TIC TAC TOE GAME BETWEEN COMPUTERS: A COMPUTATIONAL INTELIGENCE APPROACH

Ashutosh Kumar Sahu, Parthasarathi Palita*, Anupam Mohanty

asu.ku.sahu@gmail.com parthapalita107@gmail.com anupam.iter@gmail.com

Last Year B.Tech Students, Dept. of Electronics & Instrumentation Engineering

ITER, Siksha O' Anusandhan University,

Bhubaneswar, Odisha, India

Abstract : Computational intelligence is a powerful tool for game development. In this paper, an algorithm of playing the game Tic-Tac-Toe with computational intelligence is developed.In the present work two computers are suppose to play a Tic tac toe game between them using artificial intelligence technique. Direct Cable Connection (DCC) connection was taken as interfacing is required for playing tic tac toe game between two computers. The present workers were motivated to design a setup where two computers can play Tic tac toe with each other using their respective Personal Computers(PCs) by the help of microcontroller. The objectives of the work includes developing an algorithm for Tic tac toe; interfacing hardware with the two PCs and to design a real world setup for tic tac toe game using LED array / robotic arms. The algorithm proposed in this paper is fuzzy based and tested for the game. The further development of the work is related to hardware interface.

INTRODUCTION

By playing games, the machine intelligence can be revealed. For knowledge-based methods, the best move is determined by searching a game tree. For games such as Checkers, the tree spanning is very large. Tree searching will be time consuming even for a few plies. Hence, an efficient searching algorithm is an important issue. The problems are solved by forming a possible set of solutions based on the endgame condition, or searching for the set of solutions based on the current game condition. The machine cannot learn to play the games by itself. Unlike an evolutionary approach was employed to evolve and to learn for playing Tic-Tac-Toe without the need of a database. Artificial intelligence (AI) is the intelligence of machines and the branch of computer science that aims to create it. Tic-tac-toe is a pencil-and-paper game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The X player usually goes first. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game. Here using artificial intelligence two computers are suppose to play a Tic tac toe game between them. Here present workers developed an algorithm for Tic tac toe using minmax algorithm with alpha beta concept**.** As interfacing is needed for playing tic tac toe game between two computers ,so we are taking Serial and parallel (This type of cabling, called *Direct Cable Connection (DCC).*And this connection will pass through a microcontroller kit , which will have robot arm to draw 'X' or '0' on the drawing board.

RELATED LITERATURE

1.In "Playing a tic-tac-toe game against a Robot manipulator using a vision system", authors work deals with the integration of Robotics, Artificial Intelligence and Image Processing. They have explained to do the same, an experimental setup is to be assembled, composed by a CCD black and white camera, a frame grabber with 8 bits resolution and a didactic robot kitRobix RCS-6. The main idea is to implement the tic-tac-toe game where the robot manipulator plays against a human opponent. The camera scene is composed by the tic-tac-toe board and the robot can "see" it using Image Processing techniques. The board's image is acquired repeatedly and an image subtraction between two simultaneous images is done. The threshold of the resultant image is calculated and a binary image is obtained. The centroid of the remain figure is calculated and used to allows the robot decides in which board cell it should play to block its opponent. Once this decision is taken, the robot, assembled in a two degree of freedom configuration, draws an X on the game board. Algorithms for threshold calculus and the robot inverse kinematics are discussed. The scene illumination and problems that it addresses are also discussed. The program, developed in C++ language, works quite well and illustrates how those areas

(Robotics, Image Processing and Artificial Intelligence) can work together.

2. In “reinforcement learning in tic-tac-toe game and its similar variations” authors experiments of two games implemented in C++ demonstrate success of the applications of reinforcement learning to in the board games.
Board games, which could be as simple as Tic-Tac-Toe, are where human wisdom has shined since a long time ago and are what human intelligence is trying to implement for machine intelligence for decades. Many of machine learning techniques have been explored, exploited, developed and argued extensively as researchers work on these canonical problems. Perhaps, “minimax search” is seemingly the most successful one among many for it and its effective variations balance the searching efficiency and computational complexity, not to mention the glory of “Deep Blue” playing against Chess Master Kasparov. However, in common sense, minimax search is nothing but an enforced version of brute force search while it has a weak ability of automatically evaluating the board situation (or we call “state”). This makes it trivial by hand coding initially and also, sometimes, impossible to be applied. Researchers had been intensively interested in finding a method of evaluation the board state in the last century and maybe still today. In early 1990’s, Sutton and Barto systematically developed an unsupervised learning method-reinforcement learning; Watkins proposed an important online implementation called Q-learning and proved its convergence, making the online technique work powerfully. If applied to games, reinforcement learning only needs the values of final states, which are easy to determine. For example, Tesauro utilized this method and succeeded in the solution of Gammon game.

3. In “Evolution of no-loss strategies for the game of tic-tac-toe” work, authors have applied a genetic algorithm for evolving no-loss strategies for the game of Tic-tac-toe. First, authors have described a scheme for representing a strategy in a GA and discussed the initialization scheme and GA operators for the minimization of proportion of losses in all possible game scenarios. By carefully designing the GA procedure, authors have been able to find not only one, but as many as 72,657 different strategies which will never lose a game.
The game of Tic-tac-toe is one of the most commonly known games. This game does not allow one to win all the time and a significant proportion of games played results in a draw. Thus, the best a player can hope is to not lose the game. This study is aimed at evolving a number of no-loss strategies using genetic algorithms and comparing them with existing methodologies. To efficiently evolve no-loss strategies, authors have developed innovative ways of representing and evaluating a solution, initializing the GA population, developing GA operators including an elite preserving scheme. Interestingly, their GA implementation is able to find more than 72 thousand no-loss strategies for playing the game. Moreover, an analysis of these solutions has given us insights about how to play the game to not lose it. Based on this experience, authors have developed specialized efficient strategies having a high win-to-draw ratio. The study and its results are interesting and can be encouraging for the techniques to be applied to other board games for finding efficient strategies.

MOTIVATION AND OBJECTIVE

A lot of work on the game has approached in many ways. As per the literature survey, it was observed that the work done on Tic tac toe problem were based on user vs. computer or comp vs. comp in a single PC. The various approaches were based on Microcontroller based, Artificial Intelligence based, and also Image Processing based.

In our work we propose the method of Embedded System based. Some of the work also carried out by the authors, but still there is scope to work in this area. Here the objective is to design such a setup where two computers can play Tic tac toe with each other using their respective PCs controlled by Embedded processor. It will be based on Real Time System.

For the successful performance of the setup the following steps are mandatory.

Developing an algorithm for Tic tac toe

Interfacing unit among two computers.

To design a real world setup for tic tac toe game using VHDL programming.

PROPOSED ALGORITHM FOR GAME DESIGN

The following techniques are tested and designed for user vs. computer on

Random selection algorithm

Minimax algorithm with alpha beta concept

The above individual techniques were then merged to a single form for user vs. computer having different difficulty levels. Then, it is modified for computer vs. computer gaming in single PC having 3 modes of difficulties.

A. VERY HARD MODE:

In this mode the objective is to put the values in the best position. For this MINMAX algorithm is followed.
As far it is the most effective algorithm for dual games such as Chess, Tic tac toe. In this mode of the game the probability of computer of losing is 0.

Using Heuristics in Games : Games are an important test-bed for heuristic algorithms. Two-person games are more complicated than a simple puzzle because they involve an unpredictable opponent

Minimax Procedure
The Game of Nim: A number of tokens are placed on a table between two opponents. At each move the player must divide the pile of tokens into two nonempty piles of different sizes. Thus, 6 tokens may be divided into 5 and 1, 4 and 2, but not 3 and 3. The first player who is unable to make a move loses the game. For a small number of tokens the search space can be searched exhaustively. The next figure gives the complete space for a 7-token game.

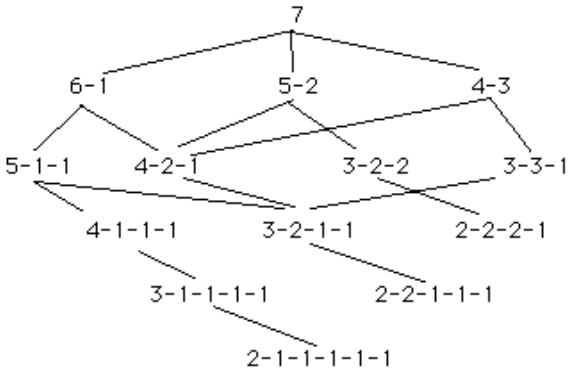

Fig. 1. Search graph of 7-token game

In a two-person game, you must assume that your opponent has the same knowledge that you do and applies it as well as you do. So at each stage of the game you must assume your opponent makes the best available move. This is the basis of the minimax procedure. In minimax, the players are referred to as MAX (the player) and MIN (the opponent). Both try to maximize their moves. MAX is the player, trying to MAXimize his score. And MIN is the opponent trying to MINimize MAX's score.

Minimax Procedure on a Complete Search Space

1. Label each level of the search space according to whose move it is at that level.
2. Starting at the leaf nodes, label each leaf node with a 1 or 0 depending on whether it is a win for MAX (1) or MIN (0).
3. Propagate upwards: if the parent state is MAX, give it the MAX of its children.
4. Propagate upwards: if the parent state is MIN, give it the MIN of its children.

Consider the minimax graph for the game of Nim. The value at each state represents the value of the best state that this player can hope to achieve. The derived values are used to choose among alternative moves.

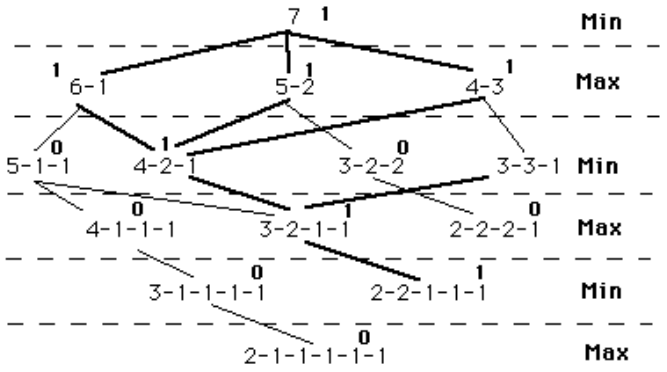

Fig. 2. Minimax search graph

Some of the questions below make the understanding by analyzing the graph for application in this application. That leads towards the MINIMAX procedure.

- Who makes the first move in this particular game?
- Under what circumstances will MIN win the game?

- Suppose MAX is in the middle state (5-2) on her first move. What move does minimax recommend?
- How good are MAX's prospects in this game?

Heuristic Minimax

For most games it is impossible to expand the graph to its leaf nodes. Instead an n-move look-ahead strategy is used. The state space is expanded to n levels. Each leaf node in this sub graph is given a value according to the heuristic evaluation function. Values then are propagated back to the root node. The value propagated back represents the heuristic value of the best state that can be reached from that node.

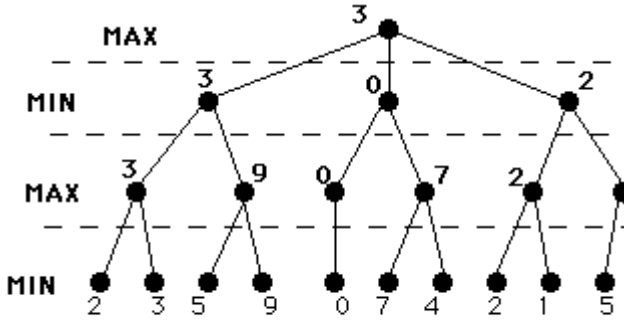

Fig. 3. Hypothetical search space graph

The Alpha-Beta Procedure

Alpha-beta pruning is a procedure to reduce the amount of computation and searching during minimax. Minimax is a two-pass search, one pass is used to assign heuristic values to the nodes at the ply depth and the second is used to propagate the values up the tree.

Alpha-beta search proceeds in a depth-first fashion. An alpha value is an initial or temporary value associated with a MAX node. Because MAX nodes are given the maximum value among their children, an alpha value can never decrease; it can only go up. A beta value is an initial or temporary value associated with a MIN node. Because MIN nodes are given the minimum value among their children, abeta value can never increase; it can only go down.

For example, suppose a MAX node's alpha = 6. Then the search needn't consider any branches emanating from a MIN descendant that has a beta value that is less-than-or-equal to 6. So if you know that a MAX node has an alpha of 6, and you know that one of its MIN descendants has a beta that is less than or equal to 6, you needn't search any further below that MIN node. This is called alpha pruning. The reason is that no matter what happens below that MIN node, it cannot take on a value that is greater than 6. So its value cannot be propagated up to its MAX (alpha) parent. Similarly, if a MIN node's beta value = 6, you needn't search any further below a descendant MAX that has acquired an alpha value of 6 or more. This is called beta pruning. The reason again is that no matter what happens below that MAX node, it cannot take on a value that is less than 6. So its value cannot be propagated up to its MIN (beta) parent.

Rules for Alpha-beta Pruning

- Alpha Pruning: Search can be stopped below any MIN node having a beta value less than or equal to the alpha value of any of its MAX ancestors.
- Beta Pruning: Search can be stopped below any MAX node having a alpha value greater than or equal to the beta value of any of its MIN ancestors.

Example: A left-to-right alpha-beta prune of the hypothetical minimax search space:

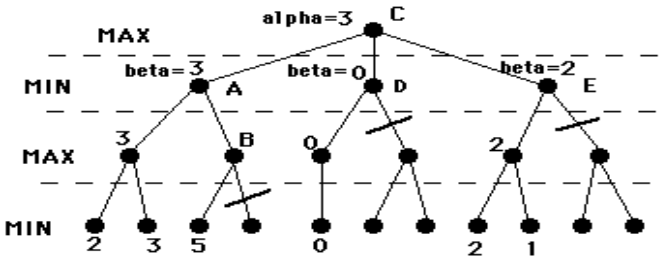

Fig. 4. Search graph of Alpha-Beta Concept

B. HARD MODE:

In hard mode the motivation of the algorithm is to first check the possibilities of win. If any possibility exists then put the value and end the game. If there is no possibility then check the opponents move and if the opponent has a chance of winning in the next move then try to block it. else put the value in random position.

Here the code is designed for both Offence and Defence.

In this mode the probability of losing is more than that of VERY HARD mode because it is not effective against all the advanced tricks.

C. EASY MODE:

In this mode the computer is going to put the value randomly in any position if that position is not filled up.

The probability of losing is highest among all the 3 modes of difficulties.

P (winning of Very Hard) > P (winning of Hard)>P (winning of Easy).

Conclusion

Intelligence can be a property of *any* purpose-driven decision maker. This basic idea has been suggested many times. An algorithm of playing Tic-Tac-Toe has been presented and tested that works in efficient way. It must apply equally well to humans, colonies of ants, robots, social groups, and so forth.

Acknowledgement : We are thankful to *Prof. Mihir Narayan Mohanty*, H.O.D., Deptt. of EIE as our project supervisor. His sincere effort and encouragement shows a new direction in technical education and our improvement.

REFERENCES :

[1] Alvaro Monoef de Souza soares,"Playing a tic-tac toe game against a robot manipulator using vision system", Rev.clenc.exatas.taubate, vol.12, no.1, pp.123-128, 2006.

[2] CornelisJ.Franken,"PSO-based coevolutionary game learning",University of Pretoria etd – Franken C J (2004).

[3] Peng Ding and Tao Mao ,"Reinforcement Learning in Tic-Tac-Toe Game and Its Similar Variations" , Thayer School of Engineering at Dartmouth College

[4] R. Sutton and A. Barto. Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA: pp. 10-15, 156. 1998.

[5] Anurag Bhatt, Pratul Varshney and Kalyanmoy Dev, "Evolution of No-loss Strategies for the Game of Tic-Tac-Toe", KanGAL Report Number 2007002.

[6] Sebastian Siegel,"Training an artificial neural network to play TIC-TAC-TOE",ECE 539 TERM PROJECT, 2011.