Intelligence Artificielle – Semestre 1

Le grand tournoi des Masters sur un plateau

Vous devez rendre un joueur de Reversi (version spéciale de Othello) qui puisse s'intégrer au tournoi entre les joueurs. Le tournoi sera organisé (chacun rencontre les autres dans deux matchs (en tant que Noirs puis en tant que Blancs)).

Notations : un certain nombre de points (à titre indicatif, 4 points) seront directement donnés par rapport à votre classement dans le tournoi. 3 autres points seront donnés par rapport à votre capacité à battre des IA de différent niveaux (joueur aléatoire, minimax niveau 1, minimax niveau 3, ...). Le reste des points sera donné suivant le code que vous aurez rendu (expliquez vos méthodes dans le code!) et les techniques que vous aurez employées. Il faudra nous rendre, dans l'archive un fichier README.txt qui contiendra une description des points forts de votre joueur (faites court, listez les techniques, décrivez l'heuristique codée, précisez les structures de données, ...). Le projet sera à rendre via le Moodle qui sera associé à ce cours. Soyez donc vigilant aux instructions données via Moodle.

Règles du jeu

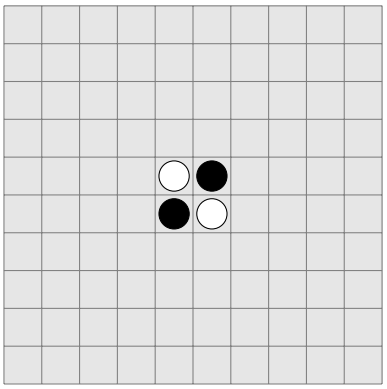

FIGURE 1 – Position des pièces au départ sur notre damier 10x10

Vous pouvez trouver les règles du jeu sur internet (wikipedia / Reversi). Par contre, **nous allons jouer sur un plateau 10x10** pour pimenter un peu le jeu. La position de départ est celle vue en TD et la même que pour Reversi (voir image ci-dessus). Noir commence. On ne peut poser une pièce qui si on prend au moins une pièce adverse lors de la pose. Si on ne peut poser de pièce, il faut passer son tour. Si les deux joueurs passent leur tour à la suite, le jeu est terminé. Celui qui a le plus de pièces de sa couleur remporte la partie. En cas de doute, vous devez vous en tenir aux règles de jeu implémentées dans la classe `Board` du script `Reversi.py`.

Le tournoi donnera 3 points en cas de victoire, 1 point en cas d'égalité et 0 point en cas de défaite. Celui qui aura le plus de point remportera le tournoi.

Chaque joueur n'aura le droit qu'à 5 minutes de temps réel de réflexion sur toute la partie. Il est interdit à un joueur de consommer du temps CPU en dehors des appels explicites aux fonctions de

l'interface (vous ne devez par exemple pas répondre à la méthode `getPlayerMove()` en laissant trainer un thread après votre réponse).

1 Interface de votre joueur

Pour pouvoir être interfacé avec les autres joueurs, votre IA doit implémenter l'interface donnée dans le script `playerInterface.py` dont le code est donné figure2. Les explications de chaque méthode sont données dans le fichier. Un exemple de joueur aléatoire implémentant l'interface est donné dans le fichier `myPlayer.py`.

Cette interface est importante : votre joueur sera mis en réseau grâce à un script python qui appellera vos fonctions. Pour vous donner une idée de comment le tournoi sera organisé, un autre script python (`localGame.py`) permet de connecter deux joueurs via leurs interfaces. Ce script ne sera pas utilisé pour le tournoi car chaque joueur n'est pas correctement isolé dans un processus indépendant. Durant le tournoi, les joueurs communiquerons en réseau avec l'arbitre. Tout sera géré automatiquement via l'interface donnée.

2 Modalités de rendu du tournoi

Vous devez rendre une archive contenant un répertoire ayant votre nom d'équipe (sans espaces). Si on lance `python localGame.py` dans votre répertoire, cela doit lancer un match de votre I.A. contre elle même (ce script ne doit normalement pas être modifié). Vous donnerez également, comme indiqué plus haut, un fichier README.txt décrivant les techniques utilisées dans votre I.A. Vous devrez modifier le fichier `myPlayer.py` pour qu'il implante votre propre jouer. Attention : vous n'avez pas le droit d'écrire dans votre répertoire pendant le match. Le tournoi pouvant lancer plusieurs instances de votre I.A. et votre répertoire sera en lecture seule (vous pourrez écrire dans /tmp mais il faudra bien penser à nommer vos fichiers de manière unique). Les sorties standards seront sauvegardées pour afficher les matchs (mettez y des informations importantes). Par exemple, si vous voulez faire un match de votre IA contre le random player, il vous faudra copier `myPlayer` en `randomPlayer` et copier `localGame` pour qu'il charge le joueur aléatoire avec le votre. **Important :** pour le rendu, `localGame` doit ne lancer que votre I.A. contre elle-même et `myPlayer` doit implanter **votre** joueur, c'est à dire le joueur qui devra entrer dans le tounoi. Il est déconseillé, pour la version tournoi, de laisser un constructeur ayant des paramètres. Si vous décidez de laisser des paramètres (comme la profondeur maximal de votre minimax) à donner au constucteur, il faut absolument clarifier quels paramètres doivent être donnés.

Vous êtes libre d'écrire votre propre classe `Board` à la manière du script `Reversi.py` pour accélérer votre I.A., mais il faut absolument laisser ce script intact pour que `localGame` se passe bien. Typiquement, si vous voulez le modifier, il faudra en faire une copie et renommer la classe et le fichier. Vous êtes libres d'organiser des tournois entre vous. Tous les matchs que nous lancerons seront accessibles sur internet. Attention, donc, à ce que vous écrivez.

```python
class PlayerInterface():
    # Returns your player name, as to be displayed during the game
    def getPlayerName(self):
        return "Not Defined"

    # Returns your move. The move must be a couple of two integers,
    # Which are the coordinates of where you want to put your piece
    # on the board. Coordinates are the coordinates given by the Reversy.py
    # methods (e.g. validMove(board, x, y) must be true of you play '(x,y)')
    # You can also answer (-1,-1) as "pass". Note: the referee will never
    # call your function if the game is over
    def getPlayerMove(self):
        return (-1,-1)

    # Inform you that the oponent has played this move. You must play it
    # with no search (just update your local variables to take it into account)
    def playOpponentMove(self, x,y):
        pass

    # Starts a new game, and give you your color.
    # As defined in Reversi.py : color=1 for BLACK, and color=2 for WHITE
    def newGame(self, color):
        pass

    # You can get a feedback on the winner
    # This function gives you the color of the winner
    def endGame(self, color):
        pass
```

FIGURE 2 – Interface à implémenter pour pouvoir jouer en tournoi