

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/265272507>

Hybrid System of Reinforcement Learning and Flocking Control in Multi-robot Domain

ARTICLE

CITATIONS

2

READS

33

3 AUTHORS:



[Hung Manh La](#)

University of Nevada, Reno

50 PUBLICATIONS 178 CITATIONS

[SEE PROFILE](#)



[Ronny Salim Lim](#)

Oklahoma State University - Stillwater

11 PUBLICATIONS 80 CITATIONS

[SEE PROFILE](#)



[Weihua Sheng](#)

Oklahoma State University - Stillwater

151 PUBLICATIONS 1,033 CITATIONS

[SEE PROFILE](#)

Hybrid System of Reinforcement Learning and Flocking Control in Multi-robot Domain

Hung Manh La, Ronny Lim and Weihua Sheng

Abstract—In multi-robot domain, one of the important problems is to achieve cooperation among robots. In this paper we propose a hybrid system that integrates reinforcement learning and flocking control in order to create adaptive and intelligent multi-robot systems. We study two problems of multi-robot concurrent learning of cooperative behaviors: (1) how to generate efficient combination of high level behaviors (discrete states and actions) and low level behavior (continuous states and actions) for multi-robot cooperation; (2) how to coordinate concurrent learning process in a distributed fashion. To evaluate our theoretic framework we apply it to solve the problem of how a multi-robot network learns to avoid predator while maintaining network topology and connectivity. The experiments and simulations are performed to demonstrate the effectiveness of the proposed hybrid system.

Keywords: Reinforcement learning, Flocking control, multi-robot systems, hybrid system.

I. INTRODUCTION

In multi-robot system, one of the important research problems is to achieve cooperation among robots by distributed control methodology. In recent years, machine learning techniques such as reinforcement learning have been developed for multi-robot systems to allow the robots to learn cooperation [1], [2]. However, traditional reinforcement learning assumes discrete and finite state/action spaces; therefore, it is difficult to be directly applied to most real applications that inherently involve with continuous and infinite space. Furthermore, even if the states can be discretized, the learned elementary behaviors are still discrete. In addition, the switching of discrete behaviors usually causes the control of the robots become non-smooth, which is undesirable in most applications. To tackle this issue, several methods have been proposed to make the reinforcement learning methods work in continuous environment [3], [4], [5]. However, these methods mostly focus on the approach of action approximation such as trying to estimate the state/action values based on known environments instead of using discrete lookup tables. In this paper, we propose a hybrid system that integrates reinforcement learning and flocking control. Furthermore, to evaluate our theoretic framework we apply it to solve the problem of how a mobile robot network learns to avoid predators while maintaining topology and connectivity.

From biology we known that there are several anti-predator functions in animal aggregations [6], [7], [8]. One reason why fish schools and bird flocks move together is

that it becomes difficult for predators to pick out individual prey from groups because many moving preys create a sensory overload of the predator's visual channel [6]. Based on the fact that all preys should form a group to avoid predator efficiently our paper focuses on distributed decision making problem where individuals have a number of options (safe places) to choose from. Often in these decisions there is a benefit for consensus, where all individuals choose the same of the available options. However, the consensus methods [9], [10] require a connected network in which all robots can communicate together. This may not be valid in real environments because sometimes some robots may not connect to the network. Therefore, in this paper we are interested in the problem of reaching consensus even when robots cannot connect to the network all the time, but they still can make right decisions based on the proposed reinforcement learning algorithm. We address two problems of multi-robot concurrent learning of cooperative behaviors:

- How to generate efficient combination of high level behaviors (discrete states and actions) and low level behavior (continuous states and actions) in flocking control for multi-robot cooperation.
- How to coordinate concurrent learning process in a distributed fashion.

II. FLOCKING CONTROL ALGORITHM

In this section we present the distributed flocking control algorithm. Basically this algorithm allows the robots to move together without collision, avoid obstacles and track the target. First, we consider n robots moving in an m (e.g., $m = 2, 3$) dimensional Euclidean space. The dynamic equations of each robot are described as:

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

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

To describe the topology of flocks we consider a dynamic graph G consisting of a vertex set $\mathfrak{V} = \{1, 2, \dots, n\}$ and an edge set $E \subseteq \{(i, j) : i, j \in \mathfrak{V}, j \neq i\}$. In this topology each vertex denotes one member of the flock, and each edge denotes the communication link between two members. We define a neighborhood set of robot i as follows:

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

here r is an active range (radius of neighborhood circle in the case of two dimensions, $m = 2$, or radius of neighborhood sphere in the case of three dimensions, $m = 3$), and $\|\cdot\|$ is

Hung Manh La, Ronny Lim and Weihua Sheng are with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA (hung.la, ronny.lim, weihua.sheng)@okstate.edu.

the Euclidean distance. 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) [11] of robot i at time t with k obstacles is

$$N_i^\beta(t) = \left\{ k \in \mathcal{O}_\beta : \|\hat{q}_{i,k} - q_i\| \leq r', \mathcal{O}_\beta = \{1, 2, \dots, k\} \right\} \quad (3)$$

here r' is selected to be less than r , in our simulations $r' = 0.6r$. \mathcal{O}_β is a set of obstacles. $\hat{q}_{i,k}, \hat{p}_{i,k}$ are the position and velocity of robot i projecting on the obstacle k , respectively. 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 geometry of flocks is modeled by an α -lattice [11] that meets the following condition:

$$\|q_j - q_i\| = d, j \in N_i^\alpha, \quad (4)$$

here d is a positive constant indicating the distance between robot i and its neighbor j . However, at singular configuration ($q_i = q_j$) the collective potential used to construct the geometry of flocks is not differentiable. Therefore, the set of algebraic constrains in (4) is rewritten in term of σ -norm [11] as follows:

$$\|q_j - q_i\|_\sigma = d^\alpha, j \in N_i^\alpha,$$

here the constraint $d^\alpha = \|d\|_\sigma$ with $d = r/k_c$, where k_c is the scaling factor. The σ -norm, $\|\cdot\|_\sigma$, of a vector is a map $R^m \Rightarrow R_+$ defined as $\|z\|_\sigma = \frac{1}{\varepsilon} [\sqrt{1 + \varepsilon\|z\|^2} - 1]$ with $\varepsilon > 0$. Unlike the Euclidean norm $\|z\|$, which is not differentiable at $z = 0$, the σ -norm $\|z\|_\sigma$, is differentiable every where. This property allows to construct a smooth collective potential function for robots.

The distributed flocking control algorithm proposed in our previous work [12] (this algorithm is the extension of the flocking control algorithm in [11]) controls all robots to form an α -lattice configuration. This algorithm is presented as follows:

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

In this control protocol, $c_1^\alpha, c_2^\alpha, c_1^\beta, c_2^\beta, c_1^t, c_2^t, c_1^{mc}$ and c_2^{mc} are positive constants. $\phi_\alpha(z)$ is the action function to control the attractive or repulsive forces between robot i and its neighbor j . $\phi_\beta(z)$ is the action function to control the repulsive force between robot i and its obstacle k . n_{ij} and $\hat{n}_{i,k}$ are the vectors along the line to connect the pair (q_i, q_j) , and the pair $(\hat{q}_{i,k}, q_i)$, respectively. $a_{ij}(q)$ and $b_{i,k}(q)$ are adjacency matrices, respectively. q_t and p_t are the position and velocity of the target, respectively. In this paper we consider q_t as the safe place that the robots need to go there to escape

the predators. The pair $(\bar{q}_{(N_i^\alpha \cup \{i\})}, \bar{p}_{(N_i^\alpha \cup \{i\})})$ is defined as
$$\begin{cases} \bar{q}_{(N_i^\alpha \cup \{i\})} = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} q_i \\ \bar{p}_{(N_i^\alpha \cup \{i\})} = \frac{1}{|N_i^\alpha \cup \{i\}|} \sum_{i=1}^{|N_i^\alpha \cup \{i\}|} p_i \end{cases}$$
 here $|N_i^\alpha \cup \{i\}|$ is the number of agents in agent i 's local neighborhood including agent i itself. More details of these terms, see [11], [12].

III. GENERAL FRAMEWORK OF HYBRID SYSTEM IN MULTI-ROBOT DOMAIN

Our goals in developing the cooperative learning algorithm in a distributed fashion are to:

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

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

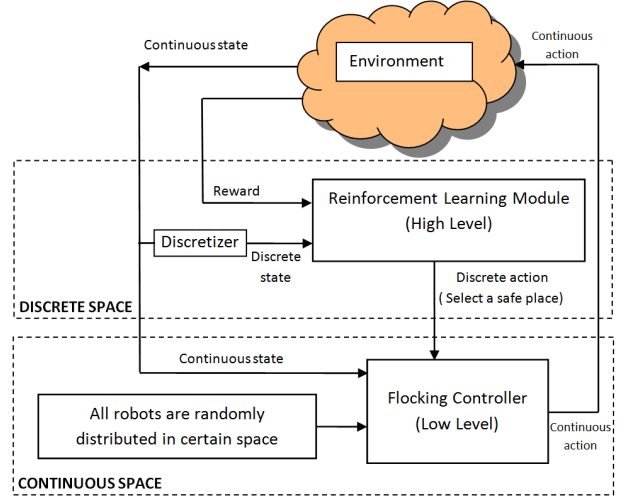


Fig. 1. The hybrid system for reinforcement learning and flocking controller in multi-robot domain.

The flocking controller (5) is the network controller to control all robots to move together without collision and track a stationary or moving target. In addition, the flocking controller allows the robots to avoid predator based on a repulsive force generated from an artificial potential field induced by the predator. However, this repulsive force can make the network break up. Therefore, we need to combine both flocking control and reinforcement learning together to allow all robots to flock together in order to avoid the predator in an efficient way while maintaining formation (topology) and connectivity.

The reinforcement learning module is the key of the controller, and it is integrated with the flocking controller. The goal is to select one of the safe places for the flocking controller to allow the prey (robot) to escape from the

predator. By retrieving the states and rewards, the reinforcement learning module allows to find the appropriate safe place for each high level behavior so that the topology and connectivity of the network can be maintained.

All robots in a cooperative multi-robot system can influence each other. Hence, it is important to ensure that the actions selected by the individual robots result in optimal decisions for the whole group. We consider the problem of computing a coordinated action for a group of n robots. Each robot i selects an individual action a_i from its action set A_i . Then, a joint action for whole group is $a = (a_1, \dots, a_n)$ which generates a reward $R(a)$ for whole team. The coordination problem is to find the optimal joint action a^* that maximizes $R(a)$, or $a^* = \arg \max_a R(a)$.

IV. MODELING AND COOPERATIVE LEARNING ALGORITHM

A. Model of multi-robot learning to avoid predator

Before we model the multi-robot learning to avoid predator we have the following assumptions:

- Each robot (prey) can sense the position and velocity of its neighbors.
- All robots can flock together (based on flocking control) with fixed topology in free space.
- Each robot can sense the location of the predator within the sensing range.

We would like to build a learning algorithm in which the robots (preys) should cooperate to avoid the predator together. We are interested in the problem of how to design an intelligent robot network to avoid predator in an efficient way. This problem can be illustrated in Figure 2. In this figure, the robots try to learn to make the same decision (select same safe place to go) so that the network will not break up, or the connectivity can be maintained.

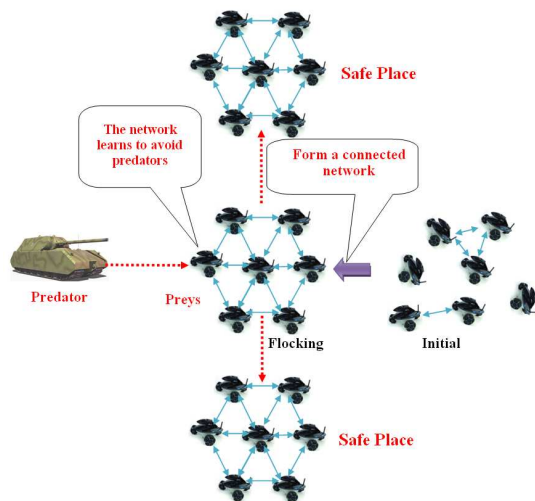


Fig. 2. Illustration of the safe places to choose.

First, we build the model of the predators as follows:

- The predator tries to go into the center of the network. This behavior of the predator is usually modeled in most situations [13], [14].

- The velocity of the predator is faster than that of the prey (robot).

These behaviors of the predator will cause the prey network to break up. As a result, the preys will not flock together [6]. This is one of the reasons that the preys have to learn in a cooperative fashion so that they can agree on the same solution such as finding the same safe place to escape the predator. From these analyses, we model the preys (robots) as follows:

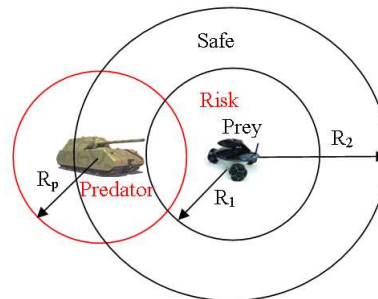


Fig. 3. Illustration of the predator and prey detection ranges. R_1 is the active range of the robot (prey), R_2 is the predator detection range, and R_p is the prey detection range.

- All robots flock together in free space and form an alpha lattice formation [11] based on the distributed flocking control law (5).
- If the predators go into the detection range (R_2), the robot should learn and select one of the safe places to go (see Figure 3).
- If the predators go into the risk area (R_1), the robot should move away based on the repulsion force.

With the built models of the predator and prey our learning goal is how the robots can cooperate together to avoid predator while they have to maintain network performance such as the connectivity and topology. To achieve this goal we propose a cooperative learning algorithm as discussed in the next subsection.

B. Cooperative Learning Algorithm

In this section we develop a cooperative reinforcement learning (Q learning) algorithm. We compare it with an independent reinforcement learning algorithm. Let the current state and action, and the reward of robot i be s_i, a_i, r_i , respectively. Then let 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 can see the predator, and each robot only communicates with its neighbors to exchange local information. We have the following models for the state, action and reward.

The state: we assume that when the learning mode starts (all robots flocked together and formed an alpha lattice formation) the state is initialized. The state is defined as the number of the predators n_p and the number of neighboring robots n_r in the viewing range of robot i , $s_i = [n_p, n_r]$. For example, if one predator and 6 neighboring robots in the viewing range of the robot i then the state for robot i is

(1, 6). If only 6 neighboring robots and no predator in the viewing range of the robot i then the state for robot i is (0, 6). If the robot i performs the action i.e. follows the virtual leader, it will keep moving until the next state changes $s_i^j \neq s_i$. The maximum number of the states depends on the number of the robots. Hence, we have the maximum total number of the states of robot i to be $(n-1, 1), (n-1, 0), (n-2, 1), (n-2, 0), \dots, (0, 1), (0, 0)$. In general, the maximum total number of the states of robot i in the case of one predator equals to $(n-1) \times 2$. Since all robots want to maintain the connection to the network, they want to avoid states (0, 1) and (0, 0).

The action: assume that the predator goes any direction in the field to attack the prey (robot) in convenient way, hence the desired actions of the robots to avoid predator are: go to one of four safe places to escape. If we encode 4 safe places as numbers 1, 2, 3, 4, we have for each robots as $A_i = [1, 2, 3, 4]$. In the case the predator enters the risk area then the robot will generate the repulsive force to keep away from the predator. Additional actions can be introduced if desired. The illustration of this scenario is shown in Figure 4. The action, selecting one of safe places,

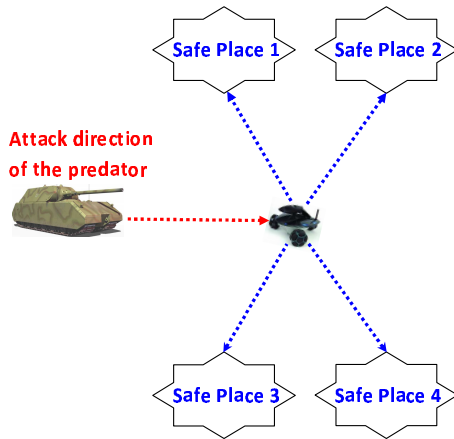


Fig. 4. The predator avoidance strategy.

is generated in Reinforcement Learning module. Then, this action is implemented in the flocking controller. Basically, it is one of four rendezvous points that all robots are desired to go there.

The reinforcement reward: the reinforcement reward signal changes in the experiments, depending on the input data that is received. Two behaviors of the robot are considered as avoiding predator and increasing the number of the neighboring robots up to a certain number. For this purpose, the reward is defined as follows:

The positive reward is given if the robot maintains the number of its neighbors. The negative reward is given if the robot loses its neighbors. Specifically, the higher negative reward is given if the robot loses more neighbors. In hexagonal lattice configuration, the robot inside the network has 6 neighbors, and the robot on the border of the network has 1 to 5 neighbors. Based on this fact we can define the reward function as follows:

```

if  $|N_i^\alpha| < 6$  then
    |  $r_i = |N_i^\alpha|$ ;
else
    |  $r_i = 6$  (keep an  $\alpha$ -lattice configuration);
    
```

The delayed reward approach has been followed, so the reward is computed after each learning trial.

1) *Cooperative Learning Algorithm Description:* In cooperative fashion, each robot collects its own Q value based on its action/state and its neighbors actions/states. The idea of this collection is illustrated in Figure (5)

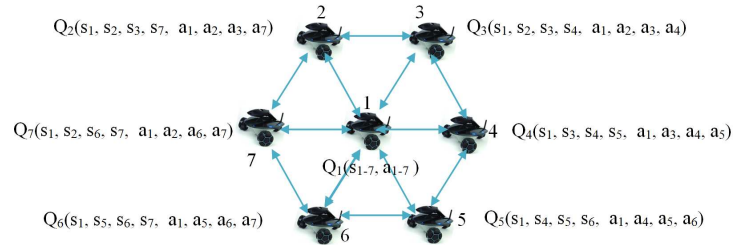


Fig. 5. Q value collection based on the robot's action/state and its neighboring actions/states.

Algorithm 1: Majority Action Following Algorithm (MAF)

```

for each robot  $i$  do
    Step 1. Ask/observe its neighbors decisions.
    Step 2. Select the action that most robots in the
    inclusive neighborhood set  $(i \cup N_i^\alpha)$  follow.
    if the number of robots in each group in the set
     $(i \cup N_i^\alpha)$  selecting the same action are the same then
        | Go to Step 3;
        | Step 3. The robot  $i$  will keep its own decision;
    else
        | Go back to Step 2.
    
```

For decision making, first each prey select next action based on maximum Q value. Then the final decision is based on Majority Action Following Algorithm (MAF) as shown in Algorithm 1. In general, we propose the cooperative learning algorithm as shown in Algorithm 2.

V. EXPERIMENT AND SIMULATION RESULTS

In this section we test our cooperative learning algorithm (Algorithm 2) combining with the distributed flocking control algorithm (5) in term of connectivity, topology and reward. To do the experiment we use 8 real robots, however for evaluating of connectivity, topology and reward, we use 15 robots in simulation and run our algorithm in several training episodes.

Experimental setup:

In this experiment we use 8 Rovio robots [15] that have omni-directional motion capability. Especially, 7 Rovio

Algorithm 2: Cooperative Learning in Distributed Fashion

 Set parameters α and γ .

Build the state list and action list for each robot

for each episode do
for each robot i do
Initialization phase:

- Initialize the matrix Q_i
- Find initial state (s_i) that corresponds with the one in the state list.
- Randomly select one of actions (a_i) in the action list.

Update phase (after robot i do the selected action):

- Update the next state (s'_i).
- Select the next action (a'_i) based on maximum Q_i value.
- Compute the reward r_i .
- Compute Q_i value:

$$Q_i(s_i, a_i) \leftarrow Q_i(s_i, a_i) + \alpha[r_i + \gamma Q_i(s'_i, a'_i) - Q_i(s_i, a_i)]$$

- Update Q_i based on its neighbors:

$$Q_i(s_i, a_i) \leftarrow Q_i(s_i, a_i) + \sum_{j=1}^{|N_i^\alpha|} Q_j(s_j, a_j)$$

- Set the next state as the current state.

Action implementation phase:

- Final action is selected based on Majority Action Following Algorithm (Algorithm 1).

end
end

robots are used as preys, and one Rovio robot is used as a predator. Basically, these robots can freely move in 6 directions. The dynamic model of the Rovio robot can be approximated by Equation (1). However, the accuracy of the localization of the Rovio robot is low, and the robot does not have any sensing device to sense the pose (position and velocity) of its neighbors or the obstacles. Hence we use a VICON system setup [16] in our lab (Figure 6) that includes 12 infrared cameras to track moving objects. This tracking system can give the location and velocity of each moving object with high accuracy.

The active range of each robot is $r = 600\text{mm}$. The desired distance among robots is $d = 550\text{mm}$, and $\epsilon = 0.1$ for the σ -norm.

Number of actions = 4 (4 safe places to escape predator). This results $4^{15} = 1073741824$ (≈ 1 billion) possible joint actions (huge search space).

In each learning episode:

- The robots have different initial deployments (randomly).
- The environments are changed (different obstacle distributions, and different trajectories and initial locations of the predator)

- The learning task is to find out one of 4 optimal joint

actions.

For example if we encode 4 safe places (4 actions) as numbers 1, 2, 3, 4, then 4 optimal joint actions are:

1111111111111111; 2222222222222222; 3333333333333333 and 4444444444444444.

This means that all preys have to agree on the same action in order to satisfy the tasks such as maintain the same topology and connectivity.

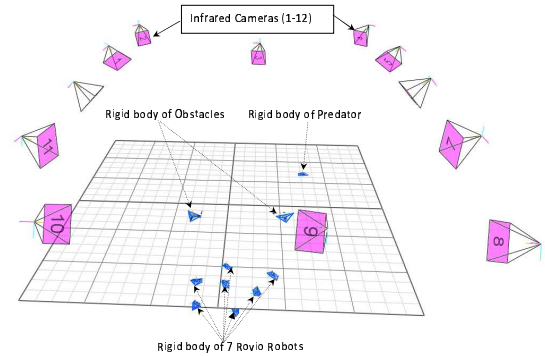


Fig. 6. Infrared cameras tracking system for experimental setup.

Figure 7 shows the result of the first training episode. Since in the first time the robots do not have any experience of the environment (the behavior of the predator), they failed to agree on the same action. Hence, the network is broken. In the second episode (it is not shown here since the space is limited), the learning result is better since more than 50 percent of the robots agree on the same safe place to go. In the third episode the learning is converged and all of the robots choose the same action (same safe place) (see Figure 8). Therefore the connectivity is maintained.

Connectivity Evaluation:

To analyze the connectivity of the network we define a connectivity matrix $[c_{ij}(t)]$ as follows:

$$[c_{ij}(t)] = \begin{cases} 1, & \text{if } j \in N_i^\alpha(t), i \neq j \\ 0, & \text{if } j \notin N_i^\alpha(t), i \neq j \end{cases}$$

and $c_{ii} = 0$. Since the rank of Laplacian [11] of a connected graph $[c_{ij}(t)]$ of order n is at most $(n-1)$ or $\text{rank}([c_{ij}(t)]) \leq (n-1)$, the relative connectivity of a network at time t is defined as: $C(t) = \frac{1}{n-1} \text{rank}([c_{ij}(t)])$. If $0 \leq C(t) < 1$ the network is broken, and if $C(t) = 1$ the network is connected. From the result in Figure 9 we can see with the

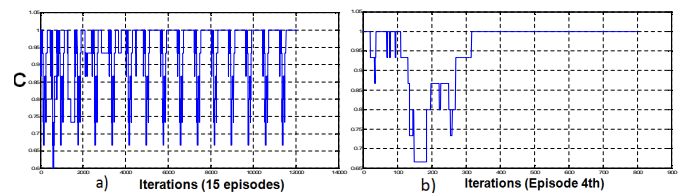


Fig. 9. Connectivity evaluation for our proposed cooperative learning algorithm (a, b), here (b) is a zoom-in at episode 4th of (a)

proposed cooperative learning algorithm the connectivity of the network is maintained after 3 training episodes.

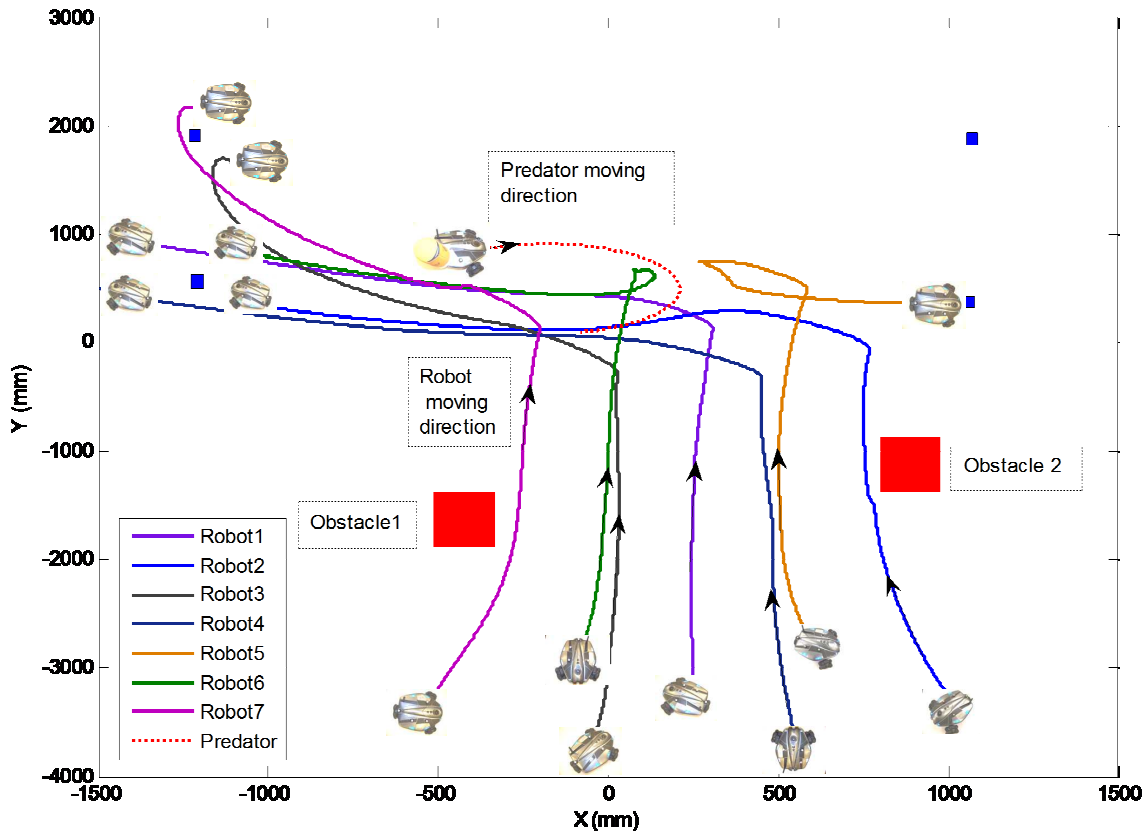


Fig. 7. The trajectories of 7 Rovio robots and one predator in the first learning episode. The green small squares are the safe places, and the filled red squares are the obstacles.

Topology Evaluation:

The topology of the network is evaluated based on the following algorithm: We clearly see that if $T = 0$ the

```

for each robot  $i$  do
  if  $|N_i^\alpha|$  changes then
    Topology:  $T_i = \text{abs}(|N_i^\alpha(k)| - |N_i^\alpha(k-1)|)$  ( $k$  is
    time step or iteration)
  else if  $|N_i^\alpha|$  does not change, but indices of  $|N_i^\alpha|$ 
  change then
    Topology:  $T_i = \text{number of index changes}$ 
  else
    Topology:  $T_i = 0$ 
    
```

For the whole network :Topology: $T = \sum_{i=1}^n T_i$

topology of the network does not change, and if $T > 0$ the topology of the network changes. In addition, if T is big the topology changes much, and otherwise it change a little. From the result in Figure 10 we can see with our proposed cooperative learning algorithm the topology of the network does not change after 3 training episodes.

Reward Evaluation:

The global reward is computed as $R = \sum_{i=1}^n r_i$. From the result in Figure 11 with our proposed cooperative learning algorithm we can obtain a maximum global reward with a

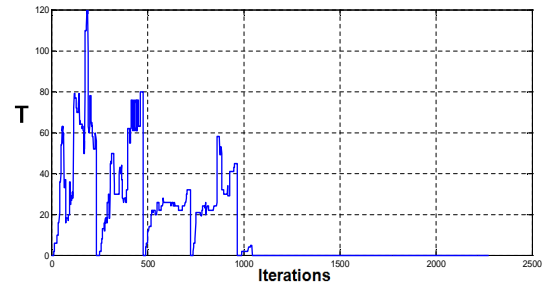


Fig. 10. Topology evaluation for our proposed cooperative learning algorithm.

value of 62 after about 2000 iterations.

For more details about these results please see some video files which are available at the following website: <http://www.hungla.org/research.php>

VI. CONCLUSION

We proposed a hybrid system that integrates flocking control and reinforcement learning to allow robots to behave intelligently in continuous space. Reinforcement learning is developed based on cooperative Q learning and Majority Action Following Strategy (MAF). We evaluated the proposed hybrid system for the case of multi-robot learning to avoid predator. We showed that the proposed cooperative Q learning allows the network to quickly find out

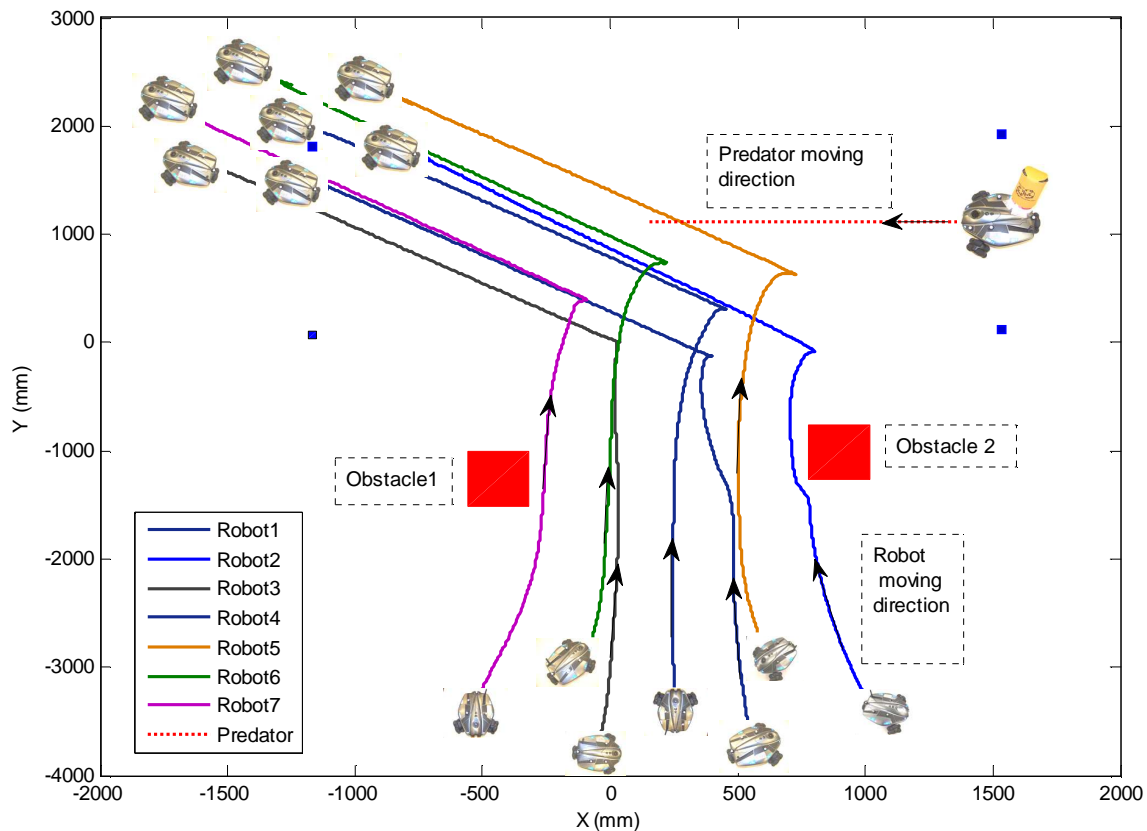


Fig. 8. The trajectories of 7 Rovio robots and one predator in the third learning episode. The green small squares are the safe places, and the filled red squares are the obstacles.

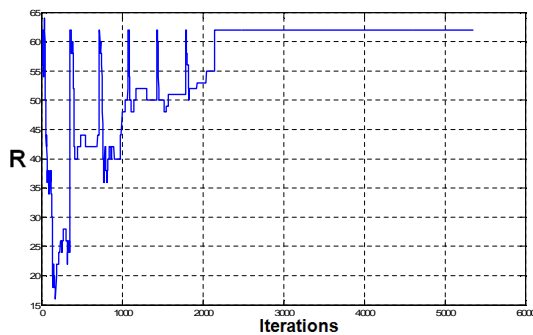


Fig. 11. Global reward evaluation.

the optimal joint action. This also allows the network to maintain its topology and connectivity while avoiding the predator. Experimental and simulation results are collected to demonstrate the effectiveness of our proposed system.

REFERENCES

[1] 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.

[2] F. Fernandez, D. Borrajo, and L. Parker. A reinforcement learning algorithm in cooperative multi-robot domain. *Journal of Intelligent and Robotic Systems*, 43:161–174, 2005.

[3] C. Gaskett, D. Wettergreen, and A. Zelinsky. Q-learning in continuous state and action spaces. *AI'09, LNAI 1747, Springer-Verlag, Berlin, Heidelberg*, pages 417–428, 1999.

[4] J. C. Santamaria, R. S. Sutton, and A. Ram. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behaviour*, 6(2):163–218, 1998.

[5] W. D. Smart and L. P. Kaelbling. Practical reinforcement learning in continuous spaces. *Proceedings of the International Conference on Machine Learning*, pages 903–910, 2000.

[6] H. Milinski and R. Heller. Influence of a predator on the optimal foraging behavior of sticklebacks. *Nature* 275, pages 642–644, 1978.

[7] G. Roberts. Why individual vigilance increases as group size increases. *Animal Behavior*, 51:1077–1806, 1996.

[8] 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.

[9] Z. Wang and D. Gu. A survey on application of consensus protocol to flocking control. *Proceedings of the 12th Chinese Automation and Computing Society Conference in the UK, England*, pages 1–8, 2006.

[10] W. Ren, R. W. Beard, and E. M. Atkins. A survey of consensus problems in multi-agent coordination. *Proceedings of the American Control Conference*, pages 1859–1864, 2005.

[11] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[12] 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*, 2009.

[13] S. H. Lee. Predator's attack-induced phase-like transition in prey flock. *Physical Letters A*, 357:270–274, 2006.

[14] 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.

[15] Rovio robot: <http://www.wowwee.com/en/support/rovio>.

[16] VICON motion system: <http://www.vicon.com/>.