Gomoku AI Searching Algorithm

Pengyu Chen, Bao Hu, Jiawen Zhang, Xinyi Li, Angle Zhou, Chenghao Wu

Gabriel Simmons

ECS 170

June 11, 2024

1. Introduction

This project brings the classic board game Gomoku, also known as Five in a Row, into the digital world, featuring a graphical interface and an AI opponent. Gomoku is played on a 15x15 grid where two players alternate placing black or white stones, and the winner is the first player to form five stones in a row either vertically, horizontally, or diagonally. The goal of our project is to provide an intuitive interface for players, develop intelligent AI opponents, and create an engaging experience for casual players and professional players. The motivation behind this project is to combine traditional strategy gaming with modern AI techniques, offering a platform that not only entertains but also shows the capabilities of AI in game development.

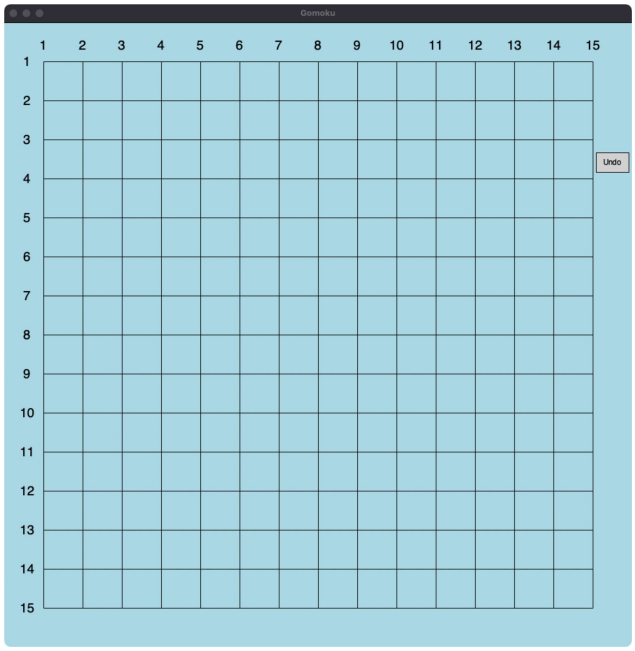
Link to our source code: https://github.com/hubao666/Gomoku-Searching-Algorithm

2. Background

To fully understand our project, you need to be familiar with the following concepts:

Gomoku Rules: The game is played on a 15x15 board, and the objective is to be the first to align five stones in a row. Players take turns placing stones of their respective colors (black or white) on the board.

Graphical User Interface (GUI): Our project leverages Python's graphical libraries to create an interactive and visually appealing game board. The interface allows players to place stones, undo moves, and observe the AI's decisions.



Artificial Intelligence: The AI component of our project is designed to provide a formidable opponent for human players.

Alpha-Beta Pruning: An optimization technique for the Minimax algorithm, Alpha-Beta pruning eliminates the need to evaluate certain branches of the game tree, thereby enhancing the efficiency and performance of the AI.

Heuristics: These are strategies or methods used to speed up the process of finding a good solution. In our project, heuristics help prioritize moves that are more likely to impact the game's outcome, focusing on critical areas of the board.

3. Methodology

Our project employs several AI technologies and methods to create a challenging and responsive AI opponent. Here is an overview of the methodologies and tools used:

Game Configuration:

Move List: Each time a player places a stone on the board, the move is recorded in a list that keeps track of the order and positions of all moves played. This move list is critical for several game functionalities:

1. Player Move List: Two lists tracking humans' and AI's moves and a list tracking all stones on the playboard (human + AI).
2. End-of-Game Checks: By analyzing the move list, the game can efficiently check for win conditions (five stones in a row vertically, horizontally, or diagonally) after each move.
3. AI Algorithms: The AI algorithms use those lists to predict all the possible future states and evaluate the leaf state by using the evaluation function.

AI Algorithms:

Minimax Algorithm: We implemented the Minimax algorithm to enable the AI to make strategic decisions by simulating potential future moves and evaluating their outcomes.

Alpha-Beta Pruning: This technique optimizes the Minimax algorithm by pruning branches that do not need to be explored, reducing the number of nodes evaluated and improving performance.

Heuristics and Optimization in the Code:

has_neighbor Function: This function checks if a given point on the board has neighboring stones within a radius of 1, helping to prioritize moves that are more likely to affect the game outcome.

order Function: This function prioritizes potential moves based on their proximity to the last move made, reducing the search space and focusing on more critical moves.

Data: The AI's decision-making process is guided by the current state of the board, evaluating possible moves based on the placement of stones and their potential to form rows of five.

Graphical User Interface: We used 'graphics.py' library to create the graphical user interface, providing an interactive and visually appealing game board. The interface includes features such as placing stones, undoing moves, and displaying the game's status. The link to the library is in the next section.

Non-AI Software: In addition to AI components, we implemented various software functionalities to enhance the user experience, such as the undo feature. The undo functionality allows players to retract a false move, although it does not allow undoing twice in one turn or undoing and then playing the same move again.

Third-Party Tools:

User interface (graphics.py)： https://mcsp.wartburg.edu/zelle/python/graphics.py

Numpy: Utilized for efficient array manipulations and calculations within the AI algorithms.

By combining these methodologies and tools, our project delivers a sophisticated and enjoyable Gomoku playing experience.

4. Evaluation

Evaluating the performance of the Gomoku AI system involves assessing several key metrics. Here, we will use three performance metrics: win rate, memory usage, and AI move time.

Win Rate

Definition: The win rate measures the percentage of games won by the AI over a relatively large number of games against various opponents.

Method: We can use the Central Limit Theorem (CLT) to estimate the win rate with a certain confidence interval. By playing a significant number of games, we can calculate the sample mean win rate and its standard deviation. (We understand that the more games we play, the more accurate our winning rates will be. However, since each game takes about 10-20 minutes, it will take a lot of time to gather a substantial amount of data. Therefore, it is not feasible for us to obtain such a large data set. Therefore, we chose the number 30 for our sample size. Using 30 games allows us to apply the Central Limit Theorem, which provides a more accurate estimate of our winning rates. This number is a good balance as it is large enough to ensure the data is statistically significant without requiring an excessive amount of time to collect. By using 30 games, we can achieve a reliable analysis without compromising objectivity due to insufficient data.)

Implementation: Run a series of games (e.g., 30 games) and record the number of wins, losses, and draws. Use the sample mean and standard deviation to estimate the population mean and construct confidence intervals.

Evaluation: Winning Rate = 21/30 = 0.70 (70%)

$\widehat{P}$ = winning rate = 0.7, n = 30, α = 0.05, $z_{\alpha/2} = 1.96$,

$$\text{Confidence Interval: } P \in \widehat{P} \pm z_{\alpha/2}\sqrt{\frac{\widehat{P}(1-\widehat{P})}{n}}$$

$$P \in [0.536,\ 0.863]$$

This means that, based on 30 simulated games with the AI, we are 95% sure that the population winning rate is between 53.6% and 86.3%. The win rate reaches the goal of our program.

Memory Usage

Definition: Memory usage measures the amount of memory consumed by the AI during its operation, particularly while calculating moves.

Method: Record the memory usage during game play.

eg. Copied from the terminal while playing the game, following is a random move I made and shows the CPU usage and Memory usage.

(CPU usage: 3.90%, Memory usage: 98.73 MB)

Implementation: Profile the memory usage while running the AI for a series of moves. Record the peak memory usage and average memory usage over a large number of moves.

AI Moving Time

Definition: AI move time measures the time taken by the AI to make a move. This is particularly interesting as the complexity of the board increases.

Method: Measure the time taken for the AI to make each move, especially noting how the time

increases as the distance between human moves and the AI's best move increases.

eg. Following first line is the first move that the AI chose and shows the time it took to calculate. And the second line is the one of the last few moves that the AI chose, which secured the victory, and shows its time. We can see that the difference in time is significant. This is because as more pieces are placed on the board, the algorithm requires more time to calculate the best move. The increasing complexity of the game state demands more computational power and time for the AI to evaluate potential moves and outcomes.

AI chose position: (10, 6) in 0.2058 seconds.

AI chose position: (3, 7) in 20.9517 seconds.

Implementation: Use Python's time module to measure the duration of the AI's move calculations. Record the time for each move and analyze how it varies with the board state complexity.

5. Results

Our project aimed to create an interactive application that enhances the game of Gomoku by integrating an AI opponent, making the game more engaging. We are pleased to report that we have successfully achieved these goals.

AI Opponent:

We initially planned to make 2 versions of the AI algorithm, one easy mode and one expert mode. Then we could use the winning rates of both versions to quantify the AI's performance. However, we eventually decided not to create the easy mode. One reason was that it felt insufficiently challenging and wouldn't have much practical significance. Therefore, we dedicated a significant amount of time to developing and refining the expert algorithm.

Game Modes:

AI vs. Human: Players can choose to either go first or let the AI make the first move, providing flexibility in gameplay

The algorithm of the player going first and AI going first has almost the same algorithm, except one thing, the ratio of human evaluation score.

The AI evaluation score function returns ai_score - human_score * a specific ratio.

While the ratio is low, the AI algorithm will not really care about the human’s move, so it will try its best to win. However, while the ratio is relatively low, the AI algorithm will try his best to block the human player's moves and prevent them from winning will be its primary goal. Therefore, when the ratio is high, it is suitable for the AI to play first, focusing on offense and trying to win. When the ratio is low, it is relatively better for the AI to play second, focusing on defense and blocking the human player's moves.

In theory, the first player in Gomoku has a guaranteed winning strategy. To balance the game, international rules have been established to limit the first player's advantage, making the game

more fair and challenging. These rules are known as "forbidden moves," which dictate that if the first player places a piece in a forbidden position, they lose the game. However, our algorithm does not take forbidden moves into account because most people are not familiar with these specific rules, and including them could easily mislead players. Therefore, without considering forbidden moves, if the first player AI operates without bugs, it is theoretically unbeatable. We assign a high ratio to the first player AI to encourage aggressive play, aiming for a high winning rate. If the winning rate is high, it indicates that our first player AI performs well.

6. Discussion

Significance of Results:

The results of our Gomoku project are significant for several reasons. Firstly, the successful integration of AI into the game has enhanced its complexity and appeal, offering players a challenging opponent that simulates expert-level gameplay. This achievement underscores the potential of AI in strategic games, demonstrating how advanced algorithms can create engaging and competitive experiences for users.

The high win rate of our AI algorithm indicates that our implementation of the Minimax algorithm with Alpha-Beta pruning is effective and robust. This result validates our choice of algorithms and optimization techniques, proving that even with a depth of 2, the AI can make strategic and well-informed decisions.

Lessons Learned:

Throughout the project, we gained valuable insights into both AI development and game design. Here are some key takeaways:

1. **Algorithm Efficiency**: The importance of optimizing algorithms to balance depth and performance was a critical lesson. Our experience showed that deeper searches do not always guarantee better moves, emphasizing the need for effective heuristics and pruning techniques.
2. **User Experience**: Creating an intuitive and visually appealing graphical interface significantly increases user engagement. From the feedback, having a well-designed user interface is very important to make the game more accessible and fun.

Future Improvements:

While our project achieved its main goals, we can add other functions for future improvements:

- **AI (Expert) vs. AI (Expert) Mode**: Implementing a game mode where two expert-level AIs compete against each other would be a valuable addition. This feature would allow players to observe high-level strategies and learn advanced techniques for playing Gomoku.

7. Conclusion

The Gomoku project effectively demonstrates the successful use of artificial intelligence in strategy games. The results reflect the effectiveness of our approach. By continuing to refine the AI and expanding the capabilities of the application, we can provide a more engaging and educational experience for Gomoku enthusiasts.

8. Contribution

Jiawen Zhang：

- Worked on Project Report
- Test on the code

Pengyu Chen:

- Optimization (has_neighbor and order)
- Evaluation function (including computing scores)
- UI (undo button)
- Revised report (Result part)

Bao Hu:

- UI design
- Game logic and configuration
- Min-Max Alpha-Beta pruning algorithm
- Different color of AI's newest placed stone
- Revised report

Chenghao Wu:

- Testing the Algorithm to find possible bugs and spaces for improvements, adjusting some eval values
- Worked on the PPT
- Reviewed the report

Angel Zhou:

- Test and find improvement for the algorithm
- Worked on the report
- Worked on the PPT

Xinyi Li:

- Testing/reviewing the Algorithm
- Worked on the PPT
- Reviewed the report