Received: 27 May 2020 | Revised: 29 September 2020 | Accepted: 14 October 2020

DOI: 10.1002/int.22331

RESEARCH ARTICLE

WILEY

A reinforcement learning-based approach for modeling and coverage of an unknown field using a team of autonomous ground vehicles

Saba Faryadi | Javad Mohammadpour Velni

School of Electrical & Computer Engineering, University of Georgia, Athens, Georgia, USA

Correspondence
Javad Mohammadpour Velni, School of Electrical & Computer Engineering, University of Georgia, Athens, GA 30602, USA.
Email: javadm@uga.edu

Funding information
US National Science Foundation, Grant/Award Number: 1934481; National Institute of Food and Agriculture, Grant/Award Number: 2017-67021-25928

Abstract

Precision maps are useful in agricultural farm management for providing farmers (and field researchers) with locational information. Having an environmental model that includes geo-referenced data would facilitate the deployment of multi-robot systems, that has emerged in precision agriculture. It further allows developing automated techniques and tools for map reconstruction and field coverage for farming purposes. In this study, a reinforcement learning-based method (and in particular dyna-Q+) is presented for a team of unmanned ground vehicles (UGVs) to cooperatively learn an unknown dynamic field (and in particular, an agricultural field). The problem we address here is to deploy UGVs to map plant rows, find obstacles, whose locations are not known a priori, and define regions of interest in the field (e.g., areas with high water stress). Once an environment model is built, the UGVs are then distributed to provide full coverage of plants and update the reconstructed map simultaneously. Simulation results are finally presented to demonstrate that the proposed method for simultaneous learning and planning can successfully learn a model of the field and monitor the coverage area.

KEYWORDS
coverage control, multiagent systems, reinforcement learning

1 | INTRODUCTION

With the world population growing, global food demand has been on the rise. The need for a sustainable food supply has led to the integration of modern computing technology to facilitate management and monitoring of both outdoor (such as farms) and indoor (such as greenhouses) agricultural environments. Furthermore, penetration of global positioning system (GPS), advanced sensing technologies, unmanned vehicles, and artificial intelligence (AI) in agricultural environments has led to a breakthrough in smart farming and enabled automating farm operations at different levels.

Machine learning (ML) integrated with big data technologies have emerged in precision agriculture for machines to understand and quantify data and learn from experiences automatically while they are not explicitly programmed.[1] One of the exciting areas in ML is reinforcement learning (RL), which has been developed to train the machine for making a sequence of decisions and finding the optimal solution by learning from past experiences. In RL, a machine (a.k.a. agent) learns by itself based on the rewards and payoffs obtained through interacting with the environment with the eventual goal of maximizing the cumulative reward.[2]

Reinforcement learning, and in general AI, have been widely employed in precision agriculture for various purposes including yield prediction/quantification, weed detection, and crop quality assessment. In addition to crop and livestock management, precision agriculture has also benefited from machine learning techniques by employing them for multi-robot (multiagent) systems deployment in farms. In using multiagent reinforcement learning, there are two main objectives: learning and control (decision making), where robots have to coordinate their actions and accelerate the learning process cooperatively.[3] A difficulty arises due to the inconsistency in farm terrain and planting patterns throughout the growth cycle, which increases the complexity of controlling mobile robots.[4] To address this issue, few studies focused on developing learning-based methods aiming at crop rows identification and visually guided navigation.[5,6] Furthermore, authors in references [4,7] proposed a genetic algorithm (GA)-based method as a self-learning system for both calibration of robot controller and path following activities in different environmental conditions. Although AI applications to control unmanned vehicles in an unknown field have received attention in the past few years, there are very limited work on field map reconstruction using RL-based methods. Precision maps are essential for positioning and localizing multi-robot systems in agricultural fields, and hence, several steps are needed first to provide locational data before deploying robots in the farm. A common practice to do this is through deploying drones over the farm to collect aerial images and then analyze them and build the farm's map. Moreover, at different stages of plant growth, agricultural field changes, and hence it is not a static environment; therefore, there is a need for an automated system capable of updating field models in real time, which we address in this paper.

During the last few years, the deployment of multi-robot systems in agricultural environments (both indoor and field) has received significant attention and is rapidly becoming reality even for large farms because of modern technologies and advancements in artificial intelligence (see references [8–10] and references therein). In the very near future, a large number of farm workers will receive significant assistance from automated ground vehicles and drones.[11] There are also a number of studies focusing on multi-robot-based distributed coverage, path planning, and automated navigation for agricultural applications.[12–14] Although such collaboration can improve farm management and crop monitoring, the underlying systems are not fully autonomous due to the lack of a detailed map of the farm containing information of the plant rows and important areas. Hence, it is necessary to develop a system of crops-land mapping for employing mobile robots in smart farming. Weiss et al.[15] employed MEMS-based 3D LiDAR sensors to detect and segment plants and ground with the purpose of automating localization, mapping, and navigation for farm robots. As described in reference [16], the major limitation of vision-based methods is the sensitivity to ambient lighting conditions. To address this problem, Hiremath et al. developed a particle filter-based (PF) algorithm for autonomous navigation in a maize field.[16]

Besides visual-based approaches, recently, the use of RL-based techniques (and in particular, Q-learning) has been proposed in autonomous robotic systems, due to its self-training ability with no prior knowledge about the environment.[17] The application of RL and deep learning methods to train agents navigating autonomously is fundamental to their intelligent behavior. Konar et al.[18] developed a Q-learning based approach for multi-robot path planning application. In comparison with the classical Q-learning, the proposed method in reference [18] is more efficient in terms of the total traveling time, number of visited states, and 90° turns required. In [19], a mapless motion planning method has been developed based on an asynchronous deep reinforcement learning method without any predefined features.

There have been several studies in the literature on applying RL to solve the coverage problem for multiple unmanned aerial vehicles (UAVs) in an unknown environment. In reference [20], authors proposed a nonoptimal solution to the problem of free dynamic area coverage using reinforcement learning. Pham et al.[21] developed a multiagent reinforcement learning (MARL) algorithm to enforce UAVs to learn cooperatively for fully covering an unknown field with the purpose of minimizing the overlaps among their fields of view.

Most of the algorithms developed for aerial vehicles are not often useful for ground robots due to their limitations and differences. Authors in reference [22] established an area coverage control law using RL methods by training multiple autonomous agents and showed that the control law would asymptotically converge to an optimal configuration. In reference [23], authors proposed a method to combine deep Q-learning with convolutional neural networks (CNNs) to analyze the exact situation using image data on its environment and guided the robots to navigate based on the results extracted from the deep Q-learning based analysis.

One of the interesting challenges in mapless navigation in the field is how to cope with a dynamic environment that changes frequently. Authors in reference [24] presented a navigation approach in a complex environment using reinforcement learning in such a way that data efficiency and task performance are considerably improved. Yue et al.[25] employed reinforcement learning for multi-UAV sea area search map considering models of the environment, UAV dynamics, dynamics of goals, and sensor detection.

The main contribution of this study is *in filling the gap between mapless navigation in dynamic environments and coverage control using multiagent systems*. To achieve that, we first adopt a model-based RL (namely, Dyna-Q+) approach for enforcing ground robots to navigate in a dynamic environment and building the precision map (environmental model) assuming no prior knowledge about the environment; then, we design a distributed control algorithm for the coverage of the dynamic field using ground vehicles. It is noted that there exist a very limited body of work that uses the field's map for discrete partitioning and coverage control of multi-robot systems in a farm represented by a weighted graph. However, unlike our proposed method, past works assumed that the entire field is known, and before solving the locational optimization problem, a graph-based topological map must be available for representing the area. Building a precision map requires extra efforts and cost, and this was our prime motivation to develop an automated method for multi-robot systems enforcing them to learn the environment with no initial action and knowledge.

The remainder of this paper is organized as follows. Section 2 describes the problem statement and methodology. Validation of the proposed method in a simulated agricultural environment will be discussed in Section 3, followed by concluding remarks in Section 4.

2 | PROBLEM STATEMENT AND PROPOSED SOLUTION METHOD

In this article, we first employ a model-driven reinforcement learning (RL)-based method for deploying multiple unmanned ground vehicles (UGVs) in an unknown agricultural field to cooperatively build a model (graph) of the field including plants, obstacles, and important areas (goals) and update the model in real time while it changes. After the model is created (or updated in real time), the field graph is then generated for solving the problem of optimally distributing the team of UGVs in the coverage area.

2.1 | Environmental model learning based on Dyna-Q+ algorithm

Most of the RL algorithms are model-free methods and applicable to problems where the environment's model is not available. Dyna-Q is a model-based RL algorithm that combines Q-learning and Q-planning, in which the planning is done online while the agents interact with (and hence explore) the environment to learn its model.[2] In many applications, the main challenge in using Q-learning is its poor performance under unknown and dynamic environments, that is, its lack of capability to update the environment model while it changes during the learning process. To address this problem, Dyna-Q+ method was developed as an improved Q-learning algorithm to accelerate the training process under unpredictable conditions.[2] To apply RL for building a map of the field, let us consider a group of M ground vehicles moving at different speeds. The vehicles that are collectively responsible for exploring the environment are equipped with sensors and/or cameras for monitoring the plants growth and phenotypic traits. Considering sensors' footprint and the radius of their coverage as shown in Figure 1, the field is divided into several states and modeled as an unknown *grid world*. In this study, $S = \{s_1, s_2, ..., s_n\}$ represents a set of n states of the grid world. During the learning process, each agent starts from one state s, interacts with neighboring cells by taking an action from a set of actions $A = \{a_{(up)}, a_{(down)}, a_{(left)}, a_{(right)}\}$ and then moves to the new state s' and receives a reward r (Figure 2).

In this study, the main goal is to enforce agents to find free states, the states blocked with obstacles (unexpected objects or plant rows), and goal states (regions of interest) aiming at creating the field's model and its graph. To this end, we assume one state as a terminal state $s_{terminal}$ which is the furthest from starting state to make sure that agents explore the whole field. Based on this definition, by taking action $a \in A$ in current state s, there are three different cases for mapping the pair of $(state, action)$ to $(next \quad state, reward)$ pair as

$$(s, a) \rightarrow \left\{ \begin{array}{lll} (s', 1) & \text{if} & s' = free, \\ (s, -1) & \text{if} & s' = obstacle, \\ (s', 10) & \text{if} & s' = terminal. \end{array} \right) \tag{1}$$

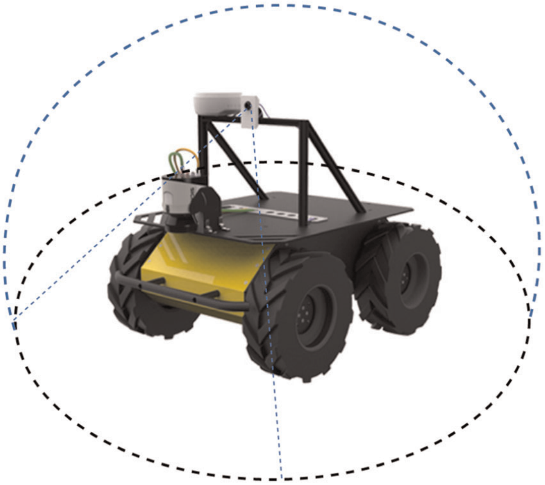

FIGURE 1 An illustration of visible area around a ground robot [Color figure can be viewed at wileyonlinelibrary.com]

As interpreted from this function, if taking action a in current state s maps agent to a free state, it moves towards the next state s' and gets reward $r = 1$, but if (s, a) results in moving towards a blocked state (obstacle or plant), the agent stays in the current state s and would be punished by receiving reward $r = -1$. In the last scenario, if the agent reaches the terminal state $s_{terminal}$ by choosing action a in state s, it moves towards the terminal state and would be rewarded with $r = 10$.

To learn a model of the maze (field) using a team of agents, inspired by the method presented in Kivelevitch and Cohen,[26] we define the problem as follows: *Deploy a group of Magents to first cooperatively explore the environment and then, by experiencing different episodes, determine their path towards the terminal state*. It is expected that after playing a few episodes ending at terminal state, agents can build a model of the field.

To share tasks, avoid collisions between agents and make the training process as fast as possible, first, it is assumed that there is a plain maze with no goal or obstacle.

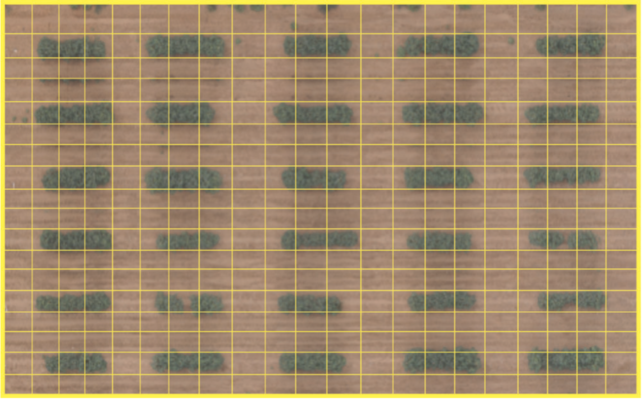

FIGURE 2 An example of a farm converted into the grid world [Color figure can be viewed at wileyonlinelibrary.com]

Then, considering the current state of each agent (starting state) and its speed, all states are divided among the agents similar to a simple Voronoi diagram. In the next step, they start exploring the states of their Voronoi sub-states, and at each step, agents update their sub-states and share the sub-model ($Model(s, a) \leftarrow (s', r)$) with other agents.

Next, we describe the model-based Q-learning method, Dyna-Q+, which is developed by combining simple Q-learning and Q-planning algorithms (see Figure 3) and which is an improved Dyna-Q method that is applicable for a complex maze.[2] In this method, first, each agent chooses an action in state s based on $\epsilon - greedy$ method to achieve a trade-off between exploration and exploitation in such a way that for a small ϵ $(0 \le \epsilon \le 1)$, agent picks an action at random for a portion ϵ (exploration) or selects the best action for a portion $1 - \epsilon$ (exploitation).

To cope with stochastic environments whose models may be inaccurate, agents need to experience new behaviors (actions) in previously observed states when the maze is changing. In Dyna-Q+ algorithm, the agent records the last time that each $(state, \quad action)$ was experienced, and the longer time after experiencing that pair of state-action means the greater possibility of changing the dynamics of this pair and the corresponding model. Dyna-Q+ methods encourage agents to try actions that were not experienced for a long time by adding a special bonus on reward as a function of how many time steps T the state-action pair was last tried. Then, the total reward for a transition is calculated as follows:

$$R = r + k\sqrt{T}, \tag{2}$$

where r is the transition reward based on the function defined in (1) and k is a small number defined as a time weight parameter. After executing chosen action a at state s, based on the agent's observation, a model associated with this pair of state-action is recorded as:

$$Model(s, a) \leftarrow (s', R). \tag{3}$$

The main step in Q-learning is to find the best action (optimal policy) that can maximize the total reward (see Algorithm 1). In other words, at each step, the values for state-action pairs are updated in a Q-table until it converges to the maximum value. A simple update rule for Q value is

$$Q(s, a) \leftarrow Q(s, a) + \beta[R + \gamma \max_a(Q(s', a) - Q(s, a))], \tag{4}$$

where β is the learning rate and $\gamma \in [0, 1]$ is a discount factor adjusted for striking a balance between future reward and immediate reward received by acting at each state.

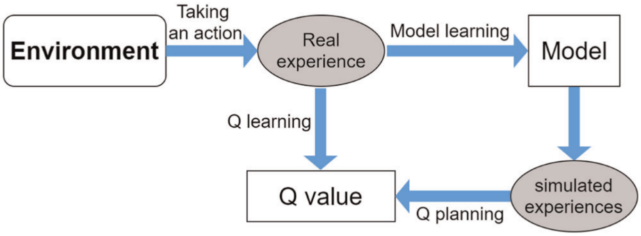

FIGURE 3 Relationship among learning, planning and acting in Dyna-Q+ algorithm [Color figure can be viewed at wileyonlinelibrary.com]

As shown in Figure 3, in Dyna-Q algorithm, agents interact with the environment, update Q table and learn its model through the real experience. Then, this model is used to generate simulated experiences for planning step that is repeated and Q values are updated throughout both direct RL and planning updates. At the end of each learning step, the recorded model is shared among all agents and also updated when solving the coverage control problem discussed in the next section.

2.2 | Coverage control algorithm

After searching the grid world and experiencing a number of episodes, agents build a model of maze (field) with N states. Next, the field model is converted into a weighted graph $G(S, E, C)$, where $S \in \{1, 2, ..., N\}$ is the set of states, $E \subseteq |S| \times |S|$ denotes the set of edges whose elements indicate whether or not there is a path between the center points of each pair of states. Furthermore, C is the weight matrix representing metric lengths of the edges.[27] In this study, the main purpose of finding the field graph and important states is to optimally distribute UGVs over the coverage area by minimizing the corresponding cost function. To achieve this goal, since UGVs may move at different speeds, we formulate the cost as a function of traveling time defined as follows:

$$t_{(s_{p_i}, s_q)} = \frac{d_{(s_{p_i}, s_q)}}{v_i}, \tag{5}$$

where v_i denotes the speed of the ith vehicle, and $d_{(s_{p_i}, s_q)} \in C$ is the Euclidean distance between the ith vehicle's current state p_i and state q. Moreover, by knowing the optimal path starting from state p_i and ending at state s, total optimal traveling time for each vehicle is the sum of the travel times (costs) from its current state p_i to state s. In this study, the shortest path between each pair of states is computed using Dijkstra's algorithm[28] and then by knowing $path = \{p, q, ..., r, s\}$, the total time τ is

$$\tau_{(s_p, s_s)} = t_{s_p, s_q} + \cdots + t_{s_r, s_s}. \tag{6}$$

To minimize the corresponding cost function, after finding the minimum time between each pair of states, the field is partitioned into M Voronoi sub-graphs g_{r_i} for $i \in \{1, 2, ..., M\}$ to share tasks among M vehicles proportionally. To this end, based on the Lloyd's algorithm,[12,27,29] the optimal Voronoi diagram g_{r_i} for ith vehicle is a partition of the area calculated as

$$g_{r_i} = \left\{ s_q \in S | \tau_{(s_{p_i}, s_q)} \leq \tau_{(s_{p_j}, s_q)}, \forall\ i \neq j \right\}. \tag{7}$$

Note that, in this study, it is assumed that all vehicles are initially fully charged and remain sufficiently charged for completing their assigned task.

Using the result of the Voronoi partitioning, the ith vehicle is responsible for covering the states (associated graph vertices) in its sub-graph g_{r_i}. Then, as explained in references [14,27], the total cost is formulated as follows:

$$H(p, g_r) = \sum_{i=1}^{M} \sum_{q \in g_{r_i}} \tau_{(s_{p_i}, s_q)} \phi(q), \tag{8}$$

where $\phi(q)$ is the priority value corresponding to state q.[13] As the field is converted into a graph, the goal states are given higher priority values and lowest values are assigned to states that are far from the important states. Then, the total traveling time (cost) will be minimized, and an optimal solution is reached only when the distance between the current state of the vehicle and the goal state $d_{(s_{p_i}, s_q)}$ converges to zero.

Algorithm 1: Implementation of the proposed method.

```
Input: Field dimension, sensors' footprint, agents' speed, learning parameters
Output: Environmental model of the field, field graph, Voronoi sub-states, optimal paths
Learning process:
1. Receive locational information of vehicles
2. Convert the field into a grid world (cells)
3. Compute the Voronoi sub-states for each vehicle
4. Initialize $Q(s, a)$ and $Model(s, a)$ for all $s \in S, a \in A$
5. While steps $\leq$ maximum steps:
for $episode = 1, 2, \ldots$ do:
$s \leftarrow$ current non-terminal state
$a \leftarrow \epsilon - greedy \quad (s, Q)$
$(s', r) \leftarrow$ execute action $a$
$T_{(step)} \leftarrow T_{(step-1)} + 1$ for all non-visited states
$R \leftarrow r + k\sqrt{T}$
$Q(s, a) \leftarrow Q(s, a) + \beta[R + \gamma \max_a(Q(s', a) - Q(s, a))]$
$Model(s, a) \leftarrow (s', R)$
Update Voronoi diagram based on learned model and current state
for $n$ = sampling steps repeat:
$s \leftarrow$ previously observed state
$a \leftarrow$ random action taken previously in state $s$
$(s', R) \leftarrow Model(s, a)$.
$Q(s, a) \leftarrow Q(s, a) + \beta[R + \gamma \max_a(Q(s', a) - Q(s, a))]$
end while
Coverage control
6. Generate weighted graph of field $G$
7. Set the edges that end to an obstacle to zero
8. Find shortest path between center points of each pair of cells.
9. Compute initial Voronoi regions for each vehicle
10. While cost function has not converged:
 if there is a neighboring cell $u$ so that $H(u, g_i)$ is minimum in $g(r_i)$, then:
 Set $u$ as the next best point
 Move towards state $u$
 if state $u$ is blocked:
 Update field model and Voronoi diagram, and Repeat step 10.
  end if
  end if
   end while
```

3 | SIMULATION RESULTS AND DISCUSSION

In this section, we show the results of validating our proposed method for learning and monitoring important areas in an agricultural field through simulation studies. A small farm sized $\times 21\ m^2$ is considered as an unknown environment and simulated as a maze (see Figure 2). In our proposed setting for field model learning and coverage, different regions of interest can be viewed as being associated with different tasks. For instance, one region can be defined from the point of view of water stress, while another from that of biotic stress in plants. It is expected that after building the field model, these regions would be discovered and defined as goal states in the proposed model-based RL method. Based on the field size, in all scenarios, we assumed that two vehicles (agents) with different speeds are deployed to explore the environment and extract the field model and then use it to solve the coverage problem. Both agents are equipped with appropriate sensors to cover the area, collect data and observe their neighboring cells. Furthermore, based on the initial pre-processing of collected information, they are expected to identify important plots and assign a corresponding task to each one of them. All steps including the simulation of the field as a grid world (maze), learning process and coverage problem are implemented in *Python* programming language*. To find the size of states, we assumed that the optimum sensing radius (Figure 1) is $0.5\ m$ (an arbitrary value) so each state covers $1 \times 1 = 1\ m^2$ and hence the field is converted into a maze with $18 \times 21 = 378$ cells (Figure 4).

Here, two scenarios are considered for the validation of the proposed method. In both scenarios, first agent gets into the simulated field from the bottom left corner of the maze where

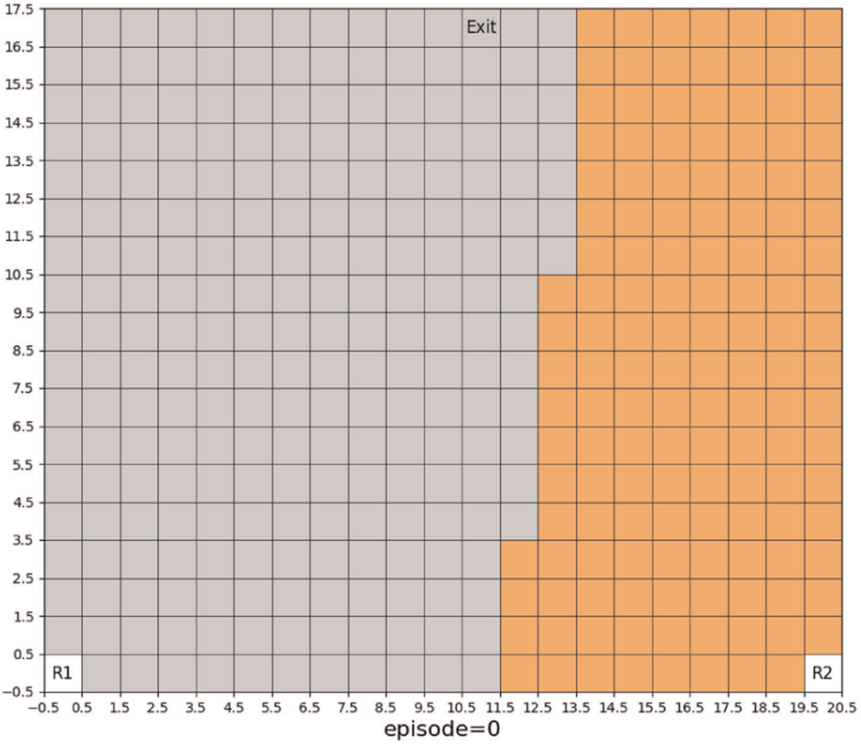

FIGURE 4 Voronoi diagram before the model learning begins in both scenarios [Color figure can be viewed at wileyonlinelibrary.com]

the corresponding cell is [0, 0], and second agent starts from the bottom right corner which is state [0, 20]. To solve the reinforcement learning problem as an episodic problem and for enforcing agents to explore the whole maze, state [17, 11] is considered as terminal (exit) state (see Figure 4). In each episode, agents begin experiencing from starting cell and find their trajectories towards the exit cell. For limiting total experience steps, considering the maze's size and its complexity (the number of unpredictable obstacles and goals), a maximum threshold for steps is defined, which means that when the cumulative steps of all episodes exceeds the maximum step, the learning process terminates.

One of the key factors to achieve reliable results from Q-learning and Q-planning steps is the selection of the best set of learning parameters. All the parameters were tuned by trial and

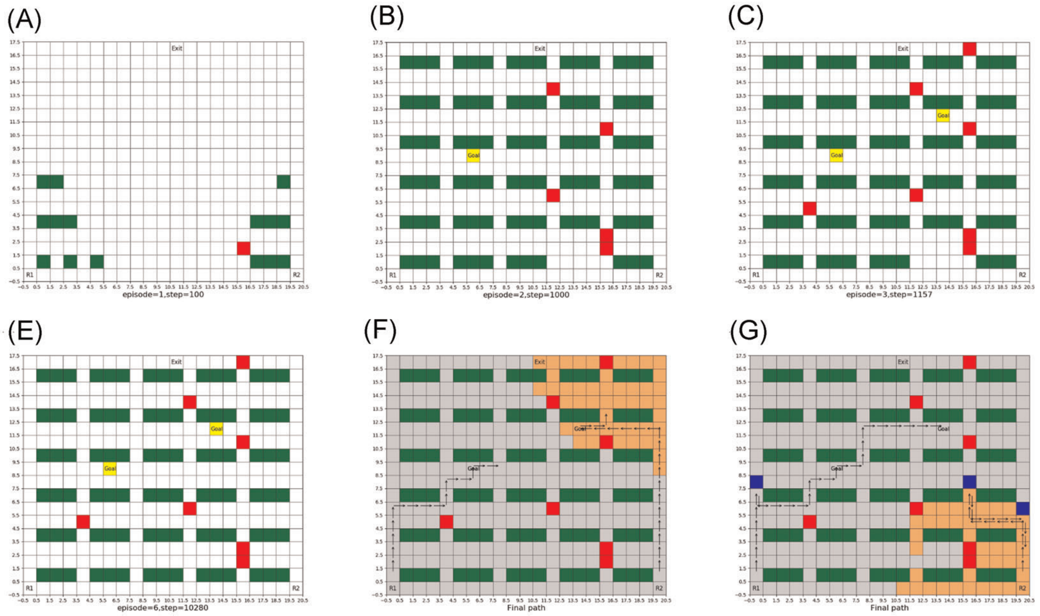

FIGURE 5 Plots show simulation results for the first scenario. The starting states for the two agents are shown by R1 and R2, and the terminal state is shown by Exit. Gray cells are Voronoi sub-states of the first agent, and the orange cells are those of the second agent. There are three types of obstacles found during learning and coverage process: green cells which represent plant rows, red cells representing unknown objects blocking cells/paths detected during learning process and blue cells representing new obstacles found while coverage problem is solved in real time. Subplot (E) illustrates the final Voronoi diagrams for the case that no obstacle is added while coverage problem is being solved. In comparison, subplot (F) shows final Voronoi partitions and trajectories for the case that maze changes and new obstacles (blue cells) are added (and hence learned online). This leads the vehicles to alter their directions to find free paths towards the goals and the centers of their partitions. It is noted that as (E) shows, each agent was initially tasked to cover one goal, but because of the new obstacles and hence re-learning of the field in real time, first agent ended up covering both goals. Another important point to make here is that once the goal states are covered, the agents can either stop moving or move towards the center of their Voronoi partitions—in this scenario, we considered the latter (A) Learned model 100 steps into the first episode; (B) learned model 1000 steps into the second episode; (C) Learned model at the end of the third episode (step = 1157); (D) Learned model at the end of the last episode (step = 10,280); (E) Final model of maze after 20,000 steps. Black arrows indicate trajectories of agents towards goal states if offline learned model of the environment is used for coverage. (F) Final model of maze after 20,000 steps assuming that some states (blue cells) have changed while coverage problem is being solved [Color figure can be viewed at wileyonlinelibrary.com]

error aiming at minimizing the total running steps by ensuring that the whole maze is explored and the model is updated when it changes. For learning rate and discount factor described earlier (that appear in (4)), we chose $\beta = 0.7$ and $\gamma = 0.95$, respectively. Furthermore, $\epsilon = 0.1$ was considered as a probability for exploration and the time weight in (2) was set to $k = 10^{-2}$. As learning the model of a dynamic field is one of the main objectives of this study, we assumed that maze is changing at some steps meaning that some blocked cells are added or removed and some cells are defined as important states during the learning process. Next, we describe each scenario and demonstrate and discuss associated results (Figures 5 and 6).

Scenario 1: In this scenario, first agent is assumed to be faster than the second one and moves at the speed of $v_1 = 20\,\text{cm}/s$ while second one moves at the speed of $v_2 = 15\,\text{cm}/s$. It is assumed that agents know only the number of cells and have no knowledge about the states blocked with plant rows or obstacles, and goal states. As shown in Figure 4, before starting the first episode, agents solve Voronoi partitioning problem described earlier and the field cells are divided between two ground vehicles considering their speeds. Then, first episode starts and vehicles try to find their path to the exit state. As agents are exploring the maze for a free path, they also update the model of the field in real time. It is noted that the number of steps at each episode is not fixed and an episode ends when agents visit the exit state.

Figure 5 illustrates the results of simulating the first scenario. The maximum running step is set to 20,000 steps, and as observed from the results, agents find the environment model after experiencing six episodes. In this setup, during the learning process, the maze changes, and some obstacles or goals are added at random running steps. In this simulated environment, only two important regions are considered and the corresponding cells are assigned to the goal states for further investigation. Then, the field graph is generated using its model, and agents are deployed in the field to make the best coverage over the regions of

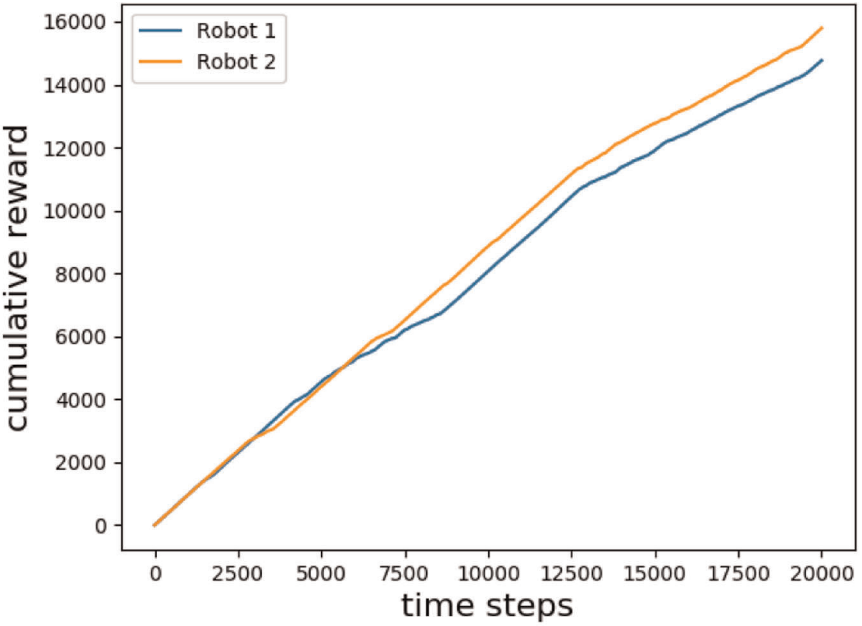

FIGURE 6 Cumulative rewards for first scenario after 20,000 steps. The agents are enforced to use past experience to explore which actions lead to higher cumulative rewards [Color figure can be viewed at wileyonlinelibrary.com]

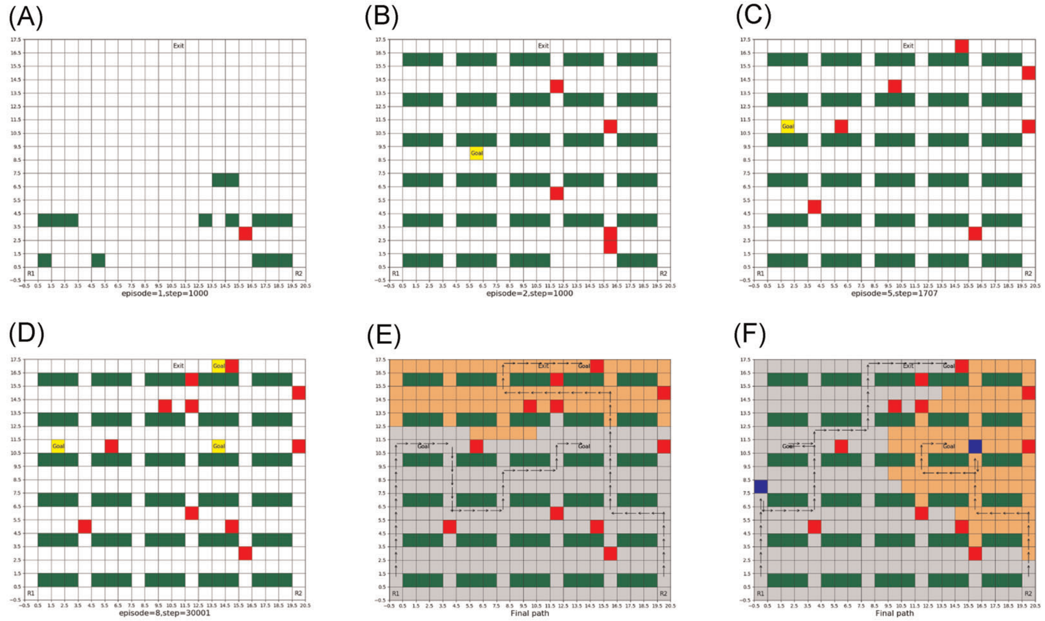

FIGURE 7 Plots show simulation results for the second scenario. In this experiment, three regions of interest (goals) are found during the learning process. Plot (E) illustrates the final Voronoi diagrams and vehicles' trajectories to reach the goals or the center of their Voronoi partitions while the maze is static after learning the field's model. In contrast, plot (F) shows final partitioning and trajectories for the case that field changes while the coverage problem is being solved. In this case, second vehicle that is slower moves towards (and covers) one of the goals while the first vehicle finds an optimal path to monitor two other goals. As observed from the plots (E) and (F), with the addition of two obstacles, both assigned tasks and trajectories have changed. It is noted that in this scenario, we enforce the agents to stop moving once all the goal states are visited (A) Learned model 1000 steps into the first episode. (B) Learned model 1000 steps into the second episode. (C) Learned model at the end of the fifth episode (step = 1707). (D) Learned model at the end of the last episode (step = 30,000). (E) Final model of the maze after 30,000 steps. Black arrows indicate final trajectories of agents towards goal states. (F) Final model of the maze after 30,000 steps. As observed, some states (blue cells) have changed while coverage problem is being solved [Color figure can be viewed at wileyonlinelibrary.com]

interest (see Figures 2 and 5). As observed from Figure 5E, first, it is assumed that the maze does not change after learning its model and no obstacle is added while agents are finding the next best points to move to. Therefore, each agent moves towards the closest important area to have better coverage over the field. In the second case illustrated in Figure 5F, the maze alters while coverage problem is being solved, and agents find new obstacles (blue cells) along their route, update field's graph and Voronoi diagrams, and hence change their paths. In this case, although one of the goals is close to the first vehicle and the other one is further away from it, both goals are eventually covered by the first vehicle because the second vehicle's path to reach the target is blocked by several obstacles. Nevertheless, second vehicle moves towards the state at the center of its Voronoi diagram (i.e., the centroid) to minimize the corresponding cost function.

Scenario 2: Similar to the first scenario, we assume that $v_1 = 20\,\text{cm/s}$ and $v_2 = 15\,\text{cm/s}$, and hence the initial Voronoi diagrams for both agents are the same as those shown in Figure 4. The maze is assumed to change more often during the learning process and hence more

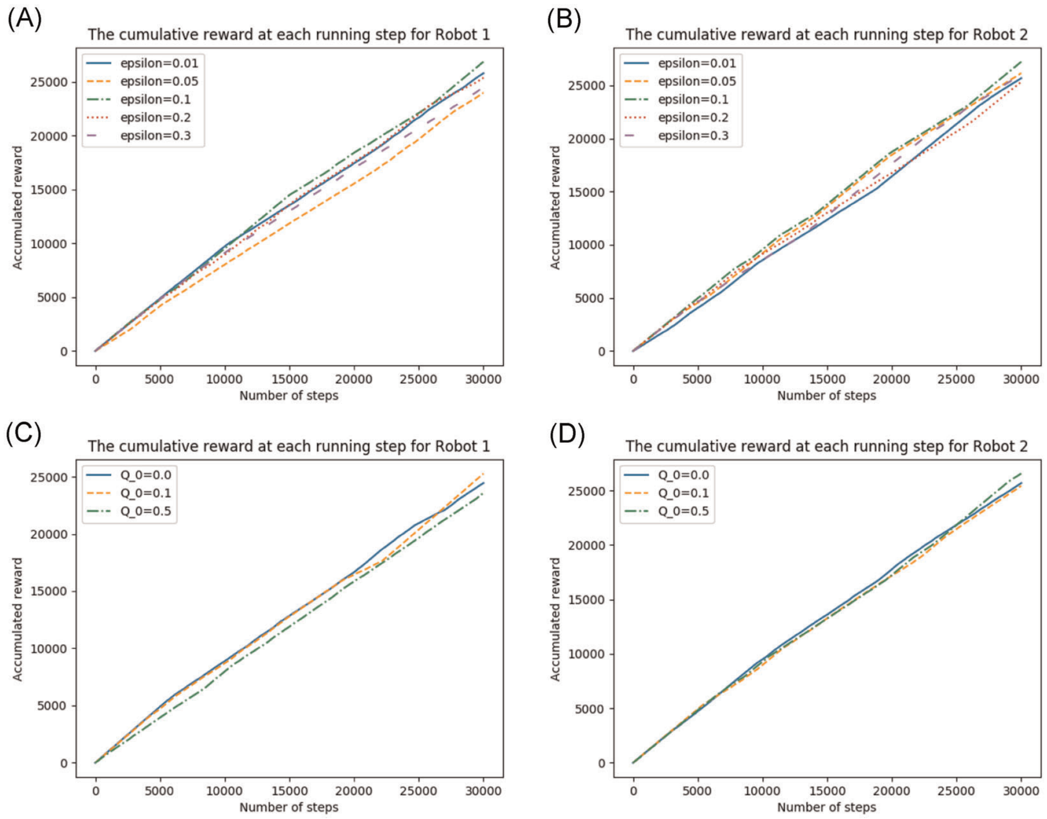

FIGURE 8 Plots show cumulative rewards for the second scenario after 30,000 steps using different parameters of initial Q-function and ϵ. As observed from plots (A) and (B), the case $\epsilon = 0.1$ outperforms other cases throughout the learning process. As seen from plots (C) and (D), the case $Q_0 = 0$ outperforms the other two cases examined here for the first episodes but then is beaten in the very last episode because it has already learned the model. However, considering the combination of the rewards associated with the two robots, the case with $Q_0 = 0$ still outperforms other ones. It is noted that due to the selected reward and punishment values (1 and −1, respectively) for identifying or missing free cells and blocked states (obstacle or plant), a line with the unity slope indicates a perfect model learning; from the plots, it is apparent that the case with $\epsilon = 0.1$ and $Q_0 = 0$ is the closest (A) Cumulative reward for Robot 1 when ϵ varies; (B) Cumulative reward for Robot 2 when ϵ varies; (c) Cumulative reward for Robot 1 when the initial values of Q matrix vary; (d) Cumulative reward for Robot 2 when the initial values of Q matrix vary [Color figure can be viewed at wileyonlinelibrary.com]

complex compared to the simulated maze in the first scenario. For this reason, the maximum step is set to 30,000 iterations which is long enough to build the field model. Subplots in Figure 7 illustrate the results of the second scenario simulations. It is shown that after running eight episodes, the environment model is learned and agents solve the coverage problem while the maze is changing; they also update the corresponding graph in real time (Figure 7F). Moreover, as discussed in Section 2, our main goal in using Q-learning is to maximize the total reward which enforces agents to choose the best action. As shown in Figures 6 and 8, the cumulative reward for both scenarios increases with the running steps.

Furthermore, we examine the performance of the proposed model-based RL approach by varying the parameters involved in dyna-Q+ algorithm. In particular: (a) we fix the initial

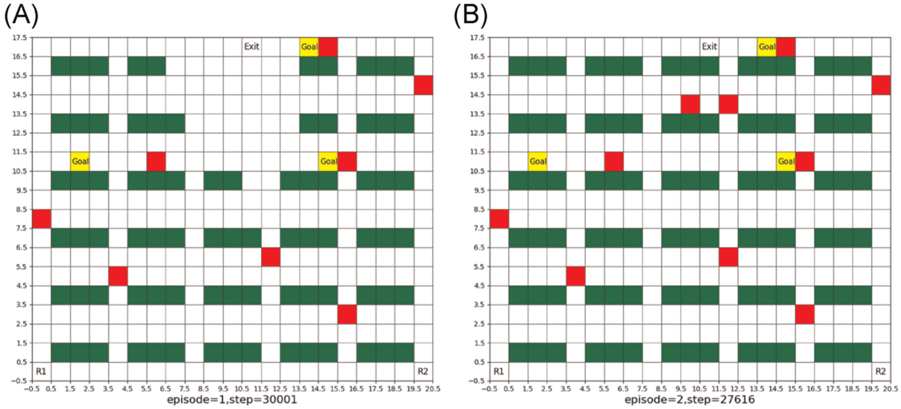

FIGURE 9 Comparison of the learned models using episodic and non-episodic experiences (A) Learned model after 30,000 steps in a non-episodic experience. (B) Learned model 27,616 steps into the second episode of an episodic experience [Color figure can be viewed at wileyonlinelibrary.com]

values of Q-function and vary the randomness in the action selection (i.e., value of ϵ), and (b) we fix the value of ϵ and vary the initial values of Q-function. The results shown in Figure 8 demonstrate that the pair $\epsilon = 0.1$ and initial Q-function of zero $(Q_0 = 0)$ is the optimal combination for these two parameters. The model learned using dyna-Q+ giving the plots in Figure 7 also use this pair of parameters ϵ and Q_0.

Remark. It is noted that terminal state in the two simulation scenarios discussed above was not selected randomly; it was chosen to be the furthest state from the agents' starting states. In non-episodic environment (without terminal state), there is no goal state to achieve, and agents are enforced to just explore the maze to find free states and obstacles. In this approach, since agents take actions, for example, based on greedy algorithms, and past experiences affect the current action, the learning process will become biased and agents tend to explore surrounding free states over and over. To address this issue and make the learning process faster, a terminal state was defined in such a way that agents are forced to explore the whole maze. As an illustrative example, Figure 9 shows the model learning results using episodic and non-episodic approaches for a given number of steps associated with Scenario 2 discussed before. It is obvious from Figure 9A that agents could not explore all cells by just moving around the field and some states remained unexplored at the end of the experience. In comparison, the episodic approach results (Figure 9B) indicate that the model of the whole maze is built before reaching the maximum running steps.

4 | CONCLUSION

In this paper, a cooperative learning approach was presented to model a dynamic environment/field (in which multiple agents are present) and then optimally distribute agents in the field for coverage and data collection. Using Dyna-Q+ algorithm (a model-based reinforcement learning method), a reward and punishment function was designed to enforce vehicles to find exit state and learn the environment at the same time. The extended (cooperative) learning method introduced and formulated here for multiagent systems used the Voronoi partitioning. The proposed approach eventually led to dividing the search space among the agents while they could share their observations through a communication network. Simulation results for different scenarios were presented to demonstrate the efficacy of the proposed model learning and planning approach. One of the challenges in applying our proposed method in the real world would be the scalability. Larger farms result in a larger number of states and hence more agents. As the field size increases, the learning process requires more running steps to build the whole field map. This issue can be examined as a future research direction by dividing the large-sized fields into a number of small sub-fields and then deploying robotic system to each sub-field for cooperatively learning its model.

ACKNOWLEDGMENTS

This study was supported by the NSF/NIFA National Robotics Initiative (NIFA grant #2017-67021-25928), as well as the US National Science Foundation under award #1934481.

ENDNOTES

*https://github.com/SabaFaryadi/Master/blob/master/RL-based%20modeling%20and%20coverage/smallField.py

ORCID

Saba Faryadi https://orcid.org/0000-0003-1743-238X
Javad Mohammadpour Velni https://orcid.org/0000-0001-8546-221X

REFERENCES

1. Liakos KG, Busato P, Moshou D, Pearson S, Bochtis D. Machine learning in agriculture: A review. *Sensors*. 2018;18(8):2674.
2. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press; 2018.
3. Yang E, Gu D. Multiagent reinforcement learning for multi-robot systems: A survey. Tech. rep; 2004.
4. Hagras H, Colley M, Callaghan V, Carr-West M. Online learning and adaptation of autonomous mobile robots for sustainable agriculture. *Auton Robots*. 2002;13(1):37-52.
5. Ball D, Ross P, English A, et al. Farm workers of the future: Vision-based robotics for broad-acre agriculture. *IEEE Robot Autom Mag*. 2017;24(3):97-107.
6. Pajares G, García-Santillán I, Campos Y, et al. Machine-vision systems selection for agricultural vehicles: A guide. *J Imaging*. 2016;2(4):34.
7. Hagras H, Colley M. An embedded-agent architecture for online learning and control in intelligent machines. In: Mohammadin M, ed. *New Frontiers in Computational Intelligence and its Applications*. Amsterdam, The Netherlands: IOS Press; 1999:165-174.
8. Emmi L, Gonzalez-de Soto M, Pajares G, Gonzalez-de Santos P. New trends in robotics for agriculture: integration and assessment of a real fleet of robots. *Sci World J*. 2014;2014:1-21.
9. Duckett T, Pearson S, Blackmore S, et al. Agricultural robotics: the future of robotic agriculture. arXiv preprint arXiv:1806.06762; 2018.
10. Shamshiri RR, Weltzien C, Hameed IA, et al. Research and development in agricultural robotics: A perspective of digital farming. *Chin Soc Agric Eng*. 2018.
11. Vasconez JP, Kantor GA, Cheein FAA. Human-robot interaction in agriculture: A survey and current challenges. *Biosyst Eng*. 2019;179:35-48.
12. Yun SK, Rus D. Distributed coverage with mobile robots on a graph: locational optimization and equal-mass partitioning. *Robotica*. 2014;32(2):257-277.
13. Faryadi S, Davoodi M, Mohammadpour Velni J. Agricultural Field Coverage Using Cooperating Unmanned Ground Vehicles. In: Dynamic Systems and Control Conference, vol. 59155. American Society of Mechanical Engineers; 2019: V002T25A003.

14. Davoodi M, Velni JM, Li C. Coverage control with multiple ground robots for precision agriculture. *Mech Eng*. 2018;140(06):S4-S8.
15. Weiss U, Biber P. Plant detection and mapping for agricultural robots using a 3D LIDAR sensor. *Robot Auton Syst*. 2011;59(5):265-273.
16. Hiremath SA, Van Der Heijden GW, Van Evert FK, Stein A, TerBraak CJ. Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter. *Comput Electron Agric*. 2014;100: 41-50.
17. Low ES, Ong P, Cheah KC. Solving the optimal path planning of a mobile robot using improved Q-learning. *Robot Auton Syst*. 2019;115:143-161.
18. Konar A, Chakraborty IG, Singh SJ, Jain LC, Nagar AK. A deterministic improved Q-learning for path planning of a mobile robot. *IEEE Trans Syst Man Cybern Syst*. 2013;43(5):1141-1153.
19. Tai L, Paolo G, Liu M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) IEEE; 2017: 31-36.
20. Xiao J, Wang G, Zhang Y, Cheng L. A distributed multi-agent dynamic area coverage algorithm based on reinforcement learning. *IEEE Access*. 2020;8:33511-33521.
21. Pham HX, La HM, Feil-Seifer D, Nefian A. Cooperative and distributed reinforcement learning of drones for field coverage. arXiv preprint arXiv:1803.07250; 2018.
22. Adepegba AA, Miah S, Spinello D. Multi-agent area coverage control using reinforcement learning. In: The Twenty-Ninth International Flairs Conference; 2016.
23. Bae H, Kim G, Kim J, Qian D, Lee S. Multi-robot path planning method using reinforcement learning. *Appl Sci*. 2019;9(15):3057.
24. Mirowski P, Pascanu R, Viola F, et al. Learning to navigate in complex environments. arXiv preprint arXiv:1611.03673; 2016.
25. Yue W, Guan X, Wang L. A novel searching method using reinforcement learning scheme for multi-UAVs in unknown environments. *Appl Sci*. 2019;9(22):4964.
26. Kivelevitch EH, Cohen K. Multi-agent maze exploration. *J Aeros Comput Inf Commun*. 2010;7(12):391-405.
27. Javanmard Alitappeh R, Pereira GAS, Araújo AR, Pimenta LCA. Multi-robot deployment using topological maps. *J Intell Robot Syst*. 2017;86(3):641-661.
28. Dijkstra EW. A note on two problems in connexion with graphs. *Numer Math*. 1959;1(1):269-271.
29. Cortes J, Martinez S, Karatas T, Bullo F. Coverage control for mobile sensing networks. *IEEE Trans Robot Autom*. 2004;20(2):243-255.

How to cite this article: Faryadi S, Mohammadpour Velni J. A reinforcement learning-based approach for modeling and coverage of an unknown field using a team of autonomous ground vehicles. *Int J Intell Syst*. 2021;36:1069–1084. https://doi.org/10.1002/int.22331