****

**A MINI PROJECT REPORT**

**ON -“TIC TAC TOE”**

Submitted by – T.RENUKA(40020728)

Submitted to GENESIS TEAM~

(G.BARATH & PAGALA PRITHVI SEKHAR)

LEAD-Technical Education-Global Engineering Academy

**L&T Technology Services Limited**

Submission Date: 22 nd November 2021

**Table of Contents :**

CHAPTER 1: INTRODUCTION.................................................................................................... 2

1.1OBJECTIVES……………………………………………………………………………………………..…..2

CHAPTER 2: AGENT.................................................................................................................... 3

2.1 Agent....................................................................................................................3

2.2 Agent Environment Description ..........................................................................3

CHAPTER 3: PROBLEM SPECIFICATION........................................................................................ 4

CHAPTER 4: STATE SPACE REPRESENATION................................................................................ 5

CHAPTER 5: ALGORITHM USED................................................................................................... 6

**5.1**MINIMAL Algorithm**…….................................................................................6**

5.2 Alpha-Beta Pruning ........................................................................................... 6 CHAPTER 6: OUTCOME ANALYSIS................................................................................................. 7

CHAPTER 7:CONCLUSION ............................................................................................................. 8

**CHAPTER1:INTRODUCTION**

Tic-tac-toe also known as noughts and crosses is a paper and pencil game for two players, who take turns marking the spaces in a 3 x 3 grid traditionally. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row wins the game. It is a zero-sum of perfect information game. This means that it is deterministic, with fully observable environments in which two agents act alternately and the utility values at the end of the game are always equal and opposite. Because of the simplicity of tic-tac-toe, it is often used as pedagogical tool in artificial intelligence to deal with searching of game trees. The optimal move for this game can be gained by using minimax algorithm, where the opposition between the utility functions makes the situation adversarial, hence requiring adversarial search supported by minimax algorithm with alpha beta pruning concept in artificial intelligence.

**1.1 OBJECTIVES:**

1. To develop Artificial intelligence-based tic-tac-toe game for human Vs AI by implementing minimax algorithm with adversarial search concept.

2. To Analyse the complexity of minimax algorithm through 4x4 tic tac toe game.

3. To study and implement alpha-beta pruning concept for improved speed of searching the optimal choice in tic-tac toe game.

4. To study optimizing methods for alpha-beta pruning using heuristic evaluation function R

**CHAPTER 2: AGENT**

2.1 Agent An agent perceives its environment through sensors and acts on the environment through actuators. Its behavior is described by its function that maps the percept to action. In tic tac toe there are two agents, the computer system and human. In the AI-based tic-tac toe game the function is mapped using the minimax algorithm alongside alpha beta pruning. The agent can be further described by following factors: Performance Measure: Number of wins or draws. Environment: A 4x4 grid with 16 cells respectively, with opponent (human agent). The cell once occupied by a mark cannot be used again. Task environment is deterministic, fully observable, static, multi-agent, discrete, sequential. Actuators: Display, Mouse (human agent) Sensors: The opponent’s (human agent) input as a mouse pointer on a cell Furthermore, it is a utility-based agent since it uses heuristics and pruning concept as utility function that measures its preferences among the states and chooses the action that leads to the best expected utility i.e. winning or tie condition for the computer system agent. The utility function is taken as +10 for winning situation of maximizing agent or AI, -10 for winning situation of minimizing agent or human player and 0 for draw condition

2 .2 Agent Environment Description The basic environment for tic tac toe game agent is a 4x4 grid with 16 cells respectively with opponent as human agent. The cell once occupied by a mark cannot be used again. The agent has access to complete state of environment at any point and can detect all aspects that are relevant to the choice of action making the environment fully observable. Also, the next state of environment can be completely determined by current state and action executed which means environment is deterministic. It is a two-player game with the opponent as human agent, so it is a multi-agent environment. Beside the game environment has a finite number of states, percept’s and actions, making the environment static and discrete. Also, the environment is sequential since the current choice of cell could affect all future decisions. Thus, the task environment for the tic tac toe system agent is sequential, deterministic, fully observable, static, discrete and a multi-agent.

**CHAPTER 3: PROBLEM SPECIFICATION**

Tic tac toe game has 16 cells for a 4x4 grid. The two players with their respective marks as ‘X’ and ‘O’ are required to place their marks in their turns one by one. Once the cell is occupied by a mark it cannot be used again. The game is won if the agent is able to make a row or column or a diagonal occupied completely with their respective marks. The game terminates once the winning situation is gained or the cells are fully occupied. The problem specification for this game is given below: Problem: Given a 4x4 grid, the agents has to find the optimal cell to fill with respective marks. Goals: To find the optimal cell to fill with respective marks and in order to win the game, the cell must be filled such that one of the following criteria is satisfied:

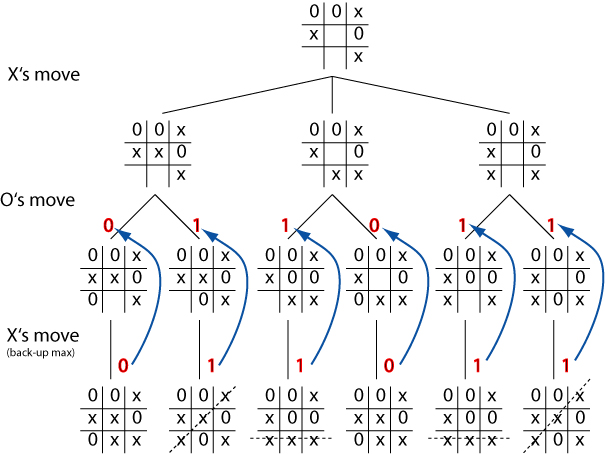
1. A row is completely filled by a mark ‘X’ or ‘O’.

2. A diagonal is completely filled by a mark ‘X’ or ‘O’.

3. A column is completely filled by a mark ‘X’ or ‘O’. If these criteria are not satisfied by Both agents

**Constraints**: 1. Once the cell is occupied by a mark, it cannot be reused. 2. Agents place the mark alternatively. So, consecutive moves from any agent is not allowed.

**CHAPTER 4: STATE SPACE REPRESENATION:**



**CHAPTER 5: ALGORITHM USED**

5.1 Minimax Algorithm Minimax is a recursive algorithm which is used to choose an optimal move for a player assuming that the opponent is also playing optimally. Its objective is to minimize the maximum loss. This algorithm is based on adversarial search technique. In a normal search problem, the optimal solution would be a sequence of actions leading to a goal state. Rather in adversarial search MAX finds the contingent strategy, which specifies the MAX’s moves in the initial state, then MAX’s moves in the states resulting from every possible response by MIN and continues till the termination condition comes alternately. Furthermore, given a choice, MAX prefers to move to a state of maximum value whereas MIN prefers a state of minimum value.

5.2 Alpha-Beta Pruning The problem with minimax search is that the number of game states it has to examine is exponential in the depth of the tree. The minimax algorithm recursively calls itself until any one of the agent wins or the board is full which takes a lot of computation time and makes it impossible to solve the 4X4 grid using standard minimax algorithm. To solve this issue, we can use alpha-beta pruning algorithm which eliminates large parts of the tree from considerations by pruning the tree. When applied to a standard minimax tree, it returns the same move as minimax would, but it prunes away branches that cannot possibly influence the final decision. Basically, this algorithm applies the principle that there is no use expending search time to find out exactly how bad an alternative is if you have a better alternative. Alpha-beta pruning gets its name from the parameters that bound on the backed-up values that appears anywhere along the path:

1.α= the value of the best choices (i.e. highest value) so far at any choice point along the path for MAX

2. β= the value of the best choice (i.e. lowest value) so far at any choice point along the path for MIN

**CHAPTER 6: OUTCOME ANALYSIS**

The minimax algorithm performs a complete depth-first exploration of the game tree. It recursively calls itself until a winning state of any agent is found or the grid is full. If the maximum depth of the tree is m and there are b legal moves at each point, then the time complexity of the minimax algorithm is O (bm). The space complexity is O(bm) for generating all actions at once. For 3X3 board, the possible number of moves is (3 \* 3)! = 9! If the board size is increased to 4X4 there is exponential growth and we experience an explosion of computational time. This means we now get 16! Possible states which is an incredibly large amount. So, we can estimate that computational time will be million times higher than 3X3 board. Hence, the time cost is totally impractical, making this algorithm infeasible in 4X4 board, but it serves as the basis for the mathematical analysis of games and for more practical algorithm. To implement the minimax algorithm for 4X4 board we have applied the concept of alpha beta pruning for eliminating the solutions which would not make impact on the final decision. Considering above parameters the time complexity or alpha-beta pruning is O (bm/2) in best case i.e. depth is reduced to half in the best possible scenario. Even though there is such huge reduction in time, the time complexity is still very large considering our board. So, to decrease the computation time, the search can be limited to certain level i.e. reduce the depth of search tree. The maximum depth is taken as 9 in our system. But when reducing the depth of search tree, the output is not optimal as we have cut-off 9 levels of our search tree which is a very large number of nodes. Hence, we have used heuristic evaluation function that approximates the true utility of a state without doing a complete search. For this problem, the heuristic function used is given by: E(n) = M(n) – O(n) where, M(n) is total of possible winning path of AI O(n) is total of possible winning path of human player E(n) is total evaluation for state n This heuristic gives positive value if AI i.e. maximizing agent has more chance or winning and negative value if the human player i.e. minimizing agent has more chance of winning

**CHAPTER 7: CONCLUSION ]**

With the basis of minimax algorithm for mathematical analysis alongside speeding up the computation by alpha beta pruning concept and optimizing the utility function using heuristic function, the 4x4 tic tac toe game was developed. We explored that a 4x4 tic tac toe, an adversary search technique game in artificial intelligence, can be developed using these techniques. Increasing the size of this game would create a huge time complexity issue with the same algorithm and techniques, for which other logics must be further researched