GOMOKU & Minimax-alphabeta search

Mykola Shevchenko 130708081

***Abstract*— Gomoku is an abstract strategy board game, also called Omok or Five in a Row. This paper explains the implementation of an AI with minimax alpha-beta search with a sequences pattern recognition.**

I. INTRODUCTION

Gomoku is usually played on a board of 15x15 consists in position black and white stones on a board.[1] The player with five consecutive stones of the same color wins the game. The minimax algorithm focuses on minimizing the possible loss for a worst case (maximum loss) scenario. The recursive implementation allows the algorithm to be considerably fast with reasonable winning moves.

II. IMPLEMENTATION

A. Strategy

The strategy is hidden in the evaluation function, which is responsible for determining how good or bad a move is. This is achievable by constructing a tree with a specified depth and to give each position of stones on the board a score that would represent the gravity of future probable situation. In fact, it is important to understand that the minimax algorithm always implied the opponent will make the best possible move. Therefore the final outcome will depend only on the sensible weight specified to different combinations.

B. Minimax implementation

This algorithm allows the evaluation of a position and decides how efficient it would be for a player to reach that position.[2] However, it is not the most optimal solution because it might get lost in evaluating poor scenarios while there would be some obvious winning once. In fact, whenever it is applied to games like Gomoku with board 8x8 (=64 cells with three possible states: white, black or null), the time to compute all the possible combinations is unacceptable.

$$64^3 = 262144$$

The algorithm minimax will have to analyze all 262144 scenarios. To make the algorithm more efficient, alpha-beta pruning is integrated into the method. It seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It consists of passing two key values to determine if a branch is worth being analysed.

C. Alpha-beta integration

The min. (alpha) and max. values (beta) are recursively passed into the arguments of the function and get assigned the best score so far. The cutoff happens whenever we want to skip a tree as we know the move we already did is good enough. This allows our program to think quicker and still be reasonable. Whenever a specified depth is reached, it stops creating new moves for both players. This is where evalution of the board happens.

D. Evaluation function

The evaluation function is what analyzes the best moves and give each board a score. The depth in the search determines the amount of moves the algorithm will predict. In the current scenario the depth is four which means two plies for each player will be on the board.

E. Pattern Recognition

The implementation in this paper is based on translation of rows, coloumns and diagonals into sequences of string. After every translation, each sequence is analyzed on multiple values and possible strategic combinations. The biggest score is assigned to a five in a row "GGGGG" (e.g = 1000). The least dangerous combinations are given a lower score. Therefore combinations

$$x4 = 100, x3 = 10, x2 = 1.$$

In order to achieve better performance during evaluation it is crucial to increase the possible combinations. In fact, the second most strategic possible combos are the empty spaces around streaks. In fact whenever on the board there is a space, it is translated in the straing as an underscore. This allows the evaluation to search for patterns as GG_GG or G_GGG where the underscore is currently the space in the middle of the stones translated as G.

III. CONCLUSIONS

The current algorithm is able to block a casual user with a common intention of creating four in a row. These possible moves are predicted and blocked, while at the same time the algorithm considers the best strategic moves. The current depth is 4 and it is relateively fast (less than 0.5 seconds) and responsive.

ACKNOWLEDGMENT

I would like to thank my companion Henrik Skogmo for intense coding and exciting Waterloo battles.

REFERENCES

[1] YourTurnMyTurn.com team, Introduction and object of the board game, Go-moku rules, accessed 3rd March 2016, http://www.yourturnmyturn.com/rules/go-moku.php

[2] ME 575 - Optimization, Minimax and Max-imin Optimization, accessed 3rd March 2016, ¡ http://apmonitor.com/me575/index.php/Main/MiniMax¿