Anthony Tran
ID: 160594898

ECS629U – Artificial Intelligence – Assessment 1 Report

The first step is to save the passed arguments (current state of board, and the computer's/player's coloured piece) from the overridden method **`chooseMove()`** to private variables, allowing the access these values in other methods. After this, the method **`alphaBetaPruning()`** is called which takes four inputs: the depth level (in this case it is 4), alpha (which is the minimal value – negative infinity), beta (which is the maximum value – positive infinity) and a boolean (which keeps track of whether the depths are minimising or maximising). This then returns an integer array of size three, which are: current score, and the row and column corresponding to the score. As the name suggests, the method uses the concept of Alpha-Beta pruning in which 'cuts off branches' when the value of alpha is bigger or equal to the value of beta. Doing this allows for an increase in time efficiency for each move. Before the pruning starts, the method **`checkboard()`** is called which searches through the current state of the board and returns a list of all possible coordinates (rows and columns) that are available.

To evaluate the score the method **`scoreEvaluation()`** goes through all of the possible 5-in-a-row combinations (first checking the rows, then the columns and finally the diagonals) using nested for-loops to traverse each of the combinations. The score is calculated through calling the method **`lineEvaluation()`** in which takes the coordinates of each of the 5-in-a-row pieces. With this the score is based on heuristic board evaluation, this is essentially scoring based on how much in-a-row of the pieces have. The higher the number of in-a-row the pieces have, the higher score (for when the computer is playing a piece) or the lower score (for when the opponent is playing the piece).

The strategy used was: for 1-in-a-row the score is 1 if the piece there is the computer's colour, -1 if the piece is the opponent's colour and otherwise the score is 0 (as the piece is null). Now as said before, the higher the number of in-a-row there is the higher the exponential increase/decrease of the score. However, the opponent's score will have a higher exponential decrease compared the exponential increase of the computer. This is because the situation is worse for the computer if the opponent has a higher degree of in-a-row than if the computer itself has a higher degree of in-a-row. Therefore, score assigned would be: for 2-in-a-row the computer will have 10 and the opponent will have -20; for 3-in-a-row the computer will have 100 and the opponent will have -300; 4-in-a-row the computer will have 1,000 and the computer will have -40,000; and finally for 5-in-a-row the computer will have 10,000 whereas the opponent will have -5,000,000. This way the computer would play more defensive than offensive and have a lower chance of losing.