Multi-Actor-Attention-Critic Reinforcement Learning for Central Place Foraging Swarms

Ning Yang
College of Computer Science
National University of Defense Technology
Changsha, China
yangning@nudt.edu.cn

Qi Lu
Department of Computer Science
University of Texas at San Antonio
San Antonio, USA
qi.lu@utsa.edu

Kele Xu
College of Computer Science
National University of Defense Technology
Changsha, China
kelele.xu@gmail.com

Bo Ding*
College of Computer Science
National University of Defense Technology
Changsha, China
dingbo@nudt.edu.cn

Zijian Gao
College of Computer Science
National University of Defense Technology
Changsha, China
gaozijian19@nudt.edu.cn

***Abstract*—Multiple agents with relatively low cost, decentralized control, and robustness have the advantages of completing a foraging task more efficiently than a single advanced robot. Despite many foraging algorithms are efficient in multiple robot systems, most are pre-designed or not very adaptive to different environments since they have to evolve the parameters of the foraging algorithm in each different environment. Besides, designing an efficient collision avoidance strategy for multiple agents is a challenge. Addressing these issues, we introduce the multi-actor-attention-critic(MAAC) reinforcement learning method into the multiple foraging agents. We train the foraging strategy for multiple simulated agents. We compare our approach with existing foraging algorithms for multiple robots, the Central Place Foraging Algorithm (CPFA) and the Distributed Deterministic Spiral Algorithm (DDSA). Experimental results demonstrate that our approach outperforms the two algorithms. Also, we illustrate that our approach has a better performance in avoiding obstacles and adapting to different environments.**

***Index Terms*—multi-agent reinforcement learning, swarm intelligence, foraging, multi-agent systems**

I. INTRODUCTION

Swarm robots behave well in many complex tasks for their inexpensive price and robust performance comparing with an individual robot. A huge task could be distributed to many sub-tasks by space, time, or both. Many swarm robots have the demands for searching for and collecting targets. The tasks include planetary surveys [1], survivor location in hazardous environments [2], collection of hazardous material and natural resources [3] and so on. If the system asks robots to deposit the targets in a single nest, it is a central place foraging task.

To solve the foraging problem, various approaches were proposed, such as the Central Place Foraging Algorithm (CPFA) [4] and the Distributed Deterministic Spiral Algorithm (DDSA) [5]. Both of them need to use artificially designed parameters to control the actions of robots. However, the real-world environment is too complicated for humans to design effective strategies. For robots to have the ability to complete tasks in a complex and constantly changing real world, we need robots that can automatically generate the strategies without the need of designing complex parameters.

We abstract robots into agents to control its actions, building several foraging scenarios in the multi-agent particle environment (MPE) [6] and using the idea of the multi-actor-attention-critic (MAAC) [7] method to construct effective agents that handle foraging tasks by maximizing future rewards we defined. We compared our method with the CPFA and the DDSA algorithms. First, we showed that the agents have a better foraging performance. Second, we showed that our method is more adaptive to different environments without updating the model. Finally, we proved that our method can get a better collision avoidance strategy and therefore reduce collisions in the foraging task.

The rest of this paper is structured as follows: Section II reviews some related work of foraging problems and summaries the theoretical foundation of multi-agent reinforcement learning. Section III introduces a brief notation of MAAC and our method's details. Different aspects of our model's performance will be verified in section IV. In section V, we have drawn the conclusions and the directions of our future work is also discussed.

II. RELATED WORK

In this section, we will introduce the state-of-the-art algorithms for swarm foraging and multi-agent reinforcement learning (MARL).

A. Foraging problem

The foraging problem is an abstract of many real-world tasks. In this task, robots are required to search resource and then take it back to a nest as soon as possible. Usually, each robot could hold only one resource at a time. Thus, foraging

*Corresponding author: dingbo@nudt.edu.cn

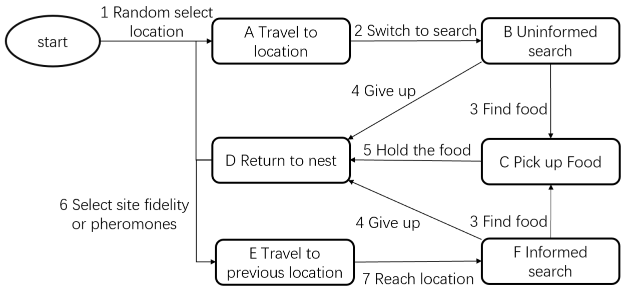

Fig. 1. The states of a robot using the CPFA algorithm.

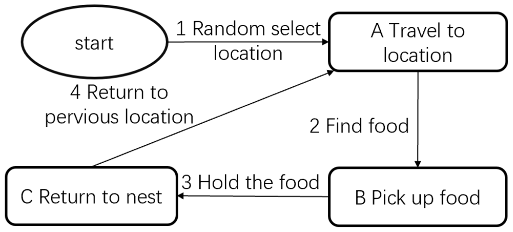

Fig. 2. The states of a robot using the DDSA algorithm.

algorithms aim to collect resources quickly within a certain time instead of collecting all the resources.

The CPFA [4] artificially designs seven parameters to control robots finishing the task. The parameters mimic seed-harvester ants, which gather resources in the desert with efficient behaviors. Fig. 1 shows the concrete states of CPFA. It uses a genetic algorithm to optimize parameters for a special case. However, CPFA needs to re-optimize the parameters to adjust the behavior every time the type of resource distribution changes, so that the robot can perform well in the new scenario.

In DDSA, agents search for resources by traveling on pre-planned spiral paths in the arena. When an agent finds a resource, it delivers the resource to the central nest and then returns to the last position where it found the resource. It resumes the search on the pre-planned path. The states of the DDSA is in Fig. 2. Because agents start foraging from the nest, they always collect the resources close to the nest first. Thus, the foraging performance of the DDSA is high at the beginning. However, avoiding obstacles is challenging in the DDSA since agents search for resources on pre-defined paths. The pre-planned paths are computed without considering the distribution of obstacles.

The NeatFA [8] uses Neuroevolution of Augmented Topologies (NEAT) to evolve a controller. It has a competitive performance with The CPFA and the DDSA. However, it still needs to design several parameters to control the robots. Our method lets the agents move into the area without any action design. Besides, neatFA evolves the same strategy for all robots, but our approach evolves individual strategy for each agent. It is helpful for future work such as cooperative foraging and different types of agents completing tasks.

The Multi-Place Foraging Algorithm (MPFA) [9]–[12] is an extension of the CPFA. It uses the same parameters as the CPFA but extends one nest to several nests. After an agent finds a resource, it will return to the nearest nest to it and exchange the waypoints only between robots in the same nest. The MPFA affects swarm foraging performance by drastically reducing the travel time. In this paper, we focus on the central place foraging task, taking the MPFA in future work.

B. Multi-Agent Reinforcement Learning

There are two basic types of methods in the reinforcement learning field: value-based and policy-based. [13] proposed a value calculation method, which records rewards of agent's states in a table. One issue is that the states of many practical tasks are continuous, a limited table cannot store unlimited state space. Deep learning method [14] introduced to Q-learning to address this problem. Later [15] was proposed to let agents can take actions in a continuous space. However, Policy Gradients (PG) algorithm has a high estimated variance due to the sparse rewards. [16] combined Q-learning and PG. An actor utilizes PG to choose actions, meanwhile, a critic estimates these actions and return the potential rewards of states in the form of value. [17] attempted to solve the problem of convergence by introducing a deep neural network [14]. [18] took the entropy into the actor-critic method for agents to better explore the environment and a stable system. To solve the challenge of multi-agent, these RL methods were extended to the field of multi-agent and achieved good results in some basic scenarios [6], [7]. Although those applications in basic scenarios of multi-agents have few elements in environments, these methods have not been applied to complex scenarios, such as foraging problems. To the best of our knowledge, this is the first paper is to solve multi-agent foraging problem through the multi-agent reinforcement learning method [19].

III. Methods

In this section, we first introduce some fundamental notations of our approach and then describe our approach in details.

A. Notations

Markov Games In this work, Markov Games [20], the multi-agent extension of Markov decision processes (MDP) has been utilized as a framework. A Markov Games is defined as a set of states S and a collection of action A_1, A_2, ..., A_N, for N agents in the environment. A state transition function $S \times A_1 \times A_2 ... \times A_N \rightarrow S^{'}$, which is a probability distribution produces the next state. Each agent i has an observation o_i and an associated reward function r_i: $S \times A_1 \times A_2 ... \times A_N \rightarrow R$. Each agent use a policy $\pi_i : o_i \rightarrow P(A_i)$ to choose actions, where P is the action probability distribution. The goal of each agent is to maximize its expected discounted returns $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \in [0, 1]$ is a discount factor and T is the total time of one episode.

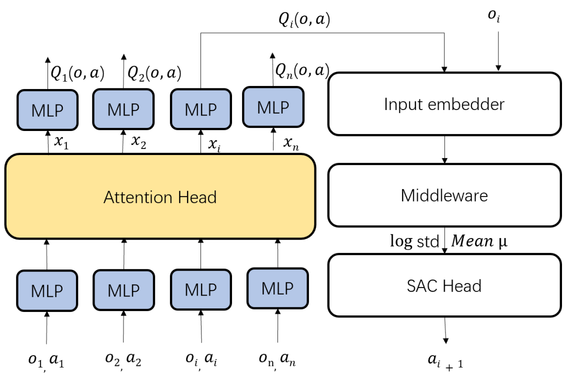

Fig. 3. The network structure of Multi-Actor-Attention-Critic(MAAC) for n agents. The MAAC has a centralized critic and decentralized actors. For agent i, Critic inputs are observations and actions of all agents. Actor input is observations of agent i.

Policy gradient It is a policy-based technology of reinforcement learning tasks. Instead of outputting value, it aims to solve the failure of continuous action space by directly output actions through maximizing the objective $J(\theta) = \mathbb{E}_{s\sim p^{\pi}, a\sim\pi_\theta}[R]$, which updates the form through the following form:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^{\pi}, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi}(s,a)] \tag{1}$$

Actor-critic Actor-critic algorithm combines Q-learning and PG so as to ameliorate the high variance of returns between episodes. A value function finds the potential rewards:

$$\begin{aligned} &\mathcal{L}_Q(\varphi) = \mathbb{E}_{(s,a,r,s')\sim D}[(Q_\varphi(s,a) - y)^2] \\ &where\ y = r(s,a) + \gamma \mathbb{E}_{a'\sim\pi(s')}[Q_{\bar{\varphi}}(s',a')] \end{aligned} \tag{2}$$

Multi-Actor-Attention-Critic MAAC is the foundation of our methods. It aims to solve the credit assignment issue by introducing COMA and attention. Using the concept of cross-entropy promotes the agent to use the surrounding information for learning more effectively when exploring by reducing the loss set as in (3).

$$\begin{aligned} &\mathcal{L}_Q(\varphi) = \sum_{i=1}^{N} \mathbb{E}_{(o,a,r,o')\sim D}[(Q_i^{\varphi}(o,a) - y_i)^2] \\ &where\ y = r_i + \gamma \mathbb{E}_{a'\sim\pi_{\bar{\theta}}(o')}[Q_i^{\bar{\varphi}}(o',a') - \alpha \log(\pi_{\bar{\theta}_i}(a_i'|o_i'))] \end{aligned} \tag{3}$$

Gradient updates following the (4).

$$\begin{aligned} \nabla_{\theta_i} J(\pi_\theta) = &\mathbb{E}_{o\sim D, a\sim\pi}[\nabla_{\theta_i} \log(\pi_{\theta_i}(a_i|o_i)) \\ &(-\alpha \log(\pi_{\theta_i}(a_i|o_i)) + Q_i^{\varphi}(o,a) - b(o, a_{\setminus i}))] \end{aligned} \tag{4}$$

where $a_{\setminus i}$ donates the set of all agents without agent i. The main architecture of the MAAC is shown in Fig. 3

B. Our methods

MPE-foraging We construct an environment called MPE-foraging to represent the foraging problem. Several agents are placed surrounding the nest. A limited time will be set to let them search in the area. The task of an agent is to find the resource and take it back to the nest. An agent only take one resource at each time. Each agent receives the information from other agents about detected resources in the environment and then takes actions using this information.

Reward function Now, we describe the design of reward function. The whole reward consists of two parts: global reward and personal reward. A centralized critic would receive the state and reward of the environment, and then calculate the joint action-value function. Each agent uses this function to choose their own action. At each time step t, each agent receives a partial observation $o_{ti} = [a_t, f_t]$, where a_t is other agents' relative position to agent i, and f_t is the resource which is sited in 10 times the agent's size area (max is 7). The personal reward of an agent in each time step t is designed as following:

$$R_{pt} = \begin{cases} r_{coll} C^t + r_{dis} D^t_{resource}, & \text{if agent hold = None;} \\ r_{coll} C^t + r_{dis} D^t_{nest}, & \text{otherwise.} \end{cases} \tag{5}$$

where the C^t is the number of collisions with other agents. The $D^t_{resource}$ is the distance between the agent and the closest resource to it. $r_{coll} = -3$ affect the penalty level for agent collision. $r_{dis} = -0.5$ affect the penalty level for the distance between the agent and the closest resource. The D^t_{nest} is the distance between the agent and the nest to it.

$$R_{gt} = r_{hold} H^t + r_{dep} D^t \tag{6}$$

Equation (6) is the global reward of an agent at time step t. H^t is the number of agents which hold resources at time step t, and the value of r_{hold} is +5. D^t is the number of agents which deposit resources in the nest at time step t. The reward of it is $r_{dep} = +5$.

Training Approach This task trains for 35000 episodes. In each episode, agents move 100 steps. In a training, actors have partial observations. Central critic has global observations. Input of the actor network for agent i is $o_i = [p_{\setminus i}, f_{pos}]$. $p_{\setminus i}$ represents other agents' relative position to agent i. f_{pos} denotes seven closest resources' relative position to agent i. Output of the actor network determines agent i's action a_i. Without any pre-designed parameters, a_i consist of five dimensions determining the force on agent i. Formula (7) shows how the force transits into the movement of one step.

$$Position_{t+1} = Position_t + f(t(a_i)) \tag{7}$$

where $t(a_i)$ transfers the five-dimensional output of the network into a two-dimensional action direction. The velocity is the result of $t(a_i)$ times acceleration. If the velocity is more than max speed, it will be scaled down to max speed. For fairness, we set the speed of agents in the DDSA and the CPFA to 0.034 m/s. The speed of agents in MAAC-foraging cannot exceed 0.04 m/s, because their speed is continuous rather than constant.

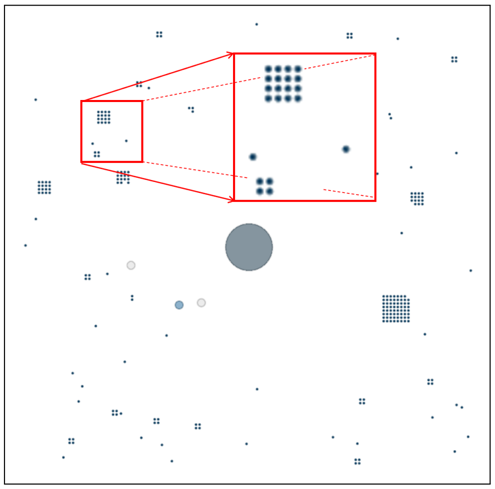

Fig. 4. Three semi-clustered distribution of foraging scenario. The big blue-gray dot in the center of the screen is the nest. The small blue-gray dots randomly scattered throughout the scene are 256 resources. The resources are in groups of 1×1, 2×2, 4×4, and 8×8. The 6 medium gray dots are agents. When the agents collect resources, they will turn to blue. When agents are delivering resources to the nest, they turn back to gray. A small area is magnified to show the clusters of resources.

TABLE I
EXPERIMENTAL CONFIGURATION IN MPE-FORAGING

Size of the arena (m)	10*10
Number of nests	1
Radius of nests(m)	0.5
Number of resources	256
Number of robots	6
Foraging step	1000
Radius of resources(m)	0.015
Radius of agents(m)	0.085

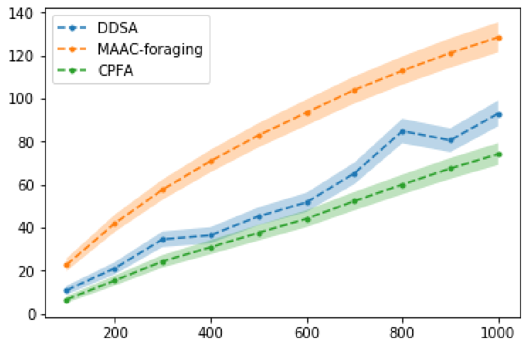

Fig. 5. Comparison of the performance of three algorithms (the CPFA, the DDSA and the MAAC-foraging) in semi-clustered scenario. The green dot is the CPFA. Blue dot is the DDSA. The orange dot is the MAAC-foraging. Experiments are 6 agents in 1000 steps for one episodes. The dots indicate the number of resources agents deposit in the nest for each 100 steps.

IV. Experiments and Results

In this section, we will introduce our experiments from three parts: the experimental environments, results of various experiments, and the corresponding analysis.

A. Experimental environments

We use the multi-agent partial environment to build our experimental environments. The multi-agent particle environment foraging scenario consists of N agents, one nest, M resource, and K blocks (K could be zero) inhabiting a two-dimensional world with continuous space and discrete-time. The agents' position and velocity both consist of two float numbers. All elements in the arena are points with their size. In each step, agents could choose an action in the environment. Agents move 1000 steps for each episode. MPE-foraging has several scenarios with different resource distribution to test agents' ability to finish the task in various scenarios. Fig. 4 shows the semi-clustered foraging scenario.

The area is 10×10m and the nest is in the center of the arena with a 0.1m radius and 4.9m away from the edge. In a random distribution, 256 resources are randomly placed. In semi-clustered distribution the same 256 resources are grouped in 1, 4, 16, and 64, each arranged in a square with each group placed randomly in the arena. In clustered distribution, the 256 resources are divided into 4 groups, each with 64 resources. There are 8 resources in each row and column, arranged in a square, randomly scattered in the area. Table I shows the configuration of the environment.

B. Foraging performance

Experiment 1 compares the performance of the MAAC-foraging model with the CPFA and the DDSA in the semi-clustered scenario. The task of the foraging problem aims to gain as many resources as possible in a limited time. Thus, this experiment separately compares the number of resources of agents collecting and depositing to nest at the step of 100, 200, 300, 400, 500, 600, 700, 800, 900, and 1000. Each plotted point in Fig. 5 is the average of ten results. We train 6 agents in the scenario for 35000 episodes, each episode the agents move 100 steps (to reduce the calculation) to search for resources and deposit them to the central nest. During testing, each agent moves 1000 steps in one episode. Only the resources deposited in the central nest successfully are counted.

It shows that the MAAC-foraging has low variance than the DDSA. In DDSA, agents cannot search the entire arena in a limited time. Agents always collect resources close to the nest first. The distributions of resources may vary the performance in some sense. The rate of the DDSA collecting resources gradually slows down as the step increases. While the MAAC-foraging could keep a stable rate. As the step grows, the resources near the nest have been taken to the nest. The DDSA needs to spend more time traveling and searching as a result of that they have to go far away from the nest.

C. Robustness

Experiment 2 shows the robustness of the MAAC-foraging model with the CPFA. This experiment aims to build model agents that can effectively use without knowing the distributions of resources ahead of time. In this experiment, both the algorithms only train once in the semi-clustered scenario. Then the model will be used in three different resource distributions.

The number in Table II is the average of 10 runs. Table

TABLE II
MAAC-FORAGING OVER DISTRIBUTIONS

Distribution ways	MAAC-foraging	CPFA
Random	145.67($\uparrow$ 9.05%)	57.7($\downarrow$ 21.91%)
Semi-clustered	133.58	73.89
Clustered	118.5($\downarrow$ 11.28%)	51.22($\downarrow$ 30.68%)

II shows how MAAC-foraging model and perform in three different distributions. When using the model trained in semi-clustered distribution to the other two distributions, the CPFA has a high change, decreasing by 21.91% and 30.68% respectively. The MAAC-foraging has less change, increasing 9.05% in random distribution and decreasing 11.28% in clustered distribution, which are both less than 15%. The MAAC-foraging performs better in random distribution than in semi-clustered distribution. It may because that resource is more clustered in semi-clustered distribution. Thus, agents need to spend more time avoiding collisions with other agents.

D. Collision Avoidance

Experiment 3 compared the ability of collision avoidance between the MAAC-foraging and DDSA. MAAC-foraging could let agents own the ability to avoid the collision with other agents and blocks.

Collision with other agents To avoid losses of devices, agents should have the ability to avoid collisions with each other. In practice, agents often need to spend a lot of time avoiding each other. We use MAAC-foraging, which allows agents to learn the ability to avoid collisions by penalizing agents when they have collisions. The experiment set six agents to train in three different scenarios for 35000 episodes, and each episode agents can take 100 steps. When testing, each episode agents take 1000 steps. The experiment counts the number of collisions with other agents between MAAC-foraging and the DDSA in different scenarios, as shown in Fig. 6. The blue bar is the result of DDSA, and the red one is the result of MAAC-foraging. From left to right are the three different resource distributions: random, semi-clustered, and clustered.

The results are an average of 10 episodes. It shows that MAAC-foraging learned a policy that can actively avoid collisions with other agents. In all food distribution, MAAC-foraging has less collision than the DDSA, which is all under one hundred. the DDSA collide more in the situation where resources are fully clustered. In clustered distribution, food is highly concentrated, and as result agents collecting food here easily lead to continuous collisions.

Collision with blocks In the real world, the scene is usually not completely smooth but has many obstacles. Thus, agents must have the ability to avoid them instead of hitting obstacles and causing damage. Based on the previous random scene of MPE-foraging, we constructed a new scene with several blocks, as in Fig. 7. We set up 6 agents in this scene to learn to forage while avoiding collisions with these blocks. In this

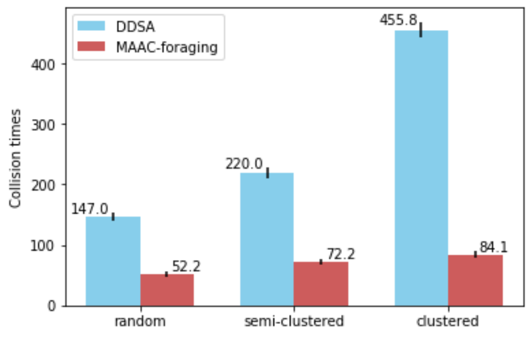

Fig. 6. Comparison of avoiding collisions with other agents of MAAC-foraging and the DDSA. Blue bar represents collision times of the DDSA. Red bar represents collision times of the MAAC-foraging.

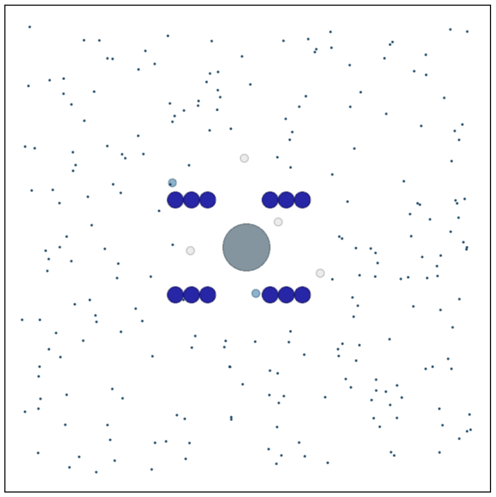

Fig. 7. The scenario of MPE-foraging random distribution with obstacles (12 dark blue points are randomly placed in the center of the area, 3 in a row). The big blue-gray dot in the center of the screen is the nest. The small blue-gray dots randomly scattered throughout the scene are 256 resources. The resources are randomly placed. The 6 medium gray dots are agents. When the agents collect resources, they will turn blue. When they put resources back in the nest, they turn back to gray.

experiment, we added 12 blocks to the scene. Agents move 100 steps in an episode and train for 30000 episodes. We record the collision times of agents with blocks in 10 episodes, in which agents move 1000 steps in one episode. Because the block is larger than agents, sometimes agents will collide with the blocks for several steps. In our setting, that calls one collision.

The total collision times with blocks is shown in Fig. 8. The CPFA has the most collisions. The parameter of the CPFA does not take care of collisions. During the agents' searching, they may just near the blocks and continuously collide the blocks. The DDSA has a low variance, which is caused by the simple strategy they have. Agents will not continue to explore the same place. No matter where the blocks appear, DDSA keeps exploring here a certain number of times. The MAAC-foraging has the smallest number of collisions, learned to bypass blocks to find food. The average number of collisions of the MAAC-

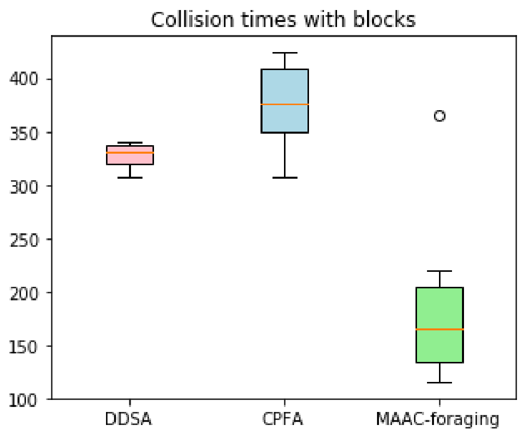

Fig. 8. Comparison of avoiding collisions with blocks of MAAC-foraging, the CPFA and the DDSA. Pink represents collision times of the DDSA. Blue represents collision times of the CPFA. Green represents collision times of the MAAC-foraging.

foraging is 43.9% less than that of the DDSA, and 50.93% less than that of the CPFA, according to the results of ten experiments.

V. Conclusion

In this work, we introduce the reinforcement learning algorithm multi-actor-attention-critic (MAAC) to train a policy that improves the foraging performance of multiple agents in an unknown environment. We have shown that MAAC-foraging outperforms other two foraging algorithms, the CPFA and the DDSA. We fix the input observation to let the agents train in a few resources environment with small steps but test in a whole resources environment with large steps. Through the deep reinforcement learning network and a simple reward function, the agents can learn how to explore resources and deliver them to the nest. The training model is remarkably simple, but it is very efficient. Besides, the experimental results show that multiple agents using MAAC-foraging are more adaptable to environmental changes like different resource distribution. Last, we demonstrate that agents trained by using MAAC-foraging have the ability of avoiding other agents and obstacles more efficient. These results demonstrate that the feasibility of incorporating fully automatic design of foraging algorithms using deep reinforcement learning.

A future work is to apply this approach to the issue of number expansion, building a scalable system, or in the jointcloud computing scenarios [21]. We also plan to utilize extra information for agents to better exploring the unknown environments.

Acknowledgment

This work is partially supported by the major Science and Technology Innovation 2030 ”New Generation Artificial Intelligence” project 2020AAA0104803 and the National Natural Science Foundation of China (No. 61751208). This work is also partially funded by the foundation of PDL (6142110190401).

References

[1] W. Fink, J. M. Dohm, M. A. Tarbell, T. M. Hare, and V. R. Baker, “Next-generation robotic planetary reconnaissance missions: a paradigm shift,” *Planetary and Space Science*, vol. 53, no. 14-15, pp. 1419–1426, 2005.

[2] A. Birk and S. Carpin, “Rescue robotics—a crucial milestone on the road to autonomous systems,” *Advanced Robotics*, vol. 20, no. 5, pp. 595–605, 2006.

[3] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[4] J. P. Hecker and M. E. Moses, “Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms,” *Swarm Intelligence*, vol. 9, no. 1, pp. 43–70, 2015.

[5] G. M. Fricke, J. P. Hecker, A. D. Griego, L. T. Tran, and M. E. Moses, “A distributed deterministic spiral search algorithm for swarms,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4430–4436.

[6] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6382–6393.

[7] S. Iqbal and F. Sha, “Actor-attention-critic for multi-agent reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 2961–2970.

[8] J. Ericksen, M. Moses, and S. Forrest, “Automatically evolving a general controller for robot swarms,” in *2017 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2017, pp. 1–8.

[9] Q. Lu, M. E. Moses, and J. P. Hecker, “A scalable and adaptable multiple-place foraging algorithm for ant-inspired robot swarms.” 2016 Robotics Science and Systems Conference (RSS) workshop on On-line decision-making in multi-robot coordination, 2016.

[10] Q. Lu, J. P. Hecker, and M. E. Moses, “The MPFA: A multiple-place foraging algorithm for biologically-inspired robot swarms,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 3815–3821.

[11] Q. Lu, J. P. Hecker, and M. E. Moses, “Multiple-place swarm foraging with dynamic depots,” *Autonomous Robots*, vol. 42, no. 4, pp. 909–926, Jan. 2019.

[12] Q. Lu, G. M. Fricke, T. Tsuno, and M. E. Moses, “A bio-inspired transportation network for scalable swarm foraging,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6120–6126.

[13] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[15] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[16] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[19] Q. Lu, G. M. Fricke, J. C. Ericksen, and M. E. Moses, “Swarm foraging review: Closing the gap between proof and practice,” *Current Robotics Reports*, pp. 1–11, 2020.

[20] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[21] H. Wang, P. Shi, and Y. Zhang, “Jointcloud: A cross-cloud cooperation architecture for integrated internet service customization,” in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017, pp. 1846–1855.