Harnessing Artificial Intelligence Algorithms
Yoel Ferdman

1. **Problem Definition**

Program an opponent for a game – 3x3 Tic-Tac-Toe, and 4x4 extension – in Python and demonstrate its performance against a human player. I chose to improve and adapt code for 3x3 tic tac toe and its computer moves AI algorithm as well as to create a version that is 4x4 tic-tac-toe. You can test it out. I bet you can't beat my AI!

2. **Solution Specification**

Adapted original 3x3 computer opponent from online.

The original code had the computer play under 4 simple rules, playing them in order, if possible:

1. First, take the winning move if it exists.
2. Then, block the player from their winning move if they have one.
3. Next, take a random available corner (if any).
4. Finally, take the center (if available).
5. (Then finally take a random edge/side if nothing else is available).

I played this computer several times and realized it actually wasn't any good because I could beat it quite frequently. One case in which I found I could win was this: If the player goes first and takes a corner, the computer randomly selects from the three remaining corners. However, this is not optimal because if the player selects the corner opposite from his original choice, then he will guarantee a win. (The computer will be forced to choose the center for its second choice to prevent a player win, then the player will have the last corner, leading to a two-way guarantee). One good choice is for the computer to take the opposite corner. This

prevents that scenario from occurring. However, the most conservative move in this case would be to take the center. As I will explain in the following section, a move like the one mentioned could result in a guaranteed computer loss if the player plays his next hand right. I added this section to the piece of code determining which corner to select. The case is different for 4x4 so I only included this is the 3x3 version. (Note: the 4x4 piece of code is simply for purpose of extending this project to an further domain. If I gain tools and understanding of optimal moves and strategies for the 4x4 version in the future, then I can definitely go back and add in the computer AI moves to this). I took examples like this, among several others (i.e. when the computer has an opportunity for a guaranteed win in the following move) and added them to the code. The guaranteed wins often require a setup beforehand which I had to hard-code in. Another one I chose to include was the third example in the graphics in next section (Analysis of Solution) (guaranteed wins image).

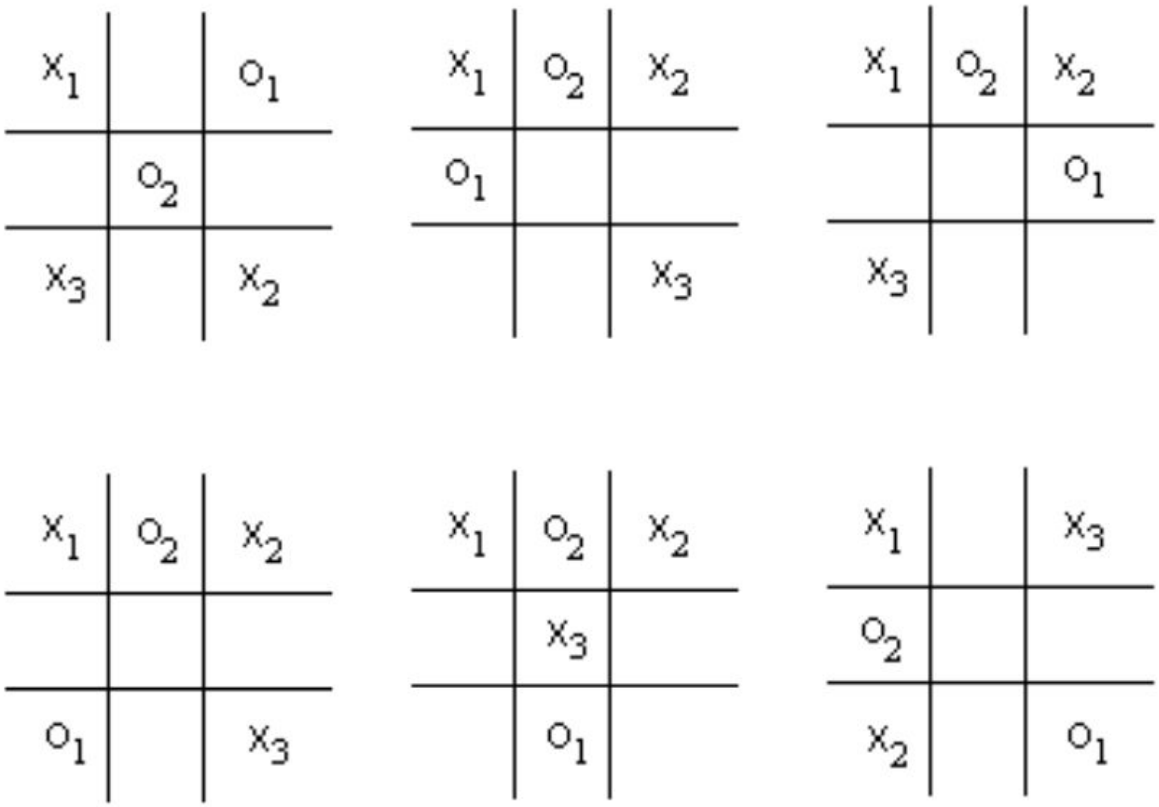
Another reason why the computer would lose when I played against it is that it would prioritize corners even when the center is the way to go. If the computer plays second, and the player's first move is a corner, the best thing to do would be to take the center and hope for a player mistake or settle for a cat's game (tie). The reason is that if the computer doesn't take the center, then the player can use one of the guaranteed wins shown below which will defeat the computer.

3. **Analysis of Solution**

As mentioned in the previous section, the first addition I included in my project was the particular case in which player has first move and takes a corner. The optimal computer move in that situation would be the opposite corner.

If both players of a tic-tac-toe game are playing totally optimally, the game will result in a tie, regardless of who starts. If one of the players makes a mistake, then the other player might win. The only way to never lose is to never make mistakes. I decided to improve the tic tac toe AI computer rules to make the computer unbeatable (this doesn't necessarily mean it wins every time, simply that it does not lose). The best way to win tic-tac-toe is a move that guarantees a win. I added to the algorithm, some code that spots if there is a move that guarantees a win the following move, no matter what the player opponent chooses.

Here are some examples of guaranteed wins from (Aycock, 2002):



Please note that it is very difficult to win if you don't go first. However, this computer program **will not lose** even if it doesn't go first.

Another addition I made was for the case in which the computer plays first a corner, then the player plays an adjacent side space immediately after. The optimal move wouldn't be for the computer to choose a corner at random but rather the corner on the axis orthogonal to the direction of the player's chosen side space. This will lead to a certain victory.

4. References:

1. https://www.cs.jhu.edu/~jorgev/cs106/ttt.pdf
2. https://inventwithpython.com/chapter10.html
3. http://www.instructables.com/id/Winning-tic-tac-toe-strategies/