Bheemanpally, Jeshwanth
CSE 240 - Winter 2022

Assignment 2 Write Up

1. **What heuristic did you use? Why?**

   When deciding on a utility function, there were a few aspects that came into consideration. One is measuring the number of diagonal, vertical, and horizontal paths that exists for a given user and the length of each path, since we should reward our agent for creating longer paths. We also want to measure the number and length of paths generated by the opponents, as this would give the agent a heads up on potential winning moves for the opponent beforehand. It's also important to note that the paths we do take into consideration have to be open paths that can be potentially extended. For example, a closed off 3-in a row vertical shouldn't really be considered, as the opponent can't turn the path to a four in a row. The heuristic function used is defined below. The weights give the agent an encouragement for longer paths and the subtraction of the current player path value and opponent player path value helps the agent see potential situations where the opponent might win and counter those moves. We penalize for finding opponent paths that are open to extend. One mechanism we lack in this evaluation function is checking to see if one can extend a path on the next move and don't require multiple moves to grow the path.

   Utility(board) = Current Player (30 * #four in rows + 5 * # three in rows + 1 * # two in rows + 0.5 * # one in rows) - Opponent Player (20 * # three in rows + 1 * # two in rows + 0.5 * # one in rows)

2. **Describe how your algorithm performs given different time constraints. How much of the tree can you explore given 5 seconds per turn? 10 seconds? 3 seconds?**

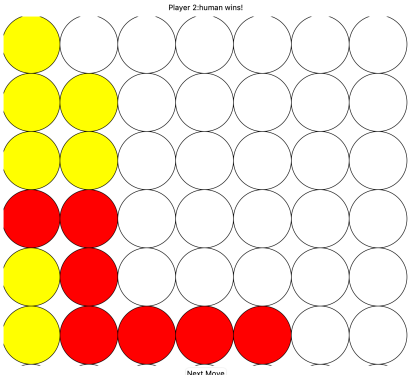
Below is a table listing how deep the expectimax and alpha-beta pruning algorithms perform, given the time constraint. An observation is that the alpha-beta pruning algorithm can't return a decision with a tree depth greater than five without needing more than 30 seconds of time. Expectimax is able to run for depth 3 in 5 seconds and can run up to depth 5 in 30 seconds. What could speed up the process is definitely the tree construction, as this is done before the actual algorithm is ran.

Algorithm	Tree Depth	Time
Alpha-Beta Pruning	4	3
Expectimax	3	3
Alpha-Beta Pruning	4	5
Expectimax	3	5
Alpha-Beta Pruning	4	10
Expectimax	4	10
Alpha-Beta Pruning	4	20
Expectimax	4	20
Alpha-Beta Pruning	5	30
Expectimax	5	30

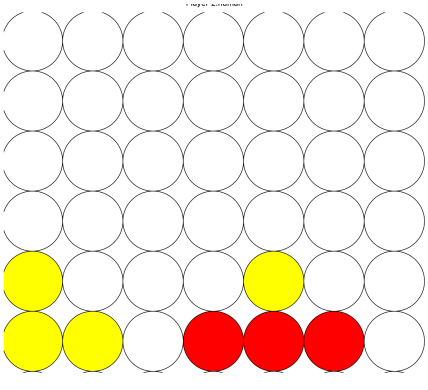
3. **Can you beat your algorithm?**

For an agent using alpha-beta pruning for decisions and tree size 4, I can fairly easily beat the agent. This could be since the heuristic function isn't as robust to help the agent make more strategic decisions. For example, as you observe below, the agent (yellow) didn't detect beforehand that I (red) created a path of three so that regardless of what the agent chooses, I'll win. This could be fixed if the heuristic function penalized the model from allowing the opponent to create paths with multiple ways to extend the path. I can also beat an agent running the expectimax algorithm, as shown in the bottom left image. The agent (yellow) has trouble realizing that it can't produce a four in a row on the first column and ignores my (red) creation of a horizontal path. Further tweaking of the heuristic function is required to help the algorithm ignore paths that can't produce four in a row in a given direction.

Expectimax



Alpha-Beta Pruning



4. **If your algorithm plays itself, does the player that goes first do better or worse in general? Share some of the results.**

I ran an experiment four times, with each AI agent having (4, 3, 2, 1 decrementing per experiment) layers of tree depth and using alpha-beta pruning. The results are shown below. Player 1 wins most of the time, except for experiment 4, but this can be due to the search tree depth being 1. In general, the player that goes first does better.

Experiment	Winner
1	Player 1
2	Player 1
3	Player 1
4	Player 2