Proceedings of the 2nd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/76/20240596

A-star algorithm vs. Q-learning algorithm for path planning: A comparative analysis

Junxiao Gao

College of Engineering, China Agricultural University, Beijing, 100083, China

2021307150324@cau.edu.cn

Abstract. In robotics research, the core focus is on path planning, as it's essential for robots to navigate according to human objectives to be valuable. This paper provides an in-depth analysis of the commonly used A-Star algorithm and Q-learning algorithm in mobile robot path planning. It specifically examines the advantages and disadvantages of both the A-Star algorithm and the Q-learning algorithm, highlighting their roles in the advancement of robotic intelligence. The A-star algorithm is an algorithm with a heuristic function to guide path selection direction. Also, the A-star algorithm is guaranteed to identify the most efficient path in terms of traffic cost when facing static obstacles. However, its performance degrades when dealing with trap nodes. Specifically, the algorithm occupies a large amount of memory and takes a long computational time. In contrast, the Q-learning algorithm is effective for learning continuous actions with the same network data structure for local path planning tasks.

Keywords: Robot, Path planning, A-Star algorithm, Q-learning algorithm, Planning comparison.

1. Introduction

The robotics field has seen increased demand for automation in modern society, highlighting the necessity for robots to autonomously navigate within specific areas. Path planning is crucial for this autonomy, allowing robots to establish precise, feasible, and safe routes from starting points to endpoints, reducing human intervention. This technology is vital for mobile robots to perform complex tasks effectively.

Path planning involves creating an environment model as a map to generate efficient waypoints connecting starting and ending points while avoiding obstacles. There are two main approaches: global and local. Global path planning relies on existing environment knowledge, suitable for known or stable environments [1]. In contrast, local path planning utilizes sensor data, requiring more computational resources and may lead to suboptimal paths.

This research focuses on two primary global path planning algorithms: Q-learning and A-Star. Q-learning, an incremental dynamic programming method, iteratively refines search to find the shortest path but is limited by spatial and temporal complexity. A-Star, a map-based search algorithm, prioritizes conflict hotspot reduction to enhance path control and safety.

Comparing these algorithms aims to enhance operational efficiency in path planning. Q-learning introduces dynamic search factors for efficiency improvement, while A-Star focuses on conflict hotspot reduction for safety enhancement. The research suggests using Q-learning-based path planning when robot conflict levels exceed a threshold; otherwise, A-Star-based planning is recommended. This

DOI: 10.54254/2755-2721/76/20240596

approach offers insights into algorithm differences and their applications, contributing to a nuanced understanding of strengths and weaknesses in various scenarios.

2. A-Star algorithm

2.1. Description

2.1.1. Overview of the A-Star algorithm. In a static road network, the A-star algorithm stands out as the most efficient direct search technique for identifying the shortest path, which is formulated as follows:

$$F(n) = G(n) + H(n) \tag{1}$$

G(n) represents the initial node's movement cost to a given node n, while H(n) indicates the estimated cost from node n to the goal node, determined by the heuristic function. The choice of heuristic function significantly influences the resulting F(n) value in the A-star algorithm. When H(n) is significantly lower than G(n), F(n) will approximate G(n), resembling Dijkstra's algorithm. This can lead to exploring numerous nodes, increasing memory usage and computational time, thus reducing efficiency. Conversely, when H(n) is much larger than G(n), F(n) tends to mirror H(n), resembling Breadth-first search behavior. This may result in quickly finding a path, but not necessarily the shortest one.

2.1.2. How A-Star uses heuristics and a cost function to find the optimal path. In actual path planning, the actual consumption G(n) can be directly calculated, while the estimated consumption H(n) is only an estimation that needs to be determined based on empirical estimation of environmental characteristics. These characteristics are heuristic information, which can shorten the solution time. The A-Star algorithm also uses an evaluation function f(n) to select the nodes to be extended.

The flowchart of the A-Star algorithm is shown in Figure 1.

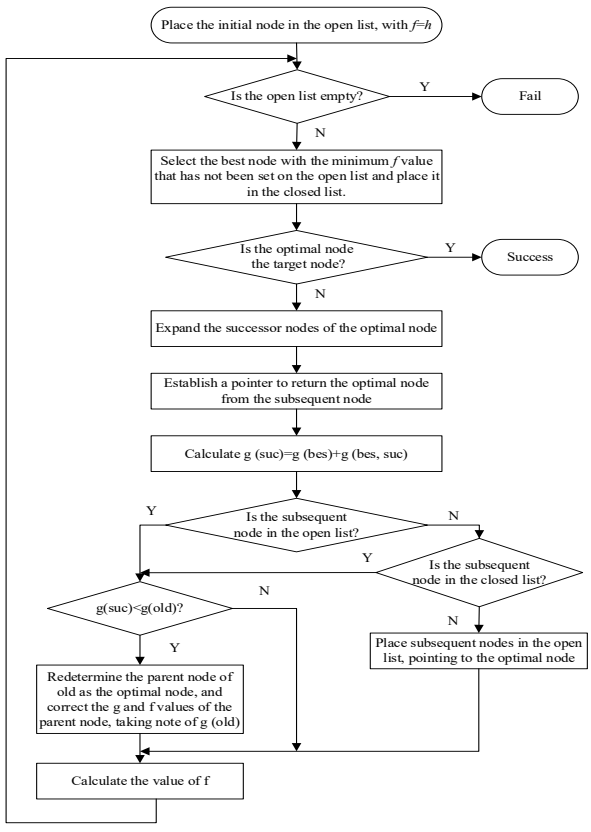

Figure 1. A-Star algorithm flowchart

DOI: 10.54254/2755-2721/76/20240596

In a uniform grid environment, establishing the evaluation function for the A-Star algorithm is straightforward, as the map's structure requires adapting path planning algorithms to match the environment's specific traits. The formula for calculating h(n) is:

$$h(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2} \tag{2}$$

In the equation, g(n) represents the geometric distance from node n to the initial node in the environment. This enables the creation of a global path using available environmental data in a static environment, ensuring the mobile robot safely reaches its target by avoiding locally detected obstacles. If the intended route is completely blocked, the robot switches from its initial route strategy to a comprehensive path planning approach.

2.2. Use cases

2.2.1. Examples of situations where A-Star is effective. Assuming that an obstacle suddenly appears in front of the robot during its movement, the robot will output an output angle and fixed step size based on the values of input variables d and θ, and the robot's rotation angle is φ [2]. But after the robot rotates, it does not immediately move forward due to not knowing the information of the obstacle ahead, otherwise it may collide with the obstacle. The robot must measure the distance to the obstacle ahead and then compare this distance with the predetermined step length required for the robot to move. If the distance from the obstacle is greater than the fixed step length, the robot moves forward with a fixed step length. Otherwise, the robot stays in place and continues to deflect the angle in the original direction to detect the distance between the robot and the obstacle, The advancement will not occur at a fixed pace until a safe direction is identified. The mobile planning of the robot is shown in Figure 2.

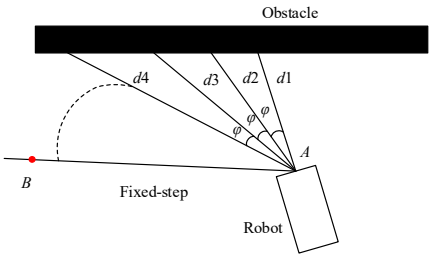

Figure 2. Robot Forward Planning

According to Figure 2, the robot is positioned at point A. Since the distance is smaller than the robot's fixed step size and the target point is assumed to be on the left side of the robot, it deflects to the left by an angle and continues detection. The obstacle distance in the direction of the robot's movement is still smaller than the fixed step size, so it deflects to the left again by an angle. This detection cycle continues until no obstacles are found within the fixed step size range. The robot then advances with a fixed step size to point B to continue the detection process.

An illustration of path planning using the A-star algorithm is depicted in Figure 3. The starting node is situated on the left, the target node on the right, and the gray color represents impassable obstacle nodes.

DOI: 10.54254/2755-2721/76/20240596

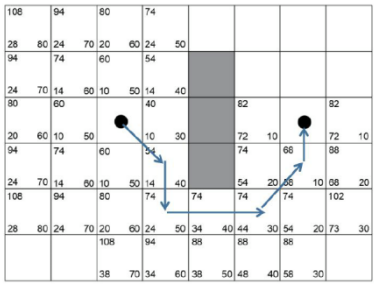

Figure 3. Schematic diagram of A-star algorithm planning

2.2.2. Discussion on real-time and deterministic nature of A-Star. Build a grid map for robot simulation on the MATLAB platform and write a simulation control program. All map information is stored in the map matrix, with black grids representing obstacle areas and blank grids representing areas that the robot can directly traverse, as shown in Figure 4.

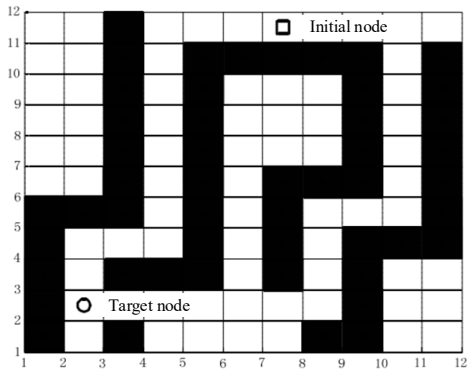

Figure 4. Simulation Environment Map

Using the A-star algorithm for robot path planning, the obtained robot motion path is shown in Figure 5.

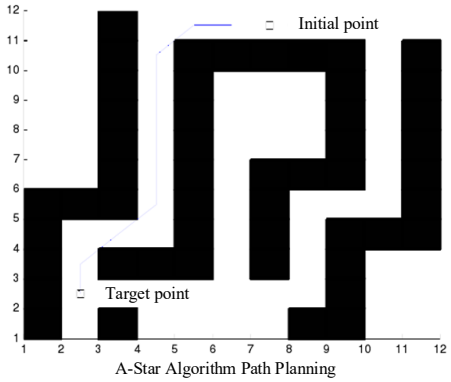

Figure 5. A-star Algorithm Path Planning Robot Running Route

According to robot operation roadmap 5, with multiple paths available, the robot can follow the shortest path. However, the A-Star algorithm also exhibits its drawbacks [3]. In A-Star algorithm planning, robots typically navigate along the edges, corners, and extremities of obstacles. In an ideal scenario, this path planning approach poses no significant issues for the robot. However, in practical environments, due to limitations such as robot walking errors and obstacle shapes, the A-Star algorithm is prone to causing collisions with obstacles, leading to path planning failure [4].

DOI: 10.54254/2755-2721/76/20240596

3. Q-Learning Algorithm

3.1. Explanation of how Q-Learning employs reinforcement learning to discover optimal paths

The Q-Learning algorithm is an unsupervised learning approach, can learn to select the optimal action through interaction with the environment to achieve the long-term goals of the algorithm [5]. The key elements of reinforcement learning algorithms include strategies, reward functions, value functions, and environmental models.

The robot using the Q-Learning algorithm determines its next action based on the current state of the environment and interacts with it after completing the corresponding action [6]. The environment assigns a reward value generated by the current action and updates it to a new state to learn the updated strategy. This process is repeated until the optimization strategy task is completed. Q-learning, as a typical model-independent algorithm, selects the next action based on the feedback value generated after each interaction with the environment. The implementation process of the Q-Learning algorithm is illustrated in Figure 6.

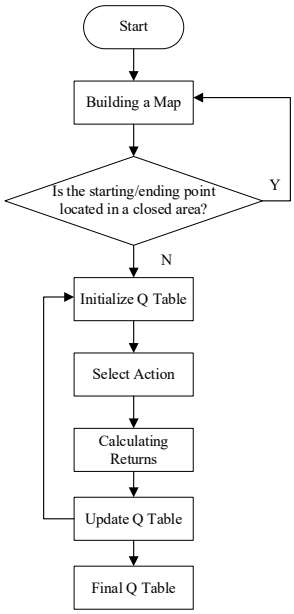

Figure 6. The Relationship between Reinforcement Learning System and Environment

3.2. Use cases

*3.2.1. Examples of scenarios where Q-Learning is applicable.*Q-learning is a type of reinforcement learning algorithm that operates without the need for environmental models and can be considered as a form of asynchronous dynamic programming [7]. When faced with unknown initial conditions, the Q-Learning algorithm must determine its course of action. Watkins regards the Q-Learning algorithm as an incremental dynamic programming, using a one-step approach to determine strategies and hoping to find a strategy (action sequence) that maximizes the sum of evaluations [8]. If the environmental model (i.e. state transition probability and evaluation model) is known or estimated from observations, the above problems can be solved using dynamic programming (DP) [9]. The concept behind the Q-Learning algorithm is to optimize a computable Q-function iteratively, rather than estimating the environmental model. Watkins defines this Q-function as executing actions at state time and then accumulating reinforcement values based on the discount when executing the optimal action sequence, namely:

$$Q_{t+1}(s_t, a_t) = r_t + \gamma \max_{a_t}\{Q(s_{t+1}, a_t) | a_t \in A\} \tag{3}$$

DOI: 10.54254/2755-2721/76/20240596

In robot path planning, adjusting the Q value based on real-time conditions is essential, which is:

$$Q_t(s_t, a_t) = \begin{cases} (1-\alpha_t)Q_{t-1}(s_t, a_t) + \alpha_t[r_t + \gamma V(s_{t+1})] \\ Q_{t-1}(s_t, a_t) \end{cases} \quad (4)$$

In the equation, there is:

$$V(s_{t+1}) = \underset{a \in A}{max}\{Q_{t-1}(s_{t+1}, a)\} \quad (5)$$

In the initial stage of learning, the Q value may not accurately reflect the strategy they define, and the initial value of $Q_0(s, a)$ is assumed to be given for all states and actions.

3.2.2. Discussion on learning and adaptation aspects of Q-Learning. Local path planning in robot navigation tasks is the process of dynamically devising a collision-free path for a robot from its initial position to its destination, particularly when faced with unknown or partially known environmental conditions, and ensuring that the path is optimal or suboptimal [10].

Reinforcement learning performance is primarily assessed based on two factors: convergence and convergence speed. This can be evaluated by improving the convergence trend and operating trajectory of the signal. Learning from a specific starting and endpoint demonstrates that the robot can swiftly acquire a collision-free navigation strategy through learning, resulting in a smooth path trajectory, as depicted in Figure 7.

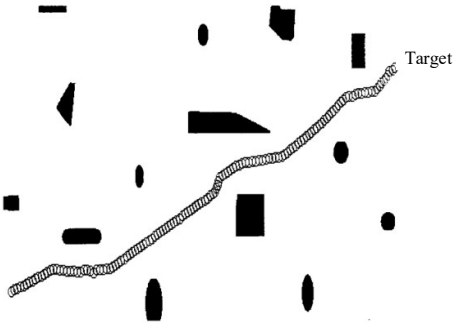

Figure 7. Q-Learning Algorithm Robot Path Planning

Figure 7 illustrates the trajectory of the robot within the simulation environment as it sequentially reaches a series of sub-targets, continues to advance towards the next sub target, and finally reaches the endpoint. During this period, the robot did not encounter any collisions.

4. Comparative analysis

By computing robot trajectories using the A-Star algorithm for path planning and evaluating the time taken for both the A-Star and Q-Learning algorithms to plan identical paths, we aim to compare their efficiencies. The running time for path planning is calculated for low and high conflict levels, respectively.

Table 1 illustrates the comparison of running data results for both algorithms under low and high conflict scenarios in path planning.

Table 1. Presents a comparative analysis of operational data results for two path planning algorithms.

Conflict situation	A-Star algorithm time	Q-Learning algorithm time
Low conflict	502s	471s
High conflict	516s	537s

DOI: 10.54254/2755-2721/76/20240596

The A-Star algorithm showed execution times of 502 seconds in low conflict scenarios and 516 seconds in high conflict scenarios. In contrast, Q-Learning recorded 471 seconds in low conflict and 537 seconds in high conflict situations.

Analyzing the results reveals a trend: Q-Learning is more efficient in low conflict scenarios, outperforming A-Star in execution time. Conversely, A-Star performs better in high conflict scenarios, showing shorter execution times than Q-Learning. The choice of algorithm may depend on conflict levels in robotic navigation contexts.

Both algorithms effectively plan optimal paths for robot navigation. Q-Learning is more efficient in low conflict situations, while A-Star addresses prolonged traversal times in high conflict scenarios. Dynamic path adjustment is utilized to resolve conflicts between different robot operations.

5. Conclusion

In conclusion, this research presents a comparative analysis of the A-Star and Q-learning algorithms for robot path planning, emphasizing their respective advantages in different conflict scenarios. Despite valuable insights, the study acknowledges limitations. Future enhancements may involve exploring hybrid models merging both algorithms' strengths, addressing identified computational inefficiencies, and validating theoretical findings through real-world testing. Subsequent research should prioritize enhancing these algorithms' adaptability in dynamic environments. This might entail integrating advanced machine learning techniques or real-time data analysis to effectively manage unexpected obstacles and environmental changes.

References

[1] Guo X, Peng G, Meng Y. A modified Q-learning algorithm for robot path planning in a digital twin assembly system[J]. The International Journal of Advanced Manufacturing Technology, 2022(5/6): 119. DOI: 10.1007/s00170-021-08597-9.

[2] Song Q, Li S, Yang J, et al. Intelligent Optimization Algorithm-Based Path Planning for a Mobile Robot.[C]//Comput Intell Neurosci.Comput Intell Neurosci, 2021:8025730. DOI: 10.1155/2021/8025730.

[3] Xianglong L U, Chundu W U, Guanxue Y, et al. Path Planning of Orchard Spray Robot Based on Improved A* and DWA Algorithm[J]. Computer Engineering and Applications, 2023, 59(18): 323-328. DOI: 10.3778/j.issn.1002-8331.2208-0003.

[4] Guo J, Huo X, Guo S , et al. A Path Planning Method for the Spherical Amphibious Robot Based on Improved A-star Algorithm[C]//IEEE International Conference on Mechatronics and Automation.IEEE, 2021. DOI: 10.1109/ICMA52036.2021.9512805.

[5] Meng H, Zhang H. Mobile Robot Path Planning Method Based on Deep Reinforcement Learning Algorithm[J]. Journal of circuits, systems and computers, 2022. DOI: 10.1142/S0218126622502589.

[6] Zhang C, Qin W, Fan M C, et al. A Q-Learning-Based Parameters Adaptive Algorithm for Formation Tracking Control of Multi-Mobile Robot Systems[J]. Complexity, 2022. DOI: 10.1155/2022/5093277.

[7] Zhao, M., Lu, H., Yang, S., & Guo, F. (2020). The Experience-Memory Q-Learning Algorithm for Robot Path Planning in Unknown Environment. IEEE Access, 8, 47824–47844. doi:10.1109/access.2020.2978077

[8] Watkins C J C H. Learning from delayed rewards[J]. 1989.

[9] Seo J, Na Y S, Kim B, et al. Development of an operation trajectory design algorithm for control of multiple 0D parameters using deep reinforcement learning in KSTAR[J]. IOP Publishing Ltd[2023-11-17]. DOI: 10.1088/1741-4326/ac79be.

[10] Xiong K, Wei C. Q-Learning-Based Target Selection for Bearings-Only Autonomous Navigation[J]. Journal of Systems Science and Complexity, 2021(5): 1-25. DOI: 10.1007/s11424-020-9265-y.