Performance Evaluation of Puzzle Solving Algorithms

Ahmed Ghoneim

4/21/2023

Abstract

This report presents a comparative analysis of three search algorithms – Breadth-First Search (BFS), Depth-First Search (DFS), and Dijkstra's algorithm – in solving the 8-puzzle problem. We evaluate the performance of these algorithms based on their efficiency, optimality, and resource usage. The findings of this study will help in selecting an appropriate algorithm for solving the 8-puzzle problem based on specific requirements and constraints.

Introduction

The 8-puzzle is a classic combinatorial problem that consists of a 3x3 grid containing eight numbered tiles and a blank space. The objective is to rearrange the tiles from a given initial configuration to a specified goal state by sliding the blank space horizontally or vertically adjacent to the numbered tiles. In this report, we examine the performance of three search algorithms – Breadth-First Search (BFS), Depth-First Search (DFS), and Dijkstra's algorithm – in solving the 8-puzzle problem. The report is structured as follows: Section 2 describes the 8-puzzle problem and the search algorithms used, Section 3 presents the experimental setup, Section 4 discusses the results, and Section 5 concludes the report with a summary of the findings.

Background

2.1. The 8-Puzzle Problem

The 8-puzzle problem is a classic sliding puzzle that has been widely studied in the field of Artificial Intelligence (AI) as a benchmark for search algorithms. The puzzle consists of a 3x3 grid containing eight numbered tiles and a blank space, with the goal of rearranging the tiles from a given initial configuration to a specified goal state by sliding the blank space horizontally or vertically adjacent to the numbered tiles. The 8-puzzle problem is an instance of a more general class of sliding puzzles, which can have different grid sizes and number of tiles, such as the 15-puzzle (a 4x4 grid with fifteen numbered tiles and a blank space).

2.2. Search Algorithms

In this study, we consider three search algorithms for solving the 8-puzzle problem: Breadth-First Search (BFS), Depth-First Search (DFS), and Dijkstra's algorithm. Each algorithm has its own set of characteristics that may affect their performance in solving the problem.

2.2.1. Breadth-First Search (BFS)

BFS is a graph traversal algorithm that explores all the vertices of a graph in breadth-first order, i.e., it visits all the vertices at the same level before moving on to the next level. In the context of the 8-puzzle problem, BFS guarantees an optimal solution (i.e., the shortest sequence of moves to reach the goal state) but may have high memory usage and execution times due to the need to store and process a large number of states in the search tree.

2.2.2. Depth-First Search (DFS)

DFS is another graph traversal algorithm that explores all the vertices of a graph in depth-first order, i.e., it visits a vertex and recursively explores all its adjacent vertices before backtracking. In the context of the 8-puzzle problem, DFS is more resource-efficient than BFS, as it does not need to store all the visited states in memory. However, DFS may produce suboptimal solutions (i.e., longer sequences of moves to reach the goal state) due to its depth-first exploration strategy.

2.2.3. Dijkstra's Algorithm

Dijkstra's algorithm is a shortest-path algorithm that finds the shortest path from a source vertex to all other vertices in a weighted graph. In the context of the 8-puzzle problem, Dijkstra's algorithm can be adapted to find the shortest sequence of moves to reach the goal state by treating each state as a vertex and each move as an edge with a unit weight. Dijkstra's algorithm guarantees an optimal solution and has relatively lower memory usage compared to BFS. However, its execution time may be longer than DFS due to the use of a priority queue for processing states in the search tree.

Experimental Setup

To evaluate the performance of the three search algorithms, we implemented them in Python and applied them to a set of randomly generated 8-puzzle instances. For each algorithm, we measured the following metrics:

Solution optimality: Whether the algorithm found the shortest sequence of moves to reach the goal state.

Execution time: The time taken by the algorithm to find a solution.

Memory usage: The amount of memory used by the algorithm during the search process.

We also compared the algorithms based on their ability to solve the 8-puzzle instances within a reasonable amount of time and resources, considering the trade-offs between optimality, execution time, and memory usage.

Results and Discussion

The results of our experiments showed that:

BFS consistently found optimal solutions for all the tested instances. However, its memory usage and execution times were significantly higher compared to DFS and Dijkstra's algorithm, especially for more complex instances.

DFS had the lowest memory usage and the fastest execution times among the three algorithms. However, it often produced suboptimal solutions, with the number of moves to reach the goal state sometimes being much higher than the optimal solution.

Dijkstra's algorithm found optimal solutions for all the tested instances, with lower memory usage compared to BFS. Its execution times were longer than DFS but shorter than BFS.
Based on these results, we can draw the following conclusions:

If optimality is the primary concern and resource usage is not a constraint, BFS is the most suitable algorithm for solving the 8-puzzle problem.
If resource efficiency (i.e., low memory usage and fast execution times) is the primary concern and optimality is not a constraint, DFS is the most suitable algorithm.
If a balance between optimality and resource efficiency is desired, Dijkstra's algorithm is the most suitable algorithm for solving the 8-puzzle problem.

Conclusion

In this report, we presented a comparative analysis of three search algorithms – Breadth-First Search (BFS), Depth-First Search (DFS), and Dijkstra's algorithm – in solving the 8-puzzle problem. Our experiments showed that BFS consistently found optimal solutions but had high memory usage and execution times, DFS had the lowest memory usage and the fastest execution times but often produced suboptimal solutions, and Dijkstra's algorithm provided a balance between optimality and resource efficiency. These findings can be useful for selecting an appropriate algorithm for solving the 8-puzzle problem based on specific requirements and constraints.