Projet 3I025 : Ultimate Tic-Tac-Toe

De Rycke Théophile & Simon-Fine Thibaut

Contenu du rendu

- Compte rendu du projet au format PDF
- Code source du projet :
    - Archive zip jointe
    - Projet sur GitHub : https://github.com/ThibautSF/3I025_UltimateTicTacToe

Lancer le projet

Pour exécuter le projet il faut lancer UltimateTicTacToeV2.py (codé en python 3.5+). Le jeu s'arrête lorsqu'un joueur a gagné ou que le jeu est rempli.

Pour choisir les stratégies à appliquer pour chaque IA il suffit de modifier le dernier argument de la fonction "jouer" dans le fichier UltimateTicTacToeV2.py (lignes 203 à 208).
Attention, les stratégies RANDOM et RANDOM2 ont la possibilité de générer une erreur de stack overflow lors de la recherche aléatoire d'un nouveau terrain où jouer (risque dépendant du nombre de terrains remplis).

Les implémentations

Manhattan

Le manhattan revient à faire un simple calcul de distance entre deux cases du jeu. Il compte le nombre de case qu'il faudrait au personnage pour atteindre sa destination sans prendre en compte les obstacles sur le chemin. Nous l'utilisons dans l'algorithme du A-Star pour calculer l'heuristique de chaque position tester par ce dernier.

Fonction : manhattan(position,objectif)

A-Star

Pour implémenter l'algorithme A* nous nous sommes basés sur les codes vu en cours ainsi que d'un pseudocode de l'algorithme issu de la page Wikipédia anglaise de l'algorithme (https://en.wikipedia.org/wiki/A*_search_algorithm#Pseudocode).

Tout les noeuds (cases) ont un coût de traversée de 1.
La collision entre joueurs n'a pas été implémenté mais peut-être un ajout possible via l'ajouts d'une condition dans la fonction de récupération de la frontière.

Fonction : astar(depart,objectif,wallStates)

Fonctions utilisées :

- manhattan(position,objectif) pour le calcul de l'heuristique
- lowestNode(openSet,fScore) pour trouver le noeud minimal dans l'ensemble des noeuds ouverts
- getChoix(position,wallStates) pour récupérer la frontière du noeud en cours
- getChemin(cameFrom, pos) pour récupérer le chemin à parcourir jusqu'à l'objectif

Vérification du gain

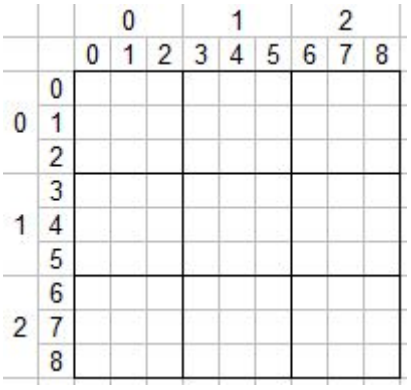
La fonction isVictoire(pos,matrice) permet de déterminer si un joueur à gagner dans un carré et si il a gagner le jeu. Elle renvoie -1 si personne ne gagne, 0 si le joueur 0 gagne et 1 si le joueur 1 gagne.

Pour vérifier la victoire on a deux matrice utilisé: playedTicTacToes qui indique les positions des fioles dans le jeux (initialement rempli de -1), et matriceVictoire qui indique les carrés gagné.

Pour vérifier si un carré est gagné par un joueur on applique isVictoire sur playedTicTacToes à chaque fois qu'une fiole est posée à un emplacement "pos" et on renvoie la valeur dans matriceVictoire à la position du carré. À noter que si le carré est déjà gagner, on n'utilise pas isVictoire. Ensuite si un carré est gagner par un joueur on test isVictoire sur la matriceVictoire avec "pos" = (0,0) pour tester si l'un des joueurs a gagné.

Fonction: isVictoire(pos,matrice) parcours, dans toutes les positions permettant la victoire, un carré de la matrice et cherche si ce carré est gagnant.

Les stratégies



Les stratégies se basent sur une matrice 9x9 (playedTicTacToeStates) initialisée à -1 (pour case libre) qui représente les matrices 3x3 des 9 terrains côte à côte.
La fonction isPlein(pos,matrice) est alors régulièrement utilisé pour savoir si la sous-matrice 3x3 aux coordonnées pos de la matrice principale est pleine.
NB : Donner pos=(0,0) pour une matrice 3x3 aura le même résultat

La fonction appelant les stratégies est :
jouer(numJ, playedTicTacToeStates, playIn, previousPlay, tour, playerStrat)
Celle-ci s'occupe d'appeler la fonction adéquat à la stratégie 'playerStrat' donnée.

Aléatoire Naïf ("RANDOM")

La stratégie aléatoire naïve joue toujours dans le carré indiqué par le joueur précédent et joue aléatoirement à l'intérieur de celui-ci. Si c'est le début de la partie ou que le carré indiqué est plein la stratégie joue à un endroit aléatoire dans la grille du jeu.

Fonction : stratAleatoire(numJ,playedTicTacToeStates, playIn, isNaif=True)

Fonction utilisée :

- getCarreDeJeu(playIn,isNaif,playedTicTacToeStates) pour récupérer un carré de jeu (gestion carré de jeu plein)

Aléatoire Non Naïf ("RANDOM2")

La différence entre cette stratégie et la stratégie aléatoire naïve est que celle ci prend en compte cette partie de la règle alternative : "Si le carré où le joueur doit jouer est déjà gagner, celui ci peut jouer dans n'importe quel autre carré". Dans ce cas là le personnage joue aléatoirement dans n'importe quel carré.

Utilise les mêmes fonctions que la stratégie aléatoire naïve (RANDOM) mais avec le paramètre 'isNaif' à False.

Stratégie Gagnante ("GAGNANTE")

Cette stratégie consiste à obliger l'adversaire à jouer dans le même carré. Le joueur avec cette stratégie joue pendant 8 tour sur la même case, puis joue sa 9eme fiole à la position correspondant à la position du carré dans le grand Tic-Tac-Toe dans le petit où il doit jouer. Il recommence ainsi durant les prochain tour. Si il ne peut pas jouer à la position où il veut car il y a déjà une fiole, il jouera à la position opposé dans ce carré. Si aucune de ces positions n'est possible il finit par suivre la stratégie TERRAIN.
C'est une stratégie gagnante lorsqu'on suit les règles de base de l'Ultimate Tic-Tac-Toe, donc si opposition avec une stratégie naïve, et si joué en première.

Fonction : stratGagnante(numJ,nprec,mprec,tour,playedTicTacToeStates, playIn = (-1,-1))

Fonctions utilisées

- isPlein(playIn, playedTicTacToeStates)
- stratGainTerrain(numJ,playedTicTacToeStates, playIn)

Stratégie Gain de Terrains Naïf (“TERRAIN”)

Cette stratégie consiste à essayer de gagner les terrains où elle joue (application d’une stratégie de gain d’un Tic-Tac-Toe au sein d’un terrain). Si le terrain où jouer est plein, la stratégie consiste à aller jouer dans le terrain qui permettra d’aligner des terrains (la matrice des terrain est vu comme un Tic-Tac-Toe).
En priorité la stratégie essaie de gagner (si un alignement durant le tour est possible), sinon elle essaie de bloquer l’adversaire (si un alignement est possible pour son tour), sinon joue pour avoir un alignement).
Dans le cas où aucune solution n’est trouvée, la stratégie joue aléatoirement.

Fonction : stratGainTerrain(numJ,playedTicTacToeStates, playIn, isNaif=False) avec isNaif à True

Fonctions utilisées :

- isPlein(playIn, playedTicTacToeStates)
- getPlay(playIn, playedTicTacToeStates, numJ) : pour trouver où jouer
- stratAleatoire(numJ, playedTicTacToeStates, playIn, isNaif)

Stratégie Gain de Terrains Non Naïf (“TERRAIN2”)

Cette stratégie est la même que TERRAIN mais prend en compte cette partie de la règle alternative : “Si le carré où le joueur doit jouer est déjà gagner, celui ci peut jouer dans n’importe quel autre carré”.

Utilise les mêmes fonctions que la stratégie gain de terrains naïve mais avec le paramètre ‘isNaif’ à False.