Decentralized Function Approximated Q-Learning in Multi-Robot Systems For Predator Avoidance

Revanth Konda, Hung Manh La, and Jun Zhang

***Abstract*—The nature-inspired behavior of collective motion is found to be an optimal solution in swarming systems for predator avoidance and survival. In this work, we propose a two-level control architecture for multi-robot systems (MRS), which leverages the advantages of flocking control and function approximated reinforcement learning for predator avoidance task. Reinforcement learning in multi-agent systems has gained a tremendous amount of interest in recent few years. Computationally intensive architectures such as deep reinforcement learning and actor-critic approaches have been extensively developed and have proved to be extremely efficient. The proposed approach, comprising of cooperative function approximated Q-learning, is applied such that it ensures formation maintenance in MRS while predator avoidance. A consensus filter is incorporated into the control architecture, to sense predators in close vicinity in a distributed and cooperative fashion to ensure consensus on states among the robots in the system. The proposed approach is proved to be convergent and results in superior performance in unexplored states and reduced number of variables. Simulation results confirm the effectiveness of the proposed approach over existing methods. We expect that the proposed approach can be conveniently applied to many other areas with little modifications, such as fire-fighting robots, surveillance and patrolling robots.**

***Index Terms*—Multi-robot systems, obstacle avoidance, reinforcement learning.**

I. Introduction

NATURE has presented various forms of swarm behaviors, such as bird flocking, tetrapod herding, and fish schooling [1]. These swarm behaviors often involve collective motions of a large number of animals and are critical for foraging benefits and safety from predation [2]. By analyzing and utilizing the intelligence in animal's coordination and cooperation, multi-robot systems significantly enhance their capabilities in coordination and control to solve team-level and global tasks based on local interaction rules [3], [4]. Multi-robot systems (MRS) can be robustly deployed to complete complex tasks in a wide varieties of areas, such as fire fighting [5], [6], environmental monitoring and exploration [7], home surveillance [8], search and rescue [9], industrial purposes [10], precision agriculture [11], and predator avoidance [12]. For example, when a team of unmanned aerial vehicles is deployed to implement a surveillance task, the robot team can coordinate their individual motions and tasks. When under attack, the robot team can avoid the attack by collectively moving to a safe location.

There have been a lot of successful demonstrations and developments on the modeling, coordination, and control of MRS [4], [13]. Topics on flocking and coordination modeling [14], formation control [15], [16], coverage control [17], task allocation [18] and learning [12] have been studied extensively. While different approaches on coordination, cooperation, and consensus for MRS exist, it is often extremely difficult to find an optimal solution for a given problem [13]. The application of multi-agent reinforcement learning (MARL) techniques in MRS is highly suitable and has sparked lot of interest mainly because of its efficiency in finding an optimal solution [19], [20]. Research on MARL has mainly focused on policy selection [21], credit assignment [22] and cooperative learning [23]. MARL has been applied in many fields including multi-agent game theory [24], differential and stochastic games [25], and multi-robot systems [26].

By using reinforcement learning (RL), it has been proved that flocking and collective motion is the optimal solution for survival and predator avoidance in swarming system [27]. Using this information, we propose a two-level control architecture which leverages the advantages of flocking control and function approximated learning. We apply it for predator avoidance in MRS such that they maintain full connectivity while avoiding predators by choosing an appropriate safe place. Maintaining formation enables the robots in the system to have a wider sensing range and facilitates accurate sensing of the environment [27], [28]. Most of the current algorithms do not focus on formation maintenance while predator avoidance. To tackle this problem, the cooperative Q-learning was proposed in our previous study [12], however, it was found that the size of the state-space increased exponentially with the number of predators and discretization of the direction. A consensus filter [28] can be designed in MRS to sense the presence of a predator in close vicinity of the system in a cooperative and

Manuscript received March 23, 2020; accepted July 27, 2020. Date of publication August 4, 2020; date of current version August 12, 2020. This letter was recommended for publication by Associate Editor M. A. Hsieh and Editor N. Y. Chong upon evaluation of the reviewers' comments. This work was supported in part by the U.S. National Science Foundation (NSF) under Grants NSF-CAREER: 1846513 and NSF-PFI-TT: 1919127, and in part by the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (USDOT/OST-R) under Grant 69A3551747126 through INSPIRE University Transportation Center. The views, opinions, findings and conclusions reflected in this publication are solely those of the authors and do not represent the official policy or position of the NSF and USDOT/OST-R. *(Corresponding author: Revanth Konda.)*

Revanth Konda and Jun Zhang are with the Department of Mechanical Engineering, University of Nevada, Reno, NV 89557 USA (e-mail: rkonda@nevada.unr.edu; jun@unr.edu).

Hung Manh La is with the Department of Computer Science and Engineering, University of Nevada, Reno, NV 89557 USA (e-mail: hla@unr.edu).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3013920

decentralized way. This is presented as cooperative predator sensing algorithm (CPSA) in this letter. With the combination of function approximation learning, flocking control and CPSA, in this work, a novel control architecture is proposed and applied for MRS. The high-level controller realized based on function approximated reinforcement learning algorithm makes the entire control framework capable of working in a continuous mode. The learning approach in the proposed architecture is similar to state-of-the-art methods; however, there are notable differences, which have been highlighted below.

Firstly, MARL has been predominantly applied in the classic form of Q-learning [29], whose accurate performance is heavily relied on dense discretization of the state space. Based on the state, the agent selects an action by using a look-up table called Q-table [30]. For many applications involving large scale scenarios, the discretization process leads to a huge state-space, consequently resulting in storage of many variables and computationally intensive RL architectures. Furthermore, the approach demands the exploration of all states. Secondly, in recent times, many state-of-the-art algorithms have been developed using actor-critic reinforcement learning (ACRL) and deep reinforcement learning (DRL) methods [31], [32] for multi-agent systems. These frameworks have been applied as function approximators [33], [34] to overcome huge state-space while maintaining good performance in unexplored states. Function approximation has been realized in reinforcement learning scenario [35], where accurate performance and low data storage and computation can be simultaneously achieved.

The need for computationally intensive frameworks such as DRL and ACRL arises from the problem of huge state-space with many dimensions. The algorithm presented in [22] uses centralized critic and decentralized actors. The method presented in [36] is close to what we propose, the difference being that they used actor-critic approach while we use function approximated Q-learning. Also, in [36], the agents always rely on RL to take the necessary actions and thus do not require consensus on states. The proposed method utilizes only local rewards and local information for learning. In our previous work [12], the state-space was huge due to many dimensions. However, in the present work, the CPSA plays a critical role in reducing the state-space by reducing its dimensionality. This makes the problem relatively simple and leads to the use of a simpler RL mechanism such as linear function approximated Q-learning.

The main contributions of this letter are as follows:

- Development of a two-level control architecture which includes a distributed RL algorithm and its application in MRS for predator avoidance. The convergence of the proposed learning algorithm is derived.
- Integration of a consensus filter (CPSA) to the proposed algorithm for ensuring state consensus among all robots in the system, which proves to be critical for learning convergence as well as use of simple RL architecture.
- Confirmed effectiveness of the proposed algorithm for predator avoidance in MRS over existing methods.

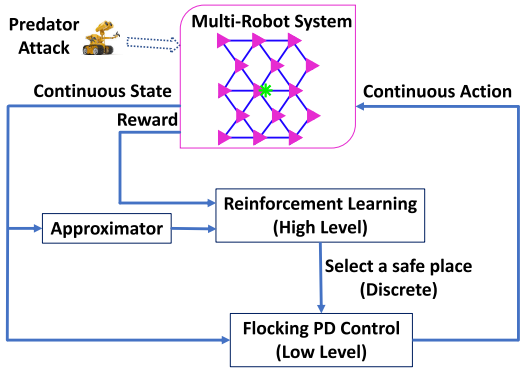

Fig. 1. Architecture of the proposed FA-MARL algorithm.

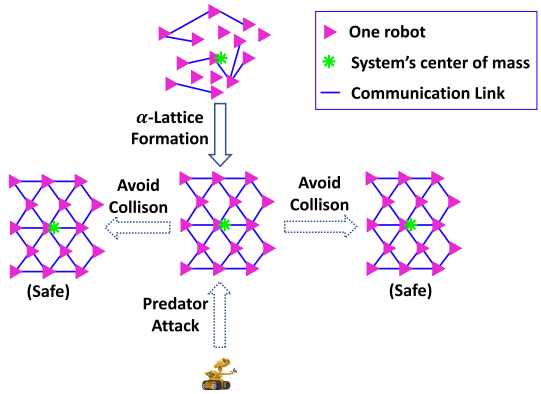

Fig. 2. Illustration of α-lattice formation in a MRS with 15 robots, and the possible motions of the system when under attack by a predator.

II. System Overview

Besides ensuring full connectivity during predator avoidance maneuver, the main goal of the proposed function approximated MARL (FA-MARL) algorithm is to consume less space for data storage compared to traditional cooperative learning. To achieve this goal, cooperative learning based on function approximation, which enables learning in a continuous form, is proposed.

The control architecture, as shown in Fig. 1, consists of two levels: In the low-level, a proportional–derivative (PD) controller drives the robots to maintain a specified formation whilst tracking a target and avoiding collision with each other and external agents. In the high-level, a function approximated reinforcement learning assigns safe place when a predator is detected and ensures full connectivity of robots which avoiding the predator.

While different formation profiles can be used, without loss of generality, this study considers an α-lattice formation with 15 robots, as shown in Fig. 2. In this paper, we consider a stationary target: the control algorithm generates stationary safe places for the system. Hence the target positions are a set of constant coordinates in space and target velocities are zero. Also, the safe places are generated by the system when a predator is detected. The safe places are generated based on the position of the predator relative to the system.

The avoidance logic is implemented such that once the predator is detected by the system, the robots in the system try to move to a safe place. However, if the predator comes into close proximity of the system, then the robots try to avoid it through the usage of repulsive artificial potential fields, under which case there is a possibility of the connectivity to be broken. Therefore, the learning and flocking algorithms should be implemented simultaneously to ensure full connectivity while also avoiding the predator. The higher level reinforcement learning module is a critical part. The goal is to achieve consensus on one of the safe places for the flocking controller. By retrieving the states and the action selection of a robot and its neighbors, the reinforcement learning module selects the appropriate action (safe place), such that the connectivity is maintained. This is depicted in Fig. 2.

III. Review of Multi-Robot Cooperative Control and Q-Learning

A. Flocking Control

Flocking control in MRS, describes the motion of each robot in the system by considering the target position, the position of its neighbours, and the position of obstacles. The dynamic model of each robot i in the system is described as

$$\begin{cases} \dot{p_i} = q_i \\ \dot{q_i} = u_i, \ i = 1, 2, \ldots, n, \end{cases} \tag{1}$$

where p_i and q_i are the position and velocity of the robot i, respectively, n is the number of robots, and u_i is the control input to robot i.

The system of all robots is modeled as a dynamic graph G with a set of vertices $V = \{1, 2, \ldots, n\}$ and an edge set $E = \{(i, j) : i, j \in V, i \neq j\}$. Each vertex in the graph corresponds to a robot in the system and each edge depicts that the two robots are connected and can communicate with each other. If the robot j is within the sensing range R_1 of robot i, then the robot j is a neighbour of robot i. For robot i in the system, the neighbourhood set at a given time t is defined as

$$N_i^{\delta}(t) = \{j \in V : ||p_j - p_i|| \leq R_1, V = \{1, 2, \ldots, n\}, j \neq i\}, \tag{2}$$

where the superscript δ indicates the actual neighbours of a robot i, and $||\cdot||$ is the Euclidean distance. Similarly, virtual neighbours can be defined in the case of predator and/or obstacle avoidance. For robot i in the system, the set of virtual neighbours set at a given time t is defined as

$$N_i^{\beta}(t) = \{k \in V_{\beta} : ||\hat{p}_{i,k} - p_i|| \leq R_2, V_{\beta} = \{1, 2, \ldots, k\}\}, \tag{3}$$

where superscript β indicates virtual neighbours, R_2 is the obstacle/predator sensing range, V_{β} is a set of obstacles, and $\hat{p}_{i,k}$ is the position of the virtual neighbour projected by robot i on predator k. These projections are used to generate repulsive forces by the robots in the system from the obstacles. Dynamic obstacles which are not stationary are considered to be predators.

Finally, the control input u_i for each robot i in the system can be expressed based on the computed parameters as

$$\begin{aligned} u_i = {} & c_1^{\delta} \sum_{j \in N_i^{\delta}} F_{\delta}(||p_j - p_i||_{\sigma}) n_{ij} + c_2^{\delta} \sum_{j \in N_i^{\delta}} (q_j - q_i) a_{ij}(p) \\ & + c_1^{\beta} \sum_{k \in N_i^{\beta}} F_{\beta}(||\hat{p}_{i,k} - p_i||) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} (\hat{q}_{i,k} - q_i) b_{i,k}(p) \\ & - c_1^t (p_i - p_t) - c_2^t (q_i - q_t), \end{aligned} \tag{4}$$

where c_1^{δ}, c_2^{δ}, c_1^{β}, c_2^{β}, c_1^t, and c_2^t are positive constants, and p_t and q_t are the target position and target velocity, respectively. In this study, these parameters denote the safe places which the agents choose to avoid the predators. More details about the flocking control in MRS can be found in [14].

B. Q-Learning

Q-learning [30] is a commonly used technique to apply reinforcement learning to a system. In this technique, the agent seeks to learn the expected reward it can get when choosing a particular action under a particular state. This value is called the state-action Q-value. During the learning phase, the agent stores the Q-values of all state-action pairs in a Q-table, which it later uses as a look-up table for taking an action. For a particular state-action pair, the update law of Q-value is given by the following equation:

$$\begin{aligned} Q^{k+1}(s, a) \leftarrow Q^k(s, a) + \alpha [r^{k+1} + \gamma \max_{a' \in A} Q^k(s,' a') \\ - Q^k(s, a)], \end{aligned} \tag{5}$$

where s and a are the current state and action respectively, r is the reward acquired by the agent, s' and a' are the state and action at the next iteration respectively, α is the learning rate, k is the iteration index, and γ is the discount factor. The details on application of cooperative learning through Q-learning can be found in [12]. In function approximation, these Q-values are approximated by a function, and this function is used to predict the expected reward for a given state-action pair. For more details, readers are referred to [37].

IV. Proposed Multi-Robot Cooperative Learning

When a team of agents are under attack by a predator, it is advantageous to move collectively to one place than move independently [27], [38]. In this section, a FA-MARL algorithm is proposed, such that when under attack, all the agents move to the same safe place and they not only avoid collision with the predator but also maintain full connectivity. Based on the movement of the predator and the choice made by its neighbours, each agent chooses an action to move to one of multiple safe places generated by the system in real-time.

A. Predator Detection

A cooperative predator detection algorithm is described, which enables collective sensing of predators in a distributed fashion within the system at each instance of time. Each robot has two sensing ranges R_1 and R_2, as shown in Fig. 3, such that

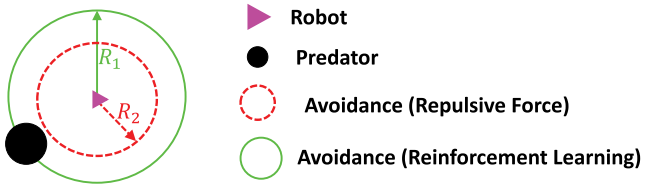

Fig. 3. Depiction of the robot with two sensing ranges R_1 and R_2.

the former is greater than the latter. The collective sensing of predator is enabled when the predator enters the sensing range R_1. At the instant shown in Fig. 3, the reinforcement learning is activated and each robot chooses a safe place. Once the predator enters the sensing range R_2, the robots start moving based on the repulsive force generated by the flocking algorithm.

1) Cooperative Predator Sensing Algorithm (CPSA): The CPSA is based on the algorithms presented in [28]. The purpose of the CPSA is two-fold: firstly, it enables a switching mechanism in the system to go between level one and level two of the control architecture. Secondly, it combines the directions of multiple predators into a single predator. At each iteration, this algorithm runs numerous sub-iterations to achieve consensus on predator sensing. This plays an important role in ensuring that all the robots in the system are in consensus of their respective states, consequently strengthening collective motion when the system is under attack.

Let the measurement, through which each robot i is able to sense the presence of predator, be a measurement m_i^l at a time-step l. Typically, this measurement is coupled with noise. To reduce the effect of noise, CPSA runs multiple sub-iterations through which a consensus on the predator position measurement is achieved. The following equation describes this operation:

$$m_i^{l,k+1} = v m_i^{l,k} + \frac{(1-v)}{|N_i^\delta|} \sum_{j=1}^{|N_i^\delta|} m_j^{l,k}, \tag{6}$$

where the index k is the index of sub-iteration at a given iteration l, $|N_i^\delta|$ is the number of neighbours of robot i, and v is a confidence weight designed such that $0 \leq v \leq 1$. Suppose that only a subset of robots are able to sense the predator. In that case, $|N_i^\delta|$ is replaced with the number of neighbours with a measurement. Under different cases,

$$m_i^{l,k+1} = \begin{cases} m_i^{l,k}, \text{if } |N_i^{\delta,m}| = 0; \\ \frac{\sum_{j=1}^{|N_i^{\delta,m}|} m_j^{l,k}}{|N_i^{\delta,m}|}, \text{if } m_i^{l,k} = NA; \\ v m_i^{l,k} + \frac{(1-v)}{|N_i^{\delta,m}|} \sum_{j=1}^{|N_i^{\delta,m}|} m_j^{l,k}, \text{if } m_i^{l,k} \neq NA, \end{cases} \tag{7}$$

where $N_i^{\delta,m}$ denotes the number of neighbours with a measurement at k-th sub-iteration, NA denotes "not available".

In this work, CPSA is used to gain consensus on presence of predator and the direction in which it is attacking, and both noisy and without noise cases are considered. (7) is capable of handling both cases. The CPSA is summarized in Algorithm 1.

Algorithm 1: Cooperative Predator Sensing Algorithm (CPSA) [28].

Set max number of iterations
for each iteration k **do**
  for each robot i **do**
    Count number of neighbours $|N_i^{\delta,m}|$
    if $|N_i^{\delta,m}| \neq 0$ **then**
      if $m_i^k = NA$ **then**
        with $m_j^k \neq NA$
        $m_i^k = \frac{\sum_{j=1}^{|N_i^{\delta,m}|} m_j^{k-1}}{|N_i^{\delta,m}|}$
      else
        $m_i^k = v m_i^{k-1} + \frac{(1-v)}{|N_i^{\delta,m}|} \sum_{j=1}^{|N_i^{\delta,m}|} m_j^{k-1}$
      end if
    else
      $m_i^k = m_i^{k-1}$
    end if
  end for
end for

B. Proposed Learning Algorithm

The proposed reinforcement learning framework is applied for predator avoidance in MRS. The current approach is based on consensus between each agent in the network and its neighbours. Before getting into the details of cooperative learning, the state, action, reward and traditional independent learning approach are described.

1) Independent Learning: For every reinforcement learning problem, the state of the learning agent, the actions it can choose, and the rewards it gets based on its current state and the action it selected are described. It is noted that the performance of the independent learning algorithm will be compared to that of the proposed cooperative learning algorithms in Section VI.

The state is mapped into a continuous space using a mapping function. The commonly used mapping functions are linear mapping, fixed sparse representation, radial basis functions and coarse coding [33], [34]. In this work, linear mapping is used and the state is described by the direction in which the predator is attacking. The direction is represented by a unit vector. This mapping is continuous and not discrete – unlike Q-learning, the direction is not categorized into four categories [12] but represented in a continuous form. The state mapping function is represented as $\psi(s)$:

$$\psi(s) = \text{CPSA}(d_1, d_2, \ldots, d_i, \ldots, d_n), \tag{8}$$

where d_i is the direction of attack of the predator detected by i-th robot in the system. The actions which an agent can choose are the safe places to which it can go to escape from the predator. In this study, four safe places in different directions are chosen, namely, left, right, up, and down. It is noted that other choices of safe places are also feasible. Hence the action set is expressed as {"1", "2", "3", "4"} , where each element in the vector denotes a safe place. In the case of function approximation, a state-action vector $\phi(\psi(s), a)$ is constructed, whose magnitude and length

are expressed as

$$|\phi(\psi(s), a)| = 1,$$

$$\text{length of } \phi(\psi(s), a) = 2 \times (\text{number of action choices}). \quad (9)$$

It is noted that in simulation, two-dimensional analysis is considered. The length of the state-action vector can be changed accordingly when higher-dimensional analysis is conducted. For example, if we have four actions and one predator, then the length of ϕ would be 8; If the direction in which the predator is attacking is $(-0.92, 0.37)$, action is "1", then the following can be obtained [30]:

$$\psi(s) = [-0.92, 0.37], \text{ and}$$
$$\phi(\psi(s), a) = [-0.92, 0.37, 0, 0, 0, 0, 0, 0].$$

The reward which a robot i gets at the end of each iteration is as follows. Once a predator gets within R_1, the RL is. triggered. At this point the reward of a robot i is the number of neighbors it has. The formation control portion of the flocking control algorithm enables the system to achieve an α-lattice structure in which, ideally, each robot has a maximum of six neighbors forming a perfect hexagonal structure around itself. Hence, in order to force the robots in the network to maintain an α-lattice formation, the maximum reward it can get is 6 – if $|N_i^\delta| \leq 6$, then $r_i = |N_i^\delta|$; otherwise, $r_i = 6$, where r_i is the reward for robot i. Using CPSA, the robots share information about the position of a predator. If a predator enters sensing range R_2 of a robot i in the system, then the robots which are directly or indirectly connected with robot i are assigned a reward of -5. If the network is fully connected, then all the robots in the system will attain this reward.

Finally, based on the state, action and reward, the independent learning algorithm is described. Each robot selects actions and updates its learning variables only based on the reward secured by itself and does not take into consideration the learning variables or actions taken by its neighbours. Function approximation is mainly used to minimize the number of variables required to be stored for the learning algorithms to work. The function approximation version of reinforcement learning consists of a parameter vector θ which is analogous to the Q-table in Q-learning [37]. Furthermore, the length of θ is equal to the length of $\phi(\psi(s), a)$. The general form of learning through function approximation is given by the following equation [37]:

$$\theta_i^{k+1} \leftarrow \theta_i^k + \alpha\phi(\psi(s_i^k), a_i^k)[r_i^{k+1} + \gamma \max_{a_i^{k+1} \in A_i} \phi^T(\psi(s_i^{k+1}), a_i^{k+1})\theta_i^k - \phi^T(\psi(s_i^k), a_i^k)\theta_i^k], \quad (10)$$

where θ_i is the parameter vector of robot i, a_i^k, a_i^{k+1} are the current and next action respectively, both belonging to the action list A_i, s_i, s_i' are the current and next states respectively.

Each robot i in the system has its own set of variables θ_i and chooses its actions based on its θ_i values for a given state-action ϕ. Eq. (10) describes how θ values in function approximation are updated at each iteration.

2) Cooperative Learning: Unlike independent learning, in cooperative learning, the θ_i-value update for each robot i takes place in two stages: In the first stage, the update is done based on Eq. (10) to obtain an intermediate update; In the second stage, the intermediate update is used along with the θ-values of its neighbours to obtain the final update. The intuition behind doing this procedure is the fact that each robot chooses an action based on its θ-vector. Hence, if consensus on action selection has to be achieved, the robot i must take into account the θ-vector which its neighbours are using to select an action. The update law is thus expressed as

$$\zeta_i^{k+1} \leftarrow \theta_i^k + \alpha\phi(\psi(s_i^k), a_i^k)[r_i^{k+1} + \gamma \max_{a_i^{k+1} \in A_i} \phi^T(\psi(s_i^{k+1}), a_i^{k+1})\theta_i^k - \phi^T(\psi(s_i^k), a_i^k)\theta_i^k],$$

$$\theta_i^{k+1} \leftarrow w\zeta_i^{k+1} + \frac{(1-w)}{|N_i^\delta|} \sum_{j=1}^{|N_i^\delta|} \theta_j^k, \quad (11)$$

where ζ_i^{k+1} denotes the intermediate update in the parameter vector for robot i at time step $k+1$, w is a weight and $0 \leq w \leq 1$. Both ζ and θ denote the same parameter vector. Different symbols are used to differentiate between intermediate update and the final update. In other words, the update for each robot is done by taking a fraction of θ-vector obtained through Eq. (10) and a fraction of the average value of θ-vectors of its neighbours. Through this mode of learning, each agent learns based on the experience of its neighbours and the experience it gained in the previous iteration. The confidence it has on its own experience compared to its neighbours' experience is determined by w.

The action selection of each robot is usually done through the greedy policy. In function approximation approach, the action which yields the maximum value of $[\phi^T(\psi(s_i^k), a_i^k)\theta_i]$ is selected. The proposed method will require significantly less data storage and number of variables when compared to Q-learning. The entire training algorithm with function approximation is summarized in Algorithm 2.

V. Convergence Analysis

In this section, we present the proof to show that the learning algorithm (Algorithm 2 in Section IV.B) converges to an optimal solution. For the learning to converge, we need to show that the system does not update anymore: i.e., the mean of the difference between θ values at time steps k and $k+1$ eventually goes to 0.

Consider a simple MRS with two robots. Without lose of generality, the findings from the analysis of this simple system can be lifted and applied to a more complex system with n robots. For simplicity, it is assumed that α, γ, w values for all robots are the same. At two consecutive time steps, the difference in θ for robot i is expressed as

$$\Delta\theta_i^{k+1} = \theta_i^{k+1} - \theta_i^k.$$

From Eq. (11), the above equation can be written as

$$\begin{aligned}\Delta\theta_i^{k+1} &= w(\zeta_i^{k+1} - \zeta_i^k) + (1-w)(\theta_j^k - \theta_j^{k-1}) \\ &= w(\Delta\theta_i^k + \alpha\phi[\Delta r_i^{k+1} + \gamma\phi^T\Delta\theta_i^k - \phi^T\Delta\theta_i^k]) \\ &\quad + (1-w)\Delta\theta_j^k.\end{aligned}$$

Algorithm 2: Proposed Training Algorithm.

Select α, γ, w values
Initialize θ values
for each episode e **do**
  Initialize robots' positions
  Initialize Predators' position and direction
  for each iteration k **do**
    for each robot i **do**
      Run CPSA
      Check predators' presence
      if Predator is present **then**
        -*Compute* $\psi(s_i^k)$ and $\phi(\psi(s_i^k), a_i^k)$
        -*Select Action* based on
        $\max[\theta^k \phi^T(\psi(s_i^k), a_i^k)]$
        -*Compute* rewards r_i^k
        -*Compute* intermediate update:
        $\zeta_i^{k+1} \leftarrow \theta_i^k + \alpha[r_i^{k+1} + \gamma \max_{a_i^{k+1} \in A_i} \theta_i^k$
        $\phi^T(\psi(s_i)^{k+1}, a_i^{k+1}) - \phi^T(\psi(s_i^k), a_i^k)\theta_i^k]$
        $\phi(\psi(s_i^k), a_i^k)$
        -*Obtain* final update based on neighbours:
        $\theta_i^{k+1} \leftarrow w\zeta_i^{k+1} + \frac{(1-w)}{|N_i^\alpha|} \sum_{j=1}^{|N_i^\alpha|} \theta_j^k$
      end if
    end for
  end for
Training is terminated when θ values converge
end for

Since r_i^{k+1} and r_i^k will be the same as an optimal policy is selected, Δr_i^{k+1} will be zero. Further simplification leads to the following equation:

$$\Delta\theta_i^{k+1} = w(I - \alpha\phi\phi^T + \alpha\gamma\phi\phi^T)\Delta\theta_i^k + (1-w)\Delta\theta_j^k,$$

where I is the identity matrix with dimensions suitable to perform matrix multiplication with $\Delta\theta_i^k$. Due to consensus, the state-action vector ϕ will be same for all robots. The equation for robot j can be written as

$$\Delta\theta_j^{k+1} = w(I - \alpha\phi\phi^T + \alpha\gamma\phi\phi^T)\Delta\theta_j^k + (1-w)\Delta\theta_i^k.$$

For the learning algorithm to converge, it is sufficient to show that the sum of $\Delta\theta$ values of all robots in the system goes to 0. This value is denoted by $\Delta\Theta$, and

$$\Delta\theta_i^{k+1} + \Delta\theta_j^{k+1} = (I - w\alpha\phi\phi^T + w\alpha\gamma\phi\phi^T)(\Delta\theta_i^k + \Delta\theta_j^k),$$
$$\Delta\Theta^{k+1} = (I - w\alpha\phi\phi^T + w\alpha\gamma\phi\phi^T)\Delta\Theta^k,$$
$$\Delta\Theta^{k+1} = \Phi\Delta\Theta^k. \tag{12}$$

Eq. (12) forms a dynamic map for the variable Θ with Φ being the state transition matrix. For the map to be stable, the absolute values of the eigenvalues of Φ must be less than or equal to 1.

$$|1 - w\alpha + w\alpha\gamma| \leq 1,$$
$$\gamma \leq 1. \tag{13}$$

Eq. (13) illustrates the condition for the learning to converge. Besides this condition, one other important condition to ensure learning convergence is consensus on states. Hence, the application of CPSA becomes critical. As stated earlier, without lose of generality, Eq. (13) can be derived for system with n robots with any type of state-mapping function ψ.

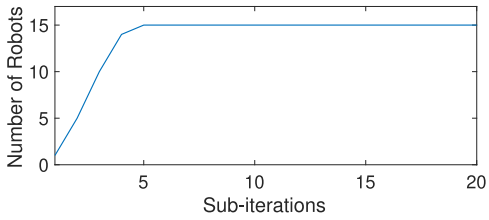

Fig. 4. The number of robots with information about the predator at every sub-iteration.

VI. Simulation Results

This section comprises of the results obtained through simulation. The results include the performance analysis of CPSA and cooperative learning and comparison between independent learning and cooperative learning in terms of convergence of rewards obtained. First the performance of CPSA is discussed. Fig. 4 shows how the CPSA works in enabling all the robots in the system to stay informed about the presence of a predator and its direction of attack, by running a predefined number of sub-iterations at each iteration. The X-axis of the plot is the number of sub-iterations at the instant when one of the robots detects the predator. Once the predator enters the predator sensing range R_1 of any of the robot in the system, CPSA starts running predefined number of sub-iterations at the instant. At the first sub-iteration, only one robot is aware of the predator's presence, this is denoted by the 'on' signal represented by '1'. In the subsequent 2-nd to 4-th iterations, all robots become aware and their signal changes from '0' to '1'. As it can be seen after the 5-th iteration, all robots are aware of the predator.

Fig. 5(a) shows the action selection of the robots during a sample trial. The sum of the actions taken by each robot is plotted against iteration. For example, if all the 15 robots choose action "3", then the sum of all the actions would be 45. The training consists of approximately twelve cycles of episodes, each cycle consisting of four episodes. Hence a total of 50 episodes are run. The four episodes in each cycle are designed such that in each of these four episodes, the predator attacks the system in a different direction. Each episode consists of 800 iterations. The total number of episodes required for the system is selected as a large number. As it can be noticed from the Fig. 5, the cooperative learning converges on optimal action selection such that it avoids the predator irrespective of which direction it is attacking. The independent learning never selects an optimal action within the range of chosen episodes.

Figs. 5(b) and 6 show the $\Delta\theta$ values at the end of each iteration and the rewards attained by the system at the end of each episode. The data presented was obtained by running 100 trials, each trial consisting of 50 episodes. For each trial, the initial learning variables were chosen randomly and at the beginning of each episode, the initial position of the robots

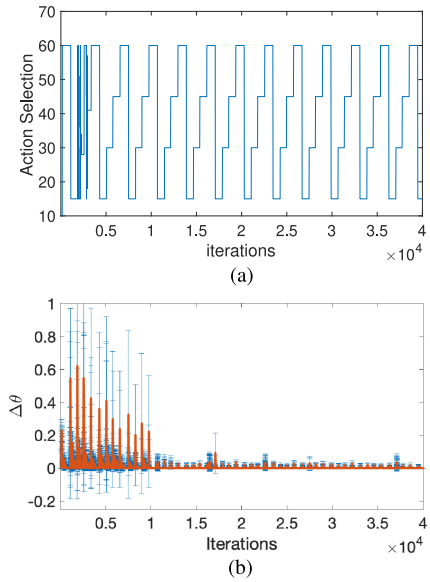

Fig. 5. (a) Sum of action selection of all robots at the end of each iteration with Cooperative Learning (b) Mean $\Delta\theta$ values at the end of each iteration with Cooperative Learning.

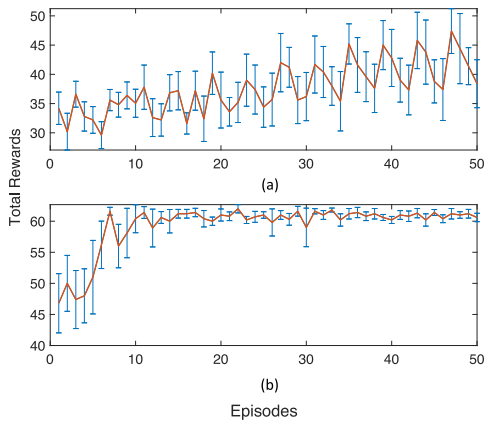

Fig. 6. Total reward attained at the end of each episode with (a) Independent Learning and (b) Cooperative Learning.

was assigned randomly. The results show how our algorithm is robust to different variations coming in the form of different initial positions during different episodes and different initial learning variables values for different trials. It shows that the proposed method's parameters converge at around 15 episodes. It was observed that the independent learning variables exhibited evident fluctuations over different episodes. This is likely because the independent learning approach requires additional constraints, such as a central unit, to assign rewards based on the overall state of the system, a different reward scheme, more initial exploration or more training scenarios. The current reward scheme scrutinizes the agent if it does not avoid collision. Out of the available safe-places, for a given predator direction, there are more than one optimal safe-places. Under these circumstances, without the consensus term, it is highly unlikely for the agents to agree upon one safe-place for a given predator direction.

To further validate the FA-MARL approach, the proposed algorithm is tested under multiple scenarios, including single predator attacking in directions different from that in training, two predators continuously attacking the system, noisy predator-detection measurements and two predators attacking the system at the same instant of time. While details can be found in the submitted video, in all the cases, the proposed algorithm exhibits good performance in terms of choosing an action such that the network does not break. The success rate for single predator scenario in testing was 82% with a standard deviation of 14.2% and the success rate for two-predator scenario was 71% with standard deviation of 15.1%. Often times, due to the predator moving at higher speeds, the network tends to break. In simulation, this problem can be easily solved by simply increasing the gains in Eq. (4). However, in real-time situation due to constraints on maximum energy generated by the system, there is a good chance of the network to break when the predators move at a very high speed. Based on the type of system considered in the study, the conditions leading to network breakage may be determined experimentally.

VII. Conclusion and Future Work

In this work, the proposed two-level control architecture, consisting of a FA-MARL algorithm and flocking control, is successfully applied to a MRS for predator avoidance. Analysis is conducted by simulating multiple training scenarios in which a MRS is attacked by a predator from different directions. Besides the obtained good performance, the proposed algorithm guarantees learning convergence under the conditions of state-consensus among robots and a discount factor less than or equal to 1. To achieve consensus on states, CPSA has been incorporated in the system. For testing purposes, the MRS is put in situations different from the ones in the training period. Even under untested circumstances, the system is able to perform equally well. Also, with the proposed method, the number of variables stored is a fixed number of 8, where as with Q-learning, the number of variables increase exponentially with further discretization of states and number of predators attacking the system.

The function approximation helps the system in approximating the unknown scenarios well and choose an appropriate action by eliminating the need for discretization of the state space. It also results in a low training period since not all combinations of the states in state space and the actions in action space need to be explored. The proposed algorithm shows promising performance in MRS where robots need to perform actions to achieve a common goal. We expect that the proposed approach can be conveniently applied to many other areas with little modifications, such as fire-fighting robots, surveillance and patrolling robots, search and rescue robots, and industrial robots. The limitations of this study and future potential improvements are briefly discussed:

- Firstly, the proposed algorithm enables the system to function in a continuous fashion by efficiently approximating the state space. However, the action space stills demands the need to be discretized. This limits the performance

of the system under scenarios where the network breakage becomes inevitable due to limited choices of actions. Hence, the algorithm can be further enhanced to consider continuous output actions.

- Secondly, we used less sophisticated predators for illustration purposes of our proposed controller and algorithm. If we consider more sophisticated predators which exhibit certain nature-inspired behaviors to attack the system, then the optimal solution will no longer be moving to a stationary safe-place. Hence, we plan to look at continuous action space scenario, where we consider this type of setting. For this purpose we believe actor-critic RL frameworks similar to [22] will be appropriate.

REFERENCES

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. London, U.K.: Oxford Univ. Press, 1999.

[2] H. Sridhar, G. Beauchamp, and K. Shanker, "Why do birds participate in mixed-species foraging flocks? A large-scale synthesis," *Animal Behaviour*, vol. 78, no. 2, pp. 337–347, 2009.

[3] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton, NJ, USA; Princeton Univ. Press, 2009.

[4] J. Cortes and M. Egerstedt, "Coordinated control of multi-robot systems: A survey," *SICE J. Control, Meas., Syst. Integration*, vol. 10, no. 6, pp. 495–503, 2017.

[5] H. Pham, H. M. La, D. Feil-Seifer, and M. Dean, "A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking," *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 50, no. 4, pp. 1537–1548, Apr. 2020.

[6] M. S. Innocente and P. Grasso, "Self-organising swarms of firefighting drones: Harnessing the power of collective intelligence in decentralised multi-robot systems," *J. Comput. Sci.*, vol. 34, pp. 80–101, 2019.

[7] X. Zhou, W. Weiping, T. Wang, Y. Lei, and F. Zhong, "Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11 691–11 703, Dec. 2019.

[8] P. Bedi, K. Qayum, and T. Kaur, "Home security surveillance system using multi-robot system," *Int. J. Comput. Appl. Technol.*, vol. 45, no. 4, pp. 272–279, 2012.

[9] A. Shallal, O. Ucan, A. Humaidi, and O. Bayat, "Multi-robot systems formation control with maneuvring target in system applicable in the hospitality and care-health industry of medical internet of things," *J. Med. Imag. Health Informat.*, vol. 10, no. 1, pp. 268–278, 2020.

[10] K. Castelli, A. Magdy, and H. Giberti, "Development of a practical tool for designing multi-robot systems in pick-and-place applications," *Robotics*, vol. 8, no. 3, 2019. [Online]. Available: https://www.mdpi.com/2218-6581/8/3/71

[11] A. Barrientos *et al.*, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *J. Field Robot.*, vol. 28, no. 5, pp. 667–689, 2011.

[12] H. M. La, R. Lim, and W. Sheng, "Multirobot cooperative learning for predator avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 1, pp. 53–63, Jan. 2015.

[13] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *Int. J. Adv. Robotic Syst.*, vol. 10, no. 12, 2013. [Online]. Available: https://journals.sagepub.com/doi/full/10.5772/57313

[14] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Trans. Autom. Control*, vol. 51, no. 3, pp. 401–420, Mar. 2006.

[15] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks:adaptive gradient climbing in a distributed environment," *IEEE Trans. Autom. Control*, vol. 49, no. 8, pp. 1292–1302, Aug. 2004.

[16] N. E. Leonard, D. A. Paley, R. E. Davis, D. M. Fratantoni, F. Lekien, and F. Zhang, "Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in monterey bay," *J. Field Robot.*, vol. 27, no. 6, pp. 718–740, 2010.

[17] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Automat.*, vol. 20, no. 2, pp. 243–255, Apr. 2004.

[18] V. Tereshchuk, J. Stewart, N. Bykov, S. Pedigo, S. Devasia, and A. G. Banerjee, "An efficient scheduling algorithm for multi-robot task allocation in assembling aircraft structures," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3844–3851, Oct. 2019.

[19] M. Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *Proc. Int. Conf. Mach. Learn.*, 1993, pp. 330–337.

[20] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Part C (Appl. Rev.)*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[21] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6244–6251.

[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[23] D. Xiao and A.-H. Tan, "Cooperative reinforcement learning in topology-based multi-agent systems," *Auton. Agents Multi-Agent Syst.*, vol. 26, pp. 86–119, 2013.

[24] T. Chu, J. Wang, L. Codeca, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.

[25] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, 2012.

[26] D. L. Cruz and W. Yu, "Path planning of multi-agent systems in unknown enviromment with neural kernal smoothing and reinforcement learning," *Neurocomputing*, vol. 233, pp. 34–42, 2017.

[27] P. Sunehag, G. Lever, S. Liu, and J. Merel, "Reinforcement learning agents acquire flocking and symbiotic behaviour in simulated ecosystems," in *Proc. Conf. Artif. Life*, 2019, pp. 103–110.

[28] H. M. La and W. Sheng, "Distributed sensor fusion for scaled field mapping using mobile sensor networks," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 766–778, Apr. 2013.

[29] X. Yin and D. Yang, "Q value reinforcement learning algorithm based on multi agent system," *J. Phys.*, vol. 1069, 2018. [Online]. Available: https://www.hindawi.com/journals/complexity/2018/7172614/#references

[30] S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2nd ed., Cambridge, MA, USA: MIT Press, 2017.

[31] A. Oroojlooyjadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," 2019, *arXiv:1908.03963*.

[32] M. Assran, J. Romoff, N. Ballas, J. Pineau, and M. Rabbat, "Gossip-based actor-learner architectures for deep reinforcement learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019, pp. 13299–13309.

[33] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "Multi-agent behavioral control system using deep reinforcement learning," *Neurocomputing*, vol. 359, pp. 58–68, 2019.

[34] Y. Wang *et al.*, "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39 974–39 982, 2019.

[35] X. Xu, L. Zuo, and Z. Huang, "Reinforcement learning algorithms with function approximation: Recent advances and applications," *Inf. Sci.*, vol. 261, pp. 1–31, 2014.

[36] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5872–5881.

[37] L. Busoniu, R. Babuska, D. B. Schutter, and D. Ernst, *Reinforcement Learning and Dynamic Programming Using Function Approximators*. Boca Raton, FL, USA: CRC Press, 2010.

[38] J. Krause, R. G. D., and D. Rubenstein, "Is there always an influence of shoal size on predator hunting success?" *J. Fish Biol.*, vol. 52, no. 3, 2005, pp. 494–501.