David Olson

CS 470

Reversi

Basic Implementation

My first task in writing this program was to code up and test the minimax algorithm, shortly followed by adding alpha beta pruning to the code. The minimax algorithm simulates turn taking by the AI and another player. The program generates a tree of possible move sequences, and assumes that the AI will try to maximize its score each turn, and the other player will try to minimize the AI's score on his turn. Score is determined by the count of the AI's pieces at the end of each turn. Each node in this tree either returns a score if it is decided to be a leaf node, or else returns the optimal move based on what happens in the nodes below, which represent turns taken after its turn. The base node returns the final optimal score for the AI, which is the move that it decides to take in the game.

Alpha-beta pruning does not change the theoretical structure of the tree, but it cuts off its search sooner than the regular minimax algorithm. If it is determined that a node's parent will not depend at all on the move that the node makes, then there is no point in searching down that path. Making such a decision is possible when the parent node has several moves to choose from, one has already been searched and looks promising, and then it starts to search another node but immediately that path looks very promising for the opponent.

These two steps in writing this program unfortunately took up a lot of the overall time. Testing and debugging large search trees, along with using a lot of code written by someone else, is very time consuming. The tree for an 8x8 reversi board is quite large, and has not been completely solved. Minimax combined with alpha-beta pruning helps the AI begin to make sense of good and bad moves, but heuristics need to be added to have any chance of beating seasoned human players.

Heuristic and Other Improvements

I haven't played very much reversi, but I do know that winning the corners just about always means winning the game. Therefore, I made my program place a piece on the corners whenever they were available. When the program encountered a corner square somewhere deeper in the tree, both the AI and the opponent always favored the move in the corner. I also attempted to make an even better AI by rating the edge pieces (except the ones right next to the corners) higher than the average square but less than the corners. This is because edges are also important, but the ones right next to the corners often allow the opponent to win the corner. One last creative idea I implemented was to limit the search tree just to the first two nodes when the game was in the first 10 rounds. This was to not use up any time while the game was so young that as far as my reversi knowledge goes not much strategy occurs.

I left a lot of known heuristics unexplored, but I believe the point was to try a couple for now. In preparation for the final reversi tournament, now that the base algorithm is written, there will be more time for testing heuristics. Moving forward I would also like to fine-tune the heuristics that I used.

How the AI Fared

When the AI played the random player using minimax and alpha-beta pruning, it was able to win most of the time. This was encouraging, but to be expected. Even if the algorithm just searches down one level into the tree, it already is finding the move that will win the most pieces each turn, while random is at the mercy of its random number generator. However, if the random player was lucky enough to win some corners, it would beat the AI. It was at this point I realized two things. One is that the importance of corner pieces was an evaluation function that any reversi-playing AI desperately needs to have. The other was that the search tree for the game was large enough that by itself it does not do the AI a whole lot of good.

The search tree grows exponentially, as it is the average number of moves raised to the depth of the tree. I usually tested my program using ten minute games, and found that I needed to limit my tree depth to 6-7 levels. After adding on the alpha-beta pruning feature, that number rose up to 8-9. The AI would need to look even further than that to just by brute force find out that a corner is a good square to win. I did get some benefit in regards to time by not expanding the tree when the game was in its early stages. My testing, which was just playing a few games with the AI, was not rigid but I did not notice any decrease in the performance of the AI. Not much can happen in the first ten moves of the game when the AI is playing a random player or a human that is not that good.

Adding my heuristics to the AI definitely improved its game. This was especially apparent against the random player. It is very unlikely for the random player to win more than a few edges or corners against an AI that aims to obtain the corners and most of the edges. As for its performance against myself, it was harder for me to win than before. I won every game against the basic algorithm, but was down to about 60% against the improved AI. I had to be very careful to not let the AI win the corners, or I was done for. Future work will involve making the AI good enough that I can't outsmart it.

One last note. As the AI is right now, it decides most of its moves very fast. This is because it heavily prefers the corners and edges, so many other branches of the search space are pruned away when one branch comes across an edge or a corner. This happens quite often in the late game. At this point I am pleased enough with an algorithm that for sure will not lose by timeout, but still is an improvement over the basic algorithm. Moving forward though, I must strike a balance between the pruning and continuing to search down the tree.

<u>Other</u>

Time spent: 17 hours

Suggestions on the lab: Somehow make the initial coding easier and leave more time for trying different heuristics. This may include cleaner given code, or starter code for the minimax algorithm

Late Days: I'm not sure if this is being turned in after this lab started to be graded, but if so I think I have some late days to use ☺