Application de l'algorithme Min-Max avec élagage $\alpha\beta$ sur les jeux de Tafl, Ard-ri ou Tablut

Le but de ce projet est d'utiliser l'algorithme Min-Max avec élagage $\alpha\beta$ sur le jeu du ard-ri et si vous en avez le temps la recherche arborescente de Monte-Carlo.

1 Présentation du jeu

Les informations de cette section sont extraites de la page wikipedia sur les jeux de Tafl [1].

Le ard-ri et le tablut font partie d'une famille de jeu dit de Tafl (« table » en vieux norrois) dont les traces les plus anciennes remontent au IV*e* siècle. Les règles ont été reconstituées et plusieurs variantes existent. Nous nous fonderons sur les règles suivantes.

Le ard-ri se joue à deux sur une tablier de 7×7 cases et le tablut sur un tablier 9×9 (voir figure 1).

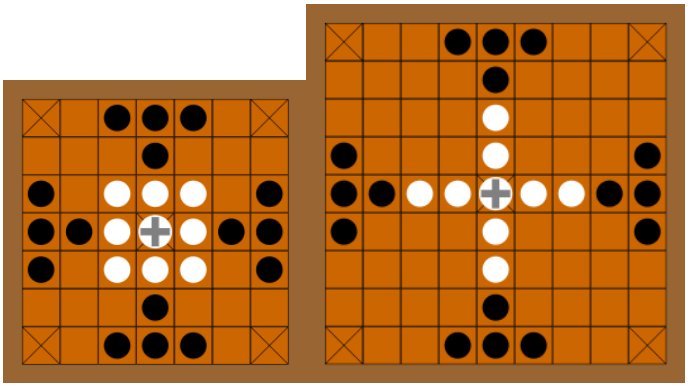

FIGURE 1 – Positions de départ (à gauche) et du ard-Ri et du tablut (à droite).

Le but est :

— pour le joueur noir, de prendre le roi adverse - le pion blanc marqué d'une croix sur les illustrations

— pour le joueur blanc, d'amener le roi dans une forteresse - les cases dans les coins du tablier.

Le roi est défendu par sa garde constituée de 8 pions blancs. Le camp des assaillants est constitué de 12 pions noirs.

1. `https://fr.wikipedia.org/wiki/Jeux_de_tafl`

Déplacements :

— Les assaillants jouent en premier.
— Les pions se déplacent horizontalement ou verticalement vers un emplacement libre, d'une ou plusieurs cases, comme la tour au jeu d'échecs.
— Toutes les cases empruntées lors du déplacement doivent être libres.
— Les assaillants ne peuvent accéder aux quatre forteresses situées aux coins, ni à la forteresse centrale où le roi commence la partie. Mais ils peuvent passer au-dessus de la case centrale si elle est libre.

Prise d'un pion :

— Un pion est capturé (et retiré du tablier) s'il se retrouve pris en tenaille par deux adversaires. Cette tenaille est effectuée - lors d'un déplacement - sur les côtés opposés d'un alignement (vertical ou horizontal) jouxtant la case qu'il occupe.
— Le roi peut participer à la capture d'un pion adverse mais on ne peut pas le battre de la même façon.
— Un pion est également battu s'il se trouve pris en tenaille entre un adversaire et une forteresse.
— Il est possible de prendre deux ou trois pions en déplaçant un seul pion.
— Un pion qui se place lui-même entre deux adversaires n'est pas capturé.

Capture du roi :

— La capture du roi constitue le but du jeu pour les assaillants.
— Pour capturer le roi, il faut l'entourer sur ses quatre côtés.
— Si le roi est sur un bord, trois pions suffisent à le capturer.
— Il est également possible de capturer un roi qui touche sa forteresse centrale avec trois pions seulement.
— Un roi qui touche une forteresse de coin peut même être capturé avec deux pions seulement.
— Pour capturer un roi qui se regroupe avec un ou plusieurs de ses partisans, l'assaillant doit entourer la totalité du groupe.

Victoire du défenseur :

— Pour gagner, le défenseur blanc doit amener son roi sur l'une des quatre forteresses situées aux coins.

Pour voir des illustrations des règles et connaître l'ensemble des variantes des jeux de tafl, vous pouvez consulter la page wikipedia `https://fr.wikipedia.org/wiki/Jeux_de_tafl`.

2 Travail demandé

Vous devez implémenter efficacement la recherche arborescente Min-Max avec élagage $\alpha\beta$ sur le jeu du Ard-ri et si vous en avez le temps la recherche arborescente de Monte-Carlo.

2.1 Modélisation du jeu

Portez un attention particulière au temps de traitement et à l'espace mémoire utilisés.

Vous devez définir :

— la représentation informatique d'un état du jeu.
— les actions possibles.
— la fonction d'évaluation à appliquer à chaque état pour indiquer si un état vous est plus ou moins favorable.

— implémenter la recherche arborescente Min-Max avec élagage $\alpha\beta$
— implémenter, si vous avez le temps, la recherche arborescente de Monte-Carlo.

Pour la conception, vous pouvez vous inspirer des éléments suivants (d'après le livre "Artificial Intelligence for Games", Ian Millington, John Funge, 2nd Edition, Morgan Kaufmann, 2009).

Recherche arborescente de Monte-Carlo :
Lorsque la fonction d'évaluation est difficile à définir et/ou que le temps de recherche de l'algorithme alpha-beta est important, l'évaluation des états atteignables en un coup, laquelle permet de choisir le prochain coup à jouer, peut être faite à l'aide de la recherche de Monte-Carlo. Cette approche génère un enchaînement aléatoire de coups jusqu'à obtenir une fin de partie et retient le résultat (ex. : $\text{Gain} = 1, \text{Nul} = 0, \text{Perte} = -1$). Un nombre variable d'enchaînements de coups en fonction du temps disponible est réalisé et l'évaluation de l'état correspond à la moyenne des résultats. En appliquant cette évaluation à chaque état atteignable en un coup, il devient possible de sélectionner le prochain coup, lequel correspond à l'état ayant l'évaluation la plus forte.

2.2 Réalisation

Le choix du langage de programmation est libre.

Paramètre modifiable dans l'interface :
— la profondeur limite de recherche.
— Le nombre d'enchaînements de coups à réaliser pour évaluer l'intérêt d'un état (pour la recherche arborescente de Monte-Carlo).

Informations à afficher :
— la valeur de l'évaluation des coups immédiatement possibles ;

— la profondeur moyenne trouvée pour une fin de partie (pour la recherche arborescente de Monte-Carlo) ;
— toute autre information jugée utile.

Jeu en réseau : Le code d'un client-serveur vous sera proposé pour jouer en opposition. Il faudra transmettre 4 entiers correspondant respectivement aux coordonnées x, y de la position de départ de la pièce à bouger et aux coordonnées x', y' de la position d'arrivée. Vous recevrez donc également 4 entiers correspondant au déplacement joué par votre adversaire. Une information sera aussi transmisse pour indiquer le joueur qui débute et la taille du tablier pour indiquer si vous jouer au ard-ri ou au tablut.

2.3 Évaluation

La présentation individuelle des réalisations aura lieu lors de la dernière séance de TP.