ECS629 ARTIFICIAL INTELLIGENCE GOMOKU GAME PLAYER

EFTHYMIOS CHATZIATHANASIADIS
150359131

1. INTRODUCTION

The goal of this project is to implement an AI player for Gomoku following a tree search approach with alpha beta pruning. Main focus was targeted on the development of an admissible heuristic function which was mainly inspired by human game playing strategy.

2. MINIMAX

Minimax algorithm tackles the problem of choosing the optimal move by simulating the game and by building a tree of all possible moves. However, the tree has a branching factor of up to 64 (8 x 8) which leads to a huge search space.

2.1 OPTIMIZATION

To speed up the search it was vital to adopt optimization techniques, such as alpha beta pruning with depth limited search and a suitable heuristic. In addition, programming strategies such as implementing recursive minimax with minimum object support were utilized to allow search at depth 5.

2.2 EVALUATION FUNCTION

The main basis of the evaluation function is counting the number of winning sequences.
Definition: A valid winning sequence is a sequence containing 5 consecutive cells in any direction (i.e. horizontally, vertically, diagonally) which belong only to the player and not the opponent.
Winning sequences show the potential of cells such that making moves on those cells could lead to victory.

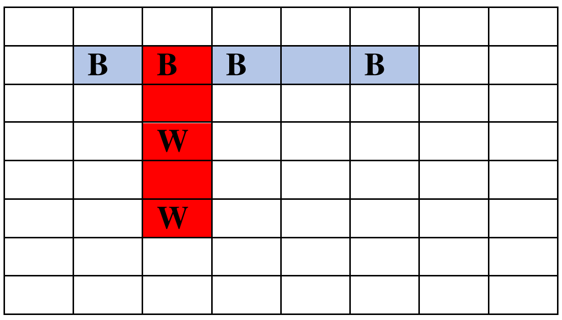

Figure 1: Example of winning (blue), not winning(red) sequence for player Black.

Winning sequences are differentiated based on the number of player cells the sequence contains. The player has higher chance of victory in cases where there are many winning sequences with high number of player cells.
Thus, for each state two vectors are created to measure player attack and defense level:

- ***Vattack=[F1,F2,F3,F4]*** where player is MAX
- ***Vdefense=[-F1,-F2,-F3,-F4]*** where player is MIN

Each vector contains the following features:

- ***F1***: # of winning sequences containing 1 player cell
- ***F2***: # of winning sequences containing 2 player cells
- ***F3***: # of winning sequences containing 3 player cells
- ***F4***: # of winning sequences containing 4 player cells

Therefore, a state can be evaluated as the weighted sum(weights are set to prioritize victory) of the attack and defense vectors:

Vattack + Vdefense =
w1*(F1attack-F1defense)+
w2*(F2attack-F2defense)+
w3*(F3attack-F3defense)+
w4*(F4attack-F4defense)

To count all possible winning sequences an algorithm was developed such that for each cell in the board and for each direction (i.e. horizontal, vertical, diagonal) a sequence of 5 cells is captured. This sequence is then tested to see if it's a valid winning sequence (i.e. contains only player cells). If it's a valid sequence then it is characterized by the number of player cells it contains. For example in Figure 1 the blue sequence is a valid winning sequence of 4 for player Black.
Thus the evaluation function for a given state can be defined as follows:

EVAL(S) =

- +infinity if player wins
- --infinity if player loses
- ***Vattack + Vdefense*** in other cases