Charlotte LANUEL

Lucas SCHWAB

M1 Informatique

**Rapport de projet Intelligence Artificielle**

Implémentation de l’algorithme MCTS dans un jeu de Puissance 4

# Question 1

Pour répondre aux consignes du projet, nous avons utilisé le programme jeu.c fourni par le professeur.

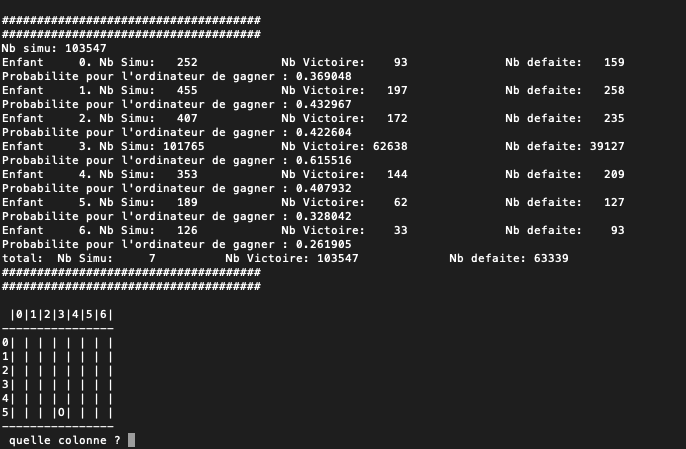
La partie la plus importante à expliquer est celle qui restait à développer : l’algorithme de Monte-Carlo.

Il a pour but de trouver le meilleur coup possible à jouer pour l’ordinateur dans un temps fini donné. Cet algorithme est constitué d’une grande boucle : tant que le temps n’est pas écoulé alors on continue de chercher. Dans cette boucle, on en place une autre qui va forcer MCTS à chercher tant que la branche choisie n’a pas été parcourue dans sa totalité.

Sachant que le testFin qui détermine l’issu de la partie est calculé à chaque nœud parcouru, on a implémenté une deuxième boucle : tant que l’issu de la partie est NON (donc la partie n’est pas terminée), on vérifie que le nombre de fils du nœud est 0. Si oui, on récupère un tableau de coups qu’il est possible de jouer, et on créé un fils pour chacun de ces coups.

On calcul ensuite pour chacun de ses fils la meilleure B-Valeur selon la formule donnée en cours. On teste à chaque calcul si la fin de partie est une victoire de l’ordinateur. Si oui, on remonte jusqu’au tout premier parent, et ce coup sera joué.

Un affichage de ce programme, à chaque fois que c’est au tour de l’ordinateur de jouer, sera comme ceci :



# Question 2

Pour cette question, on change le temps imparti pour Monte-Carlo afin de trouver le temps donné pour lequel l’ordinateur devient imbattable.

Pour TEMPS = 10 : à partir du sixième coup de l’ordinateur, il a plus d’une chance sur deux de gagner peu importe là où il joue. A partir du dixième coup, il a 100% de chance de gagner à chaque coup.

Pour TEMPS = 15 : à partir du troisième coup, l’ordinateur a plus d’une chance sur deux de gagner peu importe l’emplacement joué. A partir du quatrième coup, il a plus de 80% de chances de gagner peu importe son coup. Pour TEMPS = 10, l’ordinateur a déjà toutes les chances de gagner.

# Question 3

Si l’ordinateur se focalise sur les coups qui le font gagner, en revanche il ne prend plus du tout en compte ceux qui peuvent empêcher le joueur de le faire perdre. Dans l’ancienne version, l’ordinateur mettait un point d’honneur à bloquer les coups de l’adversaire avant que celui-ci puisse gagner, et plus encore : c’était le joueur humain qui devait faire attention à bloquer ses coups avant qu’il ne puisse aligner quatre coups. Maintenant, l’ordinateur ne prend plus du tout ce joueur en compte.

# Question 4

Contrairement à la commande gcc -o, la commande gcc -o3 permet une optimisation et une accélération de la compilation du projet, au détriment de la mémoire.

# Question 5

En réalité, les coups joués ne sont pas si différents lorsqu’on compare les deux algos avec un choix « max » et l’autre avec un choix « robuste ».

La seule différence notée, c’est qu’avec le choix robuste, l’ordinateur s’offre des chances de gagner mais prête un peu moins attention au jeu du joueur. Il ne pense donc pas à empêcher certains coups du joueur qui peuvent le rapprocher de la victoire.

# Question 6

On sait que l’algorithme tel qu’il est programmé peut explorer environ 120 000 nœuds en 5 secondes.

Le nombre de coups possibles au Puissance 4, en enlevant les coups gagnants, est 1.6 \* 10^13.

Pour trouver le temps approximatif de l’algorithme Minimax, on a le calcul suivant : 1.6\*10^13 / 120000, ce qui donne 1.3\*10^8. On multiplie ce résultat par 5 (le temps en secondes) et on a : 6.66\*10^8 secondes , soit 185 000 heures environ.