Report for Machine Problem 2

Name: Jianghua Wang, Perm: 7310618, Email: jianghuawang@ucsb.edu

Architecture

My code in gobang.py contains one class and one main function for executing the gobang game. The class contains several functions including the constructor for initializing some important properties and setting up the board, threatening detect function for deciding whether the opponent causes some threats that I have to block, heuristic function for deciding the searching order of empty cells, min-max function for choosing the best move for the AI in this round, evaluation function for evaluate the current board and giving a score, AI turn function for calling min-max and making the decision, and some function for checking a move, making a move.

Search

The search algorithm I use is the min-max function with alpha-beta pruning and depth = 2. When the board size is less than 24*24, the AI will have the min-max function search through all the empty cells in the order heuristic function provided and find the best move. What the heuristic function does is that it assumes the empty cell is taken by the current turn player and find whether it can form open three, capped three, gapped three, open four, capped four, gapped four, died four, five consecutive with the previous moves. The function gives a score for every empty cell based on how it can lead the player to win. When the depth = 0, the evaluation function will evaluate the whole board for each player, checking how many open three, capped three, gapped three, open four, capped four, gapped four, died four, five consecutive, open two the board has, and assign a score to each player. The final score of a board will be calculated by subtracting the opponent's score from the AI's score. Then, the min-max function will find me the best move for the current turn. When the board size is larger or equal to 24, my AI will change the strategy. It will call the threatening detect function to find if there is any threatening state caused by the opponent. If there is, then the AI will block the threatening state. If not, the AI will simply call searching space function to give a heuristic score to every empty cell, and directly return the one with the largest score.

This strategy will make every decision in l seconds and will allow me to drag the game out until the opponent times out.

Challenge

My initial evaluation function is simply evaluating AI's score and it can only find the best move for itself, but it does not know when to block the opponents. Therefore, I write the function threat detect to detect threat states made by the opponents. However, it turns out it will still miss some threatening states that I do not specify. I read many articles online, and I finally decide to calculate the evaluation score by subtracting the AI's score from the human's score. This way, the AI will automatically block the human, when it finds that it will benefit more than creating its own consecutive.

Weaknesses

First, I did not make many optimizations, and my program will make every decision in more than 5 seconds when the board size is larger than 15. Therefore, my program may lose the game due to the limited seconds. Second, my program cannot detect patterns that have 2 gaps, and if other's programs can create such situations, then my program may not block such threaten states.