

Rapport de projet

Intelligence Artificielle

Dylan Licho
Nicolas Kleinhentz

2019 - 2020
Université de Lorraine

Fonctionnement du programme

Compilation

La compilation, à partir du terminal, se fait avec la commande *make*, en effet le Makefile permet la création du fichier exécutable et la commande *make clean* permet de supprimer le fichier exécutable.

Execution

Le programme s'exécute avec la commande *./puissance temps* avec le temps étant le temps à accorder au MCTS-UCT. Une option d'affichage de debug existe en utilisant la commande *./puissance temps 1*. Cet affichage étant uniquement du debug il n'est en rien esthétique.

Fonctionnement

Le programme commence par demander le critère voulu ainsi que le choix du premier joueur. Ensuite le choix du coup par l'utilisateur se fait en choisissant la colonne dans laquelle il veut jouer. Pour le coup de l'ordinateur l'algorithme du MCTS-UCT permettra de jouer le coup le plus adapté avec une certaine limite de temps pour le calcul.

Questions

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.

Voir execution du programme.

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 ?

A partir de 14 secondes, l'ordinateur change de stratégie et à partir de 16 secondes, il devient imbattable.

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.

La qualité des coups joués par l'ordinateur est meilleure avec l'optimisation. En effet l'optimisation permet à l'IA d'allouer plus de temps de calcul pour l'exploration car la simulation est raccourcie au possible.

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.

L'option -O3 permet d'utiliser toutes les optimizations d'exécution et de compilation supportées par gcc. Cela nous permettra d'avoir une IA qui ira parcourir l'arbre des états de jeu bien plus profondément que sans l'optimisation, et on aura donc, supposément, une meilleure IA pour un même temps de calcul. Le nombres de simulations double par rapport à l'execution sans l'option.

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 ?

Les critères "max" et "robuste" devraient conduire à des coups quelques fois différent. En effet le critère « max » est axé sur le ratio victoires sur simulations tandis que le critère « robuste » est axé sur les simulations uniquement. L'exploration en utilisant le critère « max » est le plus efficace pour notre programme. L'exploration en utilisant le critère « robuste » n'est pas efficace dans notre programme mais ceci pourrait être dû à une mauvaise compréhension du MCTS-UCT ou d'une erreur de programmation.

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).

En estimant le facteur de branchement moyen à 6, et la profondeur moyenne à 21 (42 étant le nombres de cases du plateau, on estime le remplissage à la moitié en moyenne), on a donc une complexité de temps de calcul de l'ordre de $O(6^{21})$ donc $3 \cdot 10^{35}$. L'âge de l'univers étant $14 \cdot 10^9$. On considère un processeur 3GHz donc $3 \cdot 10^9$ opérations par secondes. En une année on a $365 \cdot 24 \cdot 60 \cdot 60$ secondes donc $31 \cdot 10^6$. On peut donc executer $3 \cdot 10^9 \cdot 31 \cdot 10^6 = 9,3 \cdot 10^{16}$ opérations par années. On a donc une complexité de $3 \cdot 10^{35}$ et $9,3 \cdot 10^{16}$ opérations par année, ce qui équivaut à $3,22 \cdot 10^{18}$ et donc pas envisageable en terme de calcul.