Maxime BLANCHARD

Lingyi CHENG

Louis ROLLAND

Projet IA Teeko :

Professeur : Fabrice LAURI IA41 : Automne 2022

Table des matières

I – Sujet Teeko …………………………………………………………………………………………………

1 – Choix du sujet …………………………………………………………………………………

2 – Analyse générale du problème ………………………………………………………

3 – Application dans le langage Python (struct données + traitements)

II – Fonction éval / machines à états ………………………………………………………………

1 – Description hiérarchie …………………………………………………………………….

2 – Justification des choix …………………………………………………………………….

III – Problèmes rencontrés ……………………………………………………………………………

1 – Fonction eval ..………………………………………………………………………………….

2 – Choix meilleure situation (pbs choix aligner 3 rather than blocking)

3 – Application Algo MinMax (réalisation)

…

IV – Pistes d’améliorations ………………………………………………………………………………..

V – Conclusion ………………………………………………………………………………………………….

---------------------------------------------------------------------------------------------------------

Annexe 1 : Exemples en jeu

Annexe 2 : next\_states et next\_states\_beginning

I – Sujet Teeko :

Choix du Sujet -

Etant tous débutants en intelligence artificielle, nous avons d’abord décidé de choisir un sujet de difficulté novice, pour nous permettre d’avoir le temps de proposer un rendu à la hauteur de nos ambitions.

Le sujet Teeko nous a semblé être intéressant à traiter car il nous permettait d’implémenter l’algorithme MinMax avec élagage alpha-bêta, procédé abstrait que nous n’avions pas tout à fait saisi lors de sa présentation en cours. Néanmoins, cet algorithme avait piqué notre attention, et c’est d’une part pour cela que nous avons choisi ce sujet.

D’autre part, le Teeko nous a tout de suite paru être un jeu assez amusant, rappelant à la fois le « Puissance 4 » auquel nombre d’entre nous avions déjà joué, et créant une suite hypothétique à ce jeu où l’égalité n’arrive que trop souvent.

Cela paraît assez fou de se dire qu’il est possible de programmer une intelligence artificielle sur un jeu aussi stratégique que celui-ci, ce qui n’a fait qu’attiser notre intérêt pour ce sujet.

C’est pourquoi nous avons fait le choix d’interpréter ce sujet comme la création d’une intelligence artificielle capable de jouer une partie de Teeko contre un être humain, et supposément, de gagner la quasi-intégralité de ses matchs.

Analyse générale du problème –

Après avoir lu plusieurs fois le descriptif du sujet, nous avons déterminé la manière optimale de procéder pour venir à bout du projet.

Tout d’abord, nous allions créer une fonction qui prendrait un état de jeu en paramètres et qui permettrait de renvoyer l’ensemble des états fils de cet état. Le Teeko étant un jeu comportant deux phases (la première où chaque joueur pose tour à tour un jeton sur la grille, et la deuxième où chaque joueur déplace l’un après l’autre un de ses jetons), il faudra donc créer deux fonctions distinctes pour chacune de ces phases.

Ici, on nommera ces fonctions respectivement next\_states\_beginning et next\_states

Ensuite, il faudrait être capable de choisir le meilleur état fils pour une situation donnée. Nous avons donc décidé d’attribuer une « note » à chaque situation pour que le programme puisse choisir un unique coup à jouer, supposément le meilleur.

Finalement, il serait préférable que le programme trouve la meilleure situation rapidement et avec une certaine profondeur : on souhaiterait que l’IA décide du coup à jouer en examinant les conséquences de ses choix sur plusieurs tours à l’avance.

Application en python –

Nous avons d’abord considéré qu’un état serait caractérisé par le joueur auquel c’est le tour de jouer, ainsi que la grille de jeu sous la forme d’une matrice à deux dimensions de taille 5x5.

Ensuite nous avons créé la fonction next\_states\_beginning qui renverra l’ensemble des plateaux de jeu où le joueur auquel il appartient de jouer pose un nouveau jeton, puis la fonction next\_states qui renverra l’ensemble des grilles de jeu où le joueur dont c’est le tour déplace un de ses jetons là où il le peut (le résultat sous forme d’une liste d’états).

Voir les fonctions python de manière concrète dans l’Annexe 2.

Pour attribuer une note à un état, nous avons ensuite construit une fonction qui prend en paramètre un état, et qui renvoie un score qu’elle attribue à cet état en fonction de différents paramètres arbitrairement choisis :

* Le nombre de jetons alignés horizontalement
* Le nombre de jetons alignés verticalement
* Le nombre de jetons alignés en diagonale
* Le nombre de pièces en position ‘carré’
* La distance moyenne des jetons par rapport au centre du plateau

Cette fonction prend le score associé aux jetons posés par l’IA, et soustrait le score correspondant aux jetons du joueur à ce dernier.

Ainsi, dans le cas où le joueur est en position avantageuse, le score renvoyé sera négatif. Dans le cas contraire, le score associé à la position de jeu sera positif.

Une image contenant texte

Description générée automatiquement

*Score négatif car le joueur (-1 représenté par des -) a l’avantage*

Ensuite, nous avons fait appel à l’algorithme MinMax accompagné de l’élagage alpha-bêta vu brièvement en cours.

Le rôle de cet algorithme est de parcourir le plus efficacement possible l’arbre des états fils d’une situation de jeu, dans le but de trouver la situation la plus avantageuse pour l’IA en vérifiant plusieurs tours à l’avance les conséquences d’un coup.

II – Fonction éval :

Description hiérarchie -

Coefficients not chosen yet

Justification des choix -

Coefficients not chosen yet

III – Problèmes rencontrés :

Fonction eval -

Beaucoup de problèmes ont été rencontrés lors de la création et de la réalisation en python de cette fonction.

En effet, cette fonction est à la clé de la réussite de notre programme : il fallait donc trouver les paramètres optimaux pour cette dernière.

Très vite, nous avons pensé à 4 premiers paramètres :

* Si 4 jetons étaient alignés en :
  + Diagonal
  + Horizontal
  + Vertical
  + carré

que nous avons ensuite affinés en « nombre de jetons alignés » dans les mêmes dispositions pour chaque joueur, de manière à obtenir des scores plus diversifiés et adaptés à toutes les situations de jeu.

Pour programmer ceci en python, de nombreuses difficultés ont été rencontrées, mais le temps et la détermination nous ont permis d’en venir à bout.

Cependant, nous nous sommes vite rendu compte de l’insuffisance de ces simples paramètres. Nous avons donc commencé à réfléchir à quels paramètres pourrait avoir du sens sur un plateau de Teeko.

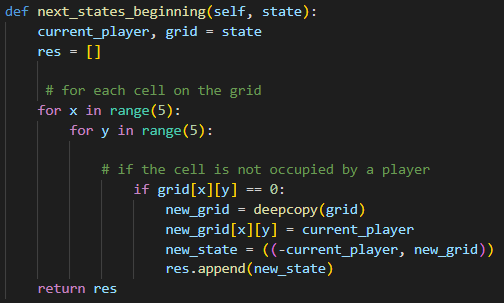
C’est alors que nous est venu l’idée de prendre en compte la distance moyenne des jetons par rapport au centre : en effet, plus les jetons sont proches du centre de plateau, plus la chance d’en aligner 4 devient grande.

<Voir pour rajouter un autre paramètre>

Fonction eval -

Annexe 2 :

next\_states\_beginning :



next\_states :

