AI Assignment 1: Xtreme Tic Tac Toe

How the Bot works

In the initial report submitted, we planned to use Monte-Carlo Search on Tree. However, due to constraints added by the submission format, we used a simple Minimax with Alpha-Beta pruning, and custom made heuristics. Optimisations were also made to exclude moves that might lead to local victories for the opponent that might not be considered by the search itself. Possible extensions we considered were to use a lookup table generated by MCTS, but we decided against it in favour of improving the bot with better heuristics.

Heuristics

We tested the bot with three heuristics, ranging from absolutely trivial to rather complex. Afterwards, we added some additional considerations on top of the third heuristic to form an improved **fourth**, which is the one we used eventually.

	Heuristic 1	**Heuristic 2**	**Heuristic 3**	**Heuristic 4**
Definition	Play the first move available to the bot	If smallboard won in future move, then +2, if opponent wins then -1.	Same as Heuristic 2, but with weighted increase or decrease based on weights assigned to corners and sides.	Same as Heuristic 3, with local optimisation to block any wins for opponent if possible
Performance Against Random	96%	97%	97%	98%

Results and Analysis

On the whole, we were pretty disappointed with the results. In a lot of cases the heuristics applied did not seem to be reflected in the decisions eventually taken by the bot.

A very notable example of this is in the local optimisation made in Heuristic 4. The way it is supposed to work is: if a move by Our Player (p1) allows Other Player (p2) to win a smallboard in a single move (i.e, it checks if there are two p2 flags in a row), then p1 will not make the move. As evident, it fails:

Case study: Local Check Heuristic Failure

> *Team 14: p2, x*
>
> *Team 24: p1, o*

Game state before move. Move to be made in Row 1, Column 0 of larger board.

```
================BigBoard State================

 x o o  x o x  o x o      - - -  x - -  - - -
 o o x  - - -  x - -      - - -  - - -  - - -
 x x -  - - -  - - -      x - -  - x x  - - -

 - - -  - - -  o o -      x - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - x  - - -
 - - -  - - -  - - -      - - -  - - -  - - -

 o o -  o o o  o - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - x -  - - -      - - -  - - -  - - -

==============SmallBoards States==============
- - -    - - -
- - -    - - -
- o -    - - -
==============================================
```

p1 identifies (0, 3, 0) as a possible move.

This move allows **p2** to win (0,0) of larger board in the next move. So it should not be allowed. But, **p1** makes the move anyway and **p2** wins the (0,0) board.

```
('CONTINUE', '-')
================BigBoard State================

 x o o  x o x  o x o      - - -  x - -  - - -
```

```
 o o x  - - -  x - -      - - -  - - -  - - -
 x x -  - - -  - - -      x - -  - x x  - - -

 o - -  - - -  o o -      x - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - x  - - -
 - - -  - - -  - - -      - - -  - - -  - - -

 o o -  o o o  o - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - x -  - - -      - - -  - - -  - - -

==============SmallBoards States==============
- - -    - - -
- - -    - - -
- o -    - - -
==============================================


('CONTINUE', '-')
================BigBoard State================

 x o o  x o x  o x o      - - -  x - -  - - -
 o o x  - - -  x - -      - - -  - - -  - - -
 x x x  - - -  - - -      x - -  - x x  - - -

 o - -  - - -  o o -      x - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - x  - - -
 - - -  - - -  - - -      - - -  - - -  - - -

 o o -  o o o  o - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - x -  - - -      - - -  - - -  - - -

==============SmallBoards States==============
x - -    - - -
- - -    - - -
- o -    - - -
==============================================
```

A possible reason for this failure is that there is no secondary move option chosen beforehand that the bot can take in such cases. A simple programming error, cost us in at least 2 games (against Bot 13, both that and ours had near identical moves with the exception of win blocking and one other move sequence).

Another heuristic we had was to optimise local board victories. As in, if the board has 2 **p1**s in a row or in any formation such that a single move wins the board, then that move should be taken over other heuristics. Again, this failed.

Case Study: Failure in Local Optimisation

> *Team 13: p2, x*
>
> *Team 24: p1, o*

Game state before move. Move made by **p2** at (0, 0, 6). **p1** has to play in the (0, 0) of the larger board, where it can make a single move (0, 1, 1) to win the local board, which is optimal at the early game.

```
================BigBoard State================

 x - o  - - -  x - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 o - -  - - -  - - -      - - -  - - -  - - -

 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -

 x - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -

==============SmallBoards States==============
- - -    - - -
- - -    - - -
- - -    - - -
================================================
```

But, the bot makes the move (0, 2, 2). It is setting up a later double victory option, but misses an obvious chance that seems to be missed in executing the code.

```
('CONTINUE', '-')
================BigBoard State================

 x - o  - - -  x - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 o - o  - - -  - - -      - - -  - - -  - - -

 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -

 x - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -
 - - -  - - -  - - -      - - -  - - -  - - -

==============SmallBoards States==============
```

```
- - -     - - -
- - -     - - -
- - -     - - -
=============================================
```

Improvements that can be done

More Attacking style of play

All heuristics promote a very defensive play style, so chances that should have been taken are missed out on.

Improvements will be made if the bugs discussed in the analysis above are fixed, but some additional things that can be done:

1. Track all boards where a single move wins the board. If the player is moved to that board, or has freedom of placement, then it should give the board-winning move(s) higher weightage when calculating utility.
2. Have a decay factor that leads to more attacking or defensive plays at different parts of the game. For instance, it could play aggressively in the endgame, or at the start. Experiments need to be run to see which performs better.

Heuristics for Global Performance

The current heuristics are all functioning on the local board, without much regard for global performance of the bot.

More improvements can be made along this front.