Rapport IA : Puissance 4 MCTS

Romain Duc

1er mars 2023

Compilation et exécution :
gcc -O3 -Wno-unused-result jeu.c -o jeu -lm && ./jeu optimisation choixMethode [temps]
avec optimisation = 'y' ou 'n', choixMethode = 'rob' ou 'max' et temps est une valeur decimale limite pour MCTS(facultatif).
Un CMake est fourni : voir le readme.

1 Réponse aux questions

Question 1 :

```
MCTS a choisi la colonne 3
30626 iterations UCT en 2.000 s
Estimated Win-Rate pour MCTS : 68.25 %

   | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
-------------------------------
 0 |   |   |   |   |   |   |   |
-------------------------------
 1 |   |   |   |   |   |   |   |
-------------------------------
 2 |   |   |   |   |   |   |   |
-------------------------------
 3 |   |   |   |   |   |   |   |
-------------------------------
 4 |   |   |   | O |   |   |   |
-------------------------------
 5 |   |   |   | X |   |   |   |
-------------------------------
 Quelle colonne ?
```

Question 2 : *En faisant commencer l'IA, on a plus de chance que cette dernière gagne. C'est le cas lorsque le temps de calcul est supérieur à 0.08s. Lorsque le joueur humain commence, l'IA a plus de chance de perdre et la limite est de 0.1s. Précisons que cela est effectué en mode robuste et sans optimisation.*

Question 3 : *A temps très faible, MCTS ne rate plus ces coups gagnants avec l'option ./jeu y [...]. Le fait de chercher les coups gagnants pendant les simulations prend plus de temps que de jouer un coup au hasard. Ainsi, moins de simulations vont être réalisées mais ces dernières seront plus précises et donc efficaces.*

Question 4 : *L'option -O3 de gcc est utilisée pour activer les optimisations de niveau 3 lors de la compilation. Ces optimisations améliorent les performances du code en réorganisant le code, en supprimant des instructions inutiles et en réduisant la taille de la mémoire utilisée. L'utilisation de -O3 fait augmenter la taille du code généré et une utilisation accrue de la mémoire, donc il est judicieux de tester les performances avec et sans cette option pour constater une amélioration de performance significative ou non.*

Avec l'opti et avec gcc -O3

```
MCTS a choisi la colonne 3
125043 iterations UCT en 2.000 s
Estimated Win-Rate pour MCTS : 79.53 %
```

Avec l'opti et sans gcc -O3

```
MCTS a choisi la colonne 3
33749 iterations UCT en 2.000 s
Estimated Win-Rate pour MCTS : 81.26 %
```

Sans l'opti et avec gcc -O3

```
MCTS a choisi la colonne 3
121226 iterations UCT en 2.000 s
Estimated Win-Rate pour MCTS : 79.47 %
```

Sans l'opti et sans gcc -O3

```
MCTS a choisi la colonne 3
33791 iterations UCT en 2.000 s
Estimated Win-Rate pour MCTS : 81.24 %
```

Question 5 : *Le critère "Robuste" utilise les simulations pour choisir le meilleur nœud parmi les enfants. Le critère "Maximum" privilégie quant-à lui le ratio récompenses / nombre de simulations pour choisir le meilleur nœud. L'adoption du critère "Maximum" de l'IA possède une win-rate plus élevée qu'avec le critère "Robuste". L'exploration de l'arbre de jeu s'avère plus efficace avec le critère "Maximum".*

Question 6 : *Si l'on implante uniquement l'algorithme Minimax, sans optimisation (sans élaguage α-β, ni limitation de profondeur), le temps avant de jouer le premier coup serait beaucoup trop élevé. En effet l'algo consiste à développer tout l'arbre pour pouvoir choisir le coup à jouer. On peut estimer le nombre de nœuds à 7^{41} (7 coups possibles pour les 42 cases du plateau, -1 car en fin de jeu le nombre de coups possibles diminue (on complète les colonnes)).*
Si un parcours de nœud se fait en une opération, et que l'on dispose d'un processeur 3 GHz, on aura 3×10^9 opérations par seconde, multiplié par 31×10^6 pour les 365,25 jours ($365.25 \times 24 \times 3600$).
On a par conséquent une complexité en $7^{42} = 3.11 \times 10^{35}$.
et 9.3×10^{16} opérations par année. Soit environ 3.34×10^{18} ans d'après un produit en croix, soit plus de 2.42×10^8 de fois l'âge de l'univers (13.8 Ga).