JUMP POINT SEARCH PATHFINDING IN 4-CONNECTED GRIDS

By The Cooks

INTRODUCTION

This project has two parts:

1) Test and benchmark JPS4 against the A* algorithm across varying grid sizes and obstacle densities

2) Implement a real-time Snake game simulator driven by the JPS4 pathfinder

BACKGROUND

A*: A classic priority-queue search on a 4-connected grid that expands every traversable neighbor.

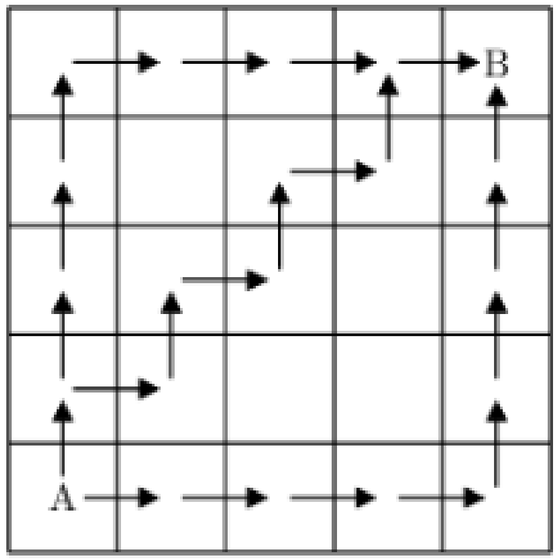
JPS8: Extends A* with 8-way canonical ordering, online neighbor pruning, and recursive “jumps” to skip symmetric nodes.

WHAT JPS4 PAPER PROPOSES

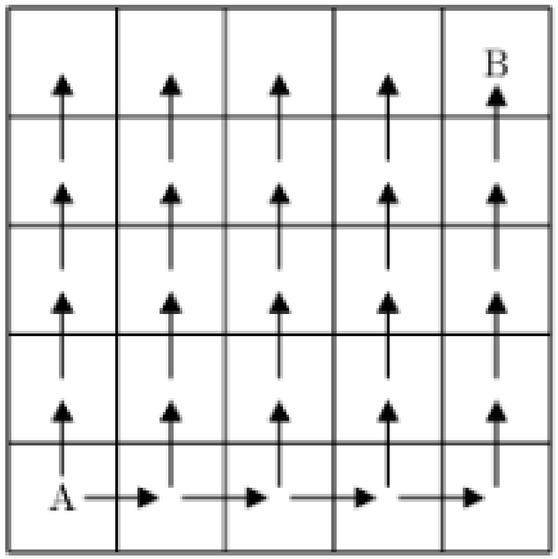
- **Canonical Ordering**: Enforces a horizontal-first rule to collapse symmetric paths into a single tree.
- **Neighbor Pruning**: Keeps only “natural“ and obstacle-induced “forced“ neighbors to cut redundant branches.
- **Jump Function**: Recursively skips straight lines until hitting an obstacle, forced neighbor, or goal, marking jump points.
- **Successor Generation**: Combines pruning + jumping so A* only expands strategic jump points, not every adjacent cell.
- **Optimizations**: Online graph pruning and directional jumps yield the same optimal paths as A* with far fewer expansions.

ALGORITHMN OVERVIEW

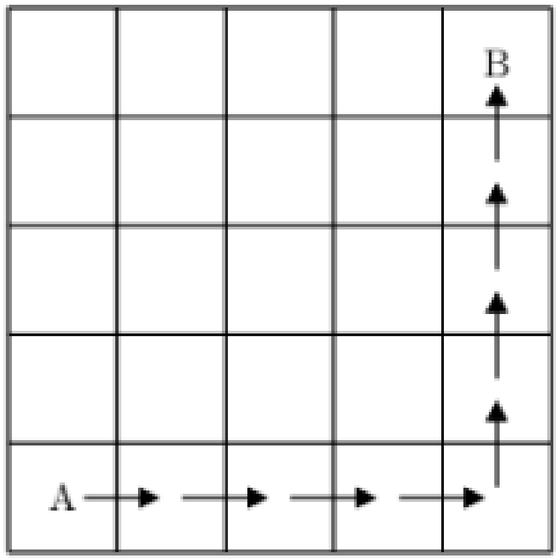
CANONICAL ORDERING:



(a) Some symmetric paths from A to B



(b) Applying our canonical ordering converts the search space from a graph into a tree by eliminating redundant paths.

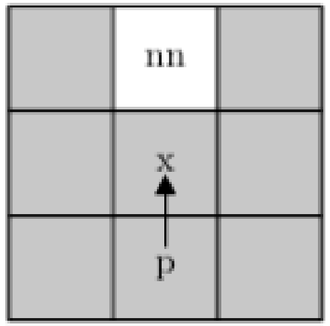


(c) The search space that results from applying our canonical ordering (figure 1b) yields only one path from A to B.

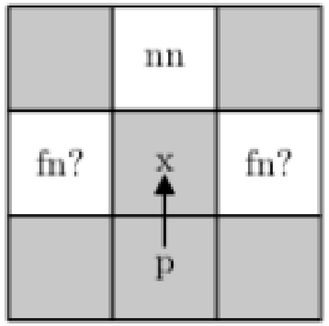
Figure 1: Elimination of symmetric paths in obstacle-free maps via canonical ordering.

ALGORITHMN OVERVIEW

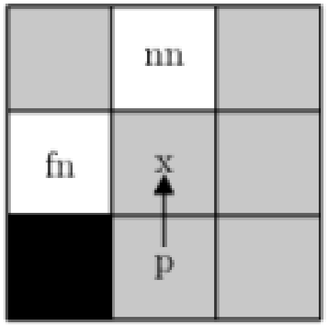
NEIGHBOR PRUNING:

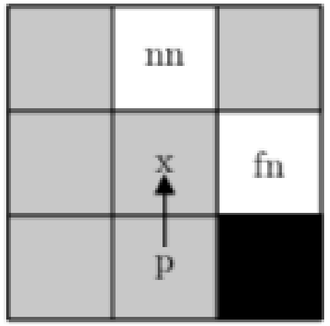


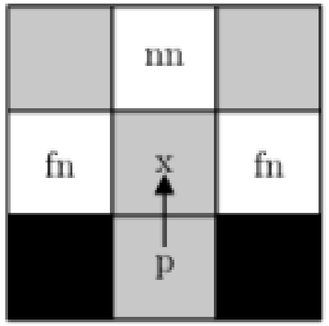
(a) There is only one natural neighbor for a vertical movement.



(b) All neighbors that are not natural neighbors in a vertical movement are potentially forced neighbors.





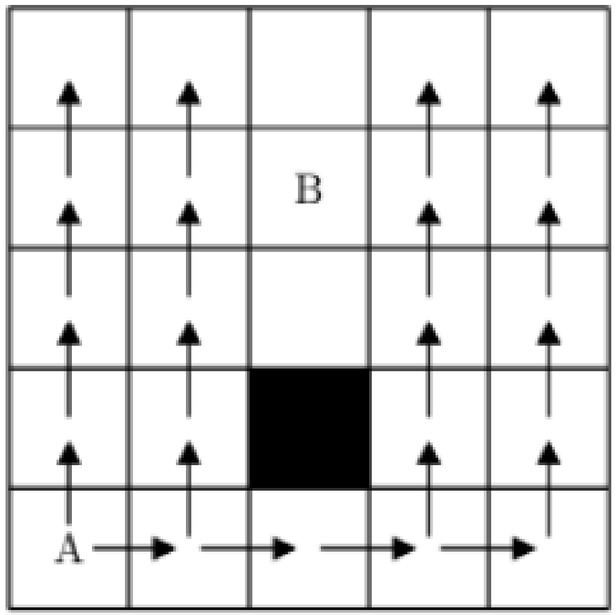


(c) Natural and forced neighbors for vertical movements in case of obstacles

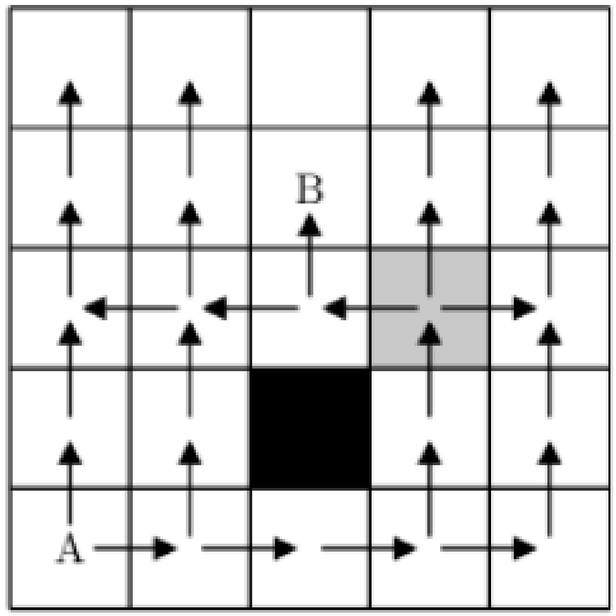
Figure 2: Natural and forced neighbors for vertical movements (black square: obstacle, p: parent node, x: current node, nn: natural neighbor, fn?: potentially forced neighbor, fn: forced neighbor)

ALGORITHMN OVERVIEW

JUMP POINTS:

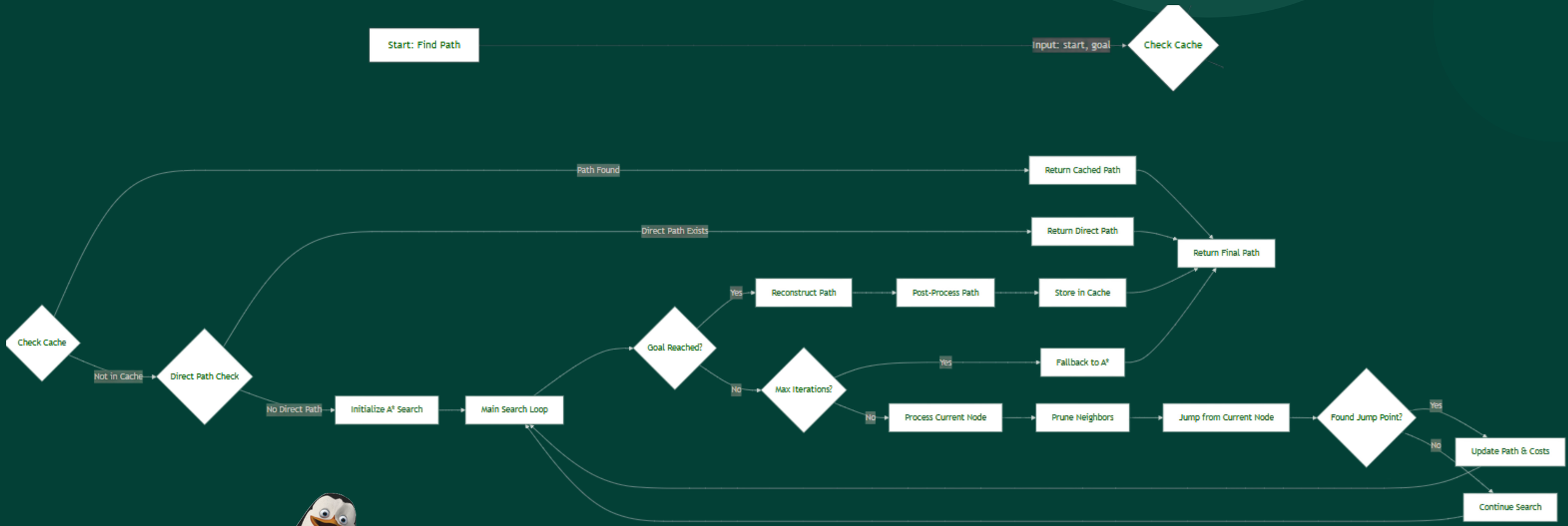


(a) Our canonical ordering misses the path from A to B because of an obstacle.



(b) The introduction of a jump point that resets the canonical ordering fixes the search space such that target B is now reachable again.

ALGORITHMN OVERVIEW



COMPLEXITY ANALYSIS

Worst case: O(n) where n is grid size (same as A)

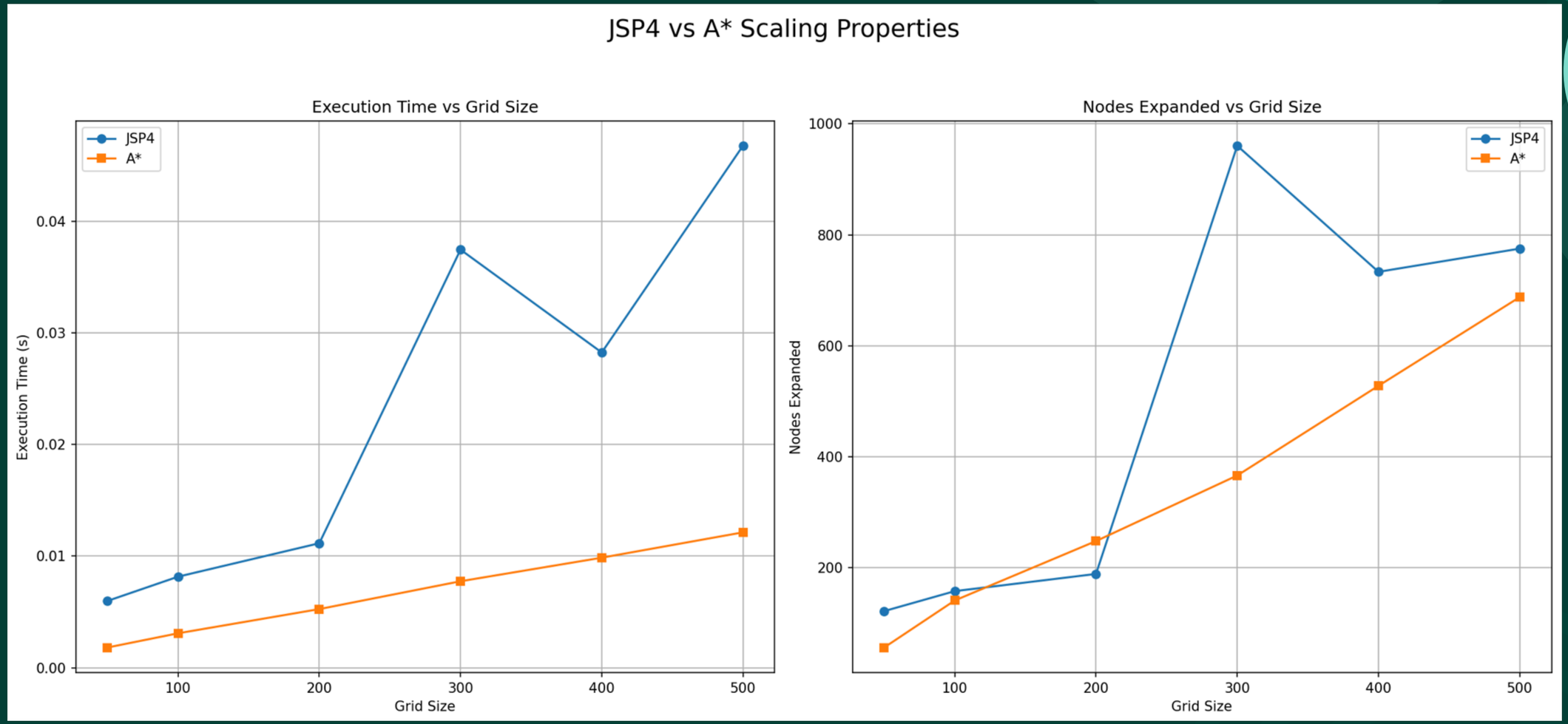
Space Complexity:
O(n) for the open/closed lists (same as A)

Our implementation:
Currently slower than A* across all benchmarks

RESULTS

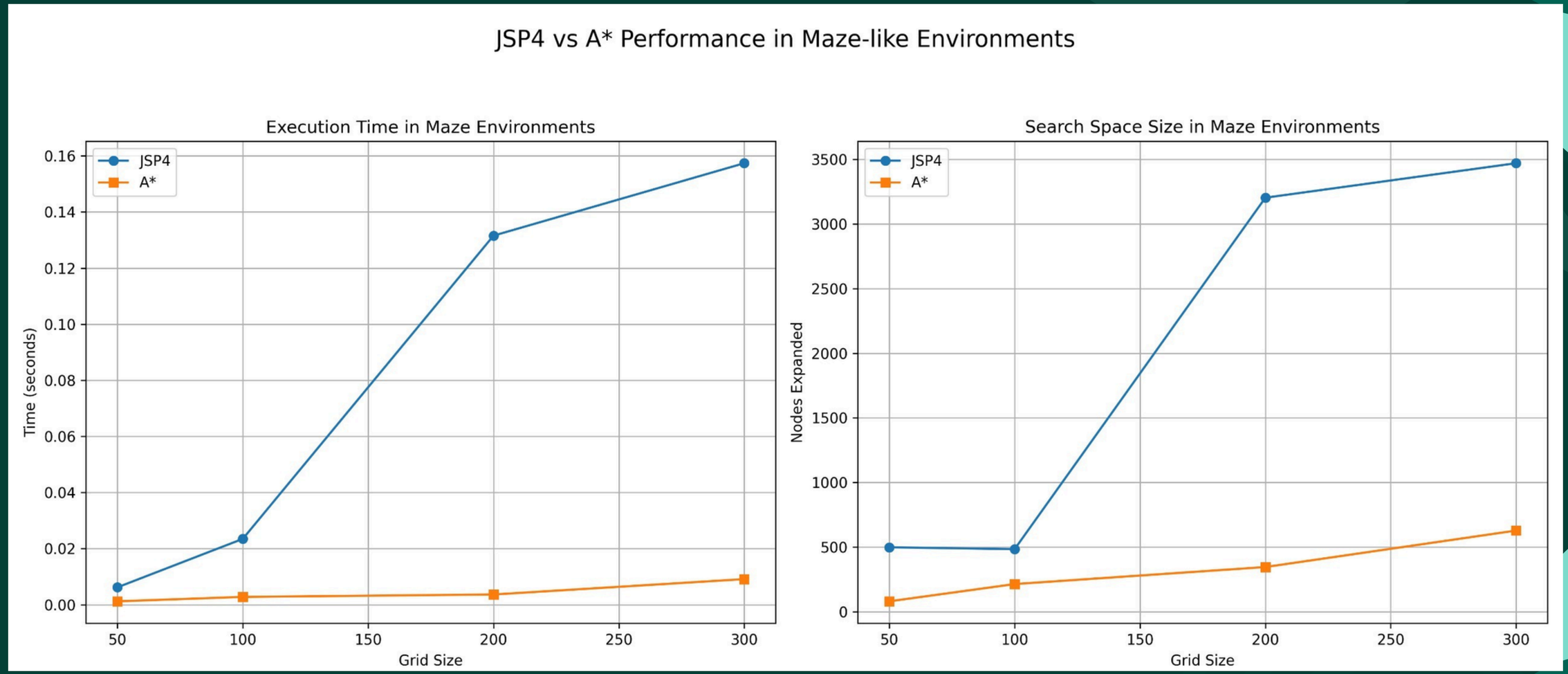
1. Conducted tests on a wide range of grid sizes (50×50 to 500×500)
2. Evaluated performance on various grid types including maze-like environments
3. Despite theoretical advantages, observed suboptimal performance across all test scenarios

RESULTS



Comparing A* with JPS4 on Grid like

RESULTS



Comparing A* with JPS4 on Maze like

DEMO

ASSUMPTIONS+LIMITATIONS

Algorithm Assumptions:

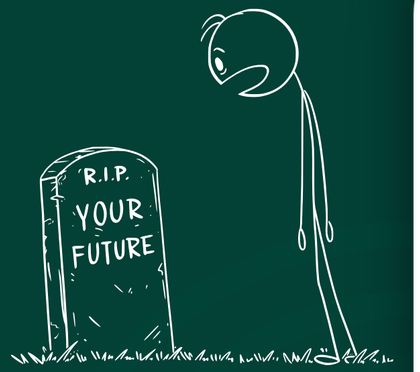
- Uniform cost for all movements (doesn't handle weighted grids)
- Requires 4-connected movement only (no diagonal movement)
- Not optimized for dynamic environments with changing obstacles

Current Limitations:

- Implementation performs slower than A* in all benchmark scenarios
- Pathfinding efficiency decreases significantly in high-density grids (>60% obstacles)
- Fails to achieve theoretical performance advantages in practice

FUTURE WORK

1. Optimize jump point detection with precomputation caching
2. Introduce specialized handling for corridor-rich environments
3. Implement adaptive pruning based on grid density
4. Add parallel processing for large-scale grid environments
5. Integrate with hierarchical pathfinding for multi-level optimization



THANK YOU

REFERENCES

- Johannes Baum, Jump Point Search Pathfinding in 4-connected Grids, arXiv:2501.14816v1,January 28, 2025.
- Van der Woude, T. B. (n.d.). grid_pathfinding [Source code]. GitHub. Retrieved April 27, 2025, from https://github.com/tbvanderwoude/grid_pathfinding