Rapport : MCTS VS UCB

1. Objectif du code

Ce programme simule 100 parties de Puissance 4 entre deux intelligences artificielles (IA) : l'une utilisant l'algorithme MCTS et l'autre utilisant une variante améliorée de l'algorithme UCB1, appelée UCB1-Tuned. L'objectif est d'évaluer les performances relatives de ces deux méthodes dans un contexte de jeu compétitif en analysant les résultats (victoires, défaites, matchs nuls).

2. Structure du code

a. Classe Connect4Game

- **Rôle** : Gère les règles et l'état du jeu Puissance 4.
- **Attributs principaux :**
  - **board** : Grille 6x7 (6 lignes, 7 colonnes) initialisée à zéro.
  - **current_player** : 1 (Rouge) ou 2 (Jaune), alternant à chaque tour.
  - **game_over** : Indique si la partie est terminée.
  - **winner** : Identifie le joueur gagnant (1, 2 ou None pour un match nul).
- **Méthodes clés :**
  - **get_valid_moves()** : Retourne les colonnes où un pion peut être joué.
  - **make_move(column**) : Place un pion dans une colonne et met à jour l'état du jeu.
  - **check_win(player)** : Vérifie si un joueur a aligné 4 pions (horizontalement, verticalement ou en diagonale).
  - **print_board()** : Affiche la grille avec 'R' pour Rouge, 'Y' pour Jaune, et des espaces pour les cases vides.

b. Classe Node

- **Rôle** : Représente un nœud dans l'arbre de recherche utilisé par MCTS et UCB1-Tuned.
- **Attributs :**
  - **game_state** : État du jeu à ce nœud.
  - **parent, move, children** : Structure de l'arbre.
  - **wins, visits, sum_of_squares** : Statistiques pour calculer les scores.
  - **untried_moves** : Liste des coups non explorés.
- **Méthodes :**
  - select_child() : Sélectionne un enfant selon la formule UCB1 standard (pour MCTS).

- **add_child() :** Ajoute un nouveau nœud enfant après un coup.
- **update(result)** : Met à jour les statistiques après une simulation.

c. Classe MCTS

- **Rôle** : Implémente l'algorithme Monte Carlo Tree Search.
- **Paramètres :** iterations (nombre de simulations, fixé à 1000).
- **Méthode principale** : get_best_move(game_state) :
  1. Crée un nœud racine.
  2. Effectue iterations simulations :
     - Sélectionne un nœud à explorer (via UCB1).
     - Étend l'arbre avec un coup non essayé.
     - Simule une partie aléatoire jusqu'à la fin.
     - Met à jour les statistiques des nœuds traversés.
  3. Retourne le coup le plus visité.

d. Classe UCB1TunedSearch

- **Rôle** : Implémente une version ajustée de UCB1 (UCB1-Tuned).
- **Différence avec MCTS** : Utilise une formule de sélection plus sophistiquée tenant compte de la variance des résultats (_select_child_ucb_tuned), ce qui ajuste l'exploration/exploitation dynamiquement.
- **Méthode principale** : get_best_move(game_state) suit un processus similaire à MCTS mais avec la sélection UCB1-Tuned.

e. Fonction run_simulation(num_games=100)

- **Rôle** : Orchestre la simulation des 100 parties.
- **Étapes :**
  1. Alterne les IA (MCTS et UCB1-Tuned) entre Rouge (premier joueur) et Jaune (second joueur) à chaque partie.
  2. Simule chaque partie jusqu'à la fin et affiche le plateau final.
  3. Calcule et affiche les statistiques :
     - Victoires de MCTS et UCB1-Tuned.
     - Matchs nuls.
     - Victoires du premier joueur (Rouge) vs second joueur (Jaune).

3. Fonctionnement des algorithmes

- MCTS : Utilise la formule UCB1 classique pour équilibrer exploration et exploitation, avec une constante d'exploration fixée à 1.41. Les simulations aléatoires permettent d'estimer la valeur des coups.
- UCB1-Tuned : Améliore UCB1 en intégrant une estimation de la variance des récompenses, rendant l'exploration plus adaptée aux résultats observés.

4. Résultats obtenus

Après 100 parties, les statistiques suivantes ont été enregistrées :

- MCTS : 45 victoires (45.0%)
- UCB1-Tuned : 54 victoires (54.0%)
- Matchs nuls : 1 (1.0%)
- Rouge (premier) : 68 victoires (68.0%)
- Jaune (second) : 31 victoires (31.0%)

Analyse :

- UCB1-Tuned surpasse légèrement MCTS avec 54% de victoires contre 45%, suggérant que sa gestion de la variance lui donne un léger avantage dans ce contexte.
- Le faible nombre de matchs nuls (1%) indique que les parties se terminent souvent par une victoire.
- L'avantage du premier joueur (Rouge, 68%) est significatif, ce qui est cohérent avec la théorie des jeux où commencer en Puissance 4 offre un avantage stratégique.

5. Limites et améliorations possibles

- Iterations : 1000 itérations par coup peuvent être insuffisantes pour un jeu comme Puissance 4 ; augmenter ce nombre pourrait améliorer les performances.
- Simulation aléatoire : Les parties simulées sont purement aléatoires, ce qui peut biaiser les résultats. Une heuristique simple (ex. : bloquer un alignement de 3) pourrait affiner les estimations.
- Équité : Le premier joueur (Rouge) a un avantage théorique dans Puissance 4, ce que les statistiques devraient refléter.

6. Conclusion

Ce programme offre une comparaison directe entre MCTS et UCB1-Tuned dans un cadre ludique bien défini. Les résultats obtenus montrent un léger avantage pour UCB1-Tuned et un fort avantage pour le premier joueur, soulignant l'importance de l'ordre de jeu et la robustesse des algorithmes testés.