Marlon Aquino

CS 4200.01

Professor Dominick Atanasio

Project 4 Isolation

Purpose

The purpose of this project is to implement a time-based two player game of Isolation. Isolation is a game where each player attempts to have their opponent in a way where they cannot make any more moves. Each player can move like a queen in chess as long as its path does not cross a square already used or currently occupied. The computer was implemented using the minimax algorithm with alpha-beta pruning. Each move the computer makes is based on an evaluation function that balances aggressiveness and defensiveness.

Summary

In order to calculate the best move possible, the Minimax algorithm was implemented. The Minimax algorithm is a popular algorithm especially in multiplayer games. The overall idea behind the algorithm is the evaluation of a state in a game where one player wants to maximize its score and the other wants to minimize its score. The algorithm helps to calculate a move in that state that would bring it closer to winning. This calculation involves a recursive call of minimum and maximum steps where the computer would try to find the best move while expecting that the opponent would make the least favorable move against the computer's goal. With this, the computer is able to evaluate future possible states in an attempt to bring the current state closer to a winning position. Each position in the board is given a score depending on the heuristic evaluation.

The alpha-beta pruning method allows for faster calculation of the Minimax algorithm. The number of possible paths that could be taken is huge especially in a game of Isolation, thus it would take a very long time to check every possible path that could be taken at any point. By using the alpha-beta pruning method, the minimax algorithm can eliminate many states because there is a better move already available.

Alpha represents the highest value max and beta represents the highest value for min. If the beta score is less than or equal to the alpha score, then there is no need to search the rest of the nodes as there is no higher score that can be found for beta.

Evaluation Function

For my evaluation function, I first decided to use a heuristic that would return the degrees of freedom that a player has, also known as the number of possible moves it could take. In order to balance aggressiveness and defensiveness, I wanted to take the difference between the number of valid moves for both the computer and opponent. In this way, the computer would make the move that would decrease the number of valid moves for the opponent as well as moving to the position that would maximize its number of valid moves.

Problems Encountered

There were many problems that I had encountered for the project. When it came to testing, it was often difficult to keep track of the bounds of the board so I had gotten a lot of ArraysOutOfBounds Exceptions. I also had trouble generating the successors of the current state of the board. At first I tried using the .clone() method to try and generate successors of the current board, however I did not know that it returned a shallow copy which in turn changed the original Board object. To solve this I created a method that would return a deep copy of the current state so as to not change the original. Generating the moves list to the right was difficult as well as there were many formatting issues that were involved and it had gotten more difficult as the moves went out of the board size.

Initially the heuristic function that I used only made moves that would maximize the number of valid moves for the computer and did not consider the opponent's

position. However, I decided that this was not optimal because after playing against the AI, it would often not make the winning move or make a move that would isolate itself.

The time constraint especially for the alpha-beta pruning algorithm was probably the most challenging for me. Since the algorithm was recursive, it was hard to keep track of the time since there were multiple callbacks. It took some time, but I eventually solved it. The time limit for the user input was still challenging because I tried to find a way to time it as the user is making an input, but to do this I had to use a java package. Overall I learned a lot from this project and thought that this was challenging, but very rewarding after completing it.