Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Multi-agent Collaboration for Feasible Collaborative Behavior Construction and Evaluation *

First A. Yunkai Wang, Second B. Shenhan Jia, Third C. Zexi Chen,
Fourth D. Zheyuan Huang and Fifth E. Rong Xiong
College of Control Science and Engineering
Zhejiang University
Hangzhou, Zhejiang Province, China
{wangyunkai, 3160104926, chenzexi, 21732054, rxiong}@zju.edu.cn

Abstract— In the case of the two-person zero-sum stochastic game with a central controller, this paper proposes a best collaborative behavior search and selection algorithm based on reinforcement learning, in response to how to choose the best collaborative object and action for the central controller. In view of the existing multi-agent collaboration and confrontation reinforcement learning methods, the methods of traversing all actions in a certain state leads to the problem of long calculation time and unsafe policy exploration. This paper proposes to construct a feasible collaborative behavior set by using action space discretization, establishing models of both sides, model-based prediction and parallel search. Then, we use the temporal difference learning method to train the scoring function to select the optimal collaboration behavior from the feasible collaborative behavior set. This method enables efficient and accurate calculation in an environment with strong confrontation, high dynamics and a large number of agents, which is verified in the RoboCup Small Size League robots ball passing collaboration.

Index Terms— Multi-agent, Reinforcement learning, RoboCup SSL, Dynamic passing ball

I. Introduction

The research on cooperation and confrontation of multi-agent system (MAS) originated in the 1980s and it is one of the research hotspots in robotics and artificial intelligence. The core problem is to establish a mechanism that enables multiple agents to cooperate with each other to accomplish target tasks and achieve complex intelligence. The research of multi-agent cooperation and confrontation is of great significance, which has been widely used in many fields such as recommendation system, traffic control, unmanned aerial vehicle (UAV) control and game confrontation.

Due to the interaction between multiple agents, the complexity of multi-agent system increases rapidly with the increase in the number of agents, the complexity of behavior and system dynamics. This presents a great challenge to the approach of pre-programming to realize the behavior of multiple agents, which leads to the research of reinforcement learning (RL)[2], evolutionary computation, game theory, complex system theory and other methods. Among them, reinforcement learning is a hot topic in recent years.

In a multi-agent collaboration and confrontation system, there is usually an agent that is most important to achieve the current goal, which is called the central controller or leader. A leader usually needs to select an optimal cooperative object and complete the cooperative behavior by executing a certain action. For example, in ball games, the leader is a player who holds the ball and needs to select an optimal receiver to complete a passing cooperation by executing the passing action. How to choose the best cooperative object and the appropriate cooperative action is a key problem in multi-agent collaboration and confrontation system. In this paper, the combination of a cooperative object and a cooperative action is defined as a cooperative behavior, called COCAP (Cooperative Object-Cooperative Action Pair), and an algorithm for searching and selecting the optimal cooperative behavior based on RL is proposed. Our method, firstly, according to the action space sampled with a certain precision and the models of both sides, predicts the behaviors and the final results of all the agents after each cooperative behavior executed, then constructs a set of all feasible collaborative behaviors, finally the best one will be selected by a scoring function which is trained by RL method.

In this paper, the RoboCup Small Size League (SSL) platform[1] is used for verification. It is a centralized and distributed hybrid system, where two teams play against each other and the robot players collaborate to score goals. In essence, it is a two-person zero-sum stochastic game process. The two-person zero-sum stochastic game process refers to that the two parties involved in the game are in a strict competitive relationship. At each time the two parties choose their actions, and they will gain benefits according to the current state and the selected actions. Then a new round of games is carried out in the

*This work was partially supported by Science and Technology Project of Zhejiang Province (2019C01043), State Key Laboratory of Industrial Control Technology (ITC1904) and Fundamental Research Funds for the Central Universities.

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

next random state. The distribution of the new random state depends on the previous state and the actions chosen by both sides. The characteristics of the RoboCup SSL platform are high dynamics and high antagonism, which has certain universality for the research of multi-agent collaboration and confrontation. In this paper, an example of RoboCup SSL robots dynamic ball passing is presented to solve the problem of multi-robot cooperative passing in high dynamic environment.

The remainder of the paper is organized as follow: section II introduces the research of solving the cooperation and confrontation problems of multiple agents by learning method. Section III introduces the best cooperative strategy learning algorithm based on RL. In section IV, the proposed algorithm is verified with the RoboCup SSL platform. Section V draws a conclusion, which completes the paper.

II. Related Work

In using RL method to solve the problem of multi-agent cooperation and confrontation, problems can be classified according to the types of game tasks, namely, fully cooperative, fully competitive and mixed tasks. Fully cooperative task means that all agents can benefit from cooperation without competition. Fully competitive task is a kind of zero-sum game, that is, the gain of one party necessarily means the loss of the other party, and there is no possibility of cooperation. Mixed task means that there is both collaboration and competition between agents. Although they are different problems, the solutions of these three problems have strong similarity and interoperability when using RL method to solve them.

In the unique case of optimal joint action, it can be solved by greedy algorithm, that is, each agent chooses the optimal action in the current state, such as Team Q-learning method[3] proposed by Littman. When the optimal joint action is not unique, it can be solved by the coordination mechanism between multiple agents, including direct and indirect coordination mechanism. In terms of direct coordination mechanism, Foerster et al.[4] proposed that Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL) can be used to learn and communicate end-to-end in a complex environment by means of deep q-learning and back-propagation error derivative of communication channel. Sukhbaatar et al.[5] proposed CommNet, which enables multiple agents to complete the task of fully cooperation through continuous communication, and enables them to learn communication while learning strategies. Peng et al.[6] proposed a Multiagent Bidirectionally-Coordinated Network (BiCNet) for communication, which builds a actor-critic framework to learn actions and achieve multi-agent cooperation and mastery of various battles in StarCraft games. Indirect coordination mechanism can use the method of agent modeling. For example, Claus et al.[7] proposed the joint action learners (JALs) which are agents that learn Q-values for joint actions as opposed to individual actions, and each agent will model for other agents and combine the model to improve the returns of state action pairs. In the case of confrontation can also use the mini-max principle to evaluate the optimization, such as Littman proposed minmax-Q algorithm[8], assuming that the opponent will take actions to minimize our benefits and maximize their own.

When solving cooperative and confrontative problems of multiple agents by RL, convergence and rationality of learning are very important. Convergence refers to whether the process of agent strategy training is stable and convergent, while rationality refers to the fact that agent's strategy can always converge to an optimal response strategy relative to other players' strategies. In order to achieve convergence and rationality, single-agent algorithm is useful to solve multi-agent problems, such as WoLF-PHC[9] algorithm proposed by Bowling et al. It contains the average strategy of each agent with the current strategy and the current strategy will refer to the average strategy and update the average strategy. The method of centralized learning plus decentralized execution is also used to improve the robustness of the algorithm, such as the MADDPG[10] algorithm proposed by Lowe et al., which enables multiple agents to find complex coordination strategies in physical and information in the cooperative and confrontative environment. And the Deep-MAHHQN[11] algorithm proposed by Fu et al. can not only adapt to the discrete-continuous mixed action space, but also show better effect than the independent parameter learning method.

Many researchers carry out agent decision-making or multi-agent cooperation and confrontation research based on RoboCup SSL platform. In the aspect of agent decision-making, Yoon et al.[13] realized the different shooting skills of robots by means of temporal-difference learning and multi-layer perceptron (MLP). Schwab et al.[14] used DDPG algorithm to realize skills learning such as ball finding and shooting, and transferred from simulation to actual robots. In the aspect of cooperation and confrontation between multiple agents, Nakanishi et al.[15] studied and analyzed the cooperative passing and shooting task of three robots, and achieved a high success rate in the test of actual robots. Trevizan et al.[16] used the method of machine learning to propose a similarity function, which compares two teams by imitating the behavior of another team, so as to formulate their strategies. Tphis function can classify opponents and decompose an unknown opponent into a combination of known opponents. Mendoza et al.[17] proposed the

Selectively Reactive Coordination (SRC) algorithm to achieve offensive coordination of multiple robots. Behzad et al.[18] used neural network to train opponent model to predict opponent's movement through game recording, so as to get the weakness of opponent's strategy.

Currently, to realize multi-agent cooperation and confrontation, RL methods usually traverse all actions in a certain state to obtain the action that maximizes the value function, which is generally approximated by deep neural network with stronger representational ability. However, in most cases, a large number of actions need to be traversed, and the deep neural network takes a long time to operate. The method of traversing all actions will make the calculation time longer, and unsafe actions will be selected in the policy exploration. In this paper, combined with the agent's model, only the behaviors in the feasible cooperative behavior set are traversed and explored, which can not only ensure the feasibility of the selected behaviors in policy exploration, but also reduce the time complexity of the algorithm.

III. Best Collaborative Strategy Learning

In response to how to choose the best collaborative object and the best collaborative action for the central controller in the case of the two-person zero-sum stochastic game, this section proposes a best collaborative behavior search and selection method based on RL. Firstly, we discretize the action space, then establish models of both sides and parallel search a feasible collaborative behavior set including actions and collaborative objects. Finally, we degenerate the problem into a Markov decision process, and use the temporal difference learning method to train the scoring function for selecting the optimal collaboration behavior from the feasible collaborative behavior set.

A. Feasible Collaborative Behavior Set Search

For searching a feasible collaborative behavior set, we predict the behaviors of the remaining agents and the final outcome after the leader performs each collaborative behavior. We use min-max principle[8] to predict the result, i.e., assuming that all agents act according to the optimal strategy model after the leader performs a collaborative behavior. Using the optimal strategy model to directly solve or deduct, we can get the prediction results in the future. And actually, in many multi-agent collaboration and confrontation systems, they typically use a substantially similar optimal strategy model.

Based on the optimal strategy model, we samples the collaborative action space of the leader according to a certain precision, and obtains the collected collaborative action set $\mathcal{A} = \{a_i\}$, where a_i represents different collaborative actions. Assuming that all the agents use the optimal strategy to cooperate or confront, and rely on the centralized parallel computing to predict the final result and actions executed by each agent after the leader executes each cooperative action a_i. Then we select all of the feasible (or beneficial to us) collaborative behaviors. In this paper, we defined a collaborative behavior consists of collaborative actions and a collaborative object as "Collaborative Actions-Collaborative Object Pair"(CACOP), and collaborative behaviors forms a feasible collaborative behavior set $\mathcal{C} = \{\langle a_j, r_p \rangle\}$, where r_p represents different collaboration objects. After that, we calculate the feature vector x_k of each CACOP which may be selected by the leader from the feasible collaborative behavior set. The feature vector x_k is extracted manually that is beneficial to the goal of MAS. Finally, the scoring function was used to calculate scores with respect to the feature vector. The CACOP with the highest score was selected, and its action was selected as the best cooperative action, and its cooperative object was selected as the best cooperative object.

For the ball passing problem, the optimal strategy of a single robot can be defined as intercepting the moving ball with the maximum movement ability and the shortest time, i.e.

$$\begin{aligned} &\min \ t \\ &\text{s.t. } v \leqslant v_{max} \\ &\quad\ \ a \leqslant a_{max} \end{aligned} \tag{1}$$

where t is the time when the robot intercepts the ball, v and a are respectively the velocity and acceleration of the robot at any time, and v_{max} and a_{max} are respectively the maximum velocity and acceleration of the robot. We adopt the bang-bang control method proposed by Kalmár-Nagy et al.[21] in 2002 for motion planning. That is, at any moment, the robot will accelerate or decelerate at its maximum acceleration, or it will move at its maximum speed.

According to this optimal strategy, the interception time of robot can be predicted. In this paper, the equal time interval search method is adopted, and the fixed minimum time interval is Δt (for example, 1/60 seconds). After the position and velocity of the robots and the ball are obtained through observation at a certain moment, we assume that the ball does uniform deceleration linear motion under the action of field friction, and calculate the position P_i that the ball can reach after any time $i\Delta t (i = 0, 1, 2, 3, ...)$. Then, we start from $i = 0$ to search all points, and predict the time T_i required for some robot to reach the P_i point. If the condition $T_k \leqslant k\Delta t$ is satisfied after the $k-th$ time interval, it is considered that P_k is the best interception point P_{best} of the robot, and $k\Delta t$ is the shortest interception time T_{best} of the robot. In the practical application of RoboCup SSL, in addition to the prediction of the ball rolling on the field, it is also necessary to predict the position of flip shot ball. That is, before preparing to flip shot the ball, we predict the first and second drop locations and arrival times of the ball after flip shot at a certain speed, as well as the subsequent approximate rolling speed. Therefore, when calculating the interception prediction of the flip shot ball, we just need to add the time of the first two jumps in addition.

The cooperative actions of the leader robot include kicking mode c, kicking direction θ and kicking speed v. In this paper, 128 bisect samples were taken from the direction of kicking the ball and 16 bisect samples were taken from the speed of kicking the ball according to the limitation of the kicking ability of the robot, thus forming the cooperative action set $\mathcal{A} = \{a_i\}$, where $a_i = \langle c_i, \theta_i, v_i \rangle$. Then, each kick mode, direction and speed were traversed, and up to 24 robots' interception information were predicted, so that all feasible CACOP could form a feasible cooperative behavior set $\mathcal{C} = \{\langle a_j, r_p \rangle\}$, where r_p represents our different cooperative robots. In order to realize real-time high-performance computing, we use $128 \times 16 \times 24$ threads on the GPU for parallel computing.

On the passing problem, the feature vector x_k can be composed of the following quantities: the interception time of the robot, the shooting angle of the passing point, the distance from the passing point to the opponent's goal, the angle of shooting after passing, and the interception time of our robot before any other enemy robot. Distance features and angle features are all normalized[19] in order to balance the numerical values of different dimensional features and accelerate training process.

Through the above work, the feasible cooperative behavior set and the feature vectors representing each feasible cooperative behavior are calculated.

B. Optimal Collaborative Behavior Scoring Function Learning

Based on the construction of the feasible cooperative behavior set, the selection of the best cooperative behavior is further considered.

A simple and effective method is linear weighted sum, but there are the following problems. First of all, it is difficult to adjust the weight parameters with this method, and it is also difficult to quantify the effects of different parameters. Secondly, the linear weighting of features has its own problems, and the weight of many features should be non-linear. Therefore, the linear weighted sum method has some limitations.

A universal and feasible method to solve the above problems is to use MLP to fit the best pass point scoring function. MLP can use its nonlinear activation function to fit the nonlinear function, and use its large number of nodes and depth to fit the higher-order function.

The common training method of MLP is supervised learning, that is, providing MLP input vector and target output vector for supervision, and training network weight by comparing the difference between the output and target output. However, using supervised learning is relatively difficult for the following reasons: First, manual annotation needs to provide ground truth, that is a best CACOP in our problem. This job sometimes is highly professional. Secondly, supervised learning needs a large amount of data as support, which means that a large amount of data needs to be annotated manually. Therefore, using supervised learning to train a MLP as a scoring function is not ideal.

Another way to train MLP is reinforcement learning. The method of RL only needs to provide a relatively obvious reward function without manual labeling, and the RL method can fully consider the cumulative return, that is, learning the causal relationship in Markov decision process. RL can not only learn offline data, including expert presentation data and opponent generated data, but also learn online to interact with the environment to generate new data. Therefore, this paper uses RL method to train the MLP to obtain the scoring function of cooperative behaviors.

To use RL method for training, first we need define a Markov decision process. In this paper, each collaboration process is regarded as a state s of Markov decision process, and the quantity describing this state is the feature vector x_k mentioned above. The action a in Markov decision process is to select different CACOPs. In this paper, it is considered that the jump from one collaboration state to another is only related to the previous collaboration state and the selected CACOP, and not related to the previous one. Such a process conforms to the conditions of Markov decision process. For example, for the robot ball passing problem, in a certain cooperation process, the robot 1 passes the ball to the robot 2, and the state of the cooperation process describing the state of s is: the interception time of the robot 2, the shooting angle of the passing point, the distance from the passing point to the opponent's goal, the angle at which the shooting ball is refracted after passing the ball, and the time when the robot 2 preferentially intercepts the ball than the enemy robot. After completing this collaborative process, we make decisions with robot 2 as leader, select the best cooperative action of the robot 2 and the best collaborative object, and then performs the next collaborative process to obtain the next cooperative state s'.

On the basis of defining the Markov decision process, it is assumed that the value function (i.e. scoring function in our problem) is $V(s)$ and our optimal policy is $\pi^\star$ which corresponds to the optimal value function:

$$\pi^\star = \arg\max_\pi V^\pi(s) \tag{2}$$

where s stands for the collaboration status. This value function is updated according to the formula of temporal difference (TD) learning:

$$V(s_t) = V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)) \tag{3}$$

Among them, α is the learning rate, γ is the discount factor, t is the time-step, r is the reward and the '=' sign is used as an update operator but not equality. Hence, when it is in the state s_t, our RL agent will choose a collaborative action (i.e. CACOP) to make the next predict $V(s_{t+1})$ maximize. And ϵ-greedy policy is used when the agent choose actions for better exploration. Reference [20], this paper also use target network to stabilize the training process and memory buffer to break data relevance, and the loss function is mean square error (MSE):

$$Loss = \frac{1}{2}(r_{t+1} + \gamma V_{target}(s_{t+1}) - V(s_t))^2 \tag{4}$$

where V_{target} is the target value function and V is the online value function. After we get a mini-batch data from memory buffer and train the online value function, we will soft-update the target value function by

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{5}$$

where θ and θ' are respectively the parameters of the online value function and the target value function, and τ is the soft update rate.

IV. Experiment

A. Background of RoboCup SSL Platform

This paper uses the RoboCup SSL platform to conduct experiments and verify the effectiveness of the method through ball passing collaboration. Fig.1 is a schematic diagram of the robot system architecture. On the actual playing field, each side has 8 robots and an orange golf ball as a football. The visual system above the field is processed according to the color of the ball and color code on the top of robots. The numbers, positions, orientations of the robots and the ball position are processed by an official dedicated software and sent to each team's computer. Then each team's computer makes decisions and sends commands to the robots on the field by wireless radio. The size of the field is $12m$ long and $9m$ wide. The diameter of the robot is $180mm$, and the maximum speed allowed for robot kicking is $6.5m/s$. The motion performance of the robot used in this experiment is shown in the Table I. In the actual game, the ball-carrying robot

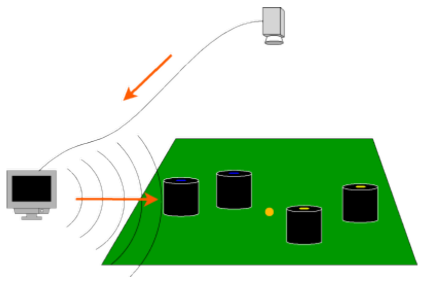

Fig. 1. RoboCup small size robot soccer system architecture

TABLE I
Robot performance in experiment

Performance names	parameters
Max speed	$3m/s$
Max acceleration	$4.5m/s^2$
Max rotational speed	$15rad/s$
Max rotational acceleration	$15rad/s^2$

can kick the ball at a certain speed in a certain direction by means of a flat shot or a shot, and then the cooperative robot moves to the ball that is calculated in advance to intercept the moving ball, and then passes the ball to the next cooperative robot.

B. Feasible Collaborative Behavior Set Construction

This paper first validates the optimal strategy model for constructing a feasible collaborative behavior set. In the experiment, the interception time of the stationary robot at different positions of the field was tested under the condition of $1m/s$ and $4m/s$. The performance of the robot in the experiment was limited according to the parameters in table I. The results are shown in Fig.2 and Fig.3. The darker areas of the heat map represent shorter interception times, while the lighter areas represent longer interception times. In Fig.2, the ball moves to the right at a slower initial speed of $1m/s$. In this case, the closer the robot is to the ball, the faster it can intercept the ball. However, when the ball speed is faster, there will be different conclusions. As shown in the Fig.3, the ball moves to the right at a faster initial speed of $4m/s$, and the robot cannot intercept the ball at the left position. There is a clear boundary in the heat map. If the position of the robot is within this boundary (i.e. dark area), the ball can be intercepted in a short time, while outside the boundary (i.e. white area) it cannot be intercepted in the field.

On the basis of obtaining the optimal strategy model, this paper visualizes the best intercept point for all feasible CACOPs calculated at two different times in the game and the best flat shot and flip shot method obtained by linear weighted sum method, as shown in Fig.4. The entire process of the algorithm can achieve an average

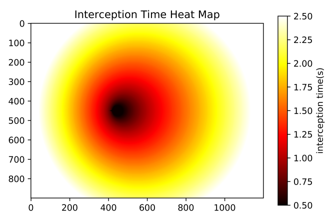

Fig. 2. $1m/s$ ball speed interception time heat map. The ball moves from the $(400cm, 450cm)$ to the right at a slower initial speed of $1m/s$.

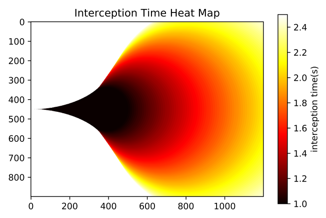

Fig. 3. $4m/s$ ball speed interception time heat map. The ball moves from the $(0cm, 450cm)$ to the right at a faster initial speed of $4m/s$.

operation speed of $4.04ms$ per frame on the GTX 1060 6G GPU, while using the method for all CACOPs traversal requires an average of $13.77ms$ per frame, which is about 3.4 times slower than the former. Therefore, the efficiency of the method can be verified. Since the maximum speed of kicking the ball is $6.5m/s$, which is larger than the robot's ability to move, the feasible pass points obtained are mostly in the vicinity of the best cooperative robot. In the case where the enemy robots are still on the field, the ball passing success rate is almost 100% by this method. In the case of enemy robots movement on the field, there may be no feasible flat pass method, but there is always a feasible flip pass method. Using the linear weighted sum method to simply adjust the parameters can give relatively reasonable results, but it is difficult to get a similar effect to manual programming.

C. Scoring Function Training by RL

After construction the feasible collaborative behavior set, the RL method is used to train the scoring function of ball passing method. Table II is the structure of the scoring function. And this paper empirically choose hyper-parameters shown in Table III.

The reward function in our problem mainly consists of two parts: the first part reward r_1 is a dense reward

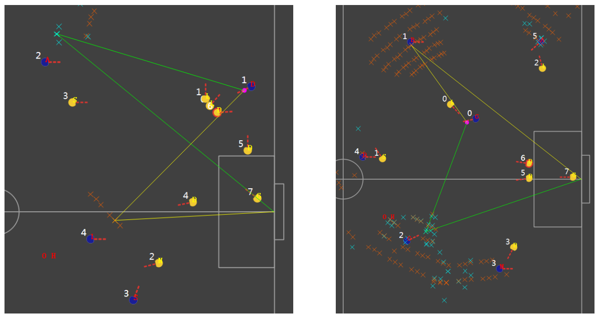

Fig. 4. Visual renderings of feasible collaborative behaviors. The light blue points represent feasible flat pass points and the orange points represent feasible flip pass points. The green line represents the best one-shot ball trajectory with a flat shot and the green line represents the best one-shot ball trajectory with a flip shot.

TABLE II
Structure of the scoring function

Layer	Channels In	Channels Out
fc1	5	256
fc2	256	256
fc3	256	1

TABLE III
Hyper-parameters in experiment

Hyper-parameter Names	Values
Activation function	Leaky ReLU
Optimizer	Adam[22]
Initial learning rate	3×10^{-4}
Batch size	128
Discount factor γ	0.99
Soft update rate τ	0.005
ϵ-greedy rate ϵ	0.9

based on the distance of the ball from the goal, in the form of

$$r_1 = e^{-\frac{x}{a}} \tag{6}$$

Where x represents the distance of the ball from the center of the goal in meters, and a is a constant coefficient. When the ball is very close to the goal, the first part reward is close to 1, and when the ball is very far from the goal, the first part reward is close to 0. According to the size of the field, when the ball is 3 meters away from the center of the goal, the first part reward is 0.5, so that the solution of a is 4.33. The second part reward r_2 is the sparse reward of whether the ball enters their penalty area. When the ball enters their penalty area, r_2 is 10, and when the ball enters the goal, r_2 is 50. Finally, the total reward function is

$$r = r_1 + r_2 \tag{7}$$

This reward function allows the RL agent to learn how to pass the ball and score a goal.

Considering that multiple robots can learn to pass well in the beginning and improve the utilization of data in online RL training, this paper uses all the official game log of the 2018 RoboCup SSL[23] as the empirical data for offline training. After the offline training on the game log, the self-confrontation competition is carried out in the simulation[24] to generate more data to train, and then repeat this training process. The total process in the experiment shows in Alg. 1. After a period of training, the trained MLP is obtained, which is the scoring function of the ball passing.

Algorithm 1 Total process in the experiment

```
Initialize online networks $V_\theta$ with random parameters $\theta$ and initialize target networks $V_\theta'$ with $\theta' = \theta$
Initialize empty replay buffer $\mathcal{B}$ and game log replay buffer $\mathcal{B}\prime$
for $t = 1$ to $T_1$ do
    Sample mini-batch of N transitions $(s_t, a, r, s_{t+1})$ from $\mathcal{B}\prime$
    Update online networks:
    $\theta \leftarrow \theta + \alpha \frac{1}{2N} \Sigma(r + \gamma V_{\theta'}(s_{t+1}) - V_\theta(s_t))^2$
    Soft-update target networks:
    $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$
end for
$\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}\prime$
for $t = 1$ to $T_2$ do
    Calculate the features as state $s_t$
    Construct a feasible collaborative behavior set $\mathcal{C}$
    Select action $a$ in $\mathcal{C}$ to maximize predicted $V_{\theta'}(s_{t+1})$ by $\epsilon$-greedy
    Observe reward $r$ and new state $s_{t+1}$
    Store transition tuple $(s_t, a, r, s_{t+1})$ in $\mathcal{B}$
    Sample mini-batch of N transitions $(s_t, a, r, s_{t+1})$ from $\mathcal{B}$
    Update online networks:
    $\theta \leftarrow \theta + \alpha \frac{1}{2N} \Sigma(r + \gamma V_{\theta'}(s_{t+1}) - V_\theta(s_t))^2$
    Soft-update target networks:
    $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$
end for
```

Using the trained scoring function to perform a game in the simulation environment, and visualizing the scores of the best intercept points corresponding to all CACOPs obtained by the algorithm, the effect is shown in Fig.5. The redder color area represents a higher pass score, and the bluer color area represents a lower pass score. The areas with low scores are mainly concentrated near our half and enemy (yellow) robots, while the areas with high scores are mainly concentrated near the opponent's penalty area and near our (blue) robot, indicating that the selected pass area is in line with human thoughts

Finally, this paper carried out 4v4 attack and defense test[1], that is, 4 robots of the offensive side only pass the ball but not shoot, 4 robots of the defensive side only defend and not grab the ball, to test the rationality of the offensive side pass the ball. Some screenshots show in Fig6. In the first screenshot, No.3 blue robot is the leader, and it holds the ball and decides to pass to No.5 blue robot with the best collaborative action, which is the best collaborative object in the high score area and no enemy defender is marking it. In the second screenshot, No.5 blue robot becomes the new leader, and it goes to catch the ball and decides to pass to No.2 blue robot in the third screenshot. This two passes formed a big defensive gap in the top side. And in the last screenshot, No.2 blue robot becomes the new leader and will do the same collaborative process. This test shows that the offensive side can pass the ball to the position with high threat degree under high dynamic condition by using the algorithm in this paper, which further verifies the effectiveness of the algorithm.

V. Conclusions

Aiming at the problem of how to choose the best cooperative object and the best cooperative action in the multi-agent collaboration and confrontation system, this paper proposes a method to search for the best cooperative behavior based on the feasible collaborative behavior set. According to the optimal strategy models of both sides, all feasible cooperative behaviors can be obtained by parallel search on GPU, and the optimal cooperative behaviors can be selected by training score function by RL method. The proposed method is verified by experiments in the RoboCup SSL robots platform, and it can make the robots cooperate with other robots in a highly dynamic and confrontational environment, so that the whole system can show high intelligence. It has been applied to actual competitions and we relied on the superiority of the algorithm to achieve the 2019 RoboCup SSL world championship.

References

[1] RoboCup Soccer Small Size League. https://www.robocup.org/leagues/7.

[2] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[3] Littman, Michael L. "Value-function reinforcement learning in Markov games." Cognitive Systems Research 2.1 (2001): 55-66.

[4] Foerster, Jakob, et al. "Learning to communicate with deep multi-agent reinforcement learning." Advances in Neural Information Processing Systems. 2016.

[5] Sukhbaatar, Sainbayar, and Rob Fergus. "Learning multiagent communication with backpropagation." Advances in Neural Information Processing Systems. 2016.

[6] Peng, Peng, et al. "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games." arXiv preprint arXiv:1703.10069 (2017).

[1] https://youtu.be/S40VmSYvlPks

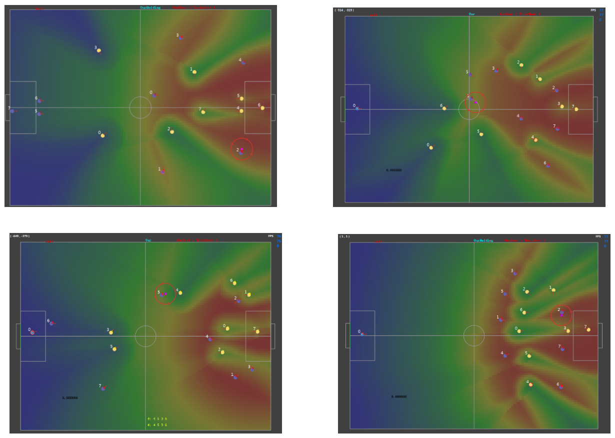

Fig. 5. The heat map of the pass score obtained by the algorithm. The redder color area represents a higher pass score, and the bluer color area represents a lower pass score. The yellow circle represents the enemy robot, the blue circle represents our robot, and the purple circle represents the ball.

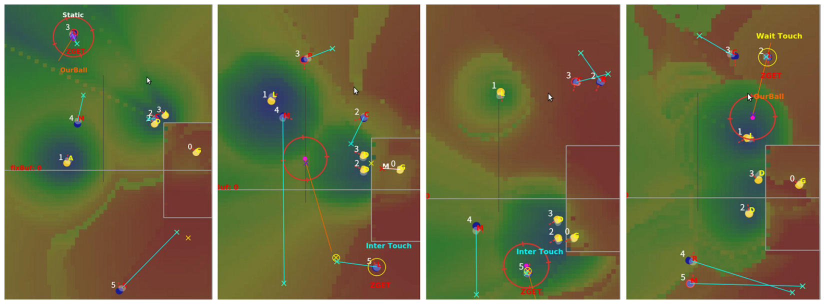

Fig. 6. Screenshots of 4v4 attack and defense example. In the first screenshot, No.3 blue robot is the leader, and it holds the ball and decides to pass to No.5 blue robot which is the best collaborative object with the best collaborative action. In the second screenshot, No.5 blue robot becomes the new leader, and it goes to catch the ball and decides to pass to No.2 blue robot in the third screenshot. And in the last screenshot, No.2 blue robot becomes the new leader and will do the same collaborative process.

[7] Claus, Caroline, and Craig Boutilier. "The dynamics of reinforcement learning in cooperative multiagent systems." AAAI/IAAI 1998.746-752 (1998): 2.

[8] Littman, Michael L. "Markov games as a framework for multi-agent reinforcement learning." Machine learning proceedings 1994. Morgan Kaufmann, 1994. 157-163.

[9] Bowling, Michael, and Manuela Veloso. "Rational and convergent learning in stochastic games." International joint conference on artificial intelligence. Vol. 17. No. 1. Lawrence Erlbaum Associates Ltd, 2001.

[10] Lowe, Ryan, et al. "Multi-agent actor-critic for mixed cooperative-competitive environments." Advances in Neural Information Processing Systems. 2017.

[11] Fu, Haotian, et al. "Deep Multi-Agent Reinforcement Learning with Discrete-Continuous Hybrid Action Spaces." arXiv preprint arXiv:1903.04959 (2019).

[12] Uther, William, and Manuela Veloso. Adversarial reinforcement learning. Technical report, Carnegie Mellon University, 1997. Unpublished, 1997.

[13] Yoon, Moonyoung. Developing basic soccer skills using reinforcement learning for the RoboCup Small Size League. Diss. Stellenbosch: Stellenbosch University, 2015.

[14] Schwab, Devin, Yifeng Zhu, and Manuela Veloso. "Learning Skills for Small Size League RoboCup." Robot World Cup. Springer, Cham, 2018.

[15] Nakanishi, Ryota, et al. "Cooperative 3-robot passing and shooting in the robocup small size league." Robot Soccer World Cup. Springer, Berlin, Heidelberg, 2006.

[16] Trevizan, Felipe W., and Manuela M. Veloso. "Learning opponent's strategies in the RoboCup small size league." Proc. AAMAS. Vol. 10. 2010.

[17] Mendoza, Juan Pablo, et al. "Selectively reactive coordination for a team of robot soccer champions." Thirtieth AAAI Conference on Artificial Intelligence. 2016.

[18] Behzad, Kian, et al. "PARSIAN 2019 Extended Team Description Paper." RoboCup Soccer Small Size League, 2019.

[19] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

[20] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.

[21] Kalmár-Nagy, Tamás, Raffaello D'Andrea, and Pritam Ganguly. "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle." Robotics and Autonomous Systems 46.1 (2004): 47-64.

[22] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[23] Gamelogs of RoboCup SSL 2018. https://tigers-mannheim.de/download/gamelogs/2018/div-a/.

[24] Monajjemi, Valiallah, Ali Koochakzadeh, and Saeed Shiry Ghidary. "grsim–robocup small size robot soccer simulator." Robot Soccer World Cup. Springer, Berlin, Heidelberg, 2011.