Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/45/20241028

Research on path planning of 4-link robot based on optimized Q-learning algorithm

Lili Liu

New H3C group, Hangzhou, China

liulili_0203@163.com

Abstract. In Q-learning algorithm, the balance between exploration and exploitation is crucial for the success of the entire algorithm. If the algorithm is too biased towards exploration, it may waste too much time on unnecessary paths, while if the algorithm is too biased towards exploitation, it may miss better paths. Ensuring the balance between exploration and exploitation in Q-learning algorithms requires comprehensive consideration of various aspects of the algorithm, including learning rate, state space representation, reward function design, policy selection, and addition of random noise. This paper proposes and verifies the construction of the state and action sets for a three-degree- of-freedom robotic arm within a specified space, where the reward values are reasonably set, and Q-learning with an introduced exploration factor is used as an improvement to prevent the roboticarm from insufficient exploration in action selection, which can lead to the algorithm failing to find a better strategy. The results of experiment illustrate that introducing the exploration factor can significantly enhance the convergence speed of the algorithm.

Keywords: Robot, Reinforcement Learning, Q-Learning Algorithm, Obstacle Avoidance, Path Planning.

1. Introduction

Biomimetic robots are robots designed and manufactured based on the principles of biomimetics. The research of biomimetic robots has gradually been applied to various fields such as robot arms, humanoid robots, legged robots, and autonomous flying vehicles. The potential of biomimetic robots in exploring the unknown, autonomous navigation, and artificial intelligence is increasingly evident, with the potential to significantly improve work efficiency and change our way of life.

The kinematic design of biomimetic robots is becoming more precise. The movement, operation, and mechanical design of biomimetic robots tend to mimic the human body structure, while giving robots more degrees of freedom in mechanical design to achieve operations that are difficult for humans to perform.

Advanced technologies such as sonar, cameras, and filters are being applied to biomimetic robots. Sensor technologies are also commonly used by biomimetic robots to simulate animal perception skills. For example, miniature robots can make more precise movement direction decisions based on the received information.

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/45/20241028

The learning ability of biomimetic robots is still being explored. Biomimetic robots usually have learning abilities and long-term self-updating and adaptation abilities, which fundamentally change the control method of traditional mechanical programs.

Medical robots are becoming more and more common. With the acceleration of the trend of an ageing population, the demand for medical robots is constantly increasing, and biomimetic robots, as the latest solutions for dealing with physiological conditions, are gaining more attention from medical workers. Biomimetic robots will also become increasingly miniaturized, intelligent, and multi-functional in the future [1].

Biomimetic robots need to operate in complex environments, so it is necessary to improve their ability to recognize and avoid external obstacles.

Path planning is an important technology in the field of robotics research. It is mainly used to guide robots to act autonomously in dynamic and unpredictable environments. It can be used in various robotic application scenarios, such as:

Automated manufacturing: On the automated assembly line in the factory, robots need to carry out material handling, assembly and other operations according to predetermined paths.

1. Logistics sorting: Robots need to identify and accurately grasp target items in logistics sorting and then transport them to designated locations along predetermined paths.
2. Agricultural robots: Agricultural robots can optimize planting, harvesting and other operationsthrough path planning guidance, improving the efficiency and quality of agricultural work.
3. Service robots: Service robots can patrol and serve the public according to planned paths through path planning.
4. Search and rescue: In disaster relief, robots can use path planning to guide search and rescue operations, act quickly, and improve search and rescue efficiency.

Path planning research in related fields began in the 1970s and primarily applied to robot obstacle avoidance. In recent years, significant advances have been made in path planning algorithms in this field. The development of path-planning algorithms for robotic arms can be divided into the following stages:

1. Path-planning based on geometric methods: Early path planning for robotic arms almost used mathematical geometry methods to directly calculate solutions to linear or nonlinear inverse kinematics problems to complete path planning, such as artificial potential fields [2, 3] and grid methods, Cao Chengcai et al. proposed a mobile robot path planning method based on geometry [4].
2. Path planning based on optimization methods: For complex motion planning problems, optimization methods such as optimization or constrained filtering methods are used to solve objectives such as minimizing energy or finding the shortest path to obtain the robotic arm's motion trajectory, including genetic algorithms and particle swarm optimization algorithms. Yushuke et al. proposed an ant colony genetic fusion algorithm for the path planning of mobile robots [5].
3. Path planning based on search algorithms: As robot application scenarios continue to expand, path planning algorithms begin to use heuristic search algorithms such as A*, D*, and RRT to efficiently and quickly solve robotic arm path planning problems and find the shortest distance or optimal trajectory. Jia Qingxuan et al. proposed obstacle avoidance path planning for a spatial robotic arm based on A* algorithm [6].
4. Path planning based on machine learning and artificial intelligence: In recent years, with the continuous development of deep learning and reinforcement learning technologies, path planning algorithms have gradually become integrated with machine learning, allowing for autonomous learningof suitable path planning to achieve more efficient and automated robotic arm path planning. Examplesinclude the Q-learning algorithm [7–10], CNN [11, 12], and DDPG [13, 14].

The research focus of path planning algorithms still lies in improving algorithm efficiency [15].

DOI: 10.54254/2755-2721/45/20241028

2. Related Work

2.1. Reinforcement Learning

Reinforcement learning is a branch of machine learning which is different from traditional Supervised learning and Unsupervised learning. It mainly focuses on the learning of agents in an interactive environment through repeated experiments by using their own behavior and experiencing feedback in the process of interacting with the environment.

The reinforcement learning algorithms commonly used in path planning include Q-learning, Deep Q-Networks, Deep Deterministic Policy Gradients, and so on.

In the field of machine learning, learning techniques can be divided into supervised learning, un-supervised learning, and reinforcement learning. Reinforcement learning is an iterative process of "exploration-exploitation", Reinforcement learning is a learning process in which an agent interacts with the environment to achieve a goal. Several important concepts in reinforcement learning include reward, agent, environment, and state. The characteristics of reinforcement learning are mainly:

(1) There is no supervisor during the learning process, only a reward signal.

(2) The feedback signal is delayed rather than instantaneous.

(3) The reinforcement learning process is related to time series and is a sequential decision-making process.

(4) The actions of the agent will affect the sequential data it receives.

Q-Learning is a value-based algorithm in reinforcement learning. Q refers to the expected reward obtained when taking action a ($a \in A$) in state s ($s \in S$) at a certain time. The environment provides corresponding reward r according to the agent's action. The main idea of the algorithm is to construct a Q-table that stores Q values for State and Action, and then select the action that can obtain the maximum reward based on the Q value.

Pseudo code:

1. Initialize Q-values as 0 or random values.
2. Repeat the following steps until a stopping condition is met:

a. Based on the Q-values, select an action a in the current state-s.

Execute action a to obtain a reward r and reach a new state s'.

b. Update the Q-value based on the formula:

$$Q(s, a) = Q(s, a) + \alpha(r + \gamma * max(Q(s', a')) - Q(s, a)) \quad (1)$$

c. Set the current state to the new state s'. The stopping condition may be reaching a maximum number of iterations or a small change in Q-values below a certain threshold. Represents the learning rate that controls the size of each update, and represents the discount factor, which balances the current reward and the expected future reward. Q(s, a) represents the expected reward obtained by selecting action a in state s,maxa'Q(s', a') represents the expected reward obtained by selecting the optimal action a' in state s'. The updated Q value will directly overwrite the original Q value.

At each decision point, choose the action with the maximum Q value to obtain the maximum immediate reward.This is the greedy strategy of Q-learning. This article attempts to optimize the Q-learning algorithm by increasing the exploration rate, which allows the robot arm to explore a wider range of state space and find better solutions.

2.2. Inverse Kinematics of Robotic Arm

Inverse kinematics of a robotic arm refers to calculating the angles of each joint needed to reach a desired position and orientation of the end effector (such as a gripper) that is set. The inverse kinematics problem of a robotic arm is a complex nonlinear problem that typically requires numerical methods to solve. Common methods include analytical methods and numerical methods. Analytical methods for inverse kinematics are used to derive analytical formulas of inverse kinematics by analyzing

DOI: 10.54254/2755-2721/45/20241028

the geometric shapes and constraint conditions of the robot arm. Numerical methods, on the other hand, use mathematical calculations to iteratively find solutions for inverse kinematics.

2.3. The Markov Decision Process

The Markov property refers to the fact that the next state of the system depends only on the current state and is independent of the previous states.

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, ..., S_t] \tag{2}$$

The Markov decision process consists of a quintuple <S, A, P, R, γ>.

S stands for a finite set of states;

A represents a finite set of actions;

P is the transition probability matrix, with

$$P^{a'} = P[S_{t+1} = s'|S_t = s, A_t = a] \tag{3}$$

R is the reward function;

γ is the discount factor, with $\gamma \in [0,1]$.

The policy is the probability distribution of actions given a state,

$$\pi(a|s) = P[A_t = a|S_t = s] \tag{4}$$

The definition of a value function is the expected future return of the Markov Decision Process when in a current state s. The value function provides the long-term value of the current state.

There are two policy-based value functions in the Markov Decision Process: (1) the expected return when in state s, representing the value of the state, (2) the expected return when taking action a in state s, representing the value of the action. These two functions consititute the value function of the Markov Decision Process. Gt is the reward value at time t.

The state-value function:

$$v_\pi(s) = E[G_t|S_t = s] \tag{5}$$

The optimal state value function of state s.

$$v_*(s) = max_\pi v_\pi(s) \tag{6}$$

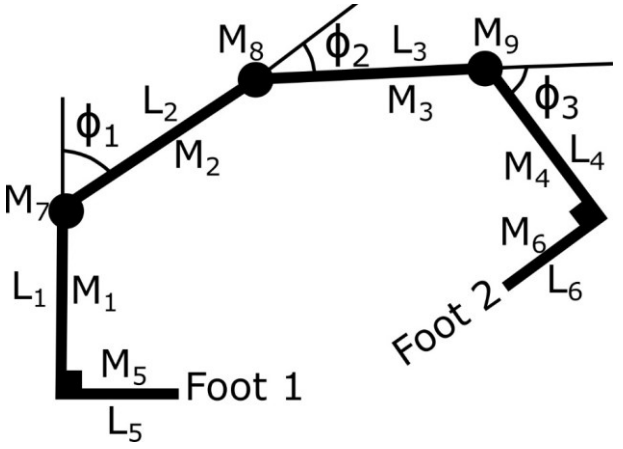


Figure 1: The 4-link robotic arm diagram.

The Action-Value Function:

$$q_\pi(s, a) = E[G_t|S_t = s, A_t = a] \tag{7}$$

The optimal action value function of taking action a in state s.

$$q_*(s, a) = max_\pi q_\pi(s, a) \tag{8}$$

DOI: 10.54254/2755-2721/45/20241028

The goal of reinforcement learning is to find the optimal policy in a given Markov decision process that maximizes the expected cumulative reward under that policy [11].

3. Research Method

3.1. The Q-learning algorithm

Q-learning is a reinforcement learning algorithm based on the Markov decision process. In Q-learning, we use a q-value function to estimate the value of each state-action pair and choose the next action based on these values. It discovers the optimal strategy by interacting with the external environment.

In the Q-learning algorithm, the agent needs to learn in the Markov decision process to decide what action it should take. The Q-learning algorithm updates the learning value based on the difference between the target q-value and the current estimated Q-value, and iteratively updates its strategy. In this way, Q-learning can learn the optimal strategy in a Markov decision process, enabling the agent to obtain the maximum reward in the environment.

3.2. Experiment Method

This article simulates a 2D 4-bar linkage bipedal (As shown in figure 1) robot by using MATLAB, fixing either foot1 or foot2 to the ground, setting obstacles within the plane area, and using the Q-learning algorithm to enable the robot to learn to avoid obstacles along the desired path. The experimental purpose is to use the Q-learning algorithm to enable the robotic arm to learn how to avoid obstacles within a specified range of movement, with the position of the obstacles being freely definable.

In the Q-learning algorithm, the reward value refers to the return value obtained after the action is taken. Its meaning is that when an agent takes a certain action, it will receive a reward value based on the environment's feedback, which represents the quality of the agent's current action. In order to learn the optimal action strategy through Q-learning, we need to use this reward value to update the Q-value of the state and action. When choosing the reward value, we should make it as clear and specific as possible to represent the quality of the agent's action, taking into account both positive and negative performances.

Table 1: Parameter settings for simulation experiments.

parameters	value
α	0.3
γ	0.9
End Reward	300
Punishment	-50
steps	40
Max Iteration	500

3.3. Improved Q-learning algorithm

In a standard Q-learning algorithm, the greedy strategy refers to selecting the action with the highest Q-value at each state as the next action. The greedy strategy assumes that there is enough information to choose the optimal action at the current state, without exploring unknown spaces. Therefore, the advantage of the greedy strategy is that it can efficiently use the known information to find the optimal solution, but the disadvantage is that it may fall into suboptimal or local optimal solutions because the algorithm does not fully explore unknown spaces. In practice, the greedy strategy is usually combined with other search strategies (such as the ε-greedy strategy) to ensure that the algorithm can find the global optimal solution while utilizing known information and exploring unknown state and action spaces.

This article introduces an exploration factor to increase the probability of algorithms choosing non-optimal actions, thereby enhancing their exploratory behavior.

Pseudo code:

DOI: 10.54254/2755-2721/45/20241028

1. Initialize Q-values as 0 or random values.
2. Repeat the following steps until a stopping condition is met:

a. Choose the next action a using an epsilon-greedy policy based on Q, with probability ε to randomly select an action, and (1-ε) to choose the action with the highest Q-value

b. Execute action a to obtain a reward r and reach a new state s'.

c. Update the Q-value based on the formula:

3.4. Experiment Result Analyze

The parameter settings in the simulation experiment are shown in Table 1.

In this experiment, the goal of the robotic arm is to move to the target position while avoiding collisions with obstacles. Therefore, we can design the reward function as follows:

- When the robotic arm reaches the target position, a large reward value is given to encourage therobotic arm to move towards the target position.
- When the robotic arm collides with an obstacle, a negative reward value is given to punish therobotic arm for moving in the wrong direction.
- For other positions in the path, a small positive reward value can be given to encourage therobotic arm to move towards the target position.

The weight and specific values of the reward function are adjusted to balance exploration and exploitation strategies, as well as to optimize the convergence speed of the Q-value function.

Under the same parameter conditions, as shown in Figures 2 and 3, the algorithm did not converge, and the robotic arm failed to find the optimal obstacle avoidance path in 500 iterations.

The change in total reward value reflects the convergence of the algorithm.By comparing Figures 2 and Figures 4, it can be observed that with the introduction of the exploration factor, the algorithm begins to converge after about 180 iterations, and the total reward value gradually stabilizes.

The exploration steps reflects the cost of the learning algorithm in the process of searching for optimal solutions. By comparing Figures 3 and Figures 5,it can be observed that with the introduction of the exploration factor, the algorithm begins to converge after about 180 iterations, the actual exploration steps significantly decreases.

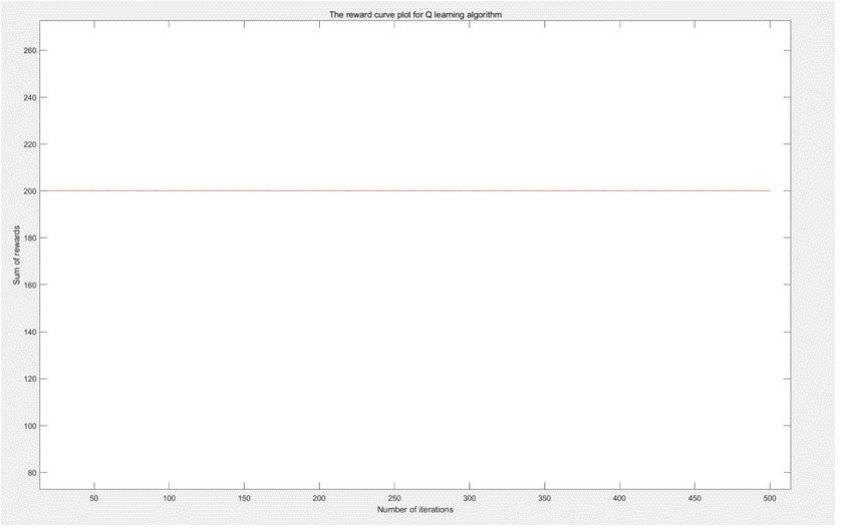


Figure 2: The rewards graph based on Q-learning algorithm.

DOI: 10.54254/2755-2721/45/20241028

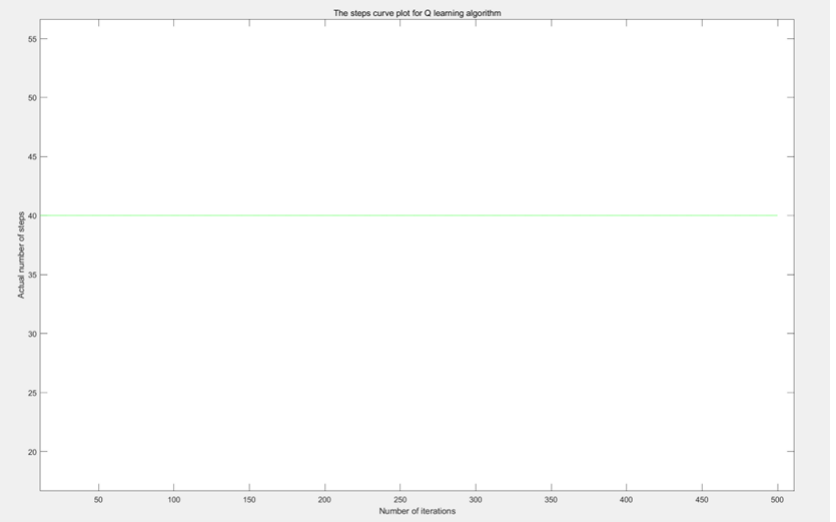


Figure 3: The steps graph based on Q-learning algorithm.

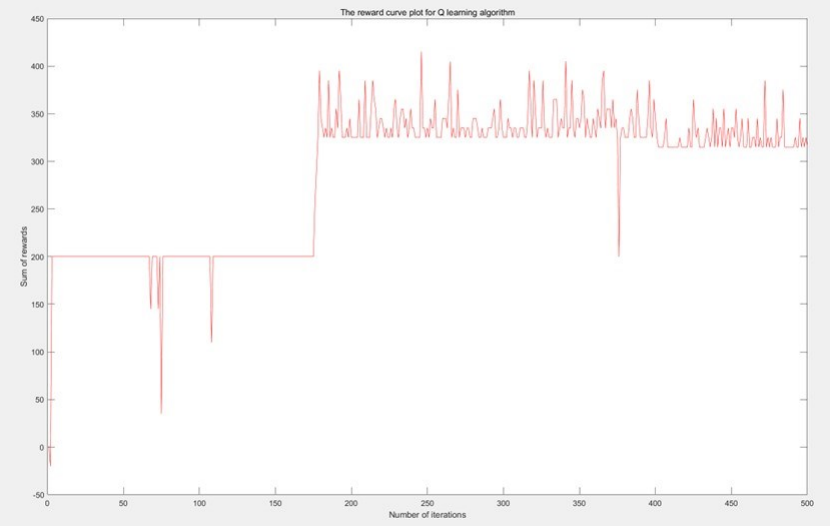


Figure 4: The rewards graph based on improved Q-learning algorithm.

DOI: 10.54254/2755-2721/45/20241028

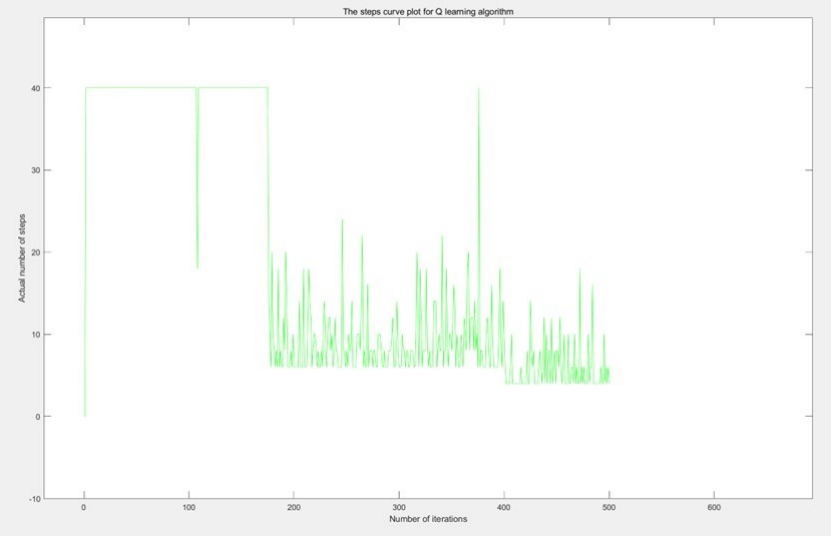


Figure 5: The steps graph based on improved Q-learning algorithm.

4. Conclusion

The Q-Learning algorithm with a greedy strategy that always selects the maximum value at each step has the problems of slow convergence speed and easy convergence to local optimization in path planning. The function of the exploration factor is to prevent the algorithm from converging to a local optimum, because in some cases, the algorithm may be attracted to a local optimum and be unable to find a better solution. The existence of the exploration factor helps the algorithm find a balance between exploration and exploitation, thereby better finding the global optimum. By introducing the exploration factor ε, it effectively expands the search space of the agent and improves the algorithm efficiency. The improved ε-Q-Learning algorithm has shown varying degrees of advantages over existing Q-Learning algorithms regarding convergence speed, search consumption.

Although Q-learning algorithm has achieved some success in path planning applications, in addition to the problems of low learning efficiency and slow convergence speed, there are still some problems and challenges that need further research and improvement:

Computational complexity: The computational complexity of Q-learning algorithm is relatively high, especially in high-dimensional environments, which may lead to slow convergence speed of the algorithm or failure to find the optimal solution.

Data volume: Q-learning algorithm requires a large amount of data to learn, which may lead to poor learning effect of the algorithm or require a lot of time and computing resources.

Environmental changes: Q-learning algorithm is sensitive to environmental changes. If the environment changes, it may need to learn again.

References

[1] Wanglihui. Research status and development direction of bionic robots. *Journal of Shanghai Normal University: Natural Sciences*, 36(6):5, 2007.

[2] Lianxiaofeng, Liuzaiwen, and Zuomin. Research on path planning method of mobile robots basedon dynamic artificial potential field. *Computer simulation*, 28(1):5, 2011.

[3] Zhujing and Yangmaying. Obstacle avoidance path planning for robot arm based on improved artificial potential field method. *Computer automatic measurement and control*, 2018.

[4] Caochengcai, Yaojin, and Wangqiang. Path planning for mobile robots based on geometry. *Com-puter Application Research*, 22(12):5, 2005.

[5] Yushuke, Zhaiguoqing, Qihongyu, and Zhaokaixin. Application of ant colony genetic fusion algorithm in path planning of mobile robots. *Firepower and Command Control*, 42(12):4, 2017.

DOI: 10.54254/2755-2721/45/20241028

[6] Jiaqingxuan, Chengang, Sunhanxu, and Zhengshuangqi. Obstacle avoidance path planning for spatial robotic arm based on a* algorithm. *Journal of Mechanical Engineering.*, (13):7, 2010.

[7] Guoxinlan. Research on trajectory planning and obstacle avoidance behavior of robotic arm basedon q-learning algorithm. *Machine Tool and Hydraulic*, 49(9):6, 2021.

[8] Maoguojun and Gushimin. Improved q-learning algorithm and its application in path planning. *Journal of Taiyuan University of Technology*, 2021.

[9] Dongyao, Geyingying, Guohongyong, Dongyongfeng, and Yangchen. Mobile robot path planningbased on deep reinforcement learning. *Computer Engineering and Applications*, 55(13):6, 2019.

[10] Wangdingxin. Path planning of agv based on improved q-learning algorithm. *Electronic DesignEngineering*, 029(004):7–10,15, 2021.

[11] Liguangchuang and Chenglianglun. Obstacle avoidance path planning for robotic arm based on deep reinforcement learning. *Software Engeering*, 22(3):4, 2019.

[12] Gupeng. *Research on Obstacle Avoidance Strategy of Hexapod Robot based on Deep Learning*. PhD thesis, Southwest University of Science and Technology, 2019.

[13] Zhangning and Gebin. Application of improved ddpg algorithm in robot path planning. *Journal of Jiamusi University (Natural Science Edition)*, 039(005):119–122, 2021.

[14] Zhoushengshi, Shanliang, Changlu, Chenjia, Liuchenglu, and Lijun. Research on robot path planning algorithm based on improved ddpg algorithm. *Journal of Nanjing University of Science and Technology*, 45(3):7, 2021.

[15] Yangjuncheng, Lishuxia, and Caizengyu. Research and development of path planning algorithms. *Control Engineering*, 24(7):8, 2017.