Ultimate Tic-Tac-Toe

BASIC INFO:

Team Name: *SRx2*

Team: Ria Mittal; Suraj Gutti; Raghav Sairam; Shubham Gupta

Source Code:

https://github.com/shubhamgpt/Ultimate_Tic_Tac_Toe_AI_Bot

Demo Video:

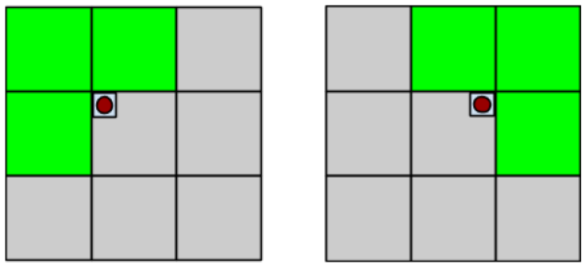
https://youtu.be/CnKF8cnNSzw

OVERVIEW OF PROJECT:

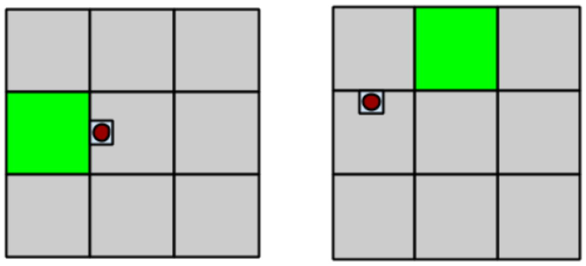
Ultimate tic-tac-toe is a board game composed of nine tic-tac-toe (simple) boards arranged in a 3-by-3 grid. Players take turns playing in the smaller tic-tac-toe boards until one of them wins in the larger tic-tac-toe board. Strategy in this game is conceptually difficult, and has proven more challenging for computers. [Wiki]

Rules:

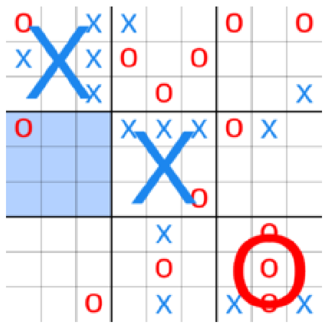
- Each small 3-by-3 tic-tac-toe board is referred to as a local board, and the larger 3-by-3 board is referred to as the global board.
- The game starts with X playing wherever he wants in any of the 81 empty spots. This move 'sends' his opponent to its relative location. In a sense, player 1 can influence or limit the moves that player 2 can make and vice versa.
- If one of the four diagonally corner (top-left. Top-right, bottom-left and bottom right) nodes are played previously, the next play should be made within the diagonal corner-most local board and also both the local boards that are immediately adjacent to it (top, down, left, right).



- If the previous play was made in either of the remaining five options (middle, top-middle, bottom-middle, left-middle and right-middle), then the next play can only be made in that corresponding local board within the global board



- For example, if X played in the middle of his local board, then O needs to play next in the local board in the middle of the global board. O can then play in any one of the nine available spots in the top-right local board, each move sending X to a different local board.
- If a move is played so that it is to win a local board by the rules of normal tic-tac-toe, then it wins that local board.
- Once a local board is already won, more moves can be played that board. But it doesn't make a difference. Unless if that board is completely filled, the player can choose any other board on the global board.
- Game play ends when either a player wins the global board, or there are no legal moves remaining.



AI Techniques:

- We implemented Heuristic scoring techniques.
- Minimax Search (Minimax tree with backtracking algorithm) with Alpha-Beta Pruning. In Minimax search Adversarial Search), we examine the problems that arise when we try to plan in a world where other agents are planning against us.
- Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree.

PREVIOUS RESEARCH:

- AI Approaches to Ultimate Tic-Tac-Toe
  - http://www.cs.huji.ac.il/~ai/projects/2013/UlitmateTic-Tac-Toe/files/report.pdf
- Group Actions on Winning Games of Super Tic-Tac-Toe
  - https://arxiv.org/pdf/1606.04779.pdf

Previous research has focused more on just implementation techniques. As part of this project we would like to implement different difficulty levels of AI bot; which is challenging in terms of defining some parameters and manipulation of algorithms for different levels. We also plan to have one extra level which will be AI bot vs AI Bot; to implement this we plan to have randomized AI bot selected for each side from the different level bots we have.

STAGES OF DEVELOPMENT (Steps of our Implementation):

1. Build the game with rules/GUI and make the AI obey the game rules
2. Implemented a heuristic scoring system for quality of move made.
3. Focusing on AI for local Tic-Tac-Toe; just a simple 3x3 Tic-Tac-Toe
4. Focusing combining the 3x3 TTT to bigger (Ultimate TTT)
5. The difficulty level is defined by the depth of the MiniMax Tree. (increase in depth => increased difficulty)

TOOLS USED:

- Languages: Python
- IDE : Atom, PyCharm
- OS : Linux

RESULTS:

- The gameplay would be command line driven.
- Different tree depths define the difficulty of the AI. Since there is no other concrete implementation of the AI which we could compare with, the different difficulty levels were pitted against random AI.
- The win-rate of the AI increased with the tree depth.

FUTURE WORK:

- Use Q-Learning to improve heuristic evaluation function for move quality.
- Implement an interactive GUI for the game.
- Implement adaptive AI difficulty based on the player's skill level.
- *There is a Game AI competition online for Ultimate Tic-Tac-Toe; We plan to join the competition with our AI bot to compete with other AI bots. http://theaigames.com/competitions/ultimate-tic-tac-toe*

LEARNINGS:

- Nuances/difficulty of implementing a heuristic function for complex adversarial games.
- Application of min-max algorithm with alpha-beta pruning.
- Even games that appear to be simple, could have complex winning strategies.