Introduction to Artificial Intelligence
Project Report

Adithya Neelakantan (SUID : 663182945)
Arya Pathrikar (SUID : 883625946)
Raj Nandini (SUID : 949041747)
Shruti Vasave (SUID : 427009403)
Siddhesh Save (SUID : 334464309)

Methodology

Our Gomoku algorithm employs a heuristic evaluation strategy to make strategic moves within the game. The base of its decision-making is the `*AdvancedGomokuAI*` class, which evaluates the entire game board from the side of a given player. The `*evaluate_board*` method runs a loop through each position, invoking the `*evaluate_position*` to assess the potential of individual stones. This potential is determined by taking into account four movement directions—horizontal, vertical, diagonal, and the anti-diagonal—using the `*evaluate_direction*` method. The cumulative score from all directions reflects the overall potential of a specific position. The `*calculate_potential*` method assigns scores based on the line score and open ends. Winning moves or imminent threats/opportunities receive high scores, with like specific values for scenarios like double-open fours, single-open fours, and central control enhancement. This approach allows the AI to evaluate the strategic importance of each position on the board. Our strategy is based on immediate board evaluation.

In the `*Submission*` class, the `*__call__*` method orchestrates the AI's move selection process. The `*get_prioritized_moves*` function orders moves by considering central focus and proximity to existing stones. The prioritized list is then iterated through as the AI evaluates each move. By temporarily placing a stone and calculating the score difference between the current player and the opponent, the AI selects the move with the highest overall score difference as the best move. The AI deviates from traditional minimax algorithms and alpha-beta pruning. Instead, it relies on a custom evaluation function, rooted in heuristics. The custom evaluation function considers immediate threats/opportunities and potential future scenarios. This approach significantly enhances the AI's decision-making efficiency and strategic depth. The formula for the evaluation function is a nuanced combination of scores for various conditions, with emphasis given to central control, potential setups for future moves, and imminent threats.

Position Evaluation : PositionScore (cell) = f (cell characteristics, neighboring cells)

Directional Evaluation : LineScore (direction) = Count of consecutive player pieces

OpenEnds (direction) = Count of open ends

PotentialScore (cell) = g (LineScore, OpenEnds)

Overall Board Evaluation : TotalScore =∑ All player – Occupied cells PotentialScore (cell)

The algorithm integrates heuristics, prioritization, and strategic evaluation and by focusing on potential moves through a custom evaluation function, the AI balances immediate threats with future opportunities.

Discussion

The AI implementation mechanism works best in cases where quick decisions are to be made – offering a direct, computationally effective way to evaluate the board without like delving into complex future state explorations. On the other hand, the lack of a deeper, future-oriented strategy, like in those found in algorithms like Minimax or MCTS algorithms, might limit its performance against advanced players or sophisticated AI opponents. Also, our AI's performance might be constrained by its deterministic nature, in that it evaluates the board based on a set of static rules without learning from past games or adapting to the opponent's game playing style. This could lead to often more predictable behaviors, which can be exploited by experienced players/users.

To enhance its performance, integrating a depth-limited search algorithm like Minimax with Alpha-Beta pruning could be a big push forward. This would allow the AI to not only consider those immediate gains but also to like strategize several moves ahead, making it more challenging and less predictable. Evaluations at every move is a great way to achieve the results required, but in the long run, we can add little boosts like incorporating machine learning, specially reinforcement learning – they could provide adaptive capabilities, allowing the AI to learn from past games and develop new strategies dynamically. This approach would require a significant investment in data collection and computational resources but could significantly enhance the AI's gameplay, making it more versatile and competitive.

References

[1] Nygren, E. (2022). "*Design Specifications for an Interactive Teaching Tool for Game AI using Gomoku"* .

[2] Kuan Liang Tan, C. H. Tan, K. C. Tan and A. Tay, "*Adaptive game AI for Gomoku,*" 2009 4th International Conference on Autonomous Robots and Agents, Wellington, New Zealand, 2009, pp. 507-512, doi: 10.1109/ICARA.2000.4804026.

[3] Junru Wang and Lan Huang, "*Evolving Gomoku solver by genetic algorithm*" 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), Ottawa, ON, Canada, 2014, pp. 1064-1067, doi: 10.1109/WARTIA.2014.6976460.

[4] Kuan Liang Tan, C. H. Tan, K. C. Tan and A. Tay, "*Adaptive game AI for Gomoku*" 2009 4th International Conference on Autonomous Robots and Agents, Wellington, New Zealand, 2009, pp. 507-512, doi: 10.1109/ICARA.2000.4804026.

[5] Öberg, V. (2015). *"EVOLUTIONARY AI IN BOARD GAMES : An evaluation of the performance of an evolutionary algorithm in two perfect information board games with low branching factor".*

[6] S. Mohandas and M. A. Nizar, "*A.I for Games with High Branching Factor*" 2018 International CET Conference on Control, Communication, and Computing (IC4), Thiruvananthapuram, India, 2018, pp. 372-376, doi: 10.1109/CETIC4.2018.8531047.

[7] Allis, Louis Victor, Hendrik Jacob Herik, and Matty PH Huntjens. "*Gomoku and threat-space search.*" Maastricht, The Netherlands: University of Limburg, Department of Computer Science, 1993.