A* Search for 8-Puzzle Problem

This report presents a detailed formulation and implementation of the 8-Puzzle problem as a state-space search problem solved using the A* algorithm. Two heuristic functions are employed: (1) the number of misplaced tiles (h1) and (2) the total Manhattan distance (h2). The analysis includes theoretical justification, experimental results, and comparison of heuristic performance.

1. Problem Formulation

The 8-puzzle consists of a 3×3 grid containing 8 numbered tiles and one blank space (denoted as B). The objective is to move the tiles into the goal configuration using the fewest possible moves.

Goal State:

1 2 3
4 5 6
7 8 B

The problem can be formulated as follows:

Component	Description
State Representation	Each state is represented as a 9-element tuple of integers 0–8, where 0 denotes the blank space. Example: (1,2,3,4,5,6,7,8,0).
Start State	A random solvable configuration that is not equal to the goal state.
Goal State	The ordered configuration (1,2,3,4,5,6,7,8,0).
Actions	Moving the blank tile up, down, left, or right if the move is within the board boundaries. Each move has a unit cost.

Path cost function: Each move has a cost of 1. The total path cost g(n) is the number of moves from the start to the current state n.

2. Heuristic Functions

Two admissible heuristic functions were implemented for A*:

1. h1(n): Number of misplaced tiles (ignores the blank).
   h1(n) = count of tiles not in their goal position.

2. h2(n): Total Manhattan distance.
  h2(n) = sum of |row_i − row_goal| + |col_i − col_goal| for all tiles i.

Both h1 and h2 are admissible (never overestimate the true remaining cost). However, h2 is more informed, as it accounts for both horizontal and vertical displacements.

3. A* Algorithm Overview

A* is a best-first search algorithm that uses both the path cost (g) and heuristic estimate (h) to select the next node for expansion. The evaluation function is defined as:

- f(n) = g(n) + h(n)

At each step, A* expands the node with the smallest f(n). The algorithm guarantees finding the optimal solution if the heuristic is admissible and consistent.

4. Experimental Setup

The algorithm was implemented in Python. The board state is represented as a tuple of integers, and the open list is maintained as a priority queue using a min-heap. The algorithm was tested on randomly generated solvable initial states. Each trial reports the path cost (number of moves), nodes expanded, and total runtime.

Parameters:

Parameter	Value
Grid Size	3×3 (8-puzzle)
Cost Function	Unit cost per move
Goal Test	State == (1,2,3,4,5,6,7,8,0)
Max Nodes	500,000 per run

5. Example Run

A sample randomly generated start state and goal state are shown below:

Start State:
5 2 1
B 6 8
7 3 4

Goal State:
1 2 3
4 5 6
7 8 B

A* was executed with both heuristics. The results are summarized below:

Heuristic	Path Cost	Nodes Expanded	Runtime (s)	Solution Length

h1 - Misplaced Tiles	25	29,402	0.2615	25
h2 - Manhattan Distance	25	3,665	0.0380	25

Both heuristics found the optimal solution (25 moves). However, the Manhattan distance heuristic (h2) was significantly more efficient, expanding nearly eight times fewer nodes and requiring much less computation time.

6. Comparative Analysis

The difference in node expansion is attributed to heuristic accuracy. The misplaced-tile heuristic (h1) provides a coarser estimate — it only measures how many tiles are incorrect but not how far they are. The Manhattan distance heuristic (h2), by considering the total distance each tile must travel, offers a more precise estimate of the remaining cost. Consequently, A* guided by h2 explores fewer nodes and converges faster to the goal.

Empirically, h2 consistently results in fewer node expansions and lower runtime, while maintaining the same optimality guarantee. Therefore, h2 is the preferred heuristic for the 8-puzzle and similar grid-based pathfinding problems.

7. Conclusion

The A* algorithm effectively works when admissible heuristics are used to solve the 8 puzzle problem. Both h1 (misplaced tiles) and h2 (Manhattan distance) are optimal but h2 has better performance, as it provides a closer approximation to the actual remaining cost. This paper illustrates the effect of heuristic quality in the search efficiency of AI search heuristic-based techniques.