École de technologie supérieure

Laboratoire Jeu plateau

Jeu de Tic Tac Toe

Patrick Cardinal **Durée** : 3 séances

Introduction

Le jeu de Tic Tac Toe est bien connu et facile à jouer. Ses règles simples et l'espace de recherche limité en font un jeu idéal pour une première implémentation des algorithmes MinMax et Alpha-Beta car, entre autres, le débogage est plus simple étant donné la taille limitée de l'arbre de recherche. Ce laboratoire servira donc de point de départ pour le laboratoire de jeu de plateau qui demande une bonne compréhension de l'algorithme Alpha-Beta.

Les objectifs de ce laboratoire sont :

- Implémentation de l'algorithme minimax
- Implémentation de l'algorithme Alpha-Beta
- Mieux comprendre l'architecture pour l'implémentation d'un agent intelligent pour un jeu de plateau à deux joueurs.

Travail à faire

Pour ce laboratoire, vous devez implémenter un agent intelligent capable de jouer au Tic Tac Toe. Normalement, étant donné la faible complexité du jeu et par conséquent, de son espace de recherche, votre agent ne devrait jamais perdre une partie. Dans le pire des cas, votre agent devrait toujours obtenir un match nul contre un joueur qui joue parfaitement.

Un modèle d'architecture est disponible. Les classes suivantes sont fournies :

- Mark.java : définition des pièces du jeu, soit le X et le O.
- Move.java : classe qui contient les informations relatives au mouvement, autrement dit, la position à laquelle une pièce est jouée.
- Board.java : classe qui contient les informations du plateau. Les signatures des trois méthodes sont fournies dans cette classe :
    - public void play(Move m, Mark mark) qui place la pièce mark à la case spécifiée par move
    - public int evaluate(Mark mark) qui doit retourner 100 pour une victoire, -100 pour une défaite et 0 pour un match nul.
    - public ArrayList<Move> generateMoves(Mark m) qui retourne la liste des coups possibles en fonction du plateau.

  Il est particulièrement important de ne pas modifier les signatures de ces méthodes, elles seront utilisées pour tester votre implémentation. C'est la même consigne pour le nom de la classe et la signature du constructeur.

- CPUPlayer.java : classe qui contient les méthodes qui implémentent l'agent intelligent. Les signatures de trois méthodes sont fournies dans cette classe :
    - public int getNumOfExploredNodes() qui retourne le nombre de nœuds explorés durant une recherche MinMax ou Alpha-Beta. La variable numExploredNodes devrait être incrémentée à chaque appel de votre méthode MinMax ou AlphaBeta. Idéalement, l'incrémentation est la première chose qui est faite dans ces deux méthodes.
    - public ArrayList<Move> getNextMoveMinMax(Board board) retourne la liste de tous les coups qui ont la même valeur en utilisant l'algorithme MinMax. Le nombre de coups avec la même valeur dépend de la configuration des pièces sur le plateau.
    - public ArrayList<Move> getNextMoveAB(Board board) retourne la liste de tous les coups qui ont la même valeur en utilisant l'algorithme Alpha-Beta. Le nombre de coups avec la même valeur dépend de la configuration des pièces sur le plateau.

Le constructeur de cette classe reçoit en paramètre le joueur Max (X ou O). Tout comme pour la classe précédente, il est important de ne pas modifier les signatures des méthodes, mais vous pouvez ajouter d'autres méthodes au besoin. Au minimum, vous devrez ajouter votre implémentation de MinMax et Alpha-Beta (deux méthodes distinctes)

Travail à effectuer

Séance 1: Vous devez implémenter les fonctions de base de votre jeu de tic tac toe :

- Demander à l'utilisateur le prochain coup à jouer
- La méthode qui vérifie si la partie est terminée
- Le générateur de mouvements
- La méthode Play()

À la fin de cette séance, vous devriez être en mesure de pouvoir jouer une partie avec un collègue.

Séance 2: Vous devez implémenter l'algorithme minimax. À la fin de cette séance, vous devriez pouvoir jouer contre votre algorithme qui ne devrait jamais perdre.

Extra: Vous pouvez implanter des niveaux de jeu. Pour ce faire, vous pouvez limiter la profondeur de recherche de l'algorithme minimax. Dans ce cas, vous allez devoir aussi ajouter une fonction d'évaluation qui retourne une valeur entre -100 et 100 en fonction des possibilités de gain du joueur Max.

Séance 3: Vous devez implémenter l'algorithme alpha-béta. À la fin de cette séance, votre jeu jouera de la même façon, mais il sera beaucoup plus efficace. Vous pouvez vérifier l'effet de l'algorithme alpha-beta en comparant le nombre de nœuds explorer avec et sans l'alpha-beta.

Extra: Vous pouvez utiliser une table de hachage pour précalculer les symétries du plateau. Cet ajout ne devrait pas modifier les coups sélectionnés par votre algorithme, mais devrait réduire encore plus le nombre de nœuds explorés.