Rapport Intelligence Artificielle - Puissance 4

Le code source du projet est disponible à cette adresse :

https://github.com/laroseg54/ProjetIntelligenceArtificielle

Description du jeu :

Le puissance 4 est un jeu de stratégie à 2 qui consiste à jouer tour à tour dans le but d'aligner 4 pions d'une même couleur dans une grille de 6 lignes sur 7 colonnes. Si les 42 pions sont posés sans alignement, la partie est alors considérée comme nulle.

Le jeu présente les caractéristiques suivantes :

- totalement observable
- déterministe
- épisodique
- statique
- multiagents (2 joueurs)

Implémentation du Jeu :

Le jeu a été crée en java , en utilisant la librairie JavaFX pour l'interface graphique, il se compose des classes suivantes :

classe Jeu :

C'est la classe centrale du programme, elle se base sur un tableau à doubles dimensions de Pions (type énumération qui peut contenir les valeurs JAUNE , ROUGE , VIDE).
A chaque fois qu'un joueur cherche à placer un pion, elle recherche si ce placement est valide (si la case n'est pas déjà occupé et que le plateau n'est pas rempli) et le cas échéant vérifie si le joueur à gagné ou non.

classe Coordonnée :

Cette classe a pour rôle de vérifier qu'une position ne sort pas de la grille.

Classe Joueur :

Classe abstraite qui contient permet à un joueur de placer un pion dans la grille, de choisir sa couleur et de connaître son adversaire.
Toutes les autres classes de joueurs descendent d'elle.

Classe Joueur Humain :

Cette classe n'implémente aucune nouvelle méthode ni attribut , elle permet seulement à la classe PlateauDeJeu de savoir à quel type de joueur elle à affaire grâce à l'utilisation du comparateur instanceof.

Classe PlateauDeJeu :

Cette classe permet le bon fonctionnement de l'interface graphique , c'est aussi elle qui s'occupe de faire jouer les différents adversaires à tour de rôles.

Classe Heuristique :

Cette classe possède des fonctions permettant de renvoyer un score pour une grille donnée par rapport à un joueur.
Plus la grille est favorable au joueur , plus le score sera élevé.
Le calcul du score se fait la façon suivante :

- L'algorithme va parcourir tous les quadruplets de cases consécutives possibles de la grille (en verticale, horizontale et diagonales)
- Chaque quadruplet va se voir attribuer un score.
- Ce score sera de 0 si aucun pion n'est présent dans le quadruplet ou bien si plusieurs pions sont présent mais qu'ils ne sont pas de même couleur
- Autrement il sera de 10^nbrPionsJoueur ou de -10^nbrPionsAdversaire
- Une fois qu'on a calculé le score d'un quadruplet , on l'ajoute à la somme de la grille

Classe ArbreMinMax :

Cette classe permet de construire un arbre pour l'algorithme min Max avec ou sans élagage Alpha Beta.
Les nœuds et feuilles de l'arbre correspondent chacun à un état de la grille de jeu.
L'arbre est construit de façon récursive par un parcours en profondeur, quand il atteint une feuille (soit dans le cas ou la partie est finie , soit dans le cas ou la profondeur maximale fixée à été atteinte) , il fait appel à la classe heuristique pour évaluer cette feuille puis fait remonter le score selon au nœud précédent.
Les nœuds qui ne sont pas des feuilles choisissent un score idéal parmi leurs enfants selon qu'ils sont des nœuds Min ou des nœuds Max.
Ainsi au moment ou l'arbre a finit de se construire , le score correspondant au choix idéal à faire pour l'ia est remonté à la racine.

Classe JoueurAuto :

Elle utilise la fonction d'évaluation de la classe heuristique pour choisir le meilleur coup parmi ceux qui lui sont possibles (sans chercher a calculer les coups de l'adversaire à l'avance). Pour cela elle simule les coups l'un après l'autre dans la grille prend le maximum de l'évaluation de ces coups.

Classe JoueurMinMax :

Elle utilise la fonction d'évaluation de la classe heuristique et la classe ArbreMinMax pour choisir le meilleur coup parmi ceux possibles en tenant compte d'un certain nombres de coups futurs.
Par défaut le mode minMax sans élagage peut calculer jusqu'à 7 coups à l'avance et le mode minMax avec élagage AlphaBeta peut lui en calculer jusqu'à 9. Au-delà de ces valeurs on obtient soit un dépassement de mémoire soit un temps de calcul beaucoup trop long pour jouer dans de bonnes conditions.

Test de l'IA :

-En jouant contre le joueur auto de base avec un poids des pions adversaires symétriques à celui des pions IA nous avons remarqué que celle-ci ne bloquait pas les alignements de 4 cases . Pour remédier à cela nous avons changé le poids des pions de l'adversaire à -15^nbrPionsAdversaire.

- Nous avons tenté aussi de diminuer pour tous les joueurs auto ,le poids de l'adversaire quand l'ia joue en premier , le but étant de la rendre plus offensive mais cela n'a pas donné de bons résultats.

- Nous avons également tenté d'augmenter le score d'une partie qui ferait égalité mais la encore , les résultats n'ont pas été très probants.

- Nous avons remarqué que les IA battaient systématiquement celles qui avaient un arbre moins profond, toutefois cette victoire ne se produit que quand la grille est proche d'être pleine.

- Au niveau des performances contre l'humain , nous avons réussi à battre quelquefois l'ia quand celle-ci avaient un arbre de faible profondeur(profondeur de 5 ou 3), globalement même à de faibles niveaux de profondeurs l'IA gagne souvent en profitant de nos faiblesses.

Pour finir, voici un tableau représentant une moyenne du temps en exécution en nanosecondes de chaque ia sur ses 4 premiers coups :

	JoueurAuto	JoueurMinMax	JoueurAlphaBeta
Coup 1	2463400	1081511800	87986900
Coup 2	101500	1237973000	44600100
Coup 4	97800	1305186700	66944500
Coup 5	88200	955896000	45541900

Pour réaliser ces moyennes les ia MinMax max et AlphaBeta ont toutes les deux été paramétrées à une profondeur maximale de 7, on peut constater que l'élagage AlphaBeta apporte une belle amélioration de performance.