DOI: http://dx.doi.org/10.26483/ijarcs.v8i7.4409

ISSN No. 0976-5697

Volume 8, No. 7, July – August 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

GAME OF TIC-TAC-TOE: SIMULATION USING MIN-MAX ALGORITHM

Roopali Garg
UIET, Panjab University,
Chandigarh,India

Deva Prasad Nayak
UIET, Panjab University,
Chandigarh,India

***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**.**

Keywords: Decision theory, Game theory, Group theory, Min-max algorithm

1. INTRODUCTION

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 branchof 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 3^9=19,683 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 the9^{th} 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. Further these positions are represented corresponding to qwerty keyboard inputs: q,w,e,a,s,d,z,x,c.

0	1	2
3	4	5
6	7	8

·

q	w	e
a	s	d
z	x	c

Fig.1. Board position corresponding to keyboard input.

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. It's little complicated as it makes use of Group Theory. The rest of the paper is organized as follows: Sections II discusses the algorithm for the tic-tac-toe game. The flowchart for the same is presented in section III. The simulation results are described in section IV. Finally the paper is concluded in next section and the references are appended at the end.

II. 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.

2.1. 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

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 twosame '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.
Block If two 'x' or 'o' of the opponent are in a row then block it, or else block opponent's fork.

2.2. Logic used by simulator

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 fouroptions, 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 2(a) 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.

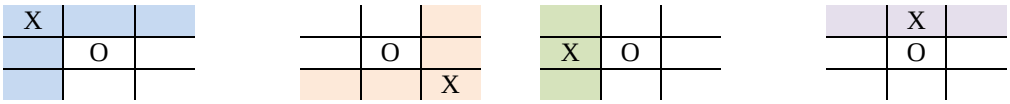

Fig.2. (a), (b), (**c**) **and** (**d**): Places booked by player when the opponent choses centre place

III. Flow chart

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 move. It 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 generated. If the winning situation is not generated, then the step which further lead to win is evaluated in block 6. And if there is no such placement so that there can be win in next step then opponent's 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.

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 i.e if there is no further wining position in that sub tree then that sub tree is removed.

IV. SIMULATION RESULTS

This simulation of tic tac toe needs input from QWERTY key board as shown in figure 1, here to book positions 0,1,2,3,4,5,6,7 and 8; keys q, w, e, a, s, d, z, x and c are used respectively.

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, key 'q' is pressed. As the algorithm has nothing to evaluate, so it places 'O' in the centre, as shown in figure 5b.

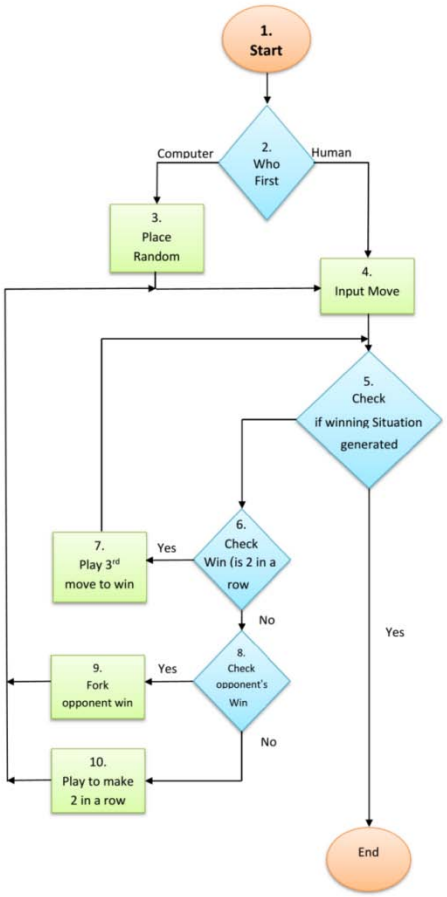

Fig.3. flowchart for tic-tac-toe game

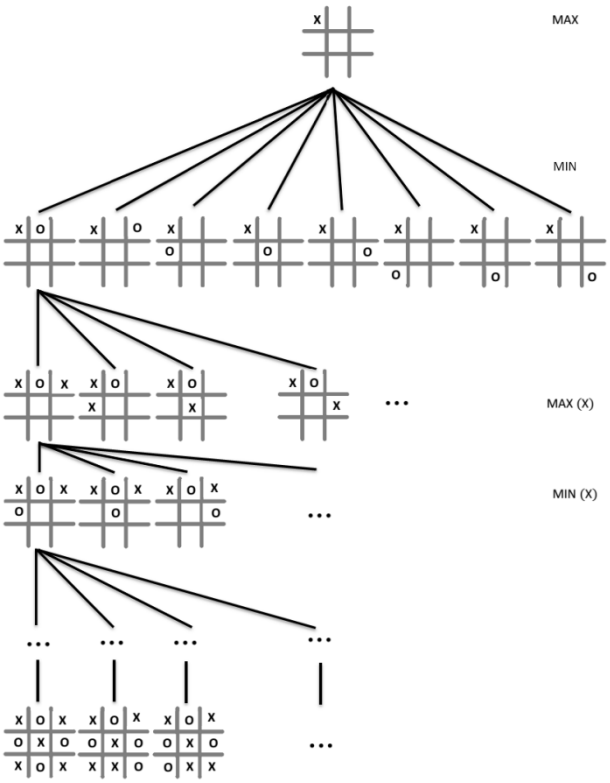

Fig.4. Reduction of sample tree using min- max algorithm

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 'a'. This is shown in figure 5d. Thus, leading to a draw. This is as shown in figure 5e.

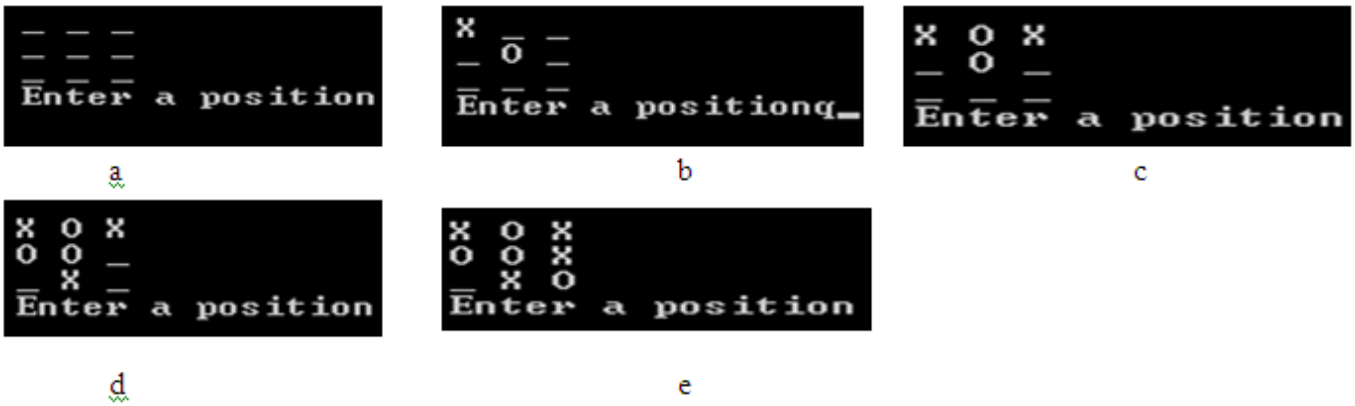

Fig.5. a)Enter the position; b)Player choses top-left corner to place cross; c)Player again enters another corner; d) player puts a 'X' to block the opponent; e) A draw

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 'q'. The algorithm plays the random move i.e. places a 'O' in the middle (as shown in figure 6a and 6b).After the player plays 2nd move and places a 'X' in middle row, first column i.e. position 'a', 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 'Z'. 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 'w'. 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-6h

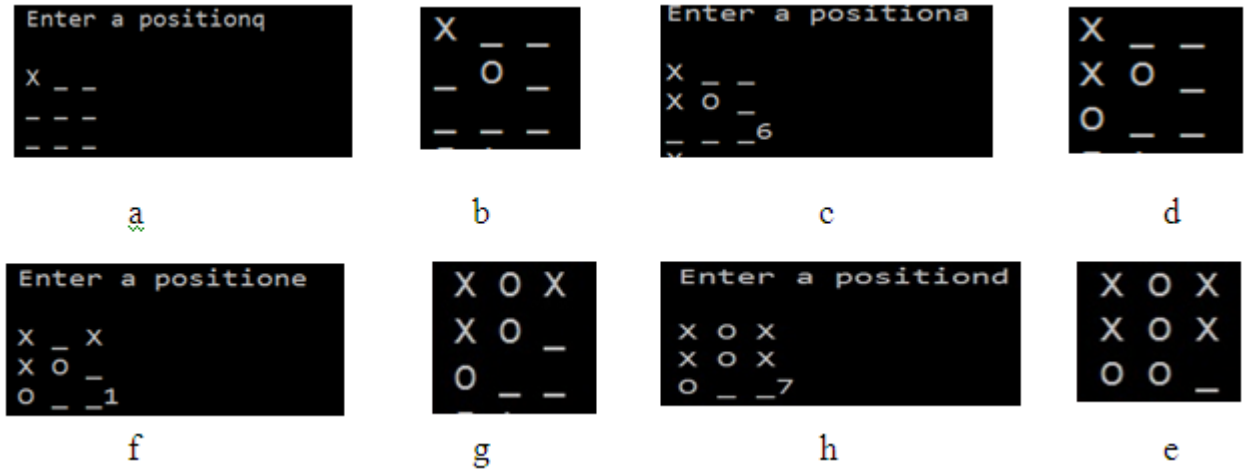

Fig.6. a)Players lead move; b)Computer’s 1st move; c)Player choses position ‘a’; d) Computer’s 2nd move; e) Player choses position ‘e’; f) Computer choses position ‘1’; g) Player choses middle row; h) Computer wins

IV. CONCLUSION

The paper presents an algorithm to predict the win cases or draw cases of the tic-tac-toe game. The 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 algorithm. In an ideal scenario, a player must calculate all the possibilities to ensure the success. Tic-tac-toe is a small game hence an unbeatable algorithm can be developed because the state space tree generated will be small. For 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.

REFERENCES

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. http://perfecttictactoe.herokuapp.com/
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", 22nd Annual 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.