Report : Alviz Assignment

Group No. 31
CS19M012 : Ashish Gupta
CS19M026 : Harsh Panchal

14 November 2019

Overview

The algorithm that was assigned to us for implementation was **Alpha-Beta Pruning**. Alpha-Beta pruning is a optimization technique over minimax algorithm. It is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, Othello etc.). It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision.Thus, making it faster than standard minimax algorithm.

Algorithm

We were given code platform **TreeCode.py** to implement alpha-beta pruning. We have defined some helper functions for our use which are as follows:

1. **setFirstChildIndexes(self,edge_list)** : This function identifies the first child index for each internal node point in the list of edges and sets the index corresponding internal node point.

2. **getNumberOfChildren(self,edge_list,x,y)** : Given the co-ordinates of the point (MAX or MIN node), it returns the number of children the point has.

3. **getFirstChildIndex(self,x,y)** : Given, the co-ordinates of the point (MAX or MIN node), it returns the index of the first child for the corresponding point (MAX or MIN node).

4. **getIndex(self,x,y)** : Given, the co-ordinates of the point(MAX or MIN node), it returns the index of the point from the list of points.

5. **max(x,y)** : Returns the maximum value among x and y.

6. **min(x,y)** : Returns the minimum value among x and y.

7. **tracePath(self,index,value,depth)** : After computing the minimax value of the root, we trace the path from root to that leaf node whose

values has propagated to the top. The index of points are pushed to the result_path list which renders the path on the GUI window.

We have implemented 3 evaluator functions which contain the actual implementation part:

1. **alphaBetaPruning(self, alpha, beta, depth, index)** : This function contains the actual implementation logic for alpha-beta pruning. It takes alpha and beta values as parameter which are initially set to large value (i.e 10000000). It also contains index and depth parameter which the index and depth of the node currently being evaluated. Our function scans the tree from left to right. It reaches the leaf nodes and starts the evaluation in a bottom up fashion and does the pruning if the condition alpha greater than beta becomes true.

2. **alphaBetaPruningRL(self, alpha, beta, depth, index)** : This function is similar to the above function except that it does the scanning from right to left rather than left to right.

3. **minimax(self,depth,index)** : In order to verify our correctness of alpha-beta pruning we have also implemented minimax algorithm which scans the entire tree from left to right and updates the node values accordingly.