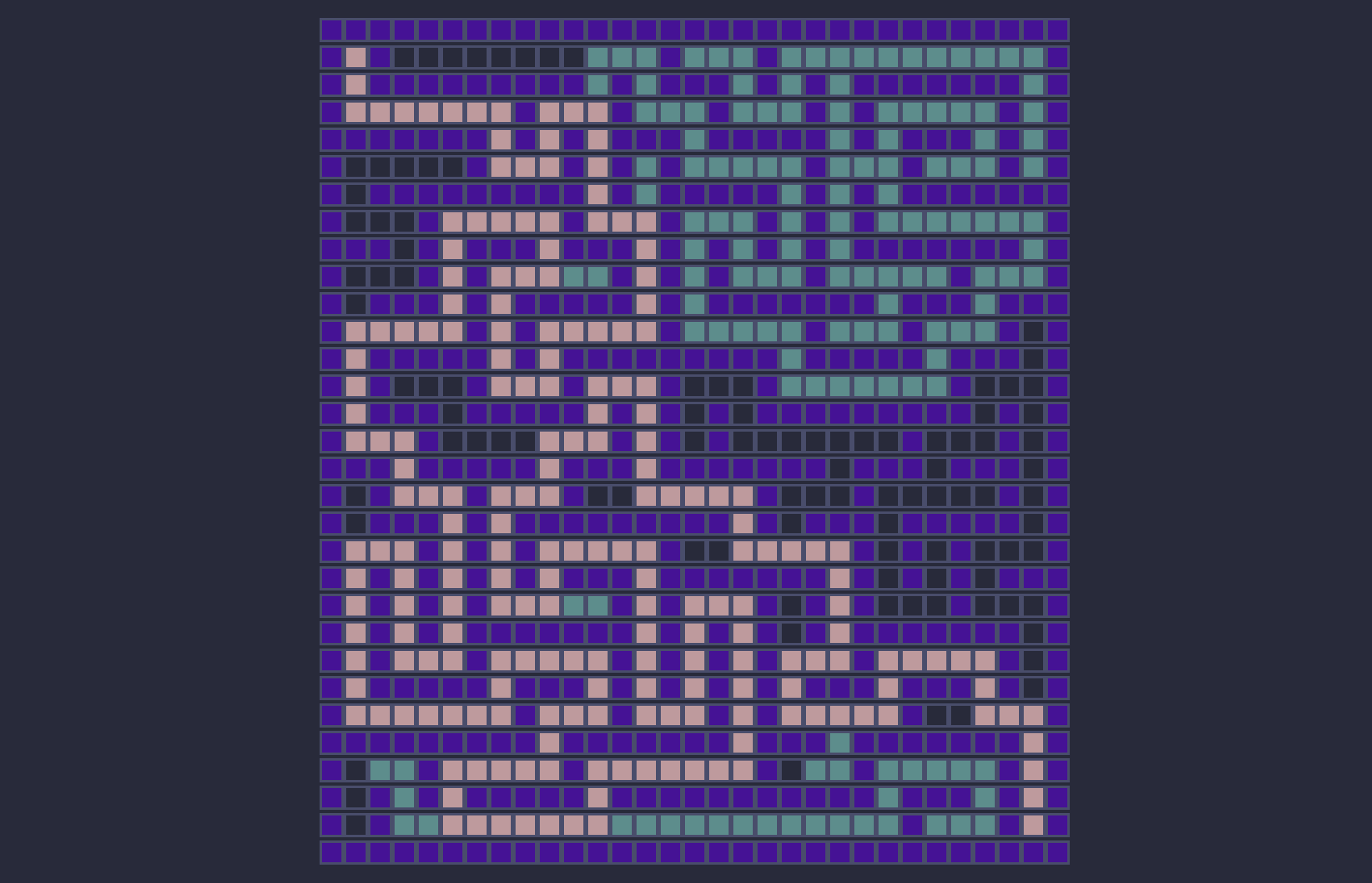
**Conclusions**  
For single-solution mazes, DFS excelled in execution time and memory usage. A\* was best for multiple solutions, consistently finding near-optimal paths. The Wall Follower showed that basic algorithms can be effective, while Dijkstra’s and BFS proved reliable for finding optimal paths.

**Generating and Visualizing Mazes**

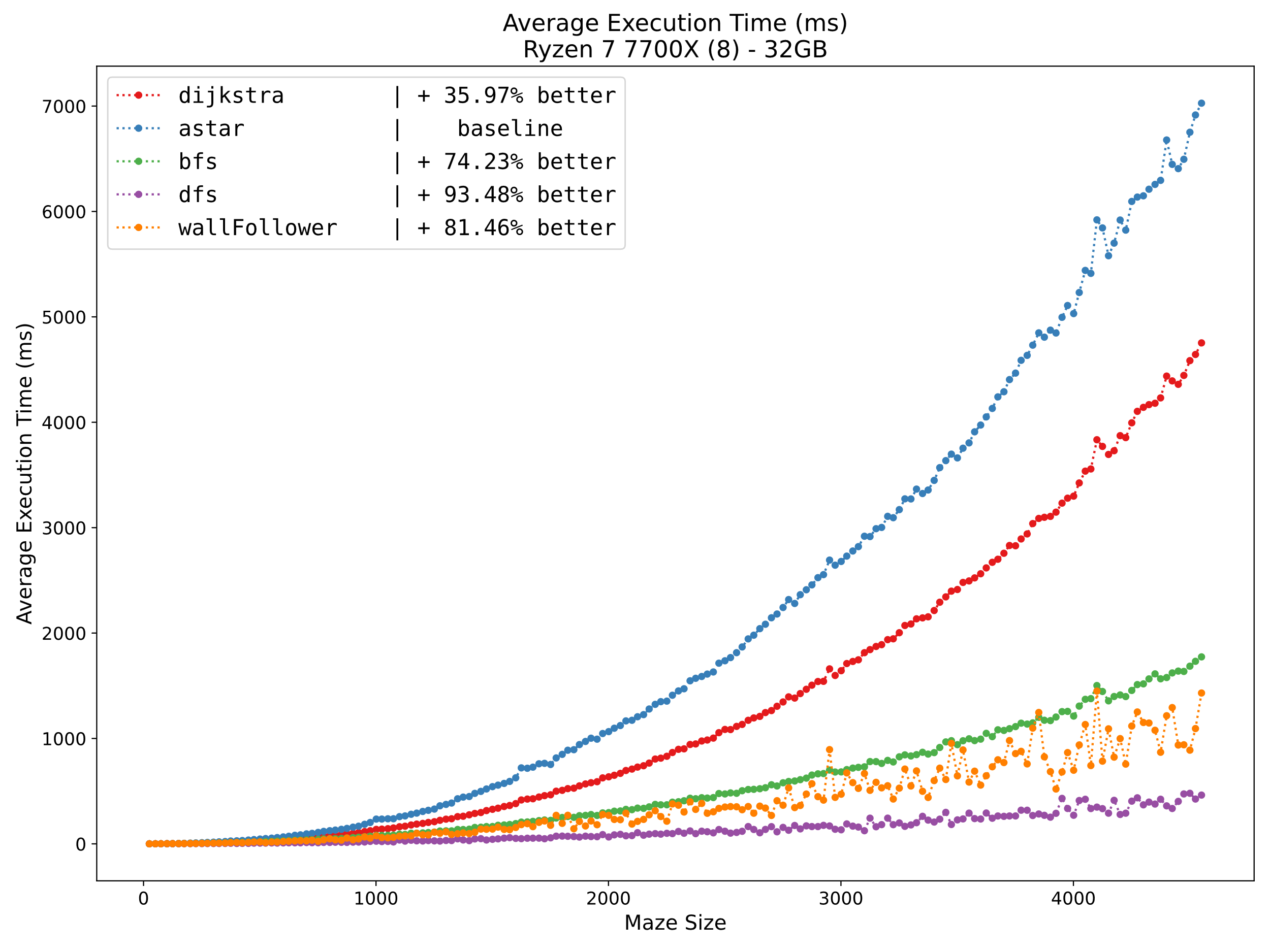
Our application uses a React.js frontend for dynamically visualizing maze-solving algorithms and a Go-based backend for generating mazes and running the algorithms. The backend handles concurrency efficiently with goroutines, speeding up the maze creation and solving process. We used a recursive backtracker algorithm to generate mazes, introducing randomness to add complexity. The frontend fetches maze data from the backend API, visualizing the maze grid and the algorithm’s progress step-by-step.  


Visualization of a Solved Maze

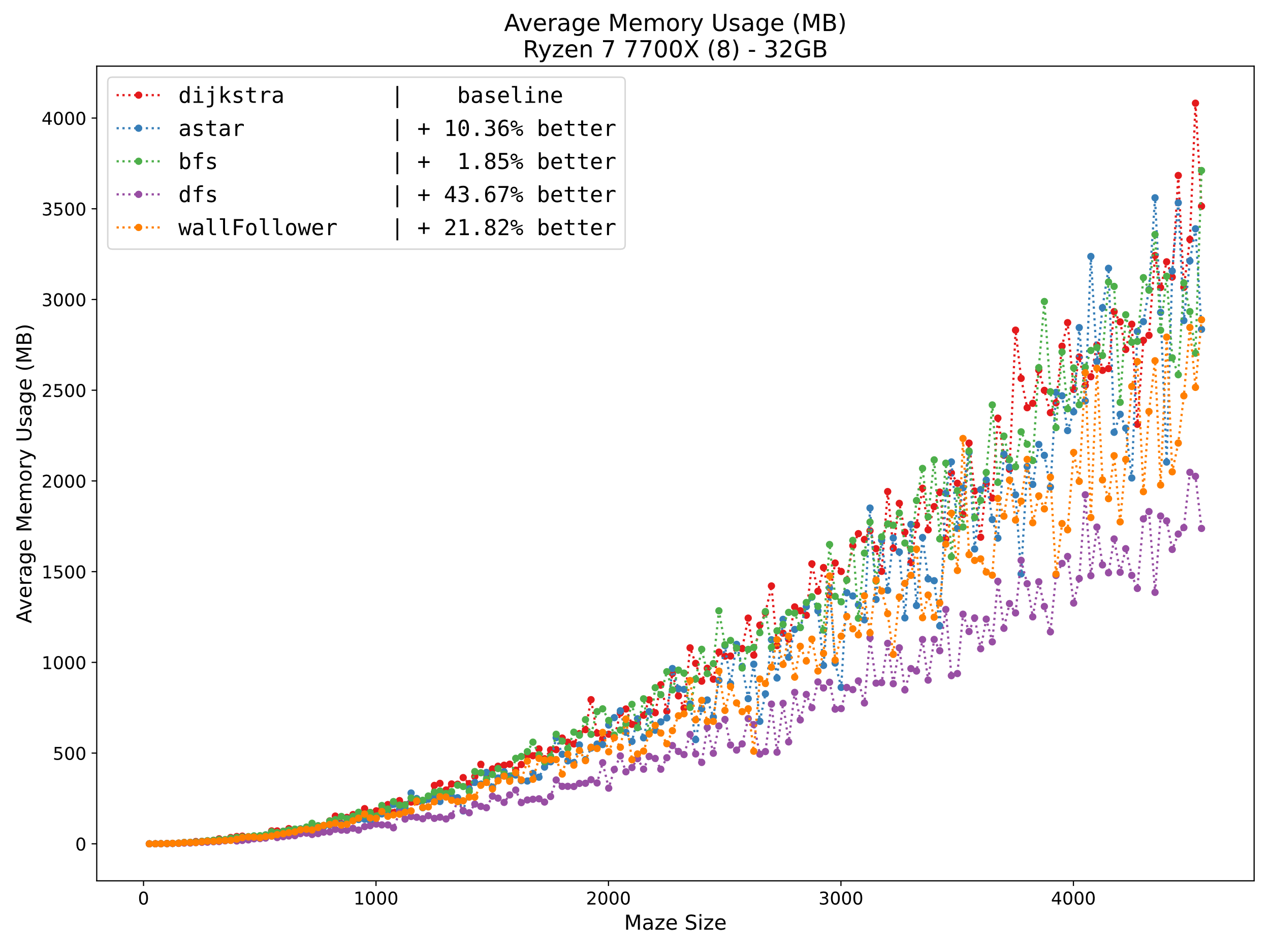
**Analysis**

Dijkstra’s Algorithm and BFS consistently found the shortest path, while A\* achieved optimal paths about 46.07% of the time, with an impressive average path length delta of only 0.61%. This low delta indicates A\*'s efficiency and potential for improvement through better heuristics, as it explored fewer nodes compared to others.

Pathfinding algorithms are used to navigate environments and find efficient paths between two points, like in Google Maps. We compared five algorithms—Dijkstra’s, A\*, DFS, BFS, and Wall Follower—by implementing them in a maze and evaluated their performance based on the optimality of the path, their speed, memory usage and percentage of the maze visited.

DFS and Wall Follower were less accurate, with DFS averaging about 50.92% for visited nodes. Dijkstra’s and BFS explored the most nodes, averaging 96-99% for single-solution mazes.In exe­­­­­cution time, DFS was the fastest, while A\* was the slowest.

DFS also used the least memory, whereas BFS was the most memory-intensive.



**COMPARISON OF PATHFINDING ALGORITHMS**

**Lovro Hauptman, Aleš Volčini**