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

Encadrant : Tom Boumba

Fiche 7 : Negamax

Nous sommes désormais en mesure d'estimer le meilleur coup possible à jouer dans une position à l'aide de l'algorithme Minimax. Nous allons l'implémenter dans sa forme Negamax, qui fonctionne avec une évaluation relative par rapport au trait des feuilles de l'arbre à une profondeur donnée.

Cependant, le parcours de l'arbre des coups possibles est très coûteux en temps de calcul. Afin d'optimiser le parcours de cet arbre, nous utiliserons l'élagage alpha-béta. Cette technique consiste à arrêter l'exploration de l'arbre lorsque l'on sait que l'un des ancètres du noeud considéré au trait opposé a déjà trouvé un coup qui lui garantirait un score plus élevé que celui renvoyé par l'un des noeuds fils du noeud considéré. Il y a alors "coupure alpha" si l'ancètre en question est un noeud maximisant, et "coupure beta" si c'est un noeud minimisant. En pratique, les seuils de coupure, qu'on note α et β, sont transmis par les noeuds à leurs fils lors des appels récursifs de l'algorithme. Dans notre cas, puisque nous utilisons une implémentation "Negamax" de l'algorithme, tous les noeuds sont minimisants, et toutes les coupures sont des coupures beta. Seulement, puisque chaque noeud cherche le maximum de l'opposé des valeurs renvoyées par ses noeuds fils, chaque noeud doit inverser le signe de l'intervalle $[\alpha, \beta]$ lors de la transmission de celui-ci à ses noeuds fils.

Implémentation

Dans le fichier `negamax.c` :

- Implémenter la fonction `int negamax(int alpha, int beta, int profondeur)` sur ce modèle. Dans notre cas, la couleur est directement associée à l'état de l'échiquier (c'est le trait). Si vous le souhaitez, vous pouvez implémenter une forme de tri des coups considérés afin de maximiser l'occurence de coupures beta (par exemple en triant les noeuds auxquels mennent les coups en fonction de leur évaluation). Si le joueur qui a le trait n'a plus de coup légal disponible, soit il n'est pas en échec et la partie est déclarée nulle (on dit qu'il y a pat), soit il y a échec et mat, auquel cas la fonction doit renvoyer l'opposé de la valeur de mat définie dans le fichier `negamax.h`. L'appel au noeud racine de

cette fonction doit aussi mettre à jour l'estimation du meilleur coup au fur et à mesure.

- Implémenter la fonction `void recherche_meilleur_coup(int profondeur)` qui affiche l'estimation du meilleur coup possible à jouer dans la position actuelle pour des profondeurs croissantes de parcours de l'arbre donnée (on appelle cette technique l'iterative deepening). À l'appel initial, la fonction `negamax` doit être appelée avec l'intervalle $[\alpha, \beta] = [-$`VALEUR_MAX`$, +$`VALEUR_MAX`$]$.
  - Testez l'estimation du meilleur coup par votre moteur pour différentes profondeurs de recherche, par exemple :

```
generer_masques_d_attaque_statiques();
parse_fen(tricky_position);
print_echiquier();
recherche_meilleur_coup(4);
```

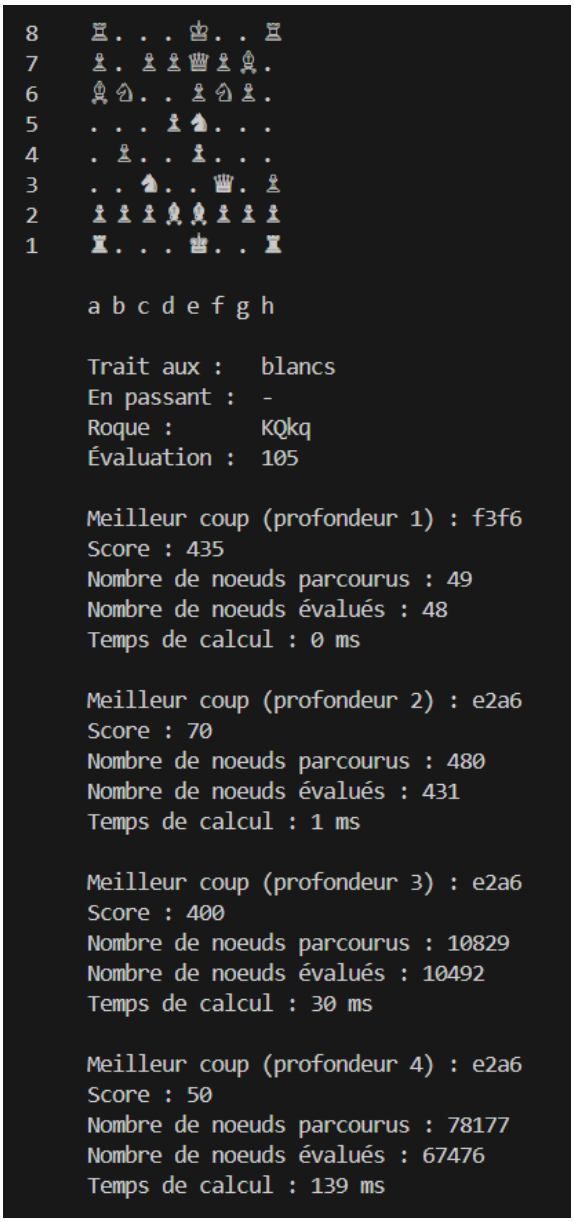

Figure 1: Estimation du meilleur coup à la profondeur 4 dans une position complexe.