Assignment

Solving the 8-Puzzle Problem Using A* Algorithm

Submitted By: Gayan Ganapathi KS

4VV21EC039

Introduction

A sliding puzzle that constitutes a 3X3 grid together with tiles numbered from 1 to 8 and one blank space is called 8-puzzle problem, being a classic example of it. What needs to be done is moving these tiles around up to a point where they are supposed to be. In Artificial Intelligence this is the best area of search algorithms to explore especially with the efficiency of A* (A-star) algorithm in locating the fastest track to the answer.

Problem Formulation

1. **State Representation**: Each state of the puzzle can be represented by a 3x3 matrix where each cell contains a number from 1 to 8 or 0 to represent the blank space.
2. **Initial State**: This is the starting configuration of the puzzle.
3. **Goal State**: The target configuration we want to reach.
4. **Actions**: The possible moves are sliding a tile into the blank space. This can be represented as four possible actions: move blank up, down, left, or right.

Solution Methodology:

To solve the 8-puzzle problem, we use the A* algorithm, which is a best-first search algorithm that finds the least-cost path from a given initial state to the goal state. The key components of A* are:

- **Cost Function (f(n))**: This combines the actual cost to reach a node (g(n)) and the estimated cost to reach the goal from that node (h(n)):
  g(n)g(n)g(n): The cost to reach the current node from the start node.
  h(n)h(n)h(n): The heuristic estimate of the cost to reach the goal from the current node.

1. **Heuristic Function**: A* uses heuristics to estimate the cost to reach the goal. Common heuristics for the 8-puzzle include:

- **Manhattan Distance**: The sum of the absolute values of the differences in the tile's positions between the current state and the goal state. It is both admissible overestimates and consistent.
- **Misplaced Tiles**: The number of tiles that are not in their goal position.

Implementation Details

The implementation of the A* algorithm for solving the 8-puzzle problem involves the following steps:

1. **Node Structure**: Define a node structure to represent each state of the puzzle. Each node contains:

    - The current board configuration.
    - The cost to reach this state (g).
    - The heuristic estimate to the goal (h).
    - The total cost (f = g + h).
    - A pointer to the parent node for solution reconstruction.

2. **Minheap for Open Set**: A priority queue (min-heap) is used to manage the nodes to be explored based on their f values.
3. **State Expansion**: Expand the current node by generating its possible child nodes (resulting from valid moves of the blank space) and calculating their g, h, and f values.
4. **Goal Test**: Check if the current node matches the goal state. If yes, reconstruct the path from the start state to the goal state by following parent pointers.
5. **Closed Set**: Keep track of the visited states to avoid revisiting and expanding the same state.

Conclusion

The 8-puzzle problem exemplifies how heuristic search algorithms like A* can efficiently solve combinatorial problems. By leveraging a cost function that combines both the actual cost to reach a state and an estimated cost to reach the goal, A* navigates the state space effectively, ensuring that the shortest path to the solution is found. This approach, while implemented here for the 8-puzzle, is widely applicable to various other search problems in artificial intelligence.

References:

Nayak D and B Majhi 2014 Analysis and Implementation of Admissible Heuristics in 8 Puzzle Problem (Department of Computer Science and Engineering National Institute of Technology Rourkela Rourkela).

Tuccar M 2012 Analyzing the A* Search Heuristics with the Solutions of the 8 (University of Massachusetts Lowell).

Paderborn A R 1993 Complete Solution of the Eight-Puzzle and the Bene t of Node Ordering in IDA* (Center for Parallel Computing Warburger, Paderborn, Germany).