



Research Article

An Improved Quantum-Behaved Particle Swarm Optimization Algorithm Combined with Reinforcement Learning for AUV Path Planning

HanBin Zhang ^{1,2} and XianPeng Shi ¹

¹National Deep Sea Center, Qingdao 266237, China

²School of Automation and Electronic Engineering, Qingdao University of Science and Technology, Qingdao 266100, China

Correspondence should be addressed to XianPeng Shi; xpsh@ndsc.org.cn

Received 23 November 2022; Revised 18 December 2022; Accepted 6 January 2023; Published 5 April 2023

Academic Editor: Changsheng Li

Copyright © 2023 HanBin Zhang and XianPeng Shi. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In order to solve the problem of fast path planning and effective obstacle avoidance for autonomous underwater vehicles (AUVs) in two-dimensional underwater environment, a path planning algorithm based on deep Q-network and Quantum particle swarm optimization (DQN-QPSO) was proposed. Five actions are defined first: normal, exploration, particle explode, random mutation, and fine-tuning operation. After that, the five actions are selected by DQN decision thinking, and the position information of particles is dynamically updated in each iteration according to the selected actions. Finally, considering the complexity of underwater environment, the fitness function is designed, and the route length, deflection angle, and the influence of ocean current are considered comprehensively, so that the algorithm can find the solution path with the shortest energy consumption in underwater environment. Experimental results show that DQN-QPSO algorithm is an effective algorithm, and its performance is better than traditional methods.

1. Introduction

Autonomous underwater vehicles have been developed since the 1950s and have evolved to become one of the most critical tools in current marine exploration research. Compared with other underwater robots, AUVs can combine advanced intelligent algorithms to perform autonomous and uncrewed operations in all aspects, which are ideal for search, identification, and detection work on the seafloor and are an economical and safe means of offshore investigation [1–3]. Among them, underwater path planning technology is an essential technology for AUVs. It mainly addresses the problem of how to plan a practical path and avoid obstacles autonomously in a complex underwater environment. Compared to other robots, AUVs in this field must adapt to complicated underwater environments and consider the effects of unpredictable environmental factors such as currents. Furthermore, there are numerous hurdles

in online obstacle avoidance, 3D path planning, and robustness of algorithms [4–6].

Currently, much research has been conducted on underwater path planning. In general, the robot path planning problem can be viewed as finding collision-free optimal or suboptimal trajectory based on certain performance indicators in a working environment with obstacles. The path planning algorithms can be categorized as global path planning for known static obstacles and local path planning for unknown dynamic barriers. In global path planning, barriers are static, their locations and shapes can be measured in advance. Based on it, a global map of the environment can be produced, then we can use algorithms to determine the optimal route. Examples include the algorithm in [7], the Dijkstra algorithm [8], the particle swarm algorithm [9], and the ant colony algorithm [10]. However, in a complex underwater environment, it is challenging to construct all the information of obstacles through maps, so

local path planning algorithms must be designed to avoid these unknown dynamic obstacles. These algorithms include the artificial potential field method [11] and fuzzy logic algorithm [12]. In recent years, based on the development of deep neural networks, some self-learning algorithms, such as neural networks [13] and reinforcement learning algorithms [14], have been introduced to local path planning for AUVs. These self-learning algorithms do not require prior knowledge of the environment and can handle real-time dynamic path planning problems.

The particle swarm algorithm (PSO) is a well-known evolutionary algorithm due to its robustness and easy parameter tuning advantages. Considering the complex and variable characteristics of the underwater environment, the PSO algorithm and its variants have been widely used in AUV path planning obstacle avoidance experiments [15–19]. Wu et al. [17] proposed a method to optimize parameters which affect performance of the PSO algorithm by using Rauch-Tung-Striebel (RTS) smoother. It can eliminate the irregular error of the PSO updated position and to smooth the produced path. Zhang et al. [20] presented an improved path planning based on the hybrid multiobjective barebones particle swarm optimization with differential evolution. Cheng et al. [21] proposed a combined particle swarm and wolf swarm algorithm (PSO-GWO). It combines the global search ability of the wolf swarm algorithm and the robustness of the particle swarm algorithm so that it can adapt to the underwater variable environment. In addition, due to its powerful global search ability, quantum particle swarm optimization (QPSO) has also been applied to the path search algorithm of AUV. Experiments indicate that the QPSO algorithm can find the best path solution in complex terrain more effectively [22, 23]. However, the algorithm still has the problems of insufficient global convergence and lack of operational accuracy.

Based on the questions above, this article proposes an improved quantum particle swarm algorithm based on deep Q-network (DQN-QPSO). It combines the learning mechanism of DQN with QPSO to make decisions from five behaviors using neural networks by inputting particles' information and updating the particles' position information according to each step of the decision. The strategy enables the particle to choose the appropriate action in various circumstances, significantly enhancing the algorithm's global search capability. And the accuracy of the algorithm can be improved by fine-tuning operations. In addition, this article designs a fitness function that is better suited for the underwater environment. Considering the influence of route path length, deflection angle, and currents, the algorithm can better adapt to the underwater environment and locate the solution path with the lowest energy consumption. The simulation results demonstrate that the algorithm can better solve the problem of premature convergence and it outperforms the standard algorithm in terms of global search capability and search precision.

The rest of this article starts with describing the modeling method for the underwater environment and obstacles in Section 2. Section 3 introduces the QPSO, DQN, and the algorithms used for path smoothing. In Section 4, The

DQN-QPSO algorithm designed in this article is presented, and the simulation and experimental studies are carried out in Section 5. Finally, Section 6 gives this article's conclusions and future research directions.

2. Map Representation and Construction

2.1. Research Model of the Seamount. In the process of AUV underwater navigation, where there will be various obstacles, different terrain corresponds to other routes. Hence, accurately modeling the environment is essential for planning the route. After studying the various obstacles on the seafloor, this article discovered that the most common underwater environment is the seamount, whose mathematical model can be expressed as follows:

$$Z(x, y) = \sum_{i=1}^n h_i \exp \left[-\left(\frac{x - x_i}{x_{si}} \right)^2 - \left(\frac{y - y_i}{y_{si}} \right)^2 \right], \quad (1)$$

where x_i and y_i are the center coordinates of the i th seamount, h_i is the height of the seamount, and x_{si} and y_{si} are the decay control slope of the i th seamount along the x -axis and y -axis directions. And n denotes the total number of seamounts environmental model. In this article, there are 20 different seamounts randomly distributed. At the same time, x_i and y_i are taken as random numbers, and the height of seamounts follows the normal distribution. The simulation results are shown in Figure 1.

2.2. Three-Dimensional Current Model Construction. AUVs are impacted by currents while navigating underwater, causing them to increase excessive energy consumption and even stray from their intended trajectory; therefore, it is essential to account for the current element while designing the course. Therefore, this work employs the proposed method derived from an examination of the motion of actual currents [3], the motion state of which may be reduced to a viscous Lamb vortex, as shown in the following equation:

$$\begin{cases} V_x(r) = -\lambda \cdot \frac{y - y_0}{2\pi(r - r_0)} \cdot \left(1 - e^{-(r-r_0/\zeta)^2} \right), \\ V_y(r) = \lambda \cdot \frac{x - x_0}{2\pi(r - r_0)} \cdot \left(1 - e^{-(r-r_0/\zeta)^2} \right), \\ V_z(r) = \frac{\lambda}{\pi\zeta^2} \cdot e^{-(r-r_0/\zeta)^2}, \end{cases} \quad (2)$$

where $V_x(r)$, $V_y(r)$, and $V_z(r)$ are the velocity components of the current along the x -axis, y -axis, and z -axis directions, the parameters ζ and r_0 are the vortex radius and coordinates of the vortex center position, respectively. And λ denotes the vortex intensity; if it is positive, the vortex is counterclockwise; otherwise, the vortex is clockwise.

MATLAB is used to simulate three-dimensional space, and ten different vortices are randomly generated, of which five are clockwise vortices and five are counterclockwise

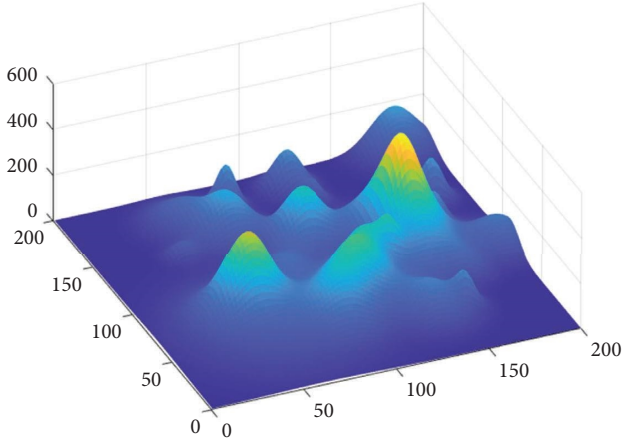


FIGURE 1: Seamount model diagram.

vortices. The discrete ocean current model shown in the figure can be obtained by uniformly setting discrete interval points and performing superposition calculations on the values of vortices at each point, as shown in Figure 2. In addition, this study only analyzes the condition in which the currents vary at different locations, and their speed is believed to be constant throughout the AUV's whole navigation period.

3. Related Research

3.1. QPSO Algorithm. The quantum particle swarm algorithm is an improved swarm intelligence algorithm, first proposed by Fang et al. [24] by deriving the principles of quantum physics and combining it with the PSO algorithm,

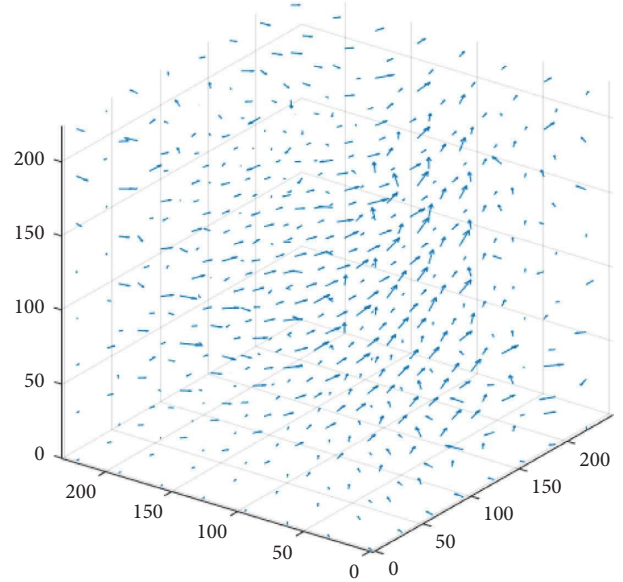


FIGURE 2: Three-dimensional current map.

which substantially improves its global search capability. In the quantum system, the position of particle appearance is no longer represented by a fixed position and velocity vector. Instead, the probability of particle appearance at a given point in space is obtained by solving the Schrodinger function, causing the position of particle appearance to deviate significantly from the local best. Combining this information with the standard PSO, we obtain the following equations:

$$x_{i,d}(t+1) = \begin{cases} q_{i,d}(t) - \beta \cdot |m_{i,d}(t) - X_{i,d}(t)| \cdot \ln\left(\frac{1}{u}\right), & u > 0.5, \\ q_{i,d}(t) + \beta \cdot |m_{i,d}(t) - X_{i,d}(t)| \cdot \ln\left(\frac{1}{u}\right), & u \leq 0.5, \end{cases} \quad (3)$$

$$q_{i,d}(t) = \phi \cdot p_{i,d}(t) + (1 - \phi) \cdot g_d(t), \quad (4)$$

$$m_{i,d}(t) = \frac{1}{n} \cdot \sum_{i=1}^n p_{i,d}(t), \quad (5)$$

where t is the number of iterations ($1 \leq t \leq T$), d is the dimension of particles ($1 \leq d \leq D$), N is the number of particles, $x_{i,d}(t)$ denotes the position vector of the particles, $p_{i,d}(t)$ and $g_d(t)$ are the personal best position and global best position, respectively, $m_{i,d}(t)$ is the mean best position, defined as the average of the individual best positions of all particles in the particle population, and ϕ is a random variable whose value is between 0 and 1. Furthermore, the parameter β is the contraction-expansion (CE) coefficient, which can adjust the convergence behavior of QPSO.

3.2. DQN Algorithm. As a research hotspot in artificial intelligence, machine learning, and automatic control, reinforcement learning is regarded as one of the fundamental technologies for intelligent system design. DQN algorithm is based on the combination of Q-learning and neural network algorithms, which combine the characteristics of deep learning algorithms that are easy to solve high-dimensional continuous problems and can fit the learning results of reinforcement learning algorithms. In the standard DQN model, the agent observes the current state of the system environment. It decides to act on the following state through

the neural network, after which the environment will provide rewards or penalties for this behavior. After completing a series of behaviors, the neural network is trained through the data so that subsequent decisions can achieve greater rewards, as shown in Figure 3.

In the DQN decision process, the neural network is used as the carrier of the value function to obtain the approximate Q value, as shown in the following equation:

$$Q(s_t, a_t) = f(s_t, a_t, \omega), \quad (6)$$

where s_t and a_t represent the state and the action taken at the time of t , respectively, and ω is the weight value of each node in the neural network. As mentioned above, in the DQN, there are two networks, the estimation network and the target network. These two networks share an identical structural design but have different network parameters. The output of the estimation network is $Q(s_t, a_t, \omega)$, which is used to estimate the value function of the current state action, while the output of the target network is donated as $Q(s_t, a_t, \omega')$. By using the untrained target network, we ensure that the target Q value remains stable for at least a short period of time and the weights from the estimation network are replicated to the target network after a preset number of steps.

In addition, in the neural network, the input is the current state S and the output is a sequence of Q values corresponding to a series of actions. This makes it easier to select actions and update the Q values using the Q-learning method. The update formula and the loss function of the Q-learning algorithm can be formulated as follows:

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a') - q(s_t, a_t)], \quad (7)$$

$$L(\omega) = E \left[\left(r_{t+1} + \gamma \max_{a'} q(s_{t+1}, a', \omega') - q(s_t, a_t, \omega) \right)^2 \right]. \quad (8)$$

The parameters of the Q sequence obtained from the Q-network are corrected by applying the above equation, and the neural network is trained by using equation (7). The parameter α is the learning rate and γ is the discount factor. In addition, to solve the problem of interrelated training data in DQN, the concept of experience replay is introduced, and before training the neural network, the agent interacts with the environment a certain number of times and the data is stored in the experience pool in the format of (s_t, a_t, r_t, s_{t+1}) , and during the training process, a certain number of samples are taken out from the experience pool each time and updated using the gradient descent algorithm.

3.3. Design of Fitness Function. In the QPSO optimization problem, it is crucial to develop a suitable fitness function to measure the merit of the particles. Designing the corresponding fitness function for a specific situation enhances the algorithm's precision and accelerates its processes. For example, for the Qianlong III AUV with a fixed battery capacity, the energy consumption of the path is an essential

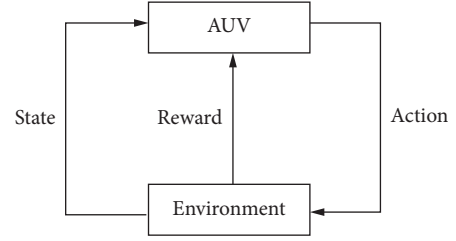


FIGURE 3: Application of reinforcement learning to AUV.

factor affecting the quality of the path. Therefore, the energy consumption of the AUV is used in this article to design the algorithm's fitness function. In general, the energy consumption of AUV is primarily influenced by three factors: the route path length, the route deflection angle, and the influence of current.

Corresponding evaluation functions are designed for the three factors, respectively, and the corresponding weighting factors are set to weigh the importance of the three in practical application, which is defined as follows:

$$F_c(X_i(t)) = \sum_{k=1}^3 f_k F_k(X_i(t)), \quad (9)$$

where f_k is the weight factor, which typically takes a value between 0 and 1; first, the path length function is $F_1(X_i(t))$, defined as the sum of the distances on each path segment connected by two consecutive path points, as shown in equation (11), whose value is positive and is the primary evaluation function in the path planning problem. Therefore, the weight of this evaluation function is also assigned a more significant value.

$$F_1(X_i(t)) = \sum_{k=1}^{m-1} \|x_{i,k}(t) - x_{i,k+1}(t)\|. \quad (10)$$

Set the deflection angle function as $F_2(X_i(t))$, first use the cosine theorem of three consecutive path points to determine the inner angle of the point $\text{ang}_{i,k}$, take the complementary angle to it to get the deflection angle of the route, and compare it with the maximum deflection angle of AUV φ_{\max} ; if it exceeds, it must add additional energy consumption for the large angle turn, where μ is the angle energy gain coefficient, the formula is as follows:

$$F_2(X_i(t)) = \sum_{k=2}^{m-1} F_2'(180^\circ - \text{ang}_{i,k}), \quad (11)$$

$$F_2'(\varphi) = \begin{cases} 0, & \varphi \leq \varphi_{\max} \\ \mu \cdot \frac{\varphi}{\varphi_{\max \max}}, & \varphi > \varphi_{\max} \end{cases} \quad (12)$$

In the calculation of $F_3(X_i(t))$, the whole path segment should be divided equally into p segments and calculate the coordinates of the end points. For each point, we used equation (2) to solve the current information of the point in the x -axis $V_x(r)$, y -axis $V_y(r)$, and z -axis $V_z(r)$. Then, synthesize them into $V(r)$ to get the current vector at the

driving point, and then decompose them into the projection in the direction of navigation and vertical navigation $V_1(r)$ and $V_2(r)$, and the parameter θ is the angle of the projection, as shown in Figure 4.

Through analysis, the decomposed vector $V_1(r)$ can operate directly on the AUV causing an increase or decrease in energy consumption, while $V_2(r)$ will produce a course shift, resulting in a direct energy loss. Therefore, the computation for $F_3(X_i(t))$ may be performed using equation (14). The value is positive when θ is greater than 90 degrees and negative when θ is less than 90 degrees.

$$F_3(X_i(t)) = - \sum_{k=1}^p (V(r) \cdot \cos \theta - V(r) \cdot \sin \theta). \quad (13)$$

In addition, this article designed the fitness function with three objectives of path length, path deflection angle, and ocean current influence. However, the results of these three objectives were of different orders of magnitude in the experiment, so it was necessary to carry out normalization operations on them, respectively, before summing and obtaining the final fitness value by equation (14).

3.4. Cubic B-Spline. The algorithm proposed in this article can generate a series of path points that, in turn, generates a folded segment path, which cannot meet the actual navigation requirements of AUV. Therefore, the method of three times b-sample is employed to smooth the path further. Its path points are generated according to the curve function as the control points of the b-sample curve. $P(t)$ generates a smooth path with continuous curvature as a discrete series of path points, defined as equation (15), where $B_{i,k}(t)$ is defined by Cox and DeBoor, defined as equation (16).

$$P(u) = \sum_{i=0}^3 P_i B_{i,k}(u), \quad (14)$$

$$\begin{aligned} B_{0,3} &= \frac{1}{6}(1-u)^3 B_{1,3} = \frac{1}{6}(3u^3 - 6u^2 + 4) B_{2,3} \\ &= \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1) B_{3,3} = \frac{1}{6}u^3. \end{aligned} \quad (15)$$

4. DQN-QPSO Algorithm

4.1. Structure of the Algorithm. The general structure of the DQN-QPSO algorithm is depicted in Figure 5. The

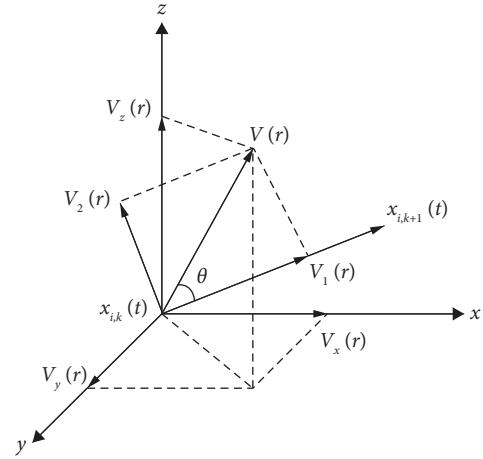


FIGURE 4: Vector projection of sea currents.

algorithm treats the particle population as the agent. And the environment is the shortest distance exploration of the AUV from the starting point to the endpoint. The state represents the current position information of the particle population, denoted by $m_{i,d}(t)$, $g_{i,d}(t)$, and diversity (S). Actions are defined as the current operational states of the particles, i.e., normal, exploration, particle explosion, random mutation, or fine-tuning. As shown in Figure 5, the normal action is used to make the algorithm converge, the exploration, particle explosion, and random variation are used to improve the global search capability of the algorithm, and the fine-tuning operation is used to improve the algorithm's final result's precision. Algorithm 1 illustrates the proposed DQN-QPSO search procedure.

In this algorithm described in this article, a multilayer neural network is built, whose input layer is 7-dimensional data containing particle population position information and the output is 5-dimensional action data. Moreover, this network has two hidden layers in the middle, with six neurons in each layer, to realize the mapping from state to action. The particle population adjusts its location to get a particular reward and continues this process until the maximum number of iteration steps is reached. The reward values of the algorithm at each generation are shown as follows:

$$r_t = \begin{cases} \cdot (F_c(g_d(t)) - F_c(g_d(t-1))), & \delta F_c(g_d(t)) < F_c(g_d(t-1)), \\ -1, & F_c(g_d(t)) \geq F_c(g_d(t-1)), \end{cases} \quad (16)$$

where δ is the fine-tuning damping factor in balancing the global search capability of the algorithm and in preventing falling into a local optimum prematurely and having difficulty

choosing other behaviors to leap out of it owing to overreliance on the rewards supplied by the fine-tuning operation in the early stage of the search (as further clarified in Section 4.5).

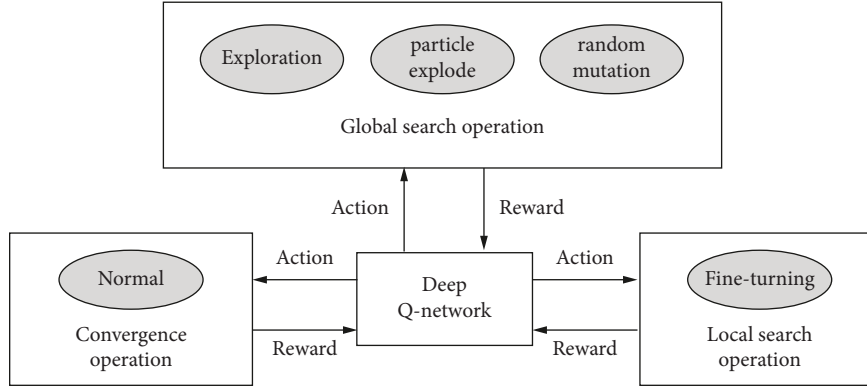


FIGURE 5: Algorithm structure diagram.

```

Initialize the particles' positions, global best, and personal best with their fitness value
Initialize the weight vector of deep Q-network
Compute the mean best and diversity of the particles using equations (5) and (18)
While  $i = 1$  to Maxiter
  Do for each particles
    Choose the best action  $a_t$ 
    Switch action
    Case normal
      Update the particles using equations (3), (4), and (19)
    Case exploration
      Update the particles using equations (3) and (4)
    Case particle explode
      Initialize the mbest
    Case random mutation
      Update the particles using equations (3), (4), and (20)
    Case Fine-tuning operation
      While  $j = 1$  to 3
        While  $k = 1$  to  $K$ 
          Update the particles using equations (21), (22), and (23)
          Compute the fitness value of personal best
        End
      End
       $i = i + K - 1$ 
    Set an immediate reward using equation (17)
  End
  Update the global best and personal best with their fitness value
  Compute the mean best and diversity of the particles using equations (5) and (18)
  Store transition  $(s_t, a_t, r_t, s_{t+1})$  in  $D$ 
  Sample random mini-batch of transitions  $(s_t, a_t, r_t, s_{t+1})$  from  $D$ 
  Calculate target value function  $y_j = \begin{cases} r_j, & \text{if episode terminates} \\ r_j + \gamma \max q(s_{j+1}, a'_t; \omega'), & \text{otherwise} \end{cases}$ 
  Perform a gradient descent step on  $(y_j - q(s_{j+1}, a'_t; \omega'))^2$ 
   $i = i + 1$ 
End

```

ALGORITHM 1: DQN-QPSO.

During the operation of the algorithm, a seven-dimensional information representing the current state information of the particle population is passed into the neural network in each iteration. And a best action is derived through the operation of the neural network to update the

information of the particle population and calculate the reward value r_t . Then, pass the data of (s_t, a_t, r_t, s_{t+1}) into experience pool to wait for training. When the accumulated data reach a certain amount, we start training the neural network to make it have better decision-making ability.

4.2. Diversity Parameters. In 2006, Sun et al. introduced the idea of diversity control into the QPSO model [25]. The search in the algorithm is guided by a diversity measure diversity (S), and the parameter β in the algorithm is altered by setting a threshold value so that the algorithm can achieve convergence and divergence at the appropriate time. In this article, the same parameter was used to describe the current diversity state of the particle population. It is added to the input side of the neural network as position information, enabling it to make better judgments. The following equation describes the diversity parameter:

$$\text{diversity}(S) = \frac{1}{N \cdot |A|} \cdot \sum_{i=1}^N \sqrt{\sum_{d=1}^D (x_{i,d} - \bar{x}_d)^2}, \quad (17)$$

where S donates the particle population and $|A|$ is the length of the diagonal in the search space.

4.3. Normal and Exploration States. Normal and exploration are both tuned to the parameter β in QPSO, which corresponds to the QPSO running state of convergence and the divergent state which is more likely to jump out of the local optimum, respectively. According to the research [22], this parameter can effectively control the algorithm's convergence rate. The algorithm diverges when $\beta > 1.778$ and converges when $\beta < 1.778$. The normal and exploration state operations are designed according to this characteristic. When the algorithm is in the normal action, the value of β is taken as shown in the following equation:

$$\beta = \beta_{\max} - \frac{t}{T} (\beta_{\max} - \beta_{\min}), \quad (18)$$

where β_{\max} and β_{\min} are the maximum and minimum values of parameter β . In the normal action, the value of β is set to be less than one; hence, they are typically set to 1.0 and 0.5, and in the exploration action, the value of β is set to 1.8.

4.4. Particle Explode and Random Mutation. The above exploration states can effectively broaden the search range of the algorithm and reduce the probability of the algorithm falling into a local optimum. Based on this, two more approaches are added as actions to enhance the global search capability of the algorithm, which combined with the exploration operation can result in the algorithm selecting a better solution to the present challenge when it reaches a local optimum.

Particles explode and random mutation operations have been used in some QPSO-based variants, where the particle explode operation is to initialize the mean best position $m_{i,d}(t)$. The reason is as follows: when the particles fall into a local optimum, the distance between the particle and $m_{i,d}(t)$ is too small, as can be seen from equation (3), resulting in a shorter particle step size and making it more difficult to escape after the particle falls into a local optimum. Therefore, initializing $m_{i,d}(t)$ will increase the distance between the particle and $m_{i,d}(t)$, thus causing the effect of particle explosion.

Furthermore, the random mutation operation is a mutation operation on the best location of a randomly selected particle by adding a random value associated with the search boundary to the best position and randomly modifying the location of the particle as follows:

$$p_{i,d}(t) = p_{i,d}(t) + \lambda \cdot x_{\max,d} \cdot \phi, \quad (19)$$

where ϕ is the random number of the standard normal distribution $N(0, 1)$, λ is the weight factor, and $x_{\max,d}$ is the maximum search boundary of particles.

4.5. Fine-Tuning Operation. The fine-tuning operation is used to fine-tune the particle $p_i(t)$ on each dimension, which is three-dimensional in the AUV path planning problem described in this article, and on this basis, combined with the ISPO model proposed in the literature [26], the individual best values of the particles are adjusted more precisely by independently adjusting the position information in the three dimensions, followed by optimizing the global optimum. In the ISPO model, the velocity and position in each dimension are calculated as follows:

$$V_{i,d} = a \cdot \frac{1}{k} r + b \cdot L_{i,d}, \quad (20)$$

$$p_{i,d}(k) = \begin{cases} p_{i,d}(k) + V_{i,d}, & p_{i,d}(k) > p_{i,d}(k-1), \\ p_{i,d}(k), & p_{i,d}(k) \leq p_{i,d}(k-1). \end{cases} \quad (21)$$

The parameters a and b are the acceleration factors, r is a uniformly distributed random value between $[0.5, 0.5]$, and k is the current number of iterations. The algorithm will iterate K a total of three times, and the values of $V_{i,d}$ and $L_{i,d}$ will be updated during each iteration. The variable $L_{i,d}$ indicates the speed-controlled learning value. If the fitness function increases during the iteration, the value of $L_{i,d}$ is doubled to approach the peak of the fitness function more quickly, the value of $L_{i,d}$ is halved, and the algorithm's precision is enhanced. The equation is as follows:

$$L_{i,d} = \begin{cases} 2V_{i,d}, & p_{i,d}(k) > p_{i,d}(k-1), \\ 0.5L_{i,d}, & p_{i,d}(k) \leq p_{i,d}(k-1). \end{cases} \quad (22)$$

The fine-tuning operation is very useful in improving the accuracy of the operation. However, as mentioned earlier, it is given a low reward in the first and middle of the algorithm, mainly because there is a high possibility that the fine-tuning operation can update the algorithm's global best position and increase the fitness value. However, if the fine-tuning operation is used frequently in the early stage of the algorithm, it will be more likely to make the algorithm fall into the local optimum and fail to jump out. In addition, the fine-tuning algorithm itself will have K cycles, which consume about K times as much time as the other operations. Therefore, to avoid the long running time of the algorithm, a cost parameter C is introduced as an internal delay between successive fine-tuning operations. Each time after performing the fine-tuning operation, it needs to interval C steps before using the fine-tuning operation again.

5. Experiment Results and Analysis

To verify the effectiveness of the DQN-QPSO algorithm, a stochastic map model is established using the environment modeling method mentioned in Section 2. The proposed algorithm and the quantum particle swarm algorithm are compared in the same environment. For the underwater environment, the experiment is separated into two parts, i.e., path planning in the sea current-only environment and path planning in the environment with obstacles. The map size is set to $200 \times 200 \times 200$, and 100 Monte Carlo simulation trials are conducted, respectively, to initialize the map information and record the mean and variance of the fitness function for effectiveness analysis.

The parameters of QPSO are based on the problem of path planning and the parameters setting of DQN and ISPO as in [5, 26]. For the setting of weights of the fitness function $f_1, f_2,$ and f_3 , we are based on the actual situation to analyze. AUVs generate high energy consumption for navigation distance, the second effect is the sea currents; therefore, we give them relatively high weights. Although the deflection angle is less energy-intensive, frequent steering also has a negative impact on navigation safety. So, we set $f_1 = 1$, $f_2 = 0.25$, and $f_3 = 0.5$. The relevant parameters are set as shown in Table 1.

5.1. Path Planning in Sea Current Environment. In the actual AUV navigation, there are no obstacles at all times. To ensure the comprehensiveness of the algorithm in practical applications, the environment model with only sea currents is designed, setting the starting point as (0,0,0), and the endpoint is (200, 200, 100). The algorithm in this article is used to compare with the standard PSO algorithm, QPSO algorithm, and PSO-GWO algorithm [21] in the path planning and the fitness function graph for comparison, as shown in Figure 6. In the performance test of the PSO, the inertia weight ω is decreased linearly from 0.9 to 0.4 as in [20]. In the performance tests for QPSO the contraction-expansion coefficient β varies from 1.0 to 0.5 linearly when the algorithms are running [22].

Based on this, the sea current information was randomly initialized, and 100 Monte Carlo experiments were conducted separately to record the mean and variance of the fitness function, as well as the mean values of $F_1, F_2,$ and F_3 , as shown in Table 2.

It can be seen from Figure 6 that the paths designed by these methods are useful for AUV navigation in a current-only environment. However, compared with other algorithms, the algorithm used in this article can obtain smaller fitness function values, and the paths designed by it can make greater use of the currents. In the early stages of the algorithm, the DQN-QPSO algorithm converges slowly by using different decision tests due to the need to train the neural network better to make optimal decisions. However, as the iterations proceed, the algorithm can adaptively choose a better strategy to update the particle positions, keep refreshing the optimal fitness function in the middle of the algorithm, and further improve the fitness by fine-turning

TABLE 1: Parameter settings in this article.

Parameter	Value	
QPSO	N	50
	D	3
	T	300
Fitness function	$f_1, f_2,$ and f_3	1, 0.25, 0.5
	φ_{\max}	30
	p	20
	μ	10
DQN	α	0.01
	γ	0.99
	δ	10
ISPO	a	100
	b	1
	K	10
	C	3

operation in the later stages of the algorithm. It was turning operation further to improve the fitness function value by a small margin.

In contrast, compared to other algorithms, PSO and QPSO fall into local optimum very early. The PSO-GWO algorithm is inferior to the DQN-QPSO algorithm regarding local search capability. In addition, it can be concluded from the data in Table 2 that the proposed method in this article outperforms other path planning methods in terms of mean and variance.

5.2. Path Planning in Obstacle and Sea Current Environment. To simulate the navigation problem of AUV when navigating near the seafloor, a hybrid model of obstacles and currents was developed using the method in Section 2. The same four algorithms described in the previous section were compared from the path images as well as the graph of the fitness function, as shown in Figure 7. Also, 100 Monte Carlo experiments were conducted separately to record the mean and variance of the fitness function and the mean values of $F_1, F_2,$ and F_3 , as shown in Table 3. In addition, four terrains of different complexity were generated separately by changing the number of peaks to test the effectiveness of the algorithm, as shown in Figure 8.

As can be seen from the figure, the paths planned by these algorithms all ensure that the AUV successfully avoids known obstacles. All also reduce the energy consumption of the AUV by taking advantage of the favorable current flow, but the fitness function of the paths planned by DQN-QPSO is smaller. The energy consumption of the planned paths is lower.

A comprehensive analysis of the above results shows that more local optimal paths are added due to additional obstacles. From the fitness function plot, it can be seen that all four algorithms keep jumping out of the local optimum as the cycle progresses. However, both PSO and QPSO algorithms complete the convergence within 100 generations and need help finding the optimal solution. In contrast, the DQN-QPSO and PSO-GWO algorithms have a stronger global search capability. At the same time, the DQN-QPSO algorithm can further improve the accuracy of fine-turning operation, proving the algorithm's effectiveness. In addition,

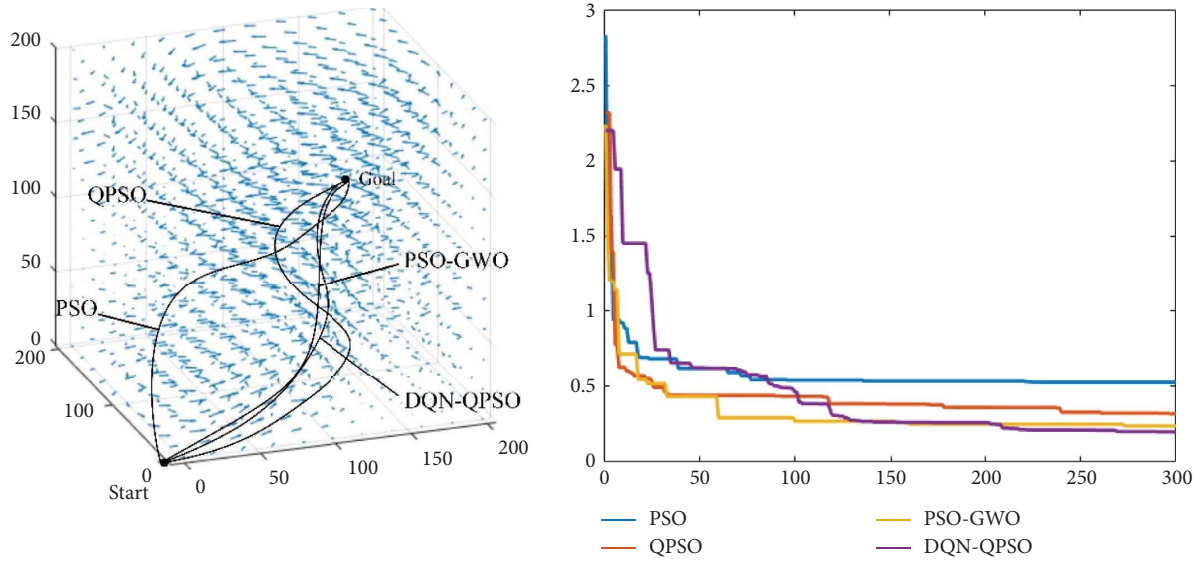


FIGURE 6: Path planning diagram in the sea current environment.

TABLE 2: Comparison of the results of the two algorithms.

Algorithm	F_c		F_1	F_2	F_3
	Average value	St. dev.	Average value	Average value	Average value
PSO	0.5794	0.9417	0.2192	0.5357	0.4526
QPSO	0.5041	0.7642	0.1935	0.4615	0.3905
PSO-GWO	0.3382	0.4335	0.1163	0.3950	0.2462
DQN-QPSO	0.3036	0.3952	0.0924	0.4015	0.2217

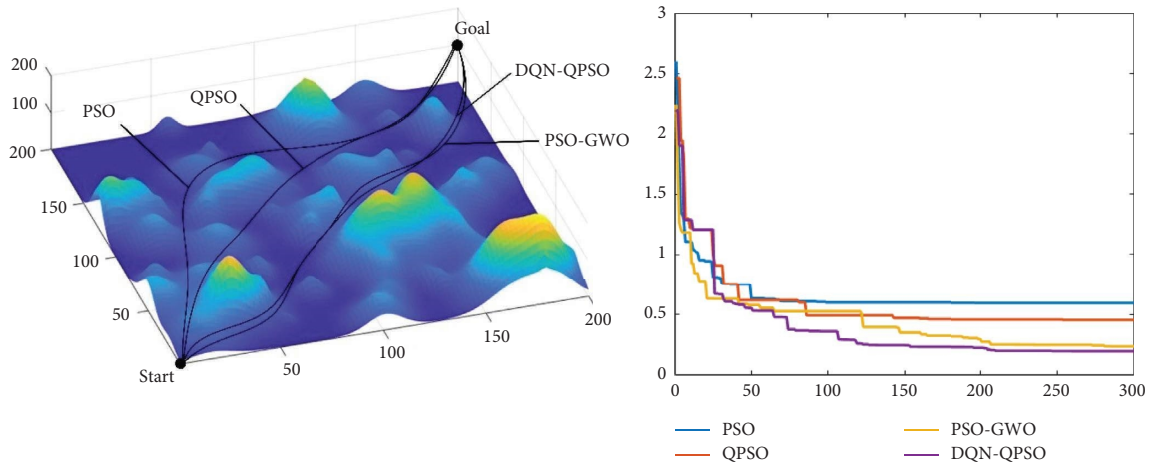


FIGURE 7: Path planning diagram in obstacle environment.

TABLE 3: Comparison of the results of the two algorithms.

Algorithm	F_c		F_1	F_2	F_3
	Average value	Variance	Average value	Average value	Average value
PSO	0.7045	0.8590	0.2540	0.6926	0.5547
QPSO	0.6019	0.6248	0.2125	0.6512	0.4533
PSO-GWO	0.4752	0.5271	0.1624	0.5780	0.3367
DQN-QPSO	0.4484	0.5393	0.1504	0.5626	0.3148

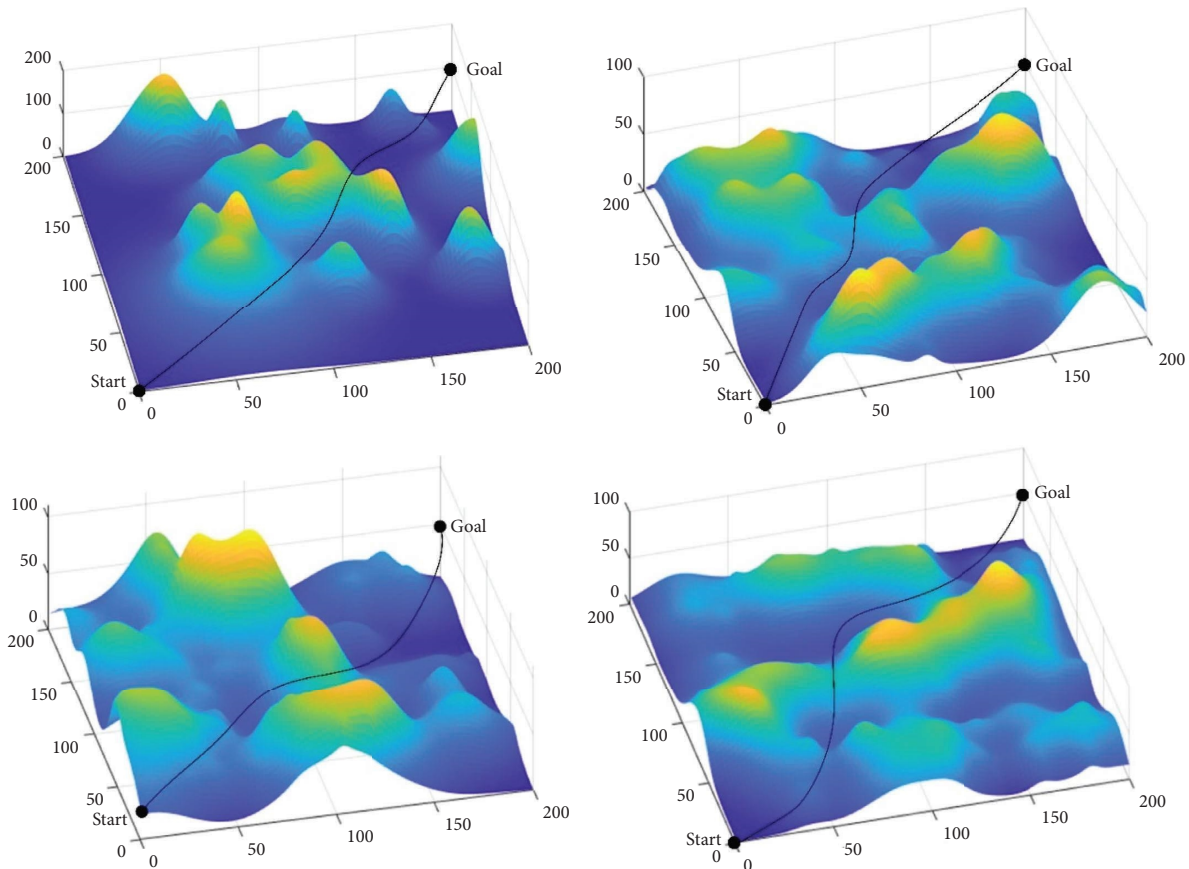


FIGURE 8: Tests of four different complexity environments.

the comprehensive test of the terrain of different complexity in Figure 8 shows that the algorithm has good path planning ability for different maps and has good practical value.

In conclusion, the improved QPSO algorithm dynamically selects five actions to update the position of the particle swarm by combining the ideas of DQN, in which three global search operations greatly improve the global search capability of the algorithm, making it possible to find the optimal path solution on the complex terrain with mixed currents and obstacles, while using fine-turning operations in the later stages of the algorithm to improve the accuracy of the path. At the same time, the fitness function designed in this article is a significant improvement for the QPSO algorithm in the underwater environment so that it can design the route by using the sea current. It can consider the path length, deflection angle, and sea current by designing different weight coefficients of the function, which can be set according to the actual situation in practical application.

6. Conclusion

This article proposes a path planning method combining deep Q-network and quantum particle swarm algorithm. Through simulation experiments, it is proved that the path planned by the improved algorithm can effectively avoid the obstacle model and further minimize the AUV energy consumption by using the current sea information. By

comparing the standard QPSO, it is found that the algorithm in this article outperforms the QPSO algorithm in terms of the fitness function, global search capability, and convergence speed, demonstrating the algorithm's effectiveness. However, the algorithm requires more parameters to be modified, and its computational speed is dramatically decreased when combined with a neural network, which still needs further research. In the subsequent research, the algorithm will be merged with the actual size of the AUV, and further research will be conducted from the perspective of obstacle avoidance to improve the algorithm's stability and solve the parameter adjustment problem.

Data Availability

The data used to support the findings of this study are simulated by MATLAB, which can be available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research work was supported and sponsored by the National Natural Science Foundation of China (Grant nos. U1806228 and U22A2044).

References

- [1] H. Jin and C. Jin, "A novel particle swarm optimization algorithm based on reinforcement learning mechanism for AUV path planning," *Complexity*, vol. 2021, Article ID 8993173, 13 pages, 2021.
- [2] M. Zhu and D. Zhu, "A workload balanced algorithm for task assignment and path planning of inhomogeneous autonomous underwater vehicle system," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 4, pp. 483–493, 2019.
- [3] Z. Zeng, K. Sammut, L. Lian, F. He, A. Lammas, and Y. Tang, "A comparison of optimization techniques for AUV path planning in environments with ocean currents," *Robotics and Autonomous Systems*, vol. 82, pp. 61–72, 2016.
- [4] J.-J. Li, R.-B. Zhang, and Y. Yang, "Multi AUV intelligent autonomous learning mechanism based on QPSO algorithm," in *Intelligent Computing Theories and Methodologies*, D.-S. Huang, K.-H. Jo, and A. Hussain, Eds., vol. 9226, Berlin, Germany, Springer International Publishing, 2015.
- [5] S. Guo, X. Zhang, Y. Du, Y. Zheng, and Z. Cao, "Path planning of coastal ships based on optimized DQN reward function," *Journal of Marine Science and Engineering*, vol. 9, no. 2, p. 210, 2021.
- [6] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, 2004.
- [7] R. Pearl and J. Pearl, "Generalized best-first search strategies and the optimality of A**," *Journal of the ACM*, vol. 32, no. 3, pp. 505–536, 1985.
- [8] S. Arinaga, S. Nakajima, H. Okabe, A. Ono, and Y. Kanayama, "A motion planning method for an AUV," in *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, Monterey, CA, USA, June 1996.
- [9] R. Kennedy and J. Kennedy, "A new optimizer using particle swarm theory MHS'95," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39–43, Nagoya, Japan, June 1995.
- [10] A. Colomi, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," in *Proceedings of the ECAL91 - European Conference on Artificial Life*, Paris, France, September 1991.
- [11] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds., Springer New York, New York, NY, USA, 1986.
- [12] D. A. Nyongesa and H. O. Nyongesa, "Genetic algorithms for fuzzy control.1. Offline system development and application," *IEE Proceedings - Control Theory and Applications*, vol. 142, no. 3, pp. 161–176, 1995.
- [13] V. Pshikhopov, "Implementation of intelligent control system for autonomous underwater vehicle," *Applied Mechanics and Materials*, vol. 701-702, pp. 704–710, 2014.
- [14] X. Hou, J. Du, J. Wang, and Y. Ren, "AUV path planning with kinematic constraints in unknown environment using reinforcement learning," in *Proceedings of the 2020 4th International Conference on Digital Signal Processing*, pp. 274–278, Chengdu China, June 2020.
- [15] Z. Yan, J. Li, J. Zou, J. Li, and R. Zhong, "A hybrid PSO-WG algorithm for AUV path planning in unknown oceanic environment," in *Proceedings of the 2018 IEEE 8th International Conference on Underwater System Technology: Theory and Applications (USYS)*, Wuhan, China, December 2018.
- [16] H. S. Lim, S. Fan, C. K. H. Chin, S. Chai, and N. Bose, "Particle swarm optimization algorithms with selective differential evolution for AUV path planning," *IAES International Journal of Robotics and Automation*, vol. 9, no. 2, p. 94, 2020.
- [17] X. Wu, W. Bai, Y. Xie, X. Sun, C. Deng, and H. Cui, "A hybrid algorithm of particle swarm optimization, metropolis criterion and RTS smoother for path planning of UAVs," *Applied Soft Computing*, vol. 73, no. 12, pp. 735–747, 2018.
- [18] X. Wu, Y. Yang, Y. Sun, Y. Xie, X. Song, and B. Huang, "Dynamic regional splitting planning of remote sensing satellite swarm using parallel genetic PSO algorithm," *Acta Astronautica*, vol. 204, pp. 531–551, 2023.
- [19] J. Zhang and Y. Gong, "A niching PSO-based multi-robot cooperation method for localizing odor sources," *Neurocomputing*, vol. 123, pp. 308–317, 2014.
- [20] J. Zhang, Y. Zhou, and Y. Zhou, "Path planning of mobile robot based on hybrid multi-objective bare bones particle swarm optimization with differential evolution," *IEEE Access*, vol. 6, pp. 44542–44555, 2018.
- [21] X. Cheng, J. Li, C. Zheng, J. Zhang, and M. Zhao, "An improved PSO-gwo algorithm with chaos and adaptive inertial weight for robot path planning," *Frontiers in Neurobotics*, vol. 15, Article ID 770361, 2021.
- [22] J. Sun, W. Xu, and B. Feng, "Adaptive parameter control for quantum-behaved particle swarm optimization on individual level," in *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3049–3054, Waikoloa, HI, USA, May 2005.
- [23] W. Wang, Z. Zhou, and Y. Yao, "Research on autonomous planning method based on improved quantum particle swarm optimization for autonomous underwater vehicle," in *Proceedings of the OCEANS 2016 MTS/IEEE Monterey*, Monterey, CA, USA, September 2016.
- [24] W. Fang, J. Sun, Y. Ding, X. Wu, and W. Xu, "A review of quantum-behaved particle swarm optimization," *IETE Technical Review*, vol. 27, no. 4, p. 336, 2010.
- [25] J. Sun, W. Xu, and W. Fang, "Quantum-Behaved Particle Swarm Optimization Algorithm with Controlled Diversity," *Computational Science - ICCS*, vol. 3993, p. 8, 2006.
- [26] Z. Ji, H. Liao, Y. Wang, and Q. H. Wu, "A novel intelligent particle optimizer for global optimization of multimodal functions," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 3272–3275, Singapore, September 2007.