Project title: 2048 game with Artificial Intelligence algorithms

CSE 4110: Artificial Intelligence Laboratory

Submitted To:

Md. Shahidul Salim
Lecturer
Most. Kaniz Fatema Isha
Lecturer
Department of Computer Science and Engineering

Submitted By:

Sadia Afrin
1907037
Arpita Kundu
1907050
Year: 4th
Semester: 1st

Khulna University of Engineering & Technology, Khulna-9203

Objectives:

The objectives of this project are given below:

- To develop a 2048 game where Artificial Intelligence (AI) algorithms are used to optimize gameplay strategies.
- To employ the Alpha-Beta Pruning algorithm to optimize the AI's move selection, improving computational efficiency.
- To utilize Fuzzy logic for enhancing decision-making processes during gameplay.
- To implement Genetic Algorithms (GA) for evolving effective strategies within a given time frame to achieve higher scores in the game.
- To compare the performance of alpha-beta pruning, fuzzy logic and genetic algorithms in solving the 2048 game.
- To provide insights into the practical implementation of AI techniques learned during the laboratory sessions.
- To offer a user-friendly interface that allows for the visualization of the AI's decision-making process.

Introduction:

The game called 2048 has been taking up a lot of the limelight since Gabriele Cirulli has released it in 2014. The game was created with a very concise and straightforward gameplay; therefore, the user's goal is to put the numbered tiles on a 4 × 4 grid together so that a tile with the value of 2048 is achieved. It takes not only intelligence but also strategy to score at a high level. It is indeed a very challenging game, which is highly enjoyed by the players.

Players who like playing the game 2048 were required to come up with different AI algorithms in order to be more optimized. The ones that are of the authors' concern are fuzzy logic, genetic algorithms, and the alpha-beta pruning algorithm. These methods have been selected for their exclusive features in the process of decision-making, optimization, and strategic planning. The fuzzy logic technique is used to process the uncertainty and doubt that have their roots in the game's environment, while the genetic algorithms are used for the technical aspect of the problem that is, how to evolve the best strategies through the many rounds. The alpha-beta pruning algorithm is a method for determining quickly the outcomes of each move and thereby the AI player indirectly selects the best moves since they require the least amount of computational power.

The purpose of adding these methods is to advance the 2048 computer player's capabilities by going after higher scores, as well as to experiment with a more challenging human opponent in this gaming scenario. The report is going to describe the implementation phase, estimate the performance of every single algorithm, and make the comparison of their effectiveness in learning the 2048 game. Furthermore, the project acts as a real-world demonstration of cutting-edge AI procedures, which serve as the reinforcement element of the topics got in the lab and illustrate the opportunity for the learning of practical skills through the maximal application of the knowledge gained.

Methodology:

The methodology section outlines the approaches used to implement AI techniques in the 2048 game. This includes the application of the alpha-beta pruning, fuzzy logic, genetic algorithms. These algorithms are used to optimize gameplay and improve the AI's performance.

1. Alpha-Beta Pruning Algorithm

The alpha-beta pruning algorithm is an optimization technique for the minimax algorithm, commonly used in decision-making processes for two-player games and strategic AI. The minimax algorithm evaluates all possible moves by simulating decisions for both the player aiming to minimize potential losses while maximizing gains. However, this process can become computationally intensive as it requires evaluating an exponentially growing number of game states as the depth of the search increases. Alpha-beta pruning enhances the efficiency of the minimax algorithm by "pruning" or eliminating branches in the game tree that cannot influence the final decision. This is achieved by maintaining two threshold values:

- **Alpha:** The best value that the maximizer can guarantee at that level or above.

- **Beta:** The best value that the minimizer can guarantee at that level or above.

During the traversal of the game tree, if the minimizer encounters a move that results in a value less than or equal to alpha, the maximizer can disregard the remaining branches from that point (since a better option is already available). Similarly, if the maximizer encounters a move that results in a value greater than or equal to beta, the minimizer prunes the remaining branches. This pruning significantly reduces the number of nodes that need to be evaluated, leading to faster decision-making.

Application of Alpha-Beta Pruning in the 2048 Game

In the 2048 game, the AI leverages alpha-beta pruning to optimize its strategy for both shuffling the board and adding new tiles in a manner that maximizes the chances of creating high-value tiles while preventing the grid from filling up prematurely. Here’s how the algorithm is applied step-by-step:

1. Game Tree Construction:

- The AI begins by constructing a game tree that represents all possible future states of the 2048 grid following a move. Each node in this tree corresponds to a specific configuration of the board after a move and the addition of a new tile.

- For instance, if the AI decides to shift tiles to the left, the resulting grid configuration is captured as a node. The AI then considers possible placements of new tiles (typically a 2 or a 4) in any empty space on the grid, which creates additional branches in the tree.

2. Minimax Algorithm:

- The minimax algorithm is applied to evaluate this game tree, simulating both the AI’s strategic moves and the placement of new tiles (which can be seen as an adversarial or environmental factor). The AI's goal is to maximize its score by selecting moves that optimize tile merging while minimizing the risk of filling the grid too quickly with low-value tiles.

- For example, the AI might evaluate whether shifting tiles upwards will allow it to combine two 64 tiles into a 128 tile, while also considering where the worst-case placement of a new 2 tile could occur.

3. Alpha-Beta Pruning:

- As the AI evaluates the branches of the game tree using the minimax algorithm, alpha-beta pruning is employed to eliminate branches that do not need further exploration.

- Suppose the AI determines that shifting tiles to the right will lead to a lower score compared to shifting them to the left, and this rightward shift also leads to a less favorable board setup. In that case, the AI prunes this branch, disregarding it from further evaluation.

- This pruning process reduces the number of potential moves the AI needs to consider, enabling it to focus only on the most promising strategies for creating high-value tiles while maintaining an open grid.

4. Move Selection:

- After pruning the game tree, the AI selects the move that maximizes its chances of success based on the remaining nodes. The selected move not only aims to achieve the highest immediate score but also sets up the board for favorable future moves.

- For example, the AI might choose to shift tiles downward if this move consolidates high-value tiles at the bottom of the grid, making it easier to merge them in future moves while keeping the upper rows clear for new tiles.

- The AI’s final decision is based on the minimax evaluation with alpha-beta pruning, ensuring the best possible outcome given the current and potential future states of the game.

2. Fuzzy Logic Algorithm

Fuzzy logic is a form of many-valued logic that deals with approximate reasoning rather than fixed and exact values. Unlike traditional binary logic, where variables are either true or false (1 or 0), fuzzy logic allows for a spectrum of truth values ranging between 0 and 1. This enables the system to handle uncertainty and partial truth, making it particularly useful in complex decision-making processes where binary decisions are inadequate. In fuzzy logic, rules are formulated using linguistic variables (e.g., "low," "medium," "high") rather than precise numerical values. These rules are then used to evaluate inputs and generate outputs based on degrees of truth. *Fuzzy logic systems consist of three main components:*

- Fuzzification: This process converts crisp input values into fuzzy values by mapping them onto a predefined set of fuzzy sets. For example, a score of 512 in the 2048 game might be mapped to fuzzy sets like "low" and "medium" with varying degrees of membership.

- Inference: The inference engine applies fuzzy rules to the fuzzified inputs to determine the output. These rules might be in the form of "If the score is low and the grid is sparse, then the game state is favorable.

- Defuzzification: The final step converts the fuzzy output back into a crisp value that can be used for decision-making. This might involve calculating a weighted average of possible outcomes to determine the best move.

Application of Fuzzy Logic in the 2048 Game

In the 2048 game, fuzzy logic is implemented to manage the uncertainty and complexity involved in decision-making processes, particularly when it comes to determining the best move or when to introduce a new tile. Here’s how fuzzy logic is applied step-by-step:

1. Fuzzification:

- The AI begins by fuzzifying the current state of the game board. Key aspects such as the tile distribution, available empty spaces, and the current score are converted into fuzzy sets. For example:

- The number of empty tiles might be categorized into fuzzy sets like "few," "moderate," and "many."

- The distribution of high-value tiles could be categorized as "clustered" or "spread out."

- This allows the AI to process these values in a way that accounts for the inherent uncertainty and variability in the game.

2. Inference:

- The fuzzy inference engine applies a set of predefined fuzzy rules to the fuzzified inputs. These rules are designed to reflect effective strategies for playing the 2048 game. Examples of such rules might include:

- "If the number of empty tiles is few and the highest tile is in a corner, then the game state is critical."

- "If the score is high and the grid is balanced, then the game state is favorable."

- These rules help the AI determine the overall game state and decide whether to prioritize certain moves, such as merging tiles or creating space on the grid.

3. Defuzzification:

- After evaluating the fuzzy rules, the AI needs to make a concrete decision, such as selecting the next move or determining the best position for a new tile. This is done through defuzzification, which converts the fuzzy output back into a precise action.

- For example, if the fuzzy inference suggests that the grid is becoming too crowded (with a high degree of membership in the "critical" fuzzy set), the AI might prioritize moves that create more empty space, even if they don't immediately result in high-value merges.

- Alternatively, if the game state is deemed favorable, the AI might focus on combining high-value tiles to maximize the score.

3. Genetic Algorithm

Genetic Algorithms (GAs) are a class of optimization algorithms inspired by the process of natural selection in biological evolution. GA operates on a population of potential solutions, applying the principles of selection, crossover (recombination), and mutation to evolve better solutions over successive generations. The key components of a genetic algorithm are:

- Population: A set of potential solutions (often called individuals or chromosomes) to the problem at hand. Each individual represents a possible configuration or strategy.

- Fitness Function: A function that evaluates how "fit" or effective each individual is concerning the problem. In the context of the 2048 game, fitness might be related to the score achieved by the configuration or strategy.

- Selection: The process of choosing individuals from the current population to be parents for the next generation. Individuals with higher fitness are more likely to be selected.

- Crossover (Recombination): A process where two parent individuals combine to produce offspring that inherit characteristics from both parents. This mimics the genetic crossover in biological reproduction.

- Mutation: A process where random changes are introduced to an individual's characteristics to maintain diversity within the population and prevent premature convergence on suboptimal solutions.

GAs are particularly effective in searching large, complex spaces for optimal or near-optimal solutions, making them suitable for tasks like evolving strategies for the 2048 game.

Application of Genetic Algorithms in the 2048 Game

In the 2048 game, Genetic Algorithms are used to evolve strategies that help the AI achieve higher scores by optimizing its moves and tile placement over time. Here’s how the genetic algorithm is applied step-by-step:

1. Population Initialization:

- The AI begins by generating an initial population of random strategies for playing the 2048 game. Each strategy can be represented as a sequence of potential moves or as a set of rules that dictate how the AI should respond to different game states.

- These strategies serve as the starting point for the genetic algorithm.

2. Fitness Evaluation:

- Each strategy in the population is evaluated using a fitness function, which measures its effectiveness in achieving high scores in the 2048 game. For example, the fitness function might be based on:

- The final score achieved by the AI using that strategy.

- The highest tile value reached.

- The efficiency of tile merging and the ability to maintain open spaces on the grid.

- The strategies that perform better according to the fitness function are deemed more "fit" and have a higher chance of being selected for the next generation.

3. Selection:

- The AI selects the fittest strategies from the current population to serve as parents for the next generation. Various selection methods can be used, such as:

- ***Roulette Wheel Selection***: Where the probability of selection is proportional to the strategy's fitness.

- ***Tournament Selection***: Where a subset of strategies is chosen at random, and the fittest among them is selected.

- This process ensures that better strategies have a higher likelihood of passing their characteristics to the next generation.

4. Crossover (Recombination):

- Selected parent strategies are paired, and crossover is applied to produce offspring that inherit characteristics from both parents. For example:

- If one parent prioritizes merging high-value tiles and another focuses on maintaining grid space, the offspring might combine these strategies to optimize both aspects.

- Crossover points are chosen where the strategies are split and recombined, creating new strategies that incorporate elements from both parents.

5. Mutation:

- To maintain diversity in the population and prevent stagnation, mutation is applied to some of the offspring. This involves making random changes to parts of a strategy, such as altering the priority of certain moves or introducing new decision rules.

- Mutation helps explore new strategies that might not have been reached through crossover alone, potentially leading to novel and more effective approaches.

6. Next Generation and Iteration:

- The new generation of strategies, consisting of offspring and some of the original population, replaces the old generation. The genetic algorithm then repeats the process of fitness evaluation, selection, crossover, and mutation over multiple generations.

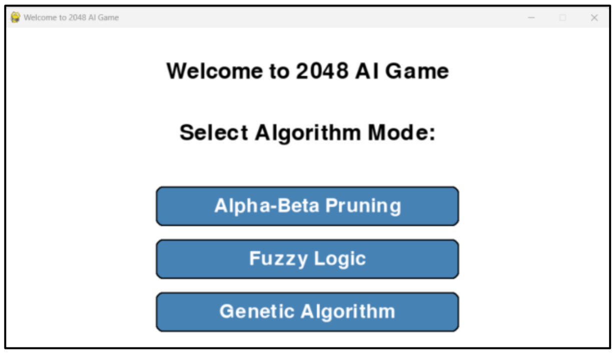
- Over time, the population evolves towards more effective strategies for playing the 2048 game, with each generation potentially outperforming the previous one.

Game Design and Interface

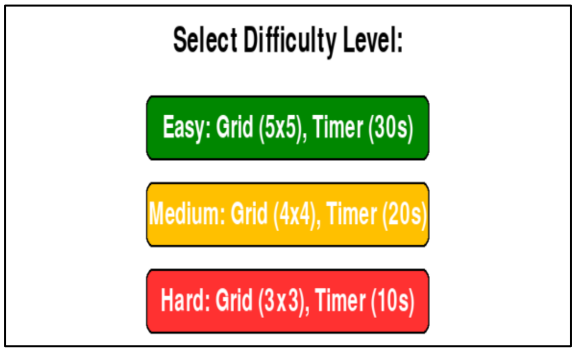
The 2048 AI Game is designed to offer an engaging and visually appealing experience, with an intuitive interface that caters to both beginners and seasoned players. The game is built using Pygame, a versatile library for creating graphical applications and games in Python.

Main Screen Layout:

Upon launching the game, players are greeted with a clean and minimalistic home screen. The screen displays a welcoming message, followed by options to select the AI algorithm mode and the difficulty level. The algorithm selection buttons are positioned vertically for easy access, while the difficulty levels are presented in a vertical layout, detailing about parameters, making it straightforward for users to set up their preferred game parameters (like grid size, timer etc.).



Algorithm Mode Selection

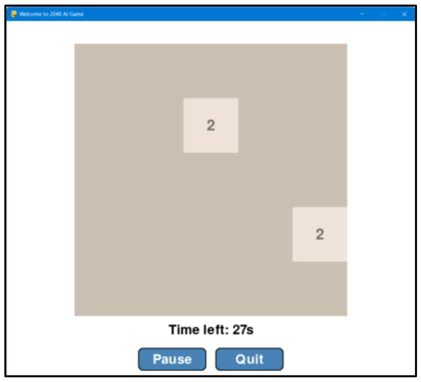

Select Difficulty Level:
Easy: Grid (5x5), Timer (30s)
Medium: Grid (4x4), Timer (20s)
Hard: Grid (3x3), Timer (10s)

Difficulty Level Selection

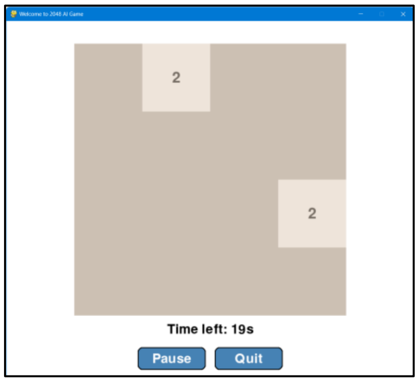
Fig 1: Main Screen Layout

In-Game Interface:

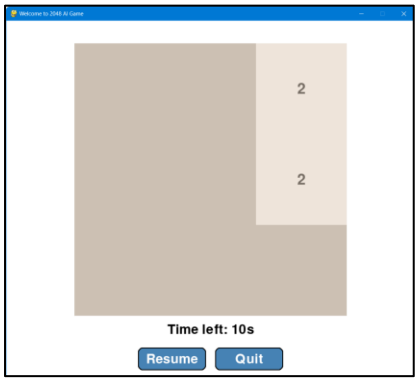
Once the game begins, players are presented with a grid that dynamically adjusts in size based on the selected difficulty level. The grid is centered on the screen, with tiles displaying values that combine as the player moves them. Each tile is color-coded according to its value, enhancing visual clarity and helping players quickly identify their next moves. A timer is displayed below the grid, indicating the remaining time for the current round. Beneath the timer, pause and quit buttons are centered, allowing players to manage their game session with ease.



Easy (5x5 grid, 30s timer)



Medium (4x4 grid, 20s timer)

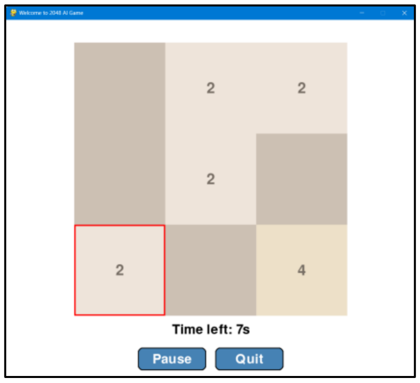


Hard (3x3 grid, 10s timer)

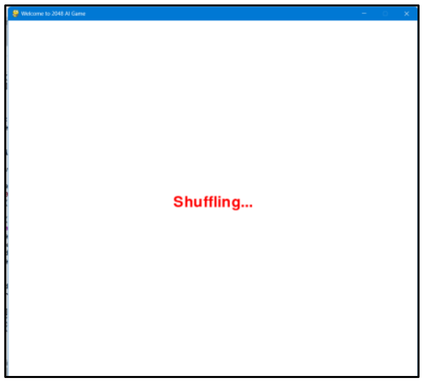
Fig 2: Different Game Layout Based on Difficulty Level Selection

User Interaction:

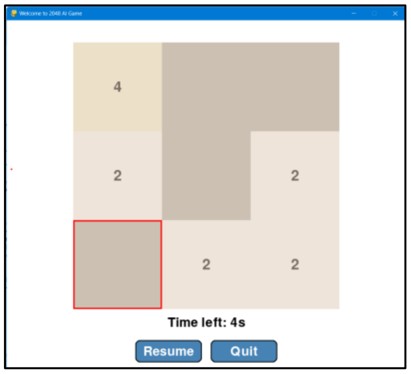
The game supports intuitive tile selection and movement through keyboard input. Players can easily select and move tiles in four directions—up, down, left, and right—by pressing the corresponding arrow keys. Additionally, the game includes a shuffle feature, triggered when the timer runs out, adding an extra layer of challenge by rearranging the tiles on the board.



Before shuffle



Shuffling message



After shuffle

Fig 3: Red square indicating selection and Shuffling when time runs out

Feedback and Notifications:

Visual cues are provided to guide the player throughout the game. For instance, selected tiles are highlighted with a red border, and a "Game Over" message appears prominently when no more moves are possible. The pause feature freezes the timer and changes the pause button text to "Resume," ensuring players have full control over their game experience.

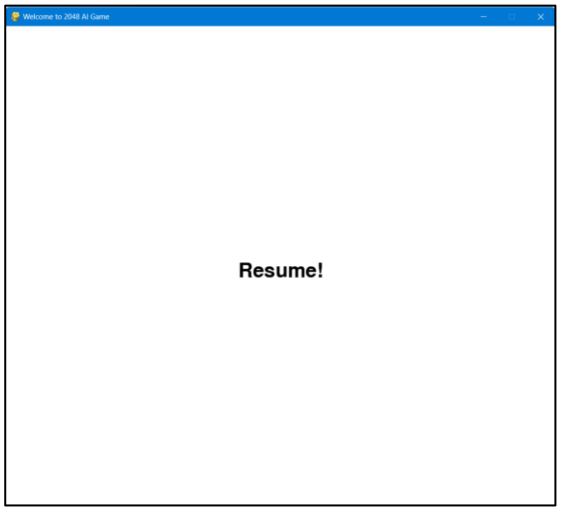


Fig 4: Resume message after shuffling

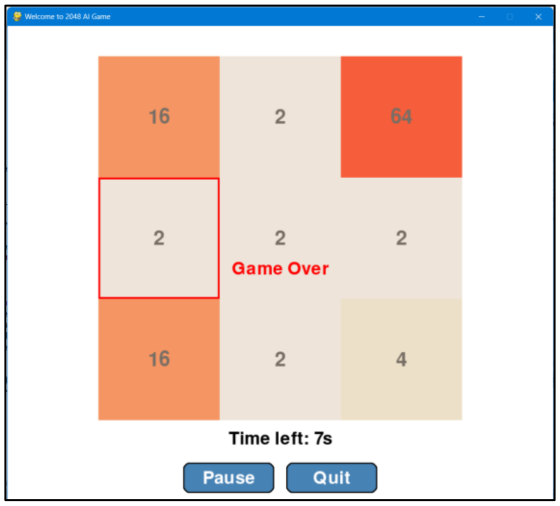


Fig 5: Game over message at unplayable state

Overall, the game's design emphasizes simplicity and user-friendliness, ensuring that the focus remains on the gameplay itself while providing a polished and cohesive interface.

Conclusion:

The implementation of fuzzy logic, genetic algorithms, and the alpha-beta pruning algorithm in the 2048 game demonstrates the potential of AI techniques in optimizing complex decision-making processes. Each algorithm contributed unique strengths to the AI's gameplay, allowing it to achieve higher scores and make more strategic moves than a typical human player. Fuzzy logic provided a flexible framework for handling the game's uncertainty and ambiguity, enabling the AI to make nuanced decisions based on the state of the board. Genetic algorithms allowed the AI to evolve and refine its strategies over time, adapting to different gameplay scenarios and continuously improving its performance. Alpha-beta pruning significantly enhanced the efficiency of the AI by reducing the computational complexity involved in evaluating potential moves, allowing for faster and more effective decision-making. The comparison of these methods revealed their individual advantages and limitations in the context of the 2048 game. While fuzzy logic excelled in dealing with uncertain situations, genetic algorithms proved valuable in evolving strategies, and alpha-beta pruning optimized decision-making speed. Together, these approaches provided a comprehensive AI solution capable of mastering the 2048 game. This project not only demonstrated the effectiveness of advanced AI techniques in a practical application but also reinforced the concepts covered in laboratory sessions. The insights gained from this implementation can be extended to other areas where similar AI strategies are applicable, highlighting the versatility and power of these algorithms in solving real-world problems. Overall, the integration of these AI methods into the 2048 game represents a successful application of theoretical knowledge to a challenging problem, paving the way for further exploration and innovation in AI-driven game design and beyond.