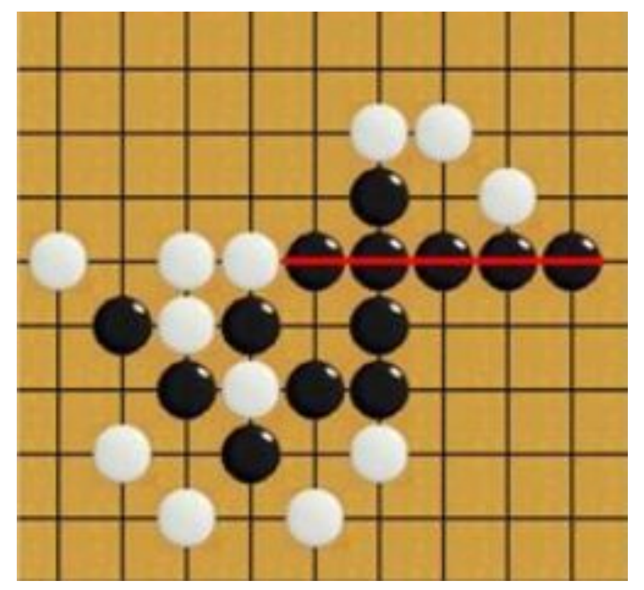
Gomoku Solver

Jiayi Chen and Siyuan Niu

Introduction

Gomoku, or five in a row, is an abstract strategy board game where two players place a stone of their color on a 15x15 empty intersection. The winner is the first whose stones form an unbroken chain of five horizontally, vertically, or diagonally. It is similar to the Tic-Tac-Toe, but it is much more difficult to solve.

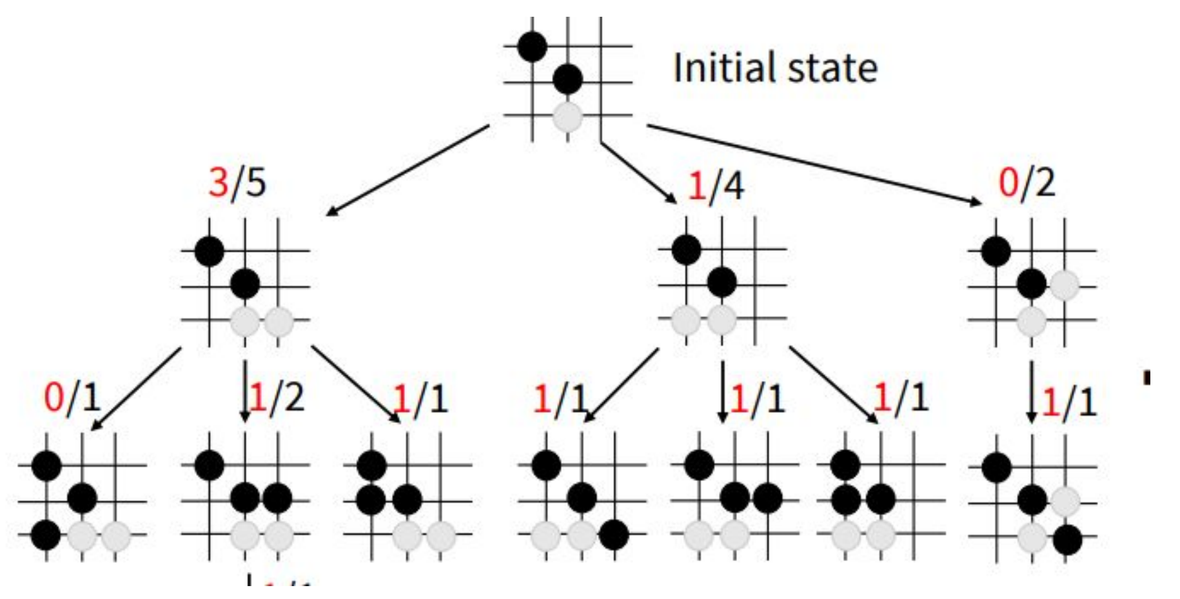


Gomoku Example - Black wins given a row of 5 connected stones https://www.microsoft.com/en-us/p/gomoku/9wzdncrd2qlm?activetab=pivot:overviewtab

Minimax is is widely used in board games involving two-player competition, e.g. Gomoku [1]. However, due to a O(b^d) time complexity, researchers have implemented various methods to improve search performance, such as alpha-beta pruning [2], Monte Carlo Search Trees [3], and various heuristic functions [2]-[4]. In this project, we intend to incorporate alpha-beta pruning and heuristic function into the minimax algorithm. We also explore how limiting relevant moves and different depth limits affect algorithm performance.

Methodology

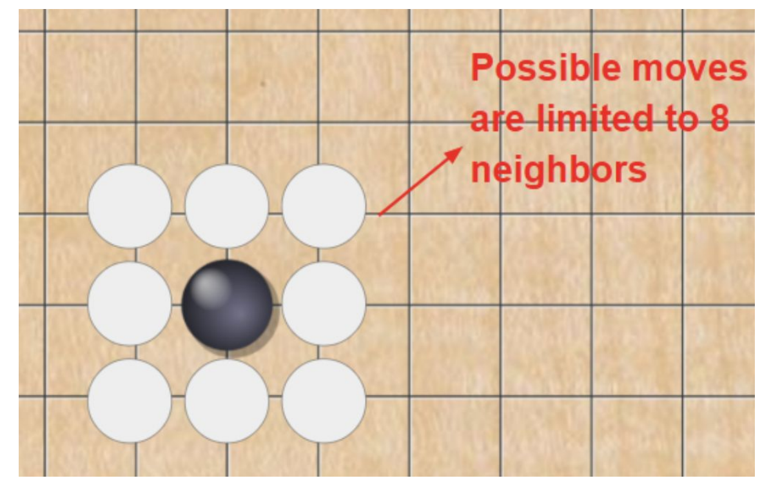
Minimax helps the agent to identity the next optimal move minimize through a recursive tree propagating minimax values back "upwards," given the assumption that the opponent will always play optimally. Meanwhile, we incorporate an evaluation function calculating the optimality of each possible move. To avoid a lengthy exhaustive search, we also set a depth limit and restrict the branching factor by limiting the number of possible moves [5].



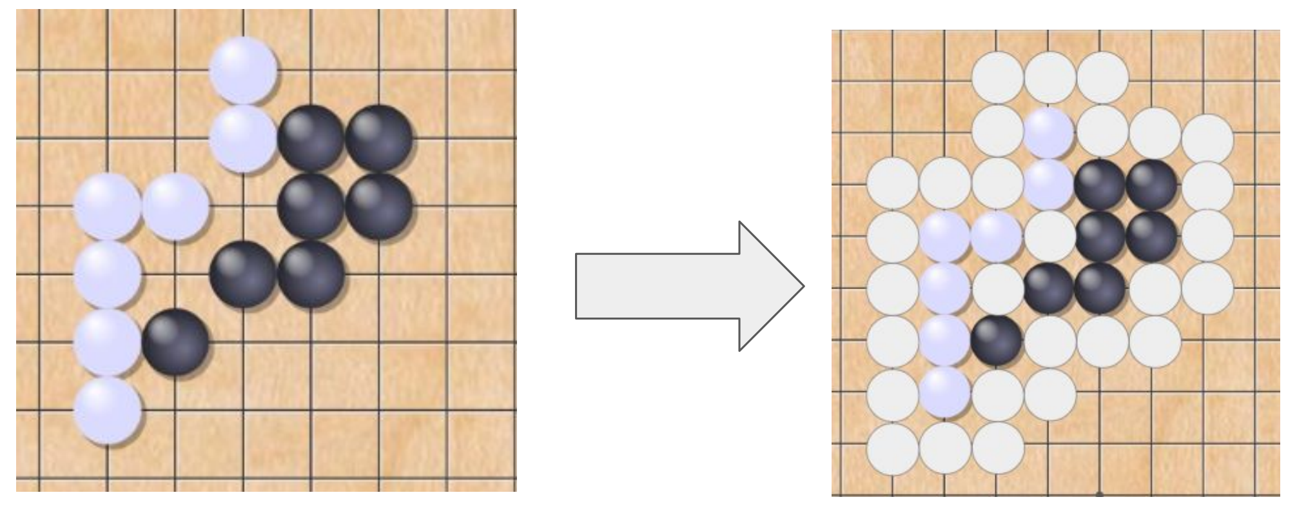
A Minimax Tree

Possible Moves

To implement the algorithm, the first step is to determine all the possible searching steps. To improve the searching efficiency and avoid a large branching factor, we assume the opponent will only place a stone in the adjacent cells of all current stones on the board. The assumption makes sense since the moves are also what most players would take in a real game.



Simple Example



Complex Example

Evaluation Function

We define our evaluation function of each possible move based on the number of threat patterns resulted on the board, i.e. 2, 3, or 4 connected stone. Meanwhile, a pattern can be either "open" or "half" - an open pattern is not blocked by an opponent stone on either side and a half pattern is blocked by one.

Pseudocode

```
def minimax(board: Gomoku, maximizing: bool):
    # base case
    if board.is_terminal() or curr_depth > depth_limit
        return eval(board)

    # recursion
    scores = []
    for board  in board.get_possible_moves():
        scores.append(minimax(board, not maximizing))
    if maximizing:
        return max(scores)
    else:
        return min(scores)
```

Result

In the best scenario, alpha-beta pruning reduces minimax's time complexity to $O(\sqrt{b^\wedge d})$ And our implementation is able to replicate the improvement, as Fig. 1 shows.

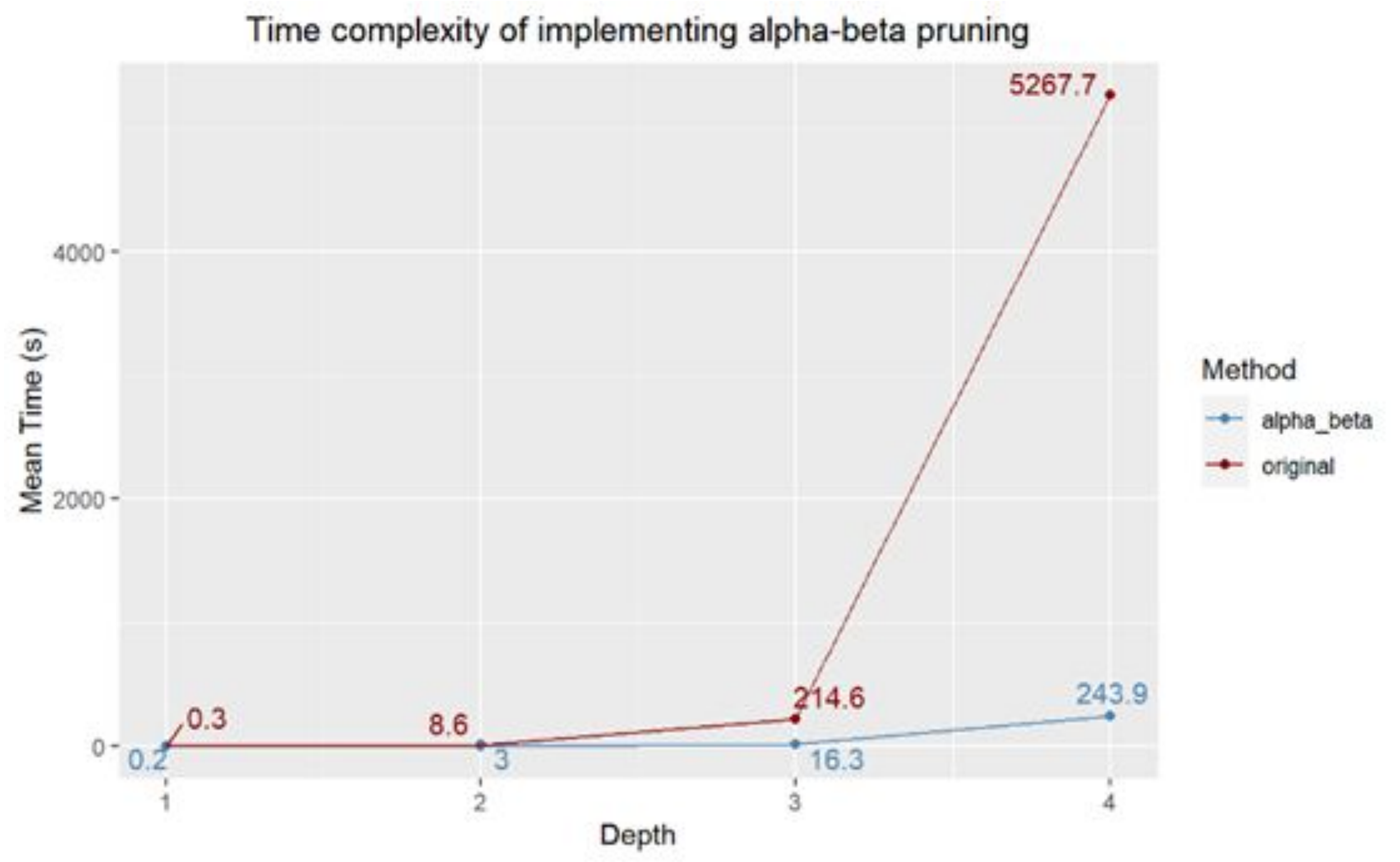
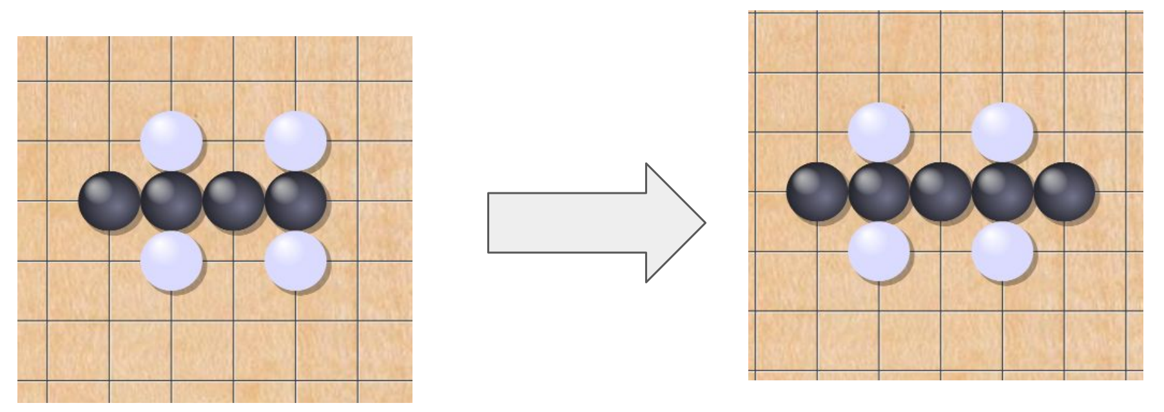


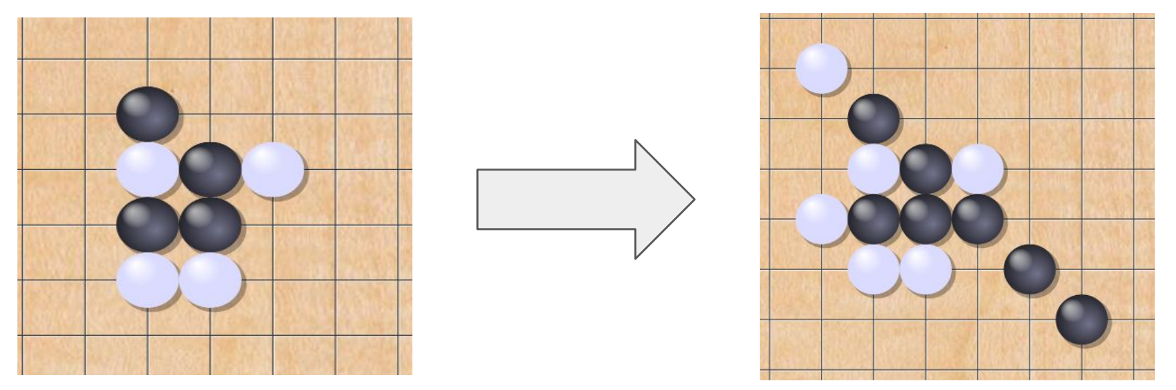
Fig. 1: Alpha-beta pruning reduces minimax's time complexity

We also use boards of Easy, Medium, and Hard difficulty to compare the win rates of the agent (black). Difficulty means the minimum number of moves it takes for black to win. For example, black may need only one correct move to win on an easy board, while five or more moves on a difficult one. Based on the controlled case where both black and white place random moves, we compare black's win rates across boards of different difficulties given a search depth limit of 0 or 1.

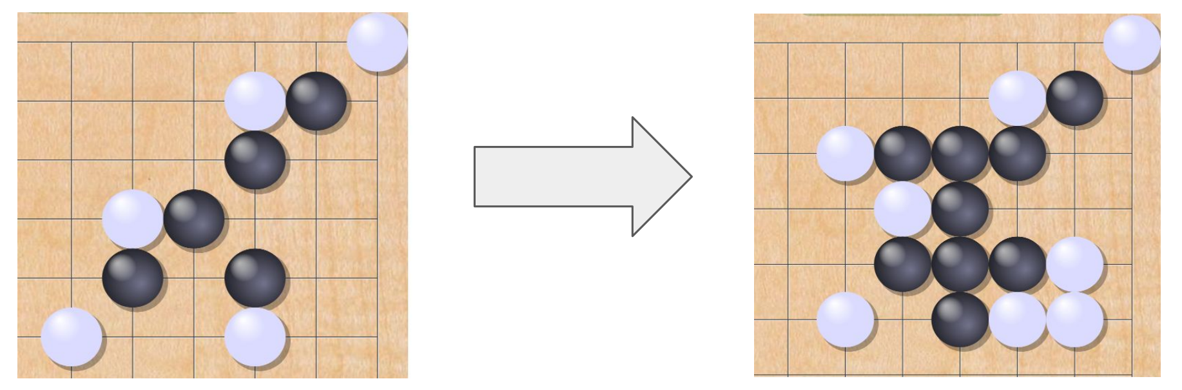
- Sample tests (assuming Black goes first):
  - Easy (takes 1 step to win):



  - Medium (takes 5 steps to win):



  - Hard (takes 7 steps to win):



Example Boards - These boards are drawn from https://www.mathsisfun.com/games/gomoku.html

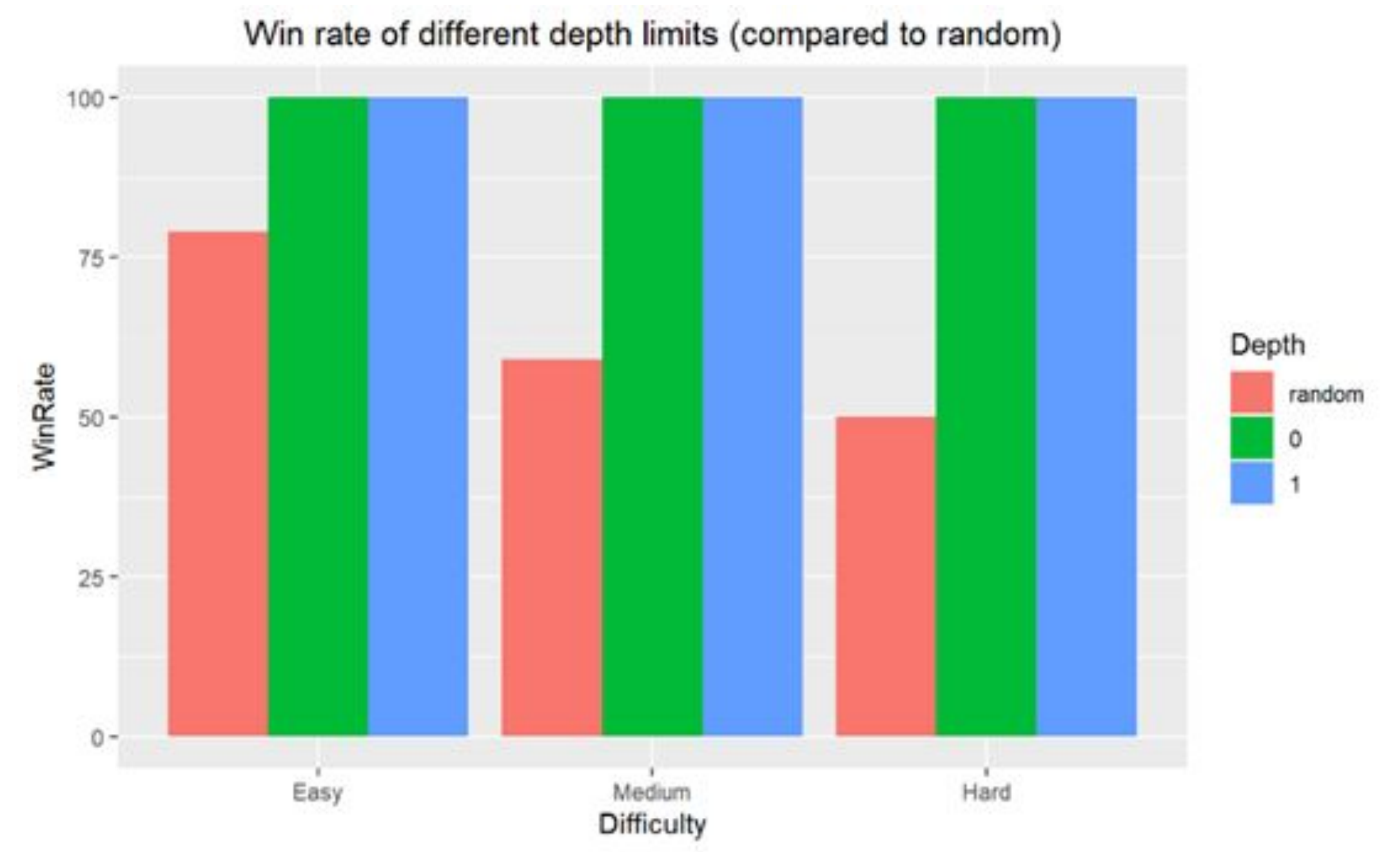


Fig. 2: As board difficulty increases, the win rate of black placing random moves drops from 76% to 50%, while black wins 100% against white placing random moves regardless of any depth limits.

References

[1] Y. PIRILDAK, "Mastering Tic-Tac-Toe with Minimax Algorithm," Medium, May 13, 2020. https://levelup.gitconnected.com/mastering-tic-tac-toe-with-minimax-algorithm-3394d65fa88f (accessed Dec. 10, 2022).

[2] E. Nygren, "Design Specifications for an Interactive Teaching Tool for Game AI using Gomoku".

[3] Yu. (2019). AI Agent for Playing Gomoku. Retrieved December 9, 2022. https://stanford-cs221.github.io/autumn2019-extra/posters/14.pdf

[4] H. Liao, "New heuristic algorithm to improve the Minimax for Gomoku artificial intelligence".

[5] "Minimax Improvements." https://blog.theofekfoundation.org/artificial-intelligence/2015/12/18/minimax-improvements/ (accessed Dec. 17, 2022).