CMPT 310

Assignment5

Connect4

Chen Zhao

301308092

Introduction:

In this assignment, I implement two classes of connect 4 board games in c++ language that use Monte Carlo Tree search algorithm and a heuristic algorithm utilizing the next two steps.

Part one:

In this part, I construct the class of connect four which includes the constructor function as well as destructor function to initialize a two dimensional array and a one-D array that simulate the game board and save the available space for next moves. Then some primary functions combined to simulate humans thoughts and start the game are listed below:

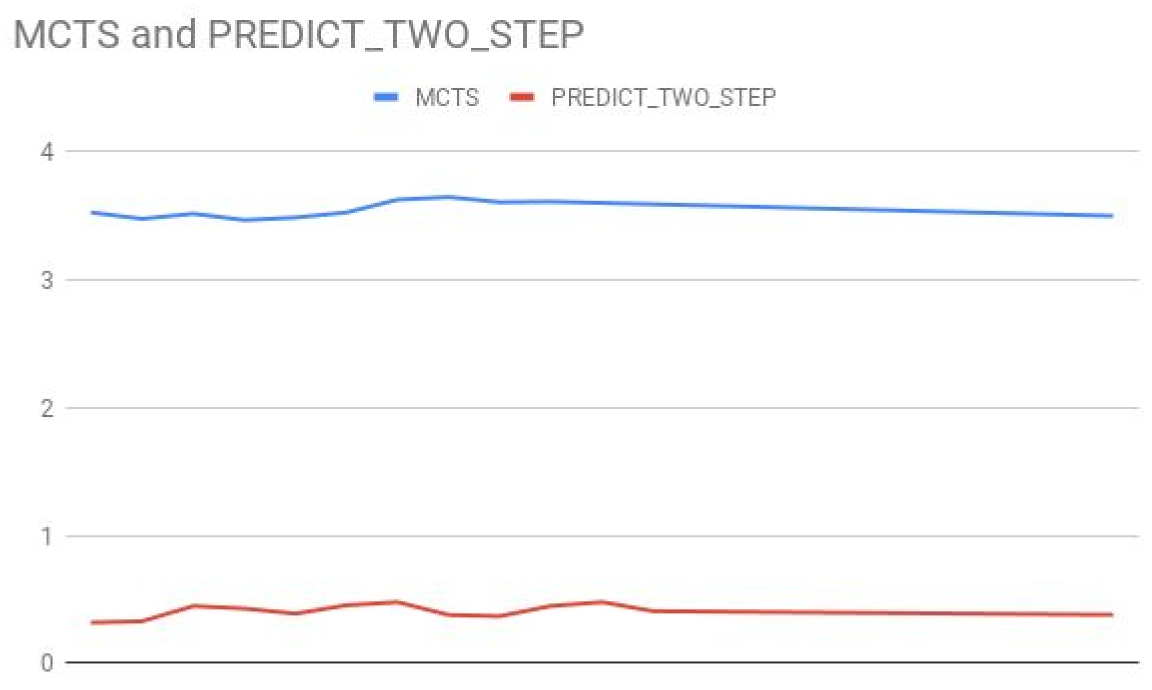
1. **get_next_move()**
   This function is used to get user's input and filter the unvalid input, such as digits not between 1 to 7 or other irrelevant symbols, until it receives the acceptable inputs.
2. **display()**
   This function is designed to draw the game board as user's interface and update when user or computer make their next moves.
3. **check_win()**
   This function is used to check if the most recent move leading to a win, lose or draw from basically four directions which includes " \ " diagonal, " / " diagonal, horizontal and vertical. During each direction, I iteratively add one to "count". If "count" once equals to four, game has ended. Otherwise, it will keep refresh to zero, until all direction has been checked and return 0 to imply draw (board has been full) and -1 to imply game should keep continues.
4. **Legal_move()**
   This is a short function to return which column is not empty by using a vector to store, since vector is flexible enough to handle this dynamic situation.
5. **check_full()**
   A helper function for check_win() function to check whether the board is full.
6. **Next_move()**
   In this function, Monte Carlo Tree Search are implemented to randomly simulate two players' moves and all available moves are iterated 10000 times for each available spot. Save the score and moves to find the highest win rate then return the corresponding move as the reaction displayed on the user interface.
7. **game_start()**
   This function are implemented to start the game with choice of 1 or 2 for user first move or computer first move respectively.

Part two:

In this part, I construct basically the same function as part one. The only thing different is the function to simulate AI thoughts.

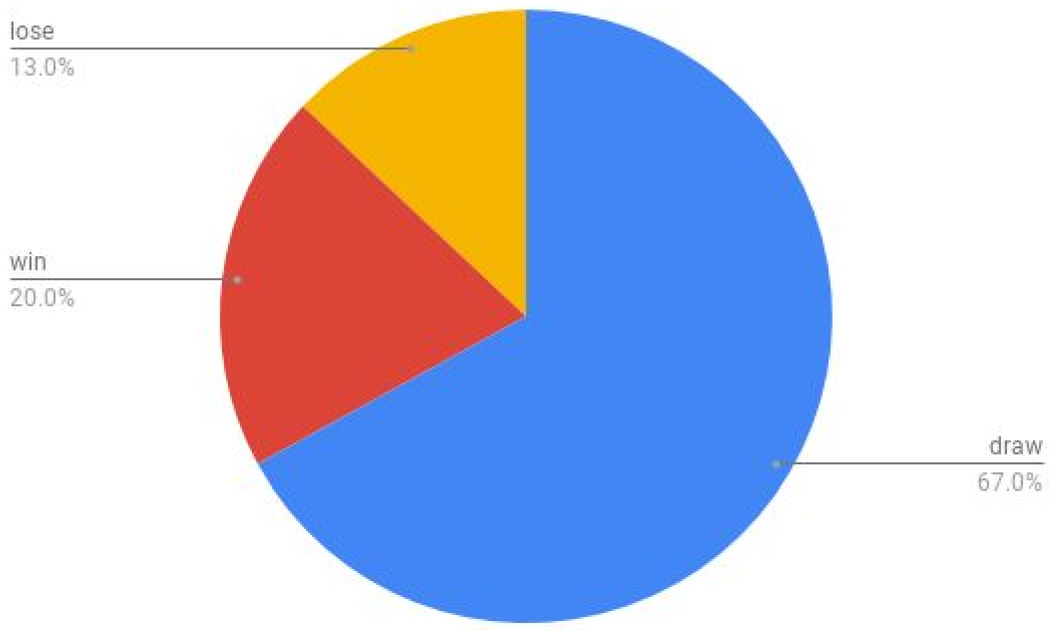
next_move()

In the previous function, I realize that the operation time has to be long enough to accurately predict the next move. Therefore, I try to use heuristic algorithm to predict the best next two steps which only focus on decreasing opponents' win rate and never lose. By analysing the result of this algorithm, this algorithm is not that smart enough to make the perfect choices but good enough at the aspects of reducing the running time and decreasing the win rate.



Comparison

By simulating 100 games of connect 4, most games are ending at draws but still Monte Carlo Tree Search has more chance to win. Conclusively, there is still some improvements for my heuristic algorithm but the speed of it is strongly shorter than MCTS.



// graph for the rate from the aspects of MCTS