

Pathfinding Optimization: Comparative Study of Jump-Point Search and Classical Search Algorithms

Sepehr Mansouri

BCIT

Burnaby, BC, Canada

smansouri7@my.bcit.ca

Binger Yu

BCIT

Burnaby, BC, Canada

gyu42@my.bcit.ca

Yansong Jia

BCIT

Burnaby, BC, Canada

yjia16@my.bcit.ca

Vibhor Malik

BCIT

Burnaby, BC, Canada

vmalik6@my.bcit.ca

1 Problem and Motivation

Efficient pathfinding is critical for robotics, autonomous vehicles, and real-time gaming systems, where navigation delays can mean the difference between mission success and failure. Classical algorithms including A* Dijkstra's, and Depth-First Search (DFS) have been widely used for optimal pathfinding but often struggle with exponential time complexity in large or complex grids [1, 3]. Jump-Point Search (JPS) offers a promising optimization technique that reduces redundant node expansions and speeds up search without losing accuracy [4]. However, comprehensive empirical analysis comparing its performance across varying grid complexities and real-world constraints remains limited. This study provides empirical evidence to evaluate JPS against traditional algorithms, identifying performance gains, trade-offs, and suitability for time-critical pathfinding applications.

2 Modern Algorithm Selection

In addressing pathfinding limitations, researchers have developed multiple optimization techniques aimed at improving search. One of the most effective of these modern approaches is Jump-Point Search (JPS). JPS is a modern, optimal pathfinding algorithm designed and introduced by Daniel Harabor and Alban Grastien in their 2011 paper, "Online Graph Pruning for Pathfinding on Grid Maps," which presented the core JPS algorithm and its symmetry-breaking approach, a method ignoring duplicate paths that differ only in move order, achieving optimal path computation without preprocessing or memory overhead. JPS functions as a macro operator that identifies and expands only specific "jump points" while skipping intermediate nodes between consecutive jump points. [5, 6]. The algorithm's core innovation addresses path symmetry in uniform-cost grids where multiple equivalent paths exist between nodes that differ only in move ordering. A uniform-cost grid would mean that each step in any direction costs the same amount, such as moving one square at a time in a chessboard. JPS eliminates redundant exploration through a pruning strategy that manipulates blocks of nodes simultaneously rather than expanding on each node individually.

This algorithm eliminates such redundancies through two methods: Neighbour Pruning, used to eliminate nodes that can be reached optimally for a parent node without traversing the current node, and Jumping, allowing the search to skip over straight lines (Horizontal,

vertical, and diagonal) in the grid until it reaches obstacle boundaries [7]. JPS significantly improves upon traditional algorithms by achieving order-of-magnitude speedups over other algorithms through dramatically reduced node expansions. By jumping over symmetric path segments, JPS maintains optimal solutions without preprocessing or memory overhead, performing especially well in open areas with prevalent symmetries while remaining effective across diverse map topologies [8]. The project is to apply grid-based pathfinding optimization in a 2D uniform-cost environment, comparing it against classical search algorithms (A*, Dijkstra's, and DFS) across various grid configurations and obstacle densities. The experimental framework will measure the runtime performance, node expansion efficiency, and path optimality to quantify the practical benefits of symmetry-breaking techniques in real-world pathfinding applications.

3 Baseline Selection

To contextualize JPS's improvements, it is essential to review the foundational algorithms that defined classical pathfinding methods. In 1959, Moore brought up "The Shortest Path Through a Maze" in the Proceedings of the IRE, an early application of pathfinding principles to maze problems [9]. Tarjan published "Depth-First Search and Linear Graph Algorithms" in the SIAM Journal on Computing, where he constructed linear-time algorithms for key graph-theoretic tasks such as strongly connected components, bridges, articulation points, and cycle detection, establishing the DFS' theoretical rigor [3].

In 1959, Dijkstra introduced his famous shortest-path algorithm. It generates optimal solutions in graphs with non-negative edge weights and eliminates redundant explorations. This innovation then became fundamental in engineering applications such as network routing [2]. Gallager, Humblet, and Spira later validated his work while proposing the distributed minimum spanning tree algorithm [10].

In 1968, Hart, Nilsson, and Raphael published "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", introducing the A* search algorithm. It combines path cost (defined by g-cost) with heuristic estimation (h-cost) to explore efficiently and reduce Dijkstra's redundancy [1]. Dechter and Pearl further generalized this framework in "Generalized Best-First Search Strategies and the Optimality of A*". It proved the algorithm's optimality under admissible (and sometimes consistent) heuristics and classified its node

expansion behavior, which has facilitated A* to be the theoretical standard for heuristic search [11].

4 Research Questions

Classical algorithms such as A*, Dijkstra's, and DFS provide the theoretical foundation for optimal pathfinding but face computational limitations in large or complex grids. This study explores how JPS improves pathfinding efficiency compared to these classical methods. Specifically, we investigate whether JPS reduces runtime and node expansions by cutting away or pruning unnecessary paths and skipping redundant nodes, while maintaining the same optimal path cost as A* on uniform-cost grids. We will also examine how grid size and obstacle density affect JPS performance, identifying whether it performs better in sparse or dense environments. Finally, we assess the trade-offs and limitations of using JPS in real-time or dynamic contexts, where preprocessing overhead may reduce responsiveness. These questions guide our experimental design and comparative evaluation with traditional pathfinding algorithms.

5 Team Plan and Timeline

The project will be conducted collaboratively with clearly defined roles and a structured timeline to ensure systematic progress. Yansong leads the implementation of baseline algorithms. Sepehr develops and tests the modern JPS algorithm. Vibhor handles evaluation and visualization, analyzing runtime, node expansion, and performance metrics. Binger, as the Project Manager, will be overseeing documentation, report writing, and the final presentation, ensuring coordination and timely completion of all milestones.

To facilitate collaboration, the team uses a shared GitHub repository for version control and joint development. The timeline includes proposal submission in Week 8 (Oct 21), prototype testing by Week 12 (Nov 18), draft report and slides by Week 13 (Nov 25), and final submission and presentation in Week 14 (Dec 2). This schedule supports consistent progress and on-time delivery of the final project. A breakdown of each milestone and their tasks can be found under Appendix A & B.

References

- [1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [2] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
- [3] R. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, 1972.
- [4] D. Harabor and A. Grastien, "Online Graph Pruning for Pathfinding on Grid Maps," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2011, pp. 1114–1119.
- [5] D. Harabor and A. Grastien, "Online graph pruning for pathfinding on grid maps," in *Proceedings of the 25th AAAI Conference on Artificial Intelligence & the 23rd Innovative Applications of Artificial Intelligence Conference (AAAI-11/IAAI-11)*, 2011, pp. 1114–1119.
- [6] D. Harabor and A. Grastien, "The JPS pathfinding system," in *Proceedings of the 5th Annual Symposium on Combinatorial Search (SoCS 2012)*, Niagara Falls, Canada, Jun. 2012.
- [7] D. Harabor, "Jump Point Search," *Shortest Path* (blog), Sept. 7, 2011. [Online]. Available: <https://harablog.wordpress.com/2011/09/07/jump-point-search/>. [Accessed: Oct. 18, 2025].
- [8] S. Rabin and F. Silva, "JPS+ – An Extreme A* Speed Optimization for Static Uniform Cost Grids," in *Game AI Pro 360: Guide to Movement and Pathfinding*, S. Rabin, Ed. Boca Raton, FL: CRC Press, 2019, pp. 95–108.
- [9] E. F. Moore, "The shortest path through a maze," *Proc. IRE*, vol. 47, no. 10, pp. 1785–1786, 1959.
- [10] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Trans. Program. Lang. Syst.*, vol. 5, no. 1, pp. 66–77, 1983.
- [11] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Artificial Intelligence*, vol. 23, no. 1, pp. 1–38, 1985.

Appendices

A Team Roles and Responsibilities

Member	Role	Key Tasks	Timeline
Yansong	Algorithm developer Lead	A*, DFS, Dijkstra's implementation & optimization. Implement baseline A*, DFS and Dijkstra's algorithm, test heuristic variations, and validate outputs.	Oct 15 – Oct 27
Sepehr	QA & Testing Lead	JPS Implementation & Testing. Implement JPS algorithm; integrate with existing grid; compare performance with A*, DFS, and Dijkstra's.	Oct 25 – Nov 15
Vibhor	Evaluation Lead	Metrics and Visualization. Collect and node expansion data; generate charts and comparison tables.	Nov 10 – Nov 24
Binger	Documentation & Presentation	Report & Preparation. Draft proposal and final paper, create visuals and slides, record demo videos.	Oct 15 – Dec 2

Table 1: Team Roles and Responsibilities

Key Milestones:

- Proposal due → Oct 21 (Week 8)
- Testing → Nov 18 (Week 12)
- Slides + README → Nov 25 (Week 13)
- Presentation + Final Paper → Dec 2 (Week 14)

B Project Timeline and Deliverables

Week	Date Range	Task	Deliverable / Notes
7	Oct 14 – Oct 20	Kick-off meeting (Oct 15); finalize topic and environment setup; Finalize proposal; assign team roles and create GitHub repository.	Baseline implementation (DFS, A*, Dijkstra), code repo initialized, Development environment defined
8	Oct 21 – Oct 27	Write and submit Research Proposal (1–2 pages) – due Oct 21 @ 11:59 PM; start reading JPS papers and preparing comparison metrics.	Completed proposal with defined problem statement, baselines (A*, Dijkstra, DFS), and assigned team responsibilities.
9-10	Oct 28 – Nov 10	Implementation of JPS, A*, Dijkstra, & DFS in a 2D grid space.	Working JPS prototype and initial performance comparison with A*, Dijkstra and DFS;
11-12	Nov 11 – Nov 24	Experimentation and Evaluation: run large-scale tests comparing A* vs JPS (runtime, path cost, convergence, adaptability); log metrics and generate plots. Draft of results log.	Performance dataset of performance metrics; visualizations (heatmaps, paths, overlays, bar charts), and written results and analysis section.
13	Nov 25 – Dec 1	Finalize paper, slides, and code. Integrate figures and tables; rehearse presentation.	README + IEEE-style paper draft + PPT slides – internal deadline due Nov 25.
14	Dec 2 (Week 14)	Final Presentation and Submission. Deliver 15–20 minute presentation + Q&A; submit final 4–5 page report.	Final deliverables submitted (paper + slides + demo video + code).

Table 2: Project Timeline and Deliverables