Projet IN104 (2025) - Programmation d'une IA pour le jeu d'échecs en C

Encadrant : Tom Boumba

Fiche 0 : Introduction

Le jeu d'échecs

Le jeu d'échecs est un jeu à deux joueurs qui se joue au tour par tour sur un plateau carré de 8×8 cases appelé "échiquier". Chacun des joueurs dispose initialement de 16 pièces, l'un de pièces blanches, l'autre de pièces noires, disposées selon une position initiale fixe. À son tour de jeu, le joueur actuel déplace l'une de ses pièces selon ses possibilités de mouvement, qui dépendent de son type. On dit que le joueur "joue un coup".

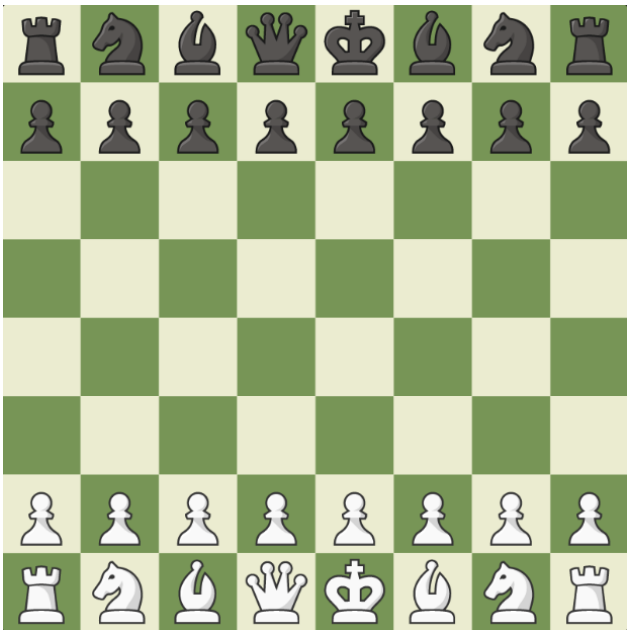

Figure 1: Position initiale de l'échiquier

Le déplacement d'une pièce s'effectue ou bien sur une case libre, ou bien sur une case occupée par une pièce du joueur adverse. Lorsque le déplacement est effectué sur une case occupée par une pièce adverse, la pièce adverse est retirée de

l'échiquier, et on parle alors d'une "capture (ou prise) de la pièce adverse".

Parmi ses 16 pièces initiales, chaque joueur dispose d'une pièce appelée "Roi". Le but du jeu d'échecs est de capturer le roi adverse. Au cours de la partie de jeu, on dit que le roi adverse est "en échec" s'il est possible qu'il soit capturé au coup suivant. Lorsque la capture du roi adverse est inévitable, on dit qu'il est "échec et mat", et la partie est remportée. Il existe aussi quelques cas dans lesquels la partie est remportée sans échec et mat, ou déclarée nulle. (voir ce lien pour une description exhaustive des règles du jeu d'échecs, et ce lien pour un tutoriel).

Au cours d'une partie de jeu, chaque joueur peut user de stratégie et de tactique pour arriver à mettre échec et mat le roi adverse. On peut capturer les pièces adverses ayant le plus de capacité de mouvement pour affaiblir la défense du roi adverse et minimiser la capacité de contre-attaque du joueur adverse, on peut prendre de l'espace sur l'échiquier pour maximiser la capacité de mouvement de ses pièces, et donc de potentialité d'attaque du roi adverse...

Programmation d'un moteur d'échecs

L'objectif de ce projet est de programmer une IA (Intelligence Artificielle) en C qui puisse jouer au jeu d'échecs (un moteur d'échecs). Idéalement, l'IA programmée aura un bon niveau de jeu, et pourra être affrontée et comparée à d'autres IA à l'aide d'une application sur interface graphique grâce au protocole UCI (Universal Chess Interface), par exemple Tarrasch Chess GUI. Actuellement, les meilleures IA pour le jeu d'échecs se divisent en deux catégories :

- Les IA qui fonctionnent selon des principes d'apprentissage par renforcement (par exemple AlphaZero)
- Les IA qui fonctionnent selon une approche Min-Max (par exemple Stockfish)

AlphaZero est un programme développé par la société de recherche et d'intelligence artificielle DeepMind, rachetée par Google. Malheureusement, il n'est pas disponible au public sous quelque forme que ce soit. De plus, la plupart des programmes d'analyse du jeu disponibles sur les sites d'échecs en ligne les plus populaires (par exemple Chess.com ou Lichess) utilisent un moteur fonctionnant selon l'approche Min-Max (Stockfish en général, qui est un programme open-source). C'est donc l'approche que nous utiliserons pour la programmation de notre IA, sur le modèle du moteur d'échecs didactique Bit Board Chess créé par Maksim Korzh, et écrit en langage C.

Consignes

- Le projet est à réaliser en binôme.
- Afin d'implémenter les diverses fonctionnalités de votre moteur d'échecs, vous vous aiderez des fiches 1 à 8.
- Le code de votre projet devra être réalisé à partir du dépôt public modèle qu'il vous faudra forker (n'oubliez pas de lire le fichier README).
- Votre fork devra être en privé, et vous devrez ajouter un accès en lecture pour le compte Github de l'encadrant : tom.boumba7@gmail.com
- Le code du projet devra être finalisé pour le 03/06/2025.
- Vous devrez téléverser sur votre dépôt un rapport structuré au format pdf détaillant le fonctionnement du code de votre projet.
- Vous présenterez également le code de votre projet à l'occasion d'une soutenance orale qui aura lieu le 03/06/2025.
- Toute fonctionnalité supplémentaire à celles décrites dans les fiches 1 à 8 sera valorisée (par exemple, l'implémentation de magic bitboards pour le calcul des masques d'attaque de pièces glissantes, l'utilisation d'une table de transpositions, diverses optimisations de l'algorithme de recherche alpha-béta, l'implémentation d'une heuristique d'évaluation plus avancée, ou même d'un réseau NNUE, etc.), à condition qu'elle soit dûment détaillée au sein de votre rapport et au cours de votre soutenance.