Artificial Intelligence And

Expert System

ASSIGNMENT # 02

AI Tic-Tac-Toe: A Comparative Study of Minimax vs Alpha-Beta Pruning

NAME	ANAS QUTBI
ROLL #	CR-22037
YEAR	THIRD
COURSE CODE	CT-361

Anas Qutbi CR-22037

Github : TIC-TAC-TOE
Youtube: https://youtu.be/-TpYLQ9ZEKg

Introduction

This assignment explores the application of Artificial Intelligence to the classic game of Tic-Tac-Toe using the Minimax algorithm and its optimization via Alpha-Beta Pruning. The goal is to implement both approaches and compare their performance based on computation time and node expansion.

Objective

- Implement the core logic of Tic-Tac-Toe.
- Develop an AI player using the Minimax algorithm.
- Enhance the algorithm with Alpha-Beta Pruning.
- Compare the performance of both algorithms.
- Allow the user to choose which AI to play against.

Game Logic Overview

The Tic-Tac-Toe board is represented as a 1D list of 9 elements. The game includes methods to:

- Make and validate moves
- Detect a winner or a draw
- Print the current state of the board

Minimax Algorithm

The Minimax algorithm is a recursive decision-making algorithm used in two-player games. It simulates all possible moves up to terminal states and assigns scores to determine the optimal move.

Alpha-Beta Pruning Optimization

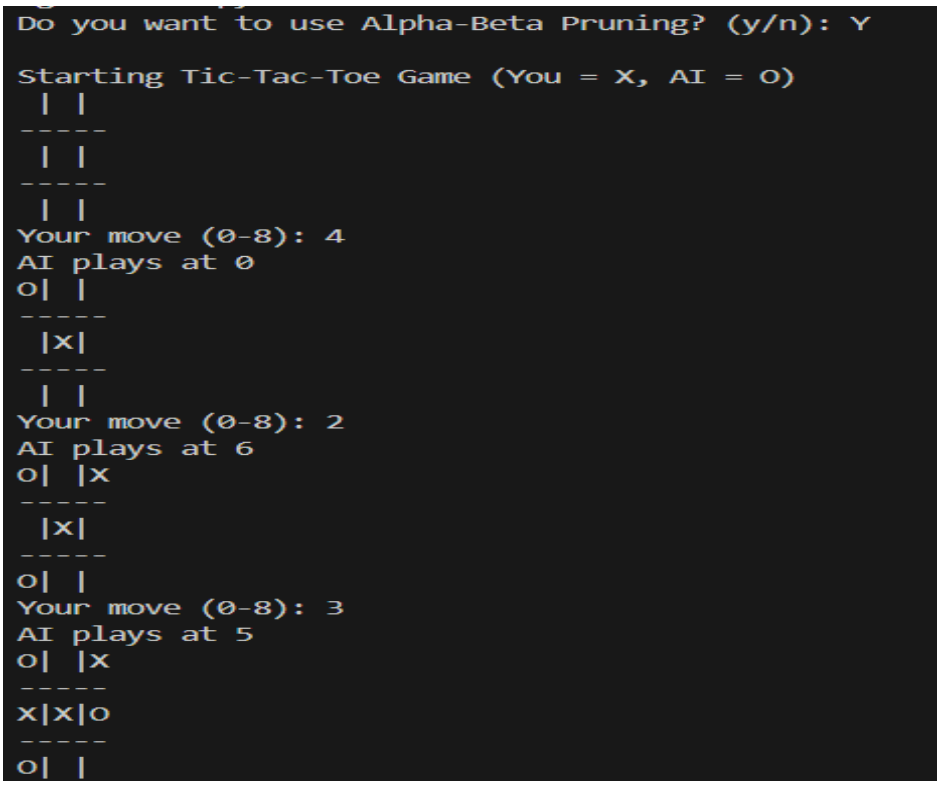
Alpha-Beta Pruning improves Minimax by pruning branches that will not affect the final decision, significantly reducing the number of nodes explored.

Anas Qutbi CR-22037

Code Features

- User chooses whether to enable Alpha-Beta pruning.
- Playable game loop vs AI (as 'O').
- Performance stats shown after the game ends.
- Uses `tabulate` to display node count and execution time comparison.

Output



Anas Qutbi CR-22037

```
Your move (0-8): 1
AI plays at 7
O|X|X
-----
X|X|O
-----
O|O|
Your move (0-8): 8
O|X|X
-----
X|X|O
-----
O|O|X
It's a draw!
```

Performance Comparison

After each game, a table is displayed showing:

- Time taken for both algorithms
- Number of nodes explored
- Efficiency ratio (how much fewer nodes Alpha-Beta used)

```
--- Performance Comparison ---
Algorithm      Time (s)    Nodes Explored  Efficiency
-----------  ----------  ----------------  ------------------
Minimax        5.79023             549945  Baseline
Alpha-Beta     0.311925             30709  17.91x fewer nodes
```

Conclusion

Alpha-Beta Pruning significantly improves performance over the standard Minimax algorithm by pruning unnecessary computations, achieving faster decisions with fewer node evaluations.