Gomoku AI Player

Report prepared by: Abdelrahman Hatem

Project Idea

The project revolves around creating an intelligent Gomoku player using the minimax algorithm, alpha-beta pruning, and heuristic functions.

Functionalities

- Playing the Gomoku game with 2 human players
- Playing the Gomoku game with an AI opponent as black
- A graphical user interface

Similar Applications on the Market

- ByteDance AI Gomoku - The Ultimate Five-in-a-Row AI
- Google's AlphaGo (later on AlphaGo Zero)
- Various applications on Github

Literature Review

Literature about Intelligent Gomuku payers using minimax algorithms was limited with only 2 papers being deemed relevant for our purposes:

- Gomoku: analysis of the game and of the player Wine by Lorenzo Piazzo, Michele Scarpiniti and Enzo Baccarelli

Which provided a complete description of the game and possible algorithms that can be employed to make intelligent players for the game

- New heuristic algorithm to improve the Minimax for Gomoku artificial intelligence by Han Liao

Which provided a description of a heuristic to be used in evaluating board patterns which laid the ground work for our own heuristic functions.

Along with the mentioned papers several Wikipedia articles were also deemed relevant and were vital to the development process:

- Alpha—beta pruning - Wikipedia
- Gomoku -Wikipedia
- Minimax – Wikipedia

Development Platform and Approach

The project was developed on Visual Studio Code with Python 3.10. Several libraries were employed in the project including numpy, matplotlib, and math.

The development approach was simple. First we started by implementing the two player game with two human players, then added functions to play against an AI player.

The algorithms employed within our implementation include the minimax algorithm with alpha-beta pruning (the main algorithm used for the AI player), and our own heuristic evaluation functions, with one relating to the evaluation of the board, and the other relating to the subset of moves considered relevant for the AI player to consider.

Results

The AI player gives the user 3 difficulties defined by the kind of algorithms the AI will use to make its move:

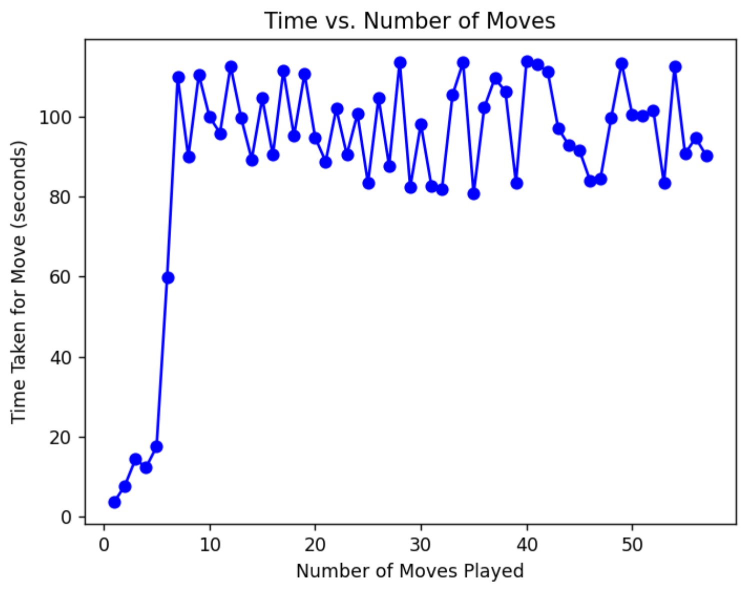
- Minimax (easy)
- Alpha Beta with Evaluation Heuristic (medium)
- Alpha Beta with Evaluation Heuristic and Relevant Moves Heuristic (hard)

The algorithms not only differ in terms of the skill of the AI player, but also in terms of efficiency. As the difficulty increases so does the efficiency, as the heuristics employed help reduce the number of possibilities the AI player has to consider significantly.

The difference in efficiency can be highlighted by looking at the different average times for the AI player to calculate its best move at the different difficulties.

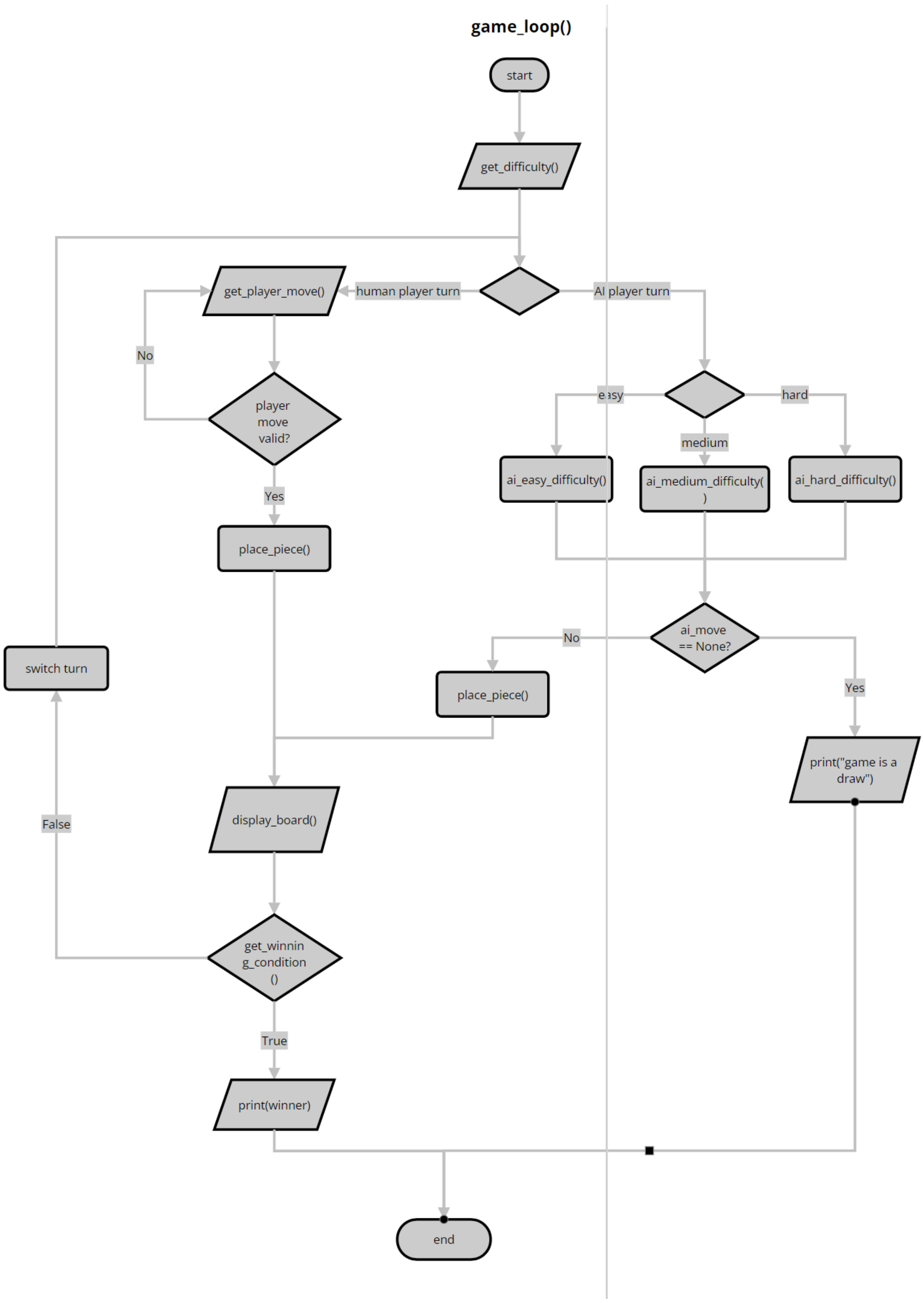
- Minimax (easy) average time: 988.4 seconds ≈ 17 minutes
- Alpha Beta with Evaluation Heuristic (medium) average time: 541.1 seconds ≈ 9 minutes
- Alpha Beta with Evaluation Heuristic and Relevant Moves Heuristic (hard) average time: 90.4 seconds

For the "Alpha Beta with Evaluation Heuristic and Relevant Moves Heuristic" player an analysis of the time taken against the number of moves played was done

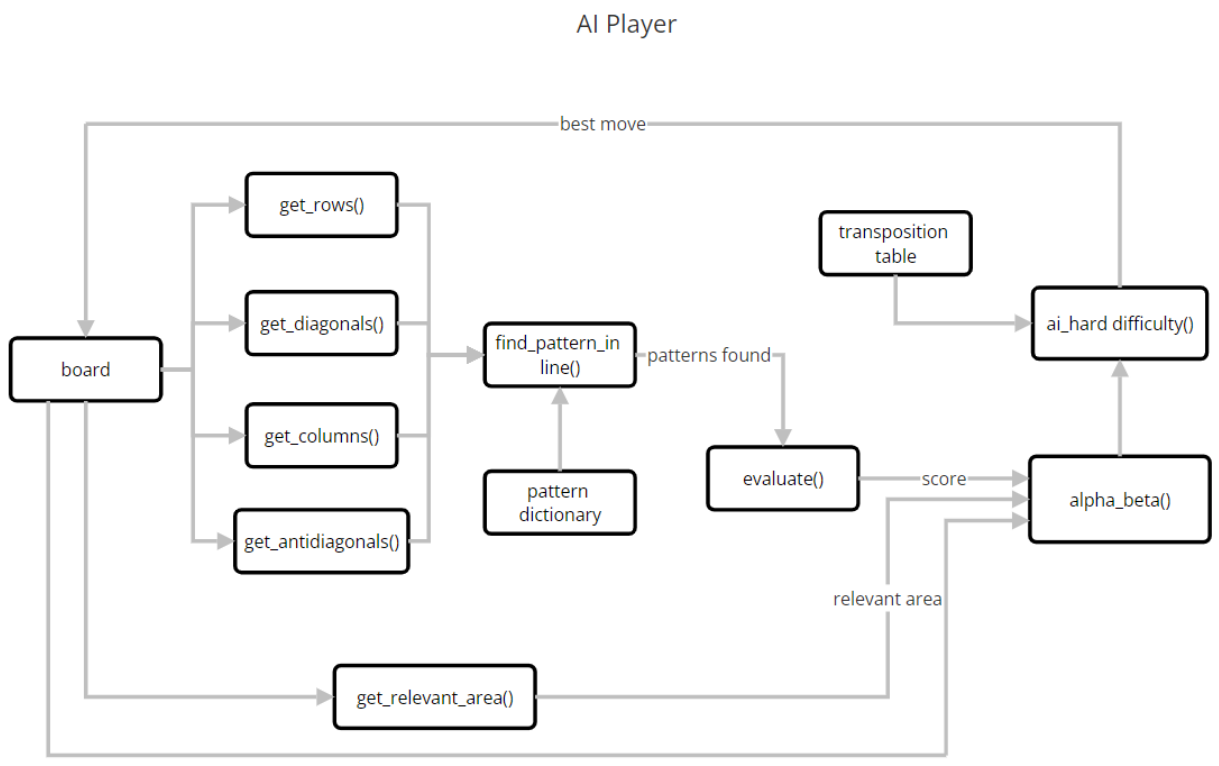


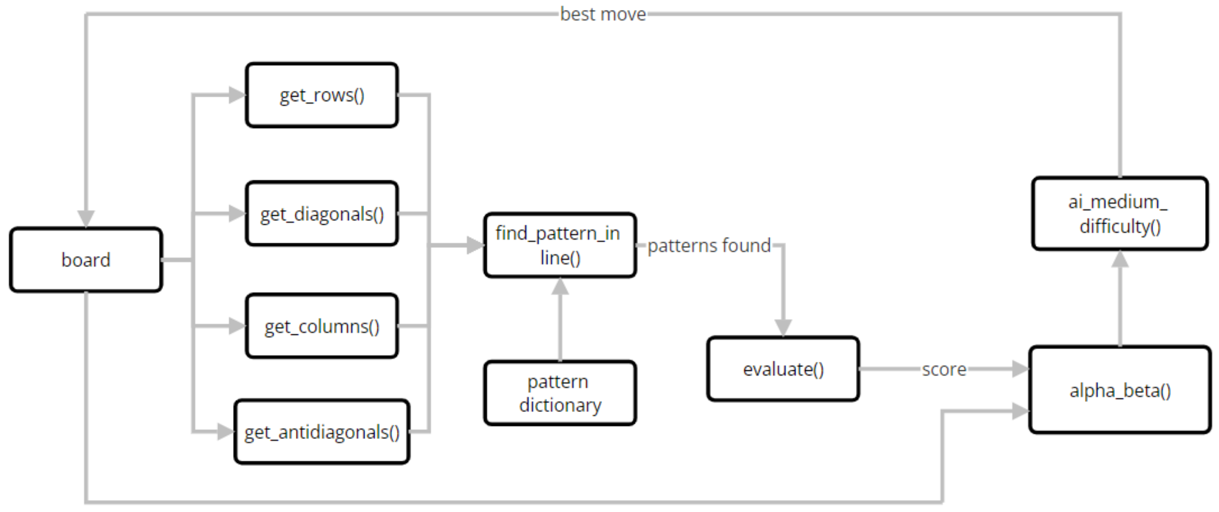
Diagrams

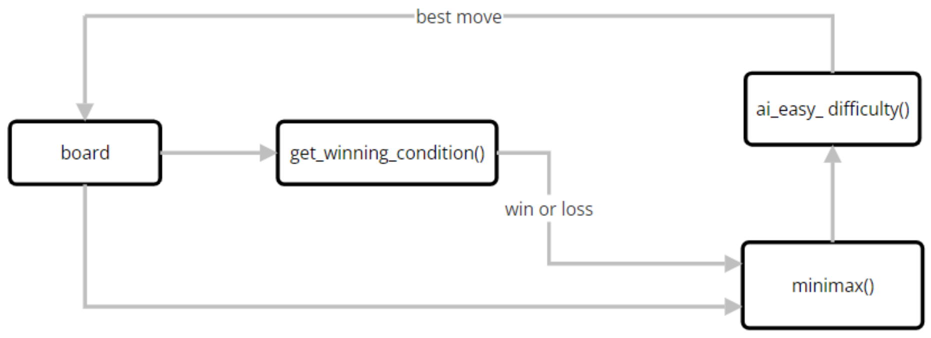
Flowchart for game logic



Block diagram for AI player


AI Player
best move
get_rows()
get_diagonals()
get_columns()
get_antidiagonals()
board
find_pattern_in line()
pattern dictionary
patterns found
evaluate()
score
transposition table
ai_hard difficulty()
alpha_beta()
relevant area
get_relevant_area()


best move
get_rows()
get_diagonals()
get_columns()
get_antidiagonals()
board
find_pattern_in line()
pattern dictionary
patterns found
evaluate()
score
ai_medium_ difficulty()
alpha_beta()


best move
board
get_winning_condition()
win or loss
ai_easy_ difficulty()
minimax()