RAPPORT PROJET INFORMATIQUE C# : MCTS

Francois LE GAC, Shon AMSALHEM
M2 ISIFAR

Ce projet informatique en C# consiste à implémenter le jeu des allumettes ainsi que le jeu du puissance 4 en codant les différentes classes et méthodes de façon à faire jouer dans un premier temps 2 utilisateurs, puis ensuite l'ordinateur au moyen de l'algorithme Monte Carlo Tree Search, qui est un algorithme utilisé dans certains jeux, très efficace pour les prises de décision grâce à l'exploration de l'arbre des possibles.

Notre travail consiste dans un premier temps à comprendre les différentes méthodes et propriétés des classes abstraites Position et Joueur afin de les implémenter de façon optimale pour ainsi faire jouer 2 utilisateurs au jeu des allumettes et au puissance 4.
Dans un second temps, nous avons consacré du temps à l'étude de la classe JMCTS afin de comprendre au mieux les prises de décision de cet algorithme dans notre cadre.

JEU DES ALLUMETTES

Q1 & Q2)

Le jeu des allumettes est un jeu assez simple. il y a un certain nombre d'allumettes sur la table. Chaque joueur peut prendre 1, 2 ou 3 allumettes. Celui qui prend la dernière allumette a perdu.

Il a fallu dans un premier temps créer une classe PositionA, dérivée de la classe Position pour représenter une certaine position dans le jeu des allumettes. En effet, Il est important de connaître le nombre d'allumettes restantes à chaque tour des joueurs afin de suivre l'avancée du jeu.
On a défini dans le constructeur la propriété NbCoups représentant un entier entre 1 et 3 qu'il est possible de jouer. Cette propriété est égale à 3 tant que le nombre d'allumettes restantes est supérieur ou égal à 3, et s'égalise par la suite au nombre d'allumettes restantes (1 ou 2 allumettes). Aussi, l'initialisation de la propriété Eval correspondant à l'état du jeu : Indéterminé (partie en cours) , j0 Gagne, j1 Gagne.

Nous avons ensuite défini les méthodes EffectuerCoups prenant en argument le nombre i d'allumettes à prendre. On comprend donc facilement que le nombre d'allumettes restantes est systématiquement déduit du nombre i d'allumettes de chaque joueur.
Cette méthode modifie également la position courante de sorte qu'elle corresponde à la position obtenue après avoir joué le (i+1) ème coup. De plus, Elle met la propriété Eval à jour et calcul le NbCoup afin de suivre l'avancé du jeu.

Cependant, on remarque une subtilité lorsque l'on fait intervenir la classe JMCTS. En effet, la méthode JeuHasard fait appel à EffectuerCoup en générant un nombre aléatoire avec

“gen.Next(0, q.NbCoups))” qui renvoie 0,1 ou 2 selon l’état de Nbcoup. Il faut donc retirer (i+1) allumettes dans EffectuerCoup afin de répondre correctement au problème.

Dans un deuxième temps, nous avons implémenté la classe JoueurHumainA qui dérive de la classe Joueur demandant un coup à l’utilisateur. La méthode Jouer renvoie donc ce coup en faisant attention à bien respecter les règles du jeu, tout en passant au joueur suivant.

JEU DU PUISSANCE 4

Q3, 4, 5)

Le puissance 4 est un jeu facile à comprendre, chaque joueur met à tour de rôle un pion de sa couleur en haut d’une colonne, et celui-ci descend en bas de la colonne. Le premier joueur qui aligne 4 pions à l'horizontale, la verticale ou en diagonale a gagné.

Comme dans le jeu des allumettes, nous avons défini la classe PositionP4, dérivé de la classe Position afin de connaître la position du jeu. Nous avons implémenté la propriété NbCoup dans le constructeur de la classe de sorte qu’il soit égale au nombre de colonne disponible, c’est à dire le nombre de colonne ou il est possible de placer un pion, donc 7 au début du jeu. Également l’initialisation de la propriété Eval correspondant à l'état du jeu (indéterminé au début).
Au cours du jeu, Eval peut être Indéterminé (partie en cours) , j0 Gagne, j1 Gagne ou partieNulle.
Il est donc naturel de dire qu’un NbCoup égal à 0 correspondant au remplissage de toutes les colonnes n’est autre qu’un match nul.

L’implémentation de la méthode EffectuerCoup prenant en paramètre le numéro de la colonne à été relativement intéressant. En effet, il a fallu procéder systématiquement à une vérification des colonnes pour observer son éventuelle disponibilité avant de placer un pion.

Il a fallu également vérifier l’état de chaque colonne (remplie ou non) pour procéder à un décalage éventuel du paramètre i pour poser le pion, ce qui a permis de faire jouer convenablement le JMCTS, sans qu’il y ait d’erreurs ou de boucles infinies empêchant le bon déroulement du jeu.

Nous avons également implémenté l’évaluation du jeu qui est importante pour suivre son avancée. À chaque appel de la méthode EffectuerCoup, l’évaluation consiste à parcourir le tableau de jeu en démarrant un compteur auxiliaire à chaque pion placé. Ce compteur va ainsi vérifier si 4 pions identiques sont placés de manière consécutives à l'horizontale, à la verticale ou à la diagonale, pour en déduire s’il y a un gagnant.

Q6)

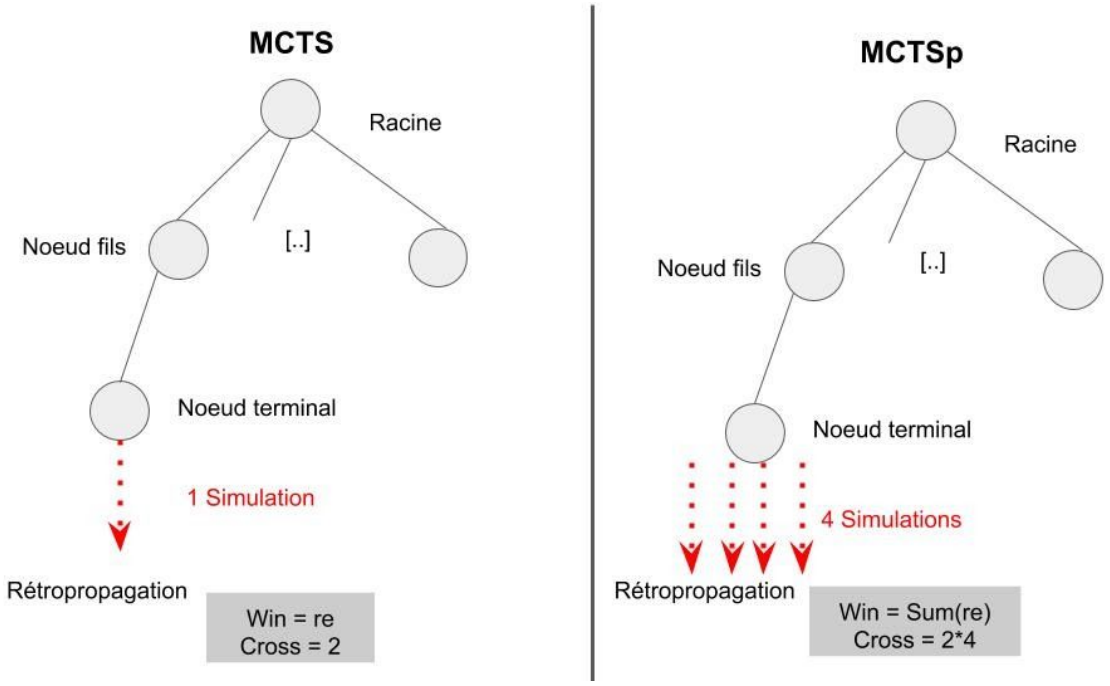
Non traitée

Q7)

Pour déterminer empiriquement le bon paramètre a on organise un championnat entre 100 joueurs différent. A chaque joueur correspond une valeur du paramètre a. On décide de faire

jouer chaque joueur contre les 99 autres restants. Le joueur qui cumule le plus de victoire est notre meilleur joueur. Le paramètre du meilleur joueur est le suivant : 10.

Q8)
On décide de mettre en place un nouvel un nouvel algorithme MCTS avec une parallélisation pour la phase de simulation.



Le nouvel algorithme permet de lancer 4 parties en même temps à partir du noeud terminal. La phase de rétropropagation contient cette fois-ci la somme des résultats des parties.

Q9)
On relance le championnat, cette fois-ci pour des joueurs JMCTSp. On trouve à nouveau le meilleur paramètre a empirique : 14. Pour un temps de 100 ms, c'est étonnamment l'algorithme MCTS qui est le plus performant entre les deux.

Q10)
Non traitée