CS 470 Reversi Lab Write-up

Jordan Henstrom and Ben Horrocks

A description of your pruning and valuation function:

Our evaluation checks if we have recursed to the end of the game. If we have, we return how many pieces we have in relation to our opponent as our utility. Otherwise, our function checks which move would be most profitable to make using some predetermined values that we give to each move, and uses that as it's utility.

A declaration of time spent by each lab partner:

Ben Horrocks: 4 hrs

Jordan Henstrom: 4 hrs

A report evaluating your algorithm. This likely will include some kind of analysis of different variations on your algorithm. For example, did your creative work improve or hurt the performance of your algorithm? Did giving your algorithm more time to move improve its abilities? How deep in the tree could you search within X seconds? Does your computer code beat you? You can use videos (short), writing (a couple of pages), etc. to report your evaluations. Make it interesting – focus on things you would like to learn:

We started out by placing our utility on number of possible moves with the assumption that being able to have more mobility in the early game would give us an edge early on. However, in the late game, mobility becomes less useful and other

heuristics become much more useful. We eventually threw away the movement approach and set up instead system that assigned each square a value. Using this we would predict a score of a certain board set up and use that as our utility. Later in the game, when our tree could fill out to the bottom of the decision tree, we would switch our strategy to simply trying to get the most pieces. This approach make the algorithm much better. It would almost always capture corners which gave it an advantage.

I think that initially our creativity hurt our algorithm, since we focused on something that didn't actually help us win. Giving our algorithm more time improved it's abilities, since it was able to recurse further in and find which state would allow it to have the best move at that point. Our tree could go about 6 layers deep without too much of a visible slow down in time. We had never played Reversi until this class, so it is pretty easy for our AI to beat us. In addition, it wins against random almost always. In looking at the one time it beat our algorithm, it's easy to see that random had accidently played a really good game.

In the future, we want to get our AI to make predictions about the strategies of it's opponent, and adjust accordingly so that it can win more reliably against logical players. We plan on doing this by having the AI build a profile based off their opponent's early moves and then compare that profile to stored profiles of different common strategies. When it finds a close enough match, it then modifies the heuristic to combat their strategy.