Research Article

Multirobot Coverage Path Planning Based on Deep Q-Network in Unknown Environment

Wenhao Li, Tao Zhao, and Songyi Dian

College of Electrical Engineering, Sichuan University, Chengdu 610065, China

Correspondence should be addressed to Tao Zhao; zhaotaozhaogang@126.com

Received 13 May 2022; Accepted 13 July 2022; Published 3 August 2022

Academic Editor: Xianfeng Yuan

Copyright © 2022 Wenhao Li et al. 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.

Aiming at the problems of security, high repetition rate, and many restrictions of multirobot coverage path planning (MCPP) in an unknown environment, Deep Q-Network (DQN) is selected as a part of the method in this paper after considering its powerful approximation ability to the optimal action value function. Then, a deduction method and some environments handling methods are proposed to improve the performance of the decision-making stage. The deduction method assumes the movement direction of each robot and counts the reward value obtained by the robots in this way and then determines the actual movement directions combined with DQN. For these reasons, the whole algorithm is divided into two parts: offline training and online decision-making. Online decision-making relies on the sliding-view method and probability statistics to deal with the nonstandard size and unknown environments and the deduction method to improve the efficiency of coverage. Simulation results show that the performance of the proposed online method is close to that of the offline algorithm which needs long time optimization, and the proposed method is more stable as well. Some performance defects of current MCPP methods in an unknown environment are ameliorated in this study.

1. Introduction

With the continuous development of robot technology, robots gradually appear in a variety of tasks to improve the efficiency of task or reduce labor costs. As a part of robot application, coverage task has been widely used in area cleaning [1, 2], disaster detection [3], postdisaster rescue, and other fields. In the first place, researchers focus on coverage path planning (CPP) for individual robot and achieve good results [4, 5]. Due to the different methods of map representation, model building, path generation, and their cross application, there are various methods of CPP. Some methods use trapezoidal decomposition or boustrophedon decomposition to decompose the map into several simple subregions and then cover them in sequence [6, 7]. The grid-based method is another basic method, including the spanning tree coverage (STC) algorithm [8], wavefront algorithm [9], etc.

However, with the increase of coverage area and difficulty of the task, a single robot is difficult to achieve the task requirements. Then, multirobot coverage path planning (MCPP) rises in response to the proper time and conditions [10]. The difficulties of MCPP lie in the communication, cooperation, and collision avoidance among robots [11, 12]. In order to solve these problems, some methods based on the single-robot CPP algorithm make great improvements. For example, the map is decomposed into subregions, and then the STC algorithm is applied in these subregions [13]. Some methods are also based on map decomposition [14], where simulation results show the effectiveness of the proposed method. The template method is also a common method in MCPP. With the study of some optimization algorithms [15], heuristic algorithms are also used to optimize the coverage problem [16]. It works well in simple environments, while in complex environments, robots may get stuck in dead zones. Aiming at this problem, the A* algorithm is applied to escape the dead zones [17]. Another basic method worth mentioning is the biologically inspired neural network (BINN) method, which uses activity of grids that help robots select actions. Some improvements are made in [18],
and the problems about large amount of calculation and big cost to escape from dead zones are solved.

All of these methods require prior information on the environment. Nevertheless, in some application scenarios, it is difficult to obtain all the environmental information and plan the path offline before the task starts, such as large-scale search and rescue [19] and mine clearance [20]. Therefore, online MCPP methods are significant in unknown environment. In contrast, there are few researches on MCPP in unknown environment, and the current methods are immature. Robots are divided into exploration and coverage robots in [21], and the explorers cover the boundaries of the current region, while coverage robots move back and forth between the explorers. When the multirobot team meets reverse-convex (RCV) obstacles, it will split into two subteams. The subteams cannot split again. However, this method has requirements for both the start points of robots and the number of robots. A method proposed in [22] can map out the approximate path and provides ideas for future researches. The multirobot hex decomposition exploration (M-HDE) method has been proposed in [23], in which the energy restriction is taken into consideration. In this method, the map environment is represented as hexagons, and then smooth, continuous paths are generated for the robots formation. But the simulation results show the paths occasionally crossed the obstacles, which may bring some danger.

Focusing on above problem, extensive researches and applications of reinforcement learning (RL) in robotics [24] catch our attention. Nevertheless, these studies usually focus on the pathfinding problem or the single-robot coverage problem in the known environment [25]. We think this is because the target of the coverage problem is to cover all nonbarrier areas, which is more complicated than pathfinding, especially in unknown environment. Consequently, using RL to solve the MCPP problem in unknown environment requires a huge amount of calculation, which is difficult to meet in practical applications. Therefore, this paper tries to reduce the amount of RL training and solves the difficulties in MCPP in unknown environment. Compared with the current methods, the main contributions of this paper can be summarized as follows:

(i) At present, most of the coverage algorithms are offline algorithms, and the planning takes a long time. The coverage problem is divided into offline training and online decision-making in the proposed method. The robots acquire prior knowledge in offline training and are able to make decisions quickly in the face of unknown environments online because different target environments only affect the decision-making stage.

(ii) Drawing on the Monte Carlo tree search method, the multirobot deduction method is proposed. This deduction method selects some possible better action groups for deduction according to the training network and judges the quality of the action groups by rewarding feedback. Through this method, under the condition of decentralized training, multirobot cooperation and collision avoidance are well solved, thereby reducing the amount of training.

(iii) In order to save the robot’s storage space and reduce the amount of training, during the training phase, the environment is known and fixed in size. In practice, however, the target environment is unknown, and it is highly likely that the size does not match the training environment. In view of this situation, the global image of the environment is generated multiple times according to the probability of the area that has been explored, and the deductive method is used in multiple images, and finally, each robot action is selected by voting. Second, if the environment size does not match the training size, the sliding-view method is used, and each robot is as centered as possible on itself, interpreting the environment information as a network input.

In order to implement this algorithm, the robot needs to be equipped with a basic computing microcomputer (such as Raspberry Pi) and a laser distance sensor (LDS). The LDS needs to sense the obstacle information in each direction, so the recommended installation position is the top of the robot.

The rest of the paper is organized as follows. Section 2 introduces the related works. The proposed method is provided in section 3. The comparison results and the conclusion are presented in sections 4 and 5, respectively.

2. Related Works

2.1. Deep Q-Network. In 2013, the concepts of deep reinforcement learning (DRL) and the Deep Q-Network (DQN) algorithm were proposed on Neural Information Processing Systems (NIPS) by the DeepMind team [26]. Forty-nine classic Atari2600 games demonstrated the powerful performance of DQN. Since then, both the DeepMind team and other researchers have made improvements on this basis to improve the performance of DQN [27, 28]. Some researchers consider double DQN, fuzzy systems [29], or prioritized experience replay. Double DQN has better ability to alleviate the overestimate problem; for this reason, double DQN is applied in this paper. The main formulas of DQN are (1) and (2):

\[ Q'(s, a) = E_r[s, a] - \gamma \max_{s'} Q^*(s', a') s, a, \]

\[ L_i(\theta) = E_{s,a,r} \left[ (E_r[s, a] - Q(s, a; \theta))^2 \right]. \]

\[ s \text{ means the state, and } a \text{ means the action; } s t \text{ and } a t \text{ mean the state and action in the next time; the very important change from double DQN is that } a t \text{ is selected by the main network but not the target network. } r \text{ is reward given by the environment according to } R_t = R_q, \gamma \text{ is the discount factor. } Q \text{ is the value of DQN, and } L_i(\theta) \text{ is the loss function in iteration } i \text{ and is used to train the network. } y \text{ is the target value, and it is equal to } r + \gamma \max_a Q'(s', a'; \theta_i'), \text{ which is influenced by the previous iteration parameter } \theta_i'. \]
2.2. Monte Carlo Tree Search. This paper is inspired by the excellent performance of the Monte Carlo tree search (MCTS) method in the chess game [30, 31] and try to reduce the consumption of training, meanwhile ensuring the quality of the coverage by this way. First, the basic processes of MCTS are introduced here, then the difficulties to apply this algorithm to CPP problems are analyzed, and finally, the modification method is described in the later section. If states are represented as nodes and actions as directed edges, then MCTS can be represented as a tree construction, as shown in Figure 1. MCTS has four basic processes: selection, expansion, simulation, and backpropagation:

(i) Selection: choose the action that is most likely to reach the unevaled state.
(ii) Expansion: in a certain state, after the action A, the new state $S_{t+1}$ is obtained; then $S_{t}$ expands $S_{t+1}$ as its child node.
(iii) Simulation: start from a certain state and choose actions by some strategy, until the end.
(iv) Backpropagation: at the end of the simulation of a certain state, the information of each parent node is updated along the way according to the outcome, until the root node.

3. Methodology

3.1. Map Presentation. In this paper, the grid-based method is used to represent the target area. The grid size depends on the effective range of the coverage sensor carried by the robot. As shown in Figure 2, $r_{s}$ represents the coverage radius of the sensor, and $l$ represents the length of grid sides. $l$ has to follow:

$$I \leq \sqrt{2}r_{s}. \quad (3)$$

Once the grid size is determined, the task area can be rasterized. $g_{i,j}$ is used to describe the grid in row $i$ and columns $j$. In the rasterization process, if a grid contains any obstacle area, the whole grid will be regarded as an obstacle grid. Conversely, a grid is considered to be a grid waiting for covering which consists entirely of reachable areas. The set $S = \{g_{1,1}, g_{1,2}, \ldots, g_{i,j}, g_{i+1,1}, \ldots, g_{i,j}\}$ represents all grids in a two-dimensional map with $i$ rows and $j$ columns; obstacle grids, grids waiting for covering, and covered grids are represented by $S_{o}$, $S_{w}$, and $S_{d}$, respectively. $S = S_{o} \cup S_{w} \cup S_{d}$ and the intersections of $S_{o}$, $S_{w}$, and $S_{d}$ are empty sets. The grids currently occupied by $n$ robots are represented by $P_{n} = \{p_{1}, p_{2}, \ldots, p_{n}\} \subseteq S_{d}$. As shown in Figure 3, black represents the obstacle area $S_{o}$, and white represents the feasible area $S_{w}$. The robot will start from the start point and move to the feasible direction, so as to cover the feasible area. The current position of the robot is represented by a red circle, and the covered grids $S_{d}$ are represented by blue. The conditions for ending the task are $S_{w} = \emptyset$, or the robots are stuck in a dead zone from which it cannot escape.

In the CPP problem, the main metrics we pay attention to are coverage rate, time cost, energy consumption, and so on. The repetition rate can be a good indicator of time cost, energy consumption, and other indicators, especially in the case of complete coverage. In the later simulation, the repetition rate is used frequently to represent the performance of the method. The coverage rate $r_{c}$ and the repetition rate $r_{r}$ are grid-based metrics and related to the path of robots. Hence, there are some definitions that should be made first. Because the paths of robots consist of ordered, repeatable, and adjacent grids, they are represented by variable-length arrays rather than sets. The path of $i$th robot can be denoted as $P_{i} = [g_{i_1,j_1}, g_{i_2,j_2}, \ldots, g_{i_{m_i,j_i}}]$. $L_{P_{i}}$ means the length of array $P_{i}$; $t = L_{P_{i}}$ is because $i$th robot has taken $t$ steps so far. Then the calculation formulas of $r_{c}$ and $r_{r}$ can be defined as shown in (4) and (5).

$$r_{c} = \frac{|S_{o}|}{(|S| - |S_{o}|)} \quad (4)$$

$$r_{r} = \frac{\sum_{i=1}^{n} L_{P_{i}} |S_{d}|}{|S| - |S_{o}|} \quad (5)$$

If the robot achieves complete coverage, $S = S_{o} \cup S_{d}$, and $S_{w} = \emptyset$. Therefore $r_{c}$ is only related to total length of paths in this condition.

In order to transform the index of the robot coverage problem into the RL model, the following settings are made: whenever the robot visits a reachable grid $g_{i,j} \in S_{w}$, the robot can get a reward $R_{1}$ and then alter the assembly state of $g_{i,j}$.
(g_{i,j} \notin S_o \text{ and } g_{i,j} \in S_f). If the robot achieves the complete coverage of the task area \((S_o = \emptyset)\), a larger negative reward \(R_2\) will be obtained. On account of speeding up the convergence of training, a small negative reward \(R_4\) is added to each step of the robot. For processing of illegal areas \(S_o\) (such as boundaries and obstacles), a negative reward \(R_4\) is given if a robot accesses the illegal grid \(g_{i,j} \notin S_o\) and then makes the robot stay in the place \(g_{i,j} \notin S_o\), \(R_4\) is set as a unit and pick other reward parameters in view of simulation results. The simulation environment is a \(6 \times 6\) map \((i = j = 6)\), and the maximum number of robot steps is limited to 180. The \(\epsilon\)-greedy strategy is used in training. There are 8 sets of parameters that have been picked as representatives, and the average of the last 10 episodes' results are taken and shown in Table 1. \(R_1 = 1\), \(R_3 = 10\), \(R_4 = -0.01\), and \(R_4 = -1\) are the best sets of parameters in our simulations.

### 3.2. Training

Multiagent reinforcement learning is an important part of RL. There are many discussions on Nash equilibrium. The fully centralized method and centralized training with the decentralized execution method are adopted primarily. Naturally, in these methods, there is more resource consumption than in decentralized training. For the MCPP problem, the huge computing consumption of training multiple centralized robots cannot be afforded, especially when the number of robots is uncertain. Finally, the single-agent reinforcement learning method (DQN) is selected for these reasons above, and how to resolve multiagent conflicts is explained later in this article.

The state is the network input in DQN. In the coverage problem of the given environment, each grid has three possibilities: \(S_o\), \(S_d\), and \(S_w\). In the initial simulations, a matrix is used, and the matrix size is corresponding to the size of the environment to represent the state, and different numbers represent the different situations of the grids (0: \(S_o\); 1: \(S_d\); 2: \(S_f\); 3: \(P_r\)). However, it is found that this state representation method leads to sluggish convergence. In our opinion, it is owing to the concentrated representation of multiple grid situations that the convolution layer has more burden of feature extraction. After this, one-hot encoding is used to modify the input states; 001, 010, and 100 represent \(S_o\), \(S_d\), and \(P_r\), respectively. If a grid is neither an obstacle nor has it been accessed or is being visited, it must be a grid waiting for the robot's coverage. Therefore, in this way, there is no need to independently encode the grid waiting for covering (or it can be regarded as 000). Finally, each bit of the encoding is set to a different channel, and thus, the matrix is transformed into a three-channel input, as shown in Figure 4. Simulations demonstrate the effectiveness of this state representation: all other conditions being equal and \(i = j = 6\), there are 3000 episodes of training for both methods, and the environment maps for each episode are randomly generated. Some of the other training parameters are shown in Table 2. In order to observe repetition rate variation clearly, the averages of repetition rates per 100 episodes are calculated. The results are shown in Figure 5. The results in Figure 5(c) are the results from Figure 5(a) minus the results from Figure 5(b). Both these methods can achieve complete coverage. The difference of repetition rates demonstrate that the three-channel input method gains the edge after 500 episodes of training. As a result, all simulations without a detailed description in this paper adopt this state representation way. In order to prove that the proposed method is applicable to a larger environment, the same method is used to train environments when \(i = j = 10\), and it can be seen in Figure 6 that although the repetition rate is high, the repetition rate curve still has a downward trend. These results prove that the wider environment requires a greater amount of training.

### Table 1: The average results of the last ten episodes when reward parameters change.

| \(R_1 = 1\) | \(R_2\) | \(R_3\) | \(R_4\) | Coverage rate | Repetition rate |
|---|---|---|---|---|---|
| 10 | 0 | -10 | 96.8% | 378% |
| 10 | -0.01 | -1 | 100% | 163.4% |
| 10 | -0.01 | -10 | 98.8% | 306.8% |
| 10 | -0.1 | -1 | 100% | 170.3% |
| 10 | -0.1 | -10 | 100% | 197.7% |
| 100 | -0.01 | -1 | 97.4% | 382% |
| 100 | -0.01 | -10 | 95.4% | 356.3% |
| 100 | -0.1 | -1 | 92% | 422.3% |

### Figure 3: Grid-based map presentation

(a) Initial environment. (b) Environment after processing. (c) The robot in processed environment.
3.3. Execution: Deduction Method. Although the reward parameters are adjusted and the encoding of state input is modified in the previous content to speed up training, due to the complexity of the CPP problem, each initial environment still requires a lot of time to train. In addition, when facing with real situations where target area information is not available in advance, the urgency of the task does not allow for prolonged training using the target environment information. In other words, robots should obtain a different initial environment and begin the coverage immediately. For this reason, a large number of different initial environments can be trained in advance to simulate various real situations so that the network can be more accurate for each state. The huge consumption of training in each environment and the large number of different environments (for example, there are about $5.4 \times 10^{18}$ kinds of different states when $i = j = 6$ and about $5.2 \times 10^{69}$ kinds of different states when $i = j = 10$) make the task difficult to accomplish.

AlphaGo can adopt MCTS in the execution stage, and there are some primary reasons: first, AlphaGo has multiple threads, central processing units (CPUs), and graphics processing units (GPUs). As a result, there is a large capacity of calculation, and an enormous amount of computation can be borne. Second, AlphaGo has both policy network and value network. The choice of action and the assessment of state use different networks, respectively. Reciprocally, computing resources are limited in CPP. What is more, the DQN-based method only has the value network. After considering these difficulties and focusing on the idea of deduction, which is the key of MCTS, in the game of Go, an action is made in a certain state, but the next state is affected by the opponent’s move. In a CPP problem, the next state is determined after the action choice. A simpler environment allows the simulation stage to use the value network and get acceptable results. On the other hand, if simulation is continued until the end of the task, the task may be impossible to complete due to the inaccuracy of the value network: the robot may move back and forth between certain grids. Another reason for no continuous simulation until the end of the task is that not only the complete coverage ability but also the repetition rate are concerned, and the repetition rate is even more important. For these reasons, the number of simulation steps is set as the parameter $N_{\max \text{ step}}$. Finally, the final decision of the MCTS is the action which is selected the most times (the more times selected, the more likely to win). In the CPP problem, the robot only has four actions at most (front, back, left, and right), and it is much less than Go. We can evaluate all the actions by the rewards in the simulation stage. The algorithm is displayed in Algorithm 1.

Comparative simulation demonstrates the effectiveness of the proposed method. The experimental group uses the deduction method, while the control group uses the value network for execution. In Table 3, for each simulation, a random map is generated for two groups to cover. Both groups use the same network which is under 3000 episodes of training. The simulation was repeated 100 times, and each time, the environment was randomly generated. Each grid has a 5% chance of being an obstacle grid, and if the connectivity of all nonobstacle grids cannot be guaranteed, all obstacle grids are regenerated. Finally, the starting point location of the robot is randomly selected in the waiting coverage area. If the map is covered completely, the corresponding data in this table are the repetition rates. The character string "NULL" is filled in the form otherwise. The experimental group achieves complete coverage in 100

| Parameters                      | Values     |
|---------------------------------|------------|
| Replay memory size              | 100000     |
| Discount factor                 | 0.99       |
| The initial value of the greedy exploration | 1          |
| The final value of the greedy exploration | 0.25      |
| Robot’s maximum number of steps each episode | 180        |
| Learning rate                   | 0.0005     |
| Target network update frequency | 50 (episodes) |

Figure 4: The schematic diagram of state input.

### Table 2: The list of parameters of training.
experiments, and the average repetition rate is 6.92%. In contrast, the control group fails to cover completely in 17 experiments, and in other experiments, the average repetition rate is 15.05%. This simulation emphasizes that the deduction method is much better than the way it uses the value network only.

The next step is to derive single-robot decisions from multirobot decisions. The difficulty of MCPP is how to allocate tasks reasonably and avoid collision. Due to the result of limited computing resources in training, extra computation is not expected to see, so the existing value network is used to solve the problem of task allocation and collision avoidance. Through experiments, it can be found that the deduction method proposed in the previous section can achieve task allocation and collision avoidance problems when the distance between robots is close. However, the robots bring little influence and can be ignored in far distance. Figure 7 illustrates how the deduction method works with two close robots. If the decision is made by the value network, robots $R_a$ and $R_b$ will often choose the same grid B as the target to cover. After grid B is covered, grid A is selected as the target by two robots. This leads to an increase in repetition rates, which represents an increase in energy expenditure and task time. The deduction method works by
the rewards obtained for each step and the discount factor. Obviously, one robot aiming at grid A and the other robot aiming at grid B will get the highest total reward. In order to make the deduction method suitable for MCPP, some modifications have been made:

(i) In multirobot coverage path planning, the objects to score are action groups instead of actions, which contain the actions selected by each robot and the priority of each robot (after decision robots run in the same time).

(ii) The number of possible actions for a single robot is 4; the number of possible action groups is $4^n$ with $n$ robots. This exponential complexity is obviously unacceptable, so at most, 64 action groups are chosen according to the value network in the beginning.

(iii) The action groups are evaluated by the total rewards of all robots and the discount factor.

(iv) Finally, select the action group with the highest score, and let each robot perform the corresponding action.

The algorithm is shown in Algorithm 2. The complexity of the whole algorithm is determined by the number of robots $n$ and the maximum simulation steps $N_{\text{max step}}$. As mentioned above, in order to avoid the rapid increase in the time complexity of the algorithm with a large number of robots, the number of action groups is limited. Otherwise, the number of action groups is equal to the cubed number of robots. In later simulations, $N_{\text{max step}}$ tends to take 3, and the discount factor $\gamma$ equals to 0.95. (The discount factor in Algorithm 2 is not equal to the discount factor at the time of training.) By this algorithm, the multirobot can finish coverage tasks in a known environment.

3.4. Unknown Environment: Probability and Statistics. Although the proposed method of the unknown environment in this paper can be used in most cases, it is better for random or uniform distribution of obstacles, such as the field environment where trees grow randomly. In an unknown environment, robots should follow these two basic requirements:

(i) In grid-based map, the sensor used by the robot for perception can at least explore the information of neighboring grids around it (including vertex adjacency).

(ii) The robot can record the information of previously explored grids.

Figure 8 shows the unknown environment in the grid-based map. The grids are represented by gray color if they have never been explored. In the beginning, the robot only knows the information about the surrounding grids. As it continues to cover, observe, and record, the robot can collect more information about the grids, and as a result, the number of unexplored grids is reduced. In known grids, there are three possible situations that are discussed before: $S_o$, $S_w$, and $S_d$. However, in unknown grids, there are only two situations: $S_o$ and $S_w$.

Therefore, when converting the current state to a three-channel input, the second channel and third channel are fixed. Only the first channel is influenced by the unexplored grids. The contents of the unknown part of the channel can be regarded as a two-point distribution that obeys probability $P_o$:

$$P_o = \frac{|S_{ko}|}{|S_{ko}| + |S_{kw}| + |S_{kd}|},$$  \hspace{1cm} (6)

where $S_{ko}$, $S_{kw}$, and $S_{kd}$ mean $S_o$, $S_w$, and $S_d$ grids within a known range, respectively. Equation (6) signifies that the grids are recorded by the robot and viewed as a sample of the entire environment. The probability of obstacles in the entire environment is estimated by treating these explored and recorded grids. All unexplored grids in the first channel are evaluated according to probability $P_o$. A total of $M$ times are repeated to generate $M$ maps, and actions are selected and voted by applying the deduction method in the previous section in $M$ maps. Finally, the action with the most votes is executed really.

**Algorithm 1:** Deduction method in CPP modified by MCTS.

```
Initialization: maximum number of simulation steps $N_{\text{max step}}$, $i$th action score $R_i$, discount factor $\gamma$

for $i = 1$ to 4 (four directions) do
  if this direction is feasible then
    while the number of current step $N_{\text{step}} < N_{\text{max step}}$ do
      Choose a feasible next action according to the value network
      $N_{\text{step}} = N_{\text{step}} + 1$
      Update $E_{\text{temp}}$ according to selected action
    end while
    Calculate $R_i$ according to $\gamma$ and the rewards for each step and the value network
  else
    $R_i = -1$
  end if
end for
```
Table 3: The single-robot repetition rates (%) of different execution methods in 100 simulations.

| Simulation number | Experimental group | Control group |
|-------------------|--------------------|---------------|
| 1                 | 8.6                | 8.6           |
| 2                 | 11.8               | 14.7          |
| 3                 | 5.7                | 5.7           |
| 4                 | 8.8                | 32.4          |
| 5                 | 0                  | 11.4          |
| 6                 | 9.7                | 22.6          |
| 7                 | 2.9                | NULL          |
| 8                 | 11.4               | 45.7          |
| 9                 | 15.2               | 27.3          |
| 10                | 14.7               | 41.2          |
| 11                | 23.5               | 26.5          |
| 12                | 5.9                | 26.5          |
| 13                | 2.9                | 22.9          |
| 14                | 5.7                | 5.7           |
| 15                | 11.4               | 11.4          |
| 16                | 8.6                | NULL          |
| 17                | 3.1                | 21.9          |
| 18                | 2.9                | 2.9           |
| 19                | 11.8               | 41.2          |
| 20                | 11.8               | 35.3          |
| 21                | 5.9                | 11.8          |
| 22                | 15.2               | 27.3          |
| 23                | 5.9                | 8.8           |
| 24                | 2.9                | 2.9           |
| 25                | 5.9                | NULL          |
| 26                | 12.1               | NULL          |
| 27                | 2.9                | 2.9           |
| 28                | 0                  | 2.9           |
| 29                | 2.8                | 13.9          |
| 30                | 0                  | 6.1           |
| 31                | 18.2               | 27.3          |
| 32                | 0                  | 5.6           |
| 33                | 11.4               | 8.6           |
| 34                | 8.6                | 20            |
| 35                | 8.6                | 37.1          |
| 36                | 8.8                | 14.7          |
| 37                | 5.9                | 11.8          |
| 38                | 0                  | 0             |
| 39                | 8.8                | 14.7          |
| 40                | 8.6                | NULL          |
| 41                | 5.6                | 11.1          |
| 42                | 0                  | 0             |
| 43                | 6.3                | NULL          |
| 44                | 12.1               | 30.3          |
| 45                | 2.9                | 2.9           |
| 46                | 5.7                | 17.1          |
| 47                | 0                  | 2.8           |
| 48                | 16.1               | NULL          |
| 49                | 0                  | NULL          |
| 50                | 2.9                | 2.8           |
| 51                | 12.5               | 18.8          |
| 52                | 8.8                | NULL          |
| 53                | 14.7               | NULL          |
| 54                | 0                  | 11.1          |
| 55                | 0                  | 5.6           |
| 56                | 5.7                | 11.4          |
| 57                | 0                  | 0             |
| 58                | 12.1               | 18.2          |
| 59                | 0                  | 0             |
| 60                | 15.6               | 15.6          |
This planning way achieve more rewards in same times

Figure 7: The deduction method works with two close robots.
Similarly, experiments are designed to verify the effectiveness of the method. \( M \) is set to 1, 2, and 5 to observe the influence on coverage performance. Here are four groups in the experiment. In group 1, the environment is known, and group 2 uses the proposed method, and parameter \( M \) is different. Group 3 and group 4 regrade unexplored grids as the feasible grids and obstacle grids, respectively. A total of 120 experiments are conducted in these four conditions. Table 4 shows the results.

When \( M \) is small, the results may be unstable but acceptable. This is because the voting session becomes meaningless when generated maps are not enough. For example, a random action will be chosen in the “1:1” conditions (and that is why it gets bad results when \( M \) equals 2). Randomness can also lead to better results: a map generated based on probability may get an appropriate action (of course, the results could be worse, with an average repetition rate of 0.1 percent higher in group 2 than in the third group over an additional 40 experiments). However, larger \( M \) will get more stable results with a longer response time. The parameter \( M \) can be set according to the real condition.

3.5. Nonstandard Size Environments: Sliding View Method.

In the previous simulations, the training and execution are carried out in \( 6 \times 6 \) environments; however, the situation must be considered that the practical size is bigger or smaller than \( 6 \times 6 \) size. A simple case is discussed first, when the representation of the target environment is smaller than the training size, for example, the target environment is expressed having \( 5 \times 5 \) size. A simple approach is to treat the boundary as an obstacle, as shown in Figure 9(a). The situation is more complicated when the environment representation is larger than the training size. The first method is to train in a larger environment as much as possible. Although the coverage performance of this method may be better, the training time will increase geometrically as the training size increases. As shown in Figure 6, during training, repetition rates decreased much more slowly in the \( 10 \times 10 \) environment than in the \( 6 \times 6 \) environment. Another processing method named sliding view is proposed in this paper.

The sliding-view method not only records all the explored grids information but also generates robots’ own

---

**Initialization:**

- Maximum number of simulation steps \( N_{\text{max step}} \)
- Discount factor \( \gamma \)
- Temporary environment information \( E_{\text{temp}} \)
- Action of each robot \( a_i \)
- Total rewards from \( i \)th action group \( R_i \)
- Current reward from selected action \( r_{\text{now}} \)

Choose \( k (k < 64) \) action groups for \( n \) robots \( A_k = (a_{k1}, a_{k2}, \ldots, a_{kn}) \)

for \( i = 1 \) to \( k \) (k action groups) do

\( N_{\text{step}} = N_{\text{max step}} \)

while the number of current step \( N_{\text{step}} < N_{\text{max step}} \) do

for \( j = 1 \) to \( n \) do

if \( a_{ij} \) is feasible then

Choose a feasible next action according to the value network,
\( r_{ij} = r_{ij} + \gamma^{N_{\text{step}}-1} r_{\text{now}} \)

Update \( E_{\text{temp}} \) according to selected action

else

\( r_{ij} = -1 \)

end if

end for

\( N_{\text{step}} = N_{\text{step}} + 1 \)

end while

\( R_i = r_{i1} + r_{i2} + \cdots + r_{in} \)

end for

Choose an action group by \( R \)

---

**Algorithm 2:** Deduction method in MCPP.

---

**Figure 8:** The unknown environment shown in the grid-based map.
views with the training size. The method keeps the robot in the center of the view as much as possible and provides special treatment if the robot is close to the boundary, as shown in Figure 9(c). If there is no grid to be covered in views with the training size. k–he method keeps the robot in waiting for covering. Because the size of view is the same as the training size, the DQN can work successfully, and other robots can select the optimal action group according to Algorithm 2. When affordable training costs are low, this isa compromise method without additional cost. k–he advantage of this approach is that the action value function does not contain all known information about the environment; in other words, the sliding-view method is a greedy method that focuses on the local.

Different size environments are dealt with the sliding-view method, with the method of regarding boundary as an obstacle. When the environment is small, two robots participate in the simulation, and three robots participate in larger environment. A set of results is picked and as shown in Figure 10, the different colored lines in the figure represent the trajectories of different robots. The robots complete the coverage task in different environments, and these results prove that the size of the environment that can be processed breaks through the limitation of the training size by the proposed methods.

| serial number | Group 1 | Group 2 | Group 3 | Group 4 | Group 5 | Repetition rate (%) |
|---------------|--------|--------|--------|--------|--------|--------------------|
| 1             | 2.778  | 2.778  | 2.778  | 0      | 5.714  | 5.714              |
| 2             | 0      | 2.941  | 11.765 | 5.882  | 8.571  | 17.143             |
| 3             | 5.714  | 5.714  | 5.714  | 11.429 | 5.714  | 8.571              |
| 4             | 2.857  | 2.857  | 5.429  | 2.857  | 2.857  | 14.706             |
| 5             | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 6             | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 7             | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 8             | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 9             | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 10            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 11            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 12            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 13            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 14            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 15            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 16            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 17            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 18            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 19            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 20            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 21            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 22            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 23            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 24            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 25            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 26            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 27            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 28            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 29            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 30            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 31            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 32            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 33            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 34            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 35            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 36            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 37            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 38            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 39            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| 40            | 2.857  | 2.857  | 11.429 | 2.857  | 2.857  | 14.706             |
| **Means**     | 6.866  | 6.497  | 7.244  | 7.676  | 8.261  | 8.638              |

Table 4: The repetition rates in different conditions in 120 simulations.
Figure 9: Different sizes of target environment and process ways. (a) Smaller environment than training size. (b) Wider environment than training size. (c) The sliding views for each robot.

Figure 10: A set of results of different environmental sizes. (a) Smaller environment than training size. (b) Normal size environment. (c) Wider environment than training size.
4. Comparison

As far as the authors know, under the similar model in this paper, there is no other MCPP method in unknown environment. Since there are two methods in known environment that are selected to contrast, the Glasius bioinspired neural network (GBNN) [18] method enables the robots to select actions by transferring activity among the grids. The parameters set in the simulation are consistent with those in that article: $\beta = 0.6$, $\alpha = 2$, $E = 100$, $c = 0.5$, and $R = \sqrt{2}$. The cell potential and motion pattern driven coverage (CPMPC) [17] method uses the heuristic algorithm to select a pattern and directs robots motion. The crossover rate (65%) and mutation rate (5%) parameters in CPMPC are the same in that article. Since the environment map is smaller than environment in that article, the population size (25) and maximum number of iterations (100) are reduced accordingly. Two kinds of environments are considered. One is the artificial partition environments, which simulates different rooms in the space; the other is the generated random environments to simulate an outdoor situations. In random environments, each grid has a 5% probability of being an obstacle grid, and the robot starting points are randomly generated in the nonobstacle grids. For different numbers of robots, 10 sets of multipartition environments and 40 sets of random environments are generated. It should be ensured that the initial conditions for each method of each set of experiments are the same. Because the simulation takes place in a large number of environments, showing all environments in figures is difficult. These environments are similar to those shown in Figure 10. Results are shown in Table 5.

The results verify that although the proposed method does not yield the best results in all cases, the results are stable and acceptable. A more important fact is that this method can work in unknown environment and complete the plan in a short time. Thus, the proposed method is more applicable in practical task. The repetition rates are 10% bigger than Table 4 because multitrobot stops exploring until all the grids are covered except obstacles. This means that when the last few remaining grids can be covered by one robot, other robots will still approach. In a small environment, this approach will increase the repetition rate significantly. GBNN is a simple and rapid method, which combines activity of grids and templates method. Templates dominate the coverage process, while activity of grids helps robots escape from dead zones. GBNN is faster than the other two methods as there are no complicated calculations in the whole process. Nevertheless, this simple principle is restricted in some environment. The results show the unstable state of GBNN. The performance of CPMPC is ideal in most situations with the help of the powerful search capabilities from GA. There is no gain without a loss. Application of heuristic algorithms means that it will cost a lot of time searching for the optimal solution with given environment. As a result, this method will not work in an urgent task. What is more, the number of selected patterns is set by users, and in that article, it was 2. In fact, the number of patterns should be set to a parameter and be changed with the complexity of the environments. In complicated artificial environments CPMPC may achieve wrong results with few selected patterns.

5. Conclusion

In this paper, taking into account large-scale and urgent tasks such as mine clearance and postdisaster relief, a grid-based MCPP method in unknown environment is proposed. DQN is used for planning in view of the great performance. However, the value network is hard to train accurately under the huge number of possible states. So the idea of deduction in MCTS is considered, and then the deduction method in MCPP is proposed. Modifying the deduction method not only reduces the time and cost of training but also completes the task assignment and avoids collision in MCPP. On the other hand, the explored grids are regarded as sampling from the whole environment to solve the problem about unknown environment. Since the real environment may be different in size from the training environment, the sliding-view method is proposed. Finally, this total method can achieve stable and decent results rapidly. The method in this article can be divided into two parts: training and decision-making, both of which require computation, so we need to balance the computational cost and algorithm performance. For example, how many episodes of training should be carried out, the number of action groups considered in decision-making, the larger the value of these parameters, the better the coverage performance of the robot, but the more time it takes to train or make decisions.

In partial comparisons, the proposed method performance in the repetition rate is a bit poor than in CPMPC; however, as an online method, the applicability is much stronger than other offline algorithms. Strong robustness is also a highlight of this approach. First, if the training environment is big enough, this method is valid in any case: the areas outside the real environment can be treated as obstacles. Second, this method is applicable to any environment, any number of robots, and any initial positions. Third, this frame may be suitable in other planning problems such as mine clearance and postdisaster relief.
as pathfinding problems. There are two points we are interested in and can be improved in the future. One is trying to use the actor-critic method to replace DQN and comparing the results. The other is reducing the repetition rate.

Data Availability

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by Sichuan Science and Technology Program (2021YZD0016) and Chengdu Science and Technology Program (2019-YF05-00958-SN).

References

[1] S. Thakar, R. K. Malhan, P. M. Bhatt, and S. K. Gupta, “Area-coverage planning for spray-based surface disinfection with a mobile manipulator,” Robotics and Autonomous Systems, vol. 147, Article ID 103920, 2022.

[2] H. H. Viet, V.-H. Dang, M. N. U. Laskar, and T. Chung, “Ba: an online complete coverage algorithm for cleaning robots,” Applied Intelligence, vol. 39, no. 2, pp. 217–235, 2013.

[3] J. Liu, T. Zhao, and S. Dian, “General type-2 fuzzy sliding mode control for motion balance adjusting of power-line inspection robot,” Soft Computing, vol. 25, no. 2, pp. 1033–1047, 2020.

[4] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” Robotics and Autonomous Systems, vol. 61, no. 12, pp. 1258–1276, 2013.

[5] T. H. Pham, Y. Bestaoui, and S. Mammar, “Aerial robot coverage path planning approach with concave obstacles in precision agriculture,” Workshop on Research, vol. 43, 2018.

[6] H. Chooser, “Coverage Path Planning: The Boustrophedon Cellular Decomposition,” Springer, London, UK, 1998.

[7] H. Chooser, E. Acar, A. A. Rizzi, and J. Luntz, “Exact cellular decompositions in terms of critical points of Morse functions,” in Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, April 2002.

[8] Y. Gabriely and E. Rimon, “Spiral-stc: an on-line coverage algorithm of grid environments by a mobile robot,” in Proceedings of the IEEE International Conference on Robotics Automation, pp. 954–960, Washington, DC, USA, May 2002.

[9] A. Zelinsky, R. A. Jarvis, and J. C. Byrne, Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot, Proceedings of International Conference on Advanced Robotics, vol. 13, 1993.

[10] H. H. Viet, V.-H. Dang, S. Choi, and T. C. Chung, “Bob: an online coverage approach for multi-robot systems,” Applied Intelligence, vol. 42, no. 2, pp. 157–173, 2015.

[11] S. SaeedVAnd, H. S. Aghdasi, and J. Baltes, “Robust multi-objective multi-humanoid robots task allocation based on novel hybrid metaheuristic algorithm,” Applied Intelligence, vol. 49, no. 12, pp. 4097–4127, 2019.

[12] A. Torreño, E. Onaindia, and O. Sapena, “Fmap: distributed cooperative multi-agent planning,” Applied Intelligence, vol. 41, no. 2, pp. 606–626, 2014.

[13] A. C. Kapoutsis, S. A. Chatzichristofis, and E. B. Kosmatopoulos, “Darp: divide areas algorithm for optimal multi-robot coverage path planning,” Journal of Intelligent and Robotic Systems, vol. 86, no. 3-4, pp. 663–680, 2017.

[14] X. Miao, H. S. Lee, and B. Y. Kang, “Multi-cleaning robots using cleaning distribution method based on map decomposition in large environments,” IEEE Access, vol. 8, 2020.

[15] Y. Chen, T. Zhao, J. Peng, and Y. Mao, “Fuzzy fraction-order stochastic parallel gradient descent approach for efficient fiber coupling,” Optical Engineering, vol. 61, no. 1, pp. 1–17, 2022.

[16] M. Kapanoglu, M. Alkalf, M. Ozkan, A. Yazici, and O. Parlaktuna, “A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time,” Journal of Intelligent Manufacturing, vol. 23, no. 4, pp. 1035–1045, 2012.

[17] M. Xu, B. Xin, L. Dou, and G. Gao, “A cell potential and motion pattern driven multi-robot coverage path planning algorithm,” in Proceedings of the Bio-Inspired Computing: Theories and Applications, vol. 1159, Zhengzhou, China, April 2020.

[18] B. Sun, D. Zhu, C. Tian, and C. Luo, “Complete coverage autonomous underwater vehicles path planning based on glasius bio-inspired neural network algorithm for discrete and centralized programming,” IEEE Transactions on Cognitive and Developmental Systems, vol. 11, no. 1, pp. 73–84, 2019.

[19] L. L. Wang and L. X. Pan, “Research on sbmpc algorithm for path planning of rescue and detection robot,” Discrete Dynamics in Nature and Society, 2020, no. 10, Article ID 7821942, 11 pages, 2020.

[20] J. D. Nicoud and M. K. Habib, “The pemex-b autonomous demining robot: perception and navigation strategies,” in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots Systems 95 Human Robot Interaction Cooperative Robots, vol. 1, pp. 419–424, Pittsburgh, PA, USA, August 1995.

[21] I. Rekleitis, A. P. New, E. S. Rankin, and H. Chooser, “Efficient boustrophedon multi-robot coverage: an algorithmic approach,” Annals of Mathematics and Artificial Intelligence, vol. 52, no. 2-4, pp. 109–142, 2009.

[22] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, “A dynamic path planning approach for multi-robot sensor-based coverage considering energy constraints,” IEEE Transactions on Cybernetics, vol. 44, no. 3, pp. 305–314, 2014.

[23] X. Kan, H. Teng, and K. Karydis, “Multi-robot field exploration in hex-decomposed environments for dubins vehicles,” in Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 449–455, Dali, China, December 2019.

[24] X. Gong, B. Ding, J. Xu, H. Wang, and D. Feng, “Parallelized synchronous multi-agent deep reinforcement learning with experience replay memory,” in Proceedings of the 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp. 325–330, San Francisco, CA, USA, April 2019.

[25] A. Krishna Lakshmanan, R. Elara Mohan, B. Ramalingam et al., “Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot,” Automation in Construction, vol. 112, Article ID 103078, 2020.
[26] V. Mnih, K. Kavukcuoglu, D. Silver et al., “Playing atari with deep reinforcement learning,” Computer Science, vol. 21, pp. 351–362, 2013.

[27] V. Mnih, K. Kavukcuoglu, D. Silver et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[28] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 2094–2100, Phoenix, AZ, USA, February 2016.

[29] T. Zhao and Z. Wei, “On characterization of rough type-2 fuzzy sets,” Mathematical Problems in Engineering, vol. 2016, Article ID 4819353, 13 pages, 2016.

[30] D. Silver, A. Huang, C. J. Maddison et al., “Mastering the game of go with deep neural networks and tree search,” Nature (London), vol. 529, no. 7587, pp. 484–489, 2016.

[31] R. Coulom, “Efficient selectivity and backup operators in monte-carlo tree search,” in Proceedings of the Computers and Games, vol. 4630, pp. 72–83, Springer, Berlin Germany, May 2007.