**COMP4983X2 CAPSTONE PROJECT**

**AI for Gomoku**

**Project Report**

Date:2021/4/7

Author:Taiqi Liu

Supervisor:Dr. Haiyi Zhang

**Content**

Background....................................................................................3

Introduction to Gomoku Rules......................................................4

User’s Interface.............................................................................5

Chess Evaluation Functions...........................................................7

Mini-Max Algorithm......................................................................10

Alpha-Beta Pruning.......................................................................11

A\* Search Algorithm.....................................................................12

Model Testing................................................................................13

Testing with Expert Level Computer Online..................................14

Future Work..................................................................................16

Reference......................................................................................17

1. **Background**

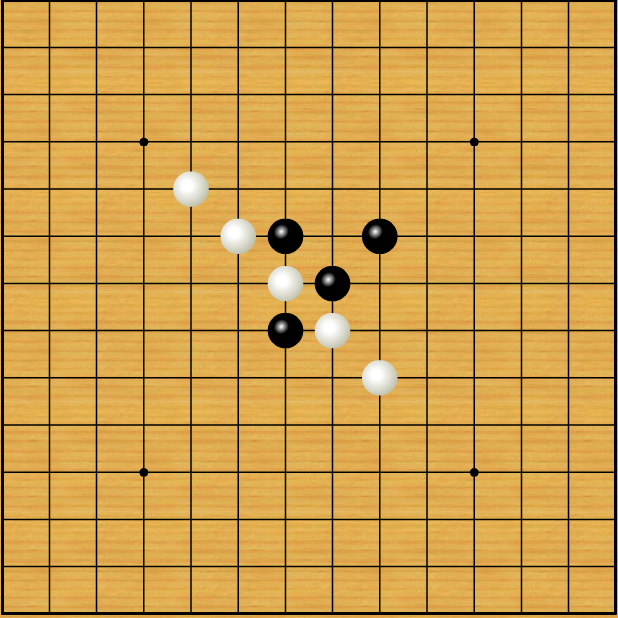
The intelligent Gomoku has the characteristics of a human-machine game and belongs to the category of artificial intelligence. Various methods in the field of artificial intelligence can be used to implement this project.At the same time, due to the simple rules of the Gomoku game, easy to understand and high popularity, the design of the Gomoku artificial intelligence project does not require too high knowledge about Gomoku.Moreover, the project does not require high hardware requirements, does not require additional equipment, and can be completed with a personal computer. Therefore, it is a good choice for individuals to explore the field of artificial intelligence.

In recent years, with the continuous development of artificial intelligence technology and the continuous improvement of computing power, more and more chess game artificial intelligence have been designed and have defeated many world champions of human chess players. But in the Gomoku field, artificial intelligence has never been able to beat the top Gomoku players. This is because the rules of gomoku are more intuitive to humans and are very in line with the way humans think, but they are not friendly to artificial intelligence programs. Because there is no difference in the value of the pieces on the Gomoku board, and there is no significant difference in the value of the position.It is difficult for the artificial intelligence project to accurately evaluate the status on the board and give a score.This makes my project more challenging and innovative.

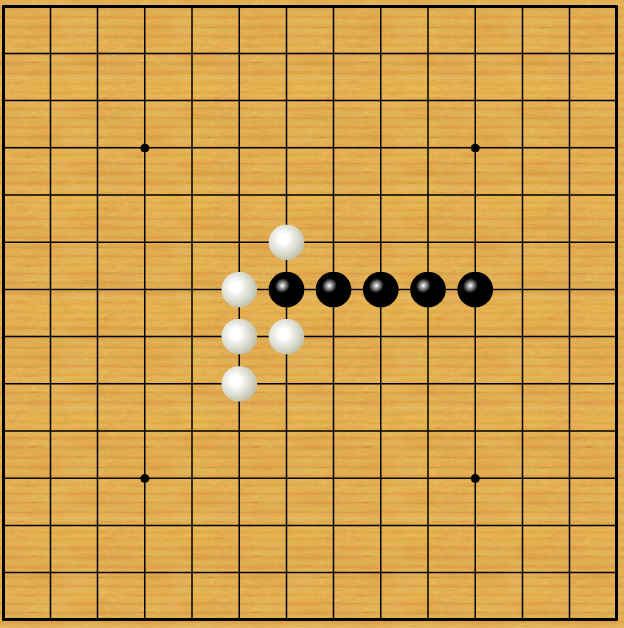
This project focus on artificial intelligence. Use different strategies, algorithms and pruning to gradually improve the search depth and estimation accuracy of the AI.And finally hope that this AI can win the Gomoku battle with other AIs under the same hardware conditions.On the user’s interface, this project is designed using PyGame basic drawing functions.I hope to make the interface as simple and clear as possible, convenient for users to use, and add a lot of user-friendly designs.

1. **Introduction to Gomoku Rules**

Both sides use black and white chess pieces and play on the intersection of the straight and horizontal lines on the chessboard. The side that first connects his or her five pieces into a straight or diagonal line wins. Normally, the player with the black piece start first.



The picture above shows White winning.



The picture above shows Black winning.

1. **User’s Interface**

The user interface is a human-computer interaction platform, so when designing the user interface, I try my best to provide easy operation and clear structure interface, and add some humanized design. It is convenient for the users to use it easy and can focus on thinking about the game.The design of the interface mainly includes the following aspects:

1. Draw chessboard

The chessboard of Gomoku is composed of fifteen horizontal and vertical straight lines, with a total of 225 intersection points for placing chess pieces.Therefore, I use pygame.draw.line() to draw 15 black horizontal lines and 15 black vertical lines.And using pygame.draw.rect() to draw five small black squares on the 5 points in the center to form a standard chessboard.

1. Draw chess

Using pygame.draw.circle() to draw black and white chess.The size of the chess pieces is determined by the size of the chessboard.The position where the chess piece appears is the nearest point where the mouse is located after clicking the mouse.

1. Show last chess piece

Players need to clearly know where the AI took the most recent step in order to know the changes on the board.Therefore, I added a red border to the last chess piece that fell.This can help players judge the situation well and conform to human thinking habits.

1. Number the chess pieces

According to the order in which the pieces were dropped, numbers appear on the pieces. White numbers appear on black pieces, and black numbers appear on white pieces.In this way, the record of a game can be clearly displayed. Allow players to conduct review research after the game is over. It also allows me to better observe and record AI strategies.

1. Display the corresponding position of the mouse

Since the pieces will fall on the intersection, the player may make operational errors. This design will display the corresponding drop point where the mouse is located to reduce player errors

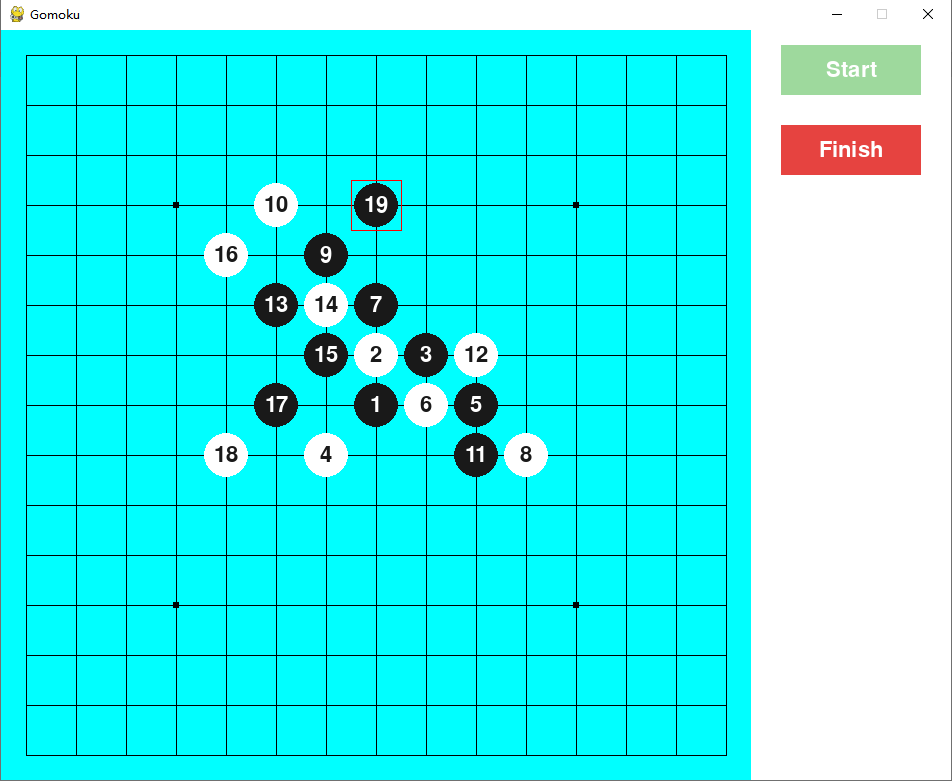
1. Buttons

There are two simple buttons on the right side of the board, allowing players to start and end the game.

1. Text display

In the AI's round, text will be displayed on the right side of the board. After the game is over, the text will also show who is the winner. Doing so can enhance the player's user experience.

The final effect of the user interface of this project is shown in the figure below:



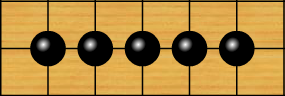
1. **Chess Evaluation Functions**

This part is the most fundamental and critical part of this project. The basic method of doing valuation is to traverse every point on the chessboard from top left to bottom right. For each point, determine whether the horizontal line, vertical line, upper left-lower right diagonal, and upper right-lower left diagonal constitute a scoring pattern. Accumulate all the scores of black and white to judge the situation on the board.Using this method will make each node of the search tree form 225 child nodes. Therefore, simply using this method will make the time complexity of the program to high when the number of search layers is high. In the later part, I will use A\* search instead of brute force search and do pruning.

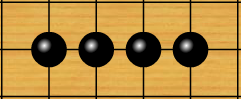
Scoring pattern is the code of conduct that AI uses to choose where to place the chess piece.Modifying the score value corresponding to each pattern will change the behavior style of AI.For example, when the AI's opponent is a human, increasing the score of the advantage on the diagonal line will make the AI perform better. Because people are not sensitive enough to the threats and advantages of the diagonal line, AI is more likely to win by connecting the diagonal line into five pieces.

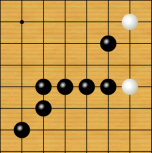
There are some typical patterns in which the AI will gain advantage with the corresponding scores. What the AI has to do is to work hard to form these patterns and prevent players from forming these patterns.

1. Victory pattern(100000 score)



1. Sure to win(10000 score)







The formation of the above several patterns means that one party will surely win in this game, therefore, they are given extremely high scores to avoid other patterns cumulative scores from exceeding their scores.When these patterns occur, the program interrupts the search for pruning.The above-mentioned patterns can appear in different forms, because in this game, horizontal lines, vertical lines and diagonal lines are equivalent.And for the internal line of missing pieces, no matter where the missing pieces are, it is equivalent.

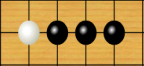
1. 500 Score





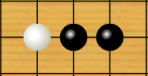


1. 100 Score





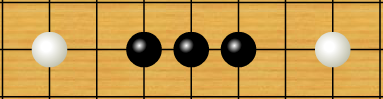
1. 20 Score



The above are some patterns that can bring advantages to a party on the board. They get different scores according to the strength of the advantage. When facing different opponents, the scores of different patterns can be adjusted according to the opponent's weakness to increase AI’s advantage.It should be noted that some patterns that do not look the same are actually the same. The basis of evaluation is to see whether these seemingly different patterns can constitute the same or equivalent greater advantage in the next step.For example, the following three cases are equivalent, because they are the same or equivalent in the greater advantage of the next step.







In addition, it should be noted that when a pattern is judged, the points involved in the pattern need to be marked as temporarily not traversable, so as not to repeatedly increase the score.

The analyseLine() function is for judging that one's own moves can form a chess pattern on a line, mine represents the value of one's own chess, and opponent represents the value of the opponent's chess. It should be judged separately according to the number of adjacent chess pieces that can be connected to the center point. The value of one's chess is set to M, the value of opponent's chess is set to P, and the value of empty points is set to X

The evaluation function can not only be scored by forming the above pattern, but also need to consider the position of the point on the chessboard. The closer to the center point, the better for us. So there is a basic score based on the position of the dropped piece. And the number of other pieces around the place where the piece is placed will also give a basic score. In this way, even if no scoring pattern can be formed, AI can still make a good choice.

In the valuation function, all scores need to be adjusted continuously. The more knowledge and experience the designer has about Gomoku, the more likely to design a more reasonable scoring system.So this is a very interesting part of this project.After completing this part, the computer has a preliminary level of artificial intelligence, but it cannot consider the situation about next few steps like an experienced Gomoku player. Therefore, it is still necessary to use other algorithms to improve AI.

1. **Mini-Max Algorithm**

Max Layer:The round of AI. AI will traverse each point and find the point that can make the valuation function return to the maximum value.

Min Layer:The round of player.AI will also traverse all the points that can be placed. Find the points that can get the highest valuation score for the player. And hope that the score for this point can be as small as possible

The Max layer and the Min layer alternately form a decision tree. According to the return value of the leaf node (the layer with the set depth), AI can determine where the chess piece should be placed in the current round.After a series of pruning and optimization, this program can calculate the result with an average time of 1.19 seconds when the search depth is 4 in my computer.The value of the leaf node is the score given by the evaluation function to the position of the leaf node.Then, assign values to non-leaf nodes through recursion.

The figure below shows the possible route choices made by this AI based on the Mini-max algorithm when the set depth is 2.The value on nodes represent the direct evaluation score for each situation. Each new layer adds far more nodes than this figure. In fact, if there is no pruning, the child nodes of each node will reach 225.

0

50

-50

0

50

1000

25

Depth=0

Depth=1

Depth=2

At present, most of the artificial algorithms in games use Mini-max.

1. **Alpha-Beta Pruning**

As mentioned before, if there is no pruning, when the number of layers reaches 4, the computing capacity of the computer will reach 225^4 \* evaluation fuction(), This may cause the AI to spend several hours thinking about each move, and of course it cannot be tolerated.So this program needs to use Alpha-Beta pruning to delete nodes that do not need to be explored.

α : the current best score of the node from searching it's child node.This is currently the most favorable value for AI.

β : the current α value of the parent node.This is worst current value for the enemy.That is equivalent to the opponent did not move.

The value of α, β is passed through recursion.

In order to make the program more concise, my project has made certain changes to the assignment of Alpha and Beta.The β value of the child node is the -α value of the parent node.The score returned to the parent node is the -α value of the child node.

Pruning condition: (In Negamax style)

When α ≥ β, cut all unsearched child nodes of this node

            if score > alpha:

                bestmove = (x, y)

                alpha = score

                if alpha >= beta:

                    break

Initialization:

Because the root node is the MAX layer, the β value is initialized to positive infinity.

For the α value, if it is the MAX layer, the initial α value is negative infinity, if it is the MIN layer, the initial α value is positive infinity.

1. **A\* Search Algorithm**

Relying on Alpha-Beta pruning to improve AI performance is very limited.In order to get results quickly when the number of search layers is 4,my project added more other methods to assist Alpha-Beta pruning, or pruning from other aspects. These methods are very effective. So that does not need to expanding the nodes one by one from the upper left corner of the board.

1. Estimating the score for each position that can be placed, and sorting these nodes according to the estimated score.Because if we can find a position with a higher score in advance, the efficiency of Alpha-Beta pruning will be greatly improved.The method of estimation is to judge the four straight lines at the point where the chess piece can be placed.The estimated score is given based on the pattern composed of these four lines.At the same time, we will also delete the nodes that are estimated to be ranked behind in advance
2. According to the observation of the Gomoku game record, almost no professional player will place a chess piece on the intersection which no other chess pieces on the surrounding 5\*5 intersection. Therefore, we can elete nodes with no other pieces within the range of 2.This method can greatly reduce the AI consideration time in the opening stage.
3. If there is a "Sure to win" or better pattern in the Estimating, Interrupting the search, because this step has been able to ensure the victory of the game.
4. **Model Testing**

Record each step in the game, the score of the status and the value of Alpha and Beta.

time:0.00 step:0 position:(7, 7), α=0 β=0 queue:0 score:0

time:1.18 step:2 position:(8, 8), α=5172 β=1148 queue:4 score:0

time:2.15 step:4 position:(9, 7), α=10361 β=1527 queue:4 score:2

time:3.72 step:6 position:(8, 6), α=15602 β=2254 queue:5 score:2

time:0.93 step:8 position:(9, 5), α=1862 β=455 queue:8 score:8

time:1.21 step:10 position:(10, 4), α=2055 β=616 queue:12 score:-2

time:1.50 step:12 position:(9, 8), α=2501 β=710 queue:11 score:-390

time:3.01 step:14 position:(7, 8), α=3607 β=1004 queue:14 score:-394

time:0.71 step:16 position:(6, 9), α=638 β=388 queue:23 score:-398

time:0.36 step:18 position:(6, 4), α=324 β=208 queue:25 score:-396

time:0.42 step:20 position:(4, 5), α=314 β=210 queue:27 score:-396

time:0.85 step:22 position:(9, 3), α=669 β=396 queue:27 score:-390

time:0.49 step:24 position:(4, 8), α=434 β=262 queue:33 score:-396

time:0.78 step:26 position:(10, 8), α=648 β=349 queue:38 score:-406

time:0.98 step:28 position:(5, 4), α=808 β=462 queue:38 score:-9000

time:0.90 step:30 position:(5, 9), α=753 β=451 queue:42 score:-9000

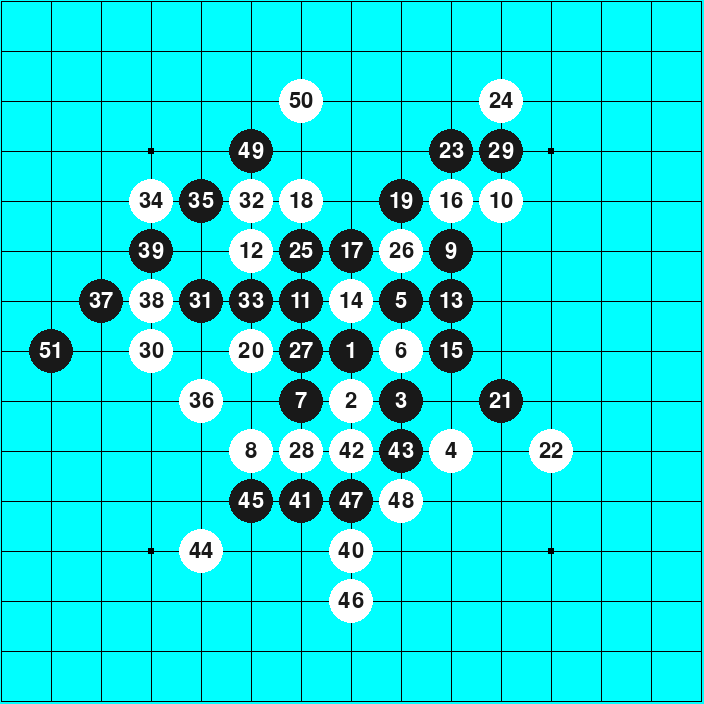
time:0.62 step:32 position:(11, 9), α=154 β=105 queue:51 score:-9030

time:0.35 step:34 position:(3, 5), α=115 β=81 queue:52 score:-100000

time:0.02 step:36 position:(2, 6), α=4 β=4 queue:48 score:-100000

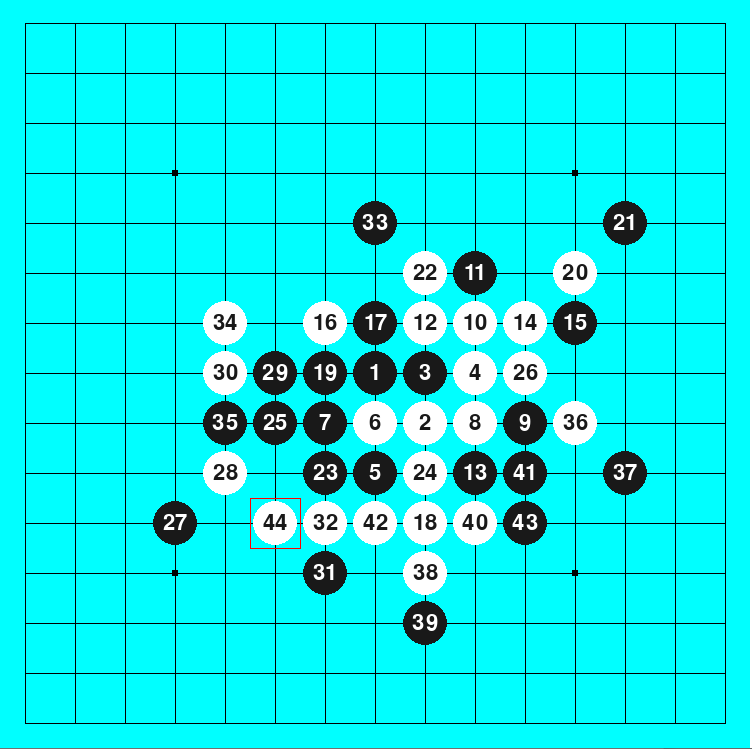
1. **Testing with Expert Level Computer Online**

**Using black to win**



http://gomoku.yjyao.com/

**Using White to win**



http://gomoku.yjyao.com/

1. **Future Work**
2. Further enhance the evaluation function’s ability to judge the situation, this requires me to learn more knowledge of Gomoku.
3. Further pruning from other aspects, so that AI can increase the number of search layers to the sixth layer.
4. Preset the game record in the AI to make the AI perform better at the beginning of the game.
5. Design a more immersive user interface and add background music.

Reference:

Game theory. (n.d.). Retrieved April 13, 2021, from http://gomokuworld.com/gomoku/1

Matthes, K. (2021, January 22). Pathfinding: A\* search algorithm. Retrieved April 13, 2021, from https://karlmatthes.medium.com/pathfinding-a-search-algorithm-d77400c89888

Minimax algorithm in game THEORY: Set 1 (Introduction). (2021, March 31). Retrieved April 13, 2021, from https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

Salamone, L. (2020, May 19). Creating an AI FOR GOMOKU. Retrieved April 13, 2021, from https://medium.com/@LukeASalamone/creating-an-ai-for-gomoku-28a4c84c7a52