

Robot Path Planning in Unknown Environments based on a Learning-guided Optimization Approach

Shi Cheng, Zhixin Wang, Jian Yang*, Yifei Sun, Mingming Zhang, Sicheng Hou, Hui Lu, Chaomin Luo, and Yuhui Shi

Abstract: A learning-guided optimization approach, *i.e.*, a Q -learning-guided memetic algorithm (QLMA), is proposed to solve path planning problems in unknown environments. Traditional path planning algorithms in unknown environments have the disadvantages of low efficiency and stability problems. Integrating learning strategies into a memetic algorithm is a natural way to enhance search performance. The proposed algorithm can be divided into two stages: learning and optimization. A Q -learning algorithm implements the learning stage, while the optimization stage is implemented by hybridization of a genetic algorithm (GA) and simulated annealing (SA) algorithm, *i.e.*, a memetic algorithm. The learning stage aims to acquire information from current search states. The Q -Learning algorithm's environmental learning capability is formed by rewards received from changes in distance to the target. Therefore, the path strategy could be adjusted continually, and effective paths could be identified. The optimization stage aims to utilize the obtained information and enhance the quality of known paths. A genetic algorithm improves the global search ability, while a simulated annealing algorithm enhances the local search ability. The experimental study was conducted on 30 test scenarios with three types of index points. The proposed algorithm outperforms the other two algorithms significantly in the test scenarios.

Key words: Developmental swarm intelligence; robot path planning; learning & optimization; memetic algorithm; unknown environments

- Shi Cheng and Zhixin Wang are with the School of Artificial Intelligence and Computer Science, Shaanxi Normal University, Xi'an, 710119, China. E-mail: {cheng,2023wzx}@snnu.edu.cn
- Jian Yang and Yuhui Shi are with the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), Shenzhen, 518055, China. E-mail: {yangj33,shiyh}@sustech.edu.cn
- Yifei Sun and Sicheng Hou are with the School of Physics and Information Technology, Shaanxi Normal University, Xi'an, 710119, China. E-mail: {yifeis,sicheng_hou}@snnu.edu.cn
- Mingming Zhang is with the Department of Mechanical and Mechatronics Engineering, The University of Auckland, Auckland, 1010, New Zealand. E-mail: mzha509@aucklanduni.ac.nz
- Hui Lu is with the School of Electronic and Information Engineering, Beihang University, Beijing, 100191, China. E-mail: mluhui@buaa.edu.cn
- Chaomin Luo is with the Department of Electrical and Computer Engineering, Mississippi State University, Starkville, MS 39762, USA. E-mail: chaomin.luo@ece.msstate.edu

* To whom correspondence should be addressed.

Nomenclature

QLMA	Q -Learning guided Memetic Algorithm
RL	Reinforcement learning
SARSA	State-Action-Reward-State-Action
SI	Swarm Intelligence
MA	Memetic Algorithm

1 Introduction

With large models, embodied intelligence and autonomy [1] has addressed many requirements for customized and autonomous tasks in sustainable cities. For example, finding a satisfying path for a robot in an unknown environment is challenging for autonomous

systems. Robot path planning aims to identify an ideal route on all collision-free roads from the starting point to the target point [2–4]. Robots' path planning in unknown environments is a significant challenge for robot applications in real-world scenarios [5]. Robots must identify collision-free paths to the target point in the shortest possible time. Autonomous path planning is the foundation of navigation, and most robot tasks rely on navigation in unknown environments. Therefore, solving navigation challenges would significantly enhance the practical applications of robots in areas such as deep space exploration or deep-sea resource investigation [6].

With the extensive application of robots in logistics, intelligent services, industrial production, and rescue detection, enhancing their autonomous navigation capabilities in complex environments has become increasingly important. Consequently, path planning technology, a core component of autonomous navigation [7], has emerged as a focal point in robotic research. By improving key performance indicators of path planning, such as path length, path smoothness, and planning time, it is possible to ensure that robots can safely complete designated tasks. In an unknown environment, if a robot lacks adaptability to its surroundings, it may fail to accomplish its task, damaging the working environment and incurring unnecessary losses.

Reinforcement learning (RL) [8, 9] and swarm intelligence (SI) algorithms share a common "trial-test-verify" framework. Based on interactions with problems (environments), the algorithm finds sufficiently good solutions by leveraging knowledge acquired from previously solved problems. The RL algorithm focuses on interacting with the environment and updating rewards from different states. In contrast, the SI algorithm focuses on cooperation among all individuals. The hybrid of RL and SI algorithms can combine the strengths of the two types of algorithms.

This paper proposes a learning-guided optimization-based robot path planning algorithm, namely *Q*-learning-guided memetic algorithm (QLMA), which combines the advantages of the *Q*-learning algorithm and a memetic algorithm to solve robot path planning problems in unknown environments. The global search ability of the genetic algorithm and the local search ability of the simulated annealing algorithm are combined in the memetic algorithm. This paper intends to attain two targets:

- (1) For researchers or practitioners who need to handle robot path planning problems in unknown environments, this paper provides a learning-guided approach to solve this problem.
- (2) This paper introduces a design approach to learning-guided memetic algorithms for researchers in swarm intelligence. By combining the advantages of knowledge acquisition and utilization, performance on complex problems could be enhanced.

Two research questions are addressed in this paper:

Q1: How efficient is the performance of the proposed QLMA algorithm on robot path planning problems? The ability of the proposed algorithm to operate in an unknown environment should be verified.

Q2: What's the function of each component in the proposed QLMA algorithm? The function of each component should be analyzed and verified.

The remainder of this paper is organized as follows. Section 2 introduces robot path planning problems and learning-guided optimization. The details of the proposed *Q*-learning-guided memetic algorithm are stated in Section 3. Section 4 examines the experimental results of solving robot path planning benchmark problems and analyzes the proposed algorithm's convergence and scalability abilities. Section 5 concludes the learning-guided swarm intelligence algorithms and provides prospects on the future directions.

2 Background

2.1 Robot Path Planning

Figure 1 shows an iterative process of perception, cognition, decision, and action for autonomous robots in unknown environments. Intelligent behaviors, such as path planning, are continually improved through multiple iterative stages. The robot perceives the data from the environment, then cognitively extracts valuable information from the data, makes a decision based on the information, and verifies the decision through action.

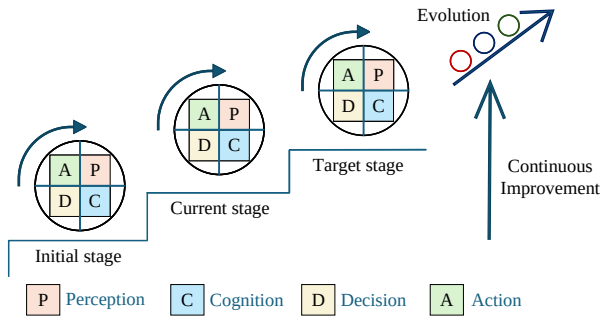


Fig. 1 An iteration process of perception, cognition, decision, and action.

Various algorithms have been proposed to solve robot path planning problems. These algorithms can be broadly classified into four main categories: fuzzy logic-based algorithms, simultaneous localization and mapping (SLAM) algorithms [10], intelligent optimization algorithms, and reinforcement learning algorithms. Table 1 compares four approaches for robot path planning in unknown environments.

Although many algorithms perform well in simplified or well-defined scenarios, several challenges remain in complex and unknown environments. The first challenge is entirely unknown environments. Robots relying solely on universal behavioral rules may not effectively ensure safety in unknown environments. The second challenge is complex environments. Algorithms are easily getting stuck in local optima in complex environments, potentially preventing robots from finding optimal paths during navigation. The search efficiency is also a challenge. Most algorithms have a slow convergence speed, which is not efficient for real-world path-planning problems. Therefore, there is a requirement for a path-planning algorithm with a generalization capability, global search ability, rapid convergence, and low computational complexity in unknown environments.

2.1.1 Fuzzy Logic-based Algorithms

Fuzzy logic algorithms have been widely used in various applications [20]. For example, fuzzy sets were utilized to represent multiple states and environmental features in path planning [21]. Each feature is quantified through a membership function, allowing each state to belong to these fuzzy sets to varying degrees.

The fuzzy algorithms store behavioral rules that the robot employs in various scenarios. The robot obtains information about its surroundings from its sensors and then queries the fuzzy rules for path planning in an

unknown environment. Map building and navigation are two foundational processes in robot path planning. Fuzzy logic could be used to build and modify the environment map; then, the A^* algorithm could plan a path based on the local information [11]. For the autonomous robot, the human driving behavior could be emulated and abstracted as fuzzy-logic rules to solve motion planning problems [12]. The fuzzy model can also be applied to multiple robots. Human thought could be mimicked to build a fuzzy model with two behaviors: path following and group cooperation [13].

Fuzzy logic algorithms can utilize partial prior knowledge to assist robots in planning safe paths within unknown environments and have been successfully applied in factory automation. However, the design of these algorithms often involves complex control theories. Furthermore, fuzzy rules rely on human expertise and are closely related to specific robotic application scenarios. As a result, the generalization ability of fuzzy logic algorithms across different environments is limited.

2.1.2 Simultaneous Localization and Mapping Algorithms

The core concept of simultaneous localization and mapping (SLAM) is to derive the robot's observation and motion models by collecting external data, thereby enabling real-time estimation of the robot's movement state [15]. Because the algorithm design considers sensor feedback information, it has been widely adopted in practical applications. Various variants of SLAM algorithms have been proposed to improve the accuracy of the estimation model. The SLAM problem within probability theory was framed, and Bayesian filtering methods were utilized for state estimation [14]. This approach allows robots to progressively update their positions and maps by fusing sensor data with motion models, thereby enhancing the reliability of SLAM in uncertain environments.

Combining a stereo visual-inertial system with loop closure detection can enhance the positioning and mapping capabilities of SLAM systems in dynamic environments [22]. Additionally, the concept of topological mapping enhances the understanding of spatial relationships by constructing a topological structure of the environment. Topological information enables robots to plan paths more effectively during navigation, particularly in complex environments. A graph-based SLAM framework utilizing graph theory

Table 1 Comparisons between typical algorithms for robot path planning in unknown environments.

Algorithms	Strengths	Weakness	Examples
Fuzzy logic	Get close to the human way of thinking, Flexibility	Computational complexity	[11–13]
SLAM	Simultaneous localization and mapping	Environmental dependencies	[14, 15]
Intelligent optimization	Strong searching ability	Requires a lot of computing resources and time	[3, 16–18]
Reinforcement Learning	No prior knowledge of any environment is required	Convergence speed is low	[19]

was developed to efficiently manage map construction and localization issues [23]. This graph structure enables the system to more effectively manage and optimize positional and map information, thereby enhancing the performance and efficiency of SLAM systems and providing new directions for future research. The Blitz-SLAM method improves the performance of SLAM systems in complex scenes by integrating semantic information, efficient feature matching, enhanced loop closure detection, and real-time performance optimization [24]. These innovations render Blitz-SLAM highly valuable in intelligent robotics and autonomous driving applications.

When robots utilize SLAM for path planning in unknown environments, differential matching of point clouds serves as a fundamental basis for localization and map construction. However, frequent matching computations, while enhancing localization accuracy, consume significant time, reducing planning efficiency. Therefore, if real-time environmental map construction is unnecessary during the planning process, the potential of SLAM technology remains underutilized.

2.1.3 Intelligent Optimization Algorithms

Intelligent optimization algorithms, including genetic algorithms, immune algorithms, ant colony algorithms, particle swarm optimization, and brain storm optimization [25] are all inspired by natural phenomena or biological populations. These algorithms enable robots to achieve robust optimization capabilities through iterative processes. Research on robot path planning is abundant, demonstrating the wide application and effectiveness of these algorithms in this field [26].

Robot path planning in an uncertain environment

is usually modeled as an optimization problem and solved by intelligent optimization algorithms. For example, the risk degree and the path distance are considered two objectives and solved by particle swarm optimization [3]. Incorporated with problem-specific knowledge, an adaptive evolutionary planner/navigator was proposed to solve path planning and navigation problems [16].

However, this process often incurs substantial time costs despite the significant improvement in path optimization capabilities through iterative calculations. Additionally, intelligent optimization algorithms typically encompass multiple parameters, and the selection of these parameters directly impacts algorithm performance. Consequently, such algorithms are generally more suited for theoretical research on path planning rather than practical application scenarios.

2.1.4 Reinforcement Learning Algorithms

In recent years, reinforcement learning algorithms based on trial-and-error mechanisms have gained widespread attention due to their independence from prior data for planning optimal or suboptimal paths. The core objective of these algorithms is to maximize cumulative rewards within a specific environment. Specifically, when the execution of an action enhances the robot's reward, the probability of selecting that action in subsequent learning processes increases correspondingly. Reinforcement learning methods include Q -Learning, state-action-reward-state-action (SARSA), deep Q -networks, and deep deterministic policy gradient, which have been effectively applied in robot path planning over the past few years.

Traditional methods for path planning in unknown

environments primarily utilize ant colony algorithms or genetic algorithms. However, when confronted with complex, high-dimensional environments, these approaches often become trapped in local optima or fail to effectively manage intricate scenarios. Consequently, reinforcement learning has garnered significant attention in recent years due to its ability to interact directly with the environment without prior knowledge and to update behavioral strategies based on the feedback it receives.

Reinforcement learning has been applied in path planning for intelligent robots, yielding significant results. For instance, a Q -Learning algorithm with an experience memory mechanism was proposed to address the slow convergence and lengthy path planning issues encountered when using Q -Learning in unknown environments [19].

2.2 Learning-Guided Optimization

Learning from the properties of problems and algorithms is beneficial for designing efficient optimization algorithms. With the learning ability of Q -learning algorithms, a swarm optimization algorithm could solve a problem more efficiently.

2.2.1 Swarm Optimization

As a group of intelligent algorithms, the swarm intelligence method contains multiple competitive and cooperative agents to solve modeling [4, 27], optimization [28], and learning [29, 30] problems in scientific computing and engineering areas [31], such as sampling method for autonomous systems [32], and decision boundary assessment problems [33].

Usually, based on the implicit information of the solved problem and the preference priority, the optimization processes can be divided into three approaches: a priori, a posteriori, and an interactive (progressive). Figure 2 shows an illustration of three forms of optimization. All information about the problem is known in the prior approach, *i.e.*, the properties of problems are static in the optimization process. Thus, the traditional optimization method is a good choice for this problem. Partita information is known in posterior and interactive approaches; the solving process is combined with the learning process. The hidden information of the problem is obtained during the previous solving process, which could guide the search in the next iteration.

Like reinforcement learning, optimization based on posterior and interactive approaches takes a Bayesian

approach to solve the problem. The posterior and interactive approaches are suitable for problems with partial or uncertain information. With the swarm intelligence algorithm or intelligent optimization algorithms, knowledge is obtained from previous searches, and this knowledge guides the current search. The solution will be found by applying knowledge acquired and utilized in the “trial-test-verify” process.

2.2.2 Q -learning algorithm

The Q -learning algorithm is a classic value-based reinforcement learning method that excels in solving complex decision-making problems [34]. It records learning strategies for agents and indicates which actions yield maximum rewards. For any finite Markov decision process [35], Q -Learning can find a policy that maximizes rewards. In this context, Q denotes $Q(s, a)$ for state(s), representing the expected rewards [36] for action (a) where ($s \in S$) and a ($a \in A$). The environment provides feedback based on the agent’s actions, and rewards are given accordingly. The algorithm constructs a Q -table to store Q -values, enabling it to select actions that maximize rewards based on these values. The Q -Learning model and Q -table structure, which includes (m) states and (n) actions, are illustrated in Figure 3, with zero representing the initial value.

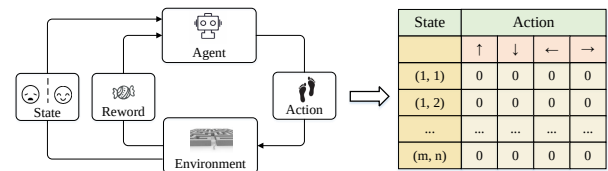


Fig. 3 Q -Learning model and Q table structure.

State, action, reward, and Q -table are the four essential components of Q -Learning, and the definition of these elements is closely related to their application. In robot path planning applications, the robot’s actions are usually defined as the direction in which it can move. In contrast, the state nodes of the path environment are defined as the coordinates, and the reward design is related to the robot’s current state nodes [37]. For example, if a robot reaches a location near an obstacle, it will receive a negative reward. Conversely, the robot will receive a positive reward if it reaches the target node.

The core of the Q -Learning algorithm lies in selecting actions from the Q -table. To calculate the Q -table, the Bellman equation was utilized to simulate the

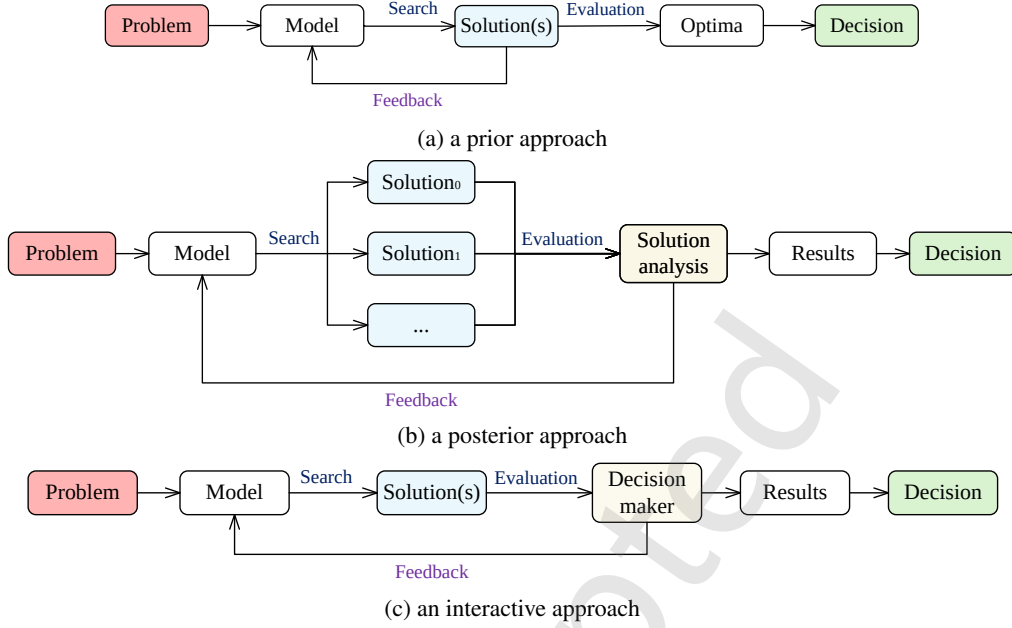


Fig. 2 Comparison on three approaches for optimization: (a) a prior approach, (b) a posterior approach, (c) an interactive approach.

agent's learning process in the Q -Learning algorithm. The definition is as follows.

$$Q(s_{t+1}, a_{t+1}) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

where α is the learning rate, γ is the discount factor, r_{t+1} is the immediate reward, $\max_a Q(s_{t+1}, a)$ is the delayed reward. If the agent performs the s_t action a_t at time t , it will receive an instant reward r_{t+1} and a delayed reward $\max_a Q(s_{t+1}, a)$. Algorithm 1 gives the basic process of the Q -learning algorithm.

Algorithm 1: The basic process of Q -Learning Algorithm

```

1 Initialize  $\alpha, \gamma$  and  $Q$ -table;
2 if  $episode \leq \max episode$  then
3   Initialize  $s_t$ ;
4   while  $s_t$  is not terminal do
5     Choose  $a_t$  using policy derived from the  $Q$ -table;
6     Execute action  $a_t$ , observe  $r_{t+1}$  and  $s_{t+1}$ ;
7     Update the  $Q$ -value via Equation (1);
8    $s_t = s_{t+1}$ ;
```

2.2.3 Learning-Guided Swarm Intelligence

Learning from unknown environments is an efficient way to solve problems with uncertainty. The learning process requires an algorithm with two abilities: knowledge acquisition and knowledge utilization.

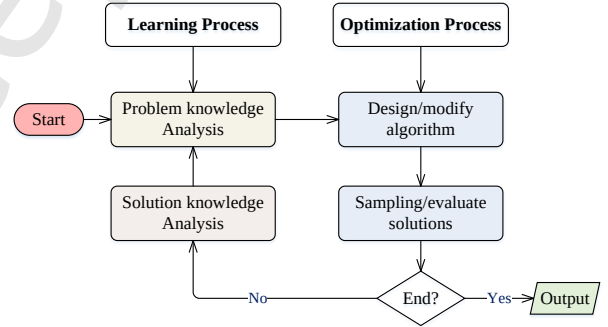


Fig. 4 A framework of learning-guided optimization.

Swarm optimization is a dynamic process that iteratively updates solutions until the end criteria are met. The learning process can guide the design and modification of the algorithm in the optimization process [38]. Similar concepts have been discussed in machine learning and evolutionary computation with different terms. For example, in the machine learning field, the learning to learn [39], learning to optimize [40–42], or meta-optimizer [40] are used to describe the optimization algorithm designing via machine learning methods, while in the evolutionary computation field, automatic design [43, 44] indicates the process of automated designing optimization algorithms based on training on problems samples.

A black-box optimization algorithm was abstracted into an iteration with three steps in the learning-to-learn framework [39]. The first step is to generate a point based on the current state of knowledge, the second

step is the response, and the last step is to produce new knowledge by updating internal statistics. Compared with the terms in evolutionary computation, the point is an individual in the search space; the response indicates the individual's fitness, *i.e.*, the quality and the knowledge are the guiding information in the search model. In this framework, many functions were used to train the optimizer, and then the function evaluation information was stored and transferred to optimize a set of black-box optimization functions.

Designing a hand-built optimization algorithm is an experience-dependent task that is inefficient in most scenarios. For a particular problem, multiple trial-and-error processes are often necessary to develop a practical algorithm. Learning to Optimize (L2O) [40–42] aims to design optimization methods automatically by using training information from many optimization problem instances. The L2O framework comprises two spaces: the algorithm (optimizer) space and the problem space [42]. Based on the knowledge of the relations between training problems and optimizers, this framework could design an efficient algorithm for unseen problem instances. An adaptive reference vector reinforcement learning approach was used to solve the constrained many-objective optimization problems of the industrial copper burdening system [45].

Figure 4 shows the framework of the learning-guided optimization algorithm. Based on the prior knowledge of the solved problem, a search algorithm is designed to sample solutions iteratively in the decision space. The distribution and fitness values of solutions are analyzed in the learning process. Some useful information, such as the best individual in the current swarm for the particle swarm optimization algorithm [46], and the differential between two random individuals for the differential evolution algorithm [47], is used to modify the parameters or structures of the designed search algorithm. The quality of solutions is enhanced through the iterative application of learning and optimization processes. Two orthogonal design engines (exploration and exploitation) are emerging in brain storm optimization algorithm to enhance the learning efficiency [25]. A Q -learning-based brainstorming optimization algorithm was introduced to solve multimodal optimization problems [48].

3 Q -Learning-Guided Memetic Algorithm

Integrating learning strategies into swarm intelligence is a natural way to enhance search performance. Despite the ability of RL to assist robots in planning optimal paths, the associated low learning efficiency limits its widespread application in real-world scenarios. Therefore, improving the convergence speed of algorithms is essential for advancing RL. Integrating SI algorithms with RL for path planning offers significant advantages in accelerating convergence. First, SI algorithms effectively explore the solution space by simulating collective behaviors in nature (such as ant colonies and fish schools), thereby avoiding local optima. Second, RL continuously optimizes the decision-making process through interaction with the environment, enabling agents to dynamically adjust their behavioral strategies based on accumulated rewards. This combination not only enhances the global search capability of the algorithm but also improves its adaptability, enabling agents to adjust to changes in complex and dynamic environments quickly. Furthermore, the collaborative characteristics of SI can expedite the learning process and increase the efficiency and reliability of path planning, enabling agents to find high-quality path solutions more rapidly in practical applications.

In the traditional Q -Learning algorithm, the robot updates the $Q(s, a)$ values using the Bellman equation, incorporating the reward obtained at each iteration. The convergence of the Q -table indicates that the robot's learning process has been completed, enabling it to make decisions based on stable and reliable Q -values. This learning paradigm has led to the widespread use of Q -Learning algorithms in the autonomous path planning of autonomous vehicles. However, as the complexity of the environment and the number of possible movement actions increase, the dimensionality of the Q -table also rises, significantly complicating the convergence of the Q -table. Additionally, inadequate exploration strategies may lead to premature convergence of the Q -table to a local optimum.

To enable the robot to efficiently complete planning tasks in unknown environments, we propose a QLMA algorithm. This algorithm integrates an adaptive genetic algorithm, a simulated annealing algorithm, and a self-learning approach based on Q -Learning. Applying the adaptive genetic algorithm and the

simulated annealing algorithm in the design of the optimized Q -table enhances the performance of the QLMA algorithm compared to the traditional Q -Learning algorithm. Furthermore, the adaptive genetic algorithm can flexibly adapt to complex environments, and its adaptive adjustments help maintain population diversity, thereby mitigating premature convergence [49]. Additionally, the incorporation of simulated annealing in complex environments enhances Q -Learning's ability to escape local optima, thereby increasing the likelihood of identifying a global optimum by allowing for the acceptance of inferior solutions.

3.1 Algorithmic Framework

When a robot employs the QLMA algorithm to plan a path in an unknown environment, it undergoes the learning and planning stage [50]. During the learning phase, the Q -values must be updated twice. First, the Q -table is initially updated by continuously interacting with the environment to learn the strategy, utilizing the Bellman equation. Second, after the robot identifies a potential path, the adaptive genetic algorithm and the simulated annealing algorithm are employed to optimize the search for the global optimal solution, leading to a second update of the Q -table. The instruction list records the state nodes of the shortest path, and when the state nodes of the instruction table cease to change, it indicates that the robot has discovered the optimal path. During the planning stage, the robot follows the state nodes obtained during the learning phase, moving incrementally to reach the target point. Figure 5 illustrates the framework of the QLMA algorithm.

In the QLMA algorithm, two updates to the Q -table were performed. The first update of the Q -table through Q -Learning establishes a foundation for strategy evaluation. The second update, facilitated by a genetic algorithm and simulated annealing, enhances the learning via global search. Since Q -learning primarily leverages existing knowledge, genetic algorithms emphasize exploring new solutions. This combination enables the algorithm to investigate potentially superior strategies based on established effective ones further. The introduction of randomness through the simulated annealing algorithm allows the approach to accept suboptimal solutions during optimization, thus preventing premature convergence to local optima. By integrating Q -Learning with genetic

algorithms and simulated annealing, the algorithm achieves more efficient learning and optimization in robot pathfinding planning applications.

The Q -table is a sparse matrix in which the rows represent the current states, and the columns represent the possible following states that can be reached by taking a specific action. Each cell in the matrix stores the Q -value, representing the expected cumulative reward of taking a particular action to transition from the current state to the next. The 'state' and 'action' in the Q -table are defined as follows:

- A state represents the agent's current position within the network. The indexed points in the road network graph determine the number of states.
- An action refers to the agent's operation to transition from the current state to a potential next state. The number of possible actions depends on how many "next states" can be reached from each current state. A "next state" corresponds to a neighboring node of the current state.

Figure 6 illustrates a road network with nodes and paths in which a reward-punishment mechanism is implemented. Positive rewards represent passable paths, while negative rewards indicate obstacles that block passage. All actions in the Q -table are initialized to zero, as shown in Table 2, in which the rows represent the current nodes (states) and the columns represent the possible actions, *i.e.*, the target nodes that the system can transition to. The Q -value represents the expected reward of taking a particular action from the current node. Then, the Q -learning algorithm is applied to train the system on the paths, resulting in the trained Q -table, as shown in Table 3. Lastly, the instruction was formed in Table 4 based on the Q -values from the trained Q -table. The instruction table records the nearest preceding node for each current node. When the instruction table no longer changes, the algorithm has found the optimal path.

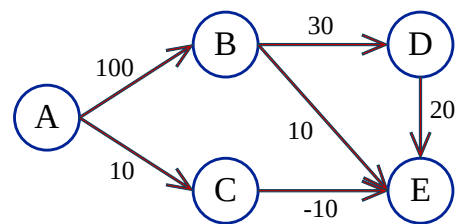


Fig. 6 Road network diagram (including nodes and paths).

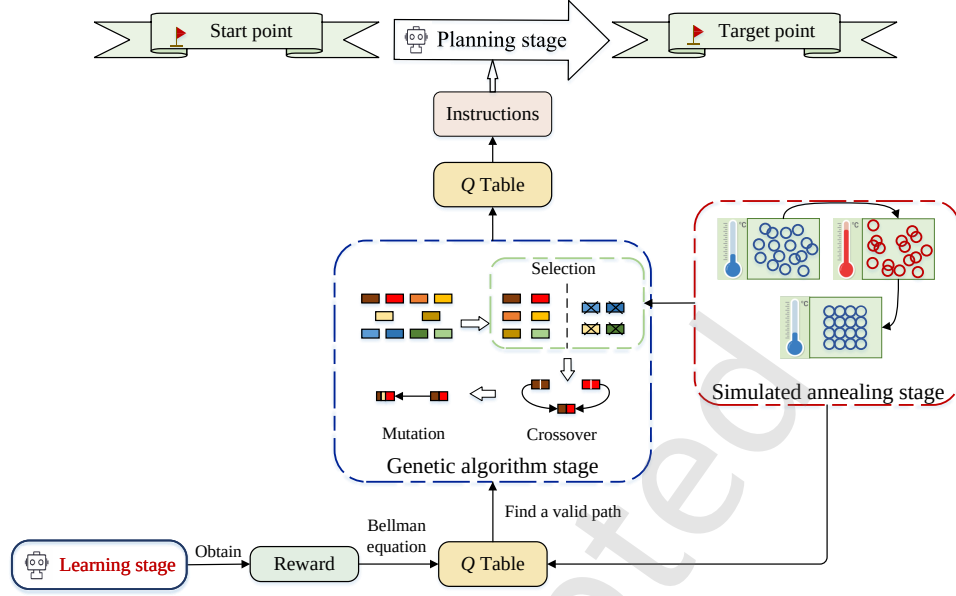


Fig. 5 A framework of QLMA algorithm.

Table 2 Initialization of Q -table.

States \ Actions					
	A	B	C	D	E
A	0	0	0	0	0
B	0	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	0	0	0

3.2 Adaptive Strategy

Adaptive strategy [51] refers to a method that dynamically adjusts the parameters or behaviors of an algorithm based on its current state, environmental changes, or input data during operation to enhance efficiency and effectiveness. To dynamically regulate the rate of population convergence and prevent premature convergence, we will incorporate the adaptive strategy into the crossover and mutation stages of the genetic algorithm. First, during the robot's learning phase, the Q -table is initialized based on the environment's state and action spaces, and the Q -table is updated continuously through reward feedback. The Q -table is defined as a set of chromosomes where each gene represents a state node in the Q -table (i.e., the robot's current position). Each state node corresponds to a Q -value, which reflects the expected reward the robot can obtain after taking a specific action from that position. When making decisions, the robot selects the corresponding Q -value based on the current state node. Then, it evaluates whether the action aligns

with expectations based on the set fitness function, thereby determining whether to execute the action. The chromosome set is defined as follows:

$$chromosome = [n_1, n_2, \dots, n_k] \quad (2)$$

where $[n_1, n_2, \dots, n_k]$ represents all state nodes along the path, from n_1 to n_k .

The objective is modeled as a maximization problem. The fitness function is defined as follows.

$$f(chromosome, T) = \sum_{i=1}^{k-1} Q(n_i, n_{i+1}) - \lambda \sum_{i=1}^{k-1} d(n_i, n_{i+1}) + T \cdot rand() + c \quad (3)$$

where $Q(n_i, n_{i+1})$ reflects the quality of the path from the current node to the next node, where a higher Q value indicates a better path, λ denotes the weight parameter. The $d(n_i, n_{i+1})$ represents the actual cost of the path, typically used to penalize the selection of longer or more distant paths. To increase exploration possibilities, the simulated annealing algorithm is introduced, where $rand()$ is a random number between 0 and 1, and T is the temperature coefficient, which controls the degree of exploration during the search process. As the temperature T gradually decreases, the algorithm increasingly tends to select the optimal solution, while the randomness gradually decreases. Additionally, a constant c is introduced in the formula, where c is defined as a sufficiently large constant relative to the reward value and path cost, ensuring that the function value remains non-negative.

Table 3 Q -table after training.

States \ Actions	A	B	C	D	E
A	0	$Q(A \rightarrow B) = 100$	$Q(A \rightarrow C) = 10$	0	0
B	0	0	0	$Q(B \rightarrow D) = 30$	0
C	0	0	0	0	$Q(B \rightarrow E) = 10$
D	0	0	0	0	$Q(C \rightarrow E) = -10$
E	0	0	0	0	$Q(D \rightarrow E) = 20$

Table 4 Instruction table.

Current node	Nearest preceding node
B	A
C	A
D	B
E	D

The fitness function integrates the advantages of the path (quantified by the quality function Q) and the costs (measured by the distance function d), while introducing randomness to enhance exploration. A higher value of this fitness function indicates a more favorable trade-off between benefits and costs for the path. Next, we employ the temperature-regulated roulette selection method to select the initialized population, as expressed in the following equation.

$$p(\text{chromosome}_i, T) = \frac{\exp\left(\frac{f(\text{chromosome}_i)}{T}\right)}{\sum_{i=1}^N \exp\left(\frac{f(\text{chromosome}_i)}{T}\right)} \quad (4)$$

where $p(\text{chromosome}_i, T)$ is the probability that chromosome_i will be selected at temperature T .

This equation introduces a simulated annealing mechanism into the genetic algorithm's selection process, adjusting the probability of selecting chromosomes at different temperatures. When the temperature T is high, the exponential term $\exp\left(\frac{f(\text{chromosome}_i)}{T}\right)$ approaches 1, leading to a more uniform selection probability. At this point, the algorithm has higher randomness and can select chromosomes with lower fitness, avoiding local optima and enhancing the diversity of the search space. When the temperature T is low, the exponential term approaches infinity, significantly amplifying the selection probability of high-fitness chromosomes. At this stage, the randomness of the simulated annealing algorithm decreases, and the selection process focuses on the optimal solution. This mechanism balances global search and local optimization, ultimately helping to find the global optimum solution.

The above Equations (3) and (4) incorporate the idea

of simulated annealing in the fitness function setting and genetic algorithm selection operation, adopting an 'explore-first then converge' approach to enhance the optimization process efficiency.

In simulated annealing, T is a key parameter that influences the algorithm's search strategy and the quality of the final solution [52]. The update of temperature is critical to the algorithm's effectiveness and performance, employing an exponential cooling strategy.

$$T_{k+1} = \alpha T_k \quad (5)$$

where α is the cooling coefficient (typically between 0.8 and 0.99); a higher α results in slower cooling, which aids in exploring the solution space more comprehensively.

The purpose of introducing simulated annealing into the Q -Learning algorithm for local search is to address the issues of local optimal solutions and the slow convergence speed of Q -Learning by enhancing randomness and exploration capabilities [53, 54]. This approach enables the individual to explore the environment more thoroughly and increases the likelihood of discovering a better strategy.

Subsequently, we enhance the efficiency of algorithm convergence by incorporating an adaptive strategy into the crossover and mutation phases, where the crossover phase employs a single-point crossover with an adaptive crossover rate.

$$p_c = p_{c_{\max}} - (p_{c_{\max}} - p_{c_{\min}}) \cdot \frac{(f_{\max} - f')}{(f_{\max} - f_{\text{avg}})} \quad (6)$$

where p_c is the calculated crossover rate, determining the probability of two individuals exchanging genes during the crossover operation. The maximum crossover rate, $p_{c_{\max}}$, is set as 0.9, indicating the highest crossover probability between individuals. Conversely, the minimum crossover rate, $p_{c_{\min}}$, is set as 0.1, representing the lowest crossover probability. The larger p_c value indicates that more individuals will undergo the crossover operation, thereby enhancing their ability to explore. A smaller p_c value suggests that

fewer individuals will take the crossover operation, and the solutions' structures are kept during the iteration. The f' denotes the bigger fitness of the two parents, while $f_{\max}$ and f_{avg} represent the maximum and average fitness of the population, respectively.

In the mutation phase, we utilize an adaptive mutation rate.

$$p_m = p_{m_{\max}} - (p_{m_{\max}} - p_{m_{\min}}) \cdot \frac{(f_{\max} - f)}{(f_{\max} - f_{\text{avg}})} \quad (7)$$

where p_m is the calculated mutation rate, which determines the probability of individual mutation in the crossover operation. $p_{m_{\max}}$ is the maximum rate of variation, which indicates the highest probability of variation that an individual can accept when the fitness is low; The $p_{m_{\min}}$ is the minimum variation rate, which indicates the lowest probability of variation that an individual can accept when the fitness is high. $f_{\max}$ is the fitness value of the individual with the highest fitness in the current population; f is the fitness value of the current individual, which reflects the performance of the individual in path planning. f_{avg} is the average fitness value for all individuals in the current population.

In the process of mutation, when the individual's fitness f is close to $f_{\max}$, the result of the equation will tend to be $p_{m_{\min}}$, that is, the mutation rate is low, and the retention of the excellent solution is encouraged. When the individual's fitness f is low, the rate of variation tends towards $p_{m_{\max}}$ to increase diversity and explore new solution spaces.

In path planning, this adaptive crossover rate strategy can effectively enhance the efficiency and solution quality of path planning algorithms. An adaptive mutation rate can effectively prevent early convergence to a local optimal solution. The algorithm can adapt to different search strategies at various stages by dynamically adjusting the mutation rate. The higher mutation rate can be used to explore more potential paths in the initial stage. In the later stage, improving population fitness and reducing the mutation rate can help stabilize and optimize the found high-quality pathways; the pseudocode for the adaptive strategy in QLMA is shown as Algorithm 2.

4 Experimental results and analysis

To address the two research questions, we test the performance of the QLMA algorithm through simulation experiments of robot path planning in an unknown environment. In the simulation experiment,

Algorithm 2: The adaptive strategy

Input: Maximum number of iterations (max_iter)

```

1 Initialize  $chromosome = []$ ,  $chromosome' = []$ ,  $iter = 0$ ;
2 Record the effective path found by  $Q$ -Learning to  $chromosome$ ;
3 while  $iter < max\_iter$  do
4   if find the effective path then
5     Perform selection operation; /* Using Eq.(4) */
6     Calculate the adaptive crossover rate  $p_c$ ; /* Using Eq.(6) */
7     Calculate the adaptive mutation rate  $p_m$ ; /* Using Eq.(7) */
      /* perform crossover and mutation to generate new chromosomes */
8      $chromosome'[] \leftarrow$  perform_crossover_mutation( $chromosome[]$ );
      /* evaluate the effective paths of the new chromosomes using a fitness function */
9     obtain_new_effective_path( $chromosome'[]$ );
10     $iter = iter + 1$ ;
```

Output: Optimal $chromosome$

we used the road network diagram to verify the algorithm. Given the practicability of the road network diagram, we analyze the experimental results from four aspects: the optimal path length, the convergence curves, the computational time, and the number of optimal path index points found by each algorithm.

4.1 Benchmark roads and parameter settings

In this study, we utilize road network data from Xi'an and Beijing for our experiments, demonstrating that QLMA outperforms both the Q -Learning algorithm and the SARSA algorithm in the experimental test.

4.1.1 Benchmark roads

The Xi'an road network in 2020 and the Beijing road network in 2008 are used in the experimental study. Three types of test scenarios have 15,000, 20,000, and 40,000 indexing points, respectively. Fig 7 shows the benchmark roads, and the red triangle indicates the start point in each test scenario. All road networks are at a scale of 1:100,000. It is important to note that the intersecting lines on the map do not necessarily indicate that the roads are connected in reality. This discrepancy arises because road network maps are often presented in 2D, and these roads may simply be projections of overpasses on the ground.

In practice, the robot receives feedback from

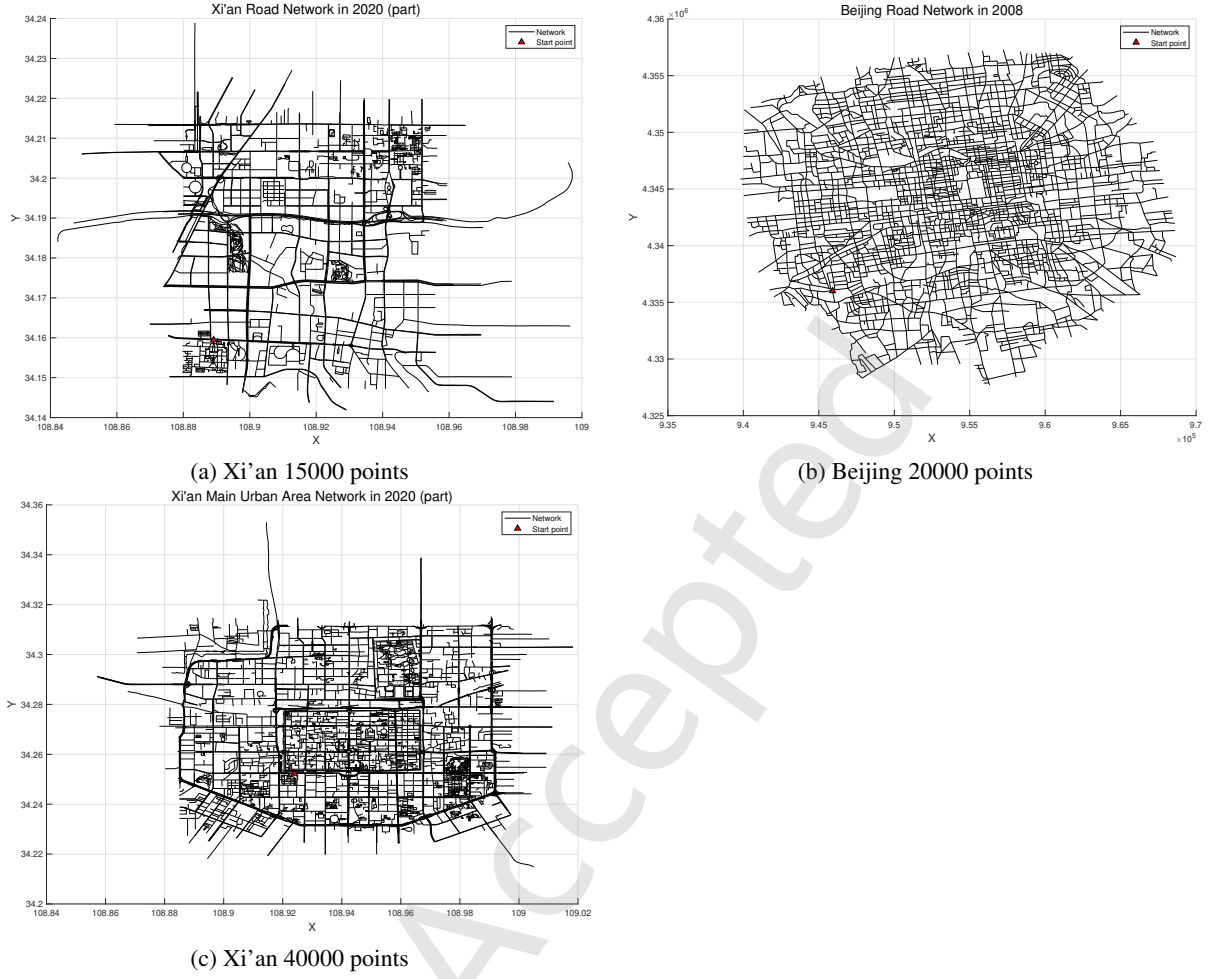


Fig. 7 The benchmark roads used in the experimental study. Three types of test scenarios have 15,000, 20,000, and 40,000 indexing points, respectively.

sensors, allowing it to identify the roads accessible at each intersection. However, the robot lacks sensor feedback in the simulation environment. Therefore, we assume the robot can obtain the corresponding road connectivity information at any intersection in an unknown environment.

The start coordinates are fixed for three types of test scenarios. The setting is as follows.

- (108.889147, 34.159450) → Xi'an 15000 points,
- (945914.844, 4336001.724) → Beijing 20000 points,
- (108.923961, 34.252269) → Xi'an 40000 points.

Ten target points given in Table 5 were selected from different locations on the map for each scenario. Due to the randomness in action selection during the learning phase, the path planned by the robot in each experiment may differ. To ensure the reliability of the experimental

results, we repeated the experiment ten times for the same path planning task.

4.1.2 Parameter settings

The relevant parameter settings are as follows:

- maximum number of iterations = 150;
- maximum step size per iteration $stepmax = 25,000$;
- learning rate $\alpha = 0.1$;
- discount factor $\gamma = 0.9$;
- exploration rate $\epsilon = 0.1$;
- the simulated annealing algorithm
 - cooling parameter $\omega = 0.95$
 - initial temperature $T_0 = 1,000$.

Table 5 Three types of test scenarios and each kind with ten scenarios.

15,000 points			20,000 points			40,000 points		
No.	Label	Target point	No.	Label	Target point	No.	Label	Target point
1	$TS15_1$	(108.942065, 34.2044)	11	$TS20_1$	(962855.068, 4347942.769)	21	$TS40_1$	(108.970349, 34.265618)
2	$TS15_2$	(108.924013, 34.1794)	12	$TS20_2$	(964409.217, 4339783.487)	22	$TS40_2$	(108.920855, 34.281867)
3	$TS15_3$	(108.924003, 34.1788)	13	$TS20_3$	(958788.378, 4353667.218)	23	$TS40_3$	(108.967471, 34.236850)
4	$TS15_4$	(108.900001, 34.1953)	14	$TS20_4$	(950706.803, 4354185.268)	24	$TS40_4$	(108.943336, 34.263396)
5	$TS15_5$	(108.913032, 34.1858)	15	$TS20_5$	(965833.854, 4340923.196)	25	$TS40_5$	(108.911292, 34.287848)
6	$TS15_6$	(108.942035, 34.2093)	16	$TS20_6$	(951121.243, 4351361.897)	26	$TS40_6$	(108.931730, 34.271359)
7	$TS15_7$	(108.928851, 34.1581)	17	$TS20_7$	(953245.247, 4346595.840)	27	$TS40_7$	(108.945416, 34.273192)
8	$TS15_8$	(108.941363, 34.1957)	18	$TS20_8$	(963191.801, 4333748.208)	28	$TS40_8$	(108.942867, 34.281698)
9	$TS15_9$	(108.934287, 34.1963)	19	$TS20_9$	(962207.506, 4354055.756)	29	$TS40_9$	(108.917667, 34.286853)
10	$TS15_{10}$	(108.934645, 34.1856)	20	$TS20_{10}$	(950577.291, 4355350.880)	30	$TS40_{10}$	(108.897342, 34.297308)
Start point		(108.889147, 34.159450)			(945914.844, 4336001.724)			(108.923961, 34.252269)

Table 6 Comparison of optimal path lengths (meter) for SARSA, Q -Learning, and QLMA in path planning.

No.	Test scenario	SARSA	Q -Learning	QLMA	No.	Test scenario	SARSA	Q -Learning	QLMA
1	$TS15_1$	29523.90	15257.39	12535.90	16	$TS20_6$	21064.11	21212.64	21041.73
2	$TS15_2$	15038.40	9914.24	7099.87	17	$TS20_7$	17188.96	17017.19	16884.31
3	$TS15_3$	6702.48	3052.95	3019.81	18	$TS20_8$	27354.16	22557.31	22550.33
4	$TS15_4$	21029.16	10929.54	7666.92	19	$TS20_9$	35168.95	35433.64	34770.31
5	$TS15_5$	14434.84	8430.15	6123.83	20	$TS20_{10}$	25712.00	25279.12	25132.44
6	$TS15_6$	33157.45	17774.94	12865.47	21	$TS40_1$	22375.65	9235.12	7051.64
7	$TS15_7$	12241.86	7412.93	5232.76	22	$TS40_2$	14090.96	6075.62	5104.18
8	$TS15_8$	33634.38	20823.09	18347.81	23	$TS40_3$	17868.10	8938.12	6807.38
9	$TS15_9$	28608.19	16492.93	11581.49	24	$TS40_4$	11148.88	4670.82	3845.23
10	$TS15_{10}$	22178.06	20667.69	12283.45	25	$TS40_5$	20546.78	12046.76	9011.28
11	$TS20_1$	28605.25	28632.96	28265.75	26	$TS40_6$	18411.65	5929.41	5007.90
12	$TS20_2$	28448.25	26835.31	26594.78	27	$TS40_7$	27025.07	8276.07	6561.39
13	$TS20_3$	29943.41	30091.12	29902.41	28	$TS40_8$	29198.88	14684.72	9563.35
14	$TS20_4$	25529.73	25924.6	25510.71	29	$TS40_9$	18386.24	10430.41	7423.56
15	$TS20_5$	27542.68	27891.98	27026.75	30	$TS40_{10}$	25709.57	18241.51	13364.35

4.2 Experimental Results

Table 6 lists the results of optimal path lengths for three algorithms in the 30 test scenarios. For each pair of start and target points, each algorithm will run ten times to ensure the reliability of results, *i.e.*, all results are average values over ten runs. The path planning efficiency of the robot using the QLMA algorithm exceeds that of both the original Q -Learning and SARSA algorithms. Additionally, the QLMA algorithm yields a shorter path length for the robot's search. These results demonstrate that the optimization capability of the QLMA algorithm surpasses that of the other two algorithms, ensuring learning efficiency during path planning.

Figure 8 illustrates the comparison of algorithms in three categories of test scenarios. It could be seen that the QLMA algorithm performs significant better than other algorithms on Xi'an scenarios (Figure 8 (a) and (c)). The results are similar for algorithms on Beijing scenarios (Figure 8 (b)). The complex properties of the road networks may cause these differences.

Research question Q1 aims to verify the efficiency of the QLMA algorithm. Thus, the QLMA algorithm was compared with SARSA, a reinforcement learning algorithm. In all experimental scenarios, the path lengths obtained by the QLMA algorithm were significantly better than those obtained by SARSA.

The QLMA algorithm could be divided into a Q -learning algorithm and a memetic algorithm. The memetic algorithm cannot find a satisfactory solution for a robot in an unknown environment within a given time frame. Thus, to evaluate the function of each part of the QLMA algorithm (the Q2), the QLMA algorithm was compared with the Q -learning algorithm.

There are 30 test cases in the experimental study, and each case will be run ten times to have a fair evaluation. The proposed algorithm has been compared with two reinforcement learning-based algorithms. The memetic algorithm has also been applied to these test cases. The experimental results have validated the performance of the proposed algorithm and the compared algorithms. Many algorithms can be applied to the robot path-

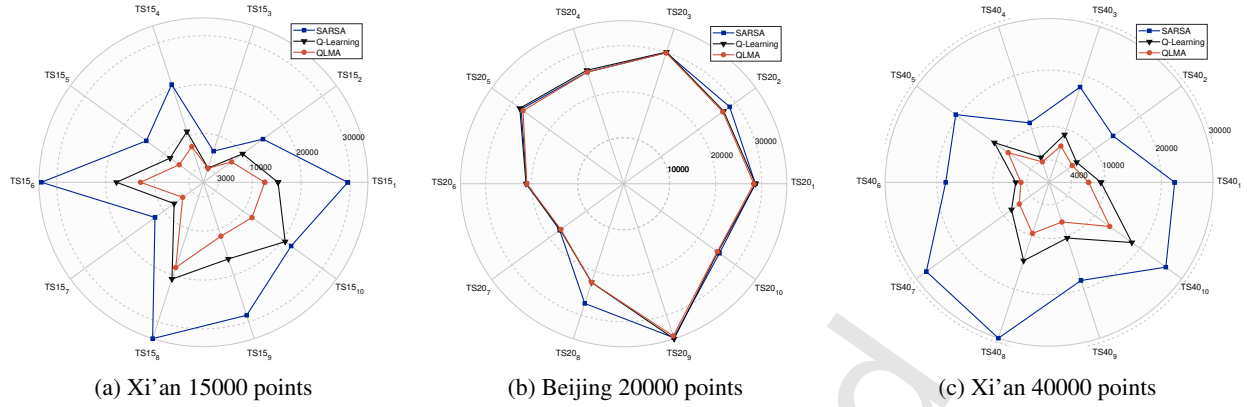


Fig. 8 Comparison of the path length among the three algorithms under the road network information with 15,000, 20,000, and 40,000 indexing points, respectively.

planning problems. Only a select few representative algorithms have been used in the comparison. The results have proved the efficiency of the learning-guided algorithm.

Figure 9 shows the convergence curve and the obtained paths for three examples of algorithms on the $TS15_1$, $TS20_1$, and $TS40_1$ scenarios, respectively. The results were randomly selected from a round of experimental studies. It can be observed that the path length of the QLMA algorithm is significantly shorter than that of the other two algorithms.

4.3 Computational Time Analysis

Table 7 lists the computational time for three algorithms. It is noteworthy that the performance of the SARSA algorithm in unknown environments differs significantly from that of the Q -Learning algorithm, particularly in terms of learning time. Consequently, SARSA is often excluded when comparing the time efficiency of the two algorithms to avoid misunderstandings regarding their performance. Thus, the comparison was conducted on Q -learning and QLMA algorithms.

Figure 10 shows the computational time of the three algorithms. The SARSA algorithm outperforms the other two algorithms significantly. However, the SARSA algorithm performs the worst among the three algorithms. The QLMA algorithm is slightly better than the Q -learning algorithm. The results indicate that the QLMA algorithm achieves good performance without increasing computation time compared to the Q -learning algorithm.

4.4 Scalability Analysis

Table 8 lists the average number of indexing points of three algorithms running ten times on 30 scenarios.

Figure 11 shows a comparison of the number of indexing points on the three algorithms under the road network information. The number of indexing points required to identify the shortest paths found by each algorithm using road network information of varying complexity, thereby demonstrating the effectiveness of the QLMA algorithm.

Both the SARSA and Q -Learning algorithms employ blind search techniques during pathfinding. In contrast, the QLMA algorithm predominantly searches nodes near the identified optimal path. Consequently, during the search phase, the QLMA algorithm requires less time compared to the blind search employed by the SARSA and Q -Learning algorithms. Thus, we can draw a preliminary conclusion that when a robot needs to plan a path in an unknown environment, the QLMA algorithm can enhance the robot's learning efficiency, aligning with practical application needs.

5 Conclusion and Future Work

Path planning in unknown environments is a critical research area in robotics. The QLMA algorithm has been proposed to enhance the path-planning capabilities of a robot. This algorithm integrates Q -learning, genetic algorithms, and simulated annealing to improve performance in complex path-planning problems. The Q -Learning allows learning from the environment, the genetic algorithm provides global search capabilities, and simulated annealing helps escape local optima. By incorporating adaptive operations and temperature regulation, this approach strikes a balance between

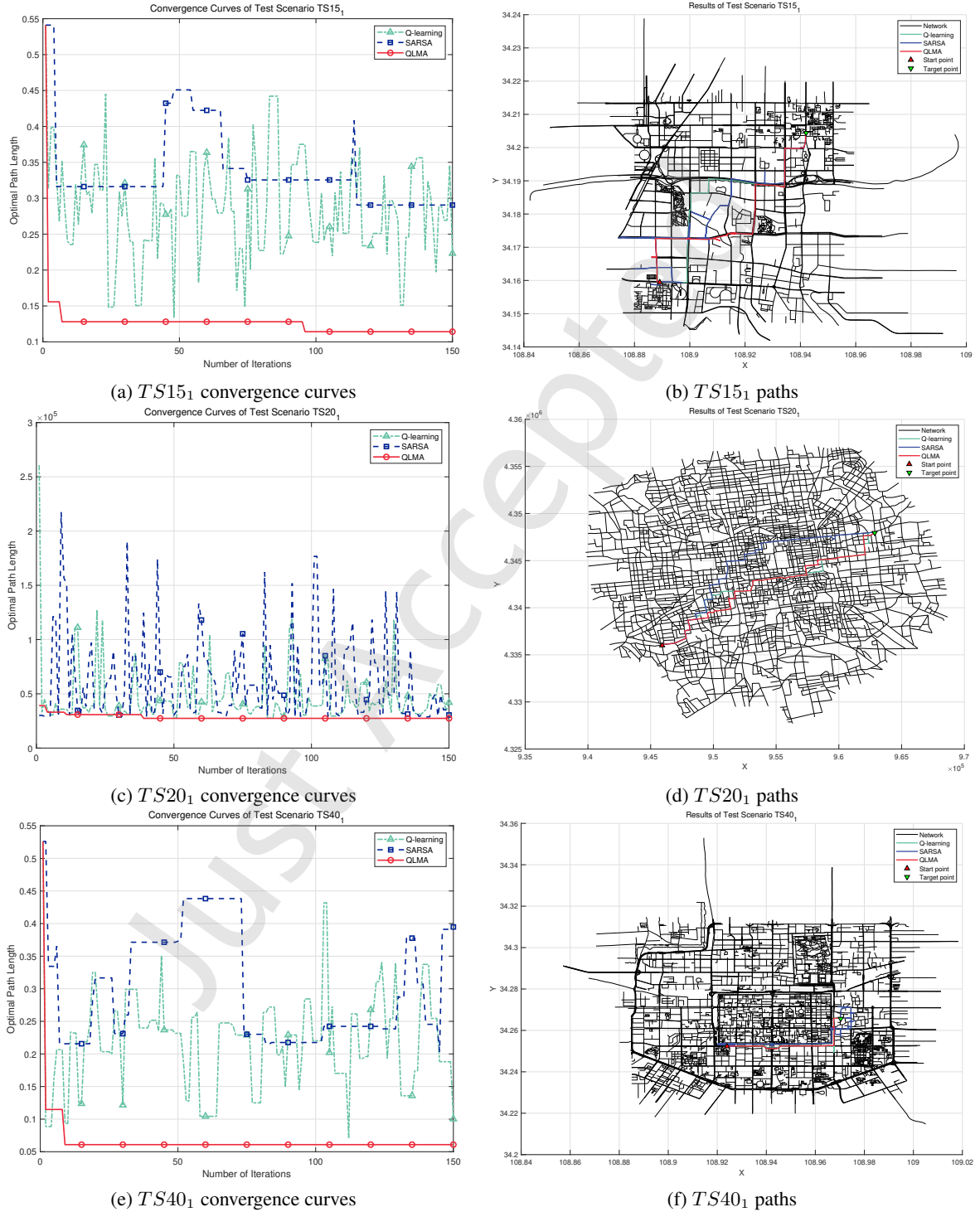
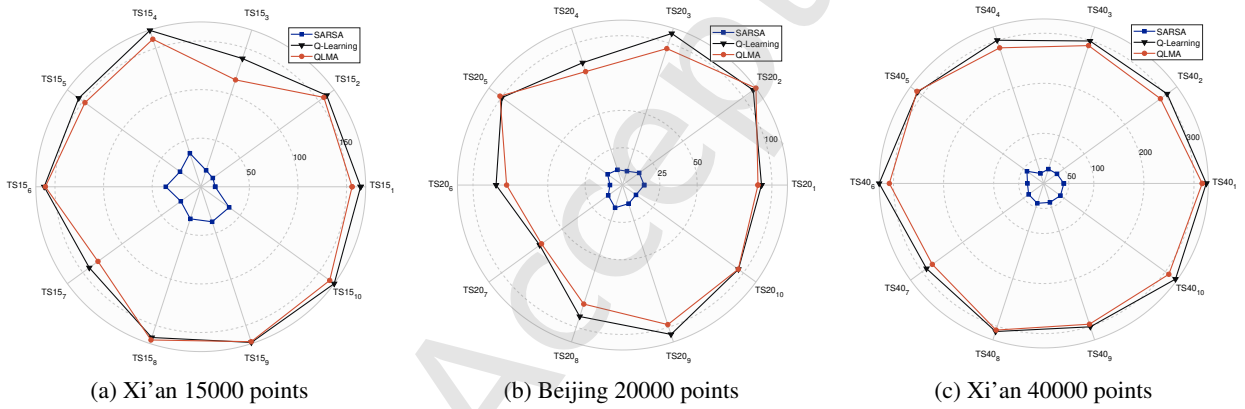


Fig. 9 The convergence curve and obtained paths of three examples of algorithms on *TS151*, *TS201*, and *TS401* scenarios, respectively.

Table 7 Results of computational time (*second*) on three algorithms. The comparison was conducted on *Q*-learning and QLMA algorithms.

No.	Scenario	SARSA	Q-Learning	QLMA	Comparison	No.	Scenario	SARSA	Q-Learning	QLMA	Comparison
1	$TS15_1$	14.8156	164.4707	155.6522	↓ 5.36%	16	$TS20_6$	8.3610	84.5592	77.5019	↓ 8.35%
2	$TS15_2$	15.3073	160.0916	156.4981	↓ 2.24 %	17	$TS20_7$	11.7735	68.4189	66.9206	↓ 2.19%
3	$TS15_3$	17.7870	138.7489	115.6826	↓ 16.62%	18	$TS20_8$	15.9712	92.5099	83.7602	↓ 9.46%
4	$TS15_4$	36.2909	169.2289	159.6857	↓ 5.64%	19	$TS20_9$	13.1255	105.1061	98.2012	↓ 6.57%
5	$TS15_5$	26.4074	155.1822	147.2656	↓ 5.10%	20	$TS20_{10}$	11.1892	95.9409	95.7719	↓ 0.18%
6	$TS15_6$	35.9991	161.0831	159.9212	↓ 0.72%	21	$TS40_1$	40.0868	324.4898	316.4016	↓ 2.49%
7	$TS15_7$	25.3856	141.7512	130.7096	↓ 7.79%	22	$TS40_2$	32.6245	304.5032	288.0148	↓ 5.41%
8	$TS15_8$	34.7195	163.1330	165.8093	↑ 1.64%	23	$TS40_3$	30.1161	299.193	289.5481	↓ 3.22%
9	$TS15_9$	37.8721	168.4742	167.6479	↓ 0.49%	24	$TS40_4$	21.0058	300.7754	284.8103	↓ 5.31%
10	$TS15_{10}$	36.1093	169.6518	163.9582	↓ 3.36%	25	$TS40_5$	41.3973	311.1038	312.8491	↑ 0.56%
11	$TS20_1$	14.5251	93.0402	90.6655	↓ 2.55%	26	$TS40_6$	32.4899	328.5654	308.3089	↓ 6.17%
12	$TS20_2$	13.8355	108.1704	110.4022	↑ 2.06%	27	$TS40_7$	37.2905	289.3069	275.014	↓ 4.94%
13	$TS20_3$	9.7822	106.6894	96.0562	↓ 9.97%	28	$TS40_8$	41.6795	311.0110	307.9878	↓ 0.97%
14	$TS20_4$	10.7586	86.1043	79.8443	↓ 7.27%	29	$TS40_9$	39.8087	301.1423	295.5864	↓ 1.84%
15	$TS20_5$	12.2524	99.6501	101.294	↑ 1.65%	30	$TS40_{10}$	41.2698	325.1217	308.7097	↓ 5.05%

**Fig. 10** Comparison of the computational time on the three algorithms under the road network information with 15,000, 20,000, and 40,000 indexing points, respectively.

exploration and exploitation in the search process. This integrated methodology can obtain superior solutions in various complex path-planning scenarios.

The experimental results show that the QLMA algorithm outperforms the original *Q*-learning and SARSA algorithms in terms of path lengths, and the QLMA algorithm requires less time than the original *Q*-learning algorithm. The QLMA algorithm efficiently solves single-robot path planning problems in three types of test scenarios. However, this algorithm requires more time in large-scale scenarios. Thus, the search efficiency and the capability to solve large-scale problems are essential for enhancing the performance of the proposed algorithm. Multiple robots can cooperate to solve problems that a single robot cannot solve.

Enhancing the search efficiency of the QLMA algorithm and utilizing this algorithm to solve multiple robot path planning problems are our future works. Additionally, other reinforcement learning strategies

could be embedded into more advanced swarm intelligence algorithms. Learning from unknown environments will help swarm intelligence algorithms cooperatively solve multiple robot path planning problems.

Acknowledgment

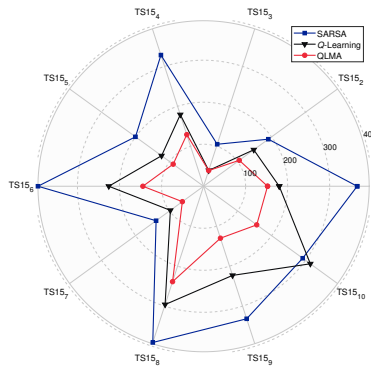
This work is partially supported by the National Natural Science Foundation of China (Grant No. 61806119), the Natural Science Basic Research Plan In Shaanxi Province of China (No. 2024JC-YBMS-516), and Fundamental Research Funds for the Central Universities (No. GK202507001).

References

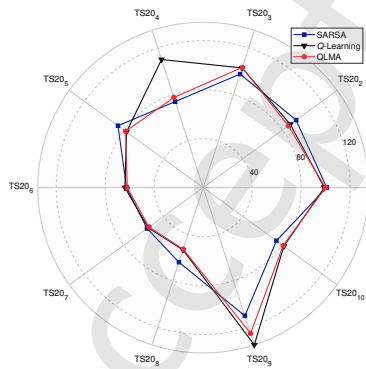
- [1] Y. Gan, B. Zhang, J. Shao, Z. Han, A. Li, X. Dai, Embodied intelligence: Bionic robot controller integrating environment perception, autonomous planning, and motion control, *IEEE Robotics and Automation Letters* 9 (5) (2024) 4559–4566. doi: [10.1109/RAFL.2024.1050000](https://doi.org/10.1109/RAFL.2024.1050000)

Table 8 Comparison of the average number of index points for SARSA, Q-Learning, and QLMA in path planning.

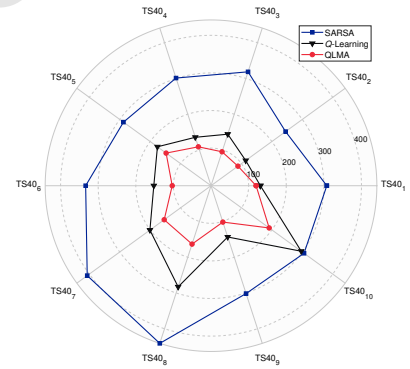
No.	Test scenario	SARSA	Q-Learning	QLMA	No.	Test scenario	SARSA	Q-Learning	QLMA
1	$TS15_1$	365.4	180	152.2	16	$TS20_6$	62.6	63.9	62
2	$TS15_2$	190.5	147.6	105	17	$TS20_7$	56.7	55.6	54.7
3	$TS15_3$	105.1	39.5	39.2	18	$TS20_8$	64.1	53.5	52.9
4	$TS15_4$	328.3	179.2	129.2	19	$TS20_9$	110.1	135.1	125.2
5	$TS15_5$	200.2	123.5	89.4	20	$TS20_{10}$	73.9	81.4	80.7
6	$TS15_6$	394	225.5	144.4	21	$TS40_1$	307.8	131.2	119.8
7	$TS15_7$	139.6	97.6	62.4	22	$TS40_2$	245	114	88.5
8	$TS15_8$	391	296.1	238.8	23	$TS40_3$	318.5	144.1	94.9
9	$TS15_9$	331.7	223.1	130	24	$TS40_4$	301.1	135.4	108.8
10	$TS15_{10}$	291.5	314	156.3	25	$TS40_5$	287.7	176.1	147.6
11	$TS20_1$	100.8	98.9	99.8	26	$TS40_6$	333.3	152.1	103.2
12	$TS20_2$	93.8	87.9	85.9	27	$TS40_7$	406.8	200.8	153.9
13	$TS20_3$	97.5	102.8	103.2	28	$TS40_8$	440.9	282.9	163.3
14	$TS20_4$	73.9	110	77.4	29	$TS40_9$	301.6	143.4	101.7
15	$TS20_5$	86	77.4	78.3	30	$TS40_{10}$	305.7	296.5	191.1



(a) Xi'an 15000 points



(b) Beijing 20000 points



(c) Xi'an 40000 points

Fig. 11 Comparison of the number of indexing points on the three algorithms under the road network information with 15,000, 20,000, and 40,000 indexing points, respectively.

10.1109/LRA.2024.3377559.

- [2] M. Zhao, H. Lu, S. Yang, Y. Guo, F. Guo, A fast robot path planning algorithm based on bidirectional associative learning, *Computers & Industrial Engineering* 155 (2021) 107173. doi:10.1016/j.cie.2021.107173.
- [3] Y. Zhang, D.-w. Gong, J.-h. Zhang, Robot path planning in uncertain environment using multi-objective particle swarm optimization, *Neurocomputing* 103 (2013) 172–185. doi:10.1016/j.neucom.2012.09.019.
- [4] H. Lu, S. Yang, M. Zhao, S. Cheng, Multi-robot indoor environment map building based on multi-stage optimization method, *Complex System Modeling and Simulation* 1 (2) (2021) 145–161. doi:10.23919/CSMS.2021.0011.
- [5] T. Lei, C. Luo, G. E. Jan, Z. Bi, Deep learning-based complete coverage path planning with re-joint and obstacle fusion paradigm, *Frontiers in Robotics*

and AI 9 (2022). doi:10.3389/frobt.2022.843816.

- [6] F. Rubio, F. Valero, C. Llopis-Albert, A review of mobile robots: Concepts, methods, theoretical framework, and applications, *International Journal of Advanced Robotic Systems* 16 (2) (2019) 1–22. doi:10.1177/1729881419839596.
- [7] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, P. Wang, Path planning techniques for mobile robots: Review and prospect, *Expert Systems with Applications* 227 (2023) 120254. doi:10.1016/j.eswa.2023.120254.
- [8] R. S. Sutton, Generalization in reinforcement learning: successful examples using sparse coarse coding, in: *Proceedings of the 8th International Conference on Neural Information Processing Systems, NIPS'95*, MIT Press, Cambridge, MA, USA, 1995, pp. 1038–1044.

- [9] M. Wiering, M. van Otterlo, Reinforcement Learning: State-of-the-Art, Vol. 12 of Adaptation, Learning, and Optimization, Springer International Publishing AG, 2012. doi:10.1007/978-3-642-27645-3.
- [10] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part I, IEEE Robotics & Automation Magazine 13 (2) (2006) 99–110. doi:10.1109/MRA.2006.1638022.
- [11] G. Oriolo, G. Ulivi, M. Vendittelli, Real-time map building and navigation for autonomous robots in unknown environments, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 28 (3) (1998) 316–333. doi:10.1109/3477.678626.
- [12] G. Antonelli, S. Chiaverini, G. Fusco, A fuzzy-logic-based approach for mobile robot path tracking, IEEE Transactions on Fuzzy Systems 15 (2) (2007) 211–221. doi:10.1109/TFUZZ.2006.879998.
- [13] H. Mehrjerdi, M. Saad, J. Ghommam, Hierarchical fuzzy cooperative control and path following for a team of mobile robots, IEEE/ASME Transactions on Mechatronics 16 (5) (2011) 907–917. doi:10.1109/TMECH.2010.2054101.
- [14] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, M. Csorba, A solution to the simultaneous localization and map building (SLAM) problem, IEEE Transactions on Robotics and Automation 17 (3) (2001) 229–241. doi:10.1109/70.938381.
- [15] G. Bresson, Z. Alsayed, L. Yu, S. Glaser, Simultaneous localization and mapping: A survey of current trends in autonomous driving, IEEE Transactions on Intelligent Vehicles 2 (3) (2017) 194–220. doi:10.1109/TIV.2017.2749181.
- [16] J. Xiao, Z. Michalewicz, L. Zhang, K. Trojanowski, Adaptive evolutionary planner/navigator for mobile robots, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 18–28. doi:10.1109/4235.585889.
- [17] T. Lei, T. Sellers, C. Luo, D. W. Carruth, Z. Bi, Graph-based robot optimal path planning with bio-inspired algorithms, Biomimetic Intelligence and Robotics 3 (3) (2023) 100119. doi:10.1016/j.birob.2023.100119.
- [18] J. Li, Y. Hu, S. X. Yang, A novel knowledge-based genetic algorithm for robot path planning in complex environments, IEEE Transactions on Evolutionary Computation 29 (2) (2025) 375–389. doi:10.1109/TEVC.2025.3534026.
- [19] M. Zhao, H. Lu, S. Yang, F. Guo, The experience-memory Q-learning algorithm for robot path planning in unknown environment, IEEE Access 8 (2020) 47824–47844. doi:10.1109/ACCESS.2020.2978077.
- [20] N. Li, B. Xue, L. Ma, M. Zhang, Automatic fuzzy architecture design for defect detection via classifier-assisted multiobjective optimization approach, IEEE Transactions on Evolutionary Computation (2025) 1–1. doi:10.1109/TEVC.2025.3530416.
- [21] T. Fukuda, N. Kubota, An intelligent robotic system based on a fuzzy approach, Proceedings of the IEEE 87 (9) (1999) 1448–1470. doi:10.1109/5.784220.
- [22] S. Wen, Y. Zhao, X. Liu, F. Sun, H. Lu, Z. Wang, Hybrid semi-dense 3D semantic-topological mapping from stereo visual-inertial odometry SLAM with loop closure detection, IEEE Transactions on Vehicular Technology 69 (12) (2020) 16057–16066. doi:10.1109/TVT.2020.3041852.
- [23] R. Bai, H. Guo, W.-Y. Yau, L. Xie, Graph-based SLAM-aware exploration with prior topo-metric information, IEEE Robotics and Automation Letters 9 (9) (2024) 7597–7604. doi:10.1109/LRA.2024.3420817.
- [24] Y. Fan, Q. Zhang, Y. Tang, S. Liu, H. Han, Blitz-SLAM: A semantic SLAM in dynamic environments, Pattern Recognition 121 (2022) 108225. doi:10.1016/j.patcog.2021.108225.
- [25] L. Ma, S. Cheng, Y. Shi, Enhancing learning efficiency of brain storm optimization via orthogonal learning design, IEEE Transactions on Systems, Man, and Cybernetics: Systems 51 (11) (2021) 6723–6742. doi:10.1109/TSMC.2020.2963943.
- [26] T. Lei, C. Luo, S. X. Yang, D. W. Carruth, Z. Bi, Bio-inspired intelligence-based multiagent navigation with safety-aware considerations, IEEE Transactions on Artificial Intelligence 5 (6) (2024) 2946–2961. doi:10.1109/TAI.2023.3334227.

- [27] S. Cheng, M. Zhang, L. Ma, H. Lu, R. Wang, Y. Shi, Brain storm optimization algorithm for solving knowledge spillover problems, *Neural Computing and Applications* 35 (2023) 12247–12260. doi:10.1007/s00521-020-05674-0.
- [28] L. Ma, M. Huang, S. Yang, R. Wang, X. Wang, An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization, *IEEE Transactions on Cybernetics* 52 (7) (2022) 6684–6696. doi:10.1109/TCYB.2020.3041212.
- [29] N. Li, L. Ma, T. Xing, G. Yu, C. Wang, Y. Wen, S. Cheng, S. Gao, Automatic design of machine learning via evolutionary computation: A survey, *Applied Soft Computing* 143 (2023) 110412. doi:10.1016/j.asoc.2023.110412.
- [30] L. Ma, N. Li, G. Yu, X. Geng, S. Cheng, X. Wang, M. Huang, Y. Jin, Pareto-wise ranking classifier for multi-objective evolutionary neural architecture search, *IEEE Transactions on Evolutionary Computation* (2023) 1–12doi:10.1109/TEVC.2023.3314766.
- [31] M. Zhao, H. Lu, S. Cheng, S. Yang, Y. Shi, A multi-robot cooperative exploration algorithm considering working efficiency and working load, *Applied Soft Computing* 128 (2022) 109482. doi:10.1016/j.asoc.2022.109482.
- [32] H. Lu, S. Wang, S. Cheng, Y. Shi, Surrogate-based decision boundary evolutionary sampling method for autonomous systems, *IEEE Transactions on Emerging Topics in Computational Intelligence* (2024) 1–15doi:10.1109/TETCI.2024.3447452.
- [33] H. Lu, Y. Hu, S. Wang, P. Zhou, S. Cheng, DTAS: An adaptive critical scenarios generation method for decision boundary assessment, *IEEE Transactions on Cognitive and Developmental Systems* (2025) 1–14doi:10.1109/TCDS.2025.3527639.
- [34] C. J. Watkins, P. Dayan, Technical note: *Q*-learning, *Machine Learning* 8 (3) (1992) 279–292. doi:10.1023/A:1022676722315.
- [35] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, Model-based reinforcement learning: A survey, *Foundations and Trends in Machine Learning* 16 (1) (2023) 1–118. doi:10.1561/22000000086.
- [36] B. Sallans, G. E. Hinton, Reinforcement learning with factored states and actions, *Journal of Machine Learning Research* 5 (2004) 1063–1088. doi:10.1007/s40747-018-0071-2.
- [37] L. Tai, M. Liu, A robot exploration strategy based on Q-learning network, in: *Proceedings of the 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2016, pp. 57–62. doi:10.1109/RCAR.2016.7784001.
- [38] S. Cheng, L. Ma, H. Lu, X. Lei, Y. Shi, Evolutionary computation for solving search-based data analytics problems, *Artificial Intelligence Review* 54 (2) (2021) 1321–1348. doi:10.1007/s10462-020-09882-x.
- [39] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, N. de Freitas, Learning to learn without gradient descent by gradient descent, in: *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, JMLR.org*, 2017, pp. 748–756.
- [40] Y. Cao, T. Chen, Z. Wang, Y. Shen, Learning to optimize in swarms, in: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, 2019, pp. 1–11.
- [41] T. Chen, X. Chen, W. Chen, Z. Wang, H. Heaton, J. Liu, W. Yin, Learning to optimize: a primer and a benchmark, *Journal of Machine Learning Research* 23 (1) (2022) 1–59.
- [42] K. Tang, X. Yao, Learn to optimize—a brief overview, *National Science Review* 11 (8) (2024) nwae132. doi:10.1093/nsr/nwae132.
- [43] Q. Zhao, T. Liu, B. Yan, Q. Duan, J. Yang, Y. Shi, Automated metaheuristic algorithm design with autoregressive learning, *IEEE Transactions on Evolutionary Computation* (2024) 1–15doi:10.1109/TEVC.2024.3464677.
- [44] Q. Zhao, B. Yan, T. Hu, X. Chen, J. Yang, S. Cheng, Y. Shi, AutoOpt: A general framework for automatically designing metaheuristic optimization algorithms with diverse structures, *IEEE Transactions on Emerging Topics in Computational Intelligence* (2025) 1–14doi:10.1109/TETCI.2025.3561629.

- [45] L. Ma, N. Li, Y. Guo, X. Wang, S. Yang, M. Huang, H. Zhang, Learning to optimize: Reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system, *IEEE Transactions on Cybernetics* 52 (12) (2022) 12698–12711. doi:10.1109/TCYB.2021.3086501.
- [46] J. Kennedy, R. Eberhart, Y. Shi, *Swarm Intelligence*, Morgan Kaufmann Publisher, San Francisco, 2001.
- [47] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization* 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.
- [48] S. Cheng, H. Jin, H. Lu, Y. Shi, A Q-learning based brainstorming optimization algorithm for solving multimodal optimization problems, *IEEE Transactions on Consumer Electronics* 71 (2) (2025) 7317–7327. doi:10.1109/TCE.2024.3502233.
- [49] H. M. Pandey, A. Chaudhary, D. Mehrotra, A comparative review of approaches to prevent premature convergence in GA, *Applied Soft Computing* 24 (2014) 1047–1077. doi:10.1016/j.asoc.2014.08.025.
- [50] A. Maoudj, A. Hentout, Optimal path planning approach based on Q-learning algorithm for mobile robots, *Applied Soft Computing* 97 (2020) 106796. doi:10.1016/j.asoc.2020.106796.
- [51] W. Gong, Á. Fialho, Z. Cai, H. Li, Adaptive strategy selection in differential evolution for numerical optimization: An empirical study, *Information Sciences* 181 (24) (2011) 5364–5386. doi:10.1016/j.ins.2011.07.049.
- [52] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, *Science* 220 (4598) (1983) 671–680. doi:10.1126/science.220.4598.671.
- [53] F. J. Rodriguez, C. Garcia-Martinez, M. Lozano, Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: Taxonomy, comparison, and synergy test, *IEEE Transactions on Evolutionary Computation* 16 (6) (2012) 787–800. doi:10.1109/TEVC.2012.2182773.
- [54] S. Biswas, S. Acharyya, A bi-objective RNN model to reconstruct gene regulatory network: A modified multi-objective simulated annealing approach, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15 (6) (2018) 2053–2059. doi:10.1109/TCBB.2017.2771360.



Shi Cheng Shi Cheng received the Ph.D. degree in electrical engineering and electronics from the University of Liverpool, Liverpool, U.K. in 2013. He is currently an associate professor with the School of Computer Science, Shaanxi Normal University, Xi'an, China. His current research interests include swarm

intelligence, automated planning, multiobjective optimization, and data mining techniques and their applications.



Zhixin Wang is currently pursuing the ME degree at School of Computer Science, Shaanxi Normal University, Xi'an, China. His research interests focus on swarm intelligence and machine learning.



Jian Yang received the Ph.D. in Mechatronics Engineering from the Harbin Institute of Technology. He is currently a Research Associate Professor at the Department of Computer Science and Engineering, Southern University of Science and Technology. His research interests focus on artificial intelligence issues in robotics science, with main research directions including intelligent robots, computational intelligence, and swarm intelligence.



Hui Lu received the Ph.D. degree in navigation, guidance and control from Harbin Engineering University, Harbin, China, in 2004. She is a professor with the School of Electronic and Information Engineering, Beihang University. Her research interests include information and communication system, intelligent optimization, test and fault diagnosis.



Yifei Sun received the Ph.D. degree in electronic science and technology from Xidian University, Xi'an, China, in 2014. She is current an Associate Professor with the School of Physics and Information Technology, Shaanxi Normal University, Xi'an, China. Her research interests include artificial intelligence, evolutionary optimization and learning, and complex networks data mining.



Chaomin Luo received his Ph.D. degree in electrical and computer engineering in the Department of Electrical and Computer Engineering at the University of Waterloo, Canada in 2008; earned his M.Sc. degree in engineering systems and computing at the University of Guelph, Canada, and his B.Eng. degree in electrical engineering from the Southeast University, Nanjing, China. He is currently an Associate Professor, in the Department of Electrical and Computer Engineering, at the Mississippi State University.



Mingming Zhang is currently pursuing the PhD degree at the Department of Mechanical and Mechatronics Engineering, The University of Auckland, Auckland, 1010, New Zealand. Her research interests focus on industry intelligence.



Yuhui Shi Yuhui Shi received the PhD degree in electronic engineering from Southeast University, Nanjing, China, in 1992. He is a Chair Professor in the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. He has nine years' working experience in automotive industry at the USA-based Electronic Data Systems Corporation (EDS) as an Applied Specialist, and over 40 years' experience in algorithm design and implementation primarily using optimization, data mining, computational intelligence, digital signal processing, and pattern recognition techniques. His main research interests include the areas of computational intelligence techniques (including swarm intelligence) and their applications.



Sicheng Hou received the Ph.D. from Waseda University, Japan, in 2024. He is currently an assistant professor in the School of Physics and Information Technology at Shaanxi Normal University, Xi'an, China. His research interests include swarm intelligence, intelligent optimization, and their practical applications.