Finding the Best Strategy in Ultimate Tic-Tac-Toe

Introduction

The famous Tic-tac-toe is a simple 2-player strategy game on a 3x3 grid where the winner is the first player to mark three in a row, column, or diagonal. However, it has been proven that with the correct strategy, the game will always end on a tie. Therefore, to make it more interesting, an extension of the original, the “Ultimate Tic-Tac-Toe”, emerges where there are nine tic-tac-toe squares on a 3x3 grid, and the objective is to be the first to win three tic-tac-toe squares in a row, column or diagonal.

The original Tic-Tac-Toe is a strategically simple game that, if played correctly, will always result in a tie. However, for the variant of Ultimate Tic-Tac-Toe I chose for the project, there is no known “best strategy” (the rules are listed below). Intrigued by its complexity, I am drawn to try and solve for an optimal strategy with the help of genetic algorithms. The methodology is described below.

Objectives

1. To find empirically the best strategy, or player character, in the game of Ultimate Tic-Tac-Toe.

2. To determine whether the resulting player of the genetic-algorithm approach is better than that of the traditional tree-search approach.

Rules of the game

The goal of the game is to win the Ultimate Tic-Tac-Toe on the big board by winning the individual Tic-Tac-Toes on the small boards. This gives 9 small boards with 81 small squares. Each turn, the player marks one of the small squares.

There are multiple variants of Ultimate Tic-Tac-Toe available on the Internet, and each has a set of rules that changes the way the game will proceed. The below is a list of the set of rules I will implement for the purposes of the project.
1. Where the current move could be placed is governed by the previous move. The current move must be placed in the board that has the same position relative to the big board as that of the previous move relative to its small board. For example, if player 1 placed an X in the center square of the small board, player 2 must play in the small board in the center. Consequently, if player 2 placed an O in the top-right of the center board, player 1 must then play in the top-right board.

2. Once a small board is won, that board is no longer playable. If a player is directed to a small board that has already been won, the player is free to choose to play on one of the other playable boards of his choice. A small board is still playable even when it is an obvious tie, and it remains so until all small squares have been filled.

3. A tied small board will count as both X and O, meaning either player may win the game with that small board being one of the three connected boards.

The set of rules of this variant are described on this website: http://mathwithbaddrawings.com/2013/06/16/ultimate-tic-tac-toe/.

Methodology and Limitations

To achieve the objective of the project, I propose to program a simulator with a group of players with different “characters”. Each player is characterized by a string, the “gene”, that dictates how it would proceed in playing the game. Using the concept of genetic algorithms, simulations will be run with players of different characters playing against each other. Hypothetically, the outcomes of the simulations will show that some characters would be more dominant than others. The more dominant characters will be “crossbred” and the others will be eliminated. A “most dominant” character should emerge after repeating this process a certain number of times, and that “genetic string” will represent the empirical best strategy to Ultimate Tic-Tac-Toe under such methodology.

Although the rationale behind such “evolution” seems logical, the elimination and crossbreeding processes have inherently arbitrary elements that we cannot guarantee that the program will produce the same “ultimate player” every time. For example, when we are about to crossbreed two relatively dominant species, how much of the gene from each species do we extract, and which part should we be extracting from each player? These are some questions I will attempt to answer during the simulation process.

Moreover, the lack of our ability to guarantee the same outcome for every set of simulations also means that we will not be able to determine the theoretical best strategy. One way to mitigate such limitation would be to have the different
“ultimate players” play against each other to find the best one out of all of them, but it would still be impossible for us to prove that it is indeed the best player.

Therefore, to evaluate whether my genetic-algorithm approach works, I propose a two-fold test where the “ultimate player” generated is compared against two benchmark players, namely a random player and a tree-search player. In the first fold, the resulting player will play against the random player. If it does have an empirical advantage over the random player, it will play against a player that uses the traditional tree-search approach. If the “ultimate players” consistently do better in the second stage, we will be able to conclude that the genetic-algorithm approach works.

**List of deliverables**
A simulation script (Python or C)
A full report detailing the data and analysis of the proposed simulations