Intelligence Artificielle
Projet : puissance 4 et MCTS

1 But

Le but du projet est d'écrire un programme pour jouer au puissance 4 (`https://fr.wikipedia.org/wiki/Puissance_4`) contre l'ordinateur. L'ordinateur choisira ses coups en appliquant l'algorithme de recherche d'arbre Monte Carlo UCT en temps limité (par exemple 3 secondes).

Pour vous aider, vous trouverez sur ARCHE le squelette `jeu.c` qui donne le canvas pour implémenter des algorithmes de jeux à deux joueurs ainsi qu'un exemple d'implémentation pour le morpion qui choisit ses coups au hasard. Vous pouvez le compiler avec `gcc jeu.c -o jeu -lm`. Vous pouvez partir de ce fichier, ou choisir de développer le programme dans un autre langage (facilement utilisable sur une machine Linux standard, à valider avec l'enseignant avant).

Remarques :

- L'algorithme MCTS travaille en général avec des récompenses positives entre 0 et 1, par exemple $\{0, 0.5, 1\}$ selon que l'adversaire gagne, qu'il y ait match nul, ou que l'ordinateur gagne. On pourra se limiter ici à des récompenses binaires dans $\{0, 1\}$ en considérant qu'un match nul est une partie perdue.
- La constante c peut être fixée à $\sqrt{2}$.

2 Rendu du projet

Le projet est à réaliser en binôme. Chaque binôme doit rendre son projet sur ARCHE avant le **1er mars 2019 à 23h** sous la forme d'une archive ZIP contenant

- un rapport détaillant le fonctionnement du programme et les réponses aux questions ci-dessous ;
- les codes sources.

Si vous choisissez de ne pas partir du fichier `jeu.c`, les structures de données et la procédure de compilation devront également être expliquées dans le rapport.

La note de projet dépendra de la qualité du programme, du rapport et du nombre de questions traitées.

3 Questions

Une fois le programme fonctionnel, répondez aux questions suivantes.

1. Affichez à chaque coup de l'ordinateur, le nombre de simulations réalisées pour calculer ce coup et une estimation de la probabilité de victoire pour l'ordinateur.
2. Testez différentes limites de temps pour l'ordinateur et comparez les résultats obtenus. A partir de quel temps de calcul l'ordinateur vous bat à tous les coups ?
3. Implémentez l'amélioration des simulations consistant à toujours choisir un coup gagnant lorsque cela est possible. Comparez la qualité de jeu de cette nouvelle version avec la précédente et expliquez à quoi cela est dû.

4. Si vous travaillez en C, quelle est l'utilité ici de compiler avec `gcc -O3` plutôt qu'avec les options par défaut ? Donnez des exemples illustratifs.

5. Comparez les critères "max" et "robuste" pour choisir le coup à jouer en fin d'algorithme. Conduisent-ils souvent à des coups différents ? Lequel paraît donner la meilleure performance ?

6. Donnez une estimation du temps de calcul nécessaire pour jouer le premier coup avec l'algorithme Minimax (sans alpha-beta ni limitation de profondeur).