

Cooperative Flocking and Learning in Multi-Robot Systems for Predator Avoidance

Hung Manh La, Ronny Salim Lim, Weihua Sheng, Jiming Chen

Abstract—In this paper we propose a hybrid system that integrates reinforcement learning and flocking control in order to create an adaptive and intelligent multi-robot system. First, we present a flocking control algorithm that allows multiple mobile robots to move together while avoiding obstacles. Second, we propose a distributed cooperative learning algorithm that can quickly enable the mobile robot network to avoid predator/enemy while maintaining the network connectivity and topology. The convergence of the cooperative learning algorithm is discussed. As a result, the hybrid system of flocking control and cooperative reinforcement learning results in an efficient integration of high level behaviors (discrete states and actions) and low level behaviors (continuous states and actions) for multi-robot cooperation. The simulations are performed to demonstrate the effectiveness of the proposed system.

Keywords: Reinforcement learning, Flocking control, Multi-robot systems.

I. INTRODUCTION

Robot teams have great potentials in performing tasks such as reconnaissance, surveillance, mine field clearance and security. When a team of robots are deployed for such missions the enemy may react to attack the robot team. This may cause the robot network to break up. In this scenario, the robot team should have the ability to avoid such enemies or predators. It is also desirable that the robot team can avoid predators while maintaining the network topology and connectivity. From biology we learn that there is an effective anti-predator function in animal aggregations [1], where the fish schools and bird flocks move together to create a sensory overload on the predator's visual channel. Our paper focuses on the distributed decision making problem: given that each individual robot has a number of options (safe places) to choose from, how should each robot move to avoid network breakup when the predators appear? Often in these decisions there is a benefit for consensus, where all individuals choose the same safe place. However, the consensus methods proposed in [2], [3] require a connected network in which all robots can communicate with each other. This may not be valid in real environments because some robots may not

This project is supported by the Vietnamese government under 322 project of the Ministry of Education and Training, and the USA Department of Defense under DoD ARO DURIP grant 55628-CS-RIP, the DEPSCoR grant W911NF-10-1-0015.

Hung Manh La and Ronny Salim Lim are with the Center for Advanced Infrastructure and Transportation, Rutgers University, Piscataway, NJ 08854, USA, emails: hung.la11@rutgers.edu, ronny.lim@rutgers.edu

Weihua Sheng is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA, email: weihua.sheng@okstate.edu

Jiming Chen is with the State Key Lab. of Industrial Control Technology, Department of Control, Zhejiang University, Hangzhou, P.R.China, 310027, email:jmchen@ieee.org

connect to the network during the escape. In that case the consensus algorithms will fail. Therefore, in this paper we are interested in the problem of reaching consensus even when the robots cannot connect to the network intermittently, but they can still make right decisions through learning from experience. Our method is based on a novel integration of flocking control [4] and reinforcement learning [5].

The rest of this paper is organized as follows. Section II presents the overview of the cooperative learning system. We discuss the basics of flocking control in Section III. Section IV presents the model of multi-robot learning and then proposes a cooperative learning algorithm. Section V analyzes the convergence of the proposed learning algorithm. Section VI shows the simulation and experiment results. Finally, conclusion and future work of this paper are given in section VII.

II. SYSTEM OVERVIEW

The goals of the proposed multi-robot learning system are:

- to allow the robots to learn in continuous states and actions.
- to coordinate the concurrent learning process to generate an efficient control policy in a distributed fashion.

With regard to the limitation of discrete and finite space, we propose a hybrid system of reinforcement learning in discrete space and flocking controller in continuous space as shown in Figure 1. This control architecture has two main parts, the reinforcement learning module (high level) and the flocking controller module (low level).

The flocking controller, which works in a continuous space, is the network controller that controls all robots to move together without collision and track a stationary or moving target. In general, the target (q_t, p_t) is defined as follows

$$\begin{cases} \dot{q}_t = p_t \\ \dot{p}_t = f_t(q_t, p_t) \end{cases} \quad (1)$$

In this paper we only consider a stationary target (a fixed point or safe place). Then q_t and p_t are considered to be constant vectors. When the predator is detected, several safe places $(q_{t_1}, q_{t_2}, \dots, q_{t_N}, N \in \mathbb{Z})$ are generated by the prey. These safe places are generated based on the moving direction of the predator to maximize the escaping probability.

The flocking controller also allows the robots to avoid the predators based on a repulsive force generated from an artificial potential field induced by the predators. However, this repulsive force usually breaks up the network. Therefore, we need combine both flocking control and reinforcement

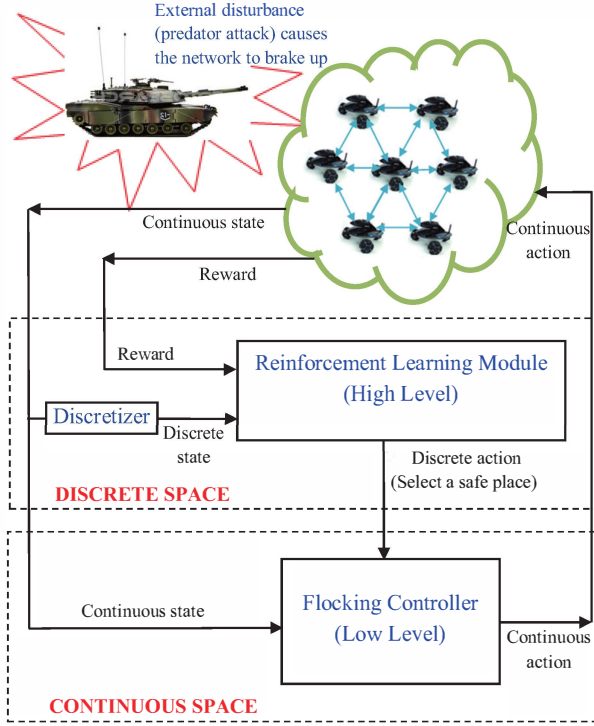


Fig. 1. The hybrid system for reinforcement learning and flocking control in multi-robot domain.

learning so that they can avoid the predators while maintaining network formation (topology) and connectivity.

The reinforcement learning module, which works in a discrete space, is the key to the controller. The goal is to agree on one of the safe places for the flocking controller. By retrieving the states and the rewards, the reinforcement learning module finds the appropriate safe place so that the network topology and connectivity can be maintained.

Our framework also works in real situations when the predators continuously attack the prey network, and the prey can learn this behavior of the predators over the attacks. Since all robots in a cooperative multi-robot system can influence each other, it is important to ensure that the actions selected by the individual robots result in effective decisions for the whole group.

III. MULTI-ROBOT COOPERATIVE CONTROL

In this section we present the distributed flocking control algorithm which allows the robots to move together without collision, avoid obstacles and track a target. First, we consider n robots moving in an two dimensional Euclidean space. The dynamic equations of each robot are described as:

$$\begin{cases} \dot{q}_i = p_i \\ \dot{p}_i = u_i, \quad i = 1, 2, \dots, n. \end{cases} \quad (2)$$

where $q_i, p_i \in R^2$ are the position and velocity of robot i , respectively, and u_i is the control input of robot i .

The topology of flocks is considered as a dynamic graph G consisting of a vertex set $\vartheta = \{1, 2, \dots, n\}$ and an edge set $E \subseteq \{(i, j) : i, j \in \vartheta, j \neq i\}$. In this topology each vertex

denotes one member of the flock, and each edge denotes the communication link between two members.

The neighborhood set of robot i at time t is defined as

$$N_i^\alpha(t) = \{j \in \vartheta : \|q_j - q_i\| \leq r, \vartheta = \{1, 2, \dots, n\}, j \neq i\}, \quad (3)$$

where r is the active/sensing range of each robot. The superscript α indicates the actual neighbors (α neighborhood robots) of robot i that is used to distinguish with virtual neighbors (β neighborhood robots) in the case of obstacle avoidance.

The set of β neighbors (virtual neighbors) [4] of robot i at time t with k obstacles is

$$N_i^\beta(t) = \left\{ k \in \vartheta_\beta : \|\hat{q}_{i,k} - q_i\| \leq r', \vartheta_\beta = \{1, 2, \dots, k\} \right\} \quad (4)$$

where r' is the obstacle detection range. ϑ_β is a set of obstacles. $\hat{q}_{i,k}$ is the position of robot i projecting on the obstacle k . The virtual neighbors are used to generate the repulsive force to push the robots away from the obstacles. The moving obstacles are considered as the predators.

The distributed flocking control algorithm is presented as follows:

$$\begin{aligned} u_i = & c_1^\alpha \sum_{j \in N_i^\alpha} \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij} + c_2^\alpha \sum_{j \in N_i^\alpha} a_{ij}(q)(p_j - p_i) \\ & + c_1^\beta \sum_{k \in N_i^\beta} \phi_\beta(\|\hat{q}_{i,k} - q_i\|_\sigma) \hat{n}_{i,k} \\ & + c_2^\beta \sum_{k \in N_i^\beta} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ & - c_1^t (q_i - q_t) - c_2^t (p_i - p_t) \\ & - c_1^{mc} (\bar{q}_{(N_i^\alpha \cup \{i\})} - q_t) - c_2^{mc} (\bar{p}_{(N_i^\alpha \cup \{i\})} - p_t). \end{aligned} \quad (5)$$

where c_1^t, c_2^t, c_1^{mc} and c_2^{mc} are positive constants. The pair $(\bar{q}_{(N_i^\alpha \cup \{i\})}, \bar{p}_{(N_i^\alpha \cup \{i\})})$ is defined as
$$\begin{cases} \bar{q}_{(N_i^\alpha \cup \{i\})} = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} q_i \\ \bar{p}_{(N_i^\alpha \cup \{i\})} = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} p_i \end{cases}$$
 where $|N_i^\alpha \cup \{i\}|$ is the number of agents in agent i 's local neighborhood including agent i itself. The virtual point q_t is considered as the safe place where the robots need to go to avoid the predators. For more details of flocking control algorithm please refer to [4], [6].

IV. MULTI-ROBOT COOPERATIVE LEARNING

In this section we build a model of multi-robots learning to avoid predators and then develop the cooperative learning algorithm.

A. Model of multi-robot learning

The multi-robot learning problem can be illustrated in Figure 2. In this figure, the robots learn to make the same decision (select the same safe place to go) so that the network will not break up, and the network topology and connectivity can be maintained. Based on the moving direction of the predator, the safe places are realtimely generated by the prey. If the prey network reaches a common safe place and the

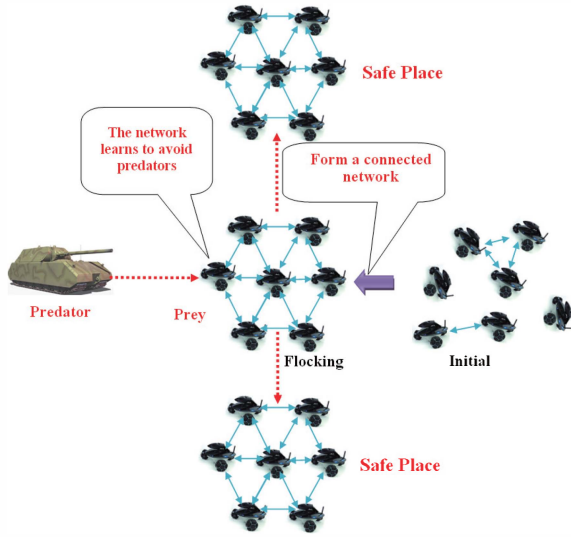


Fig. 2. Illustration of the safe places to choose.

predators keep attacking, the prey will try to find another safe place in order to escape from the predators.

When the predators attack the prey network, they typically target the center of the network. This behavior of the predators has been adopted in existing works [7], [8]. These behaviors of the predators will cause the prey network to break up. As a result, the prey will not flock or move together. This is one of the reasons that the prey have to learn in a cooperative fashion so that they can agree on the same safe place to escape from the predators [9]. Therefore, we model the prey (robots) as follows:

- All robots flock together in free space and form an α -lattice formation based on the distributed flocking control algorithm (5).
- If the predators come into the detection range (R_2), the robot (prey) can sense the location of the predators. The robot will learn and select one of the safe places to go (see Figure 3).
- If the predators come into the risk area (R_1), the robot will move away based on the repulsive force generated by the flocking control algorithm (5). Here, we can set R_1 equal to r .

B. Cooperative Learning Algorithm

In this subsection we present the cooperative learning algorithm. First, we define the state, action and reward, and then present an independent reinforcement learning algorithm. Finally, we develop a cooperative reinforcement learning algorithm based on Q-learning.

1) *State, Action and Reward*: Let the current state, action and reward of robot i be s_i, a_i, r_i , respectively, and the next state and action of robot i be s'_i, a'_i , respectively. At each moment, we have a partially observable environment. This means that not all robots are able to see the predators, and each robot only communicates with its neighbors to exchange local information. We have the following models for the state, action and reward.

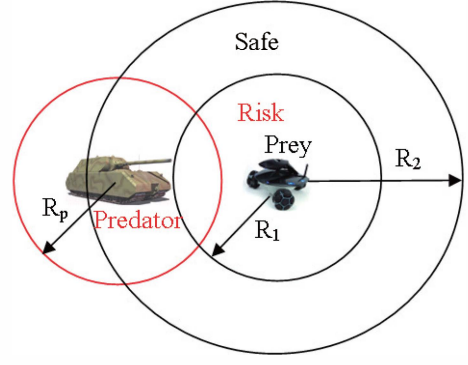


Fig. 3. Illustration of the predator and prey detection ranges. R_1 is the active range of the robot (prey), R_2 is the predator detection range, and R_p is the prey detection range.

The state: We assume that when the learning starts the state is initialized. For each robot i , the state is defined as the number of predators n_p in the detection range R_2 , the moving/attacking direction $d_p^k, k = 1, \dots, n_p$, of the predators (East, West, South, North encoded by numbers 1, 2, 3, 4, respectively), and the number of neighboring robots $|N_i^\alpha|$ in its active range r . As a result we have $s_i = [n_p, d_p, |N_i^\alpha|]^T$, $d_p = [d_p^k], k = 1, \dots, n_p$. For example, if two predators are in the detection range of robot i , one attacks the robot in the northern direction, and another one attacks in the southern direction, and six neighboring robots are in the active range of robot i , then the state for robot i is $[1, 4, 3, 6]^T$. If only six neighboring robots are in the active range, and no predator is in the detection range of robot i then the state of robot i is $[0, 0, 0, 6]^T$. If the robot i performs the action, i.e., selecting one safe place, it will keep moving until the state changes to a different state, $s'_i \neq s_i$. The maximum number of states for each robot depends on the number of the neighboring robots, predators and their attacking direction. Hence, we have the maximum number of states or the state list (S_i) of robot i is a combination of the number of the robots, predators and their attacking direction.

The action: We assume that the predators can come from any direction. However, when they detect the prey they try to move to the center of the prey network. Therefore, the desired action of the prey (robots) is to go to one of four safe places to escape. If we encode 4 safe places as numbers 1, 2, 3, 4, we have the action list for each robot $A_i = [1, 2, 3, 4]$. When the predators enter the risk area, the robot will generate the repulsive force to move away from them. The action, selecting one of the safe places, is generated in the reinforcement learning module. Then, this action is implemented in the flocking controller.

The reinforcement reward: The reinforcement reward signal changes in the experiments, depending on the input data that is received. In α -lattice configuration (hexagonal lattice configuration), a robot inside the network has six neighbors, and the robot on the border of the network has one to five neighbors. Our purpose is to maintain this network configuration, hence we define the reward as follows: if $|N_i^\alpha| < 6$ then $r_i = |N_i^\alpha|$, else $r_i = 6$ (keep an hexagonal lattice

configuration). This reward definition basically implies that the maximum reward for each agent is six which corresponds to the hexagonal lattice configuration of the network.

2) *Independent Learning*: For comparison purpose, we implement an independent learning algorithm [10] in which the robots ignore the actions and rewards of other robots, and learn their strategies independently. Each robot stores and updates an individual table, Q_i , as follows:

$$Q_i^{k+1}(s_i, a_i) \Leftarrow Q_i^k(s_i, a_i) + \alpha[r_i^k + \gamma \max_{a'_i \in A_i} Q_i^k(\acute{s}_i, \acute{a}_i) - Q_i^k(s_i, a_i)], \quad (6)$$

where α is a learning rate, and γ is a discounting factor, \acute{a}_i is the next action belonging to the action list A_i , and \acute{s}_i is a next action belonging to the action list S_i .

3) *Cooperative Learning*: In a cooperative fashion, the learning algorithm has two phases. The first phase is Q value update. The second one is action selection. In the first phase we let each robot collect its own Q value based on its action/state and its neighbor's actions/states. In the second phase, the action selection is based on the maximum Q value approach [5].

Q Value Update:

Our purpose is to allow each robot to aggregate the information of its neighbors via the Q value. Therefore each robot updates its Q value based on the following equation:

$$Q_i^{k+1}(s_i, a_i) \Leftarrow w Q_i^k(s_i, a_i) + (1-w) \sum_{j=1}^{|N_i^*|} Q_j^k(s_j, a_j), \quad (7)$$

where $Q_i^k(s_i, a_i)$ is computed based on Equation (6), and $|N_i^*|$ is the number of neighbors of robot i . w is the weight, and it is designed as $0 \leq w \leq 1$. We can clearly see that if $w = 1$, the robot only updates its Q value by itself (trusts itself), and this is exact the independent learning algorithm as discussed above (see Equation (6)). If $w = 0$ the robot updates its Q value based on its neighborhood Q values only.

Action Selection Strategy:

Usually the next action selection in reinforcement learning is based on the Boltzmann action selection strategy [11] or the maximum Q value [12], [5]. Therefore we can select the next action for the learning algorithm based on the maximum Q value as

$$a_i \equiv \max_{a_i \in A_i} Q_i(s_i, a_i), \quad (8)$$

where $Q_i(s_i, a_i)$ is computed using Equation (7).

From Equation (7) we can see that the experiences of neighboring prey/robot can be incorporated into the Q-value of robot i through a weighted linear combination of their state values. In this manner, each agent can learn from the previous experiences of its neighbors. For example, if robot A detects the predator it selects the safe place 1 to escape. This action may cause the robot to break out of the network and consequently gets the reward of 0 (no neighbor, $|N_i^\alpha| = 0$). It saves this experience into its state value. When the same scenario occurs to robot B, and robot A is within its neighborhood, robot B will have low tendency to choose the same action to escape the predator because the experience of

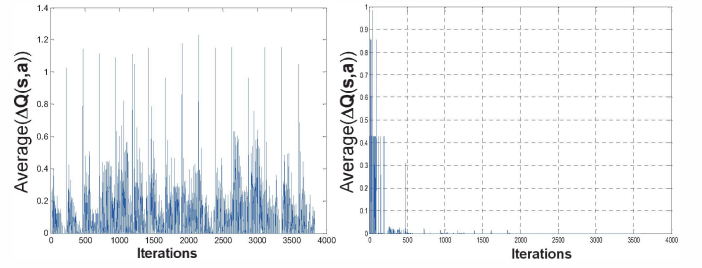


Fig. 4. Convergence of Q values for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).

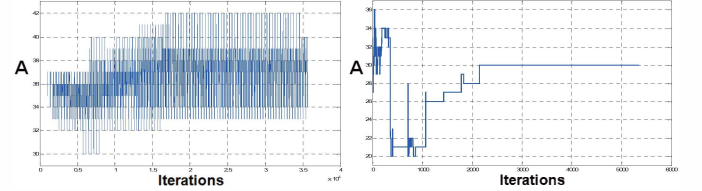


Fig. 5. Action selection evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right). A is computed by Equation (11).

robot A has been used to update the current state of robot B. Therefore, robot B could learn faster than the robots under the independent Q-learning approach.

V. CONVERGENCE OF THE SYSTEM

In this section we briefly discuss the convergence of the system. The overall idea here is to show the learning converges, that is, all prey/agents agree on the same action/safe place to go. If all prey agree on the same action the flocking control algorithm can lead all prey to move together while maintaining the network connectivity and topology.

The convergence of the flocking control algorithm (5) is already discussed in our previous work [6]. Basically, by applying the distributed control algorithm (5), the center of mass (CoM) of positions and velocities of all agents in the network will exponentially converge to the target. In addition, the formation of all mobile agents will maintain in the process of the target tracking. Therefore we just need to show the convergence of our proposed cooperative learning algorithm. We also show some experimental results of the convergence of our proposed system.

First, we evaluate the convergence of the proposed system based on the average values of the $\Delta Q(s, \mathbf{a})$ of all $\Delta Q_i = Q_i^{k+1}(s_i, a_i) - Q_i^k(s_i, a_i)$. The idea is to show the Q value of the reinforcement learning agent converges to the optimal Q^* value defined by Bellman equation in the stochastic case:

$$Q^*(s, a) = R(s, a) + \gamma \max_a \sum_{\acute{s}} P(\acute{s}|s, a) Q^*(\acute{s}, a), \quad (9)$$

where $R(s, a)$ is the reward for taking the action a giving the highest expected return, and $P(\acute{s}|s, a)$ is the state transition probability, which is the probability that the agent will be at state \acute{s} after executing action a at state s .

Theoretically we should show the expectation $E(Q_i^{k+1}(s_i, a_i))$ converges to the optimal $Q^*(s, a)$ value

as defined in Equation (9). However, for simplicity here we just show the deterministic case in which $Q_i^{k+1}(s_i, a_i)$ converges to $Q^*(s, a)$ defined as

$$Q^*(s, a) = R(s, a) + \gamma \max_a Q^*(s, a). \quad (10)$$

Hence, we can state that if the average of $\Delta \mathbf{Q}(s, \mathbf{a})$ goes to zero the proposed system is stable. From the result of the average of $\Delta \mathbf{Q}(s, \mathbf{a})$ for 15 agents as shown in Figure 4 we can see that for our proposed cooperative learning algorithm the average of $\Delta \mathbf{Q}(s, \mathbf{a})$ goes to zero after 2000 iterations while for the independent learning algorithm it does not converge to zero.

Second, we show the action performance of all agents which is evaluated based on the encoded values for the agent action selection. The total action value is computed as

$$A = \sum_{i=1}^n E_v(a_i), \quad (11)$$

where $E_v(a_i)$ is the encoded value corresponding to the selected action a_i . For example if we encode 4 safe places (4 actions) by numbers 1, 2, 3, 4, and assume that all agents select the same action 3 or $E_v = 3$, then the total valued action A of 15 agents is 45. As another example, if 6 prey choose action 1, and others choose action 2, then $A = 6 \times 1 + 9 \times 2 = 24$.

From the result in Figure 5 we can see that with our proposed cooperative learning algorithm all robots agree on the same action, which is encoded as a value of 2 after 2000 iterations (2 learning episodes). Therefore the summation of action values of 15 individuals equals to 30. This means that all prey agreed on the same action. However with the independent learning algorithm the summation of action values of individuals does not converge to a stable value. The prey change their action/decision over time, therefore the network is broken.

VI. SIMULATION RESULTS

In this section we test our cooperative learning algorithm combined with the distributed flocking control algorithm. We compare the proposed cooperative learning algorithm with the independent learning algorithm (6) in term of connectivity, convergence, action and reward. In this simulation we use 15 robots (prey), and 4 actions (4 safe places). This results in $4^{15} = 1073741824$ (≈ 1 billion) possible joint actions.

In each learning episode we randomly set up initial deployments of the prey; locations of obstacles; as well as trajectories and initial locations of the predator. The learning task is to find out one of four optimal joint actions. The parameters of flocking control are as follows: the desired distance among the prey $d = 16$; the scaling factor $k_c = 1.2$; the active range $r = k_c \times d = 19.2$; the constant $\epsilon = 0.1$ for the σ -norm. The parameters of the learning algorithm are as follows: updated weight $w = 0.5$, learning rate $\alpha = 0.2$ and discount factor $\gamma = 0.9$.

Figure 6 shows the result of trajectory and velocity profiles of the prey network in the first training episode. It can

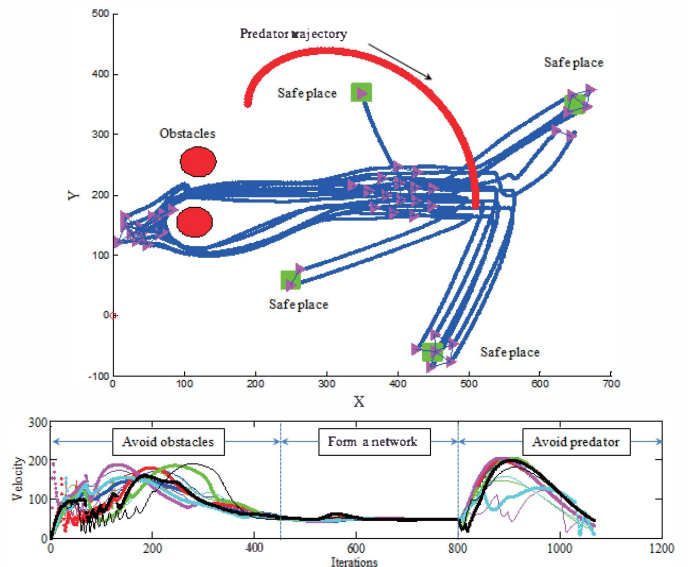


Fig. 6. Top: Trajectory/position profile of 15 agents in the first learning episode. Four green squares are four safe places. The filled red circles are the obstacles. The red trajectory is the position profile of the predator. Bottom: Velocity profile of 15 agents during the first learning episode.

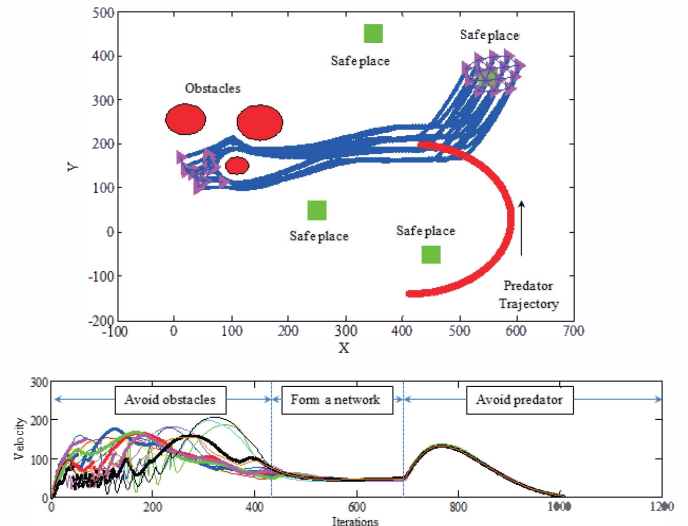


Fig. 7. Top: Trajectory/position profile of 15 agents in the third learning episode. Four green squares are four safe places. The filled red circles are the obstacles. The red trajectory is the position profile of the predator. Bottom: Velocity profile of 15 agents during the third learning episode.

be seen that at the beginning (iteration 1 to 400) the prey have to avoid obstacles, hence their velocity does not match, as shown in Figure 6 (Bottom). After avoiding obstacles the prey form a network of lattice configuration, and their velocities match (iteration 400 - 800). When the predator appears and attacks the prey network (iteration 800 - the end), each prey tries to escape by selecting its safe place. At this moment the prey increase velocity to escape the predator, and since the prey's behavior changes over time the velocity does not match. This happens because the prey do not have much learning experience. They fail to agree on the same action. Hence, the network is broken.

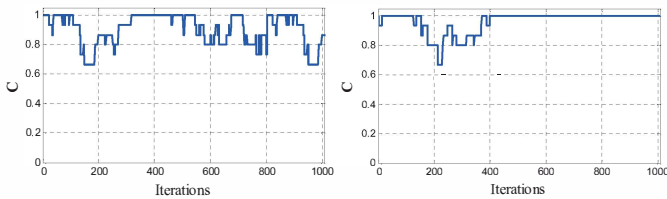


Fig. 8. Connectivity evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right) at the third learning episode. The criteria used to evaluate connectivity is referred to [13]. If $0 \leq C(t) < 1$ the network is broken, and if $C(t) = 1$ the network is connected.

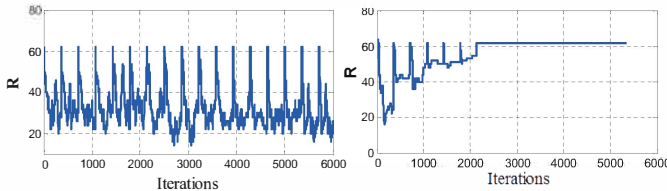


Fig. 9. Global reward evaluation for the independent learning algorithm (left) and our proposed cooperative learning algorithm (right).

In the second learning episode, we observed that the learning result is better since more than 50 percent of the prey agree on the same safe place to go. In the third episode (see Figure 7) the learning converges and all the prey select the same action and have the same velocity. Therefore the connectivity is maintained. We can see that even when the trajectories of the predator and the location of the obstacles change, the learning results still hold.

The connectivity of the network evaluation is shown in Figure 8. We can see that for the cooperative learning algorithm the connectivity is maintained while for the independent learning algorithm it is not maintained. This means that the robots do not agree on the same action. Note that in Figure 8 (right) the connectivity is only lost from iteration 1 to 400 because the prey have to avoid the obstacles. After that the predator appears, and the prey can avoid it and maintain the connectivity based on the proposed cooperative learning algorithm. In contrast, the connectivity is lost for the independent learning algorithm as shown in Figure 8 (left).

The global reward is computed as $R = \sum_{i=1}^n r_i$. From the result in Figure 9 with our proposed cooperative learning algorithm we can obtain a maximum global reward with a value of 62 after about 2000 iterations, but with the independent learning algorithm the global reward does not converge to a stable value.

VII. CONCLUSION AND FUTURE WORK

In this paper we proposed a hybrid system that integrates flocking control and reinforcement learning to allow robots to behave intelligently for predator avoidance. Reinforcement learning is developed based on the cooperative Q learning algorithm. We showed that the proposed cooperative Q learning allows the network to find out the effective joint action more quickly than the independent Q learning. This also allows the network to maintain its topology and connectivity

while avoiding the predator. Simulation results are collected to demonstrate the effectiveness of our proposed system.

REFERENCES

- [1] J. Krause, G. Ruxton, and D. Rubenstein. Is there always an influence of shoal size on predator hunting success? *Journal of Fish Biology*, 52:494–501, 1998.
- [2] L. Xiao, S. Boy, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. *Proceedings of the International Conference on Information Processing in Sensor Networks*, pages 63–70, 2005.
- [3] S. Kar and J. M. F. Moura. Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise. *IEEE Transactions on Signal Processing*, 57(1):355–369, 2009.
- [4] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.
- [5] L. Busoniu, R. Babuska, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 38(2):156–172, 2008.
- [6] H. M. La and W. Sheng. Flocking control of a mobile sensor network to track and observe a moving target. *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 3129–3134, 2009.
- [7] S. H. Lee. Predator’s attack-induced phase-like transition in prey flock. *Physical Letters A*, 357:270–274, 2006.
- [8] K. Morihiro, H. Nishimura, T. Isokawa, and N. Matsui. Learning grouping and anti-predator behaviors for multi-agent systems. *Proceedings of the International conference on Knowledge-Based Intelligent Information and Engineering Systems*, pages 426–433, 2008.
- [9] H. Milinski and R. Heller. Influence of a predator on the optimal foraging behavior of sticklebacks. *Nature* 275, pages 642–644, 1978.
- [10] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, 1998.
- [11] M. Coggan. Exploration and exploitation in reinforcement learning. *Fourth International Conference on Computational Intelligence and Multimedia Applications*, 2001.
- [12] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligent Research*, 4:237–285, 1996.
- [13] H. M. La and W. Sheng. Adaptive flocking control for dynamic target tracking in a mobile sensor network. *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), St. Louis, MO, USA, 2009*.