Proximal Policy Based Deep Reinforcement Learning Approach for Swarm Robots

Ziya TAN
Erzincan Binali Yıldırım University
Erzincan, Turkey
ziyatan@erzincan.edu.tr

Mehmet KARAKÖSE
Department of Computer Engineering
Firat University
Elazığ, Turkey
mkarakose@firat.edu.tr

***Abstract*— Artificial intelligence technology is becoming more active in all areas of our lives day by day. This technology affects our daily life by more developing in areas such as industry 4.0, security and education. Deep reinforcement learning is one of the most developed algorithms in the field of artificial intelligence. In this study, it is aimed that three different robots in a limited area learn to move without hitting each other, fixed obstacles and the boundaries of the field. These robots have been trained using the deep reinforcement learning approach and Proximal policy optimization (PPO) policy. Instead of uses value-based methods with the discrete action space, PPO that can easily manipulate the continuous action field and successfully determine the action of the robots has been proposed. PPO policy achieves successful results in multi-agent problems, especially with the use of the Actor-Critic network. In addition, information is given about environment control and learning approaches for swarm behavior. We propose parameter sharing and behavior-based method for this study. Finally, trained model is recorded and tested in 9 different environments where the obstacles are located differently. With our method, robots can perform their tasks in closed environments in the real world without damaging anyone or anything.**

Keywords—deep reinforcement learning, swarm behavior, deep learning

I. INTRODUCTION

When we examine nature, it is seen that there are many examples where the effort of a colony of limited individuals has achieved success above individual abilities. When ants are in a swarm, they carry food that a single ant cannot carry, termites build nests up to nine meters high, bees can regulate the temperature inside the hive. Birds use each other's formed air tunnels to fly further. The common point of all these actions is that their capacities have only standard and basic needs perceived and limited communication skills when examined individually [1].

To summarize briefly the benefits of swarm intelligence is shown in Fig.1:

- Derivatives free optimization; solutions to existing problems have not been previously defined. Solutions arise instantly.
- Robustness; Tasks are completed even if some agents in the swarm fail.
- Adaptation; the swarm system can also adapt to stimuli that may occur later.
- Flexibility; the swarm is responded to internal disturbances and external challenges.
- Low total cost; the costs to the target are minimized as much as possible.

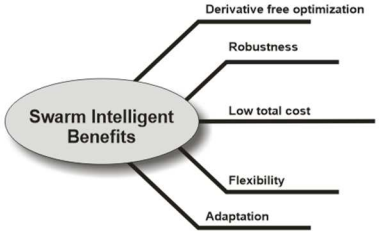

Fig. 1. Benefits of swarm intelligence

Environment control and learning approaches are of great importance in swarm organizations. In this system, it is generally adopted that agents act consistently without hitting each other. There are three different formation control and three different learning approach strategies with different advantages in the literature. It is aimed for robots to train a single policy during the training phase and to perform their actions independently from other robots. Therefore, parameter sharing is proposed as a learning approach. Furthermore, behavior-based approach is adopted as environment control to facilitate individual movement of robots within the swarm. These two strategies are explained in the second part.

This article presents a study about training swarm robots using deep reinforcement learning algorithms. Convolutional Neural Network (CNN) [2] algorithm is used in the deep reinforcement learning framework. In addition, the Proksimal Policy Optimization (PPO) [3] policy is trained in two different models using Actor-Critic architecture. PPO is a model-free, on-policy, gradient-based reinforcement learning method.

Looking at the literature, in the study of Kakish Z. et al. [4], which adopts the leader tracking organization with the reinforced learning approach, Q-learning and SARSA, which are two different Temporal-Difference learning algorithms, were used. Through these algorithms, a leader control tracking approach based on the lead agent's position in the environment is presented for populations of 10 to 100 follower agents. In another similar study, Xuejing Lan et al. [5] discuss the problem of collaborative leader follow-up control with a reinforcement learning approach. It is aimed that the swarm members in the environment move along the route determined by a virtual

dynamic agent. Furthermore, a study in which biological swarm behaviors were developed using the calculation algorithm [6] , and studies [7] proposing a new state representation for the multi-agent DRL approach were conducted. A study on decision making strategy for autonomous vehicles using deep reinforcement learning method enhanced with proximal policy optimization is presented by Teng Liu et al [8].

The rest of the work is organized as follows: In the second chapter, we provide information about learning approaches for multi-agent systems, environment control, and details of the DRL approach we use in our study. In the third section, the presented working environment and parametric adjustments are explained. In the fourth chapter, simulation and test results are given with graphics. Finally, in the fifth chapter, the results of the study are discussed.

II. Deep Reinforcement Learning For Multiple Robot Systems

In this section, learning approaches and environment controls offered for swarm problems with deep reinforcement learning algorithms are explained.

A. Environment Control

Environment control is introduced in the literature in three different ways. These are Leader tracking, Behavior-based and virtual structures [9].

Leader Tracking configures robots in the swarm in two ways. It consists of a leader robot and its followers. It shares the actions that go to the target in the environment with other followers through the leader robot. In this way, the swarm moves towards the target.

In behavior-based environment control, each robot performs its own unique action. Their experiences are used to learn these actions. Thus, the robots in the environment learn not to hit each other.

Virtual structures are the situations where all robots act as a single structure. The swarm's intelligence and actions are poor, as the system acts as a single structure.

B. Learning Approaches

There are three different learning approaches: central learning, simultaneous learning and parameter sharing [10].

Central learning approach is an approach in which all robots in the environment are trained by a single model. The trained policy determines a joint action by evaluating the action experiences from all robots.

In simultaneous learning; each robot is responsible for learning its own policy. Therefore, each robot performs its own action. Thus, more than one policy can be trained at the same time.

In parameter sharing; a single policy is trained by the swarm. However, each robot chooses its own action. Thus, the policy can be trained faster.

In this study, we present behavior-based control as environment control and parameter sharing as learning approach.

C. Deep Reinforcement Learning

Machine learning is a field of artificial intelligence that provides automated methods using patterns in data. We can talk about three different types of machine learning. Supervised learning; draws a conclusion using the tagged education data [11].

Unsupervised learning; draws conclusions from data sets consisting of input data without labeled data [12].

Reinforcement learning; It is a learning method that uses the reward-penalty system to maximize total rewards without the need for any preliminary data [13].

Deep learning is based on the function $f : X \rightarrow Y$ parameterized by (1).

$$\theta \epsilon \mathbb{R}^{n\theta} (n_\theta \in \mathbb{N}) : y = f(x;\theta) \quad (1)$$

A deep neural network is formed by the combination of many consecutive layers. Each layer consists of a nonlinear structure. These layers consist of input layer, convolution layer and output layer. Reinforcement learning(RL) [14] is the formation of an action with the experience an agent gained from the reward and penalty that receives in an educational environment. At each step, the agent determines a state from the current state (St) of environment E from all possible sets of states (S) and selects an action from all possible sets of actions (a) depending on this state. One of the popular algorithms in reinforcement learning was the development of a non-policy Temporal-Different control algorithm known as Q-learning. Basically, one-stage Q-learning is defined as given by (2):

$$(S_T, A_T) \leftarrow Q(S_T, A_T) + \alpha[\ R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_T, A_T)\ ] \quad (2)$$

$(S_T, A_T) \rightarrow$ New value

$Q(S_T, A_T) \rightarrow$ Old value

$\alpha \rightarrow$ Learning rate

$\gamma \rightarrow$ Discount factor

$R_t \rightarrow$ Reward

$max_a Q(S_{t+1}, a) \rightarrow$ Estimate of optimal future value

Reinforcement learning methods such as Q-learning are more reliable than stochastic algorithms because they reward the current action by receiving a reward for state transitions. Based on the Markov Decision Process (MDP) [15] theory, Q-learning is an ideal method for solving the problems of representative actions under different complex conditions. The table matrix (Q-Table) used in Q-learning reaches a very large size with the increase of state and motion values [16].

Deep reinforcement learning (DRL) [17] solves the problem that occurs due to the large size of the state and action values with the help of neural networks. In this way, DRL has achieved outstanding success in games such as GO [18]. DRL architecture is given by Fig. 2.

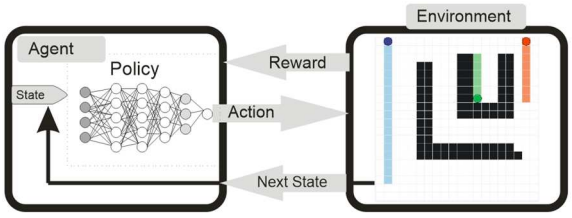

Fig. 2. Structure of deep reinforcement learning

D. *Proximal Policy Optimization*

PPO [3] is a model-free, on-policy, gradient-based reinforcement learning method. This method is a kind of policy gradient training ranging from sampling data through environmental interaction and optimizing a piece of objective function using stochastic gradient descent. The objective function improves the stability of education by limiting the extent of policy change at each step [3]. Agents using the PPO algorithm can be trained in discrete or continuous environments of observation and action [19].

In the training phase of the PPO agent:

- Estimates the probabilities of occurrence for each action in the area where the action takes place and randomly selects the actions according to these probability values.
- Interacts with the environment using the current policy before using mini-batch to update actor-critic parameters.

It adopts two main functions to calculate the policy and value function of PPO agent [20]. Interaction of agent-environment in DRL is given by Fig. 3.

Actor μ(S) - The actor takes observation S and returns the probabilities of taking each action in the action space when in state S.

Critic V(S) - Critic takes observation S and returns the expectation corresponding to discounted long-term reward.

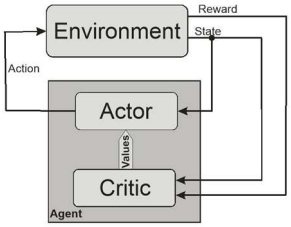

Fig. 3. Interaction of agent-environment in DRL

III. Simulation Environment And Parameter Settings

The swarm was trained with a deep reinforcement learning algorithm using the PPO policy. However, each robot determines its own action. Training was carried out in an environment of 20x20 units where have fixed obstacles. As shown in fig. 4, there are three robots represented by circles in the training environment.

During the training phase, the robots determine the action by the actor-critic neural network. The actor determines the action the robot will take by evaluating the data from the environment with the recommendation of critic network.

Robots were trained for 1500 episodes. Each episode consists of 500 steps. Robots have to perform one of 5 different actions during training. These are; wait, right, left, up, down. A training segment ends when the entire area is discovered by robots or when the step count is completed.

The reward-penalty system is applied as written below in each training step for robots.

- +1 point for scanning a previously undiscovered cell,
- -0.5 points for each illegal action (attempting to go out of bounds or hit other robots and obstacles),
- -0.05 points for each action that results in a move,
- -0.1 point for each action that does not result in a movement,
- +200 points for each robot during the episode if the training area is fully scanned

During the training, agents collect experiences until they reach the experience horizon 128-steps and then complete their training from mini-batches of 64 experiences. An objective function clip factor of 0.2 improves training stability. Long-term rewards are encouraged with a discount factor value of 0.99.

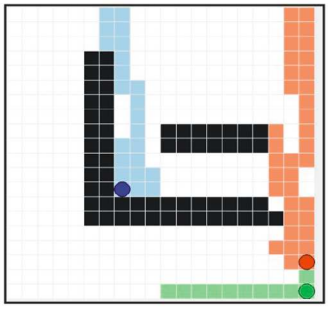

Fig. 4. Training environment

As shown in Fig. 4, there are three robots represented by circles in the training environment. Robots are represented by red, blue and green circles. Each robot paints the area it covers in its own color. The black colored squares represent the obstacles in the environment.

IV. Simulation and Test Results

After the training, robots are tested in 9 different environments with the saved model. 9 different test environments are given in fig 5. Test results are given in table 1. However, table 1 only shows the test results for 7 environments.

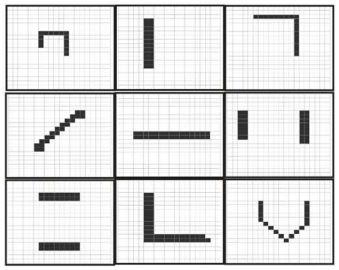

Fig. 5. 9 different environments used for testing

It has been tested in different environments to evaluate the success of the pre-trained model.

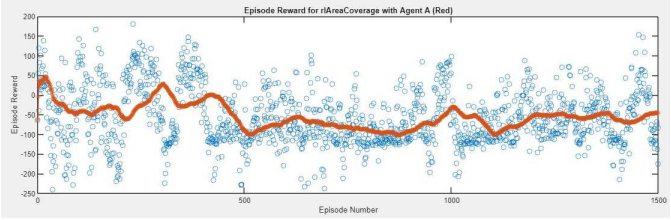

Fig. 6. Training graph of agent A(x axis=episode number, y axis= episode reward)

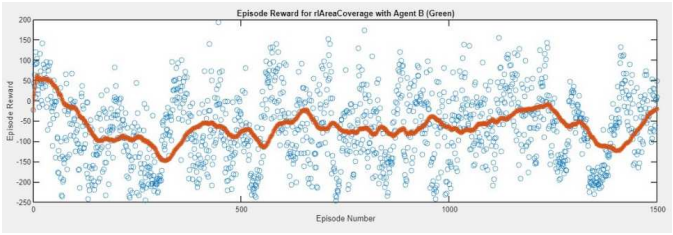

Fig. 7. Training graph of agent B(x axis=episode number, y axis= episode reward)

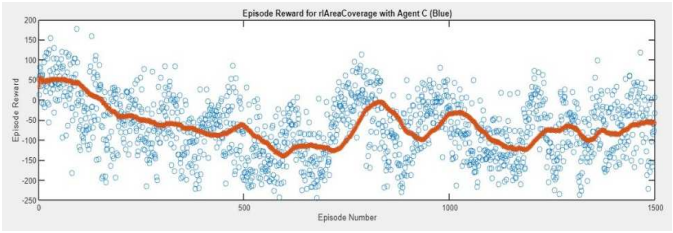

Fig. 8. Training graph of agent C (x axis=episode number, y axis= episode reward)

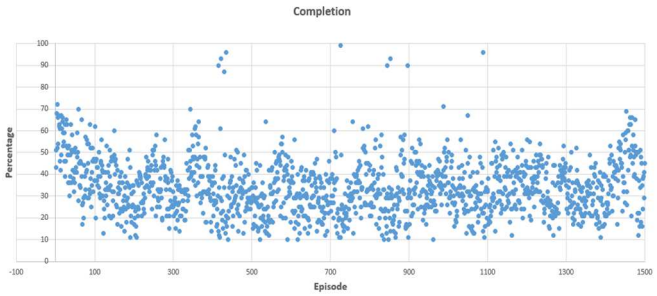

Fig. 9. Completion percentages of training environment (x axis=percentage, y axis= episode)

As shown in Fig.6, the robot led by agent A has drawn a steady reward-episode plot. During the training, the average reward is -46.21. After the first 500 steps, it is seen that the robot started to get to know the environment and how to get the most points from the episodes consistently.

Graph of agent B is shown Fig.7. When the graph is examined, it is seen that it constantly scanning the area after the 600th episode. Additionally, although there is a decrease in the average reward in the last 150 steps, it manages to compensate for this decrease towards the end of the training. The average reward that robot B received during the training is recorded as -19.57.

Fig.8 shows the training graph of agent C. When the graph is examined, agent C shows a less stable trend than the other two agents. Although it has achieved stability in the last 200 steps, it has not been successful in the previous episodes of the education. The average reward received by robot C during the training is -55.72.

Fig.9 shows the graph of how much of the available environment is completed by three robots. When the graph is examined, it is seen that the environment is completed approximately 10 times in the range of 80-100%. This number is directly related to the step number parameter in each episode being 500. Increasing this parameter may increase the scanned area surface, but the total rewards robots receive will decrease. When the last 100 steps are examined, it is noticed that the completion percentages approach to 80.

The red line shown in Fig. 6, 7, and 8 indicates the average total reward the robot has received in each episode. Blue points indicate the reward received in each episode.

The pre-trained model has been tested in 9 different environments with 500 steps each. Additionally, the completion percentages for each episode are shown. Obstacles in the test environment were determined randomly. According to table 1, the most successful completion percentage by robots is environment number 1.

TABLE 1. TEST VALUES PERFORMED IN DIFFERENT ENVIRONMENTS USING A PRE-TRAINED MODEL

Env / Agent	1	2	3	4	5	6	7
A	6,10	8,90	1,25	8,05	12,7	14,7	7,15
B	-1,25	16,1	5,35	-7,2	-38	-4,8	6,35
C	16,8	8,95	8,35	12,8	0,35	10,1	9,05
Completed (%)	91,5	90,6	84	77,3	75,4	84,7	83,2

Fig.10 shows the graph of the data in table 1. The yellow line shown in Fig. 10 shows the percentage of completion for each environment. The blue line indicates the total reward that robot A received. The red line indicates the total reward robot B has received. The gray line indicates the total reward the C robot has received.

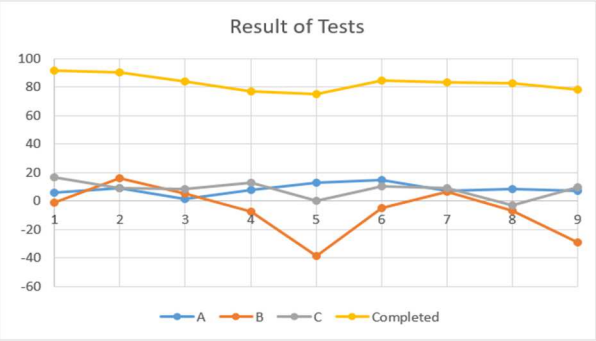

Fig.10. Test results in different environments using a pre-trained model

V. Conclusion

In the age of technology, robots now find their place in many areas of our lives. Reinforcement learning is widely used in robotics to learn complex behaviors. However, many real world applications have multi-dimensional action areas where individual actions work together to make the robot perform a desired task.

In this study, the training problem of swarm robots in a limited area is discussed. In the solution of the problem, deep reinforcement learning algorithm is trained using PPO policy and Actor-Critic model. At the end of 1500 episodes, it is concluded that three robots successfully learned the environment and tried to get the maximum score. The use of CNN artificial neural network in both actor and critic network plays an important role in the success of training. When the episode-reward graph of all three robots is examined, it is seen that they have completed the training proportionally and consistently. If one robot had received too many consecutive rewards, at least one of the other robots would fail to train. So, it can be said that the education was successful.

The average completion percentage of the test results in 9 different environments is 83.08%. This percentage can be improved with different parameters and by applying different policies. In addition, test results differ according to the location of the obstacles in the test environments and the area they cover.

Finally, the results on the testing data verify the PPO agent can adapt to different environment scenarios.

References

[1] J. Harvey, "The Blessing and Curse of Emergence in Swarm Intelligence Systems," *Foundations of Trusted Autonomy,* vol. 117, no. 3, pp. 117-124, 2018.

[2] L. Yandong, H. Zongbo and L. Hang, "Survey of convolutional neural network," *Journal of Computer Applications,* vol. 36, no. 9, pp. 2508-2515, 2016.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs.LG],* 2017.

[4] Z. Kakish, K. Elamvazhuthi and S. Berman, "Using Reinforcement Learning to Herd a Robotic Swarm to a Target Distribution," in *IEEE Robotics and Automation Letters 2020*, 2020.

[5] X. Lan, Y. Liu and Z. Zhao, "Cooperative control for swarming systems based on reinforcement learning in unknown dynamic environment," *Neurocomputing,* vol. 410, pp. 410-418, 2020.

[6] R. S. Parpinelli and H. Lopes, "New inspirations in swarm intelligence: a survey," *International Journal of Bio-Inspired Computation,* vol. 3, no. 1, pp. 1-16, 2011.

[7] M. Hüttenrauch, A. Šošic and G. Neumann, "Deep Reinforcement Learning for Swarm Systems," *Journal of Machine Learning Research,* pp. 1-31, 2019.

[8] T. Liu, H. Wang, B. Lu, J. Li and D. Cao, "Decision-making for Autonomous Vehicles on Highway: Deep Reinforcement Learning with Continuous Action Horizon," *arXiv preprint arXiv:2008.11852.,* 2020.

[9] M. Ji and M. Egerstedt, "Distributed Coordination Control of Multiagent Systems While Preserving Connectedness," *IEEE Transactions on Robotics,* vol. 23, no. 4, pp. 693-703, 2007.

[10] R. Johns., *Intelligent formation control using deep reinforcement learning,* 2018.

[11] X.-D. Zhang, "Machine learning," *A Matrix Algebra Approach to Artificial Intelligence,* pp. 223-440, 2020.

[12] Y. LeCun, Y. Bengio and G. Hinton, "Deep Learning," *Review,* no. 521, p. 436, 2015.

[13] Z. Tan and M. Karaköse, "Comparative Study for Deep Reinforcement Learning with CNN, RNN, and LSTM in Autonomous Navigation," *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI),* 2020.

[14] W. Liang, W. Huang, J. Long, K. Zhang, K.-C. Li and D. Zhang, "Deep Reinforcement Learning for Resource Protection and Real-Time Detection in IoT Environment," *IEEE Internet of Things Journal,* vol. 7, no. 7, pp. 6392 - 6401, 2020.

[15] W. Huang and W. B. Haskell, "Risk-aware Q-learning for Markov decision processes," *2017 IEEE 56th Annual Conference on Decision and Control (CDC),* 2017.

[16] N. Kantasewi, S. Marukatat, S. Thainimit and O. Manabu, "Multi Q-Table Q-Learning," *2019 10th International Conference of Information and Communication Technology for Embedded Systems (IC-ICTES),* 2019.

[17] R. S. Sutton and A. G. Barto, Reinforcement Learning:An Introduction, London: MIT Press, 2015.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv:1312.5602,* 2013.

[19] J. H. Tianpei Yang, Z. Meng, Z. Zhang, Y. Hu, Y. Cheng, C. Fan, W. Wang, Z. W. Wulong Liu and J. Peng, "Efficient Deep Reinforcement Learning via Adaptive Policy Transfer," *arXiv preprint arXiv:2002.08037,* 2020.

[20] Z. Tan and M. Karaköse, "Optimized Deep Reinforcement Learning Approach for Dynamic System," *2020 IEEE International Symposium on Systems Engineering (ISSE),* pp. 1-4, 2020.