COMS 4995 Parallel Functional Programming, Fall 2021

Final Project Proposal

Jeeho Song (js4892)

Parallel Minimax Agent

Minimax algorithm is a recursive or backtracking algorithm, based on an adversarial search. Minimax algorithm is used in decision-making or game theory to find the most optimal solution. The algorithm can be also used for two player turn-based games, such as Tic-Tac-Toe, Backgammon, and Chess and intuitively well understood through those games playing in AI. Minimax algorithm basically provides an optimal option for a player with an assumption that the opponent also chooses an optimal one in each state. That is, the goal of two players is maximizing their own utility to win.

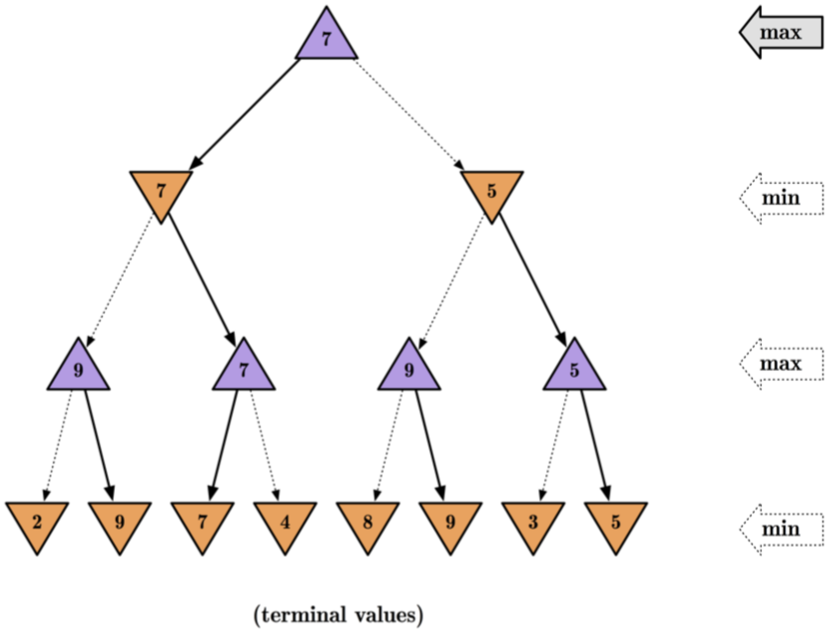

Figure 1 : a search tree

A search trees represents minimax algorithm that proceeds from the top node of the tree to the terminal node and, if any, backtracks the tree. Each node is an option containing a value that represents a utility, and a decision is made based on a comparison of two values in node pair. Terminal values are in nodes of the last level, the so-called leaf nodes. It is assumed that there are two players, MAX and MIN. In Figure 1, the MAX player chooses terminal values from leaf nodes to maximize the value to win the game. So, 9, 7, 9, 5 are selected. Then, the MIN player chooses those values to minimize the MAX's value to win, and 7, 5 are selected. In the same way, the MAX player chooses a bigger value to maximize. Finally, 7 is selected.

This minimax algorithm can be represented in a pseudo-code as below:

```
function minimax(node, isMaximizing):

    if depth == 0  or node is a terminal node then
        return the value of this node
    if isMaximizing
        for each child of node
            childValue = minimax(child, FALSE)
            value = max(value, childValue)
        return value
    else
        for each child of node
            childValue = minimax(child, TRUE)
            value = min(value, childValue)
        return value
```

Considering the consequences of the possible actions, minimax algorithm can be costly depending on its depth. To prove the algorithm, Alpha-beta pruning technique can be implemented to decrease the number of nodes that minimax algorithm evaluates. Here, alpha is the largest value for MAX across evaluated nodes and beta is the lowest value for MIN. The initial alpha value is set as a negative infinity and the initial beta is set as a positive infinity. As the algorithm goes over nodes, the values of alpha and beta are updated. And, when the alpha value is greater than the beta value, the remaining branches are pruned.

In this project, a two player turn-based game is implemented, and the minimax algorithm is utilized to find the most optimal choice for both agents. Particularly, the recurring process of minimax algorithm is dealt with parallel execution to speed up the running time of algorithm. For further improvement of algorithm, an iterative deepening search might be used and also processed in a parallel programming. Although heuristics technique is another way to improve the algorithm in this kind of problems, it will not be addressed here in consideration of the purpose and the scope of the project.