Exercice 05

Jeu d'échecs

1. Enoncé

L'objectif est de réaliser le programme qui permet à un ordinateur de jouer à un jeu de plateau comme les échecs (jeu à 2 joueurs) afin de déterminer le meilleur coup à jouer.

Le principe est de définir une architecture des classes (Designs Patterns) facilitant la mise en œuvre d'un tel programme permettant de tester différentes stratégies de jeu et de faciliter la programmation à n'importe quel jeu comme les échecs, les dames, le go, ...

Pour un tel programme, il faut définir une **fonction d'évaluation** qui s'applique sur **un état du jeu** à un instant donné.

Cet état est l'assemblage de tout un ensemble de données comme :

- la configuration du plateau de jeu (dimension, contraintes). Remarque : pour le jeu d'échec cette configuration est toujours la même mais pour certains jeux cette configuration peut être initialement pas la même (go) et pourrait même changée pendant la partie (champ de bataille).
- la disposition et la nature des pièces disposées sur le plateau
- les règles du jeu qui permet de déterminer en fonction d'un état donné du jeu (et de la configuration si celle-ci évolue) de déterminer tous les coups qu'un joueur peut jouer à partir d'un état.
- les pièces prises par chacun des joueurs qui permettent de déterminer, en fonction d'une "valeur" de chacune des pièces, une valeur totale de prises pour chacun des joueurs
- l'expérience acquise
- les caractéristiques du joueur (utilisé dans le cas de jeu de rôle. Non utilisé pour les échecs)
- … etc …

Le principe pour déterminer le "meilleur coup" d'un joueur consiste à créer un **arbre de décision** d'une certaine profondeur dont chaque nœud contient deux informations :
- l'état du jeu
- la "valeur du jeu" qui est le résultat de la fonction d'évaluation.

On construit cet arbre récursivement suivant le principe de l'algorithme minimax (https://fr.wikipedia.org/wiki/Algorithme_minimax). Cela consiste à passer en revue toutes les possibilités pour un nombre limité de coups et à leur assigner une valeur qui prend en compte les bénéfices pour le joueur et pour son adversaire. Le meilleur choix est alors celui qui minimise les pertes du joueur tout en supposant que l'adversaire cherche au contraire à les maximiser. Cela revient à appliquer l'algorithme suivant :

- minimax(p) = f(p) si p est une feuille de l'arbre où f est la fonction d'évaluation
- minimax(p) = MAX(minimax(O_1),…,minimax(O_n)) si p est un nœud Joueur avec O_i les fils
- minimax(p) = MIN(minimax(O_1),…,minimax(O_n)) si p est un nœud Opposant avec O_i les fils

Il existe beaucoup de variantes possibles de la fonction d'évaluation f(p), l'objectif étant de pouvoir tester différentes stratégies, plus ou moins sophistiquées.

2. Correction

Commentaires en cours