Scalable Robust Multi-Agent Reinforcement Learning for Model Uncertainty

Younkyung Jwa[1,*], Minseon Gwak[2,*], Jiin Kwak[3,*], Chang Wook Ahn[1,†], PooGyeon Park[2,†]

***Abstract*— A robust multi-agent reinforcement learning (MARL) algorithm using a nature actor has been shown to be effective in finding a robust Nash equilibrium (NE) of a Markov game with model uncertainty. However, since a game-size scaling increases the search space and challenges reaching the NE, the robust property of the algorithm is reduced in environments with many agents. This paper proposes an evolutionary diversity-maintaining population curriculum (EDPC) framework with a robust attention-based multi-agent deep deterministic policy gradient (RA-MADDPG) algorithm, which enables an efficient robust NE search by a structured search space expansion. In the EDPC framework, the MARL divides into several stages, and when moving on to the next stage, a population consisting of larger games is made with two parent games from the previous stage. We introduce reward-proportionate parent selection and reward-guided mutation methods to continue reinforcing superior agents and maintain the diversity of the population. Furthermore, the RA-MADDPG is used to solve the robust Markov game at each stage with nature actors with attention-based architectures. The scalability and robustness of the proposed method are evaluated for different numbers of agents and levels of model uncertainty.**

I. INTRODUCTION

Multi-agent reinforcement learning (MARL) has been actively studied for environments with multiple interacting decision-makers [1], such as connected autonomous vehicles [2]–[4], traffic lights [5], [6], and power grids [7]. Despite the advances in MARL, it still faces significant scalability issues because the expended number of agents exponentially increases the size of the observation and action spaces of agents [8]. Furthermore, although the scale of the environment aggravates the problem of uncertainty and reduces the effects of robust algorithms, it is not yet sufficiently studied about the MARL with both robustness and scalability.

In reinforcement learning (RL) research, robustness has been explored with the concern of various uncertain factors, such as observations [9], [10], actions [11], and models [12]–[15]. In multi-agent environments where agents influence each other, uncertainty makes it more difficult for an agent to predict what other agents will do, which may lead to performance degradation [16], [17]. In [18], a robust MADDPG (R-MADDPG) algorithm is proposed considering uncertainty in the rewards given to agents. To address the problem of model uncertainty in MARL, the algorithm leverages the idea of a zero-sum game between agents and uncertainty, treating the uncertainty as an additional agent [19]. The R-MADDPG achieves robust policies by reaching a robust Nash equilibrium (NE) of the game. However, as the system scales up, the expansion of the search space required to find robust NEs increases accordingly.

The scaling problem in MARL has been handled in uncertainty-ignorant environments [8], [20]–[22]. One notable scalable MARL framework is an evolutionary population curriculum (EPC) [21]. In the EPC, the learning process is divided into multiple stages, starting from an environment with fewer agents and gradually increasing the number of agents at each stage. Although the scaling method of the EPC contributes to a structured search space expansion, the population generated from the previous stage is formed according to some specified rules, which restricts the population diversity. Furthermore, it remains unclear how this approach can be extended to environments that involve model uncertainty.

To this end, we propose an evolutionary diversity-maintaining population curriculum (EDPC) framework robust attention-based multi-agent deep deterministic policy gradient (RA-MADDPG) algorithm for scalable robust MARL. In the EDPC framework, when a new population is formulated using the agents trained in the previous stage, the diversity of the population is maintained with two novel evolutionary methods, a reward-proportional parent selection and a reward-guided mutation, which can increase the probability of finding robust NEs in the next stage. The RA-MADDPG is an algorithm to solve each stage of the EDPC learning, where each agent utilizes an attention-based nature actor network. The nature actor networks are trained as decentralized reward estimators, which leads to the centralized critic network being trained with a relatively conservative reward instead of the received reward. As a result, we stabilize and accelerate the robust NE search through an improved structured search space expansion of the EDPC framework.

*: These authors contributed equally to this work.

†: Corresponding authors.

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, and Future Planning (2020R1A2C2005709). This research was supported by the Ministry of Science, ICT(MIST), Korea, under the Project-based AI Talent Fostering Program (RS-2022-00143911) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP). (Corresponding authors: Chang Wook Ahn; PooGyeon Park)

[1] Younkyung Jwa and Chang Wook Ahn are with the School of Artificial Intelligence, Gwangju Institute of Science and Technology (GIST), Gwangju, Korea. Email: whkdbsrud12@gm.gist.ac.kr, cwan@gist.ac.kr

[2] Minseon Gwak and PooGyeon Park are with the Department of Electrical Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Korea. Email: {minseon25, ppg}@postech.ac.kr

[3] Jiin Kwak is with the Artificial Intelligence Graduate School, Ulsan National Institute of Science and Technology (UNIST), Ulsan, Korea. Email: jiin1938@unist.ac.kr

II. PROBLEM DESCRIPTION

A. Robust Markov games

In MARL, the interaction between agents can be formulated as a Markov game G, which is denoted as

$$G = \langle A, \mathcal{S}, \{\mathcal{O}^i\}_{i\in A}, \{\mathcal{A}^i\}_{i\in A}, \{R^i\}_{i\in A}, T, \gamma\rangle, \tag{1}$$

where A is the set of agents, $\mathcal{S}$ is the state space, $\mathcal{O}^i$ is the observation space, $\mathcal{A}^i$ is the action space, R^i is the reward function for agent i, T is the transition probability, and γ is the discounting factor. The distribution of the action of the agent i at time t is determined by a policy π^i given a state at the time, i.e., $a_t^i \sim \pi^i(\cdot|s_t)$. The goal of the agent i is to maximize the long-term return J^i over policies $\{\pi^i\}_{i\in A}$, where the return is the exponential weighted sum of the reward $r_t^i = R^i(s_t, a_t^1, \cdots, a_t^N)$ at time t with the discounting factor γ, i.e.,

$$J^i(\pi^i, \pi^{-i}) = \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t^i \middle| s_0, a_t^i \sim \pi^i(\cdot|s_t), a_t^{-i} \sim \pi^{-i}(\cdot|s_t)\right], \tag{2}$$

where $\pi^{-i} = \{\pi^j\}_{j\in A, i\neq j}$ and N is the number of agents. A joint policy π_* at a Nash equilibrium (NE) satisfies

$$J^i(\pi_*^i, \pi_*^{-i}) \geq J^i(\pi^i, \pi_*^{-i}), \quad \forall i \in A. \tag{3}$$

Here, model uncertainty is introduced in Markov games to reduce the gap between the simulated Markov game and the real scenario. We defined model uncertainty as the uncertainty with respect to the reward function for each agent. With a noise rate parameter σ, a reward $\bar{r}_{t,\sigma}^i$ is given to agent i, where the reward is drawn from a truncated Gaussian distribution with the mean of the original reward r_t^i, the standard deviation of σ, and the threshold of $[-\theta, \theta]$, where θ is the truncation threshold parameter. As a result, each agent receives noisy reward information within a certain noise level. Then, the robust Markov game $\bar{G}_\sigma$ can be represented as follows,

$$\bar{G}_\sigma = \langle A, \mathcal{S}, \{\mathcal{O}^i\}_{i\in A}, \{\mathcal{A}^i\}_{i\in A}, \{\bar{\mathcal{R}}_s^i\}_{s\in\mathcal{S}, i\in A}, \{\bar{T}_s\}_{s\in\mathcal{S}}, \gamma\rangle, \tag{4}$$

where $\bar{\mathcal{R}}^i$ denotes the uncertainty set of all possible reward function values for agent i and $\bar{T}$ is the corresponding set of possible transition probabilities. Then, a policy π_*^i at a robust Nash equilibrium (NE) is defined in [18] by maximizing the worst-case expected reward, i.e.,

$$\begin{gathered}\pi_*^i \in \arg\max_{\pi^i} \min_{\substack{\bar{R}_s^i \in \bar{\mathcal{R}}_s^i \\ \bar{T} \in \bar{T}_s}} \sum_{a\in\mathcal{A}} \pi^i(a^i|s) \prod_{j\neq i} \pi_*^j(a^j|s) \\ \left(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} \bar{T}(s'|s,a)\bar{V}_*^i(s')\right),\end{gathered} \tag{5}$$

where

$$\begin{gathered}\bar{V}_*^i(s) = \max_{\pi^i(\cdot|s)} \min_{\substack{\bar{R}_s^i \in \bar{\mathcal{R}}_s^i \\ \bar{T} \in \bar{T}_s}} \sum_{a\in\mathcal{A}} \prod_{j=1}^{N} \pi^j(a^j|s) \\ \left(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} \bar{T}(s'|s,a)\bar{V}_*^i(s')\right).\end{gathered} \tag{6}$$

III. ROBUST ATTENTION-BASED MULTI-AGENT DEEP DETERMINISTIC POLICY GRADIENT

In this paper, we propose an actor-critic algorithm that uses a nature actor and an attention mechanism in networks for scalable robust MARL. In this section, we first describe the update equation for each element, which is adapted from [18], and then describe the attention-based architecture of the networks for environment scaling.

A. Robust actor-critic algorithm using nature actors

To find a robust Markov NE, we use the concept of the nature actor, which is considered as the virtual adversarial agent who acts against each agent. The nature actors can be denoted as a policy set $\pi^0 = \{\pi^{0,i}(\cdot|s)\}_{i\in A}$, where $\pi^{0,i}(s) \in \bar{\mathcal{R}}_s^i$, i.e., the policy for nature actor is within the uncertainty set of rewards for a given state s. Including the nature actor, the joint policy can be denoted as $\tilde{\pi}_\theta = (\pi_{\theta^0}, \pi_{\theta^1}, \cdots, \pi_{\theta^N})$, where π_{θ^i} is the parameterization of a policy π^i. For a joint policy $\tilde{\pi}_\theta$, the Q-function can be defined as follows:

$$\begin{aligned}\bar{Q}_{\tilde{\pi}_\theta}^i(s,a) &= \pi_{\theta^0}(s)[a] \\ &\quad + \gamma \sum_{s'\in S} P(s'|s,a) \sum_{a'\in\mathcal{A}} \prod_{j=1}^{N} \pi_{\theta^j}(a'^j|s')\bar{Q}_{\tilde{\pi}_\theta}^i(s',a').\end{aligned} \tag{7}$$

For the nature actor, the parameter $\theta^{0,i}$ is updated in the direction of reducing the gap between a predicted reward and an observed reward as well as reducing the predicted reward values itself. The policy gradient update equation for a nature actor can be represented as follows:

$$\nabla_{\theta^{0,i}} J^i(\theta) = \mathbb{E}_{s\sim\rho_{\pi_\theta, a\sim\pi_\theta(\cdot|s)}^{s_0}}[\nabla\pi_{\theta^{0,i}}(s)[a]]. \tag{8}$$

For the actors, the parameters are optimized in the way of maximizing the objective future reward with respect to the parameter θ^i for agent i. The policy gradient update equation for an actor can be denoted as follows:

$$\nabla_{\theta^i} J^i(\theta) = \mathbb{E}_{s\sim\rho_{\pi_\theta, a\sim\pi_\theta(\cdot|s)}^{s_0}}[\nabla log\pi_{\theta^i}(a^i|s)\bar{Q}_{\tilde{\pi}_\theta}^i(s,a)]. \tag{9}$$

The critic is updated with the reward of the nature actor instead of the base actor to produce a robust Q-value, which turns out to use a conservative but not significantly different reward compared to the originally given reward. The loss function that the critic minimizes is as follows:

$$\mathcal{L}(w^i) = \mathbb{E}\left[\left(y_t - \bar{Q}_{w^i}(s_t, a_t)\right)^2\right], \tag{10}$$

$$y_t = \pi_{\theta'^{0,i}}(s_t)[a_t] + \gamma\bar{Q}_{w'^i}(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^N). \tag{11}$$

B. Attention-based networks

In the EDPC framework, the RA-MADDPG starts to be performed in an environment with a small number of agents and is gradually performed in environments with a larger number of agents. To do this, it is necessary for networks to process varying sizes of state and action inputs. We utilize self-attention modules [21], which enables using the networks with a fixed number of parameters despite the

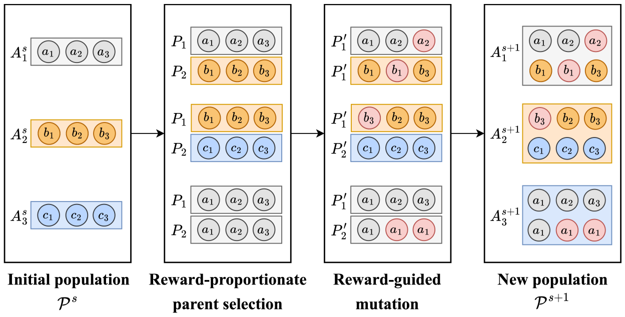

A_1^s a_1 a_2 a_3
A_2^s b_1 b_2 b_3
A_3^s c_1 c_2 c_3
P_1 a_1 a_2 a_3
P_2 b_1 b_2 b_3
P_1 b_1 b_2 b_3
P_2 c_1 c_2 c_3
P_1 a_1 a_2 a_3
P_2 a_1 a_2 a_3
P_1' a_1 a_2 a_2
P_1' b_1 b_1 b_3
P_1' b_3 b_2 b_3
P_2' c_1 c_2 c_3
P_1' a_1 a_2 a_3
P_2' a_1 a_1 a_1
A_1^{s+1} a_1 a_2 a_2 b_1 b_1 b_3
A_2^{s+1} b_3 b_2 b_3 c_1 c_2 c_3
A_3^{s+1} a_1 a_2 a_3 a_1 a_1 a_1
Initial population $\mathcal{P}^s$
Reward-proportionate parent selection
Reward-guided mutation
New population $\mathcal{P}^{s+1}$

Fig. 1: Illustration of generating a new population in the EDPC framework. A big block represents a population, a small block represents an individual (agent set), and a circle represents a gene (an agent).

change in agent number. Using the attention mechanism, the nature actor network Nat^i for agent i processes the observation and action input as follows:

$$\mathrm{Nat}^i(o_t^1, \cdots, o_t^N, a_t^1, \cdots, a_t^N) = F_i^2([F_i^1(f_i([o_t^i, a_t^i])), v_t^i]), \tag{12}$$

$$v_t^i = \sum_{j \neq i} \alpha_{ij} f_i([o_t^j, a_t^j]), \tag{13}$$

where f_i is the observation-action encoder, F_i^k is a k-layer fully connected network, o_t^i is the observation, a_t^i is the action, and v_t^i is the global attention embedding for agent i at time t. The global attention embedding v_i is the weighted sum of the observation-action encoder result for other agents. The weight α_{ij} is computed as follows:

$$\alpha_{ij} = \mathrm{softmax}(\beta_i)_j, \tag{14}$$

$$\beta_i(j) = f_i^T([o_t^i, a_t^i]) W_h^T W_l f_i([o_t^j, a_t^j]), \tag{15}$$

where $\mathrm{softmax}(u)_k = \frac{\exp(u(k))}{\sum_{i \neq k} \exp(u(i))}$ for a vector u and W_h and W_l are learning parameters. Since f_i is also composed of attention modules, its output has the same dimension regardless of the sizes of o_t^j and a_t^j. Similarly, both the critic and policy networks have architectures with attention modules like nature actor networks, and the difference is that policy networks process only observation inputs compared to the critic and nature actor networks.

IV. EVOLUTIONARY DIVERSITY-MAINTAINING POPULATION CURRICULUM LEARNING

A. Evolutionary curriculum learning for efficient NE search

We propose the EDPC as a solution for the scalable robust MARL problem. Utilizing a curriculum learning approach, where a model is first trained to solve a simple problem and gradually trained to solve more complex problems, the MARL process is divided into several stages in the EDPC. The overview of EDPC is shown in Fig. 1, which starts from the population with a small number of agents and continues on a new population of larger games. The evolutionary learning process is performed by parent selection and mutation, and the trained agents from the previous stage are reused for the next stage. From a genetic algorithm perspective, we redefine each stage as a generation, a group of agents as an individual, and an agent as a gene. Furthermore, we use the reward as a fitness metric, which determines which individual or genes will be passed on to the next generation. Through this evolutionary learning process, we can discover agents that are better suited to excel in complex environments.

Algorithm 1: EDPC

```
   Input: Population size $K$, number of stages $S$,
          mutation probability $\mu$, validation episode
          length $T_m$, test episode length $T_o$
   Output: Agent set $A^S_{i_{out}}$
 1 Initialize the population $\mathcal{P}^1 \leftarrow \{A_1^1, \cdots, A_K^1\}$
 2 Initialize and train $K$ environments
     $\mathcal{E}(A_1^1), \cdots, \mathcal{E}(A_K^1)$ with RA-MADDPG in parallel
 3 for $s \leftarrow 1$ to $S-1$ do
 4     for $k \leftarrow 1$ to $K$ do
 5         $\{r_t^i\}_{(i,t) \in A_k^s \times \mathbb{N}_{T_m}} \leftarrow$ Run $\mathcal{E}(A_k^s)$ for $T_m$
             episodes
 6     end
 7     for $k \leftarrow 1$ to $K$ do
 8         for $j \leftarrow 1, 2$ do
 9             $P_j \leftarrow A_l^s \in \mathcal{P}^s$ selected by
                reward-proportionate parent selection
                using $\{r_t^i\}$
10             $P'_j \leftarrow$ Reward-guided
                mutation$\Big(P_j, \mu, \{r_t^i\}_{(i,t) \in (A_l^s \times \mathbb{N}_{T_m})}\Big)$
11         end
12         $A_k^{s+1} \leftarrow P'_1 \cup P'_2$
13     end
14     $\mathcal{P}^{s+1} \leftarrow \{A_1^{s+1}, \cdots, A_K^{s+1}\}$
15     Train $K$ environments $\mathcal{E}(A_1^{s+1}), \cdots, \mathcal{E}(A_K^{s+1})$
         with RA-MADDPG in parallel
16 end
17 Run $\mathcal{E}(A_1^S), \cdots, \mathcal{E}(A_K^S)$ for $T_o$ episodes
18 $i_{out} \leftarrow \arg\max_{i=1,\cdots,K} \frac{1}{T_o} \sum_{t=1}^{T_o} r_t^i$
19 return $A^S_{i_{out}}$
```

The entire EDPC process is summarized in Algorithm 1. For notations, $\mathcal{P}^s$ is the population, and A_k^s is the kth individual (agent set) at sth stage. $\mathcal{E}(A)$ indicates the environment for an agent set A and it gives a reward r_t^i of agent $i \in A$ at time t. In the algorithm, we first initialize the population and train each individual using the RA-MADDPG method (lines 1 and 2). We then repeat the following steps until the final stage (lines 3-16). We conduct a short-time reward evaluation for the trained agent sets by estimating their performance over T_m episodes (lines 4-6). We select parents from the previous population using reward-proportionate parent selection (line 9) and mutate the selected parents by reward-guided mutation (line 10). We merge two parent sets to form a new agent set, effectively doubling the number of agents in the resulting individual (line 12). We form a new population with K new agent sets and train each agent set using the stored weights from the previous stage (lines 14 and 15). In the final step, we conduct a long-term reward evaluation for trained agent sets over T_o episodes and return the agent set with the highest average reward (lines 17-19). In the following subsections IV-B and IV-C, we provide a detailed explanation of reward-proportionate parent selection and reward-guided mutation, which are key components of the EDPC.

B. Reward-proportionate parent selection

One simple way to implement population curriculum learning is cloning the agent learned in the previous stage so that the individual in the next stage would have doubled agents. In the crossover phase in [21], the top-n number of individuals are selected in the order of rewards as parent individuals, and their combination is used to create the next generation. However, this method limits the population diversity; hence, we propose reward-proportionate parent selection to maintain population diversity in evolutionary learning.

Proportionate selection is a type of individual-selection mechanism used in genetic algorithms [23]. When choosing an individual from a population by the proportionate selection method, the probability to choose an individual is proportionate to their fitness value. For the parent selection in the EDPC, we use the average reward of all agents in an agent set as their fitness. To obtain the average reward, after training multiple individuals in a population $\mathcal{P}$ with RA-MADDPG, we evaluate all agent sets for T_m episodes. Since the average reward $r_A = \frac{1}{T_m|A|}\sum_{i\in A}\sum_{t=1}^{T_m} r_t^i$ of an agent set A can be negative but the proportionate selection does not allow negative fitness values, we scale the average reward r_A into r'_A as follows:

$$r'_A = e^{\alpha|r_A|/r_{max}}, \tag{16}$$

where $r_{max} = \max_{A\in\mathcal{P}} |r_A|$ and α is an exponential mapping parameter, which decides how large the gap between the highest and lowest probabilities is. Using $r'_A / \sum_{A\in\mathcal{P}} r'_A$ as the probability to select the agent set A as a parent, two parents are selected for an individual in the next generation. Therefore, whereas only the top few individuals could be selected as parents in [21], the individual, even with the lowest reward, has a probability of being selected in the EDPC.

C. Reward-guided mutation

In addition to the reward-proportionate parent selection method, we opted for mutation as an additional strategy to increase population diversity. In the EDPC, agent sets are individuals subjected to mutation, with each agent representing a gene. We utilize *replacement mutation* for all agents in an agent set, where each agent is replaced with another agent in a mutation candidate set, with the mutation probability μ.

Algorithm 2: Reward-Guided Mutation

```
   Input: Agent set $A$, mutation probability $\mu$,
          validation episode length $T_m$, evaluated
          rewards $\{r_t^i\}_{(i,t)\in A\times\mathbb{N}_{T_m}}$
   Output: Mutated agent set $A'$
1  $A' = \{\}$
2  for $i \in A$ do
3      Generate a number $s \sim U(0,1)$
4      if $s < \mu$ then
5          $C = \left\{ l \in A \middle| \frac{1}{T_m}\sum_{t=1}^{T_m} r_t^l > \frac{1}{T_m}\sum_{t=1}^{T_m} r_t^i \right\}$
6          $i' \leftarrow$ Randomly selected agent $j \in C$
7      else
8          $i' \leftarrow i$
9      end
10     $A' \leftarrow A' \cup \{i'\}$
11 end
12 return Mutated agent set $A'$
```

The mutation candidate set is determined based on two criteria: cooperative candidate selection and reward-guided candidate selection. According to the cooperative candidate selection, the mutation candidate set includes only cooperating agents in the same individual rather than all agents trained in the same stage. This criterion prevents cooperation among agents from being compromised when increasing diversity, based on the fact that agents in games learn policies by working together as a team. According to the other criteria, reward-guided candidate selection, the mutation candidate set involves agents with higher rewards than the agents to be mutated. In the conventional evolutionary learning framework in [21], there is no fitness information on genes (agents), and thus no adjustments are made for mutation candidates. However, EDPC utilizes the reward values on each agent evaluated from the validation as fitness. In Fig. 1, for example, the reward-guided mutation transforms the agent set $P_1 = \{a_1, a_2, a_3\}$ into $P'_1 = \{a_1, a_2, a_2\}$, where the agent a_2 has a higher reward evaluation than a_3. By introducing these conditions into the random selection process, we maintain the diversity of the population while increasing the probability of achieving higher rewards and reaching robust NEs.

The algorithm for the reward-guided mutation process is provided in Algorithm 2. We perform the following steps (lines 2-11) for each agent within the agent set. We generate a random number to determine whether to mutate an agent (line 3). If a mutation is decided, we execute the reward-guided mutation (lines 5 and 6). Mutation candidates consist of agents within the set that have a higher mean episode reward than the agent itself (line 5). In this case, we use the short-time reward evaluation results calculated in Algorithm 1. We randomly select an agent from mutation candidates (line 6) and create a new agent set containing the mutated agents (line 10).

V. NUMERICAL EXPERIMENTS

A. Environments

We evaluated our method using the food collection environment [21] shown in Fig. 2, which is based on an

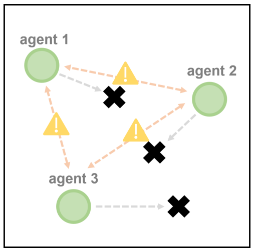


Fig. 2: Food collection environment where the MARL algorithms were evaluated. The agents (green circles) have to learn to cooperate in eating the foods (black cross marks) in order to maximize the number of foods consumed, while also avoiding collisions with other agents. Uncertainty exists in the information on the reward that each agent receives.

environment in [24]. In the environment, N cooperative agents have to learn to occupy N foods while maximizing the number of occupied foods and avoiding collisions with other agents. If any agent eats food, then each agent receives a reward of $6/N$. If two agents collide, each agent receives a reward of $-6/N$ as a penalty. Each agent also receives a navigation reward as the negative value of the distance from the nearest food.

Model uncertainty was introduced into the environments as follows. First, we sampled random values from a truncated normal distribution with a variable range of $[-\theta, \theta]$, where θ is the truncation threshold parameter. We then transformed the sampled value using the original reward value as the mean and the noise rate parameter σ as the standard deviation. The transformed values were used as the final reward values given to the agents. We used $\theta = 2$ and $\sigma = 1, 2, 3, 6$.

B. Experimental Settings

We compared four MARL algorithms, MADDPG [16], R-MADDPG [18], EPC [21], and EDPC, by evaluating their robustness with respect to *reward uncertainty*. The target system was a 12-agent system. For the EPC and EDPC, the number of stages was set to 3, and the number of agents was 3, 6, and 12 for each stage, respectively. The number of episodes was 10^5 in the first stage and 5×10^4 in the second and third stages. For a fair comparison, the number of episodes for MADDPG and R-MADDPG was set as the sum of all number of episodes of the corresponding stages of EPC and EDPC. The number of population K was 3, and the number of selections for EPC was 2. The network parameters were updated every 10^3 episode. For the EDPC, we used the mutation probability $\mu = 0.25$, validation episode length $T_m = 25$, exponential mapping parameter $\alpha = 3$, and test episode length $T_o = 10^4$.

C. Results

For the evaluation metric of the model robustness against the reward uncertainty, the average reward value for all agents was obtained since it is a fully-cooperative setting. When the average reward is higher than others though the policies were trained in a noisy environment different from the true environment, the model can be interpreted as more robust to uncertainty. After simulating $T_o = 10^4$ episodes with trained model parameters, the average reward over T_o episodes was used as the final score of learned agents. The evaluation was independently conducted in 3, 6, and 12-agent systems.

TABLE I: Robustness evaluation results

Noise rate	Number of agents	MADDPG	R-MADDPG	EPC	EDPC
1	6	-1.923	-1.958	37.775	**49.461**
	12	2.235	3.109	50.591	**54.986**
2	6	-3.051	-1.365	18.631	**33.339**
	12	0.417	2.060	36.101	**45.262**
3	6	-3.252	-2.372	0.430	**37.410**
	12	0.199	0.295	12.618	**52.755**
6	6	-5.823	-2.855	-3.796	**39.016**
	12	0.507	-1.754	7.751	**54.151**

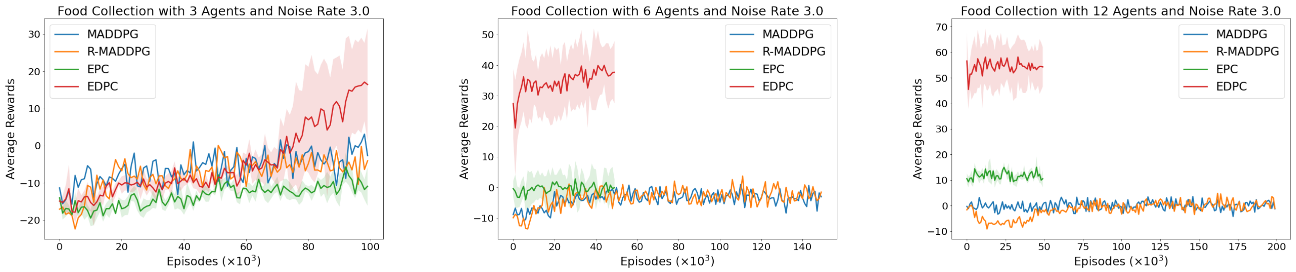
TABLE II: Effectiveness of diversity-maintaining methods

	Noise rate			
	1	2	3	6
Baseline (EPC)	50.773	42.669	37.090	31.893
+Reward-guided mutation	51.140	**52.721**	48.938	39.013
+Reward-guided mutation +Reward-proportionate parent selection	**54.986**	45.262	**52.755**	**54.151**

Fig. 3 shows the learning curve for the MARL algorithms in the training phase when the noise rate is $\sigma = 3$. When the number of agents is 3, the average reward values in Fig. 3a are similar to the comparison model because it does not include the evolutionary process in the first stage. However, due to the nature actor of the RA-MADDPG, the EDPC improves its performance as the training episode progresses. When the number of agents increases, the average reward of EDPC is much higher than others as shown in Figs. 3b and 3c.

In addition to the results from the learning curve, Table I shows the robustness evaluation results for 10^4 episodes. The results are for 6 and 12-agent systems to show how effective the curriculum learning method, which learns from a smaller system to the target system, is. For the MADDPG and R-MADDPG, the reward value tends to decrease as the noise rate increases. Although the R-MADDPG is developed regarding model uncertainty, it can be seen that the robust NE was not found well due to scaling. The EPC showed better results than MADDPG and R-MADDPG for lower noise rates but showed a rapid decrease in reward when the uncertainty increased. By contrast, the proposed EDPC showed the highest reward values among others, whether the noise rate is small or large, which implies that the EDPC can robustly learn the policies that can work well in the true environment. Furthermore, as the stage evolves in EDPC, the performance of the model increases despite the increasing number of agents.

To verify the effect of the EDPC framework, the ablation study was conducted for two components of the EDPC: the reward-proportionate parent selection and reward-guided mutation. We only represented the result of the target 12-



(a) (b) (c)

Fig. 3: Rewards that agents received in different scales of environments with model uncertainty. All reward values are the average of the reward values from the previous 10^3 episodes. The solid lines represent the average experimental results, and the shaded areas indicate the variance for three independent games. (a): 3-agent system, (b): 6-agent system, and (c): 12-agent system.

agent system. The baseline in Table II is a result of RA-MADDPG with a conventional curriculum learning, which does not consider the diversity of the population. Regardless of the noise rate, the results show that EDPC consistently learns to obtain higher rewards with each additional component. This suggests that maintaining population diversity effectively promotes the discovery of optimal robust NEs, rather than settling for suboptimal points, by exploring a diverse range of solutions.

VI. CONCLUSIONS

We proposed the EDPC and RA-MADDPG to solve the scaling problem of robust MARL concerning the model uncertainty. Experimental results showed that the EDPC with RA-MADDPG can find robust policies in scaled problems with a gap between the train and test environments. The effectiveness of the reward-proportionate parent selection and reward-guided mutation methods were also shown by an ablation study with robustness evaluation. A future direction of research is to verify the performance of the EDPC in environments where non-cooperative agents exist and develop more general and versatile MARL algorithms that can perform well.

REFERENCES

[1] G. Wang et al. A multi-group multi-agent system based on reinforcement learning and flocking. *Int. J. Control Autom. Syst.*, 20(7):2364–2378, 2022.

[2] N. Suriyarachchi et al. Multi-agent deep reinforcement learning for shock wave detection and dissipation using vehicle-to-vehicle communication. In *2022 IEEE 61st Conference on Decision and Control*, pages 4072–4077. IEEE, 2022.

[3] P. Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks*, pages 1–7. IEEE, 2020.

[4] M. Zhou et al. Development of an efficient driving strategy for connected and automated vehicles at signalized intersections: A reinforcement learning approach. *IEEE Trans. Intell. Transp.*, 21(1):433–443, 2019.

[5] K. Shibata et al. Deep reinforcement learning of event-triggered communication and control for multi-agent cooperative transport. In *2021 IEEE International Conference on Robotics and Automation*, pages 8671–8677. IEEE, 2021.

[6] T. Wu et al. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Trans. Veh. Technol.*, 69(8):8243–8256, 2020.

[7] D. Chen et al. Powernet: Multi-agent deep reinforcement learning for scalable powergrid control. *IEEE Trans. Power Syst.*, 37(2):1007–1017, 2021.

[8] G. Qu et al. Scalable multi-agent reinforcement learning for networked systems with average reward. *Advances in Neural Information Processing Systems*, 33:2074–2086, 2020.

[9] H. Zhang et al. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

[10] Z. Lu et al. Decentralized fault tolerant control for modular robot manipulators via integral terminal sliding mode and disturbance observer. *Int. J. Control Autom. Syst.*, 20(10):3274–3284, 2022.

[11] C. Tessler et al. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[12] L. Pinto et al. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[13] D. Mankowitz et al. Learning robust options. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[14] Y. Wang and S. Zou. Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34:7193–7206, 2021.

[15] V. T. Vu et al. Online actor-critic reinforcement learning control for uncertain surface vessel systems with external disturbances. *Int. J. Control Autom. Syst.*, 20(3):1029–1040, 2022.

[16] R. Lowe et al. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[17] S. Li et al. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, 2019.

[18] K. Zhang et al. Robust multi-agent reinforcement learning with model uncertainty. *Advances in neural information processing systems*, 33:10571–10583, 2020.

[19] X. Zhu et al. Data-driven multiplayer mixed-zero-sum game control of modular robot manipulators with uncertain disturbance. *Int. J. Control Autom. Syst.*, 21(2):645–657, 2023.

[20] G. Qu et al. Scalable reinforcement learning of localized policies for multi-agent networked systems. In *Learning for Dynamics and Control*, pages 256–266. PMLR, 2020.

[21] Q. Long et al. Evolutionary population curriculum for scaling multi-agent reinforcement learning. *arXiv preprint arXiv:2003.10423*, 2020.

[22] C. D Hsu et al. Scalable reinforcement learning policies for multi-agent control. In *2021 IEEE/RSJ international conference on intelligent robots and systems*, pages 4785–4791. IEEE, 2021.

[23] J. Grefenstette. Proportional selection and sampling algorithms. *Evol. Comput.*, 1:172–180, 2000.

[24] I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.