Mahias DUGUE - Valentin HULOT

IA : RAPPORT TP MCTS

I - Choix d'implémentation

A - Au niveau du jeu

Avec le squelette qui nous était fourni, nous devions implémenter le calcul des mouvement possibles des pions pour un tour de jeu, et utiliser des méthodes générales sur le **BoardChecker** pour vérifier certaines contraintes (pas d'out of bounds, présence ou non d'un pion sur une case, victoire d'un joueur, liste des unités, pion ou dames...).

Pour tous les calculs sur les listes de mouvement nous avons choisi les ArrayList comme structure de mémoire, car le tour de l'IA étant limitée en temps, nous préférerions optimiser le temps de calcul afin de lui permettre de jouer plus de parties. Contrairement à LinkedList, les ArrayList sont plus gourmands en mémoire mais beaucoup plus rapides au niveau de la recherche d'un élément ou d'une sélection aléatoire car la liste possède une référence directe vers tous les éléments contrairement aux liked liste ou chaque élément a une valeur et deux voisins.

Pour les listes de coups, nous avons choisi de faire deux fonctions auxiliaires pour les deux gros types de mouvement : les prises et les normaux. Pour ensuite construire les mouvements selon la règle "si la liste des captures est vide, on calcule les mouvements"

B - Au niveau de l'IA

Au niveau de l'algorithme MCTS, nous nous sommes rendus compte que tous les choix que nous pouvions faire étaient liés à la même problématique : trouver un compromis entre l'exploration de l'arbre et le développement des branches optimales.

Nous avons détecté deux points critiques dans l'algorithme à ce niveau :

- Le calcul de la méthode UCT qui prend en compte le score (winrate) d'un nœud en plus d'une composante sur l'exploration. Il est possible de multiplier cette composante par une constante permettant de plus ou moins privilégier l'exploration

$$uct = score + C * \sqrt{\frac{log\,(nb\ explos\ racines)}{nb\ explo\ du\ noeud\ courant}}$$

Nous avons choisi la constante sqrt(2) car selon nos recherches cette constante est considérée classique pour un jeu avec des possibilités moyennes et nous avons pensé que c'était un bon choix après plusieurs modifications

Une fois que l'expansion d'un nœud a été choisie, on peut choisir le nombre de simulation que l'on va jouer à partir du nœud. Après plusieurs tests et quelques soucis sur la non-exploration de certaines branches, nous avons choisi d'en faire qu'une afin de privilégier le nombre d'itérations et donc l'exploration de l'arbre et non celle de la branche en cours de simulation.

A un moment nous avions eu des problèmes car nous ne comparions pas les mouvements non existants avec ceux déjà dans l'arbre et donc l'arbre ne se construisait pas. Suite à une méthode de comparaison avec un nœud (et une modification de l'UCT afin de ne pas modifier par 0) nous avons fixé ce problème et réussi à avoir un vrai développement de l'arbre.

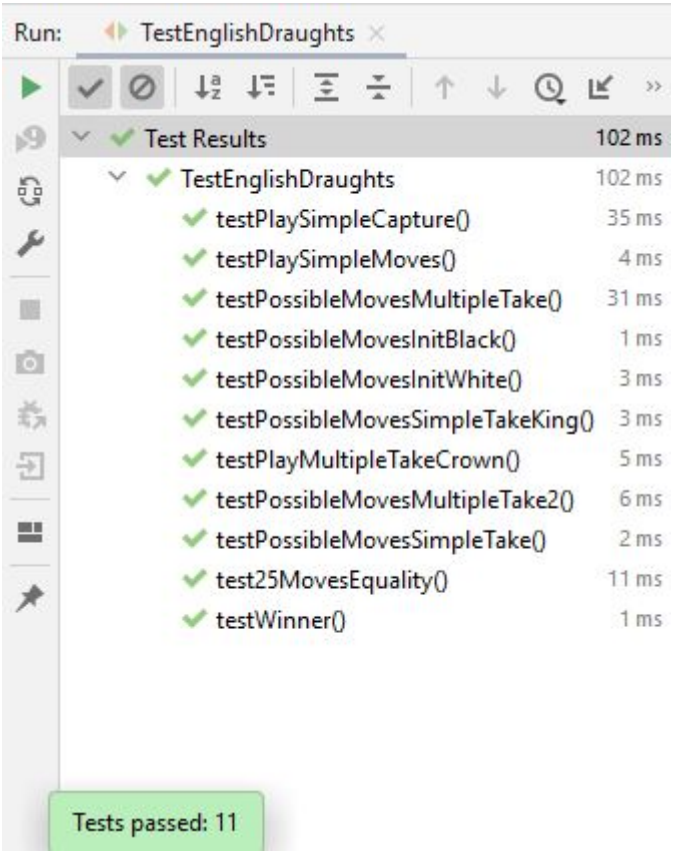
Un attribut **lastMove** a été ajouté dans **EnglishDraughts** afin de récupérer le coup. En effet, à la sélection du meilleur enfant, l'objet game ne permettait pas de récupérer le coup à jouer.

Dans la formule d'UCT , la partie score à été remplacée par 1-score, en effet, le score du prochain nœud est le score pour l'adversaire. De cette manière, on récupère donc le winrate pour le joueur courant.

II - Résultats de tests

Tests unitaires

Voici les résultats de test des fonctions de jeu. Ces tests nous ont permis d'identifier des erreurs d'oubli de cas ou d'implémentation.



Validation victoire :

Test humain vs IA 1s

Pour observer le comportement de l'IA, nous avons joué des parties contre elle. Nous nous somme aperçus que l'IA avait tendance à se suicider.En effet, dans notre fonction **getBestMove()**, nous avions maximisé le score de noeuds enfants pour obtenir le meilleur noeud.
Cependant, le noeud enfant contient la probabilité de victoire (score) de l'adversaire. Il faut donc minimiser le score pour obtenir le meilleur coup à jouer pour le joueur courant.

Une fois ce problème fixé nous avons rejoué contre l'IA, elle a gagné 3 des 5 parties que nous avons joué contre elle.

Test IA 1s vs IA 10s

Variante 8x8 :

-MCTS 1s (blancs) vs MCTS 10s (noirs) : Les pions noirs gagnent dans 6 des 8 parties testées
-MCTS 10s (blancs) vs MCTS 1s (noirs) : MCTS 10s gagne dans 3 des 4 parties jouées

Variante 10x10:

-MCTS 1s (blancs) vs MCTS 10s (noirs) : Les pions noirs gagnent dans 2 des 3 parties testées
-MCTS 10s (blancs) vs MCTS 1s (noirs) : MCTS 10s gagne dans 4 des 5 parties jouées