Analysis of a Contemporary 2x2 Super Tic-Tac-Toe Board

Chris Prijic, Kamini Janardan
August 18, 2012

Introduction, Rules, and Terms

The game of Super Tic-Tac-Toe (STTT) is played on a 3x3 Tic-Tac-Toe (TTT) board, but includes a TTT board in each of its nine squares. To win a square on the larger board, the smaller board must be won. Another added twist to the game allows players to control where the next player is restricted to play on the board; wherever one player plays on a smaller TTT board dictates where the next player is limited to play on the larger 3x3 board (see Figure 1). If a player is sent to a smaller board that has already been won, then this player receives a choose anywhere move, and may play in any open space on the board. To deal with ties on the smaller boards, ties go to the player with the most pieces on the respective board. The object of this game is to win three in a row on the large 3x3 TTT board, strategically weighing your moves and the placement of your opponents restrictions against each other. Though this compilation has not yet been mathematically analyzed, or analyzed at all for that matter, the classic 3x3 individual board has been fully analyzed. This paper, however, is not focused on the 3x3 board; this paper is an analysis of a simpler 2x2 STTT board in order to contribute to a later analysis of the 3x3 board (see Figure 1).

![Figure 1: Empty 3x3 and 2x2 STTT Board](image-url)
Some simplifications should be noted about the 2x2 STTT board. Firstly, a ‘winning’ combination on a 2x2 TTT board is any group of two pieces on the board (vertically, horizontally, or diagonally). Because of this, we shall refer to terminal positions being reached because of a count of ”2: of the same piece on a board.

Also, to avoid confusion when discussing the board and its qualities, some terms should be defined (exemplified in Figure 3). A board refers to the large game-board, and also refers to the size of the game-board as well. A field refers to the smaller game-board that lies in each square of the large game-board. A pre-terminal position refers to a field that has been won, and a terminal position refers to a finished game. Moves are described with two attributes: the field that is being played on, and the square in the field that is being taken. Lastly, the board’s fields and squares are counted from the top-left to the bottom-right, reading left to right, then top to bottom, like a book. See the following diagram for how the counting is done on the 2x2 STTT board. Note that we describe moves between brackets, first listing the field that is being played on, followed by the box in that field that is being played on. So \([1, 1]\) points to the first field, and the first box – the upper left-hand corner of the game-board.

Figure 2: Counting on the 2x2 STTT Board

### Materials and Methods

Because of the small size of the board, we took the approach of a brute-force adaptive learning program to solve the board. Using Hexapawn’s "Matchbox" style punishment system for moves, we were able to solve the game with a Python program.

Before going further, let us see how the program describes a board. This is necessary because further discussions about certain board-states in the game will require this universal board representation. A board-state is represented by a string of letters, slashes and a number, where ‘n’ and ‘N’ stand for an empty square or unwon field, respectively; ‘x’ and ‘X’ stand for player one, and ‘o’ and ‘O’ stand for player two. The last number in the string represents the
field that a player is restricted to; this is a number between 1 and 5, where 5 represents a "choose anywhere" move.

Beginning with the starting board, which is represented by

"nnnnN/nnnnN/nnnnN/nnnnN/5",

where each lowercase letter refers to a square on the small board, and the capital letters refer to the overall big-board representation. In this case, all squares are empty, all fields have not been won, and player one gets to choose to play anywhere on the board, represented by the '5' at the end of the string. Another board-state, represented by

"nxnxX/onnxN/nonnN/nnonN/3"

refers to the following board where the first field holds an 'x' in the '3' spot, the second field holds an 'x' in the '4' spot, the third field holds an 'o' in the '2' spot, and the fourth field holds an 'o' in the '3' spot. The '3' at the end tells us that player one is restricted to the third field, and the 'X' in the first field tells us that player one has won the first field, replacing the field with an 'X' instead of a small 2x2 game.

![Figure 3: Representation on the 2x2 STTT Board](image)

**Results and Findings**

Resulting from the adaptive learning program, we found that player one would win ten thousand times in a row in just under 100,000 games (because the computer plays with a random strategy at the beginning of the adaptive program, the resulting total game count varies, but always stays just under 100,000 games). Strangely, we expected a much higher number of games to be played to see every possibility, but it came to our realization that *player one can win a great many different games*, with many choices move-wise. Player two had a lot of empty matchboxes, while player one had many full matchboxes – therefore the game-count needed to win 10,000 games in a row was much smaller.
Due to the time it took to get the adaptive program done correctly, mathematical analysis of the game was unfortunately out of the picture, but to prove that our matchboxes hold the true perfect strategy for winning unfairly, we decided to draw up a game tree to prove that no matter what player two does, player one can always counteract their opponent and win unconditionally.

After these results, we wanted to develop a strategy for player one to play by, a strategy that is memorable and applicable by human beings as well. We found a basic 5-rule strategy to follow, but there are exceptions and loopholes to this strategy. As long as the player follows this strategy and looks a couple moves ahead (easy to do with such a simple board) to verify the moves effectiveness, that player one can always win. The rules are as follows, from highest effectiveness to lowest effectiveness: 1) Play where it sends your opponent to an empty field, 2) Send your opponent to a field you have played on and they have not, 3) Send your opponent to a field where only they have played, 4) Send your opponent to a field that both of you have played on, but have not won, and 5) Send your opponent to a field that has been won. These rules hold most of the time, and work very well as a general strategy, but know that players must still look ahead a couple moves to verify.

To completely prove our adaptive program correct, we decided to develop a game tree, where no matter what player two does, player one will always win. Game trees for the 2x2 game were hand written beginning at the board displayed below with a move by player one. Ever possible move for player two was given its own branch on the tree, and each move was counteracted by one of the optimal moves by player one. Essentially, it was proven through writing that no matter what player two does on any particular board, player one has a specific set of moves that he or she can play in order to win. In proving that the 2x2 board was an unfair game it was necessary to show all possible player two moves, and thus 5 moves into the game, there were already 16 branches to this game tree. At 6 moves into the game, there were 53 branches. Finally, after each branch had been exhausted in the end of a game, there were 126 branches. Seeing that each game lasts an average of 11 moves, this isn’t really an unreasonable end to the game. Each of these branches had been concluded in a player one win, thus showing that player one has the strategy to win in response to any move enacted by player two.
Lastly, we began working on an artificial intelligence for the game, in hopes of noticing working strategies we can apply to the game. Using the Minimax Algorithm, we allowed the program to look forward four moves ahead, evaluate the boards, and then choose the corresponding best move based on those ratings. What we created was a rating system based on an average of the un-won fields, as well as a weighed addition of the board’s large-game evaluation. On the 2x2 STTT board, this was quite easy: 5 points for a won field, 3 points for a piece and no opponent piece, 2 points for a piece and an opponents piece, 0 points for an empty field, -3 points for only an opponents piece, and -5 points for the fields that an opponent has won. Lastly, the board is rated in the same fashion, and then the score is tallied up as follows:

\[
\frac{\text{field}_1 \text{score} + \text{field}_2 \text{score} + \text{field}_3 \text{score} + \text{field}_4 \text{score}}{4} + 2 \times \text{boardscore}
\]

For the board shown in figure 3, the score would be, for player 1, as follows:

\[
\frac{5 + 2 + -3 + -3}{4} + 2 \times 3 = 6.25
\]

This scoring system is simple and applicable, but not optimal.

Unfortunately, because moves that a player makes at the beginning of a game can lay dormant until the end of the game, then suddenly have a great importance, it was hard to come up with a good evaluation function. We hoped that using a function that always evaluates the whole board would be a good idea, so that even the first couple moves would be evaluated on every board, up until the very end when they become paramount, but this did not work for that very reason. We hope to improve this function, and we will discuss this in the next section.

**Future Works**

In the future, there is the possibility of creating a better artificial intelligence for the 3x3 game by creating a better evaluation function. Using new strategies
to develop this evaluation function would lead to a better AI through a more difficult to beat opponent. This idea however seems rather unlikely to help, as moves made at the beginning of a STTT game can lay dormant until the end of the game, and so some moves might be rated as bad moves looking ahead four moves, but if you looked ahead twenty or thirty moves, then the moves importance would be obvious; this is something along the lines of the horizon effect. However, if there is a way to rate these dormant moves, or rate the first moves (just like there are good starting moves in chess, even if people do not see their importance until the end of the game) could lead to a descent evaluation function that could be acceptable for such an algorithm. Also, applying other algorithms, such as Alpha-Beta Pruning and Endgame Databases could also lead to an exciting and well-developed computer player.

Secondly, inspired by the applications of 2-D graphics to 3-D graphics and vice-versa, we feel that there can also be some greater mathematical analysis done in the process of solving the 3x3 board. Although the 2x2 board wasn't necessarily solved using math itself, we feel that finding mathematical patterns in the 2x2 board would provide valuable insight into the process of solving the 3x3 board. Using this mathematical analysis, we could potentially understand why a certain strategy works, and why other strategies seem good but really end up losing the game; a mathematical analysis can lead to a deeper and more all-encompassing look at the game Super Tic-Tac-Toe.

Lastly, pattern recognition would allow for an in-depth analysis of the 3x3 board using aspects of the game that were not previously recognized by a human alone, and might not be prominent or even predicted by the current type of minimax artificial intelligence. If a program can be developed to continuously play games of STTT and develop strategies based off of patterns that it notices leads to wins or losses for each player, then the program can not only work on solving the game in a new and exciting way, but also create strategies that work that are also memorizable by humans – bridging the gap from adaptive learning to creating practical strategies, making games and their solutions easier to apply in real situations with human players.