REMAX: Relational Representation for Multi-Agent Exploration

Heechang Ryu
Samsung Research
Seoul, Republic of Korea
heechang.ryu21@gmail.com

Hayong Shin
KAIST
Daejeon, Republic of Korea
hyshin@kaist.ac.kr

Jinkyoo Park
KAIST
Daejeon, Republic of Korea
jinkyoo.park@kaist.ac.kr

ABSTRACT
Training a multi-agent reinforcement learning (MARL) model with a sparse reward is generally difficult because numerous combinations of interactions among agents induce a certain outcome (i.e., success or failure). Earlier studies have tried to resolve this issue by employing an intrinsic reward to induce interactions that are helpful for learning an effective policy. However, this approach requires extensive prior knowledge for designing an intrinsic reward. To train the MARL model effectively without designing the intrinsic reward, we propose a learning-based exploration strategy to generate the initial states of a game. The proposed method adopts a variational graph autoencoder to represent a game state such that (1) the state can be compactly encoded to a latent representation by considering relationships among agents, and (2) the latent representation can be used as an effective input for a coupled surrogate model to predict an exploration score. The proposed method then finds new latent representations that maximize the exploration scores and decodes these representations to generate initial states from which the MARL model starts training in the game and thus experiences novel and rewarding states. We demonstrate that our method improves the training and performance of the MARL model more than the existing exploration methods.

KEYWORDS
Multi-Agent Reinforcement Learning; Multi-Agent Exploration

1 INTRODUCTION
Along with deep neural networks, reinforcement learning (RL) has dramatically improved in the past decades, exceeding human-level performance in challenging games [24, 31, 32]. Learning with the games or physical simulators helps to simulate and solve real-world problems, such as smart grids [9], logistics [6, 37], and distributed vehicles/robots [8, 12, 23]. The advances in RL have naturally generated a significant interest in MARL as the achievements in RL can be extended to more complex problems involving multiple interacting agents. However, the training of a MARL model is difficult even for a simple multi-agent task because it needs to learn implicitly how multiple agents interact with each other along with the environment and how these interactions induce certain task outcomes from only a reward signal. Furthermore, training the MARL model using typical exploration methods that are effective for RL has been unsuccessful because of a large number of possible interactions that the MARL model needs to explore.

Exploration methods employed by researchers for RL can be categorized into three categories: exploration bonus, goal conditioning, and initial state generation. The exploration bonus methods provide an intrinsic reward to an agent if the agent visits a state that has not been visited. Count-based exploration [2, 26, 33], and curiosity-driven exploration [4, 5, 27] are some examples of this method. Goal-conditioning methods manipulate a task’s goal in the simulator to ensure that the agent keeps receiving a reward to achieve different goals. These methods virtually endorse the agent the experience of “success” to make training faster using virtual reward signals of success [1, 13]. Finally, initial state generation methods adjust the initial state distribution in the simulator to generate initial states where the agent can easily complete a task and receive a reward [14, 15, 25, 28]. These methods assume that they can arbitrarily reset the agent into any initial state at the beginning of the training episodes in the simulator. The common purpose of these three types of exploration methods is to expose the agent to as many new and rewarding experiences as possible. In particular, goal-conditioning and initial state generation methods are mainly used for goal-oriented tasks where the reward signal is typically sparsely given and depends on whether a goal is achieved or not.

On the contrary, MARL exploration methods are limited in comparison to RL. Most MARL exploration methods focus on designing an intrinsic reward to induce certain collective behaviors of agents that are believed to help solve a multi-agent task or game. For example, intrinsic rewards are provided when one agent’s action affects other agents’ state transition [3, 36], and when all agents explore only different (novel) or same (rewardable) state space [16]. However, designing a good intrinsic reward is difficult because it requires the prior knowledge of interaction types that can help solve a multi-agent task or game. In addition, designing an effective intrinsic reward often requires an iterative reward shaping procedure until satisfactory performance is achieved.

We herein propose a learning-based exploration method called RELational representation for Multi-Agent eXploration (REMAX) to effectively generate novel and rewarding initial states for improving the training of a MARL model without designing an intrinsic reward. As an initial state generation method, REMAX has additional networks that interact with the MARL model to generate initial states in the simulator. These networks consist of two parts: state representation and generation. For state representation, REMAX employs a variational graph autoencoder (VGAE) [19] to extract latent vectors from the game states such that they can be used as an effective input for a coupled surrogate model to predict exploration scores. The exploration score quantifies the balance
between exploitation and exploration to generate states that are useful for MARL training. In particular, REMAX effectively encodes the states by representing the relationships among agents using an encoder constructed using a graph attention network (GAT) [35] in VGAE. For state generation, REMAX optimizes the surrogate model with respect to a latent vector to find new latent vectors with high exploration scores and decodes them through a VGAE decoder to generate new states. The generated states are then used as the initial states of the training episodes in the simulator to train the MARL model. To summarize, REMAX learns how to optimally generate initial states that are helpful in boosting the MARL model training through the coupled VGAE and surrogate model.

2 RELATED WORK
A representative task with sparse rewards is a goal-oriented task where a binary reward is given only when agents achieve goals (i.e., complete a task). Goal-conditioning and initial state generation methods are mainly used as RL exploration methods for goal-oriented tasks. As one of the goal-conditioning methods, HER [1] virtually exposes an agent to a task’s goal by setting the final state within a specific bound. While adjusting the initial state, REMAX also optimizes the surrogate generator network. Meanwhile, RCG [14], one of the initial state generation methods, adjusts the initial state of a training episode based on demonstrations on the task [25, 28]. DER [29] adopts a deterministic policy. MADDPG comprises individual Q-networks and policy networks for each agent. MADDPG lets the Q-network of agent i be trained by minimizing the loss (TD error) as follows: \( L(\varphi_i) = \mathbb{E}_{\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s'} \sim D} [(Q_i^\theta(o, a; \varphi_i) - y)^2], y_i = r_i + \gamma Q_j^\theta(o', a'; \varphi_i)|_{a_j = \pi_j(o'_j)}, \) where \( o = (o_1, \ldots, o_N) \) and \( a = (a_1, \ldots, a_N) \) represent the joint observation and joint action of all agents, respectively. \( D \) is an experience replay buffer that stores \((o, a, r, o')\) samples obtained from the training episodes. \( Q_i^\theta \) and \( \mu_i(o; \varphi_i) \) are target networks for the stable learning of Q and policy networks, respectively. The policy network \( \mu_i(o; \varphi_i) \), of agent i is optimized using the gradient computed as \( \mathbb{E}_{\mathbf{o}, \mathbf{a} \sim D} \left[ \nabla_{\theta_i} Q_i^\theta(o, a; \varphi_i) |_{a_i = \mu_i(o; \varphi_i)} \right] = \mathbb{E}_{\mathbf{o}, \mathbf{a} \sim D} \left[ \nabla_{\theta_i} \mu_i(o; \varphi_i) \nabla_{a_i} Q_i^\theta(o, a; \varphi_i) |_{a_i = \mu_i(o; \varphi_i)} \right]. \)

3 BACKGROUND

3.1 Multi-Agent Reinforcement Learning
We consider a partially observable Markov game [21], which is an extension of the partially observable Markov decision process for a game with multiple agents. A partially observable Markov game for N agents is defined as follows: \( S \) denotes the global state of the game; \( o_i \in S \Rightarrow O_i \) denotes a local observation correlated with the state that agent i can acquire; and \( a_i \in A_i \) is an action of agent i. The reward for agent i is obtained as a function of state s and joint action a as \( r_i : S \times A_1 \times \cdots \times A_N \rightarrow \mathbb{R} \). The state evolves to the next state according to the state transition function \( T : S \times A_1 \times \cdots \times A_N \rightarrow S \). Agent i aims to maximize its discounted return, \( R_i = \sum_{t=0}^{\infty} \gamma^t r_{i,t}, \) where \( \gamma \in [0, 1] \) is a discount factor. In MARL, each agent learns an individual policy that maps the observation to its action to maximize its expected return, which is approximated by a Q-function. While the policy can be deterministic \( (a_i = \mu_i(o_i)) \) or stochastic \( (a_i \sim \pi_i(o_i)) \), the multi-agent deep deterministic policy gradient (MADDPG) [22] adopts a deterministic policy. MADDPG comprises individuals Q-networks and policy networks for each agent. MADDPG lets the Q-network of agent i be trained by minimizing the loss (TD error) as follows: \( L(\varphi_i) = \mathbb{E}_{\mathbf{o}, \mathbf{a}, \mathbf{r}, \mathbf{o'} \sim D} [(Q_i^\theta(o, a; \varphi_i) - y_i)^2], y_i = r_i + \gamma Q_j^\theta(o', a'; \varphi_i)|_{a_j = \pi_j(o'_j)}, \) where \( o = (o_1, \ldots, o_N) \) and \( a = (a_1, \ldots, a_N) \) represent the joint observation and joint action of all agents, respectively. \( D \) is an experience replay buffer that stores \((o, a, r, o')\) samples obtained from the training episodes. \( Q_i^\theta \) and \( \mu_i(o; \varphi_i) \) are target networks for the stable learning of Q and policy networks, respectively. The policy network \( \mu_i(o; \varphi_i) \), of agent i is optimized using the gradient computed as \( \mathbb{E}_{\mathbf{o}, \mathbf{a} \sim D} \left[ \nabla_{\theta_i} Q_i^\theta(o, a; \varphi_i) |_{a_i = \mu_i(o; \varphi_i)} \right] = \mathbb{E}_{\mathbf{o}, \mathbf{a} \sim D} \left[ \nabla_{\theta_i} \mu_i(o; \varphi_i) \nabla_{a_i} Q_i^\theta(o, a; \varphi_i) |_{a_i = \mu_i(o; \varphi_i)} \right]. \)

3.2 Variational Graph Autoencoder
VGAE is an unsupervised representation learning model for graph-structured data. It consists of a graph convolutional network (GCN) [20] encoder and a simple inner product decoder. The GCN encoder performs a posterior inference for latent representations using a node feature matrix and an adjacency matrix of graph-structured data. The decoder performs an inner product between the encoded latent variables to reconstruct the adjacency matrix.
3.3 Graph Attention Network

GAT is an effective model for processing graph-structured data. It proposes to compute new node embeddings $h'_i$ for target node $i$ in a graph by aggregating previous node embeddings $h_j$ from neighboring nodes $\{ j \in N_i \}$ that are connected to target node $i$ as:

$$h'_i = \sigma(\sum_{j \in N_i} \alpha_{ij}(Wh_j)).$$

The attention coefficient $\alpha_{ij} = \text{softmax}(e_{ij})$, where $e_{ij} = a(W_i h_i, W_j h_j)$, quantifies the importance of node $j$ to node $i$ in computing $h'_i$. The attention mechanism effectively differentiates the importance of different nodes in updating the embeddings of the target node. In addition, the attention mechanism can be extended to a multi-head attention [34].

4 METHODS

Figure 1 shows how REMAX runs with a coupled MARL model. First, MARL starts training by playing a game in a simulator. When a certain number of states and associated rewards are collected from the game during MARL training, REMAX trains (1) a VGAE to represent the states as latent vectors by considering the relationships among agents and (2) a surrogate model to map the latent vectors into the exploration scores. The VGAE and surrogate model undergo an end-to-end training. Once the training is finished, REMAX gathers a set of optimized latent vectors by maximizing the exploration scores. MARL then starts the next training episodes from the new initial states generated by REMAX in the simulator and collects states again during training. The newly collected states by MARL are used to retrain REMAX. In other words, the policy training phase of MARL and the surrogate model learning are alternated during training.

4.1 State Representation Using VGAE and Surrogate Model

4.1.1 Encoder of VGAE. We propose GAT-based VGAE to transform states and their latent vectors in both directions (forward and backward) while considering the relationships among agents. Unlike the original VGAE using GCN, we use GAT in the encoder part of VGAE to effectively consider the relationships among agents with the learnable and adjustable relational attentions [22, 29].

First, the GAT encoder reshapes state $s \in \mathbb{R}^{NF}$ as a set of node features $h = \{h_1, ..., h_N \}$, where $N$ is the number of agents (nodes), and $h_i \in \mathbb{R}^F$ is the local state or observation (node feature) of agent $i$. The importance of node $j$ to node $i$ is then computed as the attention coefficient $\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{m=1}^{|N|} \exp(e_{im})}$, where $e_{ij} = \text{LeakyReLU}(v^T [Wh_i \Vert Wh_j])$. The trainable parameters $W \in \mathbb{R}^{F \times F}$ and $v \in \mathbb{R}^{2F}$ are shared for all nodes. The computed attention coefficients for every node serve as "soft" edges in a graph representing a state, which means the coefficients softly quantify the level of interactions among agents to be between 0 and 1 in the state.

Employing a multi-head attention with $K$ heads, we compute the updated node embedding $h'_i \in \mathbb{R}^{KF}$ as $h'_i = \|K_{k=1} \text{ReLU}(\sum_{j=1}^{N} \alpha^k_{ij} W^k h_j)$, where $\alpha^k$ and $W^k$ are the attention coefficients and parameters corresponding to the $k$-th head attention. In multi-agent settings, using a multi-head attention is beneficial for extracting the distinctly different interactions as there are various types of interactions among agents, especially for non-homogeneous agents.

Once $h' = \{h'_1, ..., h'_N \}$ is computed, it is concatenated as $\|_{i=1}^{N} h'_i$. The concatenated node embedding is then used to output the mean $\mu$ and standard deviation $\sigma$ for constructing the distribution for a latent vector $z$ as $z \sim q_\phi(z|s) = N(\mu, \sigma^2 I)$ with parameters $\phi$.

4.1.2 Decoder of VGAE. A multi-layer perceptron (MLP) decoder $g_\theta$ is adopted to reconstruct the states $\hat{z}$ from the latent vector $z$, i.e., $\hat{z} \sim p_\theta(z|x)$ or simply $\hat{z} = g_\theta(z)$ with parameters $\theta$.

4.1.3 Surrogate Model. While VGAE is trained to represent states as latent vectors, an MLP surrogate model $f_\psi(z)$, where $z$ is a latent vector in VGAE, is simultaneously learned to map to an exploration score $y_s = f_\psi(z)$. The exploration score can be flexibly determined depending on the MARL model. For example, if the MARL model is MADDPG, then the $Q$ and policy networks can be used to define the exploration scores $y_s$ as:

$$\frac{1}{N} \sum_{j=1}^{N} \left[ Q_j(s, a) + \lambda |Q_j(s, a) - (r_j + \gamma Q'_j(s', a'))| \right],$$

where $s$ and $a$ are, respectively, the joint state and the joint action. In MADDPG, observations and rewards corresponding to the states are obtained from the experience replay buffer. In Equation 1, the first term $\sum_{j=1}^{N} Q_j(s, a)$, which is an empirical mean of $Q$, quantifies how valuable the corresponding $(s', a')$ is for all agents. This term corresponds to the objective of policy networks to maximize their $Q$-values. Meanwhile, the second term $\sum_{j=1}^{N} |Q_j(s, a) - (r_j + \gamma Q'_j(s', a'))|$, which is an empirical mean of TD error, quantifies how novel the corresponding $(s', a')$ is for all agents. This term corresponds to the objective of $Q$-networks to minimize their TD errors as losses. In other words, the first and second terms respectively evaluate how rewardable (exploitable) and novel (explorable) the state is. It is a combination of exploration using the mean of $Q$ [7] and TD error [30]. The hyper-parameter $\lambda \geq 0$ adjusts the balance between exploitation and exploration.

The parameters $(\theta, \phi, \psi)$ of VGAE and the surrogate model are optimized by minimizing the total loss:

$$L^\text{total} = L^\text{VGAE} + \beta L^\text{surrogate},$$

where $L^\text{VGAE}$ is the negative variational lower bound expressed as $-E_z[q_\phi(z|x)] \log p_\theta(z|x) + KL(q_\phi(z|x) || p(z))$, where KL is the KL divergence, and $p(z)$ is a prior expressed as $N(0, I)$. $L^\text{surrogate}$ is

**Figure 1:** Overview of REMAX.
the mean squared error expressed as $\mathbb{E}_{z \sim q_{\phi}(z|s)}[(f_{\theta}(z) - y_{s})^2]$, where $z \sim q_{\phi}(z|s)$ is the latent vector encoded from states $s$. $\beta > 0$ is a hyper-parameter.

4.2 State Generation by Maximizing Surrogate Model

4.2.1 Optimization in Latent Space. After training the surrogate model $f_{\theta}(z)$ to predict the exploration scores of states based on their latent vectors $z$, we maximize $f_{\theta}(z)$ with respect to $z$ to find new latent vectors with high exploration scores.

First, the $N_s$ number of latent vectors $z^0 = \{z^0_1, ..., z^0_{N_s}\}$ are sampled from $\mathcal{N}(0, I)$. Every vector $z^0_i$ for $i = 1, ..., N_s$ is then optimized using the stochastic gradient ascent.

$$z^{t+1}_i = z^t_i + \delta (\nabla_{z_i} f(z^t_i) + \eta_t),$$

where $\delta$ is a step size, and $\eta_t$ is a Gaussian noise with zero mean and decreasing variance. The noise is injected to prevent the latent vectors from being stuck in local optima over a latent space. After $L$ iterations of Equation 3, a set of optimized latent vectors $z^* = \{z^*_1, ..., z^*_N\}$ is obtained.

4.2.2 Decoding Latent Vectors. Once $z^*$ is obtained, it is decoded to a set of new states $s^* = \{s^*_1, ..., s^*_N\}$ through the decoder $s^*_i = g_{\phi}(z^*_i)$ of VGAE. MARL then uses $s^*$ as the new initial states for the next training episodes in the game.

4.3 Training MARL Policy

MARL policies are then trained while playing the game with the training episodes having the initial states $s^*$ generated by REMAX. To maintain a certain level of default initial states, MARL uses the initial states generated from REMAX and the default initial states provided by the game with a certain probability, such as 0.8:0.2. After training MARL using $N_s$ training episodes, REMAX is trained again using newly collected states from MARL and generates new initial states for training MARL in the next round. In other words, for every $N_s$ number of training episodes, REMAX is retrained and generates new initial states for training MARL. The parameters $(\theta, \phi, \psi)$ of REMAX are reinitialized and retrained using only newly collected states from just previous training episodes to reflect the evolution of MARL. This is similar to employing curriculum learning in that the game condition is dynamically being adjusted depending on the current performance of the intermediate policy that is being trained.

### Table 1: Number of training episodes in maze.

| Latent space dim. | Episode($\times 10$) |
|-------------------|-----------------------|
| Random            | $640 \pm 223$         |
| HER               | $601 \pm 227$         |
| RCG               | $574 \pm 198$         |
| GENE              | $559 \pm 255$         |
| REMAX             | $456 \pm 202$         |
| MAXIM             | $495 \pm 119$         |

5 EXPERIMENTS

Figure 2 shows the environments (games) used to evaluate the performances of the proposed and comparison methods. The environments are designed to render more sparse rewards than existing environments [22, 29]. We assume that the agents in the environments have the positions and velocities (or health) of all agents as a state. In the environments, we consider random exploration and GENE as main comparison methods.

- **Goal-oriented tasks (a) and (b):** In single-and multi-agent goal-oriented tasks, we additionally consider HER and RCG for goal-oriented tasks.
- **Mixed cooperative-competitive games (c) and (d):** We exclude HER and RCG because they are not suitable for non-goal-oriented tasks. Instead, we additionally consider EDTI, which uses intrinsic rewards that quantify the influence of one agent on others.

We use MADDPG as the MARL model. Note that any MARL model can be used; however, only MADDPG, one of the most general MARL models, is used as a representative in this study because the focus of this study is on investigating the effectiveness of exploration methods, irrespective of the MARL model used. In addition, all results in this study were averaged across five different seeds.

5.1 Maze

Maze, shown in Figure 2 (a), is a single-agent environment which requires an agent (blue circle) to search for a landmark (red cross). The agent obtains $+1$ as a reward when it reaches the landmark, and $0$ otherwise. Because there is only one agent in this environment, considering the relationships among agents is unnecessary. Thus, REMAX has VGAE without GAT, which is similar to VAE. Although both REMAX and GENE use VAE, they are different in that REMAX uses the surrogate model while GENE uses KDE.
Table 2: Number of training episodes (×10) in cooperative navigation. In the table, “>2000” indicates that the policy cannot be learned to complete the task by the corresponding method within the given number of training episodes.

| Additional architecture | Number of agents |
|-------------------------|------------------|
|                         | 2    | 3    | 4    | 6    | 8    |
| Random                  | 583±271 | 834±168 | 1178±542 | >2000 | >2000 |
| HER                     | 489±140 | 790±217 | 1071±618 | >2000 | >2000 |
| RCG                     | 353±133 | 674±309 | 877±679  | >2000 | >2000 |
| GENE                    | ✓     | ✓     | ✓     | ✓     | ✓     |
| GENE with GAT           | ✓     | ✓     | ✓     | ✓     | ✓     |
| REMAX                   | ✓     | ✓     | ✓     | ✓     | ✓     |
|                         | 161±69 | 272±80 | 331±114 | 602±287 | 908±601 |

Table 1 presents the number of training episodes required for training a policy to complete the task. Here, completing a task is defined as achieving ten consecutive successes, starting from the initial states provided by the environment. Thus, a smaller number in the table means that fewer training episodes are needed to train the successful policy. The latent space dimension denotes the dimension of the latent vector $z$. GENE is reported as having the best performance with a one-dimensional latent space. The table shows that REMAX requires fewer episodes than HER (a goal-conditioning method) and RCG, although it does not need to specify the goal state during training. Note that to train and run RCG, one needs to specify the goal state. We believe that REMAX performs better than GENE because it trains the VAE and the surrogate model together in an end-to-end learning, while GENE separately trains VAE and KDE.

5.1.1 Analysis of Generating States. The first-row plots in Figure 3 show the distribution of generated states $s^*$ as green dots, and the second-row plots show how the surrogate model $f_\theta(z)$ and the optimized latent vectors $z^*$ change with the training’s progress. These states $s^*$ are generated by decoding $z^*$. As MARL and REMAX are trained with more samples, as shown in the figures, REMAX tends to generate states near the landmark because it learns that these states are easily rewardable and helpful for MARL training.

5.2 Cooperative Navigation

Cooperative navigation shown in Figure 2 (b) is a multi-agent environment where homogeneous agents (blue circles) are required to be positioned at landmarks (red crosses) at the four corners. There are as many landmarks as there are agents. Each agent obtains a reward +1 when every landmark is occupied by one agent, and 0 otherwise. Thus, all agents should be coordinated to occupy a distinct landmark and thus receive the reward.

5.2.1 Effectiveness of the REMAX Structure. We conducted ablation experiments to evaluate the synergistic effects of VGAE and the surrogate model in REMAX, and the results are summarized in Table 2. The table presents the number of training episodes required for training a policy to complete the task. In the table, GENE with GAT indicates GENE whose VAE module is replaced with VGAE with the GAT encoder. Compared with GENE, adding GAT to GENE reduces the number of training episodes, as indicated in the table. Because REMAX requires fewer episodes than the other methods, regardless of the number of agents, the use of both VGAE and the surrogate model is validated as being effective for training the MARL model, especially when the number of agents is large. In addition, the number of episodes for REMAX increases more slowly with the number of agents, which implies that REMAX enables MARL to train the policy scalably even when the number of agents increases. Meanwhile, HER and RCG require a rapidly increasing number of episodes for training the policy with an increasing number of
agents, and can no longer train the policy when the number of agents exceeds 6.

5.2.2 Analysis of Representing/Generating States. Figure 4 shows the normalized attention coefficients of GAT in the VGAE encoder in REMAX; the thicker the line is, the larger the coefficient is. The coefficients between agents increase as multiple agents with their actions (blue arrows) approach a common landmark. This trend can be observed in the second figure of Figure 4, which shows the two agents approaching the upper-right landmark, and the fifth figure of Figure 4, which shows the two agents approaching the lower-right landmark.

Figure 5 compares the two types of states: $s^0$ decoded from an initial latent vector $z^0$ and $s^*$ decoded from an optimized latent vector $z^*$. The second-row figures show $z^0$ and $z^*$ over $f_\theta(z)$. In addition, the first-and third-row figures show the decoded states, $s^0$ and $s^*$, from $z^0$ and $z^*$ using $g_\theta$. As MARL and REMAX are trained with more samples, as shown in the figure, REMAX tends to generate states near the landmarks, enabling the agents to easily complete the task.

5.2.3 Effectiveness of the REMAX Exploration Score. Table 3 presents the number of training episodes according to the types of exploration scores for the surrogate model of REMAX. $f_i(s)$ is the inverse of state visit count of agent $i$ and $m(s) = \frac{1}{N} \sum_{j=1}^{N} f_j(s)$.

| Naive reward | Intrinsic reward: Burrowing | Intrinsic reward: Covering | Proposed |
|--------------|-----------------------------|-----------------------------|----------|
| $r_{i=1}^{N} f_i(s)1[f_i(s) > m(s)]$ | $\sum_{j=1}^{N} f_j(s)1[f_j(s) < m(s)]$ | Equation 1 |
| 471 ± 193 | 392 ± 159 | 338 ± 126 | 161 ± 69 |

Figure 4: Attention of GAT in REMAX in cooperative navigation with 4 agents.

Table 3: Number of training episodes ($\times 10$) according to types of exploration scores for the surrogate model of REMAX. $f_i(s)$ is the inverse of state visit count of agent $i$ and $m(s) = \frac{1}{N} \sum_{j=1}^{N} f_j(s)$.
Figure 7: Multi-head attention of GAT in REMAX in predator-prey.

Table 5: Returns of predators in predator-prey.

| 6 vs. 2 | EDTI | GENE | REMAX |
|---------|------|------|-------|
| 3       | 1.6±0.4 | 3.7±1.0 | 14.3±4.9 | 18.7±4.6 |
| 9 vs. 3 | 2.6±0.7 | 3.2±1.1 | 9.1±2.7 | 14.4±4.1 |

Figure 6: Occupation in cooperative navigation with 20 agents.

Figure 5: Occupation rate in cooperative navigation.

Table 4: Occupation rate in cooperative navigation.

| 10 agents | Random | GENE | REMAX |
|-----------|--------|------|-------|
| 10 agents | 0.71±0.23 | 0.82±0.21 | 0.96±0.05 |
| 20 agents | 0.55±0.19 | 0.63±0.23 | 0.88±0.12 |
| 30 agents | 0.42±0.17 | 0.51±0.21 | 0.81±0.14 |

the others are designed only to induce specific behaviors of agents, while Equation 1 balances exploitation and exploration and boosts MARL training. The balance of Equation 1 changes according to the hyper-parameter $\lambda$, and it is discussed in the appendix.

5.2.4 Scalability Test. Table 4 presents the rate of occupied landmarks in cooperative navigation with more agents. In the table, REMAX has higher occupation rates than the other methods, which implies that the agents trained by REMAX cooperate effectively to occupy distinct landmarks, even when their numbers are large. Figure 6 shows how 20 agents occupy landmarks positioned in a circular shape after trained by each method. In the figure, the agents in Random and GENE tend to cluster (dotted red circles) at some landmarks, while the agents in REMAX occupy distinct landmarks.

Table 5 compares the returns of the predators. After the predators and prey are trained together by each method, the predators are validated for 200 test episodes against the prey trained by a random exploration. In the table, REMAX has higher returns than the other methods, which implies that the predators trained by REMAX cooperate effectively to capture the prey. Meanwhile, EDTI ends up achieving significantly lower returns than GENE and REMAX. These lower returns may be because EDTI cannot accurately predict the influences of intrinsic rewards. Accurate prediction requires many samples, especially when there are many agents in a continuous state-action space.

5.3 Predator-Prey

As shown in Figure 2 (c), predator-prey is a multi-agent environment where three homogeneous predator agents (red circles) aim to capture one prey agent (green diamond), which is called 3 vs. 1 predator-prey. Because the prey has a faster speed and acceleration than predators, the predators are required to cooperate for capturing the prey. In the 3 vs. 1 game, each predator obtains a reward +10 when two or more predators capture the prey simultaneously, and 0 otherwise. Meanwhile, the prey obtains a reward -10 for getting captured. If there are 6 predators and 2 preys, it is denoted as 6 vs. 2 predator-prey, and the rewards are obtained when three or more predators capture the prey simultaneously. 9 vs. 3 predator-prey is also similar to 6 vs. 2 predator-prey. In predator-prey, we used the GAT encoder with a $K = 2$ multi-head attention for REMAX because there are at least two types of relationships, predator-predator and predator-prey. We exclude HER and RCG as they are not suitable for a non-goal-oriented task. Instead, we consider EDTI, which uses intrinsic rewards quantifying the influences of one agent’s action on other agents’ rewards and transitions.

Table 5 compares the returns of the predators. After the predators and prey are trained together by each method, the predators are validated for 200 test episodes against the prey trained by a random exploration. In the table, REMAX has higher returns than the other methods, which implies that the predators trained by REMAX cooperate effectively to capture the prey. Meanwhile, EDTI ends up achieving significantly lower returns than GENE and REMAX. These lower returns may be because EDTI cannot accurately predict the influences of intrinsic rewards. Accurate prediction requires many samples, especially when there are many agents in a continuous state-action space.

5.3.1 Analysis of Representing States with Multi-Head Attention. Figure 7 shows two sets of the normalized attention coefficients, $\alpha^1$ and $\alpha^2$, of the GAT encoder with a $K = 2$ multi-head attention in REMAX. The two sets of coefficients can quantify the levels of cooperation among predators and the competition between predators and prey, respectively (upper and lower figures). In the figures, the coefficients (the thickness of the line) increase when two predators
We identified that the two predators connected by larger attention with the prey they are about to capture together (lower figure). With their actions (red arrows) approach the prey with its action (green arrows) simultaneously. For example, in the fourth figure, two predators relate to a strong attention (upper figure), and, at the same time, both predators are connected strongly with the prey they are about to capture together (lower figure). We identified that the two predators connected by larger attention coefficients tend to cooperate to capture the prey.

5.3.2 Analysis of Generating States. Figure 8 compares the two types of states: \( s^0 \) decoded from an initial latent vector \( z^0 \) and \( s^* \) decoded from an optimized latent vector \( z^* \). The second-row figures show \( z^0 \) and \( z^* \) over \( f(z) \). In addition, the first-and third-row figures show the decoded states, \( s^0 \) and \( s^* \), from \( z^0 \) and \( z^* \) using \( g\theta \). As MARL and REMAX are trained with more samples, as shown in the figure, REMAX tends to generate states where the predators are located more closely to the prey or surround the prey. This allows the predators to easily capture the prey and thus receive a reward, which expedites the MARL training.

5.4 Starcraft Multi-Agent Challenge (SMAC)

SMAC [29] is a more realistic multi-agent environment, as shown in Figure 2 (d), where three marines (agents) aim to fight against equivalent marines of game AI. SMAC originally has dense reward signals. However, we simply modify the SMAC’s rewards to be more sparse at the final timestep of each episode: a victory reward 1, while a defeat -1, with an attack reward 1 to encourage attacking the enemies, rather than running away.

Table 6 presents winning rates of controlled marines against game-AI marines in SMAC. In the table, REMAX outperforms other methods with higher winning rates.

6 CONCLUSIONS

We proposed REMAX, an exploration method that generates initial states for accelerating the training of a MARL model. Empirically, we demonstrated that REMAX generates states by representing relationships among agents, and the generated states improve the training and performance of the MARL model more than existing exploration methods.

REFERENCES

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Rob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. 2017. Hundisup: Experience replay. In Advances in neural information processing systems. 5648–5658.
[2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In Advances in neural information processing systems. 1471–1479.
[3] Wendelin Böhmer, Tabish Rashid, and Shimon Whiteson. 2019. Exploration with unreliable intrinsic reward in multi-agent reinforcement learning. arXiv preprint arXiv:1906.02138 (2019).
[4] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. 2019. Large-Scale Study of Curiosity-Driven Learning. International Conference on Learning Representations (2019).
[5] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. International Conference on Learning Representations (2019).
[6] Yongsean Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2013. An overview of recent progress in the study of distributed multi-agent coordination. IEEE Transactions on Industrial informatics 9, 1 (2013), 427–438.
[7] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. 2017. UCB exploration via Q-ensembles. arXiv preprint arXiv:1706.01502 (2017).
[8] Peter Corke, Ron Peterson, and Daniela Rus. 2005. Networked robots: Flying robot navigation using a sensor net. In Robotics research. The eleventh international symposium. Springer, 234–243.
[9] Emiliano Dall’Anese, Hao Zhu, and Georgios B Giannakis. 2013. Distributed Optimal Power Flow for Smart Microgrids. IEEE Trans. Smart Grid 4, 3 (2013), 1464–1475.

Table 6: Winning rates of controlled marines in SMAC.

|      | Random | EDTI | GENE | REMAX |
|------|--------|------|------|-------|
| 3 vs. 3 | 0.86±0.05 | 0.88±0.06 | 0.94±0.08 | 1.00±0.00 |
| 5 vs. 6 | 0.00±0.00 | 0.00±0.00 | 0.10±0.04 | 0.43±0.11 |

Figure 8: Surrogate models and generated states in predator-prey.
RELATIONAL REPRESENTATION FOR MULTI-AGENT EXPLORATION ALGORITHM

For completeness, we provide the REMAX algorithm below.

**Algorithm 1 Relational Representation for Multi-Agent Exploration**

1. Initialize an MARL model (e.g., MADDPG) for \( N \) agents
2. Initialize state buffer \( \mathcal{D} \)
3. For episode = 1 to \( M \) do
   1. Update the MARL model
   2. Store states in \( \mathcal{D} \)
   3. If episode\%\( N_s = 0 \) then
      1. Initialize VGAE and a surrogate model
      2. Train VGAE and the surrogate model together using \( \mathcal{D} \) (using Equation (1) and (2))
      3. Find \( z^* \) maximizing the surrogate model (using Equation (3))
      4. Reconstruct states \( s^* \) from \( z^* \) using VGAE
      5. Clear \( \mathcal{D} \)
   4. End if
4. Episode starts from \( s^* \) in a certain probability
5. End for

COOPERATIVE NAVIGATION

Learning Curves

Figure 9 shows the improvement in success rates when training the MARL model with the proposed and baseline exploration methods in a cooperative navigation with two agents. The success rate is defined as the rate of episodes in which ten consecutive successes are achieved, starting from the initial states provided by the environment. The figure shows that all the exploration methods improve the success rates with the training’s progress. Among them, the proposed REMAX induces the fastest convergence to a success rate of 1. The figure also shows that the GAT module added to GENE improves the performance, and using both GAT and the surrogate model (REMAX) further enhances the performance.

Choosing the Hyper-Parameter \( \lambda \)

Figure 10 shows how the value of parameter \( \lambda \), which is used to define the surrogate output in Equation 1, affects the performance (number of training episodes). When \( \lambda \) is small (the left part of Figure 9), TD error contributes more to the surrogate output, thus boosting exploration. On the other hand, when \( \lambda \) is large (the right part of Figure 9), the state action value \( Q \) contributes more to the surrogate output, thus encouraging exploitation. As shown in Figure 9, when the exploration and exploitation are well balanced by the properly selected \( \lambda \) value, the proposed method achieves the best performance (i.e., the lowest number of episodes required). We choose to use \( \lambda = 10^{-3} \), as it requires fewer training episodes than other values in the figure. It might be possible to improve the performance by optimally scheduling the variation of \( \lambda \).

![Figure 10: Performance changes according to \( \lambda \) values.](image)

MAZE

Analysis of Generating States with 2-D Surrogate Models

The first-row plots in Figure 11 represent the distribution of generated states \( s^* \) as green dots, and the second-row plots show how the surrogate model \( f^*_\psi(z) \) and the optimized latent vectors \( z^* \) change as the training proceeds. These states \( s^* \) are generated by decoding \( z^* \). Figure 11 shows the case where a two-dimensional (2-D) latent space is used for VGAE along with the associated 2-D surrogate model. As MARL and REMAX are trained with more samples, as shown in the figures, REMAX tends to generate states near the landmark because it learns that these states are easily rewardable and helpful for MARL training.

DETAILS ABOUT ENVIRONMENTS

We assume that the agents in the environments used in our study have the positions and velocities (health in SMAC, instead of velocities) of all agents as a state whose dimension is \( 4N \), which consists of \( 2N \) for x-y positions and \( 2N \) for x-y velocities of \( N \) agents. In other words, \( F \) in the main paper is 4. All the environments impose a penalty on an agent when the agent goes beyond the boundary of the environment. If \( P_a \) and \( P_l \) denote the positions of an agent and a landmark, the landmark is occupied when \( ||P_a - P_l|| < 0.1 \) in the environment.

Maze. The maze has an agent (\( N = 1 \)) within \([0, 1.5]^2\). \( P_a \) and \( P_l \) in the maze are \((1.25, 0.6)\) and \((0.25, 0.9)\), respectively.

Cooperative Navigation. The cooperative navigation has the agents (\( N = 2, 3, 4, 6, 8 \)) within \([0, 1]^2\). The agents and landmarks in the maze are \([0.25, 0.5] \times 2, \ldots \) and \([(1 - 0.25, 1 - 0.5)] \times 2, \ldots \).
the cooperative navigation are at the center and at the four corners, respectively. If there are more than four landmarks, additional landmarks are at the middle of the sides, which is the middle of the landmarks located at the corners.

**Predator-Prey.** The predator-prey has the agents \((N = 4, 8)\) within \([0, 1]^2\). The agents in the predator-prey are randomly positioned within \([0, 1]^2\).

**SMAC.** The SMAC in this study contained the agents (marines) \((N = 3)\). The three marines aim to fight against an equivalent number of marines of the game AI with very hard difficulties. SMAC originally has dense reward signals. However, we simply modify its rewards to be sparser at the final timestep of each episode: a victory reward of 1, a defeat reward of -1, and an attack reward of 1 to encourage attacking the enemies rather than running away.

**HYPER-PARAMETERS FOR EXPERIMENTS**

We used 20,000 training episodes with 50 timesteps (total 1 million timesteps) for training the models in the cooperative navigation, 50,000 training episodes with 50 timesteps (total 2.5 million timesteps) in the predator-prey, and 50,000 training episodes with 60 timesteps (total 3 million timesteps) in SMAC. We trained the models on an NVIDIA GEFORCE RTX 2080 Ti GPU with five different random seeds. To summarize, all results in this study were averaged across five different seeds. All codes used in the experiments will be released.

---

### Table 7: Hyper-parameters of MADDPG.

| Hyper-Parameter                  | Value | Value |
|----------------------------------|-------|-------|
| # Policy network MLP units       | (64, 64) | (64, 64) |
| # Q-network MLP units            | (64, 64) | (64, 64) |
| Network parameter initialization  | Xavier uniform | Xavier uniform |
| Nonlinear activation             | ReLU | ReLU |
| Policy network learning rate     | 10^-2 | 10^-2 |
| Q-network learning rate          | 10^-2 | 10^-2 |
| \(\tau\) for updating target networks | 0.95 | 0.95 |
| Replay buffer size               | 10^6 | 10^6 |
| Mini-batch size                  | 1024 | 1024 |
| Optimizer                        | Adam | Adam |

### Table 8: Hyper-parameters of REMAX.

| Hyper-Parameter                  | Value |
|----------------------------------|-------|
| \(N_s\)                         | 400   |
| \(F'\)                          | 16    |
| \(\sigma\)                      | ReLU  |
| # VGAE decoder network MLP units| (32, 32) |
| # surrogate model network MLP units | (64, 64) |
| Network parameter initialization | Xavier uniform |
| Nonlinear activation             | ReLU  |
| Training epochs                  | 3     |
| Learning rate                    | 10^-4 |
| \(\lambda\)                     | 10^-3 |
| \(\beta\)                       | 1     |
| Mini-batch size                  | 1024  |
| Optimizer                        | Adam  |
| # Loops \(L\) for optimizing \(f_\psi(z)\) w.r.t. \(z\) | 400  |
| \(\delta\)                      | 10^-1 |
| \(\eta\)                        | \(N(0, 1)\) |

### Table 9: Hyper-parameters of GENE.

| Hyper-Parameter                  | Value |
|----------------------------------|-------|
| \(N_s\)                         | 400   |
| VAE encoding space dimension     | 1     |
| # VAE encoder MLP units          | (32, 32) |
| # VAE decoder MLP units          | (32, 32) |
| Network parameter initialization  | Xavier uniform |
| Nonlinear activation             | ReLU  |
| Training epochs                  | 3     |
| Learning rate                    | 10^-4 |
| KDE bandwidth                    | 0.05  |
| KDE kernel                       | Gaussian |
| Mini-batch size                  | 1024  |
| Optimizer                        | Adam  |
Hyper-Parameters of MADDPG. The hyper-parameters of MADDPG used in the experiments are summarized in Table 7. The output layer of the policy network of MADDPG for the environments provides the action as a five-sized tensor for hold, right, left, up, and down (in SMAC, nine-sized tensor for already-dead, hold, up, down, right, left, and attack-targets for each enemy).

Hyper-Parameters of REMAX. The hyper-parameters of REMAX used in the experiments are summarized in Table 8. The number of heads $K$ for multi-head attention in REMAX is 1 for cooperative navigation and 2 for predator-prey.

Hyper-Parameters of GENE. The hyper-parameters of GENE used in the experiments are summarized in Table 9.

Hyper-Parameters of RCG. For RCG used in the experiments, $R_{\text{min}} = 0.1$ and $R_{\text{max}} = 0.9$.

Hyper-Parameters of EDTI. EDTI is based on the OpenAI implementation of PPO2 [11, 36], and its default parameters are used here. The hyper-parameters, apart from the default parameters of EDTI used in the experiments, are summarized in Table 10.

| EDTI Hyper-Parameter | Value |
|----------------------|-------|
| $\eta$               | 1     |
| $\beta_{T}$          | 10    |
| $\beta_{\text{int}}$ | 1     |
| $\beta_{\text{ext}}$ | 0.1   |
| $\beta_{\tau}$       | 0.1   |
| $\rho_{\text{plusV}}$| 0.1   |
| $\rho_{\text{plusV}}$| 0.01  |
| $\rho_{\text{ext}}$  |       |