Reinforcement Learning for Multi-robot Field Coverage Based on Local Observation

Matthew Zhu, Dennis Simon, Nachiketa Rajpurohit, Sagar Jayantkumar Kalathia, and Wencen Wu*

***Abstract*—Field coverage is a representative exploration task that has many applications ranging from household chores to navigating harsh and dangerous environments. Autonomous mobile robots are widely considered and used in such tasks due to many advantages. In particular, a collaborative multi-robot group can increase the efficiency of field coverage. In this paper, we investigate the field coverage problem using a group of collaborative robots. In practical scenarios, the model of a field is usually unavailable and the robots only have access to local information obtained from their on-board sensors. Therefore, a Q-learning algorithm is developed with the joint state space being the discretized local observation areas of the robots to reduce the computational cost. We conduct simulations to verify the algorithm and compare the performance in different settings.**

I. Introduction

The exploration of an area is an age-old challenge. This topic has a wide range of applications in military as well as civilian domains. Autonomous robots are becoming more prevalent in the modern world with the advancements to hardware and software capabilities. They provide opportunities varying from accomplishing simple household chores to navigating harsh and dangerous environments without requiring humans to intervene. The introduction of autonomous robots has made the exploration decidedly more efficient and applicable in more use cases. However, introducing autonomy increases the complexity of the algorithms necessary to accomplish this task. Adding collaboration among multiple robots further increases the complexity of the system. The multi-robot collaboration has many potential benefits including increasing the efficiency of exploration by sharing information among the robots and being more robust and adaptive to the changing environment. However, it comes with many challenges in dealing with the interaction between the robots so that they can collaborate to accomplish a common goal [1], [2].

Complete area coverage using an individual autonomous robot or multiple collaborative robots is a typical exploration [3]. This task comes bundled with obstacle detection, path planning, and communication in a fault-tolerant manner. To compound these problems, any solutions must be dynamic enough to adapt to changing variables such as the number of robots, position and count of obstacles, and so forth. A robust algorithm to solve this problem enables us to apply it in various diverse scenarios. Some of the notable applications are in the field of mapping the spread of hazard environments where humans are at increasing risk to enter, unmanned search and rescue in areas affected by natural disasters, space exploration, and autonomous driving.

There are many algorithms that have been developed and implemented to solve the field coverage problem. Some approaches use a single robot. For example, the approach in [4] builds the coverage of a set (square subsection) using the existing Boustrophedon path, allowing coverage of areas with arbitrary shape. The method of [5] takes on the coverage problem from the perspective of a vacuuming robot. Their algorithm builds a topology of the map by identifying landmarks - walls and corners - and building a fully connected graph of the landmarks. In [6], a spanning tree-based based approach is developed, which has a computationally complexity of $O(n \log n)$, with complete coverage and no overlapping. A simultaneous localization and mapping (SLAM) - based approach is identified in [7], requiring the agent to have multiple on-board sensors to create an environment of its surroundings. [8] explores possible coverage methods on a general basis, including direct coverage and environmental decomposition.

The approaches using a single robot provide a baseline for the coverage control problem. The multiple robot approaches focus on increasing efficiency, robustness, and adaptability that a single robot does not. Mapping and exploration by multiple robots involves task division and distribution, collaboration, inter-agent communication, and fault tolerance in the event of failure or malfunction of any number of participating agents. In addition to that, the problem becomes even more pronounced when the area of interest is unknown or only partially known [8][9]. In [10], a solution is developed via applying the principle of spreading a wave where each robot communicates only with its direct neighbors. Whereas, the algorithm proposed in [11] creates an ad-hoc network among the robots to address inter-robot communication.

Many traditional approaches to solve the multi-robot coverage problem rely on knowing the model of the environment. Some of the newer research have focused on utilizing reinforcement learning (RL) techniques to solve this problem since an environmental model is usually not available [12], [9], [13]. In RL, a certain goal is set by specifying a well-chosen reward function. The RL agents then learn to maximize the cumulative reward received over time in order to reach the goal. In the case of coverage problem, one goal is to maximize coverage area with minimum actuator energy of individual robot.

In this paper, we address the field coverage problem by deploying a centralized autonomous multi-robot system and utilizing reinforcement learning techniques. The robots must interact with each other as well as the environment and move to minimize the amount of overlap with each other in their coverage of the area while still covering the entire mapped field in a minimum time. A reward function penalizing repeatedly

The research work is supported by NSF grant CMMI-1917300.

M. Zhu, D. Simon, N. Rajpurohit, S. J. Kalathia, and W. Wu are with the Computer Engineering Department of San Jose State University, San Jose, CA 95192 USA. {matthew.zhu, dennis.simon, nachiketa.rajpurohit, sagarjayantkumar.kalathia, wencen.wu}@sjsu.edu.

978-1-7281-8050-2/20/$31.00 ©2020 IEEE

entering into already covered areas is designed. Q-learning is utilized to generate the optimal policy. A well-known problem in Q-learning is the computational complexity with a large amount of state-action pairs in the Q-table as the number of state / actions increases. In addition, in practice, it is unlikely that the real-world robots always have full information about the area it intends to cover. Instead, the robots sense the world using on-board sensors with limited sensing ranges. To tackle the problems, we propose a observation-based state space design instead of using the intuitive location-based state space design in the Q-learning. The observation acts as a robot's perception of its immediate surroundings, thus, the state space depends on the observation radius of the robots. To verify the proposed algorithm, we simulate an environment with the presence of obstacles and performed the RL using several robots. The contributions of this paper include (1) the formulation of a reward function that is suitable for the multi-robot field coverage problem, and (2) the design of the observation-based state used in Q-learning.

The rest of the paper is organized as follows. Section II presents the preliminary knowledge of RL. Section III formulates the multi-robot field coverage problem and introduces the proposed observation-based Q-learning algorithm. Simulation results and analysis are discussed in Section IV and Section V provides some discussions on the results. Finally, Section VI concludes our paper and presents the future work.

II. Preliminaries

In RL, agents will learn to explore the environment using the observations/perception of their surroundings and can learn optimal actions to take. By interacting with the environment and other agents directly they will receive rewards based on how optimal their actions are. At a given time step t, each agent will have a state $s \in S$ and pick an action $a \in A$ to perform within the environment based on its policy π. Based on the effect on the environment the agent will receive a reward R and jumps to the next state s'. The combination of short term reward and long term rewards are factored into the return which is the sum of discounted rewards after time t. The goal of reinforcement learning is to find the optimal policy which maximizes the expected return. The action-value function, also known as the Q-function, explains how good it is for an agent to perform an action a in given state s at time t while following policy. The Q-function output is called the Q-Value and represents the quality of taking an action at a given state.

Q-learning aims to find the optimal policy such that the expected return over all successive steps is maximized. Thus it attempts to find the optimal Q-value for each state-action pair. The optimal policy from Q-learning is known as the optimal Q-function which is denoted as Q_* and gives the maximum expected return by any policy at each state-action pair. The state-action pair Q-values are iteratively updated using the Bellman equation until the Q-function converges to the optimal Q-function. The Bellman equation is

$$Q_*(s,a) = E[R_{t+1} + \gamma max Q_*(s',a')] \tag{1}$$

Q-values for a state-action pair will be updated each time an action is taken in a specific state. The update law is given by

$$Q(s,a)^{new} = (1-\alpha)Q(s,a)^{old} + \alpha(R_{t+1} + \gamma max_{a'} Q(s',a')) \tag{2}$$

where α is the learning rate and the discount factor γ pertains to the consideration of choosing long term v.s. short term reward in Q-value computation. A higher γ value will make the long term reward (max reward at the next state) have a higher factor in Q-value computation. Q-values are stored in a table called the Q-table, whose dimensions are dependent on the set of possible actions A of an agent and the set of states S. Thus the Q-table dimensions are $n \times m$ with n representing the number of possible agent actions and m representing the total number of states.

III. Reinforcement Learning Based Multi-robot Field Coverage

Autonomous mobile robots are able to navigate to a destination based on their on-board sensors. There are a variety of dangerous situations where autonomous robots can be applied to avoid potential harm to humans. From wildfires to toxic fumes, autonomous robots can traverse these environments to provide valuable data on conditions in various areas. Multi-robot coverage is a complex topic with many possible solutions. It contains problems such as minimizing robot overlap in coverage, minimizing time spent to explore an environment, finding optimal paths of each robot, avoiding obstacles, adjusting to environmental changes, etc. In this section, we formulate the problem in the RL setting and introduce the proposed observation-based Q-learning algorithm.

A. Problem Formulation

Consider a field $\mathcal{F}(r)$ of arbitrary shape with the presence of obstacles, where r denotes the location. In the mission of mapping the field or surveillance, a group of N collaborating robots equipped with sensors are sent to the field to provide field coverage. Let r_i denote the position of the ith robot moving in the field where $i = \{1, ..., N\}$. Each robot has a limited sensing radius denoted by δ_i and a corresponding observation area $\Omega_i = \{r : d_i(r) \leq \delta_i\}$ where $d_i(r) = ||r - r_i||$. In the RL setting, the field is discretized into a grid of $M \times M$ cells. The sensing radius of a robot is then defined as the number of cells out from the robot, not including the robot itself. So, a radius of δ would result in a total observation area of $(2\delta+1)^2$ cells with the robot in the center.

The multi-robot learning system can be described using $< N, \{S\}, \{A\}, R >$, where:

- N is the number of robots in the system.
- $\{S\} = S_1 \times S_2 \times \cdots \times S_N$ is the joint state space, where S_i is the individual state space of the ith robot. At time step k, the state of the ith robot is denoted as $s_{i,k}$.
- $\{A\} = A_1 \times A_2 \times \cdots \times A_N$ is the joint action space, where A_i is the individual action space of the ith robot. At time step k, the action of the ith robot takes is denoted as $a_{i,k}$.

- $R : S \times A \longrightarrow \mathcal{R}$ is the individual reward function that specifies the immediate reward the ith robot receives by taking action $a_{i,k}$ at state $s_{i,k}$ and reaches state $s_{i,k+1}$.

In the multi-robot field coverage problem, the goal of the robots is to find the optimal policy that maximizes the expected return in the long run. The reward should penalize the overlapping of the observation areas of different robots as well as frequently revisiting the explored areas, so as to minimize the number of steps taken to cover the entire field by the multi-robot system.

B. The Observation-based Q-Learning Algorithm

In this section, we design an observation-based Q-learning algorithm to solve the problem formulated in Section III-A. We assume that the robots are identical with the same state space and action space, i.e., $S_1 = S_2 = \cdots = S_N$ and $A_1 = A_2 = \cdots = A_N$. The components of the algorithm are described as follows.

1) Environment: The discretized $M \times M$ grid field is the environment that the multi-robot system is interacting with. Each grid cell in the environment is designated as one of the types in the set E =$\{$covered, uncovered, obstacle$\}$, where "covered" means the cell has been explored by a robot and "uncovered" means the cell has not been explored. The field is initialized with surrounding obstacles, designating the edge of the environment. The initial locations of the robots can be randomly assigned to uncovered cells.

2) Action space: Since we assume the identical action space for all the robots, we drop the subscript i in the notations from now on. The action space of each robot can be designed as A= $\{$up, down, left, right$\}$, restricting movement across the grid to one space in each cardinal direction at each step. The action space can be readily expanded to eight directions.

3) State space: In many existing RL approaches, the grid cells constitute the state space of a single robot if we only consider the position of the robot. One challenge is that each robot updates its Q-table w.r.t. other robots' state and action, which means that the state-action pairs in the joint Q-table can grow exponentially if we increase the grid size and the number of collaborating robots. We refer to this state space as "location-based" state. Another challenge is that in practice, it is more likely that a single robot does not have the global knowledge of the environment. Therefore, we assume that each robot only has partial information of the environment. By limiting the information given to a robot to only its local surroundings, we can abstract perception of a real-world autonomous robot.

To deal with the above problems, we propose an observation-based state space design. Along with its location, each robot retains an $observation$ that identifies its local surroundings. This observation, whose radius is adjustable, contains information about the designation of the cells that are within the robot's observation area. In the observation-based state space design, each cell has two additional types: "robot" indicating the cell is being occupied by another robot and "out-of-bounds" indicating the cell is out of the boundary of the current environment. Thus, each cell has a type from (robot, obstacle, covered, uncovered, out-of-bounds) that is encoded as an index. The $observation$ is a 2D matrix centered at the current position of the robot with each cell indexed as a type. The individual robot's state, used for our Q-learning algorithm, is built by flattening this 2D matrix into a single, comma-delimited string.

Fig. 1 illustrates a sample 12×12 grid field with $N = 4$ robots represented by blue circles and 15 obstacles represented by grey cells. Each robot has an observation radius $\delta = 2$ so the light square area surrounding each robot indicates the observation area. Fig. 2 demonstrates the observation area of the robot in the center with radius $\delta = 2$, where green cells are "uncovered", white cells are "covered", grey cells are "obstacle", and the cell in the rightmost column containing a robot is "robot". This 5×5 observation matrix is used to construct the state space of the robot in the center.

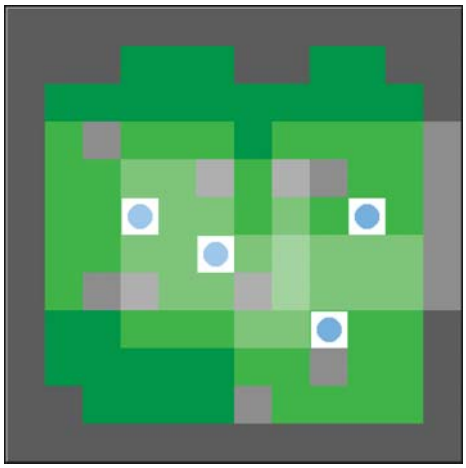

Fig. 1: A sample 12×12 grid field with $N = 4$ robots represented by blue circles and 15 obstacles represented by grey cells. Each robot has an observation radius $\delta = 2$.

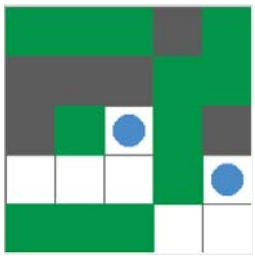

Fig. 2: The observation area of the robot in the center with different cell types. The observation radius $\delta = 2$.

In general, the state space of an individual robot consists of all the possible permutations of cell types within the robot's observation area. Therefore, the number of possible states of a single robot has an upper bound of $5^{(2\delta+1)^2-1}$, which is a function of the observation radius δ. In practice, the observation radius of a robot is usually fixed given certain types of sensors. Thus, the number of states won't grow as the size of the field to be explored increases. In other words, the design is scalable as the size of the field increases. For multiple robots, the joint state space consists of all individual robot's

state space. In Q-learning, the number of state-action pairs is $|S| \times |A|$. For the observation in Fig. 2, the maximum number of state-action pairs would be $|A| \times 5^{24}$. Though, realistically, the Q-table for a finite environment will not approach this maximum, as not every state permutation will be seen.

4) Reward: The reward function must sufficiently incentivize the robot to accomplish the goal of complete coverage of the grid in minimum steps. For this purpose, we propose the following reward function:

$$r_t = \begin{cases} +a, & \text{if uncovered cell} \\ -b, \times\{visited\_count\} & \text{otherwise,} \end{cases} \tag{3}$$

where $a > 0$ and $b > 0$ are two positive constants chosen by design. Based on the reward, a robot receives $+a$ for entering an uncovered cell. However, when visiting an already covered cell, or running into obstacles or other robots, the robot receives a reward of $-b$ multiplied by the number of times the cell has previously been visited. Therefore, the reward encourages exploring uncovered cells in minimum steps.

5) Exploration vs Exploitation: At any given step, we explore with probability ϵ_e and exploit with probability $1-\epsilon_e$, where $0 < \epsilon_e < 1$.

$$\pi(A_k) \leftarrow \begin{cases} \text{random joint action,} & \text{with probability } 1-\epsilon_e \\ \text{optimal joint action,} & \text{otherwise.} \end{cases} \tag{4}$$

By exploring, the robot chooses a random joint action a from the set of all available actions A. By exploiting, the robot chooses the learned optimal action $\text{argmax}_a Q(s,a)$. Initially, we aim for a high exploration rate, as we want to learn quickly. As the policy converges over time, we gradually transition from exploring to exploiting. This can be done through decaying ϵ_e by a decay parameter $decay^e \in (0,1)$

$$\epsilon_e^{new} = \epsilon_e^{old} * decay^e. \tag{5}$$

Based on the state space, action space, and reward function, the observation-based Q-learning algorithm for multi-robot field coverage is summarized in Algorithm 1.

The algorithm initially declares a default dictionary that sets accessed rows that are not present as a row with zeros of size (number of columns) equal to the number of actions possible for the agent. The training is divided into episodes. An episode is the number of steps required for complete coverage. The environment is reset at the end of each episode, but the updated Q-table is retained for the subsequent episodes. During reset, obstacles and the initial locations of the robots are seeded to preserve consistency through training. For each episode, the robots receive their initial observations in the environment. When a robot takes an action and transitions to the next state, it takes a new observation, which is again concatenated into a string and used as the new state.

Algorithm 1 Observation-based Multi-robot Q-Learning Algorithm

Require: $0 < \gamma, \alpha, \epsilon_e, decay^e < 1$
  $Q \leftarrow defaultdict$(lambda:zeros(shape=len(actions)) {Initialize Q-table that creates row w/ 0's when new state is seen}
  for x episodes **do**
    for each robot **do**
      $s \leftarrow$ concatenated observations {concatenate initial observations into state}
    end for
    while true **do**
      for each robot **do**
        if $rand(0,1) < \epsilon_e$ **then**
          $a \leftarrow$ random action
        else
          $a \leftarrow max_a Q[s,a]$ {action with highest reward at state s}
        end if
        $s', reward \leftarrow$ take a {next state = observation at new position}
        $next\_max \leftarrow max(Q[s',a])$
        $Q'[s,a] \leftarrow (1-\alpha)\ Q[s,a] + \alpha(reward + \gamma * next\_max)$
        $s \leftarrow s'$
      end for
      if $episode \leftarrow$ solved **then**
        $break$
      end if
    end while
    $\epsilon_e \leftarrow \epsilon_e * decay^e$
  end for

IV. Results and Analysis

In this section, we verify the proposed algorithm in simulations. The simulation environment utilizes OpenAI gym and is based on the implementation of gym-minigrid [14] with heavy changes to match the problem set. The environment is a 12×12 grid space with 15 obstacles and a wall indicating the boundary of the space. We choose the reward values $a = 10$ and $b = -1$ so a robot receives reward $+10$ when moving into uncovered cells and $-1 \times times\_visited$ when moving into cells that are uncovered, obstacle, and occupied by another robot. The learning rate $\alpha = 0.01$, the discount factor $\gamma = 0.6$, and $\epsilon_e = 0.8$.

To analyze the performance of the algorithm, we conduct a comparative analysis of various parameters including the number of robots N and the observation radius δ.

A. Comparison of the number of robots

We first simulate different number of robots in the environment for the field coverage. Fig. 3 illustrates the results of the number of steps to completion over training for the number of robots $N = 1$ to 5. Table I shows some significant values regarding the simulations. The performance as we increase the number of robots from one to three improves significantly. The maximum, minimum, and average steps for the simulation with three agents are about 3 times better than the results with only a single robot. As we increase the number of robots further, the minimum steps to completion only marginally improves.

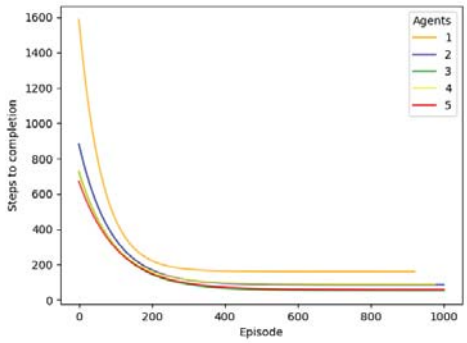

Fig. 3: Total steps to completion by episode for 1 to 5 robots.

# Robots	Max. steps	Min. steps	Avg. steps after ep. 500
1	3180	131	133.1
2	1644	55	72.6
3	1172	37	46.2
4	1364	33	79.6
5	1570	30	55.6

TABLE I: Comparisons of steps to completion for the number of robots $N = 1, \cdots, 5$.

The maximum and average steps actually increase, likely due to congestion for the given environment size.

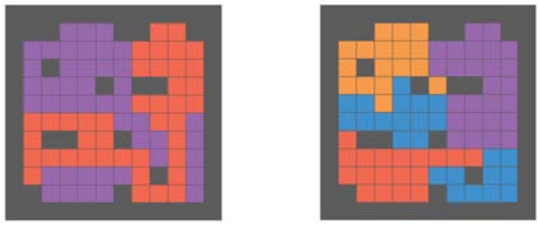

Fig. 4: Optimal solutions of the multi-robot field coverage problem in the 12×12 grid space with 2 robots (left) and 4 robots (right). Different colors indicate the areas covered by different robots.

Fig. 4 illustrates the optimal solutions of the multi-robot field coverage problem. The left figure shows the result of using 2 robots and the right figure shows the result of using 4 robots. Different colors represent the areas covered by different robots. In the right figure, there is a discontinuity of the blue area. This is because the same cells may be in the final coverage areas of more than one robots so the same cells are covered more than once in the optimal solution, yielding the covered areas of the earlier robots being discontinuous.

B. Comparison of observation radius

To investigate the effect of observation radius to the performance of the proposed algorithm, we conduct simulations using observation radius $\delta = 1, 2, 3$. Other parameters are $N = 3$ and $decay^e = 0.99$. In Fig. 5, the results of steps to completion v.s. different number of episodes are shown.

The most prominent finding is the divergence of the data for an observation radius of 1. The reason is that a small observation radius lacks the necessary number of states to

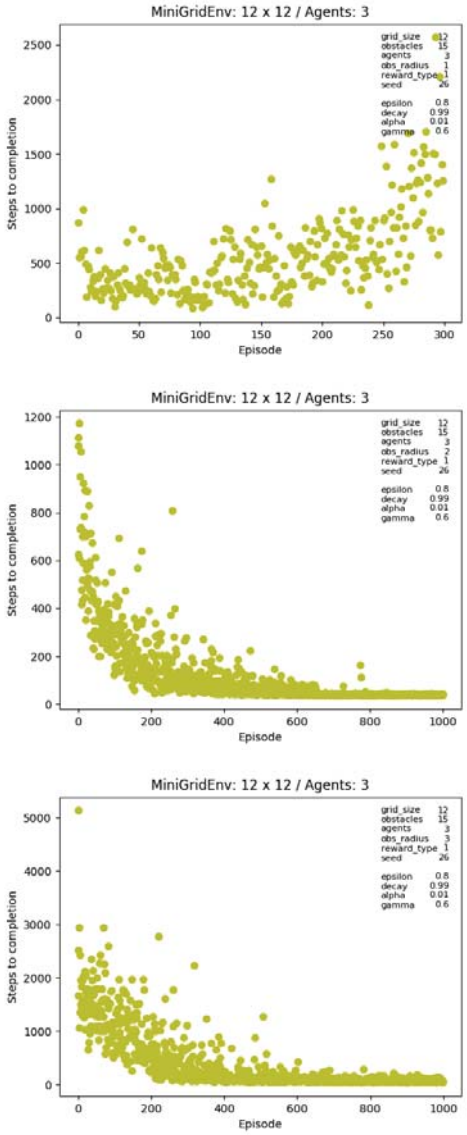

Fig. 5: Total steps to completion by episode for observation radius of 1 to 3

Observation radius	Max. steps	Min. steps	Avg. steps after ep. 500
2	1172	37	46.2
3	5134	42	95.23

TABLE II: Steps to completion for observation radius $\delta = 2, 3$.

encapsulate the environment grid for the robot to learn. For observation radii of 2 and 3, the steps to completion converge. The significant values of this data are listed in Table II.

With every metric, an observation radius of 2 performs better than the observation radius of 3. The performance for different radii depends on the given environment and hyperparameters, maintaining a balance between enough states to properly learn and not too many states that slows the propagation of Q-values. In the case of a radius of 1, the divergence is due to an insufficient number of states. In the

case of a radius of 3, the final Q-table has significantly more states than that of radius 2, which slows the learning of the algorithm. Thus, given this environment, an observation radius of 2 is the optimal among the three. Fig. 6 illustrates the optimal solutions using 3 robots with observation radius $\delta = 2$ and $\delta = 3$.

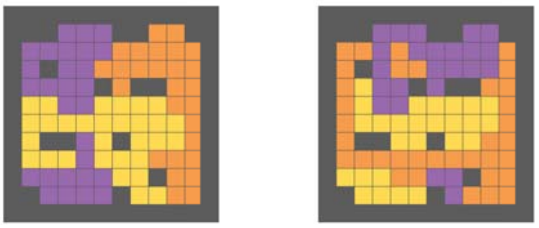

Fig. 6: Optimal solutions of the multi-robot field coverage problem in the 12×12 grid space using 3 robots with $\delta = 2$ (left) and $\delta = 3$ (right). Different colors indicate the areas covered by different robots.

V. Discussions

1) Q-table states: While many existing solutions in Q-learning employ the location-based states, i.e., each cell of the overall grid space is considered as a state and the state space depends on the number of cells in the environment grid, we propose to use the observation-based state in the Q-learning algorithm for several reasons. First, the size of the observation-based state space depends on the observation radius of the robots, which is usually fixed at certain applications, thus, the number of state-action pairs won't grow as the size of the environment grid increases. Second, observation-based states lead to more accurate results. In positional-based states, after a grid cell has been covered, the reward shifts from positive to negative, which leads to inaccurate Q-values when the state is occurred again. Observation-based states, on the other hand, are unique to what the robot sees around it; thus a particular grid changing from covered to uncovered will lead to a completely different state, which allows for consistent reward allocation. The other reason is that location-based states allow for the Q-table to have a complete view of the entire environment, which is unrealistic. In a real world scenario, the size and constraints of the environment are usually unknown to the robots as they are exploring it. Only when exploring the environment with perception should the environment be mapped out. Observation-based states allow for a real world formulation of the problem because robots learn and take actions only based on the surrounding elements they can perceive.

2) Environment variables: The choice of the number of robots and the observation radius is dependent on the given environment. A small environment with too many robots may have congestion so the field coverage results, i.e., the number of steps to completions, may only be marginally improved at an increased cost. On the contrary, too few robots in a large environment may result in slow learning and reduces convergence speed. A small observation radius may lack the necessary number of states to encapsulate the environment grid for the agent to learn. This leads to divergence, which can be seen in the first figure in Fig. 5 for an observation radius of 1. A large observation radius may create an excess number of states, causing slow propagation of Q-values across the Q-table. The performance cost increases exponentially for larger radii, but may be necessary for large environment grids.

VI. Conclusions and Future Work

In this work, we developed an observation-based Q-learning algorithm to solve the field coverage problem using a group of collaborating autonomous mobile robots. The observation-based Q-learning algorithm can effectively reduce the size of Q-table when the environment grid space increases, and is more realistic to practical constraints such as limited observation range and local information. We conducted simulations with different number of robots and different observation radii to verify and analyze the performance of the algorithm. There are various possible directions for future work including taking into consideration of more constraints in the algorithm such as kinematic / dynamic constraints of the robots, and exploring deep RL technique to further improve the algorithm.

References

[1] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[2] S. Thrun *et al.*, "Robotic mapping: A survey," *Exploring artificial intelligence in the new millennium*, vol. 1, no. 1-35, p. 1, 2002.

[3] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[4] T. Bretl and S. Hutchinson, "Robust coverage by a mobile robot of a planar workspace," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4582–4587.

[5] H. Zhang, W. Wang *et al.*, "A topological area coverage algorithm for indoor vacuuming robot," in *2007 IEEE International Conference on Automation and Logistics*. IEEE, 2007, pp. 2645–2649.

[6] G. E. Jan, C. Luo, L.-P. Hung, and S.-T. Shih, "A computationally efficient complete area coverage algorithm for intelligent mobile robot navigation," in *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2014, pp. 961–966.

[7] J. Song and S. Gupta, "Slam based shape adaptive coverage control using autonomous vehicles," in *2015 10th System of Systems Engineering Conference (SoSE)*. IEEE, 2015, pp. 268–273.

[8] Y. Kang and D. Shi, "A research on area coverage algorithm for robotics," in *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*. IEEE, 2018, pp. 6–13.

[9] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," *arXiv preprint arXiv:1803.07250*, 2018.

[10] Z. Laouici, M. Mami, and M. F. Khelfi, "Cooperative approach for an optimal area coverage and connectivity in multi-robot systems," in *2015 International Conference on Advanced Robotics (ICAR)*. IEEE, 2015, pp. 176–181.

[11] J. Tan, O. M. Lozano, N. Xi, and W. Sheng, "Multiple vehicle systems for sensor network area coverage," in *Fifth World Congress on Intelligent Control and Automation (IEEE Cat. No. 04EX788)*, vol. 5. IEEE, 2004, pp. 4666–4670.

[12] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[13] A. A. Adepegba, S. Miah, and D. Spinello, "Multi-agent area coverage control using reinforcement learning," in *The Twenty-Ninth International Flairs Conference*, 2016.

[14] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for openai gym," https://github.com/maximecb/gym-minigrid, 2018.