IA TP2 - Partie 1
Dames anglaises
avec MinMax et élagage $\alpha\beta$

Vincent Drevelle

Option IA - M1 IL/CR ISTIC - Année 2024-2025

Durée : 1 séance - En binôme
Le rendu sera commun avec celui de la deuxième partie du TP (MCTS).

Dans ce TP, nous vous fournissons l'implémentation en Java de deux jeux se opposant deux joueurs à tour de rôle : un premier jeu stupide, le morpion (*TicTacToe*) et un second plus intéressant, les dames anglaises (*English Draughts*). Le but est d'implémenter l'algorithme MinMax et sa variante alpha/beta afin de pouvoir réaliser des parties contre l'ordinateur.

Des fichiers source Java complets nécessaires à la réalisation du TP ainsi que des squelettes de classe sont fournis dans l'archive *tp_jeux_src.zip*.

La première section de ce document vous présente, à titre d'information, les règles du jeu de dames anglaises. L'implémentation du jeu est déjà réalisée dans les fichiers java. Une brève explication de l'organisation du code fourni vous est fournie en section 2. Le travail à réaliser est présenté dans la dernière section.

1 Rappel des règles des dames anglaises

Les dames anglaises sont une variante du jeu de dames se jouant sur les cases vertes d'un damier de 64 cases (8x8). Le jeu oppose un joueur avec des pions rouges (aussi appelés noirs) et un joueur avec des pions blanc, possédant chacun au départ 12 pions. Les pions noirs sont placés sur les trois rangées en haut du damier, les blancs sur les trois rangées en bas du damier.

1.1 Règles du jeu

Les cases jouables sont numérotées de 1 à 32. Au début du jeu, les pions noirs sont placés sur les cases 1 à 12, et les pions blancs sur les cases 21 à 32.

Le joueur avec les pions blancs commence la partie.

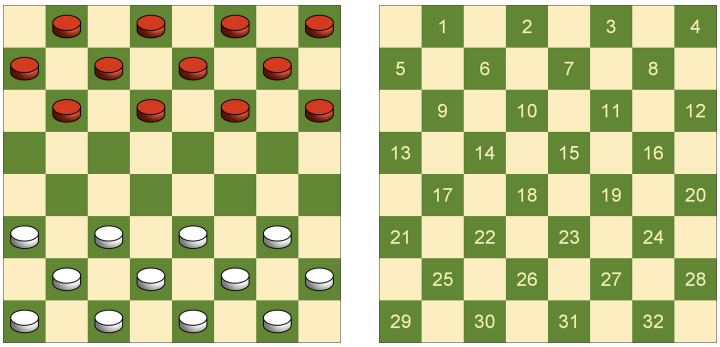

FIGURE 1 – Position de départ et numérotation des cases

1.1.1 Déplacements

Les pions se déplacent en diagonale d'une case vers l'avant (c'est à dire vers le camp opposé).

Un pion peut prendre une pièce adverse, en « sautant » par dessus en diagonale, à condition qu'il y ait une case vide derrière. La capture par un pion ne se fait qu'en diagonale vers l'avant.

Lorsqu'après une première capture, un pion se retrouve en situation de pouvoir capturer un autre pion, il doit continuer de capturer jusqu'à ce qu'il arrive à une position ne permettant plus de capturer. Il n'est pas possible de capturer deux fois le même pion au cours d'une rafle.

La capture est obligatoire (il n'est pas possible de jouer un mouvement de déplacement si on peut jouer un mouvement de capture). Par contre, il n'est pas obligatoire de jouer le mouvement entraînant la capture du maximum de pions.

1.1.2 Promotion en dame

Si un pion arrive, à la fin de son mouvement, à l'extrémité du plateau située dans le camp adverse, il est promu en *dame*. Il n'est pas possible de faire « dame en passant ».

La dame possède la capacité de se déplacer et de capturer aussi bien vers l'avant que vers l'arrière. Son mouvement reste limité à une case, ou à la prise d'une pièce immédiatement adjacente. Au cours d'une capture multiple, la dame peut alterner des déplacements vers l'avant et vers l'arrière.

1.1.3 Fin du jeu

La défaite d'un joueur est déclarée quand :

— soit tous ses pions ont été capturés par l'adversaire,

— soit, arrivé à son tour de jeu, il est bloqué et ne peut plus réaliser aucun déplacement ni capture.

Partie nulle Nous considérerons la règle suivante pour décider d'une partie nulle (c'est-à-dire égalité entre les deux joueurs) :

Une partie sera déclarée nulle si, durant 25 coups consécutifs, aucune prise n'a eu lieu, ni aucun déplacement de pion (autrement dit, pendant 25 coups, seuls des mouvements de dames ont eu lieu, sans capture).

1.2 Notation

On utilise généralement la notation Manoury pour représenter les coups joués. Les cases noires sont numérotées de 1 à 32, de gauche à droite et de haut en bas, en partant de la première case du camp des noirs en haut à gauche.

— Un tiret « - » représente un déplacement simple entre deux cases.

— Une croix « x » représente une prise.

Exemples

11-16 : le pion situé sur la case 11 se déplace vers la case 16.
11x18 : le pion situé sur la case 11 capture le pion en 15 et arrive en 18.
14x23x30 : le pion situé en 14 rafle deux pions et arrive sur la ligne adverse en case 30 (avec promotion en dame).

Cette numérotation des cases et notation des coups est implémentée dans le code fourni.

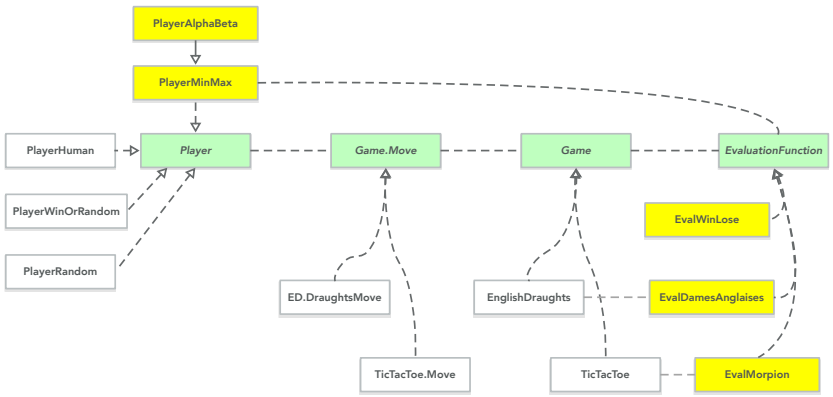

FIGURE 2 – Classes du TP (focus sur MinMax et $\alpha\beta$)

2 Explication de la structure du code existant

2.1 Organisation

Le code s'organise autour de quatre abstractions (rangées dans le package *tp_jeu.interfaces*) :

- la classe abstraite *Game*, qui représente l'état d'un jeu
- l'interface *Game.Move*, qui représente un coup
- l'interface *Player*, capable de choisir un coup à jouer à partir d'un état de jeu (ex : entrée humain au clavier, aléatoire, algo MinMax...)
- la classe abstraite *EvaluationFunction*, qui représente une fonction d'évaluation heuristique d'un état de jeu

Les fonctions de *Game* sont *possibleMoves()* qui retourne la liste des coups possibles à partir de l'état courant, *play(Move)* qui permet de jouer un coup, modifier l'état du jeu et passer au tour du joueur suivant. La fonction *player()* retourne l'identifiant du joueur courant (*ONE* ou *TWO*), et la fonction *winner()* retourne l'identité du vainqueur si la partie est finie, *NONE* en cas d'égalité, et *null* si la partie court toujours.

La fonction à implémenter dans un *Player* est la fonction *play(Game)* qui retourne un *Game.Move*, coup choisi par le joueur à partir de l'état de jeu fourni en entrée. Les classes implémentant les players sont rangées dans le package *tp_jeu.players*.

Une *EvaluationFunction* fournit *eval(Game)* qui retourne une valeur entre $-\infty$ et $+\infty$, en évaluant le jeu du point de vue du joueur sélectionné par *setPlayer(Game.PlayerId)* (par défaut, le point de vue du joueur courant, représenté par *PlayerId.NONE*). On implémente une fonction d'évaluation en surchargeant la fonction *evalForPlayer(Game, Game.PlayerId)* qui réalise l'évaluation pour un joueur passé en paramètre.

2.2 `TpJeu.java` : Boucle de jeu

La classe *TpJeu* fournit une interface console permettant la sélection du mode de jeu ainsi que du type de joueurs.

La fonction *main* implémente la boucle de jeu : demande du mouvement au joueur courant et mise à jour du plateau, jusqu'à la fin de la partie, et affichage du vainqueur.

3 MinMax et élagage $\alpha\beta$

Ce TP consiste à implémenter l'algorithme MinMax et sa variante avec élagage $\alpha\beta$ afin de déterminer le meilleur coup à jouer.

3.1 Algorithme MinMax : `PlayerMinMax.java`

Une classe *PlayerMinMax* compatible avec l'interface *Player* est fournie (dans le package *tp_jeu.players*). Complétez les fonctions *play()* et *minmax()* pour implémenter l'algorithme.

Par défaut, la fonction d'évaluation utilisée est *EvalWinLose* dont le code est déjà fourni dans le package *tp_jeu.games*. Cette fonction d'évaluation retourne $-\infty$ en cas d'état de jeu perdant, $+\infty$ pour un état de jeu gagné, et 0 dans tous les autres cas. Elle n'implémente donc aucune heuristique spécifique a un jeu. Une constante « `oo` » (deux lettres « o ») est définie afin de pouvoir utiliser « `+oo` » et « `-oo` » dans le code.

3.2 Heuristique simple de jeu

Dans le package *tp_jeu.games.draughts*, créez une classe *EvalDamesSimple* qui hérite d'EvaluationFunction (prenez modèle sur EvalWinLose). Vous implémenterez une heuristique simple d'évaluation d'un jeu de dames anglaises, basé sur le décompte des pièces du joueur et de son adversaire. Mettre une pondération plus élevée pour les dames.

Afin de rendre votre fonction d'évaluation disponible dans le menu de jeu, éditez le fichier *factories/EvaluationFunctionFactory.java* en vous basant sur les exemples en commentaires.

Comparez votre algorithme avec les deux fonctions d'évaluation.

3.3 Élagage $\alpha\beta$: `PlayerAlphaBeta.java`

Afin d'éviter d'explorer des branches inintéressantes de l'arbre de jeu, implémentez l'élagage $\alpha\beta$ dans la classe *PlayerAlphaBeta*.

Comparez le nombre d'appels récursifs obtenus avec MinMax et AlphaBeta pour une même profondeur d'exploration choisie.

À quel point AlphaBeta vous permet-il d'atteindre une plus grande profondeur en un temps raisonnable ?

L'efficacité des coupes alpha et beta est plus grande si on évalue d'abord les meilleures branches. Implémentez cette stratégie et comparez.

3.4 Pour aller plus loin (bonus)

3.4.1 Approfondissement itératif (iterative deepening)

Implémentez dans une nouvelle classe la stratégie d'approfondissement itératif pour la recherche. On incrémente successivement la profondeur jusqu'à la profondeur maximum désirée. On garde en mémoire les évaluations des coups réalisés lors de l'itération précédente, afin de guider la recherche à l'itération suivante.

Comparez les résultats avec l'alpha beta classique.

3.4.2 Recherche d'heuristiques

Cherchez une ou plusieurs heuristiques pour les dames anglaises et testez les, en les implémentant dans leur classes.

3.4.3 Un nouveau jeu ?

Pour le plaisir, et pour le bonus, vous pouvez implémenter un nouveau jeu de votre choix si vous le souhaitez.