CSE 415 Baroque Chess Agents

Team member name: Yuhan Gao, Jinlin Xiang

Intended personality of the agent: Different answer for different situation:

Case1:My turn, I choose my move here", "I'll think harder in some future game. Here's my move", "I think it will be a interesting move",

Case 2: It is easy for me to do that", "I think I'm getting closer to win", "Hi "+str(OPP_NAME)+", I think we will have a great game", "I WILL NEVER SURRENDER",

Case 3: My precise evaluation function value of "+str(value)+" is impeccable", "It is interesting to choose the best one from "+str(COUNT)+" choices",

Case 4:"It is easy for me to make choice by making "+str(CUTOFFS)+" times alpha-beta pruning"

Status of agent:

1. Finished the Zobrist hashing table
2. Finished the minimax function
3. Finished the alpha and beta pruning

Design retrospective of about 200 to 500 words describing how you designed your agent:

the first step is initializing the Zobrist hashing table. If we call the same state again, we will not c

then we build the make move function: this function builds the state by the State object and by calling the iddfs function to build get the best result

then we build the iddfs function: iddfs function first call the time test and then transfer the minmax function to find the best function

then we build the minmax function: this function combines with the minimax helper function to build recursive function to find the best solution in the limited time.

For the detail of the iddfs function, minmax function and minmax helper function, we firstly using the Zobrist hashing table, then build the children tree of the minimax function. Then using the minimax function to find the best value of all the different children. If the value is better than the current state value, we update the state. If not, we do not update. When using the alpha and beta pruning, we update the alpha and beta. Update the alpha when the value of this state is bigger than the alpha. Update the beta when the value of this state is smaller than the beta. In order to apply the alpha and beta pruning, we build the following condition before all implement. If the alpha >= beta, we break from the function (this is pruning).

partnership retrospective:

Jinlin Xiang:

operated as a team: firstly, we talked together to clearly define the problem we have and build the big map of have to solve this problem. What structure of solution, what is function of each function and their relationship?

What issues you had in terms of collaboration, how you did or did not overcome those issues what if anything you learned in terms of collaboration:

one of the problems is that the imitator does not have right function when we have a lot of runs. The problem is the imitator as P not in north, east, west and east. So, we give more limitation to the p and solve it.

Yuhan Gao:

I designed some part of our code and try to refine it. I also fixed some bugs of our code and implement a cool “personality” of our bot.

What issues you had in terms of collaboration, and how you did or did not overcome those issues, and what if anything you learned in terms of collaboration?

I meet the bug that the bot sometimes “do nothing” when it encounter some situation, and I have fixed that problem by adjust the minimax function.