Summary of project

Zining(Kimi) Liu & Lixin(Leo) Li

In this study, we perform a series of tests on the Monte Carlo Tree Search (MCTS) algorithm to find the most suitable c-value. This summary only contains description of the experimental process and results. The whole implementation is in the README. Our tests cover the results of the competition at different c-values, the number of simulations, and the simulation depth.

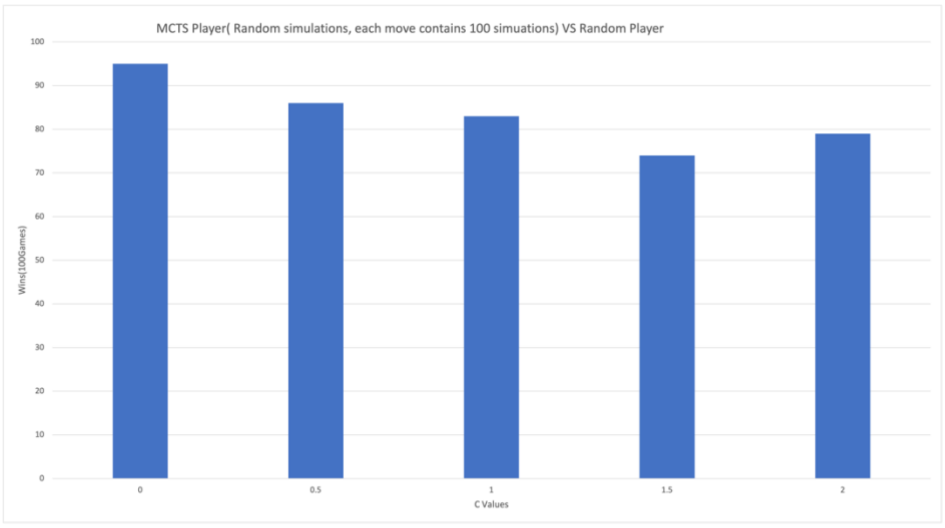
Throughout the testing process, each test takes a significant amount of time, and in many instances, we are unable to test all C values due to time constraints. For Random simulations, when the MCTS player and Random player play a game, every 100 games (for each C Value) take approximately one hour. If the MCTS player and Alpha Beta Player play a game, every 100 games take nearly two hours. For Alpha Beta simulations, every 20 games require one to two hours. Unfortunately, we do not have enough time to perfectly test each C Value. Therefore, when searching for the C value range, if we observe a noticeable drop in the win rate, we will manually terminate the program and record the results from previous tests.

In this experiment, we did not record the time it took for each test, for two main reasons. The first reason is due to time constraints. The second reason is that we used various computers to run these tests simultaneously. The devices used in this Final Project include: Windows with a 13900KF CPU (maximum frequency of 5.6GHz), MacBook

Pro with an Intel i5 CPU (maximum frequency of 2GHz), MacBook Pro with an M1 chip (maximum frequency of 3.20GHz), and school computers (specific information is unknown). We initially borrowed a Windows system with an AMD3990X CPU, but we found that having a multi-core CPU did not significantly help in running the code. The speed of executing the code mainly depends on the frequency of the base clock, so we later switched to a 13900KF CPU.

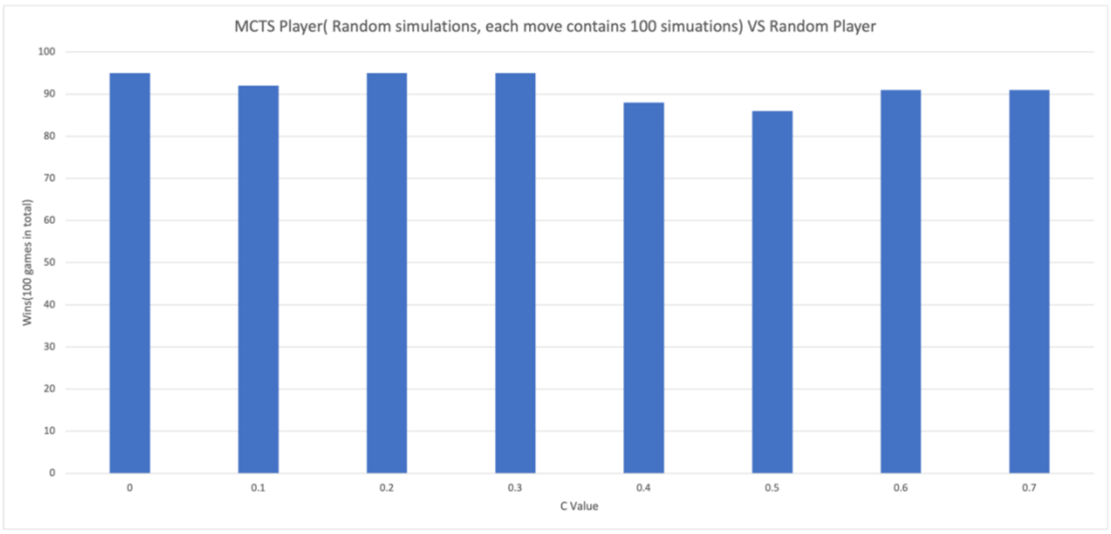
Our concept involves initially determining a range for the C Value by having the MCTS Player and Random Player compete in the game. Subsequently, we will identify the optimal C Value within this range by having the MCTS Player and Alpha Beta Player participate in the game.

First, we test MCTS Player(simulation type = random, simulations = 100) vs Random Player. The range of C is 0 – 3, step = 0.5. We test each C Value for 100 games. Because win rate decreases when C value is larger than 1, we terminate the program.(Figure 1)



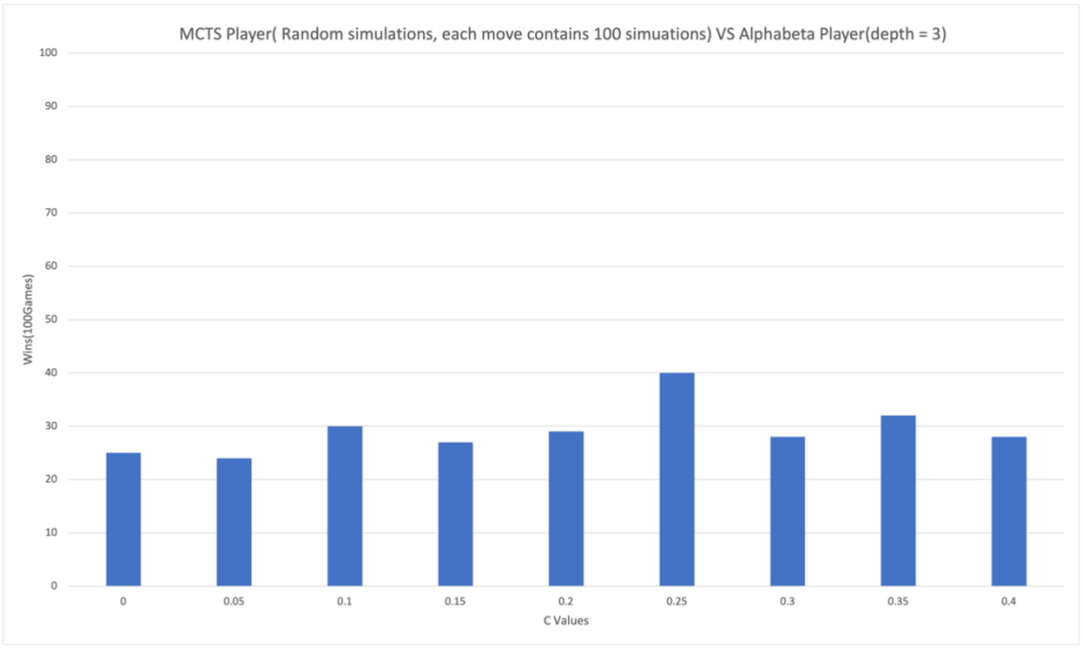
(Figure 1)

Since the win rate clearly decreases when the C Value is greater than 1, we have found a relatively large interval for C. Next, we need to narrow down the range of the C Value by changing the range of C. The Range of C is 0-1, step – 0.1. MCTS Player (simulation type = random, simulations = 100) vs Random Player. (Figure 2)



(Figure 2)

Since the winning rate significantly decreases when the C Value is greater than 0.4, we speculate that the value of C might be within the range of 0 to 0.4. Next, we will test MCTS Player (simulation type = random, simulations = 100) vs Alpha Beta Player(Depth = 3). Similarly, each C value will be tested in 100 games. (Figure 3)

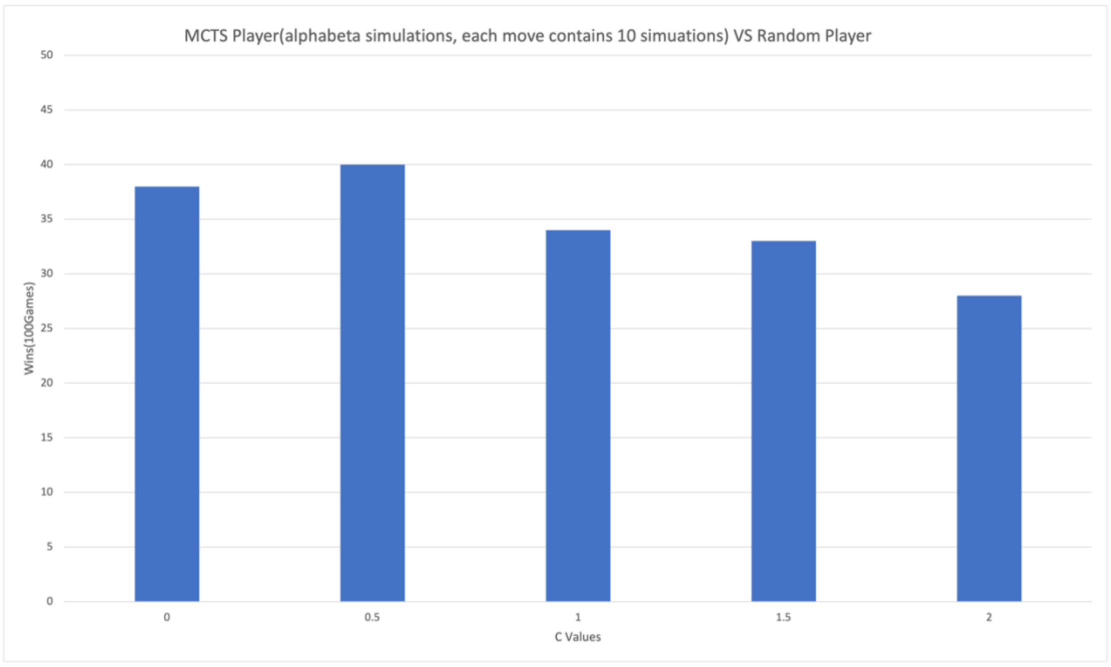


(Figure 3)

Based on the test results, we came to the final conclusion that c=0.25 is the most suitable c-value for our MCTS algorithm.

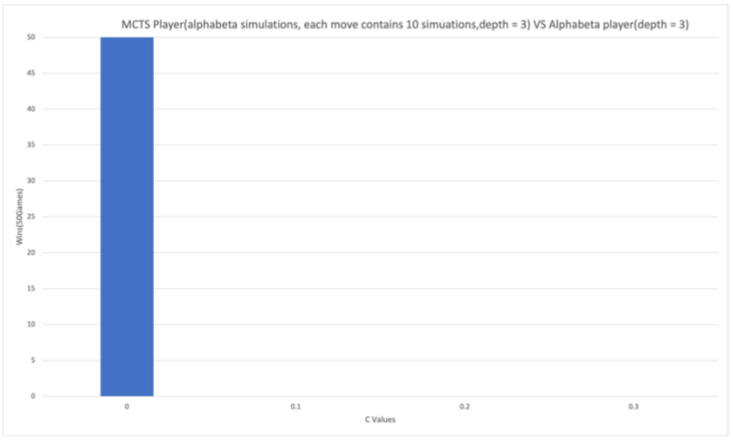
The above is about all tests for MCTS with simulation type = random. Next, we will test the MCTS Player with simulation type = alphabeta. At this point, things become more complex and interesting.

Due to the enormous amount of time spent by MCTS with simulation type = Alphabeta, we have reduced the simulations for each move to 10. Even so, the testing time is still much longer than that of random simulations. The next test is MCTS Player(Simulation type = alphabeta, simulations = 10) vs Random Player. The range of C is 0 – 4, step = 0.5. Each C value only will be tested in 50 games. (figure 4)



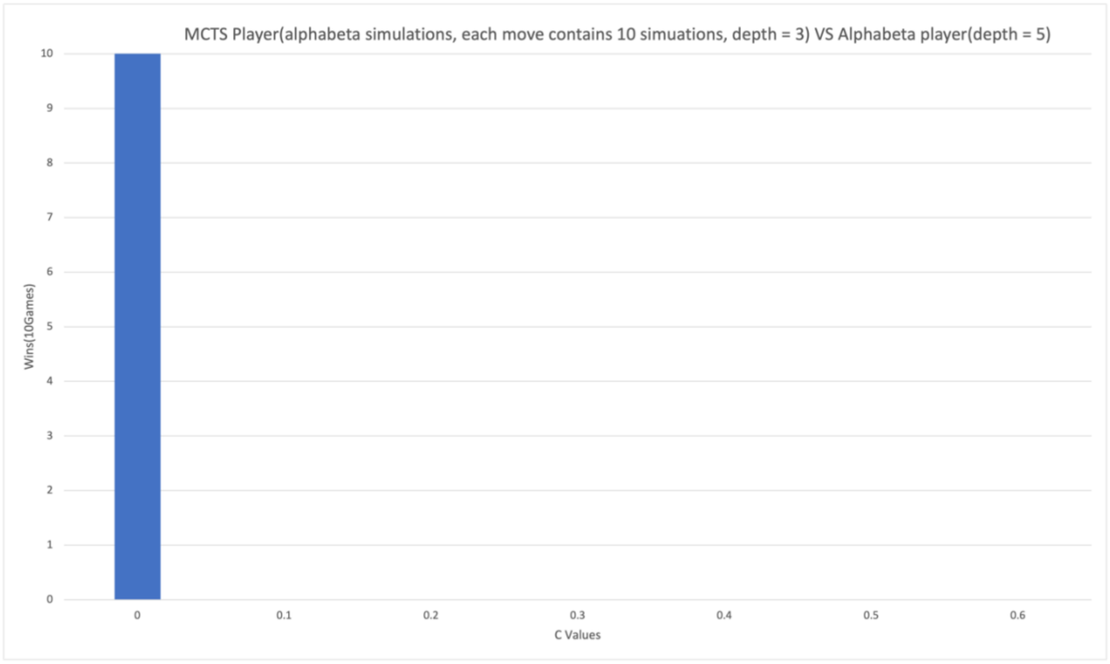
(Figure 4)

Since the win rate significantly decreases when C is greater than 2, we terminated the program. At the same time, since the win rate is highest between 0 and 1 and we don't have enough time, we speculate that the C Value should be between 0 and 1. Next, we will test MCTS Player(Simulation type = alphabeta, simulations = 10, depth = 3) versus Alphabeta Player(Depth = 3). The range of C is 0 to 1, step = 0.1. Each C Value will be tested in 50 games.(Figure 5)

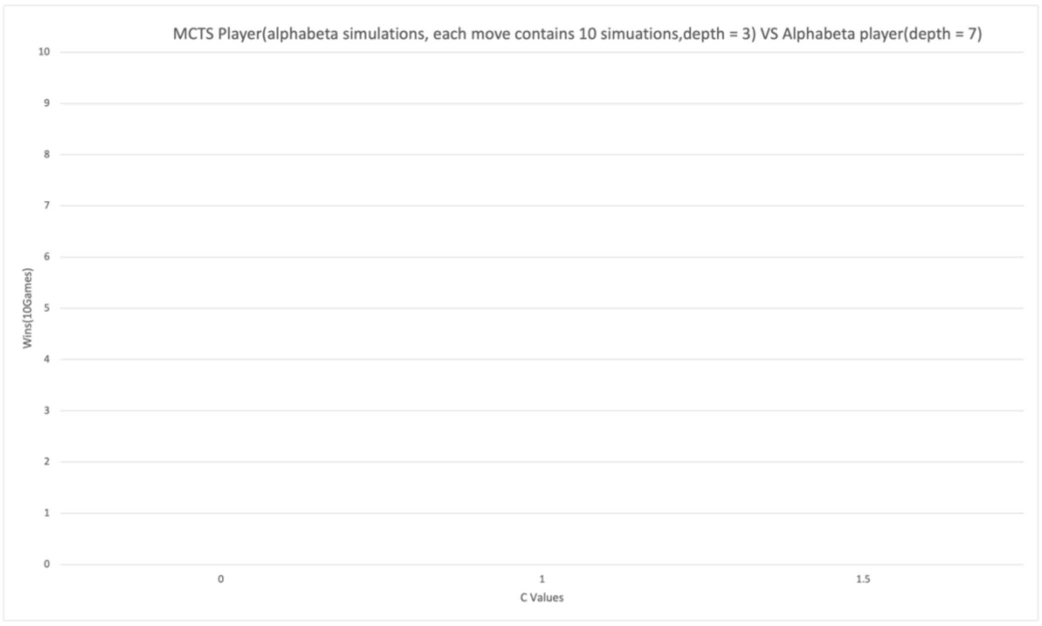


(Figure 5)

As you can see, except for C value = 0 when the winning rate is 50/50, the rest are all 0/50. We terminated the program midway at c = 0.35. At the same time, we tried to change the Depth of Alphabeta Player to find out if this strange phenomenon was caused by the Depth of Alphabeta Player. All other conditions remained the same for the next test, but the depth of Alphabeta Player changed and each c value only will be tested in 10 games(We have no more time). Alphabeta Player with depth 5(figure 6). Alphabets with Depth 7(figure 7).



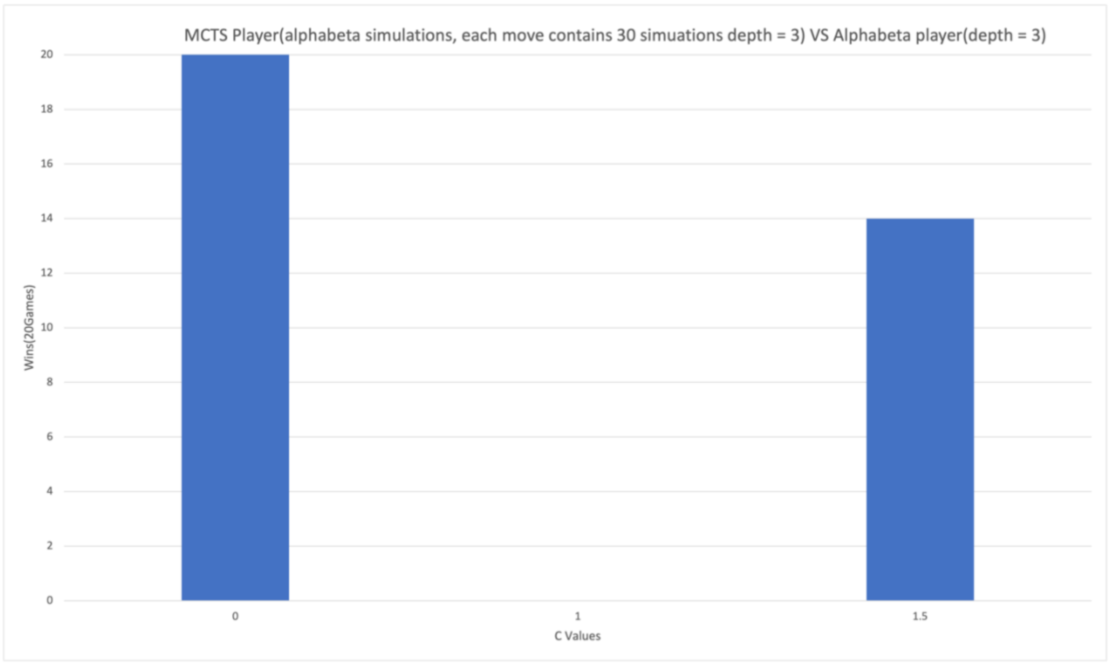
(Figure 6)



(Figure 7)

The situation does not change when the Depth of Alphabeta Player is equal to 5. When the Depth of Alphabeta Player is equal to 7, MCTS Player has not won.

For this phenomenon, we can see from the available test results that the Depth of Alphabeta Player does influence the test results. But this is not enough for us to figure out why the win rate is 0 when the C value of MCTS is not equal to 0. We guess there are two reasons, the first one is that the size of the board is so small that MCTS has few choices for each move which makes the exploration part of MCTS meaningless when MCTS uses Alphabeta Simulation. The second reason is that the number of simulations for each move of MCTS is only 10 which is probably too small. So we did the last test (we did not have time to do other tests, the time cost of each test was too high). MCTS Player(Simulation type = alphabeta, simulations = 30, depth = 3) vs Alphabets Player(depth = 3). The range of c is 0 – 3. Step = 0.5. Each C will be tested in 20 games.(We terminate the test when c = 2)



(Figure 8)

Based on the last test, we found that the win rate of MCTS was 14/20 at C=1.5. We believe that this finding proves that the number of Simulation also has an effect on the results.

In summary, we found that the c-value and number of simulations have a significant effect on the win rate of the MCTS algorithm. Also, , the depth for Alphabeta Player affects the test result. By carefully analyzing these data, we found the optimal c-value for the MCTS algorithm(simulation type = random). Unfortunately, we no longer have enough time to conduct sufficient tests for the MCTS simulation type with Alpha Beta settings. In this case, each test takes a considerable amount of time. However, fortunately, we still have some interesting findings to share. In the future, we can further optimize the simulation process to improve the win rate of the algorithm. For example, each simulation of MCTS is alphabeta vs random.