Dots and Boxes Game

AI Project – Artificial Intelligence

Section 3 – AI Department
Faculty of Computers and Artificial Intelligence
Banha University

TEAM MEMBERS

- Nesma Hesham
- Menna Haitham
- Malak Ahmed
- Nourhan Medhat
- Hemat Hamdy

COURSE INFORMATION

➢ **Discussion Supervised by:** Eng. Youssef Elbaroudy
Course Doctor: Dr. Sara Sweidan

➢ **Course Name**: Artificial Intelligence
Academic Year: 2025 – 2026

1- Game Description

Dots and Boxes is a strategic two-player game implemented by our team as part of our Artificial Intelligence course project. The game is played on a grid of dots, where players take turns connecting two adjacent dots either horizontally or vertically. The main goal is to complete boxes: when a player draws the fourth side of a square, they claim the box and earn a point.

In our implementation, the game consists of two agents:

1. Human Agent: The player interacts with the game using a Graphical User Interface (GUI), selecting lines to draw with mouse clicks.
2. Computer Agent (AI): The AI uses Alpha-Beta Pruning, an optimized version of the Minimax algorithm, to evaluate possible moves and select the most strategic action. This allows the AI to maximize its score while minimizing the human player's advantage, with reduced computation time compared to a standard Minimax approach.

The system ensures:

- Accurate score tracking for both agents.
- Correct handling of turns, ensuring the right agent plays next based on completed boxes.
- Real-time visual representation of the game board and player actions.

This project demonstrates our understanding of AI search algorithms, strategic decision-making for multiple agents, and real-time game management. The project is fully implemented and documented by the team members: Nesma Hesham, Menna Haitham, Malak Ahmed, Nourhan Medhat, and Hemat Hamdy.

2- Game Steps

1. Start Game: The board of dots is displayed, and scores are set to zero.
2. Player Turn (Human): The human agent starts first and draws lines in blue using GUI.
3. AI Turn (Computer): The computer agent draws lines in red, choosing The best move using Alpha-Beta Pruning.
4. Box Completion: If a player completes a box, they earn a point and take an extra turn. (Boxes are filled with the player's color (blue for human, red for AI).
5. Turn Switching: If no box is completed, the turn switches to the other agent.
6. Repeat Steps 2–5: Continue until all lines are drawn and all boxes are claimed.
7. End Game: Scores are compared, and the winner (human or AI) is announced.

3- Methodology

1. Game Design & Rules Understanding:
    - We started by analyzing the rules of **Dots & Boxes** and defining the gameplay between the human and AI.
    - The human player starts with **blue** and the AI player with **red** (Points are awarded when a box is completed).
2. Data Structures & State Representation:
    - Used **2D arrays** to store lines and completed boxes.
    - The **state tuple** holds all information about the board, scores, and current turn.
3. AI Implementation – Alpha-Beta Pruning:
    - Developed AI using **Alpha-Beta** with **search depth = 4** to ensure fast calculations and maintain a playable game.
    - The AI analyzes possible moves and selects the one that maximizes its score.
4. Function Design:
    - The project is divided into main functions to handle moves, scoring, turn management, and GUI.
    - **Details of each function will be explained in the following sections.**
5. GUI Development:
    - Used **Pygame** to display the board, lines, boxes, and player scores.
    - Turn indicators guide the human player.
6. Testing & Iteration:
    - The game was tested at each stage to ensure correctness and smooth gameplay.
    - AI strategy was refined to provide challenging yet fair gameplay.

4- Functions Description

1. take_action

- **Description:** Executes the chosen move (horizontal or vertical line) on the board and updates the current state including completed lines, boxes, and scores.
- **Purpose:** To update the game state after any move by the human or AI.

2. human_play_gui

- **Description:** Receives the human player's input through the graphical interface (mouse clicks on lines).
- **Purpose:** Allows the human agent to interact with the game and select moves.

3. computer_play_gui

- **Description:** Chooses the best move for the AI using **Alpha-Beta Pruning** and executes it on the board.
- **Purpose:** Represents the AI agent’s strategic move selection.

4. show_game_over

- **Description:** Displays the final game result on the screen, including the winner and the scores of both players.
- **Purpose:** Informs players of the game outcome clearly

5. evaluate

- **Description:** Calculates the difference between the AI’s score and the human player’s score in the current state.
- **Purpose:** Helps the **Alpha-Beta Pruning** algorithm estimate the value of each state during the search.

6. available_actions

- **Description:** Returns all possible moves (unclaimed lines) in the current state.
- **Purpose:** Provides the AI with all available options at each step for decision-making.

7. check_terminal

- **Description:** Checks whether the game has ended (all lines drawn) and determines the winner or a draw.
- **Purpose:** Determines the end of the game and evaluates the final outcome.

8. current_player

- **Description:** Determines whose turn it is based on the current state.
- **Purpose:** Identifies the active agent, human or AI.

9. draw_gui

- **Description:** Displays the board, completed lines, boxes, and player scores using **Pygame**.
- **Purpose:** Provides an interactive graphical representation of the game.

10. Alpha-Beta Pruning (The Ai Algorithm)

- **Description:** AI search algorithm that analyzes possible future moves and selects the best action while pruning unnecessary branches.
- In our implementation, the **Alpha-Beta Pruning algorithm** explores possible moves up to a **depth of 4**. This depth is chosen to **keep the AI's calculation fast** and ensure the game remains playable and responsive for the human player

1. **Simulate All Moves:**
   The AI considers all possible moves from the current state using `take_action`.
2. **Evaluate Outcomes Recursively:**
   For each move, it predicts the human's responses (who tries to minimize AI's score) and continues the simulation up to a set depth.
3. **Score Each Move:**
   At the end of each path (or depth limit), it evaluates the state using `evaluate` (AI score – Human score).
4. **Compare Scores & Prune:**
   The AI keeps track of the best score, and ignores moves that cannot produce a better outcome (pruning).
5. **Select the Best Move:**
   Finally, the AI executes the move with the **highest score**.

11. Dots and Boxes Initiate (__init__)

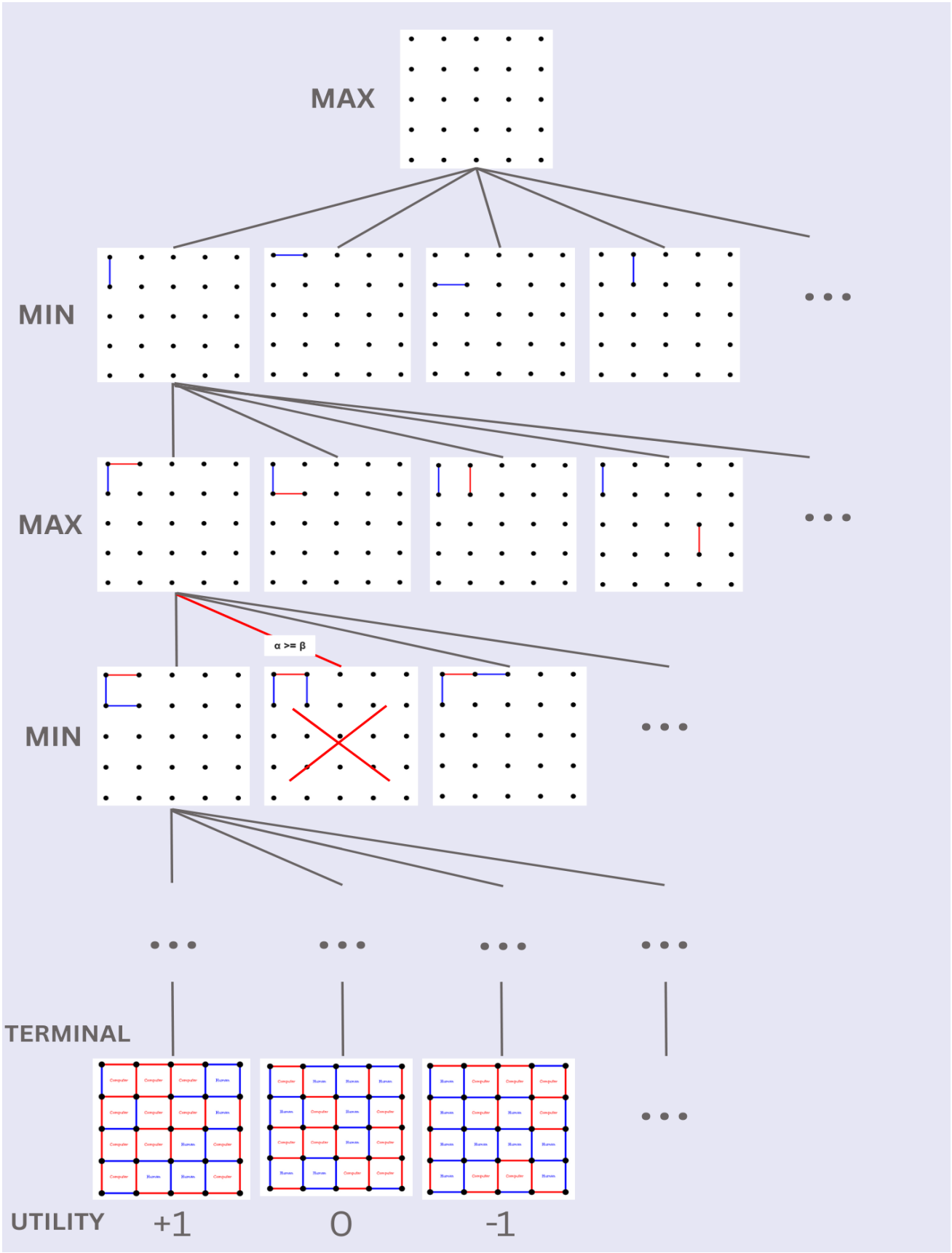
- **Description:** Initializes arrays for lines, boxes, scores, and the current turn.
- **Purpose:** Sets up the initial game state before any moves are made.

12. Game Loop

- **Description:** Main game loop that manages turns between human and AI, checks for game termination after each move, and updates the GUI.
- **Purpose:** Controls the entire flow of the game, including input handling, board updates, and determining the winner.

5- Alpha-Beta States Tree

The Game State Tree shows all possible moves and outcomes the AI considers using Alpha-Beta Pruning to choose the best action



6- Game Screenshots

Below are some screenshots demonstrating the gameplay of **Dots and Boxes – Human vs AI**. The human player is represented in **blue**, and the AI player is represented in **red**