4x4x4 Tic-Tac-Toe Game
CS 440 Programming assignment 3
Phuong Truong
11/26/2019

Problem Definition
In this programming assignment, we were asked to implement minimax or alpha-beta pruning for a 4x4x4 tic-tac-toe game, and decide the way of computing heuristic value of node both efficiently and correctly.

Method and Implementation
I decided to use the minimax algorithm with alpha-beta pruning in this game because it reduced the computing time. In the minimax algorithm, the maximizing player will try to choose the move with the highest score, while the minimum player will try to minimize the score for the maximum player. With alpha-beta pruning, it decreased the number of nodes that need to be searched. I implemented the minimax algorithm with alpha-beta pruning using the following pseudo-code:

```
function alphabeta(node, depth, α, β, maximizingPlayer)
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
         value := max(value, alphabeta(child, depth−1,α,β, FALSE))
         α := max(α, value)
         if α ≥ β then
           break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
         value := min(value, alphabeta(child, depth−1,α, β, TRUE))
         β := min(β, value)
         if α ≥ β then
           break (* α cut-off *)
         return value
```

In order to calculate the above scores, we need to decide the heuristic values. I go over every line and for each line I count how many Xs and Os, and assign value based on the results. There are three possibilities:

- If there is more X than O, then I give a value of 100.
- If there is more O than X, then I give a value of -100.
- If they are equal, then it is a draw: I give a value of 0.

Experiments
I ran two sets of experiment: my AI vs randomness AI, and my AI vs my AI (with similar settings). I used the depth of 4 for testing, which had good performance in short time. For each set of experiment, I had 100 games

played, and recorded win rate for my AI against the opponents, and time my AI used for thinking.

Results

In the first set, my AI vs random AI, I observed that the win rate is 100% for my AI, no matter which plays first. Each game took about 3-4 seconds to finish.

In the second set, my AI vs my AI with similar settings, I observed that the win rate for player 1 is 60%, and the win rate for player 2 is 40%. I also noticed that the AI that makes the first move always wins. The time for each game to finish is longer, about 20 – 22 seconds.

Discussion

In the first experiment set, the 100% wining rate means that my AI is working successfully at beating a player that just plays randomly, using minimax algorithm with alpha-beta pruning. It is also efficient since it only takes 3 seconds to finish the game.

In the second experiment set, the wining rate for player 1 is higher. This means that in 100 games, player 1 played first in 60 games, and player 2 played first in 40 games. I noticed that they always made the same moves in every game, with similar wining states, and it took much longer time to finish each game compared to the first set. This means my choice of heuristic value is not efficient enough. When 2 AIs with similar settings play against each other, the result of the game should be draw because they have similar strategy. This could be improved by choosing different heuristic values.

Conclusion

Using minimax algorithm with alpha-beta pruning and heuristic value, my AI program is able to play a 4x4x4 tic-tac-toe game. My AI beats the random AI with 100% rate in only 3-4 seconds, but the result for the second set is not very efficient. In future works, I may implement different strategies for heuristic value in order to improve the efficiency and correctness of my AI.

Credits and Bibliography

1. Russell, Stuart J., and Peter Norvig. *Artificial Intelligence: a Modern Approach*. Prentice Hall/Pearson Education, 2003.
2. https://medium.com/@alialaa/tic-tac-toe-with-javascript-es2015-ai-player-with-minimax-algorithm-59f069f46efa