculées, afin d'accélérer la prise de décision.**
- `Génération de coups possibles` **: L'IA génère des coups en ciblant les espaces adjacents aux pierres déjà placées, ce qui limite l'espace de recherche aux zones pertinentes du plateau.**

3.2 Implémentation technique

Principe de prévision et contre-mesures

L'algorithme Minimax, fonctionne en anticipant les futurs mouvements possibles sur le plateau. Un arbre de jeu est construit, où chaque nœud représente un état potentiel du plateau après un coup donné. À chaque tour, l'IA explore cet arbre jusqu'à la profondeur spécifiée, selon le niveau de jeu choisi.



Optimisation des décisions

Plutôt que de se limiter à évaluer les gains immédiats de chaque coup, l'IA utilise une méthode d'évaluation dynamique.

Élagage Alpha-Beta

Pour renforcer l'efficacité de l'algorithme Minimax, l'élagage Alpha-Beta est utilisé. Cette technique réduit le nombre de scénarios que l'IA doit évaluer, en sélectionnant des branches:

- `Définition des variables Alpha et Beta` **: Ces valeurs servent de limites pour identifier les branches de l'arbre qui ne nécessitent pas d'exploration supplémentaire.**
- `Application dans le choix des coups` **: Si le score potentiel d'un coup est hors des limites définies par Alpha et Beta, l'IA conclut que ce coup ne modifiera pas le résultat final optimal et l'élimine de l'analyse.**
- `Impact sur la performance` **: En supprimant les branches inutiles, l'élagage Alpha-Beta accélère significativement le processus de prise de décision : c'est une amélioration essentielle dans les jeux complexes, comme Gomoku.**

Alpha et Beta

Alpha est le meilleur score que MAX peut garantir jusqu'à présent.
Beta est le pire score que MIN peut assurer jusqu'à présent.

4 Stratégies d'évaluation de l'IA

La classe `Evaluation` utilise des méthodes d'évaluation pour ajuster le niveau de difficulté de l'IA.

Ces méthodes sont conçues pour analyser l'état du plateau et calculer un score basé sur la disposition des pierres, favorisant des mouvements qui maximisent les chances de victoire de l'IA tout en minimisant les opportunités pour l'adversaire.

4.1 Niveaux de difficulté

- ### Très Facile

  - **Ce mode a été le premier implémenté lors du développement du projet pour tester le comportement de base de l'IA.**

Logique du niveau Très Facile

Dans le mode Très Facile, l'IA utilise la fonction `evaluer_tres_facile` pour évaluer le plateau.Cette fonction compte le nombre de pierres alignées directement pour la couleur spécifiée.
Elle parcourt chaque position du plateau et, pour chaque pierre correspondant à la couleur de l'IA, elle vérifie combien de pierres consécutives de la même couleur se trouvent dans les quatre directions : horizontal, vertical, et les deux diagonales.

- **La méthode `compter_pierres_alignees` est appelée pour chaque pierre trouvée. Elle initialise un compteur d'alignement à 1 pour chaque direction et itère jusqu'à quatre pierres supplémentaires dans chaque direction, ajoutant chaque pierre consécutive à l'alignement.**
- **Pour chaque direction, si une suite de pierres de la même couleur est interrompue par une pierre adverse ou par la limite du plateau, le comptage s'arrête pour cette direction.**
- **Le score final, retourné par `evaluer_tres_facile`, est la somme totale des pierres alignées détectées sur tout le plateau.**

- **Facile**
  - **À ce niveau, l'IA commence à utiliser des heuristiques plus travaillées que dans le mode Très Facile.**

Logique du niveau Facile

Dans le mode Facile, l'IA utilise la fonction evaluer_facile pour analyser le plateau. Cette méthode combine une évaluation défensive, pour bloquer les menaces adverses, et une évaluation offensive, pour identifier et exploiter les opportunités d'attaque.

- ***Défense* : La méthode `evaluer_defense_simple` est utilisée pour détecter et bloquer les menaces adverses, en comptant les alignements potentiels de l'adversaire qui pourraient conduire à une victoire imminente. Elle attribue des scores négatifs élevés aux configurations menaçantes, incitant l'IA à les bloquer en priorité.**
- ***Attaque* : Si c'est le début d'une partie ou bien qu'aucune menace immédiate n'est détectée, l'IA passe à une posture plus offensive avec la méthode `evaluer_attaques_potentielles`, où elle évalue les meilleures opportunités d'attaques potentielles basées sur les configurations actuelles du plateau.**

Implémentation

- **`detecter_menace` est utilisée pour scanner le plateau et identifier les alignements adverses qui pourraient engendrer une victoire.**
- **`evaluer_attaque_potentielle` calcule un score pour chaque case vide du plateau, en évaluant le potentiel alignement à partir de cette position. Elle prend en compte les alignements directs et aussi le potentiel d'expansion de ces alignements.**

Limites et efficacité du niveau Facile

Bien que le mode Facile offre une amélioration par rapport au mode Très Facile en termes de capacité à répondre stratégiquement aux menaces et à saisir les opportunités d'attaque, il présente certaines limites :

- **Détection de menaces : L'IA se concentre principalement sur les menaces immédiates et directes, ce qui peut la rendre vulnérable à des stratégies adverses plus subtiles et planifiées sur plusieurs coups.**
- **Planification à long terme : L'IA ne planifie pas aussi profondément que dans les niveaux supérieurs.**
- **Efficacité temporelle : A mesure que la complexité du plateau augmente, l'efficacité de l'IA diminue.**

- # Moyen

Ici, l'IA intègre des stratégies plus difficiles, en combinant évaluation offensive et défensive.

Logique du niveau Moyen

Le mode Moyen représente un équilibre entre le jeu réactif et proactif, en utilisant des évaluations plus poussées pour chaque coup. L'IA à ce niveau utilise la fonction `evaluer_moyen` qui analyse le plateau avec une perspective plus globale et stratégique.

- **`Analyse combinée` : Cette approche évalue les positions offensives tout en gardant un œil sur les défenses nécessaires pour contrer les menaces adverses. Les coups sont choisis en fonction de leur potentiel à maximiser les bénéfices immédiats et à long terme.**
- **`Planification stratégique` : Contrairement aux niveaux précédents, l'IA au niveau moyen est capable d'anticiper les mouvements adverses plus complexes et de préparer des réponses stratégiques. Cela inclut l'extension de séquences existantes ou la création de multiples menaces qui peuvent se convertir en victoire.**

Implémentation

- **`evaluer_plateau` : Cette méthode est le cœur de l'évaluation à ce niveau, l'IA prend en compte les pierres adverses pour ajuster les tactiques défensives.**
- **`evaluer_alignement` : Calcule des scores pour des alignements spécifiques, ajoutant une pondération significative aux configurations qui peuvent rapidement mener à une victoire ou nécessiter une intervention défensive immédiate.**
- **`evaluer_besoins_defensifs` : Identifie et évalue les configurations adverses qui pourraient poser une menace sérieuse, permettant à l'IA de bloquer ou de contrer efficacement ces menaces.**

Limites et efficacité du niveau Moyen

Bien que nettement plus avancé que les niveaux précédents, le mode Moyen peut parfois manquer d'optimisation dans des situations de jeu complexes. L'efficacité de l'IA à ce niveau dépend fortement de sa capacité à équilibrer la défense et l'attaque, sans compromettre sa position globale :

- **Adaptabilité : L'IA peut parfois être lente à s'adapter à des changements rapides ou inattendus sur le plateau, surtout face à des adversaires qui utilisent des stratégies non conventionnelles.**
- **Complexité croissante : À mesure que le plateau devient plus complexe, l'IA peut ne pas toujours choisir le coup optimal, en particulier si elle doit évaluer un grand nombre de menaces potentielles.**

- # Difficile

  - **Le mode Difficile représente le niveau le plus difficile parmis les IAs implémentées pour Gomoku.**

Logique du niveau Difficile

Cette IA emploie des analyses et des algorithmes de recherche avancés comme l'élagage Alpha-Beta. Elle est conçue pour évaluer et optimiser ses stratégies à long terme, maximisant ses chances de former un alignement de cinq tout en bloquant l'adversaire.

- **Analyse complète : L'IA analyse le plateau en intégrant à la fois les menaces adverses et les opportunités offensives pour optimiser chaque coup. Elle prend en compte les alignements actuels mais aussi les configurations potentielles qui pourraient émerger, appliquant une stratégie proactive pour contrôler le jeu.**
- **Optimisation des coups : Les décisions sont prises en calculant les scores basés sur des méthodes d'évaluation poussées qui incluent des analyses de configurations menaçantes et des opportunités d'extension d'alignements.**

Implémentation

- **`analyser_menaces_multiples` : Cette fonction identifie et quantifie les menaces potentielles, permettant à l'IA de réagir efficacement pour les contrer. Elle recherche des alignements potentiellement gagnants de l'adversaire.**
- **`evaluer_alignement_potentiel` : Évalue le potentiel offensif d'un alignement à partir de chaque position, en calculant des scores qui favorisent des alignements stratégiques pouvant mener à une victoire rapide.**
- **`evaluer_formations_ouvertes` : Accentue les efforts sur la création et l'extension d'alignements ouverts, en augmentant les chances de former des lignes gagnantes tout en bloquant les tentatives adverses de faire de même.**

Limites et efficacité du niveau Difficile **Le niveau Difficile de l'IA peut parfois être confronté à des défis, notamment :**

- **Complexité de calcul : Les analyses approfondies requièrent beaucoup de ressources, ce qui peut entraîner des délais dans la prise de décision, notamment sur des plateaux de jeu très chargés.**

4.2 Conclusion sur les performances des différents niveaux d'IA

À travers les niveaux de difficulté implémentés pour l'IA de Gomoku, nous observons une progression claire en termes de complexité et de performance. Chaque niveau est conçu pour offrir un défi approprié à des joueurs de différentes compétences et pour répondre à des situations de jeu variées.

Niveau	Description
Très Facile	C'est le niveau le moins performant, conçu pour tester les fonctionnalités de base de l'IA. Il se concentre sur des réponses directes.
Facile	Offrant une amélioration par rapport au niveau Très Facile, cette IA commence à intégrer des stratégies basiques de défense et d'attaque. Toutefois, elle reste limitée par une capacité réduite à anticiper les mouvements futurs et à répondre à des stratégies complexes.
Moyen	Cette IA représente un équilibre entre réactivité et proactivité, avec des capacités d'anticipation et de planification améliorées.
Difficile	C'est le niveau le plus performant, utilisant des analyses complexes et des techniques de recherche avancées pour optimiser chaque coup.

5 Tournois entre configurations d'IA

5.1 Introduction

Des tournois entre différentes configurations d'intelligence artificielle ont été mis en place pour évaluer de manière concrète et comparative la performance de chaque IA, dans le jeu de Gomoku. Cette approche permet de simuler des situations de jeu réelles, où chaque niveau de difficulté de l'IA affronte les autres.

5.2 Aperçu de la gestion des tournois

Le système de tournoi repose sur deux classes principales :

- **`GestionnaireTournoi` : Cette classe est responsable de l'organisation des matchs, de la gestion des résultats et de la coordination générale des tournois.**
- **`SimulateurPartie` : Utilisée principalement pour simuler les parties entre deux intelligences artificielles.**

Classe GestionnaireTournoi

La classe `GestionnaireTournoi` gère la logistique des affrontements entre les joueurs IA, tout en maintenant les scores et en compilant les résultats de chaque match.

- **Initialisation :** Le constructeur de la classe prend en entrée la liste des joueurs (instances de `JoueurIA`) et le nombre de parties à jouer par match. À l'initialisation, un dictionnaire de scores est créé pour suivre les points accumulés par chaque joueur tout au long du tournoi.
- **Organisation des matchs :** La méthode `organiser_match` orchestre un match entre deux joueurs, en effectuant le nombre spécifié de parties. Chaque partie est simulée via une instance de `SimulateurPartie`, qui retourne le gagnant de la partie. Les résultats sont ensuite utilisés pour mettre à jour le score des joueurs en fonction des victoires, défaites ou matchs nuls.
- **Exécution du tournoi :** `organiser_tournoi` est la méthode qui lance le tournoi complet. Elle appelle `jouer_et_enregistrer_matchs` pour chaque paire de joueurs, garantissant que tous les joueurs se rencontrent, dans un format de tournoi round-robin.
- **Affichage des résultats :** Après la conclusion du tournoi, les résultats sont affichés en détaillant les scores finaux et le classement des participants, offrant une visibilité sur la performance relative des différentes IAs.
- **Réinitialisation des résultats :** Cette fonctionnalité permet de remettre à zéro les scores et les résultats pour préparer un nouveau tournoi.

Classe SimulateurPartie

La classe `SimulateurPartie` joue un rôle important dans la simulation des parties individuelles entre deux IAs, utilisant les stratégies définies par leurs configurations respectives sur un plateau de jeu Gomoku.

- **Initialisation :** Le constructeur de cette classe initialise une partie avec deux joueurs IA et un plateau. L'instance créée maintient les références aux deux joueurs et au plateau pour permettre une interaction dynamique au cours de la partie.
- **Affichage du plateau :** Avant chaque coup, l'état actuel du plateau est affiché grâce à la méthode `afficher_plateau`. Cette fonction imprime chaque ligne du plateau, permettant une visualisation claire de l'évolution du jeu, ce qui est essentiel pour suivre les développements durant le tournoi.
- **Déroulement de la partie :** La méthode `jouer_partie` orchestre le déroulement de la partie en alternant les tours entre les joueurs. Elle vérifie la validité et les résultats de chaque coup, annonce les coups joués, et détermine le vainqueur une fois que les conditions de victoire sont remplies sur le plateau.
- **Gestion des tours :** La méthode `changer_joueur` alterne le rôle du joueur actif entre les deux IA, assurant que chaque joueur ait une chance équitable de jouer. Cette gestion est essentielle pour maintenir l'équilibre et l'équité de la simulation.
- **Validation et résultats des coups :** Chaque coup est évalué pour sa validité. Un coup invalide entraîne une répétition du tour par le même joueur, tandis qu'un coup valide peut potentiellement mener à la fin de la partie si un alignement gagnant est réalisé.

Cette classe simule les interactions et la logique de jeu entre différentes configurations d'IA, offrant un aperçu précieux de leur compétence stratégique dans un environnement contrôlé. Les résultats des simulations contribuent directement aux données collectées durant le tournoi, influençant les stratégies futures et les améliorations des algorithmes d'IA.

5.3 Résultat du tournoi

```
Classement final:
1. IA Difficile avec 180 points
2. IA Facile avec 0 points
```