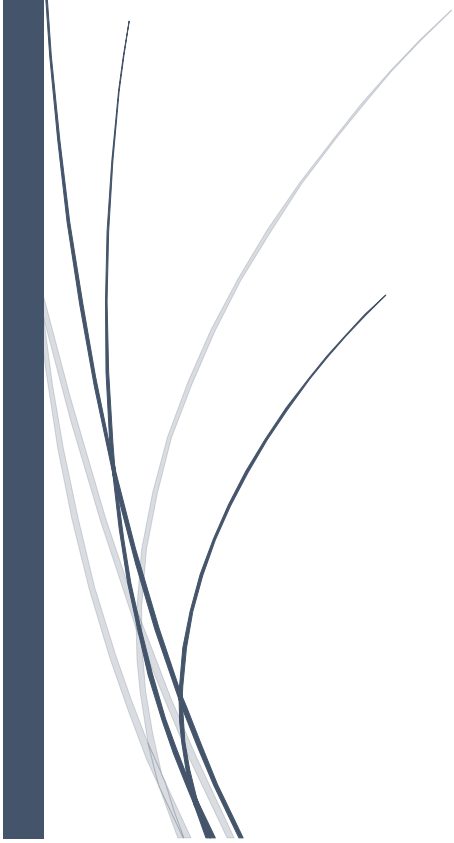
04/05/2016

Projet Tic-Tac-Toe

Partie MOIA



Casagrande Anthony & Wargnier Pierre
UNIVERSITE DE FRANCHE-COMTE

Casagrande Anthony & Wargnier Pierre

Table des matières

Introduction

Au cours du cursus de master 1, nous avions un projet de développement consistant en la création d'une IA dans un jeu de stratégie appelé le Tic-Tac-Toe.

Le jeu est constitué de 9 sous-plateaux et le but est d'aligner verticalement, horizontalement ou en diagonal, 3 points des symboles sur les plateaux. Un point est marqué lors de la victoire du sous-plateau correspondant.

L'IA de chaque groupe sera testée lors d'un tournoi qui déterminera l'ordinateur le plus performant. Cette intelligence artificielle est développée en prolog, géré par une couche java pour la gestion des coups et où chaque joueur sera connecté via un system de socket client/serveur développé en C. Dans ce rapport, nous allons vous présenter quels ont été nos choix de développement quant à la mise en place de cette intelligence artificielle.

Représentation du terrain

Lors de la création du jeu, deux solutions s'offraient à nous. La première consistait en la mise en place de 81 cases. La deuxième, en le découpage du terrain en 9 sous-plateaux eux même découpés en 9 cases.

Pour le déroulement de la partie, les cases sont initialisées à vide, et chaque joueur écrit en fonction de son symbole. Il y a donc 3 types de cases :

- x, pour le joueur 1
- o, pour le joueur 2
- ' ', pour les cases non jouées.

Principe et fonctionnement du Tic-Tac-Toe

Dans un premier temps, nous avons décidé de mettre en place deux systèmes de jeux. L'un permet de jouer contre l'IA. L'autre où l'IA joue contre elle-même. La possibilité de jouer contre l'IA n'est pas utilisée dans la finalité du projet mais elle a permis de pouvoir tester le fonctionnement de l'IA. Cela nous a été très utile pour pouvoir corriger certaines erreurs de fonctionnement. Pour ce qui est de son action, la fonction « jouer » permet tout simplement au joueur de décider sur quel emplacement jouer son coup. Le prédicat est constitué de 4^{e} arguments, dont le jeu et le joueur. On demande à l'utilisateur de choisir sur quel plateau jouer et ensuite sa position dans ce tableau. Après avoir sélectionné son coup, on fait appel à la fonction « deplacementHumain » pour effectuer le déplacement choisi et vérifier sa validité. Si le coup n'est pas correct, un message d'erreur apparait et l'utilisateur saisit un nouveau coup. On affiche le jeu avec le coup joué et on vérifie ensuite si le coup est gagnant.

Pour le cas de l'IA, le processus était plus complexe. Il va choisir le meilleur coup possible avant de pouvoir jouer. Pour cela, il fait appel au prédicat« meilleurCoup » qui sera expliqué plus précisément dans la partie suivante.

Gestion des coups

La gestion des coups va se faire via le prédicat « meilleurCoup » qui prend en compte les arguments suivants :

- Symbole du joueur (x ou o)
- L'état de la partie : si elle est gagnée, nulle ou en cours
- Coup de l'adversaire, ce qui va déterminer son prochain coup
- Et le plateau de jeu en cours, qui est initialisé à vide au départ

Lorsque l'IA commence à jouer, on a défini un coup de base pour qu'il joue dans le sous-plateau 5 et sur la case 5 du jeu. Sinon Le prédicat va retourner le meilleur déplacement de l'IA en fonction du coup adverse. Pour ce faire, un long cheminement va se dérouler, jusqu'au prédicat heuristique.

L'algorithme « mini max » permet d'optimiser des coups en minimisant le meilleur coup adverse et en maximisant son meilleur coup. Pour cela, le programme parcours l'arbre des solutions. Au moment du parcours, deux solutions sont possibles : s'il passe par le nœud des coups du joueur, alors il va retourner la meilleure solution possible pour propre coup. S'il tombe sur le nœud du parcours adverse, il va retourner le meilleur coup possible de l'adversaire. Pour obtenir justement ce meilleur coup, on fait appelle au prédicat « meilleurChoix/4 » qui nous retourne justement le meilleur coup du joueur en question. Et le prédicat « betterof/6 » qui retourne la meilleur solution dans la liste.

L'heuristique va nous permettre de parcourir l'arbre à la recherche du meilleur coup possible pour pouvoir gagner la partie. Si la valeur se rapproche de -100, alors le joueur « x » se rapproche de la victoire. À l'inverse, si la valeur se rapproche de 100, alors c'est le « o » qui se rapproche de la victoire. Le but est donc d'évaluer la qualité de ses solutions dans l'arbre pour trouver la valeur la plus élevée/basse pour gagner selon les cas.

Le déplacement dans le plateau est gérer par le prédicat « déplacement » dans mini max. Elle permet le déplacement du coup souhaité par l'IA sur la case, à condition que le coup soit en accord avec le coup suivant. Pour cela, on va notamment utiliser les prédicats de gestion de victoire et d'écriture sur les cases.

Gestion de la victoire

Il fallait gérer le système de victoire en fonction des règles du jeu. En effet, le vainqueur de l'épreuve est celui qui a marqué 3 cases avec son symbole sur 3 des sous-plateaux verticalement, horizontalement ou en diagonale. Ainsi, il était nécessaire de faire en sort que le joueur gagnant gagne un sous-plateau pour marquer ce point avec le prédicat « gagneGrandPlateau/2 » qui vérifie que les sous-plateaux correspondants ont été gagnés avec le prédicat « gagnePetitPlateau/2 ».

Pour gagner un sous-plateau, le joueur doit aligner 3 symboles de la même façon que sur le grand plateau. Il s'agit donc de vérifier que la valeur des 3 cases en question est la même. C'est le prédicat « egalite/4 » qui permet de regarder la correspondance des cases et la validité du gagnant du sous-plateau.