Recherche arborescente de Monte-Carlo sur l'awalé

Le but de ce projet est d'utiliser la recherche arborescente de Monte-Carlo sur le jeu de l'awalé.

1 Présentation du problème

L'awalé se joue à deux joueurs sur un plateau de 2 x 6 trous (voir figure 1). Au départ, chaque trou contient 4 graines.

FIGURE 1 – Position de départ du jeu Awalé

- Coup : A son tour, le joueur prend toutes les graines de l'un des 6 trous se trouvant de son côté et en dépose dans le sens antihoraire une dans chaque trou suivant celui qu'il a vidé.
- Capture : Si la dernière graine est déposée dans un des 6 trous se trouvant du côté de l'adversaire et comportant déjà 1 ou 2 graines, le joueur capture les 2 ou 3 graines résultantes. Les graines capturées sont alors sorties du jeu (grenier) et le trou est laissé vide.
- Rafle : Lorsqu'un joueur s'empare de 2 ou 3 graines, si la case précédente contient également 2 ou 3 graines, elles sont capturées aussi et ainsi de suite.

Le jeu se termine lorsqu'un joueur n'a plus de graines dans son camp car il ne peut plus jouer. L'adversaire capture alors les graines restantes (de son côté). Le joueur avec le plus de graines capturées a gagné.
Des règles supplémentaires existent [1].

2 Travail demandé

Vous devez implémenter efficacement la rechercher arborescente de Monté-Carlo sur le jeu de l'awalé.

1. `https://www.regledujeu.fr/awale/`

2.1 Modélisation du jeu

L'état du jeu peut être représenté par un tableau 2x4.

Une fonction d'évaluation est généralement définie pour appliquer un algorithme de type alpha-beta. Lorsque cette fonction 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 (Gain=1, Nul=0, 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.

Si on ne tient pas compte des cycles, c'est-à-dire que potentiellement nous pouvons retomber sur un état déjà rencontré, il est uniquement nécessite de sauvegarder l'état courant généré par la recherche et de tirer au hasard le coup suivant parmi les 6 potentiellement possibles (moins s'il n'y a pas de graine dans des trous du côté du joueur courant).

Vu le peu de temps dont vous disposez, vous pouvez ne pas mémoriser les états rencontrés. Une limite de temps au-delà de laquelle l'évaluation d'un enchaînement aléatoire sera stoppée peut cependant être nécessaire.

Pour une modélisation plus classique tenant compte des cylces, vous pouvez vous référer à la section "Pour aller plus loin" en fin de sujet.

2.2 Réalisation

Le choix du langage de programmation est libre.

Paramètre modifiable dans l'interface :
— Le nombre d'enchaînements de coups à réaliser pour évaluer l'intérêt d'un état.

Informations à afficher :
— la profondeur moyenne trouvée pour une fin de partie ;
— la valeur de l'évaluation des coups immédiatement possibles ;
— toute autre information jugée utile.

2.3 Présentation

Les soutenances auront lieu les 24 et 27 mai 2024 en fonction de vos disponibilités aux uns et aux autres. Une présentation d'environ cinq minutes suivie d'une discussion d'une dizaine de minutes vous permettront d'une part d'exposer vos résultats en précisant leurs points forts, leurs points faibles, et les améliorations possibles et d'autre part d'effectuer une démonstration de l'implantation de cette stratégie. Le support de la présentation et les sources doivent être déposés sur le site arche dédié

au module Intelligence Artificielle au plus tard le jeudi 23 mai 2024. Merci de nommer votre archive avec le nom du projet et les vôtres (ex. Awale-Nom1-Nom2-Nom3.zip).

Pour aller plus loin

L'évolution des états du jeu peut être modélisé à l'aide d'un graphe. Chaque nœud correspond à un état du jeu et chaque arc à une action possible.

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

```
class Board:
  def getMoves()
  def makeMove(move)
  def evaluate()
  def currentPlayer()
  def isGameOver()
```