UNBEATABLE TIC TAC TOE GAME

PROJECT REPORT

NEURAL NETWORK AND FUZZY CONTROL

Submitted By:

Gaurav Mittal (17BEE02227)

Arijeet Bhattacharya (17BEE0238)

Under the guidance of

PROF. SHARMILA A (SELECT)

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

ABSTRACT

Tic Tac Toe is a well-known game played across all age groups. It's a simple fun game played with two players. There is a 3x3 grid formed by two vertical and two horizontal lines. The user can fill these nine places with either crosses X or noughts O. The aim of the game is to win by placing three similar marks in a horizontal, vertical, or diagonal row. In this paper, a simulation algorithm is presented to predict the win, or draw of a game by knowing the first move of a player. The game of tic-tac-toe is simulated using Min-max algorithm. The concept of combinational game theory is utilized to implement this game. This algorithm calculates only one step ahead using min-max algorithm In an ideal scenario, a player must calculate all the possibilities to ensure the success. It's a small 3x3 game, so the state space tree generated will be short.

1. INTRODUTION

The tic-tac-toe game is well known simple game. The game is played across all the age groups. There is a 3x3 grid formed by two vertical and two horizontal lines. The user can fill these nine places with either crosses X or noughts O. The aim of the game is to win by placing three similar marks in a horizontal, vertical, or diagonal row. It is so simple that one can easily predict the win or draw. This needs graph theory or combinational game theory, a branch of mathematics that will help to predict the different outcomes of the event. The tic-tac-toe game can be considered as a tree with a root node as the initial state of the game. The child will be the corresponding future possibilities of states. Likewise, this tree can be expanded to cover all the possibilities of outcome of the game. Once the tree is constructed, it is easy to evaluate the next optimal move. The grid is formed by two horizontal and two vertical lines, making it a 3×3 grid. There are nine places that can be filled with noughts or crosses or empty position Hence, there are 3919,683 possible states Each place is unique. The aim of filling the nine spaces can be considered as filling the sequence of nine boxes [1-3]. Therefore, there are 9 = 362,880 ways to fill the9th position. The game can be converted to a huge tree of maximum of 19,683 nodes and 362,880 edges. However, most of the states are just the reflections. Therefore, these can be eliminated. There are only three types of moves initially, namely – Corner, Edge and Centre. As shown in figure 1, the positions 0,2,6,8 are called ‘Corner’, 1,3,5,7 are called ‘Edge’ and position 4 is called ‘centre’ position.

In [4] number of ways to completely fill the noughts and crosses board with 5Xs and 4 Os not including rotations and reflections is evaluated. Its little complicated as it makes use of Group Theory.

2. LITERATURE REVIEW

2.1 Tic-Tac-Toe Game

Tic-Tac-Toe is an interesting two player game in which each player takes a turn by marking the spaces in a 3x3 grid. Normally, these spaces are marked by the symbols

‘X’ and ‘0’ where X represents player one and 0 represents player 2 or vice versa. Tic-Tac-Toe produce results of three types which can be a “win”, “loss” or a “draw”. The player who succeeds placing three respective marks in a horizontal, vertical or diagonal row/column wins the game. There is both feasible and infeasible game states associated with this game.

There are certain rules by means of which we determine these states. Those conditions have been explicitly explained in the paper which are as follows [11]:

(a) The number of ‘X’ in any given state would be one more than or equal to number of ‘0’ where we assume X is played first.

(b) Both players would not move if the other has won.

These conditions play a vital part in determining the feasible game states and thereby eliminating the infeasible game states.

X	X	X
X	X	O
O	X	X

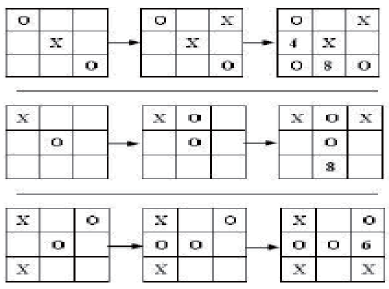
X	X	X
O	O	O

X	X	X
O		O
O		

Infeasible Game States [11]

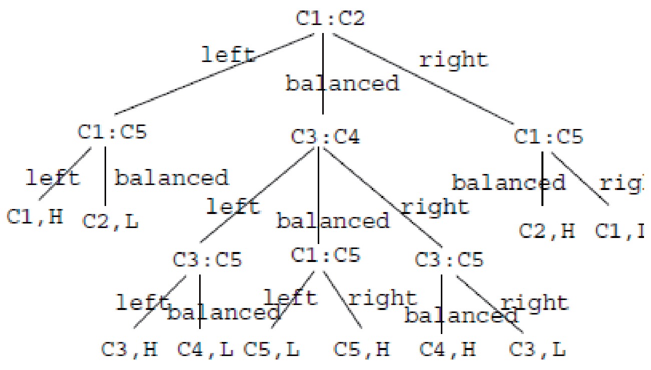
2.2 Prior works

We would like to highlight some of the major contributions made by researchers in evolving “Strategies” for the game of Tic Tac Toe. The earlier work done by the people Hochmuth [13] and Soedarmadji[14] seems to be not clear and effective. Soedarmadji suggested a decentralized decision making to find a competent strategy which forces a draw or a win depending on the proficiency of the opponent player. The paper[11], provides a clear evidence that Soedarmadji's [14] No Loss Strategy fails in three different scenarios and evolves a number of No-Loss Strategies in comparison with existing methodologies. The three different scenarios are explained by means of the diagram shown below:



Three cases in which Soedarmadji's Heuristic solution loses the game. X is played by heuristic solution

The work done by Hochmuth[13] reported a single No Loss Strategy but the study failed to explain the strategy in detail. The efficiency of parallel Minimax algorithm is discussed in the paper [12] . This paper brings out the pros and cons of Minimax algorithm and also explains an optimization strategy for the same using Alpha, Beta Cutoff. Performance comparison has been made for hybrid (multilevel,) flat and multithreaded parallel programming models. Speedup and efficiency as well as scalability in respect to size of the multi-computer and its impact on the performance of the parallel system have been estimated on the basis of experimental results. The communication computation ratio (CCR) of the parallel hybrid and flat implementations of the Minimax algorithm has been estimated. We also took a clear look of the *-Minimax performance with respect the game. 'Backgammon' by Thomas Hauk [15]. The paper provides the first performance results for Ballard's*-Minimax algorithms applied to real-world domain. The paper also presents empirical evidence that with today's sophisticated evaluation functions; good checker play in backgammon does not require deep searches. The resource taken from North Western University [16] explains the decision tree in detail [16] describes decision tree with the game 5-coin puzzle and explicitly shows the game trees associated. The diagram below explains the decision tree for the game of 5-coin puzzle:



Decision tree for the 5-coin puzzle [17]

3. PROPOSED WORK

In this section, we would like to describe the methodologies we used for developing and comparing the search algorithms (Minimax) specific to the game Tic Tac Toe.

4. ALGORITHM FOR TIC-TAC-TOE GAME

A cross 'X' in any of the 0,2,6,8 corner places is the best first move The opponent player will choose from the one of the vacant places. If a 'X' is placed at centre at position 4, then check for corner places 0, 2, 6, and 8and select the vacant position such that it is in accordance with the previous move This is a case of confirmed win. In case the place 4 is not empty then place the cross anywhere in 0,2,6,8 such that it is in accordance with the previous move. The opponent will try to block this move and insert in between so as to avoid the player's win. Then further move can be calculated as discussed in algorithm section

4.1 Algorithm

Min-max is a decision making algorithm which uses decision theory, game theory, statistics and philosophy to calculate the optimal move. It is a two player game. The mechanism evaluates minimum lose and maximum profit [5-7] .This logic can also be extended to play more complicated game like chess, checkers etc In the tic-tac-toe game, a player tries to ensure two cases:

• Maximize a player's own chances of win
• Minimize opponent's chances of win

The Minimax algorithm consists of five steps [17]:
Step 1: Generate the whole game tree, all the way down to the terminal states
Step 2: Apply the utility function to each terminal state to get it's value.
Step 3: Use the utility of the terminal states to determine the utility of the nodes one level higher up in the search tree.
Step 4: Continue backing up the values from the leaf nodes towards the root, one layer at a time.
Step 5: Eventually, the backed up values reach the top of the tree; at that point, MAX chooses the move that leads to the highest value.

4.2 Maximize profit

The profit can be maximized by either fork or win Fork Initially player will create opportunity where he can win in two ways Win If there two same X or O in a row, then play the third to get three in a row *Minimize Loss.* The loss can be minimized by a block. If two 'x' or 'o' of the opponent are in a row then block it, or else block opponents fork

5. FLOWCHART

The flowchart for tic-tac-toe game is presented in figure 3. The first move is either by a computer or a user.Here user is granted the first moveIt is checked if the winning position is created. If any of the players is winning, then the game terminates If no player is winning, then it is checked if there are two crosses or noughts. After 4th move, there is a case where wining cases are reduced. That is why there is a need to evaluate the step that can help player 1 to win, so the module now checks the wining or winning like situation is generatedIf the winning situation is not generated, then the step which further lead to win is evaluated in block 6And if there is no such placement so that there can be win in next step then opponents win is checkedin block 8. And if there no position of winning of opponent then a simple move is made in order to make two in a row in block 10. This is like creating a win-like situation. Now moving back to the block 8 if there is a winning like position of opponent then opponent win is forked in block 9. Then if block 6 detects the winning like situation that is 2 in a row, column or diagonally, and third one is vacant then that third is placed in block 7 and further control goes to block 5 that is to check the win.

5.1 Tree structure sample space

Figure 4 shows the minimal tree which can be used to evaluate the optimal movesThe min max algorithm uses this tree to evaluate the optimal moves ie if there is no further wining position in that sub tree then that sub tree is removed

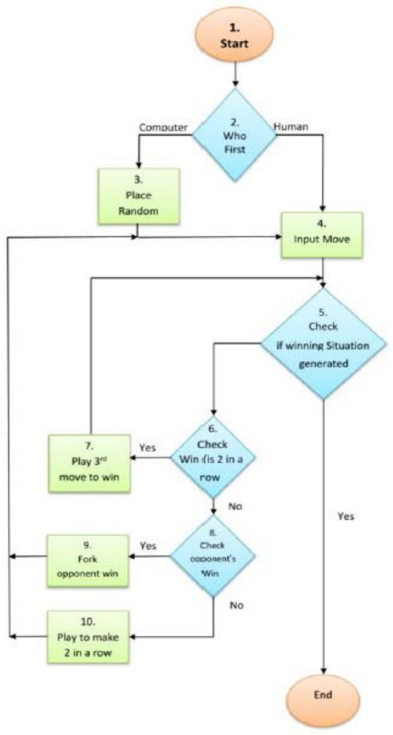


Fig.3. flowchart for tic-tac-toe game

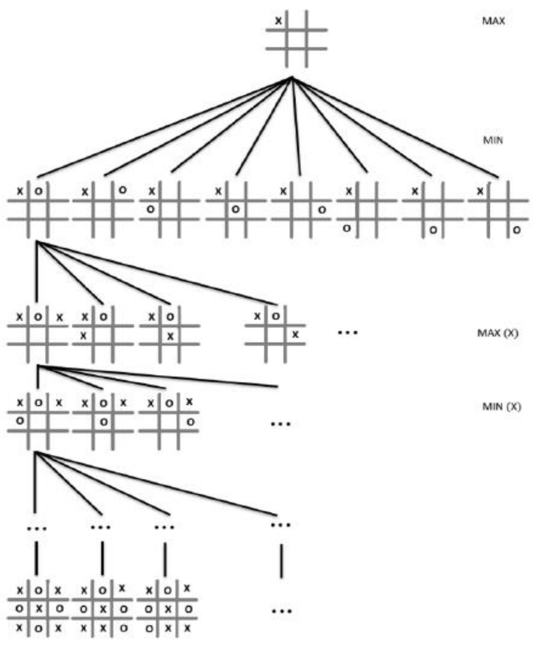


Fig.4. Reduction of sample tree using min- max algorithm

5.2 Logic used by simulator

The computer sees the board positions as numbers as shown in figure

0	1	2
3	4	5
6	7	8

There are two possibilities, as any one of the two players can start first [8-10]. Firstly, consider the case when the opponent starts first, then there are following cases that can be generated:
1. Opponent fills centre position 4
2. Opponent fills side position 1,3,5,7
3. Opponent fills corner position 0,2,6,8

Considering the case that 1st player chooses middle square. Then the middle row, middle column and the two diagonals are booked. Therefore, the other player is left with the

option of 1st and 3rd row and column Out of these four options, some will be again blocked in future moves by the opponent. Now this player has following two possibilities:
a. Filling the corner position
b. Filling edge position

Figure 2a and b shows the case when the opponent places a nought in the centre place and the other player places a cross at either position 0 or 8. The player books one row and one column by filling the corner positions. In figure 2 c and d, the player books an edge square, so in this case either one row or one column is booked. However, in both these cases there exists a case to win, so a random move can be chosen

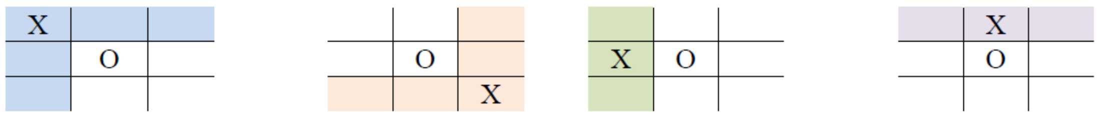


Fig.2. (a), (b), (c) and (d): Places booked by player when the opponent choses centre place

Tree structure sample space
Figure 4shows the minimal tree which can be used to evaluate the optimal moves. The min max algorithm uses this tree to evaluate the optimal moves ie if there is no further wining position in that sub tree then that sub tree is removed

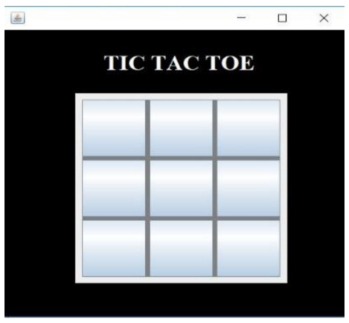
6. SIMULATION RESULTS

This simulation of tic tac toe needs input from mouse click on board as shown in figure 1, here to book positions 0,1,2,3,4,5,6,7 and 8;
Here,
“GRAY”—“X”
“BLACK”—“O”
The two cases of the simulator outputs are discussed in next section. First case involves when there is a draw and in second case the player loses.

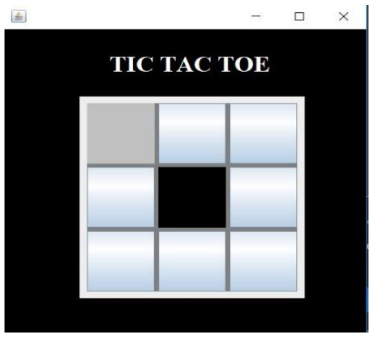
Case I

Initially player is offered the first move Hence the player can choose any position as shown in figure 5a. To choose top left corner, mouse click on top left corner. As the algorithm has nothing to evaluate, so it places ‘O’ in the centre, as shown in figure 5b

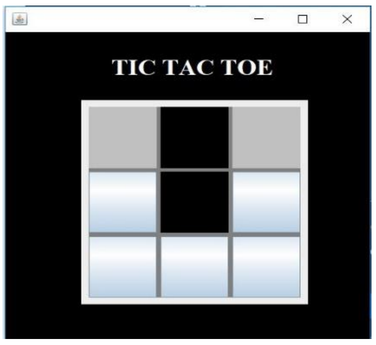
Player again enters another corner. Now algorithm calculates the min profit of ‘X’ player and hence place ‘O’ in between as shown in figure 5c . Then player puts a ‘X’ so that the opponent player doesn’t win in next move. The algorithm calculates max profit in the next step and places a ‘O’ at the position accordingly. This is shown in figure 5c. Thus, leading to a draw. This is as shown in figure 5d.



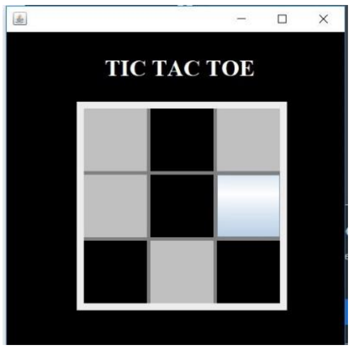
5(a)



5(b)



5(c)



5(d)

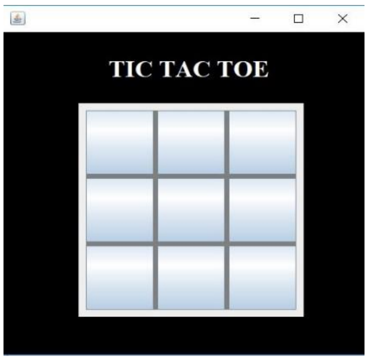
Case II

This is the case which will result in player's loss. In this case player plays the first move by placing X in position '0'. The algorithm plays the random move i.e. places a 'O' in the middle (as shown in figure 6b).After the player plays 2ndmove and places a 'X' in middle row, first column i.e. position '3', the game moves to a state where there is a less possibility of winning of the player. Hence, it calculates a

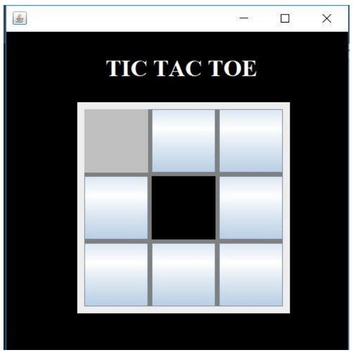
step that is optimum or winning. However, there is no winning step, so computer plays the step to block the player's winning move. So, it places a 'O' at position '6'. To block the computer from winning, the player places a 'X' at top right corner. The computer computes the next optimum move at position '1' and places a 'O' at position '1'. Now the player places a 'X' in middle row. The computer computes its next move at position '7' and places

a 'O' in bottom row middle column. The computer wins this

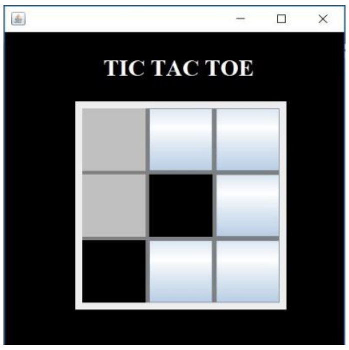
game. These moves are shown in figure 6a-6d



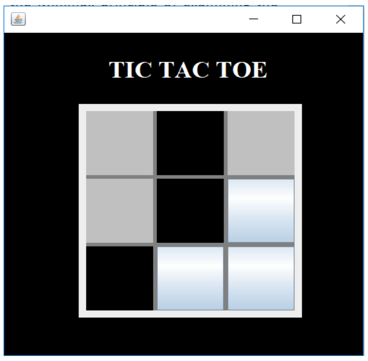
6(a)



6(b)



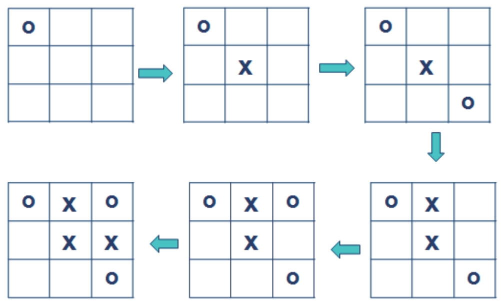
6(c)

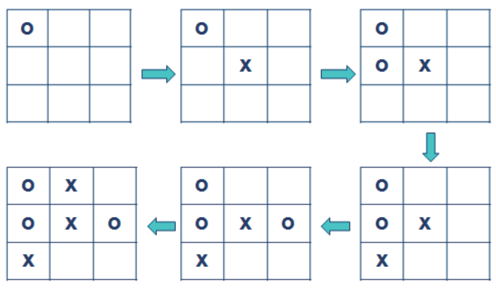


6(d)

7. COMPARISON STUDY

We were able to find out that the Minimax algorithm failed to make an optimal move when the opponent makes a sub optimal move almost in 40 different cases. The logic of the game works perfectly only when both players play the game optimally. It's very interesting that in spite of making a sub optimal move, the algorithm is able to provide a 'No Loss Strategy'. But, the total number of moves in which it achieves the terminal state does not seem to be optimal which can be proved through some of the manual experimentation such as constructing the search trees manually for those selected 'special' cases. The diagram shown below clearly explains about the methodology for constructing the heuristics. We used the same heuristics to calculate and find out the trivial states in which the Minimax failed to make an optimal move against a sub optimal player. Some of the cases we analyzed where the Minimax algorithm didn't perform quiet well against a sub optimal player are:





8. CONCLUSION

The paper presents an algorithm to predict the win cases or draw cases of the tic-tac-toe gameThe algorithm is implemented in c using min max algorithm. The concept of graph theory or combinational game theory is utilized to implement this game. This algorithm calculates only one step ahead using min-max algorithmIn an ideal scenario, a player must calculate all the possibilities to ensure the successTic-tac-toe is a small game hence an unbeatable algorithm can be developed because the state space tree generated will be smallFor future research study, this game algorithm can be extended to simulate other complicated games like chess and checkers However, in case of chess there are 64 squares with two players. This leads to several possibilities.

. An optimized approach to reduce the number of branches to be searched by Minimax search algorithm is to implement Alpha-Beta cutoffs in the Minimax search algorithm. It follows the Minimax principle of examining the cost of a player move in depth of the game tree. This approach can however be used only to improve the efficiency of the algorithm.

9. ACKNOWLEDGEMENT

We thank Mrs. Sharmila A, Professor, School of Electrical Engineering (SELECT) and IEEE Society for giving its valuable information and Data. We would like to thank our VIT University for providing all help and support for our contribution.

10. REFERENCE

1. Brendan O'Donoghue, B., "Min-Max Approximate Dynamic Programming", IEEE Multi-Conference on Systems and Control, pp. 28-30, 2011.

2. Rivest, R.L, "Game Tree Searching by Min / Max Approximation", Artificial Intelligence 12(1), pp. 77-96, 1988.

3. Chen, J.; Wu, I.; Tseng, W.; Lin, B.; and Chang, C. "Job-level alpha-beta search", IEEE Transactions on Computational Intelligence and AI in Games, 2014.

4. httpperfecttictactoeherokuappcom

5. Gelly, S.; Kocsis, L.; Schoenauer, M.; Sebag, M.; Silver, D.; Szepesvári, C.;Teytaud, O.; "The grand challenge of computer Go: Monte Carlo tree search and extensions", Communications of the ACM, 55(3), 2012.

6. Y.Cai and C.Daskalakis, "On minmaxtheorems for multiplayer games", 22ndAnnual ACM-SIAM Symposium on Discrete Algorithms (SODA '11), pp. 217-234, 2011.

7. Daskalakis, C., Papadimitriou, C. H.: ContinuousLocal Search. In: SODA (2011).

8. B. Huang, "Pruning Game Tree by Rollouts", 29th AAAI Conference on Artificial Intelligence, pp. 1165-1173, 2015.

9. S. Gal, "A discrete search game", SIAM Journal of Applied Mathematics 27(4), pp. 641-648, 1974.

10. H. W. Kuhn, Classics in Game Theory, Princeton University Press, 1997.

11. Anurag Bhatt, Pratul Varshney, and Kalyanmoy Deb "In Search of No-loss Strategies for the Game of Tic-Tac-Toe using a Customized Genetic Algorithm" ,Pages 889-896, 2008, Proceedings of the 10th annual conference on Genetic and evolutionary computation, Atlanta, GA, USA.

12. Plamenka Borovska, Milena Lazarova, "Efficiency of parallel minimax algorithm for game tree search", ACM InternationalConference Proceeding Series; Vol. 285 Proceedings of the 2007 international conference on Computer systems Bulgaria Article No. 14 Year of Publication: 2007 ISBN:978-954-9641-50-9 .

13. G. Hochmuth. “On the genetic evolution of a perfect Tic-tac-toe strategy”, pages 75–82. Stanford University Book Store 2003.

14. E. Soedarmadji. Decentralized decision making in the game of tictac-toe. In Proceedings of the 2006 IEEE Symposium on Computational Intelligence and Games, pages 34–38,2005.
15. An article take from the resource provided by the North Western University, USA www.math.northwestern.edu/~mlerma/courses/cs310-04w/notes/dm-dectrees.pdf

16. Thomas Hauk, Michael Buro and Jonathan Schaeffer, “*-Minimax Performance in Backgammon”, Computer and Games Conference, Ramat-Gan 2004, pg 61-66

17. Russell Stuart J., Norvig Peter. “Artificial Intelligence. A modern approach.”. Prentice Hall, 1995. ISBN 0-13-103805-2.

18. An Article from the resource provided by the University of California,Berkeley,USA,http://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.htm

19. An article from the lecture notes provided by the University of California, Irvine, USA, www.ics.uci.edu/~dechter/courses/ics-271/fall- 08/lecture-notes/6.games.ppt

20. Sivaraman Sriram, Rajkumar Vijayarangan, Saaisree Raghuraman, Xiaobu Yuan*, Senior Member, IEEE, 2009. Implementing a No-Loss State in the Game of Tic-Tac-Toe using a Customized Decision Tree Algorithm, International Conference on Information and Automation June 22 -25, 2009, Zhuhai/Macau, China

21. Deva Prasad Nayak UIET, Panjab University, Chandigarh, India, Volume 8, No. 7, July – August 2017. GAME OF TIC-TAC-TOE: SIMULATION USING MIN-MAX ALGORITHM, International Journal of Advanced Research in Computer Science