Research on Path Planning Method based on Graph Search Algorithm

Langyu Shi

Guangdong Country Garden School, Foshan, China

100473@yzpc.edu.cn

Abstract. Path planning, crucial in robotics and autonomous vehicles, involves finding efficient routes from start to finish while avoiding obstacles. Graph search algorithms like BFS, DFS, and A* are commonly used for this purpose. This research focuses on analyzing these algorithms' performance and applicability in diverse scenarios. The study provides a comprehensive overview of path planning's definition, application domains, and fundamental principles. It then delves into the core graph search algorithms: BFS explores nodes layer by layer, ensuring the shortest path is found first but suffering from high time complexity. DFS explores nodes in a depth-first manner, potentially leading to shorter paths but risking getting trapped in local optima. In contrast, A* combines the strengths of BFS and DFS by using heuristic functions to guide the search for the cheapest path to the goal. Despite their utility, graph search algorithms have limitations. The high time complexity of A* in large-scale environments and the potential for local optima are key challenges. To address these, research explores improved methods like GBFS and bidirectional A* search. The future direction of research includes developing more efficient heuristic functions, exploring hybrid algorithms, parallelizing path planning, and applying these methods to new domains like autonomous drones and self-driving cars. This research paves the way for more robust and efficient path planning algorithms in various applications.

Keywords: Path planning; A* algorithm; DFS; BFS.

1. Introduction

Path planning is a fundamental problem in various domains, from robotics and autonomous vehicles to aerial drones and complex urban road networks. It involves determining a sequence of actions or a trajectory that allows an agent to move from a starting point to a goal while avoiding obstacles and adhering to certain constraints. The efficiency and accuracy of path planning algorithms are critical for the successful operation of these systems.

Graph-based algorithms have emerged as a powerful tool for addressing path planning challenges. By representing the environment as a graph, where nodes correspond to points of interest and edges represent connections or transitions between these points, these algorithms can efficiently explore and search the state space to find optimal or near-optimal solutions. Common graph-based algorithms include Breadth-First Search (BFS), Depth-First Search (DFS), and the A* algorithm, each with its unique strengths and applications [1].

BFS and DFS are basic search algorithms that provide a foundation for more advanced path planning techniques. BFS systematically explores the graph by expanding all neighboring nodes at each level before moving to the next level, ensuring the shortest path is found. In contrast, DFS explores a path as deeply as possible, backing up only when necessary, which can be useful in certain scenarios where the goal lies in a deeper level of the graph.

The A* algorithm combines the advantages of BFS and DFS by incorporating a heuristic function to guide the search towards the goal. This informed search strategy enables A* to efficiently find optimal paths in complex environments, making it a preferred choice for many path planning applications.

Research in path planning methods based on graph search algorithms continues to advance, with a focus on improving efficiency, handling large-scale environments, and addressing the risk of getting trapped in local optima. Future directions include developing more sophisticated heuristics, exploring

hybrid algorithms that combine different approaches, and leveraging parallel computing capabilities [2].

In summary, path planning methods based on graph search algorithms play a crucial role in enabling intelligent and autonomous systems to navigate complex environments effectively. Understanding and mastering these algorithms is essential for addressing the challenges and meeting the demands of modern applications in robotics, transportation, and beyond.

2. Overview of Path Planning

Path planning is the process of determining a sequence of actions or waypoints that allows an agent to travel from an initial state to a goal state while avoiding obstacles and satisfying any additional constraints. It is a critical component in various fields, such as robotics, autonomous vehicles, and computer graphics.

2.1. Definition and Application Domains of Path Planning

Path planning can be defined as the problem of finding a path from a given initial configuration to a desired goal configuration in a given environment. The path must satisfy certain constraints, such as avoiding obstacles, minimizing the distance or time required to travel, and adhering to any other specific requirements of the application domain.

Different application domains have different requirements and constraints when it comes to path planning. For example, in robotics, path planning is essential for navigating a robot through a cluttered environment, avoiding collisions, and reaching a target destination. In autonomous vehicles, path planning is crucial for safe and efficient navigation on roads, highways, and in traffic-congested scenarios. In computer graphics, path planning can be used to generate realistic animations of characters moving through virtual environments.

2.2. Fundamental Principles and Steps of Path Planning

The fundamental principles of path planning involve several key steps, which are typically followed in a systematic manner. These steps include:

Environment representation: The first step in path planning is to represent the environment in a way that allows for efficient computation and analysis. This often involves creating a map or model of the environment, which includes information about the location and shape of obstacles, the terrain, and any other relevant features.

Initial and goal state definition: The next step is to define the initial state, which represents the starting point of the path, and the goal state, which represents the desired endpoint of the path. The initial and goal states should be chosen based on the specific requirements of the application domain and the desired outcome of the path planning process.

Path planning algorithm selection: Once the environment, initial, and goal states have been defined, the next step is to select an appropriate path planning algorithm. The choice of algorithm will depend on various factors, such as the complexity of the environment, the desired level of accuracy, and the computational resources available.

4) Path computation: With the environment, initial, and goal states, and algorithm in place, the next step is to compute the path. This involves applying the chosen algorithm to the defined problem, which may involve searching through a large number of possible paths or configurations to find the optimal or near-optimal solution.

5) Path execution: Once the path has been computed, the final step is to execute it. This may involve controlling a robot or vehicle to follow the path or simulating the path in a computer-generated environment. The execution of the path should be carried out in a way that ensures safety, efficiency, and adherence to any additional constraints specified in the problem.

2.3. Applications of Path Planning

2.3.1 Robotics

In robotics, path planning is crucial for enabling autonomous movement without collisions. By abstractly modeling and visually reproducing the surrounding environment, path planning assists robots in determining the optimal path of action, thus enhancing their efficiency and accuracy [2]. Additionally, in the context of robotic arm movements, path planning is also essential for constructing precise trajectories.

2.3.2 Unmanned Aerial Vehicles (UAVs)

For drones flying in the air, path planning is used to efficiently adapt to unfamiliar environments, avoid obstacles, and achieve evasion flights [3]. Moreover, path planning is also utilized in scenarios such as aerial photography, search and rescue, and environmental monitoring to plan the most optimal flight paths for drones.

2.3.3 Urban Road Networks

In the context of urban road networks, path planning technology is widely used in navigation and intelligent transportation systems. By leveraging intelligent devices like electronic real-time display screens, path planning helps drivers quickly find the best routes, reducing traffic congestion and travel time.

2.4. Graph-based algorithms

Graph-based algorithms represent the environment as a graph, with nodes representing points of interest or key locations in the environment, and edges representing connections or paths between these points. This representation allows for efficient processing of large amounts of data and flexible adaptation to dynamic changes in the environment.

The key advantage of graph-based algorithms is their ability to handle complexity and scale well. They are suitable for a wide range of applications, including robot navigation, social network analysis, logistics optimization, and more.

Some common graph-based algorithms include:

1) Breadth-First Search (BFS): This algorithm traverses or searches a graph by starting from a root (or any arbitrary node) and exploring all adjacent nodes first, before moving on to the next level of nodes. BFS is often used to find the shortest path between two nodes in a graph [4].

2) Depth-First Search (DFS): Unlike BFS, DFS explores as deeply as possible along each branch of the graph. It follows a path until it reaches a dead end, then backtracks to the previous node and continues exploring other paths. DFS is effective for tasks such as graph traversal, cycle detection, and more [5].

3) A* Algorithm: This heuristic search algorithm combines the characteristics of BFS and DFS by assigning a "cost" value to each node to guide the search process. The cost value is typically based on the actual distance from the start node to the current node and an estimated distance from the current node to the goal. The A* algorithm is widely used in path planning, game AI, and other domains where finding an optimal or near-optimal path is crucial [6].

3. Graph Search Algorithms and Their Basic Principles

3.1. Introduction to graph search algorithms

Graph search algorithms are a class of algorithms for finding paths, cycles, and minimum spanning trees in graphs. These algorithms are widely used in various fields, such as artificial intelligence, operations research, and computer science. In path planning, graph search algorithms represent the environment as a graph, where nodes represent points of interest in the environment and edges represent connections between these points. The main objective of graph search algorithms is to find a path from an initial node to a goal node while satisfying certain constraints, such as obstacle avoidance and path optimality.

Graph search algorithms can be classified into two main types: depth-first search (DFS) and breadth-first search (BFS). DFS explores as far as possible along each branch before backtracking, while BFS explores the neighbors of the current node before moving on to the next level of nodes. In addition, the A* algorithm, which is an informed search algorithm, can also be used for graph search in path planning

3.2. Breadth-First Search (BFS) and its applications

Breadth-first search (BFS) is a graph search algorithm that explores all the vertices of a graph in breadth-first order, that is, it visits all the vertices at the same level before moving on to the next level. BFS is guaranteed to find the shortest path from the initial node to the goal node in an unweighted graph.

The flowchart below is BFS algorithm, and was shown in the Figure 1.

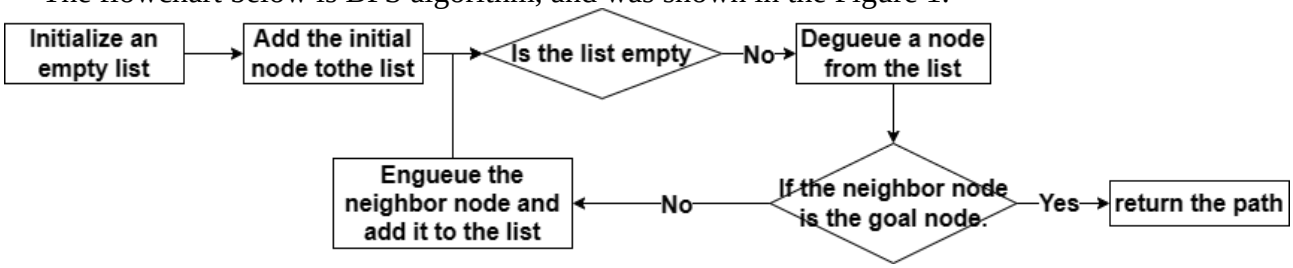

Fig. 1 Flowchart of BFS algorithm (Photo credited: Original)

The main steps of BFS include initializing an empty list and adding the initial node to the queue. Then, while the list is not empty, a node is dequeued. For each neighbor of the current node, the neighbor is enqueued, and if it happens to be the goal node, the search is stopped and the path is returned. If the goal node is not found after visiting all reachable nodes, the search ends without finding a path. BFS guarantees that the shortest path from the initial node to the goal node will be found if one exists.

BFS has various applications in path planning, such as:

1) Finding the shortest path in a map from a starting point to a destination.
2) Solving puzzles, such as mazes and crossword puzzles.
3) Traversing directed acyclic graphs (DAGs) to find the topological sorting of tasks.
4) Detecting cycles in a graph.

3.3. Depth-First Search (DFS) and its applications

Depth-first search (DFS) is a graph search algorithm that explores as far as possible along each branch before backtracking. DFS does not guarantee the shortest path to the goal node, but it can be more efficient than BFS in certain cases, especially when the goal node is close to the initial node.

The flowchart below is DFS algorithm, and was shown in the Figure 2.

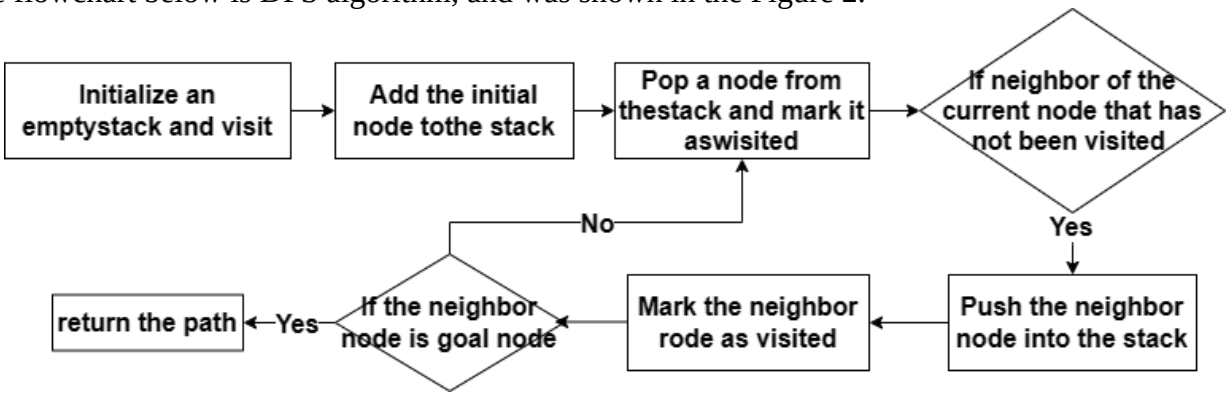

Fig. 2 Flowchart of DFS algorithm (Photo credited: Original)

DFS has various applications in path planning, such as:

1) Exploring large graphs with many branches.

2) Solving problems that can be divided into smaller subproblems, such as divide and conquer algorithms.

3) Generating spanning trees of a graph.

4) Finding connected components in a graph.

3.4. A* Algorithm and its applications

The A* (A-star) algorithm is an informed search algorithm that combines the best features of both DFS and BFS. It uses heuristic functions to estimate the cost of the cheapest path from the current node to the goal node.

The flowchart below is A* algorithm [7].

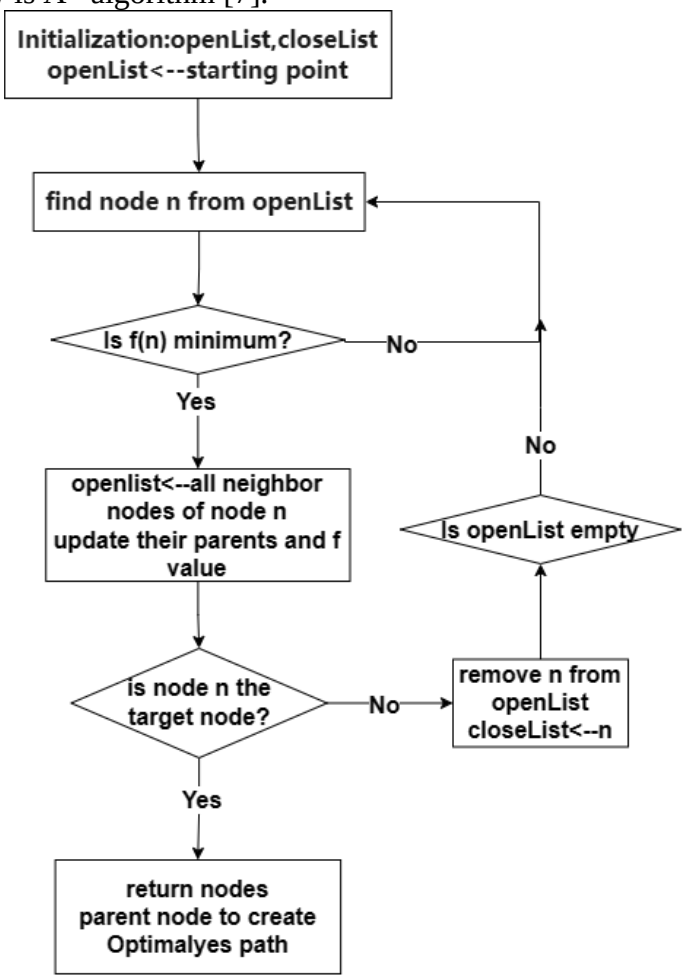

Fig. 3 Flowchart of A* algorithm [7]

The A* algorithm is a heuristic search algorithm that combines the advantages of best-first search and breadth-first search. In the A* algorithm, each node has an F-value consisting of two parts: the actual cost g from the starting point to the current node and the estimated cost h from the current node to the target.The A* algorithm will prefer to select the node with the smallest f value for expansion until the target node is found [7].

The A* algorithm is widely used in path planning applications, such as:

1) Finding the shortest path in maps and graphs with weighted edges.

2) Navigating autonomous vehicles and robots in unknown environments.

3) Solving pathfinding problems in computer games and simulations.

3.5. Problem Formulation and Analysis

Path planning is the process of determining a path from a starting point to a goal point in a given environment, while avoiding obstacles and satisfying certain constraints. It is a common problem in various fields, such as robotics, autonomous vehicles, computer games, and simulations. The main challenges in path planning include efficiency, accuracy, and robustness.

Graph search algorithms, such as BFS, DFS, and A*, are widely used in path planning due to their ability to explore the state space efficiently and find optimal or near-optimal solutions. In this section, we will focus on the research on path planning methods based on these graph search algorithms.

3.6. Research on BFS-based Path Planning Methods

BFS-based path planning methods use the Breadth-First Search algorithm to explore the state space in a layer-by-layer manner, ensuring that the shortest path is found first. However, BFS has a high time complexity, which limits its application in large-scale environments.

To overcome this limitation, various improved BFS-based path planning methods have been proposed. For example, the Bi-directional BFS (Bidirectional BFS) algorithm starts searching from both the starting point and the goal point, and combines the results when a path is found. This can significantly reduce the search space and improve the efficiency [8].

Another example is the GBFS (Generalized BFS) algorithm, which uses a heuristic function to estimate the cost of the cheapest path from the current node to the goal node [9]. By prioritizing nodes with lower heuristic costs, GBFS can find optimal or near-optimal solutions while maintaining a lower time complexity than A*.

3.7. Research on DFS-based Path Planning Methods

DFS-based path planning methods use the Depth-First Search algorithm to explore the state space in a depth-first manner, which can lead to shorter paths but may get trapped in local optima. To address this issue, various improved DFS-based path planning methods have been proposed.

For example, the UCT (Upper Confidence Bound applied to Trees) algorithm uses a heuristic function to evaluate the quality of paths, and selects the path with the highest upper confidence bound as the best path [10]. This can help avoid getting trapped in local optima and find better solutions.

3.8. Research on A*-based Path Planning Methods

A*-based path planning methods use the A* algorithm, which is an informed search algorithm that combines the best features of both DFS and BFS. It uses heuristic functions to estimate the cost of the cheapest path from the current node to the goal node, and prioritize nodes with lower total costs [6].

The A* algorithm is widely used in path planning applications, such as finding the shortest path in maps and graphs with weighted edges, navigating autonomous vehicles and robots in unknown environments, and solving path finding problems in computer games and simulations.

However, the A* algorithm has a high time complexity when dealing with large-scale environments. To address this issue, various improved A*-based path planning methods have been proposed. For example, the GBFS algorithm mentioned earlier uses a heuristic function to estimate the cost of the cheapest path. And A* path planning algorithm based on bidirectional search is another optimization method [11].

4. Discussion

4.1. Existing Challenges and Limitations

As mentioned earlier, there are several challenges and limitations in the current path planning methods based on graph search algorithms.

Firstly, the time complexity issue. The A* algorithm, for example, has a high time complexity when dealing with large-scale environments. This limits its applicability in real-world scenarios where computational resources are limited. To address this issue, researchers have proposed various improved methods, such as GBFS . However, these methods may still have limitations in terms of performance and efficiency.

Secondly, the potential to get trapped in local optima. Since graph search algorithms explore the graph in a systematic manner, they may converge to a local optimum rather than the global optimum. This is especially true when the heuristic function is not properly designed or when the problem domain has a complex structure. To overcome this limitation, researchers have proposed various techniques, such as adjusting the weights of the heuristic function or using randomization methods. However, these techniques may have limited effectiveness in practice.

4.2. Future Research Directions

Despite the challenges and limitations, there are still many opportunities for further research in path planning methods based on graph search algorithms.

One direction is to develop more efficient and effective heuristic functions. A well-designed heuristic function can significantly improve the performance of path planning algorithms. Research in this area can focus on improving existing heuristic functions or developing new ones that are more suitable for specific problem domains.

Another direction is to explore hybrid path planning algorithms that combine the advantages of different algorithms. For example, a hybrid algorithm that combines the systematic exploration of graph search algorithms with the random exploration of probabilistic methods can potentially achieve better performance in practice.

Additionally, research can also focus on parallelizing path planning algorithms. With the increasing availability of multi-core and distributed computing systems, parallelizing path planning algorithms can potentially improve their computational efficiency.

Finally, more research can be done on applying path planning methods based on graph search algorithms to new problem domains. As technology advances, new application domains emerge, such as autonomous drones, self-driving cars, and robotics in tight spaces. Research in these areas can explore the use of graph search algorithms for path planning in these new domains.

In conclusion, path planning methods based on graph search algorithms are powerful tools for solving path-finding problems in various application domains. Despite the existing challenges and limitations, further research in this area has the potential to improve their performance, efficiency, and applicability in practice.

5. Conclusion

This chapter has provided an in-depth overview of path planning methods based on graph search algorithms, including BFS, DFS, and A*. These algorithms have been widely used in various application domains, such as map navigation, autonomous vehicle navigation, and pathfinding in computer games and simulations. BFS and DFS are two basic graph search algorithms that can be used for path planning. BFS explores nodes in a breadth-first manner, while DFS explores nodes in a depth-first manner. Both algorithms have their advantages and limitations, and their choice depends on the specific requirements of the application. The A* algorithm is an informed search algorithm that combines the best features of BFS and DFS. It uses heuristic functions to estimate the cost of the cheapest path from the current node to the goal node, and prioritize nodes with lower total costs. This makes A* more efficient and effective in finding optimal or near-optimal solutions in many problem domains. Research on path planning methods based on graph search algorithms has led to the development of various improved methods. For example, the GBFS algorithm and are improved A*-based methods that address the high time complexity issue of the A* algorithm in large-scale environments. Despite the progress made in path planning methods based on graph search algorithms,

there are still challenges and limitations that need to be addressed. For example, the high time complexity issue still remains a challenge when dealing with large-scale environments. Additionally, there is a potential risk of getting trapped in local optima, especially when the heuristic function is not properly designed.

References

[1] R. Tiwari, A. Shukla, & R. Kala. Graph Based Path Planning. IGI Global, 2013, 26-52

[2] J.R. Sánchez-Ibáñez, C.J. Pérez-Del-Pulgar, & A. García-Cerezo. Path Planning for Autonomous Mobile Robots: A Review. Sensors (Basel, Switzerland), 2021, 21(23), 7898.

[3] S. Li, R. Zhang, Y. Ding, X. Qin, Y. Han, & H. Zhang. Multi-UAV Path Planning Algorithm Based on BINN-HHO. Sensors (Basel, Switzerland), 2022, 22(24), 9786.

[4] O. Artiles, & F. Saeed. TurboBFS: GPU Based Breadth-First Search (BFS) Algorithms in the Language of Linear Algebra. IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum : [proceedings]. IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum, 2021, 520–528.

[5] G. Huang, L. Tan. Nearest Edge Data Obfuscation Algorithm Based on DFS[J]. Microelectronics & Computer, 2010, 27(2):49-54.

[6] Y. Ou, Y. Fan, X. Zhang, Y. Lin, W. Yang. Improved A* Path Planning Method Based on the Grid Map. Sensors (Basel). 2022, 22(16):6198.

[7] L. Yue.An improved A-Star algorithm[J].Science & Technology Information,2017, 15(36):17-18.

[8] S. Chen, Z. Teng, Q. Hong, Q. Chen. The Case Analysis of Four Shortest Path Algorithm[J]. Computer Knowledge and Technology, 2007, 3(16):1030-1032.

[9] D. Lim, J. Jo. Path planning with the derivative of heuristic angle based on the GBFS algorithm. Front Robot AI. 2022, 9:958930.

[10] X. Wang, Y. Qian, H. Gao, C.W. Coley, Y. Mo, R. Barzilay, & K. F. Jensen. Towards efficient discovery of green synthetic pathways with Monte Carlo tree search and reinforcement learning. Chemical science, 2020, 11(40), 10959–10972.

[11] J. Chen, Y. Xu, S.Chen, L. Zhang, Z. Cai. A Bidirectional Search A* Path Planning Algorithm Based on Dynamic Heuristic Method.Modern Information Technology. 2024, 02:86-91.