WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Pathfinder Pro: Implementation of Various Pathfinding Algorithms

***Anchal Tiwari[1], Ankit Gupta[2], Alok Patel[3], Rajeshri Lanjewar[4], Phanish Kumar Sahu[5]***

[1]Dept. of Computer Science and Engineering Shri Shankaracharya Technical Campus Bhilai, India anchal.tiwari.sstc.cse@gmail.com
[2]Dept. of Computer Science and Engineering Shri Shankaracharya Technical Campus Bhilai, India ankit.gupta.sstc.cse@gmail.com
[3]Dept. of Computer Science and Engineering Shri Shankaracharya Technical Campus Bhilai, India alokpatel2810@gmail.com
[4]Dept. of Computer Science and Engineering Shri Shankaracharya Technical Campus Bhilai, India rajeshri21@gmail.com
[5]Dept. of Computer Science and Engineering Shri Shankaracharya Technical Campus Bhilai, India phanishsahu19@gmail.com

**ABSTRACT-**

Pathfinder Pro provides an interactive platform for visualizing and analyzing various pathfinding algorithms, including Dijkstra's, A*, Swarm Algorithms, Greedy Best-First Search, and Depth-First Search. Built using React.js and JavaScript, it allows users to create dynamic grids, place obstacles, and observe real-time algorithm behavior. It calculates performance metrics such as path cost, computation time, and visited nodes, helping users evaluate efficiency and effectiveness. The system aims to serve educational and practical purposes, with future plans for three dimensional visualization and dynamic obstacle handling. The live demo is available at: **https:// pathfinder-pro-visualizing-algorithms**

***Keywords—Pathfinding, Visualization, Dijkstra, A*,Greedy, Swarm Algorithm, JavaScript, React.js, Grid Navigation, Real- Time Algorithms, DFS, BFS.***

## *I.* INTRODUCTION

Pathfinding algorithms are essential in fields like robotics, navigation, and gaming, enabling intelligent decision-making in complex environments. Pathfinder Pro is an interactive platform that visualizes and analyzes key pathfinding algorithms, including Dijkstra's, A*, Greedy Best-First Search, Swarm Algorithms, and Depth-First Search. Developed using React.js and JavaScript, the system allows users to create dynamic grids, place obstacles, and observe algorithm behavior in real time.

By providing real-time visualizations and performance metrics such as path cost and computation time, Pathfinder Pro bridges the gap between theoretical learning and practical application, making it a valuable educational and research tool. Planned enhancements include 3D visualizations and dynamic obstacle handling to extend its real-world relevance. The platform features an intuitive interface and supports easy integration of new algorithms, promoting scalability and algorithmic learning.

## II. LITERATURE REVIEW

Pathfinding algorithms are foundational in applications spanning robotics, gaming, and artificial intelligence. Over the years, researchers have developed a variety of approaches for efficiently solving shortest path problems in both static and dynamic environments.

Sharma et al. (2023) introduced an early version of a Pathfinding Visualizer that aimed to simplify the understanding of algorithmic behavior through an interactive interface. Their work served as a pedagogical tool and inspired further advancements in visualization-based learning platforms such as Pathfinder Pro [1].

Singh et al. (2022) further developed the concept with their Node Path Visualizer, focusing on helping students explore algorithms like Dijkstra and A* interactively. Their tool laid the groundwork for more technically robust platforms [2].

Mandloi et al. (2022) emphasized real-time rendering for improving user intuition, reinforcing the importance of visual feedback in education [3].

Dijkstra's foundational work (1959) remains the backbone of many pathfinding algorithms, offering a guaranteed optimal path through exhaustive search [4]. Ahuja et al. (1993) expanded on this with practical applications in network flow and algorithm design [5].

A* (A-Star) Search, which enhances Dijkstra's algorithm using heuristics, remains one of the most balanced approaches in terms of speed and optimality. Mapaila (2012) and Halldorsson (2006) demonstrated its effectiveness in game engines and map navigation using fine-tuned heuristics [6], [9]. Sidhu

(2020) and Mathew et al. (2021) confirmed that while DFS and BFS have limited application in real-time, weighted environments, heuristic-based methods like A* consistently outperform them in execution time and efficiency [7], [20].

Swarm-based methods offer an adaptive, decentralized alternative. Moran (2017) and Pan & Pun-Cheng (2020) examined such algorithms in the context of dynamic systems and multi-agent environments [8], [13].

Lim et al. (2015), Yap (2002), and Lee & Lawrence (2013) contributed to the exploration of uninformed and grid- based approaches, proposing optimizations for performance in large, sparse grids [10], [11], [12].

Tabhane et al. (2021) validated the growing academic interest in visual learning tools for pathfinding, mirroring Pathfinder Pro's approach [14].

Online platforms such as Red Blob Games [15], GeeksforGeeks [16], [17], [18], and Clement Mihailescu's Pathfinding Visualizer [19] have made algorithmic learning widely accessible, influencing design and implementation strategies in modern visualization tools.

## III. METHODOLOGY AND SYSTEM DESIGN

The Pathfinder Pro project is designed to visualize and analyze the performance of various pathfinding algorithms within a grid-based environment. The platform is implemented using a modern web development stack and follows a modular design approach to ensure extensibility, responsiveness, and clarity.

### A. System Architecture:

The system is composed of the following main components:

1. **User Interface (UI):** Built using React.js, it allows users to interact with the grid, select algorithms, place walls, and set start and end nodes. It also handles user-triggered events like resetting the grid, pausing visualization, and switching themes.
2. **Algorithm Engine:** Implemented in JavaScript, this module contains independent implementations of the supported pathfinding algorithms. Each algorithm follows a common interface, making it easy to switch or compare between them.
3. **Visualization Layer:** This component uses the HTML5 Canvas API to animate algorithm execution in real time. Each visited node and the final path are rendered with color-coded feedback to enhance user understanding.
4. **Performance Tracker:** A background module records metrics such as path cost, nodes visited, and computation time. These values are displayed to the user upon completion of each run and are used to evaluate algorithm efficiency.

### B. Algorithms Implemented:

#### I. Weighted Algorithms:

These algorithms take weights (or costs) into account when finding the path between nodes.

1. **Dijkstra's Algorithm:** Explores all paths to guarantee the shortest one is found. It is reliable and optimal for weighted grids but may be slower due to its exhaustive nature.
2. **A* Search Algorithm:** Uses a combination of path cost and heuristic estimates to prioritize nodes. It guarantees the shortest path and is significantly faster than Dijkstra's in weighted grids due to its focused search strategy.
3. **Greedy Best-First Search:** Prioritizes nodes based solely on heuristic distance to the goal. It is faster and efficient in many cases but does not guarantee the shortest path in weighted grids.
4. **Swarm Algorithm:** Mimics collective search behavior by blending Dijkstra's coverage with heuristic guidance like A*. It explores outward from the start while targeting the goal but may not always find the shortest path in weighted grids.
5. **Convergent Swarm Algorithm:** An optimized variation of the Swarm Algorithm that emphasizes heuristic influence for faster convergence. It improves speed but sacrifices the guarantee of an optimal path in weighted environments.
6. **Bidirectional Swarm Algorithm:** Initiates simultaneous exploration from both the start and target nodes using swarm logic. This approach accelerates pathfinding in weighted grids but does not ensure the shortest path.

#### II. Unweighted Algorithms:

7. **Breadth-First Search:** Explores layer by layer and guarantees the shortest path in unweighted grids. It is simple, reliable, and systematic for uniform-cost environments.
8. **Depth-First Search:** Follows one path deeply before backtracking, making it suitable for full traversal but inefficient for finding the shortest path in unweighted scenarios.

### C. Data Representation:

The environment is modeled as a 2D grid where each cell represents a node. Each node has properties such as isWall, isVisited, distanceFromStart, and estimatedDistanceToEnd. The grid is updated dynamically based on user input and algorithm activity.

### D. User Interaction Flow:

1. User initializes the grid and places walls.
2. Selects a pathfinding algorithm from a dropdown menu.
3. Clicks “Visualize” to start execution.
4. The visualization engine shows the algorithm in action.
5. Performance metrics are displayed at the end.

This design promotes hands-on experimentation and intuitive comparison of multiple algorithms, fulfilling both educational and practical goals.

## IV. EXPERIMENTAL SETUP

The experimental setup for Pathfinder Pro was designed to assess the performance and behavior of various pathfinding algorithms in a customizable and interactive environment. The system is built using React.js for the user interface and JavaScript for algorithm implementation. The HTML5 Canvas API was used to render real-time animations of each algorithm as it traverses a 2D grid.

The grid size is user-configurable, typically ranging from 20×20 to 50×50 nodes. Users can manually place obstacles, define start and end nodes, and select one of the supported algorithms: Dijkstra’s, A*, Greedy Best-First Search, Depth-First Search (DFS), and Swarm. Each algorithm operates independently on the same grid to ensure consistent comparisons.

Performance metrics collected during each run include: Path cost (number of nodes in the final path)

Total nodes visited

Execution time (in milliseconds)

Testing was conducted on a machine with an Intel Core i5 processor and 8 GB RAM, using Google Chrome for consistent rendering. Each algorithm was tested under varying grid configurations and obstacle densities. A reset feature allowed repeated testing under identical conditions to support comparative analysis. The setup successfully demonstrated real-time performance variations and helped visualize trade-offs between speed, accuracy, and coverage.

## V. RESULT AND DISCUSSION

The Pathfinder Pro application was tested across multiple grid sizes and obstacle configurations using five distinct algorithms: Dijkstra’s, A*, Greedy Best-First Search, Depth-First Search (DFS), and Swarm. Each algorithm was evaluated based on three key metrics—total nodes visited, path cost, and execution time.

The results revealed that A* consistently offered the best trade-off between speed and path accuracy. It completed searches faster than Dijkstra’s while still ensuring optimal paths due to its heuristic guidance.Dijkstra’s algorithm, although accurate, was comparatively slower due to its exhaustive search pattern. Greedy Best-First Search was the fastest in most tests but often failed to find the shortest path, especially in dense obstacle scenarios.

DFS, while simple, proved inefficient and unreliable for shortest-path determination, frequently returning suboptimal or incomplete paths.

Swarm algorithm results were mixed: it performed moderately well in dynamic and clustered grids but lagged behind A* in most structured environments.

Visualizations made it easy to observe how each algorithm explored the grid, and the collected metrics supported both technical comparison and user-friendly understanding. The consistent execution environment ensured reliable data for side-by-side analysis.

*Figure 5.1 Pathfinder Pro – Welcome Tutorial Interface*

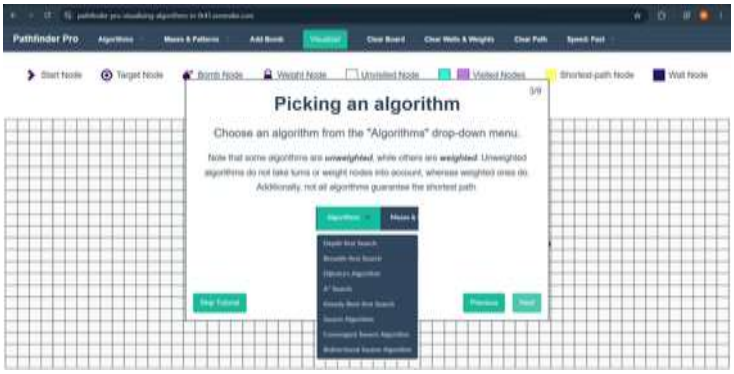


*Figure 5.2 Tutorial Screen: Algorithm Selection Menu*

*Figure 5.4 Breadth-First Search Algorithm using Simple Stair Pattern*

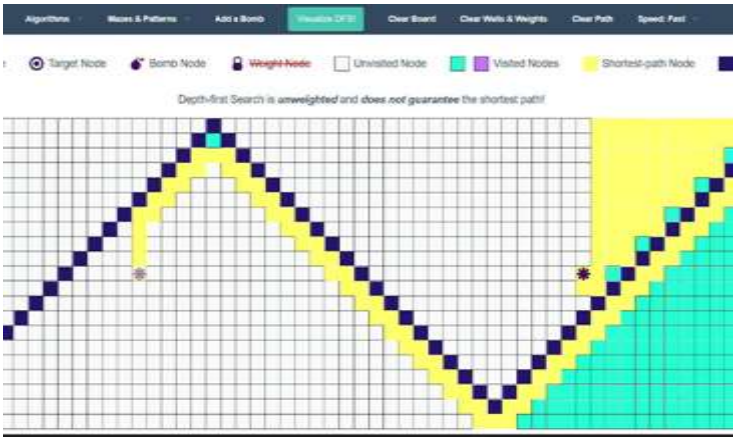


*Figure 5.3 Depth-First Search Algorithm using Simple Stair Pattern*

*Figure 5.5 Dijkstra's Algorithm using Basic Random Maze*

*Figure 5.6 A* Algorithm using Simple Stair Pattern*

*Figure 5.7 Greedy Algorithm using Simple Stair Pattern*

*Figure 5.8 Swarm Algorithm using Recursive Diversion*

*Figure 5.9 Convergent Swarm Algorithm using Recursive Diversion*


*Figure 5.10 Bidirectional Swarm Algorithm using Weight Maze*

## VI. CONCLUSION

Pathfinder Pro successfully demonstrates and compares the practical applications of several key pathfinding algorithms through an interactive and educational platform. By integrating real-time visual feedback, customizable test cases, and detailed performance metrics, the system allows users to explore how algorithms like Dijkstra's, A*, Greedy Best-First Search, DFS, and Swarm perform under different conditions. The results clearly highlight that A* is the most efficient and balanced algorithm for most use cases, combining speed with path accuracy. Dijkstra's, while dependable, is more resource- intensive. Greedy BFS is fastest but not always reliable, and DFS is generally unsuitable for optimal pathfinding. Swarm algorithms show promise in adaptive environments.

This project bridges the gap between theory and practice, providing students, educators, and developers a hands-on tool to better understand pathfinding strategies. It also sets the foundation for future enhancements such as 3D visualization, dynamic obstacle handling, and AI-based adaptive pathfinding. The platform's modular architecture supports easy integration of additional algorithms or functionalities. Its user-friendly interface encourages experimentation and deep learning through interactive exploration. Pathfinder Pro can also serve as a prototype for real-world systems such as delivery route optimizers, robot navigation planners, and game AI path modules.

## VII. FUTURE SCOPE

While Pathfinder Pro effectively demonstrates and compares core pathfinding algorithms in a 2D environment, there is considerable potential for expansion and enhancement.

Future versions of the platform could include support for 3D pathfinding environments, allowing simulations in multi-level structures such as buildings or terrains. Integration with real-world mapping APIs (like Google Maps or OpenStreetMap) could enable real-time navigation simulations and urban planning applications.

Additionally, implementing more advanced and domain- specific algorithms such as Jump Point Search (JPS), Theta*, and Ant Colony Optimization could provide broader comparison options. Dynamic pathfinding—where obstacles appear, disappear, or move during execution—is another area worth exploring for robotics and AI simulations.

Enhancing the visualization engine with more intuitive UI/UX design, sound feedback, and mobile responsiveness could further improve accessibility and learning experience. Incorporating machine learning models to suggest the best algorithm for a given scenario based on environment patterns is also a promising area for research.

Overall, Pathfinder Pro lays the groundwork for an extensible pathfinding research and learning tool that can grow with both educational and real-world applications.

### *VIII. ACKNOWLEDGEMENT*

The authors sincerely thank the mentors and academic guides for their invaluable guidance, encouragement, and constructive feedback throughout the development of this project. Their insights greatly contributed to the successful implementation of Pathfinder Pro.

We also extend our gratitude to the faculty and staff of the Department of Computer Science & Engineering, Shri Shankaracharya Technical Campus, Bhilai, for their continuous support and provision of necessary resources.

Special thanks to our peers and colleagues who offered valuable suggestions during the testing and review phases. Lastly, we express heartfelt appreciation to our families for their unwavering motivation and moral support throughout the course of this work.

## IX. REFERENCES


1. S. Sharma, R. Singh, and R. Kumar, "Pathfinding Visualizer," ResearchGate, Oct. 2023. Retrieved from: https://www.researchgate.net/publication/374848532
2. Singh, B. Singh, G. Singh, H. Kaur, and K. Singh, "Node Path Visualizer Using Shortest Path Algorithms," International Research Journal of Engineering and Technology (IRJET), vol. 9, no. 6, pp. 2021–2024, Jun. 2022.
3. C. Mandloi, A. Thomas, A. S. Yadav, and A. Soni, "Path Finding Algorithm Visualization," Research Proposal, Acropolis Institute of Technology and Research, Oct. 2022. DOI: 10.13140/RG.2.2.20685.92644
4. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," Numerische Mathematik, vol. 1, pp. 269–271, 1959.
5. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993
6. M. Mapaila, "Efficient Path Finding for Tile-Based 2D Games," M.Sc. thesis, Dept. Comp. Sci., University of Cape Town, 2012.
7. K. Sidhu, "Performance Evaluation of Pathfinding Algorithms," Electronic Theses and Dissertations, no. 8178, University of Windsor, 2020.
8. M. Moran, "On Comparative Algorithmic Pathfinding in Complex Networks for Resource-Constrained Software Agents," Ph.D. dissertation, Walden University, 2017.
9. Y. Halldorsson, "Improved Heuristics for Optimal Pathfinding on Game Maps," in Proc. AIIDE, 2006, pp. 9–14.
10. K. L. Lim, K. P. Seng, and L. S. Yeong, "Uninformed Pathfinding: A New Approach," Expert Systems with Applications, vol. 42, no. 6, pp. 2722–2730, 2015.
11. P. Yap, "Grid-Based Path-Finding," in Advances in Artificial Intelligence, Canadian AI 2002, Lecture Notes in Computer Science, vol. 2338.
12. W. Lee and R. Lawrence, "Fast Grid-Based Path Finding for Video Games," in Canadian AI 2013, Lecture Notes in Computer Science, vol. 7884.
13. T. Pan and S. C. Pun-Cheng, "A Discussion on the Evolution of the Pathfinding Algorithms," Preprints, 2020, 2020080627.
14. A. N. Tabhane, N. Likhar, K. Mohod, and M. Kadbe, "Path Finding Visualizer," International Journal of Advance Research and Innovative Ideas in Education (IJARIIE), vol. 7, no. 3, 2021.
15. "Pathfinding Grids – Red Blob Games," Retrieved from: https://www.redblobgames.com/pathfinding/grids/
16. "Graph Data Structure and Algorithms – GeeksforGeeks," Retrieved from: https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/
17. "A* Search Algorithm – GeeksforGeeks," Retrieved from: https://www.geeksforgeeks.org/a-search-algorithm/
18. "Dijkstra's Shortest Path Algorithm – GeeksforGeeks," Retrieved from: https://www.geeksforgeeks.org/dijkstras- shortest-path-algorithm-greedy-algo-7/
19. "Pathfinding Visualizer – Clement Mihailescu," Retrieved from: https://clementmihailescu.github.io/Pathfinding-Visualizer/
20. A. Mathew et al., "Performance Evaluation of Shortest Path Algorithms," Journal of Physics: Conference Series, vol. 1831, 012008, 2021.