Steven Alford

CSCI 380

12/5/18

Approaches to Isola Agent Design

For this assignment, I decided fairly early on in development that I was going to attempt to implement minimax, since I believed that this would give me the best possible chance to win, and it would allow me to get a good (possibly best) move in a small amount of time. It took quite a while to get all of minimax working properly, and I ran into an issue where I was getting seg faults and premature termination errors due to not having a base case for a recursive function (duh), which caused minimax to attempt to write outside the bounds of the grid, or make illegal moves.

Eventually, I was able to determine the issue and fixed it, giving me a total winrate, over 1000 simulations, of 57% against the random agent. I knew that this was not acceptable as a competitive winrate, and I knew that my remove function was not working the way I needed it to because it was using the same remove function as the ordered agent; removing the first square available when looping through starting at 0, 0. So this was the next point of focus for me. I determined that, if I was able to remove squares surrounding the opponent, this would increase my winrate significantly.

For the remove function, I decided that I would simply use an idea present in the ordered agent: attempting a move in each direction until one is available. Except, I would instead remove a space instead of trying to remove it, and the point of focus would be on the opponent and not my player agent. I then implemented this idea and saw my winrate increase significantly to 90%.

After getting to this point, I knew that the only thing that could increase my winrate would be my heuristics, so I attempted multiple approaches. First, when implementing minimax, I simply used the heuristic of my player's total legal moves minus the total moves of my opponent. I decided to iterate on this, giving the opponent's moves a higher weight than my player's, effectively making the algorithm see any moves the opponent is able to make as twice or three times better than one of

my moves. I attempted many multiples for my opponent's moves, but saw a decrease in effectiveness after three. Using this heuristic along with everything else, I was able to increase my winrate to 95%.

Not being satisfied with 95%, I did some research online for better heuristics. I found a few other options, such as weighting moves that keep the player in the center of the board higher, and a heuristic that included the use of counting the blank spaces to extrapolate what part of the board would be better to be in. I implemented the heuristic that attempted to keep the player in the center of the board, but it actually dropped my winrate instead of incresing it. I also decided that I didn't necessarily like the moving into a specific quadrant of the board based on blank spaces because of the possibility of being trapped before I got there. However, I did like the idea of using blank spaces to inform my next move, so I decided to implement this along with my weighted legal moves heuristic I already had in place.

For my heuristic, I would count the total number of blank spaces currently available, and, depending on the percentage of blank spaces available from the total number at the beginning, I would apply an aggressiveness to my opponent's moves. At the beginning of the game, my opponent would get an aggressive multiplier of three. Then, below 60% and above 30% of the board available as blank spaces, they would get a multiplier of two, and below 30% it would be equivalent. I believe this was best because it would allow my agent to make more accurate predictions without having to explore the tree completely as the game progressed.

Overall, with this heuristic, I was able to get a winrate of 97% against the random agent, which I believed would give me the best indicator of my ability to beat another classmate's agent, since it would be near impossible to predict their next move. I feel confident in my agent's ability, and, even though there are some implementational things I would change to make cleaner code if I was able to change code outside of my agent (being able to loop through the directional enum would be one), I think that I have put my best foot forward with this agent.