EL MERSHATI Laith
HOMSY Aladin
CARTALAS Ines
FERREIRA Marianna

RAPPORT PUISSANCE 4

Langage Python

Explication des fonctions

Nous sommes d'abord parti du morpion que nous avions codé à un TD précédents avec des fonctions classiques, tels que **Result(s,a), Terminal-Test(s), Utility(s), Actions(s)** (retourne la liste des actions possibles), ainsi qu'un algorithme **MinMax** à élagage Alpha-Beta classique. Toutefois, nous nous sommes rendu compte qu'en augmentant les dimensions, le code commençait à mouliner. Il nous fallait alors optimiser nos fonctions et implémenter nos propres heuristiques.

Optimisations fonctions classiques:

Nous avons donc regrouper **Terminal-Test(s)** et **Utility(s)** pour obtenir :

def TerminalTest_Ut(s, lastplayed, jetons) -> teste la fin du jeu et attribue une valeur à l'état ***s***

Cette fonction nous permet de renvoyer en une seule fois un *True* ou *False* ainsi que le score.

De plus, nous retenons avec cette fonction la dernière position jouée *alias* la dernière action (par l'IA, l'adversaire, ou dans l'algorithme MinMax) qui nous permet de conclure sur l'état du jeu en regardant l'entourage de la position du jeton (par rapport à la puissance) : victoire ou non en fonction des jetons qui l'entoure en diagonal, de façon verticale ou horizontale.

Nous avons également supprimé la fonction **Result(s,a)** qui appliquait l'action ***a*** à l'état s et renvoyait ainsi une matrice copiée de la matrice originiale pour éviter les erreurs dans la matrice principale. Pour remplacer cette fonction, nous modifions directement notre matrice originale dans les fonctions **MaxValueAlphaBeta(s, alpha, beta, i, j, p, jetons)** et **MinValueAlphaBeta(s, alpha, beta, i, j, p, jetons)** et retirons le dernier coup testé avec une nouvelle fonction : **removeCoup(s, a).** On évite ainsi toutes les copies de matrice.

Avec ces changements nous étions plus apte à travailler sur des matrices de plus grandes dimensions et avec plus de rapidité, mais ce n'était pas suffisant pour une optimisation complète, nous avons donc implémenté nos **heuristiques**.

EL MERSHATI Laith
HOMSY Aladin
CARTALAS Ines
FERREIRA Marianna

Heuristiques :

Il nous fallait travailler sur la profondeur, qui est une variable qui décrémente et lorsqu'elle atteint 0, on lance la fonction heuristique.

La fonction **valeur_matrice(s)** va retourner la valeur totale en sommant toutes les fonctions heuristiques sur les colonnes, lignes et diagonales.

La fonction **colonne(s, j, dimI)** permet d'analyser une colonne, que l'on va réitérer par une boucle dans la fonction précédente.

La fonction **ligne1_1(s, dimI, dimj)** va analyser la toute première ligne de la matrice et la fonction **ligne1_2(s, dimI, dimJ)** les lignes restantes.

Les fonctions **diagonale1_1(s, dimI, dimJ), diagonale1_2(s, dimI, dimJ)** analysent les deux types de diagonales qui partent de la première ligne **; diagonale2_1(s, dimI, dimJ), diagonale2_2(s, dimI, dimJ)** évaluent les diagonales au dessus.

EL MERSHATI Laith
HOMSY Aladin
CARTALAS Ines
FERREIRA Marianna

On peut illustrer nos heuristiques de la manière suivante :

Le score s'il y a victoire au prochain tour : *+ 100 000* | Dans le test terminal : *+ 5 000 000* afin d'être assez large par rapport aux score à *+ 100 000* étant donné que la matrice est assez grande.

➢ Lignes et diagonales

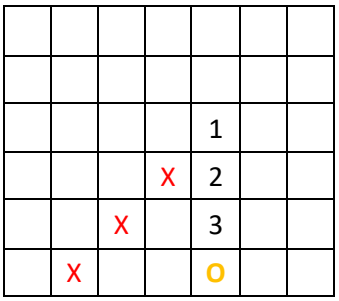
2 cases :

* Emplacement libre sur une suite de 4 doit être = 2 sinon situation = 0 point
Score + 1 000

3 cases :

* Au moins un emplacement libre sur une suite de 4 sinon situation = 0 point
Score + 10 000

Si position suivante accessible (qu'on peut jouer au prochain tour) et c'est à nous de jouer (en gros on a gagné) :
Score + 100 000



Atteindre emplacement libre :

Atteignable au prochain tour (1 case) : - *0*
2 cases : - *20*
3 cases : - *40* (ajouté au précédent)
4 cases : *-40* (ajouté au précédent)
…

Score = 10 000 - 20 - 40 - 40

➢ Colonnes

2 cases :

* Emplacement libre au-dessus sinon situation = 0 point
Score + 1 000

3 cases :

* Emplacement libre au-dessus sinon situation = 0 point
Score + 10 000

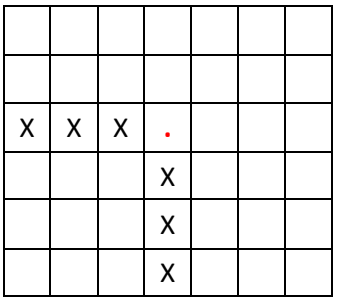
Si position suivante accessible (qu'on peut jouer au prochain tour) et c'est à nous de jouer (en gros on a gagné) : *Score + 100 000*

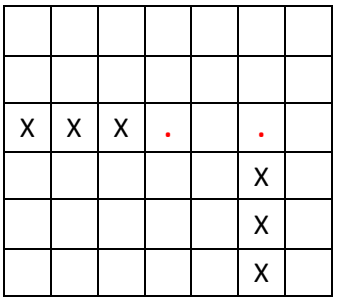
EL MERSHATI Laith
HOMSY Aladin
CARTALAS Ines
FERREIRA Marianna

- Autres

- Ajouter situations de peu importe où tu joues j'ai gagné.

- 2 moves de 3 cases qui se gagnent en jouant le jeton au même endroit ne doivent être compté qu'une fois (Le premier état nous offre moins de solution que le deuxième).

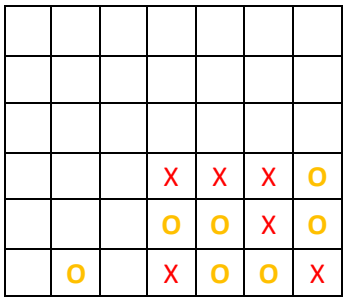


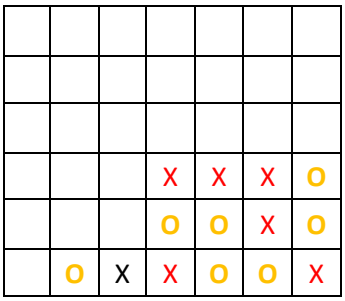


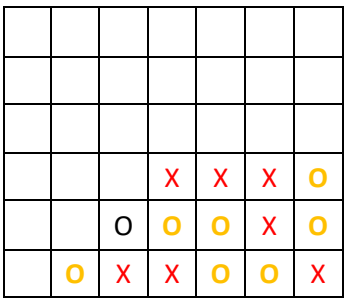
+ 10 000 seulement

+ 10 000 + 10 000

- Exploiter une fragilité dans le code de l'adversaire à notre avantage







En jouant exprès à une certaine position on peut influencer l'adversaire à jouer une position lui étant favorable : aligner 3 O, or si sa profondeur ne va pas plus loin ou qu'il cherche à gagner avant de se défendre alors on pourra exploiter cette faille pour jouer un coup gagnant.