COMP6231 COURSEWORK: PART2

INTRODUCTION

This part investigates the tic tac toe game through the minimax algorithm, the alpha beta pruning and exploiting the symmetries of the board. Furthermore, the game was expanded to a 5x5 board size with a winning condition of 4 in a row. The reason of the implementation in the bigger board is to find an effective way of reducing the complexity without searching the whole available move set.

METHODOLOGY

Minimax algorithm

Tic tac toe is a strong solved game, with not very big complexity so it is feasible for a CPU player search all the available moves and pick the best. Minimax is undoubtfully the best way to search for the best move. Minimax algorithm runs recursively playing alternately for both the human player and the CPU player evaluating every move. Specifically, in the 3x3 board size, max player if he plays first, has 9 available moves, min player 8, then max player 7 resulting in a total of 255,168 board combinations. For each move, checks if there is a winner and evaluates the node. If the min player wins the node is evaluated as -1, otherwise +1 unless it is a tie where gets evaluated as 0. For both players (max and min) minimax picks the best move for each one, so it is optimal against an optimal rival.

Alpha-beta pruning

To reduce the complexity, minimax can prune some branches if their score is lower than an already bigger one. In other words, minimax prunes leaves with moves with high scores for max player if a lower score is already found and min player will choose it (as he plays the best for him).

Symmetry

A 3x3 tic tac toe has 255,168 possible final boards.

Win in 5 moves	1,440
Win in 6 moves	5,328
Win in 7 moves	47,952
Win in 8 moves	72,576
Win in 9 moves	81,792
Draw	46,080
Total	255,168

Table 1. Total Ending boards reproduced from (link)

We can exclude symmetries from possible moves:

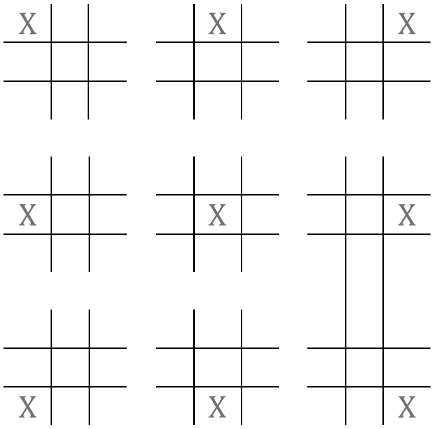


Figure 20. All Possible moves

turns into:

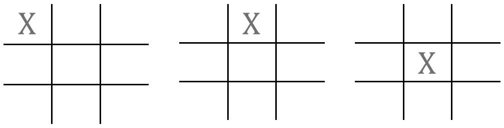
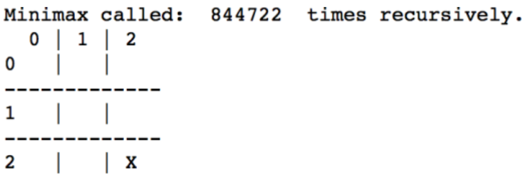


Figure 21. Possible moves of empty board without symmetric positions.

Symmetries are all the 90° rotations, plus the vertical and horizontal mirroring.

We can see in a game starting with the CPU player in an empty board result in calling recursively 844722 times the minimax (searching in depth 9) for picking the first move:



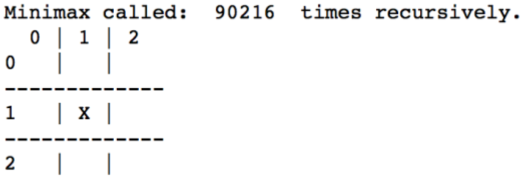
```
Minimax called:  844722  times recursively.
   0 | 1 | 2
0    |   |
-------------
1    |   |
-------------
2    |   | X
```

Figure 22. First move without symmetries.

Excluding the symmetric moves the total 255,168[2] ending boards reduce to 26,830. The latter number is only 1/8 of the former. This result calling the minimax algorithm only 90,216 times.

Win in 5 moves	172
Win in 6 moves	579
Win in 7 moves	5,115
Win in 8 moves	7,426
Win in 9 moves	8,670
Draw	4,868
Total	26,830

Table 2. Total Ending boards reproduced from (link).



```
Minimax called:  90216  times recursively.
   0 | 1 | 2
0    |   |
-------------
1    | X |
-------------
2    |   |
```

Figure 23. Excluding the symmetries from minimax.

With such a small complexity it is easy explore all the possible moves and making an unbeatable CPU player.

EXPANSION

A 5x5 grid has a complexity of around $25! \approx 10^{25}$ completed boards so it is practically impossible even we exclude the symmetries to reach in depth 25 to search for the best move.

Because of the intermediate places in the 5x5 grid symmetries include: 4 rotations of 90°, horizontal and vertical mirroring, mirroring on the main diagonal and the opposite diagonal. So, from 25 different positions, we end up with 6 in an empty board.

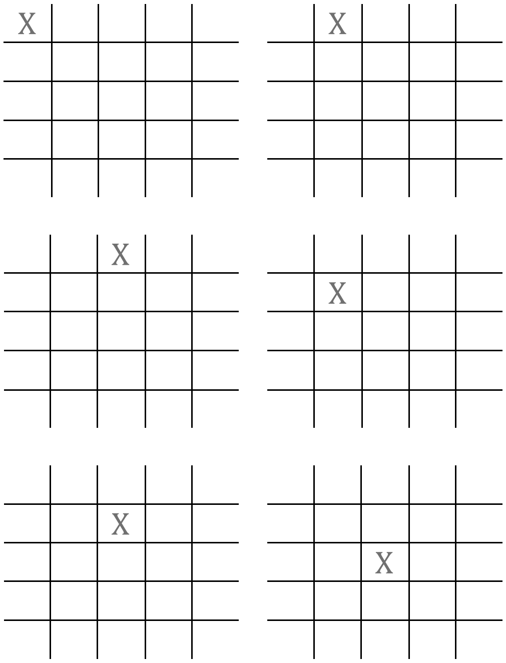
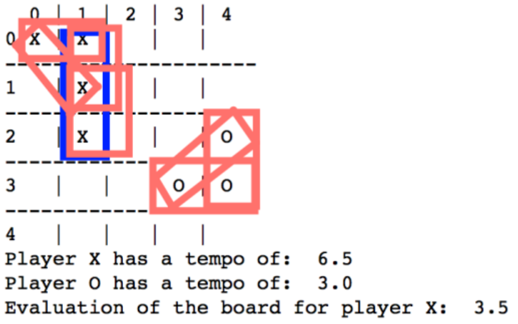


Figure 24. Initial symmetrical board states.

Board evaluation

In order to estimate who is in winning position an evaluation function was implemented giving to the player some "tempo" (named as the strong moves in chess). The tempo of the board is calculated as:

$$t(X|O) = \sum two_{tiles}^{border} + 2.5 * three_{tiles}^{border}$$



```
  0 | 1 | 2 | 3 | 4
0 X | X |   |   |
--------------------
1   | X |   |   |
--------------------
2   | X |   |   | O
--------------------
3   |   |   | O | O
--------------------
4   |   |   |   |
Player X has a tempo of:  6.5
Player O has a tempo of:  3.0
Evaluation of the board for player X:  3.5
```

Figure 25. This board favors player X.

The player with *O's* has a score of 3 as he has 3 combinations of 2 bordering *O's.* Player with *X's* has 1 row of 3 bordering *X*, and 4 rows of 2 bordering *X.*

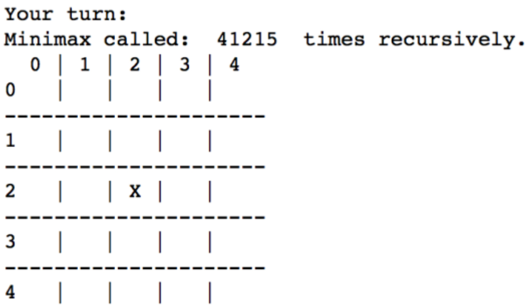
The score of the above board for player *X* is:

$$score = 1 \times 4 + 2.5 \times 1 - (1 \times 3) = 3.5$$

The factor 2.5 is added because three in a row are always better than twos in any combination.

Reducing the complexity

As far, the complexity is reduced only by searching in a depth smaller than the maximum and each time the board is evaluated. However, is big enough and looking ahead only 4 depths minimax generates 41,215 moves with an empty board and 34,750 playing as second. As the game continues, things get worse as the symmetry falls apart.



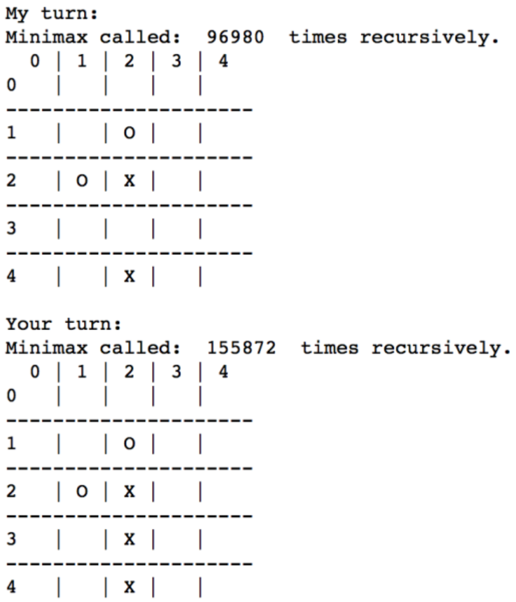
```
Your turn:
Minimax called:  41215  times recursively.
   0 | 1 | 2 | 3 | 4
0    |   |   |   |
----------------------
1    |   |   |   |
----------------------
2    |   | X |   |
----------------------
3    |   |   |   |
----------------------
4    |   |   |   |
```

Figure 26. Less complexity from classic tic tac toe (Figure 24).

Nodes minimax expanded	
1st move (x)	41,215
2nd move (o)	34,750
3rd move (x)	115,522
4th move (o)	96,980
5th move (x)	155,872
6th move (o)	126,845
7th move (x)	99,848
8th move (o)	83,866

9th move (x)	65,858
10th move (o)	50,912
11th move (x)	38,670
12th move (o)	28,784
13th move (x)	20,930
14th move (o)	14,808
15th move (x)	10,142
16th move (o)	6,680
17th move (x)	4,194
18th move (o)	2,480
19th move (x)	1,308
20th move (o)	567
21th move (x)	290
22th move (o)	104
23th move (x)	24
24th move (o)	6
25th move (x)	1
Tie!	

Table 3. A game between 2 CPU players with (max. depth 4).



```
My turn:
Minimax called:  96980  times recursively.
  0 | 1 | 2 | 3 | 4
0   |   |   |   |
---------------------
1   |   | o |   |
---------------------
2   | o | x |   |
---------------------
3   |   |   |   |
---------------------
4   |   | x |   |

Your turn:
Minimax called:  155872  times recursively.
  0 | 1 | 2 | 3 | 4
0   |   |   |   |
---------------------
1   |   | o |   |
---------------------
2   | o | x |   |
---------------------
3   |   | x |   |
---------------------
4   |   | x |   |
```

Figure 27. Searching all moves in a depth of 4, X decided to play "r=3, c=2".

As you see in Table 3, as the game advances and the symmetry breaks, the minimax is called more and more times ending in a not so efficient complexity. The complexity can be reduced either by making minimax looking at 2 depths ahead, and getting poorer results, or keeping a small number of possible moves and discarding the rest of them.

The challenge here is to order them efficiently and not discarding beneficial moves.

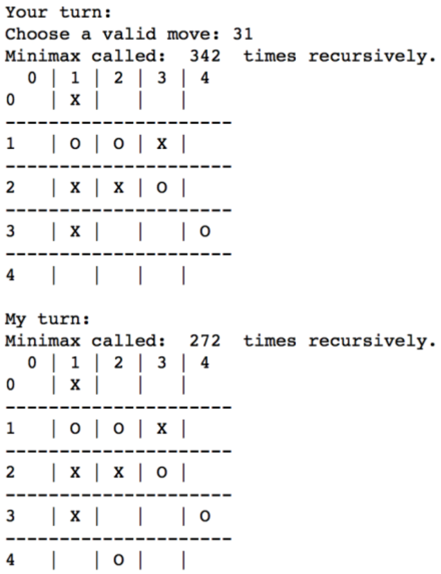
Available moves were ordered regarding the board evaluation for every move, and from the order list only the 10 best moves were kept.

This strategy works well for exploiting max player's territory, but if opponent builds traps, with no bordering tiles minimax cannot foresee them as the moves to block him are discarded.

RESULTS

The minimax algorithm on the classic tic tac toe had no interest at all as it ties always both humans and itself. However, the 5x5 board was tested on humans (every game with itself resulted in the same board; no luck included).

Humans lose only because of gaffes.



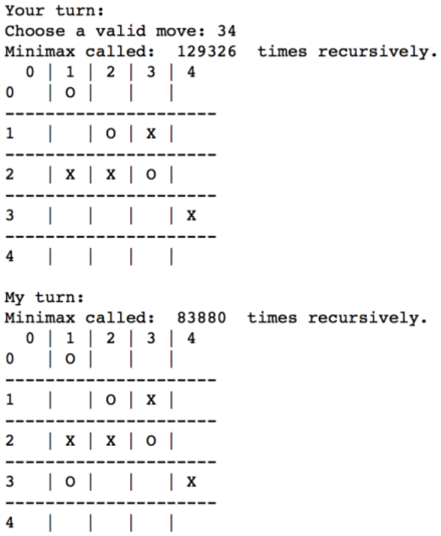
```
Your turn:
Choose a valid move: 31
Minimax called:  342  times recursively.
  0 | 1 | 2 | 3 | 4
0   | X |   |   |
---------------------
1   | O | O | X |
---------------------
2   | X | X | O |
---------------------
3   | X |   |   | O
---------------------
4   |   |   |   |

My turn:
Minimax called:  272  times recursively.
  0 | 1 | 2 | 3 | 4
0   | X |   |   |
---------------------
1   | O | O | X |
---------------------
2   | X | X | O |
---------------------
3   | X |   |   | O
---------------------
4   |   | O |   |
```

Figure 28. CPU plays O at "r=4, c=2"

The majority of the games end as the Figure 29.

Finally, the last method used to reduce complexity is to keep only the 10 best moves as maximizing player and all the moves of min player. Searching in depth 4 is a bit computational expensive but with descent results.



```
Your turn:
Choose a valid move: 34
Minimax called:  129326  times recursively.
  0 | 1 | 2 | 3 | 4
0   | O |   |   |
---------------------
1   |   | O | X |
---------------------
2   | X | X | O |
---------------------
3   |   |   |   | X
---------------------
4   |   |   |   |

My turn:
Minimax called:  83880  times recursively.
  0 | 1 | 2 | 3 | 4
0   | O |   |   |
---------------------
1   |   | O | X |
---------------------
2   | X | X | O |
---------------------
3   | O |   |   | X
---------------------
4   |   |   |   |
```

Figure 29. Great move for CPU!

FURTHER EXPANSION

Minimax algorithm dominates strong solved games with some modifications. For domains with big complexity minimax can think ahead only few moves so it needs effective pruning. In this report we took advantage of the symmetries of the board and a custom middle-game evaluation and the fact that some moves are stronger than others which should be discarded.

This method faces serious problems and can be tricked by humans easily. It keeps neglects the significance of two in a row, and as soon it gets three in a row loses.

Discarding moves that cannot form four in a row (winning condition) is not a solution as it will discard a possible defending position.

A neural network might perform well in selecting a move which maximizes the probabilities to win but with a research on the web anyone can see that neural networks do not perform better than minimax neither on the classical board.

REFERENCES:

[1] Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 2009.

[2]http://www.se16.info/hgb/tictactoe.htm