**Project Report: AI-Based 64-Puzzle Solver with Graph Traversal Visualization**

**Submitted By:**

Muneeb Baig (22K-5095)  
Rafay Ahmed (22K-5030)

**Course:**

Artificial Intelligence

**Instructor:**

Ms. Almas Ayesha Ansari

**Submission Date:**

8-May-2025

**1. Introduction**

This project demonstrates the use of Artificial Intelligence techniques to solve a complex variant of the sliding puzzle game—the **64-Puzzle (8x8 grid)**. The traditional 15-puzzle was extended to increase complexity and further challenge our AI agent's efficiency. The core objective is to develop a solver using the *A Search Algorithm*\* with **Manhattan Distance** as a heuristic and to visualize the search space through a **graph traversal**.

**2. Objectives**

* Design an AI agent capable of solving the 64-Puzzle optimally.
* Employ the A\* algorithm for informed graph search.
* Utilize the Manhattan Distance heuristic for effective cost estimation.
* Visualize the traversal of the AI agent using NetworkX and Matplotlib.
* Develop a Tkinter-based GUI for interactive puzzle simulation and animation.

**3. AI Methodology**

***3.1 A Search Algorithm*\***

The A\* algorithm was chosen for its balance between completeness, optimality, and performance. Each node in the puzzle is evaluated based on:

* g(n): Actual cost from the initial state to current state
* h(n): Estimated cost (heuristic) to reach the goal from current state
* f(n) = g(n) + h(n)

**3.2 Heuristic Function**

The **Manhattan Distance** heuristic is used, where for each tile (excluding the blank), the sum of vertical and horizontal distances from its current position to its goal position is computed. This heuristic is:

* **Admissible**: Never overestimates
* **Consistent**: Ensures optimality of A\*

**3.3 Graph Visualization**

A directed graph is dynamically constructed using **NetworkX** to represent state transitions during traversal. This provides intuitive insight into how the search progresses.

**4. Game Mechanics**

**4.1 Rules**

* The 64 tiles (including a blank tile) are arranged on an 8x8 grid.
* Only adjacent tiles can slide into the blank space.
* The goal state is defined as an ordered arrangement from 1 to 63, with the blank tile at the last position.

**4.2 GUI Interaction**

* A GUI built with **Tkinter** allows users to:
  + Generate a randomized puzzle by specifying scramble steps.
  + Automatically solve the puzzle via the AI agent.
  + Visualize each move and the entire search space.

**5. Implementation Overview**

**5.1 Languages & Libraries**

* **Python**: Core programming language
* **Tkinter**: GUI design
* **Heapq**: Priority queue management
* **NetworkX**: Directed graph construction
* **Matplotlib**: Graph visualization
* **Random, Time**: Puzzle generation and animations

**5.2 Code Highlights**

* Environment class handles grid logic and heuristics.
* Node class maintains A\* attributes (g, h, f, and parent).
* AIAgent executes the A\* search and builds the graph.
* PuzzleGUI provides user interface and visual feedback.

**6. Evaluation & Results**

* The A\* agent was tested with varying levels of scrambled puzzles (steps: 10–50).
* Real-time performance was acceptable for lower scramble depths (e.g., 10–20).
* For higher complexity (e.g., 50+), search space and memory demands increased significantly, highlighting A\*'s limitations in large state spaces.
* Graph visualization provided excellent insights into path selection and node expansion patterns.

**7. Challenges**

* The **state space for an 8x8 puzzle** is extremely large (factorial of 64), making some instances computationally intensive.
* Memory usage increased sharply with deeper searches.
* Ensuring a responsive GUI while solving complex puzzles required balancing sleep intervals and updates.

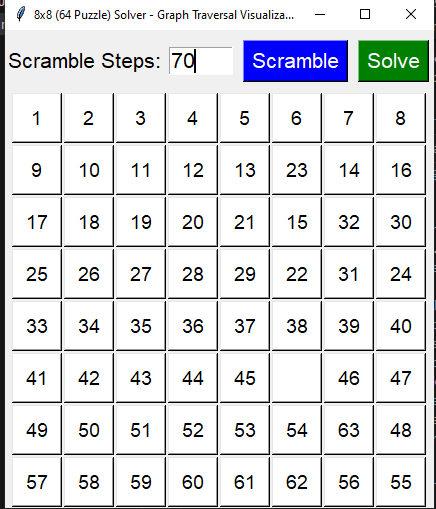
**8. Conclusion**

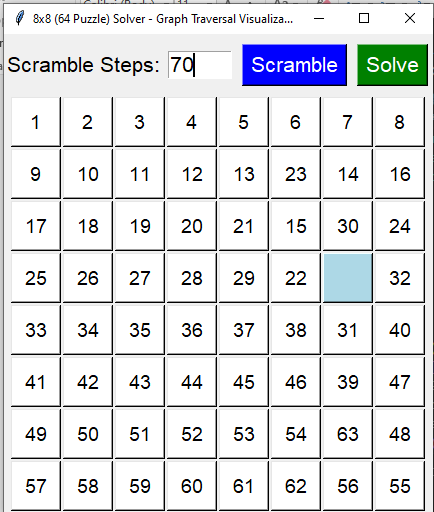
This project demonstrated the feasibility and educational value of applying A\* to solve high-complexity puzzles like the 64-tile variant. The integration of visualization tools and GUI significantly enhanced user understanding of AI traversal strategies. Though not practical for real-time solutions of all instances due to computational complexity, the project serves as a robust learning platform for AI concepts.

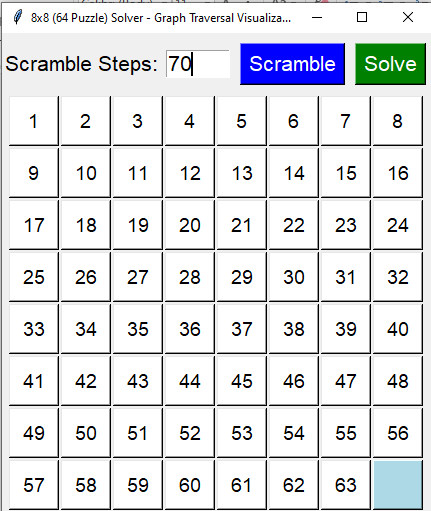
**9. Future Work**

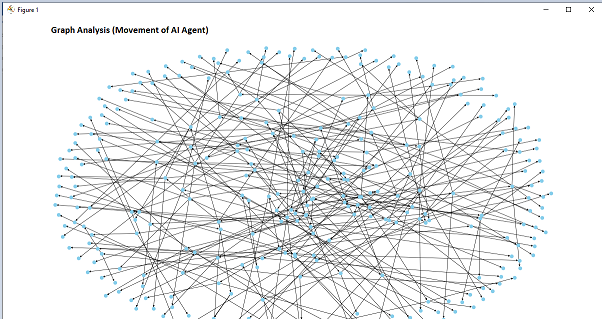
* Implement **iterative deepening A**\* (IDA\*) for better memory management.
* Optimize search using **pattern databases** or other heuristic enhancements.
* Introduce **user-controlled manual moves** for educational comparison with AI decisions.
* Add metrics like time taken, number of nodes expanded, and solution length.

**Output**







****