Evaluating Pathfinding Algorithms: A Comprehensive Analysis in Square Grid Environments

Kesley Raimundo
Londrina, Brazil
kesleyraimundo@gmail.com
0000-0001-7376-750X

***Abstract*—This research focuses on evaluating various pathfinding algorithms within the context of a square grid environment, a fundamental structure common to many computational problems. Our objective is to critically assess the performance of a broad spectrum of algorithms, ranging from traditional, uninformed methods like Breadth-First Search (BFS) and Depth-First Search (DFS) to more advanced, heuristic-driven techniques such as Dijkstra's algorithm and A*. The evaluation criteria focus on operational efficiency, adaptability to different scenarios within the grid, and the overall path optimality produced by each algorithm. Through a structured analysis, utilizing visual simulations developed with Python's Pygame library, this study aims to establish benchmarks for algorithm performance that can inform future research directions.**

***Index Terms*—Pathfinding algorithms, square grid, Breadth-First Search, Depth-First Search, Dijkstra's algorithm, A* algorithm, Jump Point Search, algorithm evaluation, computational navigation.**

PREFACE

This document marks the initial phase of an extensive investigation into pathfinding algorithms. It's important to understand that this segment intentionally omits final conclusions and specific findings. Our aim here is to establish a solid foundation, offering an introduction to the subject, delineating the problem space, and reviewing existing literature. The forthcoming section will unveil the entirety of this research, including detailed experimental findings, discussions, and the conclusions derived from the study. Given the nature of this work, it should be noted that not every algorithm mentioned may be thoroughly examined, and there is a possibility that additional algorithms not currently identified could be considered. Similarly, the reference list may undergo significant changes as the project progresses and as experimental outcomes are integrated.

I. INTRODUCTION

Pathfinding is a fundamental computational problem with wide-ranging applications in fields such as robotics, game development, logistics, and navigation systems [1]–[3]. It involves determining an efficient route between two points within a specified environment. Typically, this problem is represented through a graph, where an algorithm is tasked to find most optimal path from a starting node to a destination node, while minimizing predefined cost functions like distance, time, or resources [4], [5].

Despite its straightforward premise, the complexity of pathfinding significantly increases with the expansion of the environment. For instance, in a $n \times n$ grid employing Manhattan distance, and without obstacles, the total number of optimal paths N , those with length $2n$, can be expressed as:

$$N = \binom{2n}{n} = \frac{(2n)!}{n! \cdot n!}$$

For large values of n, this equation asymptotically approaches an exponential behavior, indicating that $N \sim 4^n$ [6], [7]. This becomes particularly significant in real-world scenarios, which commonly involve large n values. Therefore, if we aim to solve real-world pathfinding problems, it is crucial that we have efficient algorithms that can adeptly navigate this vast search space.

Pathfinding algorithms can broadly be categorized into two classes: deterministic and heuristic. Deterministic algorithms follow a fixed rule or set of instructions to determine the path from the start node to the destination, ensuring predictable behavior across multiple runs. Examples include Depth-First Search (DFS) and Breadth-First Search (BFS), which systematically explore all possible paths [4], [7]. On the other hand, heuristic algorithms, such as A* and Greedy Best-First Search, employ informed guesses to prioritize paths that appear to lead more directly towards the goal [5], [8].

While the effectiveness of these algorithms is deeply influenced by the intricacies of specific problems, employing a grid as a simplified model offers a foundational framework for systematic analysis. This approach not only facilitates the comparative evaluation of different algorithms but also highlights their adaptability and performance when faced with variations in the problem's parameters. The grid, therefore, stands as a pragmatic simplification for initial studies, providing a clear and manageable platform from which to launch a comprehensive investigation into pathfinding strategies. It acts as a conceptual bridge, connecting the theoretical foundations of algorithm design with their practical deployment in real-world scenarios [9], [10]. This dual role underscores the grid's significance not just as an academic tool, but also as a stepping

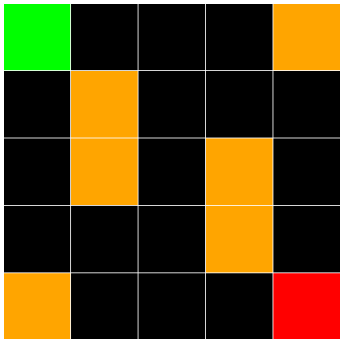

Fig. 1. An example environment setup for $n = 5$, with $k = 6$ obstacles, and start (x_i) and end (x_f) points located at $(0, 0)$ and $(4, 4)$ respectively.

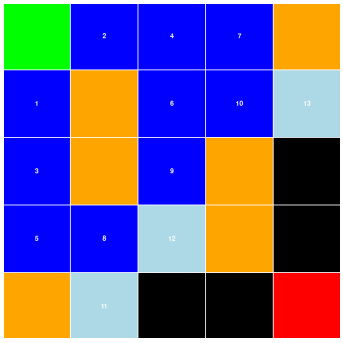

Fig. 2. Demonstration of BFS exploration after 10 steps, showcasing explored and frontier nodes.

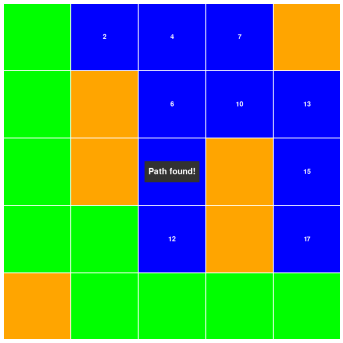

Fig. 3. Visual representation of a path found by BFS, with the path marked in green.

stone towards developing robust solutions for complex, real-life navigation problems.

Informed by these principles, this study sets out to evaluate a range of pathfinding algorithms within a square grid environment. Our examination encompasses deterministic strategies to heuristic-based techniques. The goal is to dissect the comparative effectiveness, adaptability, and performance subtleties of these algorithms as they navigate the structured yet dynamic challenges presented by the grid environment [11], [12].

II. Problem Setting

The chosen environment for our study is a square grid of size $n \times n$ with a Manhattan distance, where each cell is categorized as either empty, an obstacle, a start point, or a goal point. These cells are visually distinguished by color in the implementation: empty cells are black, obstacles are orange, the start point is green, and the goal is red, as illustrated in Figure 1.

Cells within the grid are identified by coordinates (x, y), where y begins from 0 at the top and increases downwards, and x starts from 0 on the left, increasing to the right. The primary objective for the algorithms is to discover a feasible path from the start point to the goal, navigating through a grid populated with k obstacles. Figure 2 demonstrates the exploration process of the Breadth-First Search (BFS) algorithm at its 10th step, marking already explored nodes in blue and the frontier nodes in pale blue, with their exploration order denoted by numbers within each cell.

Upon successfully identifying a path, the solution is visually indicated by coloring all cells along the path green, as depicted in Figure 3.

For the development of visual representations, we selected Python's Pygame library. This choice enables the grid setup to accommodate various scenarios, including randomly placed start and end points and a variable number of obstacles, thereby facilitating a thorough evaluation of algorithm performance under diverse conditions.

The code architecture is primarily composed of a board class, which encapsulates all attributes and methods related to the environment setup, and a main file that abstractly implements the algorithm. This structure allows the algorithm to be executed either step-by-step for visualization purposes or in its entirety to find a solution. For analytical purposes, we will execute the latter method multiple times across different scenarios: initially on a fixed $n = 10$ grid with predetermined obstacles (utilizing a constant seed for all algorithms), followed by tests on an $n = 100$ grid with randomly initialized conditions for each run.

Results will be summarized in tables and charts to visually compare the performance of each algorithm. A discussion and analysis of these results will conclude the section, highlighting key insights and observations.

III. Algorithms

This section presents the pathfinding algorithms investigated in this study, organized into two main groups: deterministic and heuristic. Reflecting on the preface, given time and length constraints, we initially concentrate on implementing and analyzing a specific set of algorithms. Yet, as the research progresses, we plan to adapt our focus, potentially incorporating additional algorithms to enrich our analysis.

A. Deterministic Algorithms

The algorithms selected under this category are celebrated for their proven efficacy across a spectrum of pathfinding scenarios. They will provide the benchmark metrics against which the performance of more advanced techniques can be measured.

1) Breadth-First Search (BFS): BFS is a straightforward graph traversal technique that systematically explores the grid from the starting node, layer by layer. It employs a queue mechanism to manage the nodes, ensuring the achievement of the goal in the fewest possible moves within an unweighted environment.

2) Depth-First Search (DFS): DFS delves deep into the grid, venturing as far as possible down each branch before retracting. Utilizing a stack for its operations, DFS is noted for its memory efficiency, albeit without assurance of the shortest path discovery.

3) Dijkstra's Algorithm: An evolution of BFS for weighted graphs, Dijkstra's algorithm leverages a priority queue to ascertain the shortest paths from the source to all other nodes, ensuring path optimality in environments with varied traversal costs.

B. Heuristic Algorithms

1) Greedy Best-First Search: This algorithm employs a heuristic to steer its exploration towards the goal, optimizing for speed rather than the assuredness of finding the shortest path. It shines in relatively straightforward pathfinding challenges.

2) A (A-Star) Algorithm:* A* represents an informed search strategy that integrates a heuristic cost function to pinpoint the shortest path efficiently. It adeptly balances the incurred cost from the start node with the estimated cost to reach the goal.

3) Jump Point Search (JPS): JPS specializes in grid-based pathfinding by identifying and utilizing jump points to contract the search area, significantly boosting A*'s efficiency without the need for additional memory resources.

REFERENCES

[1] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] I. MILLINGTON and J. FUNGE, "Artificial intelligence for games. 2ª edição," *Burlington (EUA): Taylor & Francis Inc*, 2009.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[5] 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.

[6] D. E. Knuth, *The art of computer programming*. Pearson Education, 2005.

[7] R. Sedgewick and K. Wayne, *Algorithms*. Addison-wesley professional, 2011.

[8] J. Pearl, *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[9] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 144–148, 2012.

[10] D. Harabor and A. Grastien, "Online graph pruning for pathfinding on grid maps," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 25, no. 1, 2011, pp. 1114–1119.

[11] S. Rabin and N. R. Sturtevant, "Pathfinding architecture optimizations," in *Game AI Pro 360: Guide to Movement and Pathfinding*. CRC Press, 2019, pp. 1–12.

[12] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, vol. 1, no. 1, 2005, pp. 117–122.

[13] Z. B. Zabinsky *et al.*, "Random search algorithms," *Department of Industrial and Systems Engineering, University of Washington, USA*, 2009.

[14] K. L. Lim, K. P. Seng, L. S. Yeong, L.-M. Ang, and S. I. Ch'ng, "Uninformed pathfinding: A new approach," *Expert systems with applications*, vol. 42, no. 5, pp. 2722–2730, 2015.

[15] R. Dube, A. Joshi, S. Bhagwat, P. N. Mahalle, and J. Barot, "Pathfinding visualizer: A survey of the state-of-art," in *International Conference on Information and Communication Technology for Intelligent Systems*. Springer, 2023, pp. 139–149.

[16] N. Rivera, C. Hernández, and J. Baier, "Grid pathfinding on the 2k neighborhoods," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[17] D. Harabor and A. Grastien, "An optimal any-angle pathfinding algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 23, 2013, pp. 308–311.

[18] A. Botea, "Ultra-fast optimal pathfinding without runtime search," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 7, no. 1, 2011, pp. 122–127.

[19] A. Botea, M. Müller, and J. Schaeffer, "Near optimal hierarchical pathfinding." *J. Game Dev.*, vol. 1, no. 1, pp. 1–30, 2004.

[20] D. S. Ashish, S. Munjal, M. Mani, and S. Srivastava, "Path finding algorithms," in *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2020, Volume 1*. Springer, 2021, pp. 331–338.

[21] A. Chen and Z. Ji, "Path finding under uncertainty," *Journal of advanced transportation*, vol. 39, no. 1, pp. 19–37, 2005.

[22] D. Demyen and M. Buro, "Efficient triangulation-based pathfinding," in *Aaai*, vol. 6, 2006, pp. 942–947.

[23] A. Kherrour, M. Robol, M. Roveri, and P. Giorgini, "Evaluating heuristic search algorithms in pathfinding: A comprehensive study on performance metrics and domain parameters," *arXiv preprint arXiv:2310.02346*, 2023.