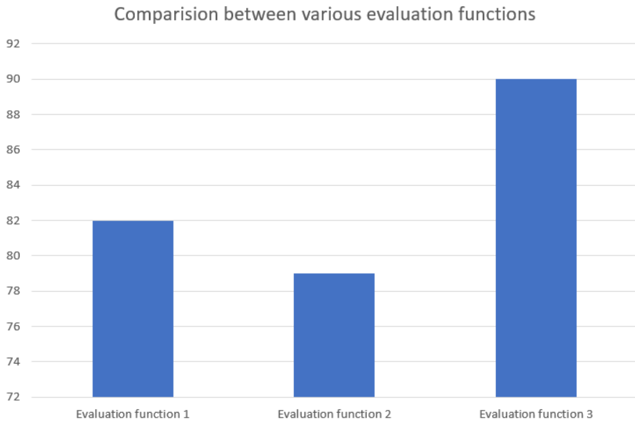
# AI Assignment 2 - Report

Abhiram Ajith
2021A7PS2525G

1. The heuristic evaluation function used by the Game-Tree player program uses a method for counting the number of pieces of the given player in a particular row, column or diagonal and assigns scores to these evaluations wherein higher score is given to configurations which have more number of pieces. The function also gives higher weightage to pieces in the middle of the board as they have a higher chance to produce a favourable outcome (chances of winning from pieces in middle rows and columns are higher, as diagonals can be easily formed).

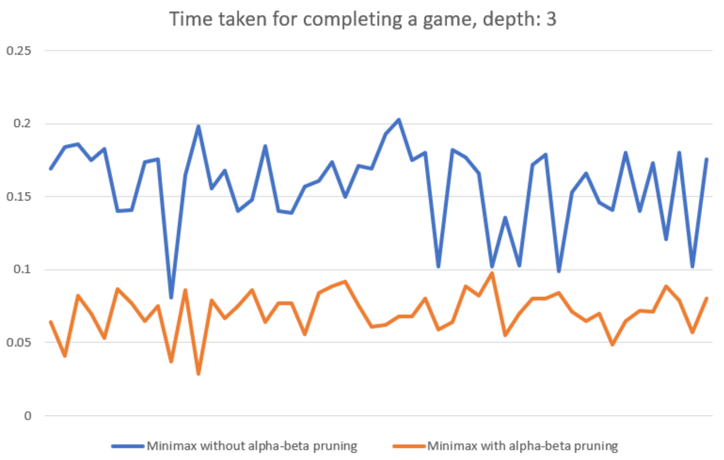
   Here, we have compared different approaches to the evaluation function and have chosen one which produced the best results when tested. The evaluation function which consistently gives the best winning rate was chosen, which is able to produce a favourable outcome for ~90% of games even with a game tree depth of 3.

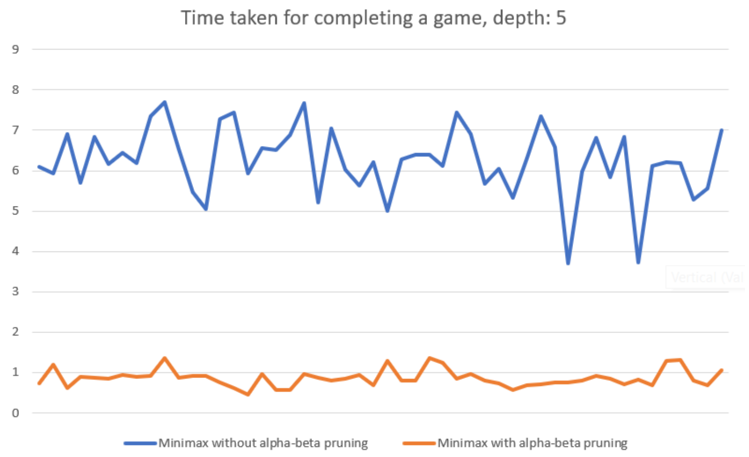


2. Alpha-beta pruning was added to the minimax algorithm, in order to reduce the time the program took to make a decision on which move to do, by pruning away other branches which did not affect the final outcome.

   After testing the effectiveness of alpha-beta pruning, it was found that for a game tree with depth 3, the average time for completing a game went from 0.16s to 0.071s, a 56% reduction in runtime. This effect is even greater for a game tree with depth 5: the average time for completing a game went from 6.237s to 0.864s, a 86% reduction in runtime.

   These results show the effectiveness of alpha-beta pruning in helping the minimax algorithm arrive at an optimal solution in a much more efficient way.



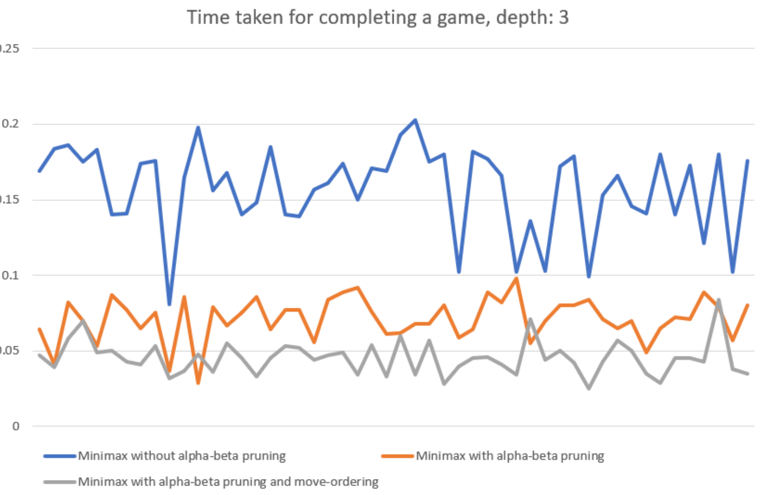


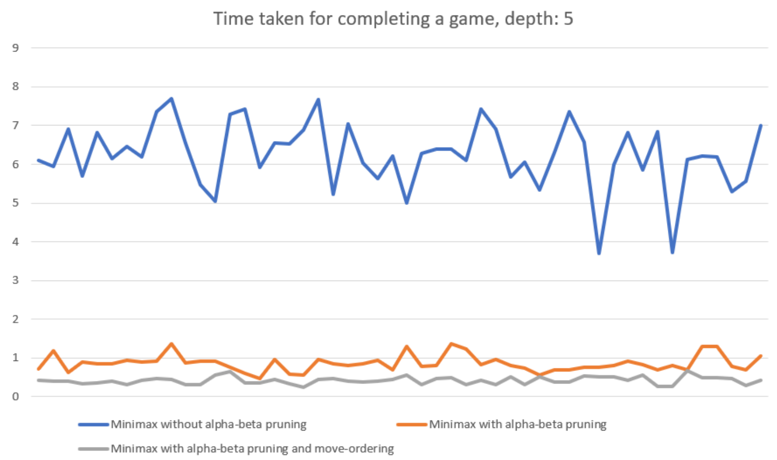
3. The next step to improve the runtime performance is to implement move ordering heuristic. Since our heuristic evaluation function gives higher weightage to pieces in the middle of the board, it follows that the minimax algorithm will prefer to place a piece on the centre of the board. Therefore the possible locations to place a piece should be checked in such an order where the middle columns are checked first, and then it progresses to the side columns i.e. when surveying the valid locations to place a piece, the columns should be checked in the order [3, 2, 4, 1, 5, 0, 6].

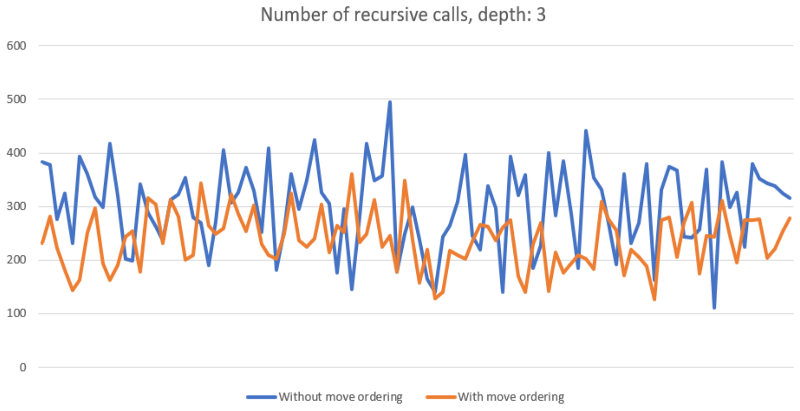
   On implementing the move ordering heuristic, when compared to the minimax algorithm with alpha-beta pruning without move ordering, it showed a decrease in runtime from 0.071s to 0.045s, a 37% reduction for a game tree with depth 3, and from 0.864s to 0.418s, a 52% reduction for a game tree with depth 5.

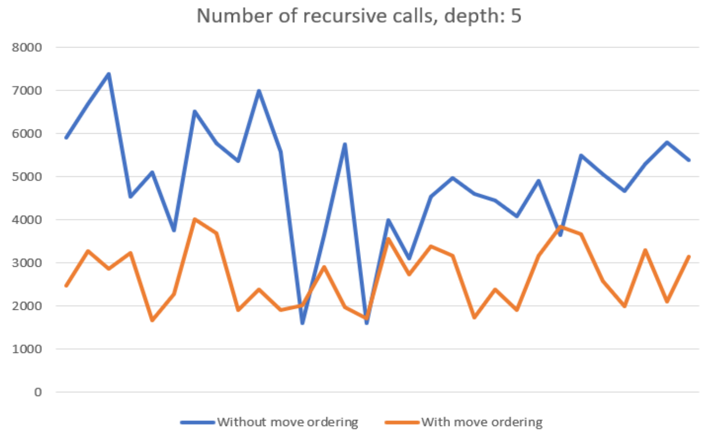
   Correspondingly, the average number of recursive calls in a game reduced from 300 to 237, showing a 21% reduction for a game tree with depth 3 and from 4872 to 2698, showing a 45% reduction for a game tree with depth 5.

   These results show the effectiveness of an efficient move ordering scheme in addition to alpha-beta pruning from improved performance.


Time taken for completing a game, depth: 3
.25
0.2
.15
0.1
.05
0
Minimax without alpha-beta pruning
Minimax with alpha-beta pruning
Minimax with alpha-beta pruning and move-ordering


Time taken for completing a game, depth: 5
9
8
7
6
5
4
3
2
1
0
Minimax without alpha-beta pruning
Minimax with alpha-beta pruning
Minimax with alpha-beta pruning and move-ordering

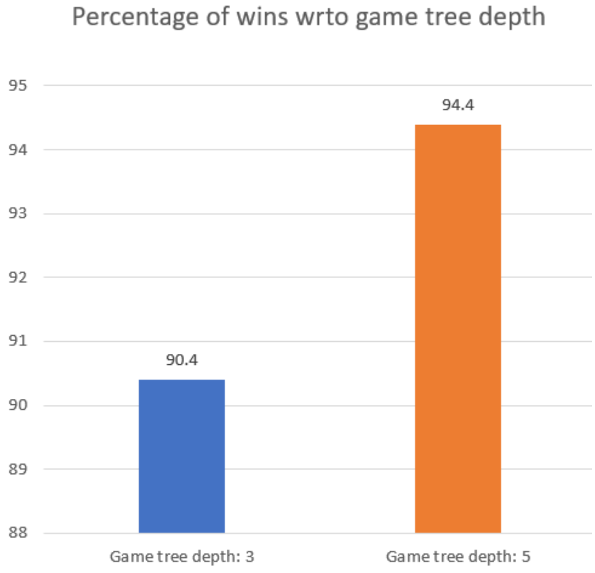

Number of recursive calls, depth: 3
600
500
400
300
200
100
0
Without move ordering
With move ordering


Number of recursive calls, depth: 5
8000
7000
6000
5000
4000
3000
2000
1000
0
Without move ordering
With move ordering

4. The game tree, designed using the Minimax algorithm, with the depth of the game tree set to 3, was able to beat the myopic player in 90.4% of the games (out of 250 games). When the depth of the game tree was increased to 5, it showed an improvement; it was able to win 94.4% of the total number of games.

Therefore, increasing the game tree depth shows a small but significant increase in the frequency of winning. The rather smaller increase in frequency of winning is due to the fact that the opponent is an imperfect myopic player so the game tree player does not need to look a lot further to beat it.

However, when the average number of moves before a win was compared between a game tree depth of 3 and 5, it showed no improvement; the average number of moves remained 21. This result is due to the fact that the minimax algorithm does not try to minimise the number of moves before it wins, it just tries to get the sequence of actions which gives the highest chance of a win.


Percentage of wins wrto game tree depth
Game tree depth: 3
Game tree depth: 5