Advanced Algorithms

Project-1: TIC TAC TOE

Student: Zalal YOUSSEF

Professor: Marc Dinh

Paris, 2024

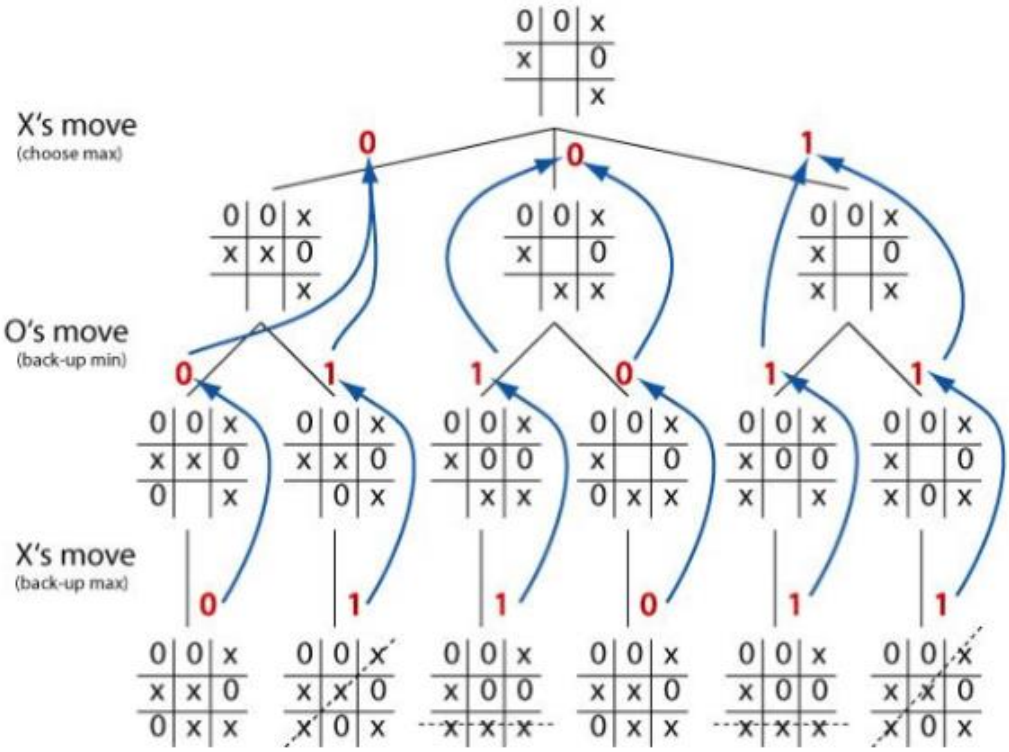
Description of the Project: Tic-Tac-Toe with AI

Presentation of the Problem

The objective of this project is to develop a Tic-Tac-Toe game that includes decision-making capabilities through artificial intelligence (AI). The AI will allow the computer to play against a human or another AI, making strategic decisions to optimize its chances of winning.

Tic-Tac-Toe is a simple yet classic game where two players take turns marking spaces in a 3x3 grid. The first player to align three of their marks (horizontally, vertically, or diagonally) wins the game. If all nine squares are filled without either player achieving three in a row, the game ends in a draw.

In this project, we aim to implement an AI that can analyze the game state and determine the best possible moves to maximize its chances of winning. The AI will use a tree of possibilities to explore different game scenarios and make decisions based on these calculations.



Principle of the Algorithm

Since Tic-Tac-Toe is a game of complete information, each player can theoretically calculate all possible moves and their outcomes. The challenge for the AI is to choose a path that leads to the most favorable final situation, ideally resulting in a victory. The algorithm must therefore "think ahead" and evaluate the potential future states of the game to make the best current move.

1. **Deployment of Game Situations**: The first step in the algorithm is to generate all possible game situations from the current state. Each possible move leads to a new game state, which in turn can lead to further moves. This branching continues until the end of the game,

creating a tree of possible game states. At the end of each branch, the game either results in a win, loss, or draw.

2. **Evaluation Function**: To make decisions, the AI needs to evaluate the quality of each game state. An evaluation function assigns a score to each state, reflecting its desirability. For instance:
    - +1000 for a win by the AI
    - 0 for a draw
    - -1000 for a win by the opponent

These scores help the AI determine the best moves by comparing the potential outcomes.

3. **Minimax Algorithm**: This algorithm is used to propagate the scores from the end states back to the initial state. It assumes that the opponent will always play optimally to minimize the AI's chances of winning. Therefore, the AI's decision-making process involves selecting the move that maximizes its minimum possible score (hence the name Minimax).
4. **Depth Limitation**: In practice, the AI may limit its search depth to reduce computation time. For instance, limiting the search to a depth of 1 means the AI only considers the immediate next move. Increasing the depth allows the AI to look further ahead, making more informed decisions but requiring more computational power.

Working Scenario

The implemented project allows users to simulate different types of Tic-Tac-Toe games:

1. **Human vs. AI**: A human player competes against the computer. The AI analyzes the game state, evaluates possible moves, and selects the optimal move based on the Minimax algorithm. The human player takes turns to make their moves manually.
2. **AI vs. AI**: Two AI players compete against each other. Each AI uses the Minimax algorithm to make decisions, allowing the simulation of a fully automated game.
3. **Human vs. Human**: Two human players take turns making moves without any AI intervention. This mode allows users to play a traditional game of Tic-Tac-Toe.
4. **Adjustable Difficulty**: The AI's difficulty can be adjusted by changing the depth of its search. A higher depth results in more strategic play but increases the computation time.
5. **Evaluation Scores**: During the game, the program can display the evaluation scores for the current generation of moves, along with their minimum and maximum values. This helps users understand the AI's decision-making process.
6. **Optimal Sequences**: The program can display the ideal sequence of moves for both the AI and the opponent to win from the current state to the end of the game tree.

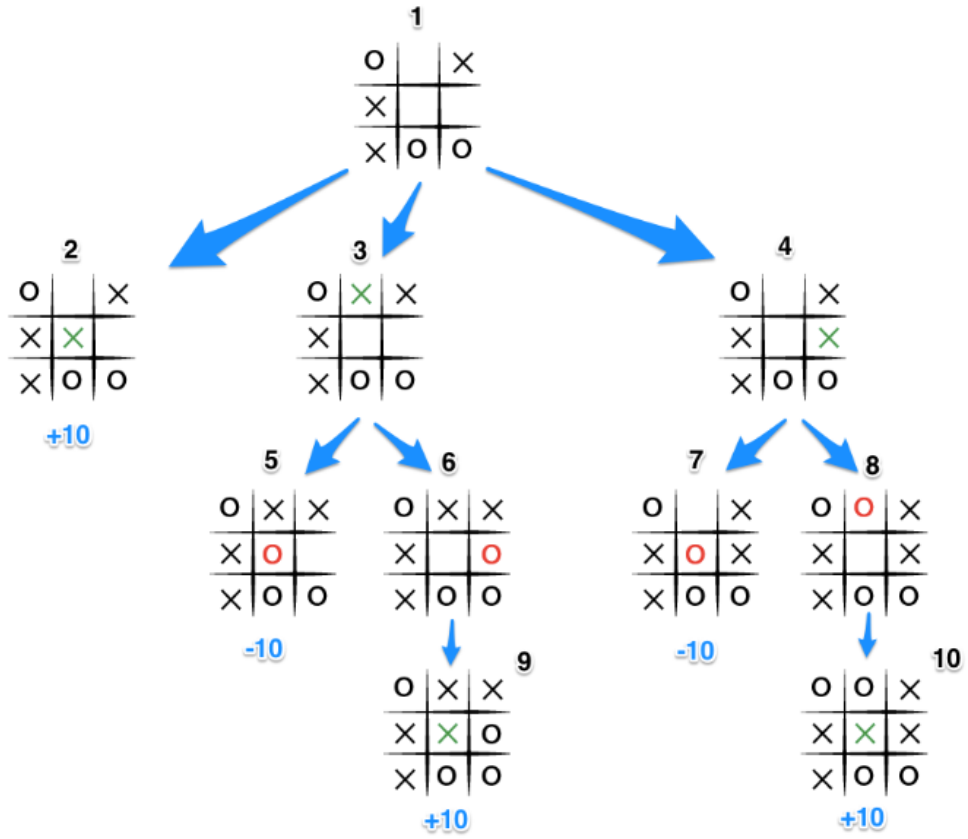
The goal of this project is not only to create a functional Tic-Tac-Toe game but also to explore the implementation and implications of AI decision-making in a simple game environment. Through

this project, we aim to gain insights into game theory, algorithm design, and the practical challenges of implementing AI in games.

Description of the Algorithms

Minimax Algorithm

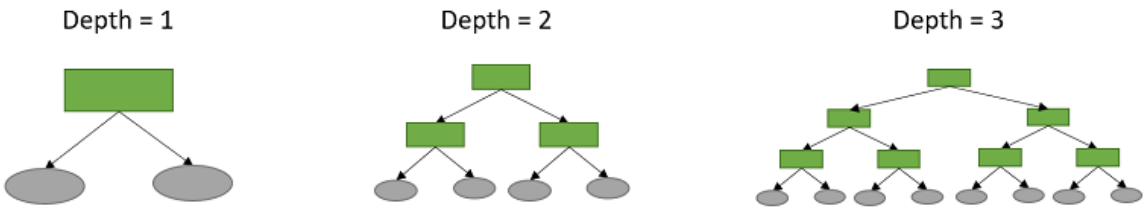
The Minimax algorithm is a decision-making tool used in AI for games like Tic-Tac-Toe. It generates a tree of game states, evaluating the best possible outcomes for both players, assuming optimal play.



Steps:

1. **Generate Game Tree:** Create a tree of all possible future game states from the current state.
2. **Evaluate Terminal States:** Assign scores to end states: +1000 for AI win, -1000 for opponent win, 0 for a draw.
3. **Backpropagate Scores:** Propagate scores up the tree:
    - Maximize score for AI's turn.
    - Minimize score for opponent's turn.
4. **Select Optimal Move:** Choose the move at the root with the highest score.

Depth-Limited Search (Depth Search)



Depth-Limited Search is a variant of Minimax that limits the search to a specific depth to manage computational feasibility.

Steps:

1. **Generate Partial Tree:** Create game states up to a set depth limit.
2. **Evaluate Non-Terminal States:** Use a heuristic to estimate desirability at the depth limit.
3. **Backpropagate Scores:** Propagate scores up the tree:
    - Maximize score for AI's turn.
    - Minimize score for opponent's turn.
4. **Select Optimal Move:** Choose the move at the root with the highest score.

Advantages:

- **Reduced Computation Time:** Limits the number of states evaluated.
- **Scalability:** Handles more complex games with larger state spaces.

Disadvantages:

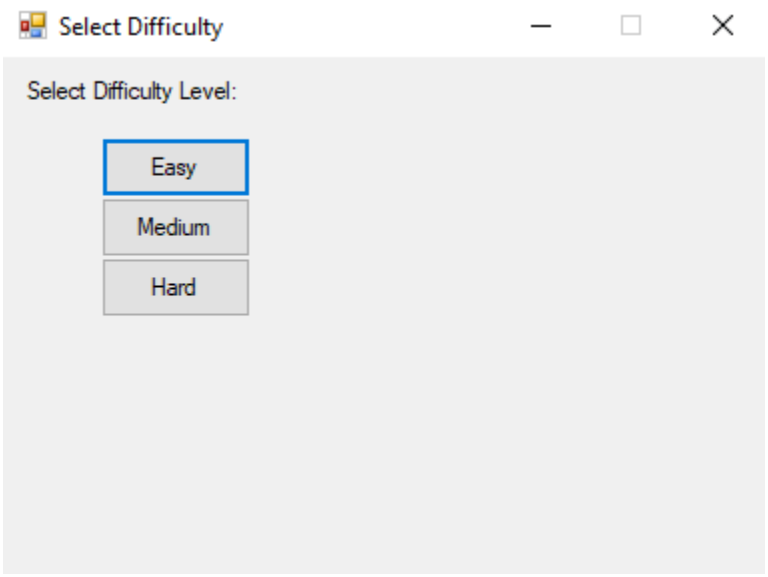
- **Less Accurate:** Heuristic evaluations at depth limit may lead to suboptimal decisions.

Example

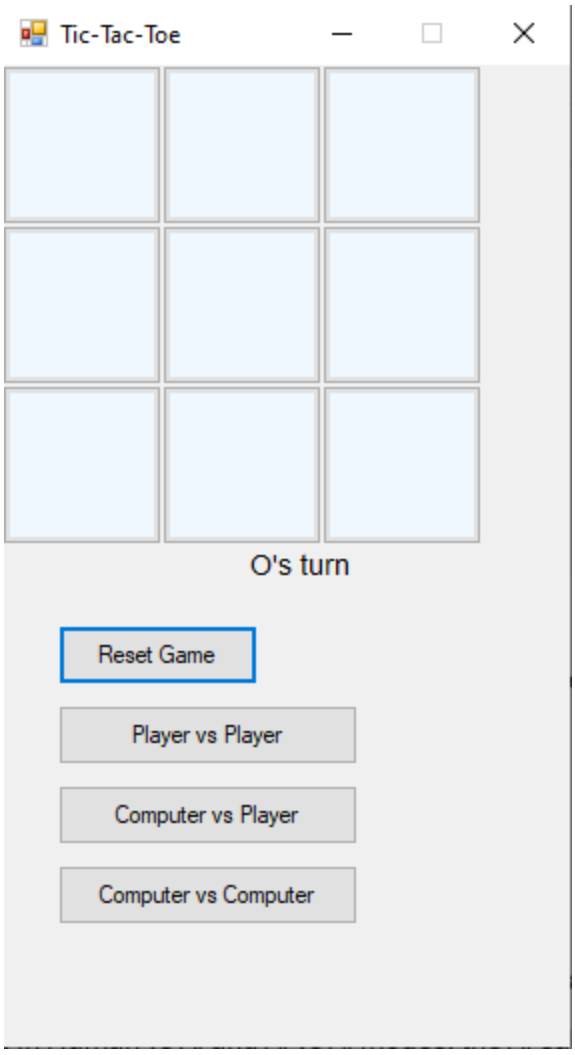
In Tic-Tac-Toe, the AI uses Minimax with Depth-Limited Search to make strategic decisions, balancing thoroughness and computational efficiency. The AI looks ahead a few moves, evaluates the game states using a heuristic, and chooses the move with the best predicted outcome.

Working Scenario

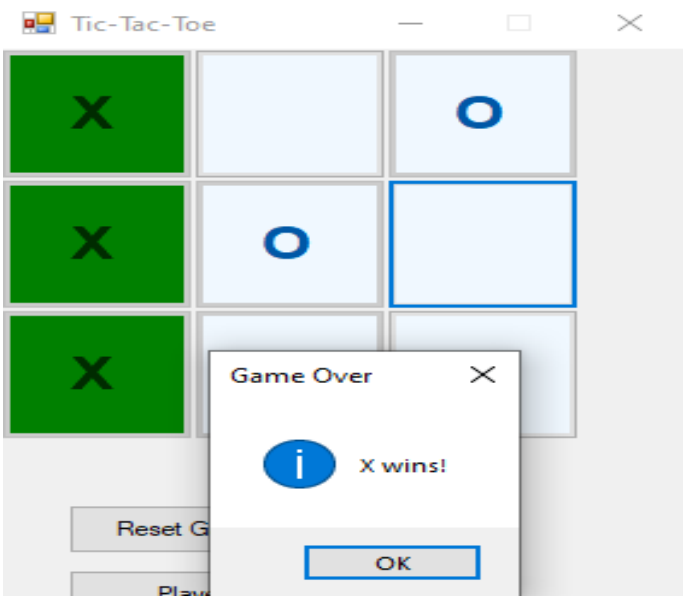
1. **Select Difficulty:**
    - **Easy**
    - **Medium**
    - **Hard**.



2. **Select Game Mode:**
    - **Human vs Human:** Two players take turns playing against each other.
    - **Human vs AI:** One player competes against the computer.
    - **AI vs AI:** Watch the computer play against itself.



3. **Enjoy the Game:**
    - **Make Moves:** Players take turns clicking on the board to place their symbols.
    - **AI Decisions:** In Human vs AI and AI vs AI modes, the AI calculates its moves using the Minimax algorithm and depth-limited search.
    - **Game End:** The game declares a winner when a player aligns three symbols, or a draw if no moves are left. Winning cells are highlighted, and scores are displayed.


Tic-Tac-Toe
X
O
X
O
X
Game Over
X wins!
OK
Reset G