Guillaume MONTAVON
Benoît MEILHAC

MOIA

Rapport de projet : Le jeu de Yoté

1. Présentation du programme

Notre programme Prolog est constitué d'un seul fichier qui se nomme *ia.pl*. Les exemples d'utilisation des différents prédicats se trouvent, quand à eux, dans le fichier *README_IA.txt*.

Notre projet possède un seul prédicat principal qui se nomme *jouer* et qui est appelé dans notre programme Java (utilisant Jasper). On lui donne les pions des 2 joueurs, le dernier coup, ainsi que le nombre de pions restants dans la main du joueur, et le prédicat nous donne le meilleur coup en fonction de la configuration du plateau afin de l'envoyer à l'arbitre.

Prédicat principal : **jouer(ListePionsJ1, ListePionsJ2, NbPionsJ1, DernierDeplacement, TypeCoups, CaseDepart, CaseArrivee, Prend2emePion).**

- **ListePionsJ1** : Liste des coordonnées des pions que possède le joueur 1 (nous)
- **ListePionsJ2** : Liste des coordonnées des pions que possède le joueur 2 (adversaire)
- **NbPionsJ1** : nombre de pions qu'il reste dans la main du joueur 1 (nous)
- **DernierDeplacement** : ensemble de deux coordonnées : case de départ du dernier déplacement et case d'arrivée du dernier déplacement
- **TypeCoups** : type de coups que l'IA a choisi de jouer
- **CaseDepart** : case de départ du coup que l'IA a choisi de jouer
- **CaseArrivee** : case d'arrivée du coup que l'IA a choisi de jouer
- **Prend2emePion** : 2eme pion que l'IA a choisi de prendre

Exemple d'utilisation avec notre joueur qui possède un pion sur la case [0,0] et l'adversaire qui possède 2 pions sur les cases [1,0] et [0,3], le dernier coup joué par notre joueur est [1,0] → [0,0], et il lui reste 11 pions dans la main :

jouer([[0,0]], [[0,1], [0,3]], 11, [[1,0],[0,0]], TypeCoups, CaseDepart, CaseArrivee, Prend2emePion).

2. Structure de données choisie

Comme vu précédemment, nous avons choisit de représenter les pions qui sont sur le plateau par 2 listes :

- Liste des coordonnées des pions que possède le joueur 1 (nous)
- Liste des coordonnées des pions que possède le joueur 2 (adversaire)

Une autre méthode aurait consisté à n'avoir qu'une seule liste qui contiendrait chaque case du plateau suivi d'une lettre qui représenterait soit que le joueurs 1 possède un pion sur cette case, soit le joueur 2, soit que la case est vide.

Nous avons choisit la première solution des 2 listes car ces dernières sont beaucoup plus simple d'utilisation, ainsi lorsque le plateau est vide, les 2 listes des 2 joueurs sont tous simplement vides, ce qui simplifie grandement les calculs.

Exemple de plateau avec les configurations suivantes :

ListePionsJ1: [[0,3],[1,1],[1,3],[1,4],[2,3],[2,4]]

ListePionsJ2: [[1,0],[1,2],[3,1],[3,3],[4,3]]

Coordonnées en Y

Coordonnées en X	0	1	2	3	4	5
0				1		
1	2	1	2	1	1	
2				1	1	
3		2		2		
4				2		

3. Description de l'algorithme

Notre algorithme se décompose en 4 grandes parties, classés de la plus prioritaire à la moins prioritaire :

- **La prise d'un pion de l'adversaire :** On va vérifier lors de cette étape si un de tous nos pions sur le plateau a la possibilité de prendre un pion adverse. Si un pion a cette occasion, il va la saisir et prendre le pion adverse;
- **Le déplacement d'un pion menacé :** Lorsque l'étape précédente n'aboutis pas, on va vérifier que aucun de nos pion est menacé, si tel est le cas on va en déplacer un sur une case où il ne pourra pas être pris par l'adversaire;
- **La pose de pions :** Lorsqu'aucune des étapes précédentes n'est possible, l'IA va décider de poser un pion sur le plateau si bien sûr il en reste dans notre main. Le pion sera posé en priorité dans les 4 angles, puis dans les bords et enfin au milieu. Le pion sera bien évidement posé sur une case où il ne pourra pas être pris par l'adversaire;
- **Le déplacement :** Lorsqu'aucune des étapes précédentes n'est possible, un de nos pions se déplacera d'une case, en vérifiant bien que sur sa nouvelle case, il ne pourra pas être pris par l'adversaire.

4. Caractéristiques du programme

Notre programme permet de jouer une partie de Yoté, de manière assez défensive. Par contre, nous n'avons malheureusement pas eu le temps d'implémenter un algorithme de MiniMax. La faiblesse de notre programme est donc qu'il ne peut pas prévoir les coups à l'avance.

L'avantage de notre algorithme est qu'il est très rapide, il permet de répondre presque instantanément, en donnant un coup valide où il ne pourra pas être pris. Nous sommes donc sûr, lors du tournoi de ne pas avoir de timeout, donc de ne pas perdre bêtement. Notre algorithme nous permet donc de se déplacer lorsque qu'un de nos pions est menacé, de prendre un adversaire dès que possible, de se déplacer, et de poser des pions (dans les bords, si possible). Notre programme est donc assez défensif, et nous permettra, nous l'espérons d'obtenir un grand nombre de match nul lors du tournoi.