AI Assignment 1 Report

Sanchit Saini
20171191

Aayush Goel
20171188

1) Search Algorithm

Implementation

- We are planning to use **depth limited search** algorithm as a part of searching for the best nodes. The most basic advantage it offers over other algorithms is its completeness and its complexity also bounded by the maximum depth limit we set.
- For its implementation, we use the basic concept of depth first search but if we encounter any node with depth greater

than the specified limit, we consider it to be a leaf node and assume that it does not have any further children.

Advantages

- It is memory efficient(uses linear space in terms of number of nodes).
- It is complete when a solution is present at a depth less than defined depth of search tree.

Disadvantages

- It is not complete when the solution is present at a node with a depth beyond the depth limit specified in the algorithm.
- It does not provide the optimal solution when multiple solution nodes are present.

Minimax Algorithm

- We are planning to use minimax algorithm for computing the minimax decision for the current state.

- We would also use alpha – beta prunning to optimise the algorithm by minimising the number of visited nodes by eliminating nodes which won’t affect the value of the current state.

- We would also randomise the order of child nodes to be explored which in turn makes the algorithm more efficient. (Move Ordering)

- We would also avoid repeated states by keeping a transposition table so that we don’t have to recompute the value of previously computed states.

- We would also use the concept of singular extension to mitigate the horizon effect.

2) Heuristics

When we reach a node with depth equal to the defined limit or run out of specified time limit for a single move, we estimate its utility function by assigning an evaluation function(heuristic).

We would make the heuristic as a weighted linear function of various states with weights assigned to each states based on its utility.

- Initially we will make a function to calculate the utility of moving on to a particular cell of a particular 3 * 3 board by iterating over all valid positions.

- This would be done by calculating the number of zeros and ones in each column after making the move.

- The utility would be an exponential function of the number of ones or zeros present in a row or column.

- We would calculate this function for each cell of the current 3 * 3 board and add all these utilities to get the utility of making a move to this 3 * 3 board.

- We would consider the predefined cost of the 3 * 3 board (Corners – 4 points, Center – 3 points, Rest cells – 6 points) in the utility function of the 3 * 3 board.

- After calculating the utility of each 3 * 3 board, we would calculate the probability associated with each 3 * 3 board to be won by the player. With this information, we would calculate the utility for the whole miniboard by taking into account the patterns that are forming in the miniboard which would be exponentially greater than the utilities of the smaller 3*3 boards.

- The utilities of both the mini - boards is calculated in this way, and the max of both utilities added to the sum of both utilities is the utility of the state.

This is a rough estimate of the heuristic we plan to implement. But if we get a better one, we would update it accordingly.