Project 3: Utility Based Agent

In this project, I used Monte Carlo Tree Search(4 steps: selection, expansion, simulation, backpropagation) to make plan for the agent. I set the number of iterations for the search to be 100(due to computational power of my laptop, it takes so long to get result of 10,000 runs for higher number of iterations). For each iteration, the algorithm will traverse the by going from the root node and choose one of the children(each children represents a possible action) using UCB1 formula. For each child node, the UCB1 value for this node is: $exploitationTerm + explorationTerm$ where

$$exploitationTerm = value\ of\ the\ node\ in\ current\ state$$

$$explorationTerm = 200 * \sqrt{\frac{\log\left(number\ of\ visits\ of\ parent\ node\right)}{number\ visit\ of\ child\ node}}$$

The constant is set to 200 to balance with the exploration term because the value for each node is high. By setting this constant, the algorithm will explore more paths.

The algorithm will pick the child node that has maximum value from UCB1 formula.

After selecting the node, if the node has not been visited, the algorithm will do simulation step on this node, it will perform all possible actions on the current state and return the maximum value of the values of the states resulting from the actions as the value of the node. Then, the algorithm performs propagation step, the value of the current node will be added to its parent nodes along the way back to the root node.

If the node has been visited, the algorithm will expand the node and pick the first child node to do simulation and back propagation.

The algorithm performs these 4 steps for 100 iterations. Finally, it chooses the best action to take from the root node based on the values of the nodes.

The Monte Carlo Tree Search is implemented in MCTS.java file, and the evaluation function to evaluate the value of a certain state is implemented in WorldState.java.

I used Breath First Search at first, but it took so much time to traverse the tree if its depth is high. If the depth is low, the agent cannot make optimal moves. Therefore, I switched to Monte Carlo Tree Search.

The performance of my agent decreases a lot compared to the Model Based Agent. I think the problem comes from my evaluation function. The agent in my implementation spends a lot of moves in small area and do not explore the environment much.

For running the code faster, please set the debugMode variable in AgentFunction class to prevent the output for each trial to be printed out. The number of trials is also set by trial variable in AgentFunction class