15 Projet : MCTS

À rendre au plus tard le lundi 10/01/2024 sur Moodle
Le projet peut-être réalisé en binôme ou seul.
On donnera les fichiers sources (`.py`) du programme ainsi qu'un rapport décrivant les expériences menées et les résultats obtenus.

Sur Moodle, on trouvera dans le répertoire "Projet2023" plusieurs classes permettant de mettre en œuvre l'algorithme MCTS pour un jeu à deux joueurs, à coups alternés, à savoir le Morpion. A ce jeu, les joueurs inscrivent tour à tour leur symbole sur une grille n x n. Le premier qui parvient à aligner cinq de ses symboles horizontalement, verticalement ou en diagonale gagne.

15.1 Description des classes

15.1.1 La classe PM

Cette classe permet de représenter l'état du jeu au cours de la partie. C'est dans cette classe que sont codées les règles du jeu.

Les attributs importants (qui servent pour les autres classes) sont :

— `j1aletrait` : booléen qui vaut `True` si le joueur 1 a le trait (c'est-à-dire qu'il doit jouer) et `False` si le joueur 2 a le trait.
— `coups` : une liste des coups possibles pour le joueur qui a le trait (ici c'est une liste de tuples, chaque tuple correspond à une case libre sur la grille)
— `nbcoups` : un entier égal au nombre de coups possibles pour le joueur qui a le trait. Il vaut 0 si la partie est terminée
— `eval` : un entier qui donne le résultat d'une position terminale : 1 si le joueur 1 a gagné, 2 si l'autre joueur a gagné, 0 si la partie est nulle (toute la grille est remplie et il n'y a aucun alignement)

Les méthodes importantes sont :

— `Clone(self)` : qui renvoie une copie de l'instance `self`. Aucune référence ne doit être partagée entre la copie et `self`.
— `EffectuerCoup(self,x)` : modifie l'instance de sorte qu'elle corresponde à la position obtenue après avoir joué le coup `x`. Cette méthode met à jour les attributs `j1aletrait`, `nbcoups`, `coups` et `eval`.

15.1.2 La classe Partie

Elle permet d'organiser une partie entre deux joueurs. La méthode `__init__` a trois arguments : `j1, j2` et `pinitiale`.

`pinitiale` doit être une instance de `PM` et représente l'état du jeu au départ.

`j1` et `j2` représente les deux joueurs : ces objets doivent avoir une méthode `Jouer(p)` qui prend en argument une position `p` (une instance de PM) et renvoie le coup choisi (un élement

de `p.coups`). Ils doivent avoir aussi une méthode `NouvellePartie()` dont la vocation est de réinitialiser le joueur pour une nouvelle partie (quand cela est nécessaire).

15.1.3 La classe Jmcts

Elle correspond à un joueur (via la méthode `Jouer(p)`) et met en œuvre l'algorithme MCTS (voir sur `https://en.wikipedia.org/wiki/Monte_Carlo_tree_search`) pour un temps de calcul fixé (l'attribut `temps`). L'attribut `a` est un paramètre strictement positif (un float) qui intervient pour le choix du coup le plus prometteur lors de la phase de sélection. La position fille choisie est celle pour lequel

$$\frac{W+a}{C+a}$$

est maximum où W est le nombre de victoires et C le nombre de traversées.

15.2 Travail à faire

On travaillera avec une grille de taille 9×9 et un temps limite de 1 seconde.

1. Écrire une classe `Jhasard` correspondant à un joueur jouant complètement au hasard. Vérifier que `Jmcts` est meilleur que `Jhasard`.
2. Écrire une classe `Jhumain` permettant à un utilisateur de jouer.
3. Déterminer quelle est la meilleure valeur du paramètre a entre $a = 1$ et $a = 5$. Pour cela, organiser 40 parties entre `Jmcts(a=1)` et `Jmcts(a=5)`. Chaque joueur jouera 20 parties en commençant en premier (puisque celui qui commence est avantagé).
4. On souhaite améliorer l'algorithme MCTS pour le jeu du Morpion. Pour cela on considère la variante suivante : dans un position donnée, on restreint les coups explorés à ceux qui concernent une case libre voisine d'un des pions déjà présents sur la grille.
   Concevoir une classe `JmctsM` qui met en œuvre cette variante. Il sera commode d'utiliser une classe auxiliaire similaire à `PM` mais où les coups possibles sont restreints. La classe `JmctsM` doit pouvoir être utilisée avec `PM` au sens que sa méthode `Jouer` doit accepter une instance de `PM` en argument.
   Est-ce que `JmctsM(a=1)` est meilleur que `Jmcts(a=1)` ? (on organisera 40 parties).
5. Une autre modification que l'on veut implémenter est la suivante : dans la phase de simulation (où les coups sont choisis au hasard jusqu'à la fin) on veut limiter le nombre total de coups effectués à un nombre m donné. Si la partie n'est pas terminée après m coups, on renvoie un résultat correspondant à une partie nulle.
   Modifier la classe `JmctsM` pour mettre en œuvre cette variante. On introduira un nouvel attribut `max_sim` correspondant à m.
   Comparer les performances
   de `JmctsM(a=1,max_sim=24)` et de `JmctsM(a=1,max_sim=100)`.

6. Créer une classe `Jchamp`, compatible avec les classes `PM` et `Partie`, qui corresponde à un joueur le meilleur possible pour une grille 9x9 et un temps limite de 1 seconde. Un championnat entre tous les projets rendus sera organisé.
   On prendra soin de spécifier des valeurs par défaut pour tous les arguments du constructeur de `Jchamp` (de sorte que `Jchamp()` renvoie une instance). De plus la classe `Jchamp` doit être écrite dans un script *NOM*`.py` où *NOM* est votre nom (ou vos noms) et la méthode `__str__` de `Jchamp` doit renvoyer *NOM*. Aucun autre fichier source ne doit être nécessaire pour utiliser votre classe `Jchamp`.