Monte-Carlo Search and Games
Project: Gomuku

Zhe Huang & Linghao Zeng

April 14, 2024

1 Introduction

Gomoku, also known as Five in a Row, is a classic board game traditionally played on a Go board. The objective is to be the first to line up five consecutive pieces horizontally, vertically, or diagonally in a 15×15 grid, as in Figure 1. This simple yet profound game serves as an excellent platform for exploring various artificial intelligence strategies due to its extensive search space and strategic complexity. Unlike games with limited move sets, Gomoku's open grid and minimal ruleset amplify the potential strategies a player can adopt, presenting unique challenges for AI design [1].

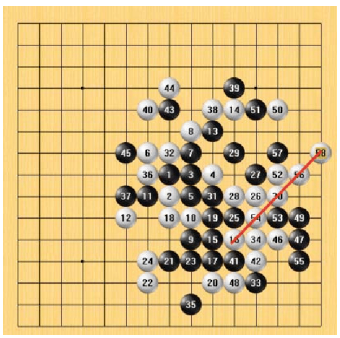

Figure 1: A standard Gomoku game, where the white wins the game at last after 58 moves.

As part of the coursework for "Monte-Carlo Search and Games," we conducted experiments to implement and evaluate a Monte Carlo-based algorithm introduced in the class specifically tailored for Gomoku. These experiments were designed not just to test the effectiveness of Monte Carlo methods in game strategy, but also to refine these techniques to better suit the intricate dynamics of Gomoku. The following sections of this report will detail our experimental setup, present our findings, and discuss the implications of our results. Through this investigation, we aim to illustrate the robust capabilities of Monte Carlo algorithms in navigating the strategic complexities inherent in Gomoku.

2 Experiments

2.1 Experimental Setup

The primary aim of our study was to evaluate the performance of a Monte Carlo-based algorithm implemented for the game of Gomoku. To facilitate this, we designed our experiments around a 9x9 version of the traditional Gomoku board, a smaller scale than the standard 15x15 grid. This decision was made to allow for a more manageable computational demand while maintaining enough complexity to assess strategic decision-making effectively. All experiments were runned on a Apple MacBook Pro with M1 Pro.

2.2 Results

Table 1: Algorithm Performance Comparison

White	**Black (opponent)**	**Win Rate**	**Time (seconds)**
Flat	Random	100%	120.23
Flat	Flat	48%	735.56
UCB	Flat	54%	649.57
UCT	Flat	57%	70.99
RAVE	Flat	54%	71.89
GRAVE	Flat	52%	69.06
Sequential Halving	Flat	82%	654.01
SHUSS	Flat	97%	738.45
Nested Flat (Misere)	Flat	59%	560.71
Nested UCT (Misere)	Flat	48%	69.58

In our experimental framework, Flat Monte Carlo and UCB was tested with 300 playouts per move to explore various potential outcomes. The goal was to identify the most advantageous moves based on the accumulated outcomes from these simulations. We conducted 100 game simulations for each algorithm setup. The results from these games were used to calculate the win rate.

As can be seen from Table 1, the Flat Monte Carlo algorithm, when playing against a Random opponent, achieves a perfect win rate of 100%. Notably, when the Flat algorithm faces itself, the win rate for White is significantly lower at 28%. This decrease is largely due to the first-move advantage inherent to the player with black chess.

The UCB and UCT constants are set at 0.4, as discussed in the class. The UCT runs significantly faster than UCB, largely due to the use of a hashtable, similar to the enhancements seen with RAVE and GRAVE algorithms. Among the strategies tested, Sequential Halving and SHUSS achieved the highest win rates, at 82% and 97% respectively. This success can be attributed to the strategy employed by the white pieces, which is predicated on the assumption that black is using either RAVE or GRAVE strategies. Additionally, we experimented with Nested Monte Carlo and Nested UCT on a misere board, where the scoring is adjusted based on the number of rounds played.

3 Conclusion

This study has demonstrated the effectiveness of Monte Carlo methods in the strategic board game Gomoku. Through the use of different strategies and algorithms, including Flat Monte Carlo, UCB, UCT, RAVE, GRAVE, Sequential Halving, SHUSS, and Nested variations, we were able to evaluate the effectiveness of these techniques in a controlled environment. Key findings reveal that UCT performs faster due to hashtable usage, and strategies like Sequential Halving and SHUSS achieve the highest win rates, indicating their superior ability to navigate complex game dynamics. Furthermore, experiments with Nested Monte Carlo and Nested UCT on a misere board highlight the adaptability of these methods to different game settings. The results affirm the potential of Monte Carlo algorithms in game AI development, suggesting broader applications in AI research. Future work may explore extending these strategies to other strategic games and decision-making scenarios.

References

[1] Kun Shao, Dongbin Zhao, Zhentao Tang, and Yuanheng Zhu. Move prediction in gomoku using deep learning. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 292–297, 2016.