Received April 4, 2022, accepted April 21, 2022, date of publication April 28, 2022, date of current version May 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3171053

A Novel and Efficient Influence-Seeking Exploration in Deep Multiagent Reinforcement Learning

BYUNGHYUN YOO[1], DEVARANI DEVI NINGOMBAM[2], SUNGWON YI[1], HYUN WOO KIM[1], EUISOK CHUNG[1], RAN HAN[1], AND HWA JEON SONG[1]

[1]Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, South Korea
[2]Department of Computer Science and Engineering, GITAM University, Visakhapatnam 530045, India

Corresponding authors: Byunghyun Yoo (bhyoo@etri.re.kr) and Hwa Jeon Song (songhj@etri.re.kr)

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [22ZS1100, Core Technology Research for Self-Improving Integrated Artificial Intelligence System], and in part by ETRI grants 19YE1410 and AFOSR grants FA2386-19-1-4020.

ABSTRACT Although recent years witnessed notable success for a cooperative setting in multi-agent reinforcement learning (MARL), efficient explorations are still challenging primarily due to the complex dynamics of inter-agent interactions constituting the high dimension of action spaces. For an efficient exploration, it is necessary to quantify influences that can represent interactions among agents and use them to obtain more information about the complexity of multi-agent systems. In this paper, we propose a novel influence-seeking exploration (ISE) scheme, which encourages agents to preferably explore action spaces significantly influenced by others and thus helps in speeding up the learning curve. To measure the influence of other agents in action selection, we use the variance of joint action-values with different action sets of agents that obtained by an estimation technique to lessen computation overhead. To this end, we first present an analytical approach inspired by the concept of approximated variance propagation and then apply it to an exploration scheme. We evaluate the proposed exploration method on a set of StarCraft II micromanagement as well as modified predator-prey tasks. Compared to state-of-the-art methods, the proposed method achieved performance improvements of 10% in StarCraft II micromanagement and 50% in modified predator-prey tasks approximately.

INDEX TERMS Multi-agent systems, reinforcement learning, deep learning.

I. INTRODUCTION

As successful at Go, StarCraft, and DoTA II [1], [2], multi-agent reinforcement learning (MARL) has gained increasing attention for its pivotal role in many real-world applications such as autonomous cars and humanoid robots. Nevertheless, finding an efficient solution is still considered as challenging primarily because of the high dimension of action spaces. In addition, constant interactions among multiple agents further extend the size of the problem into another level.

Unlike single agent reinforcement learning, where the reward of an action is solely given by the environment, the reward can be significantly affected by the choice of other agents in MARL. In addition, the actions of an agent are often not observable in many real-world scenarios, which makes the problem more intractable. Thus, developing an efficient exploration strategy, in particular, the ability to capture the influence of other agents has become essential in dealing with the problems that we face in our daily lives.

In cooperative MARL, a state-of-the-art approach widely adopts to learn the policy of each agent by decomposing the joint value function [3]–[8] and are known to outperform another class, an actor-critic in general [9], [10]. For exploration, even the former mostly rely on a simple mechanism called ϵ-greedy, where exploration solely resorts to a small probability of randomness. Thus, the existing exploration technique lacks the ability to capture the influence of other agents.

In this paper, we define influence as the measure of evaluating the amount of interaction and propose a novel

The associate editor coordinating the review of this manuscript and approving it for publication was Qingli Li.

TABLE 1. List of notations.

s, s'	Current and successor true state of environment
i	Agent ID
u	Action of agent
$\mathbf{u}$	Joint action for all agents
P	State transition probability
r, p	Reward and penalty
γ	Discount factor
O	Observation function
z	Individual and partial observation
τ	Action-observation history
Q_{jt}	Joint action-value function
Q_i	Individual action-value function (utility function)
Q^*	Unrestricted joint action-value function
y	Target value for training
w	Weight factor
α, β	Weight value
V_{jt}	Difference between optimal action-value and linear sum of individual action-values
$\bar{u}_i$	Action maximizing individual action-value function
$\bar{\mathbf{u}}$	Joint action maximizing individual action-value functions
μ, σ	Mean value and standard deviation
$\mathbf{u}_{-i}$	Joint action except the action of agent i
M	Number of actions
f	Function
f_m	Function for mixing network
g	Function of agent network
Σ	Covariance matrix
J	Jacobian
ν	Variance of individual action-values
L	Number of layers in a deep neural network
ϵ	ϵ-greedy policy
π	Policy
ψ	Linear combination of influence and utility function
λ_{opt}	Weight value for the equality condition in QTRAN
λ_{nopt}	Weight value for the inequality condition in QTRAN
I	Influence affected by other agents
t	Time step
t_A	Allowed computation time
T	Total number of training step
N_{max}	Maximum number of agents

TABLE 2. List of acronyms.

MARL	Multi-Agent Reinforcement Learning
ISE	Influence-Seeking Exploration
SMAC	StarCraft Multi-Agent Challenge
MPP	Modified Predator-Prey
CTDE	Centralized Training and Decentralized Execution
VDN	Value Decomposition Network
IGM	Individual-Global-Max
CW	Centrally Weighting
OW	Optimistically Weighting
RO	Relative Overgeneralization
Dec-POMDP	Decentralized Partially Observable Markov Decision Process

influence-seeking exploration (ISE) scheme as shown in Fig. 1. To estimate the influence of other agents in action selection, we use the variance of expected return with different action sets of agents that obtained by an estimation technique to lessen computation overhead. To this end, we first present an analytical approach inspired by the concept of an approximated variance propagation and apply it to an exploration scheme. Note that we limit the scope of MARL approach under analysis to value-based ones since 1) it outperforms the others in general and 2) it makes our analysis on the inter-agent influence simple.

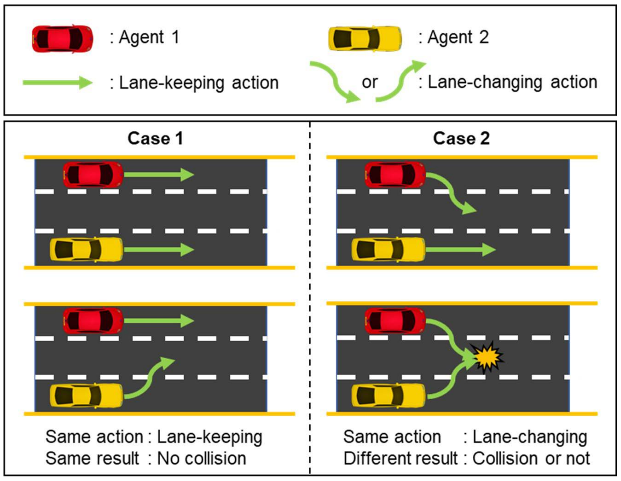

: Agent 1
: Agent 2
: Lane-keeping action
or
: Lane-changing action
Case 1
Case 2
Same action : Lane-keeping
Same result : No collision
Same action : Lane-changing
Different result : Collision or not

FIGURE 1. An example of influence-seeking exploration in a two-car system. In Case 1, when agent 1 (red car) maintains the lane, the result (collision or not) is not changed by agent 2's actions. However, in Case 2, when agent 1 changes lanes, the result is significantly changed by agent 2's actions. From the perspective of agent 1, the result of lane-changing behavior is highly influenced by other agents, and this behavior represents an area where interactions between agents occur greatly. Therefore, the lane-changing behavior of agent 1 is regarded as a behavior that is highly influenced by other agents, and the proposed method, influence-seeking exploration, refers to a method to frequently select this kind of behavior.

We evaluated the proposed exploration scheme under two popular experimental environments: StarCraft multi-agent challenge (SMAC) [11] and modified predator-prey (MPP) which is a small modification to well-known predator-prey [12]. In the simulation-based experiments, the proposed scheme showed improved performance approximately 10% in SMAC tasks and 50% in MPP tasks on average. In particular, the proposed scheme significantly accelerated the convergence speed of the value-based methods used as the base model. Specifically, the contributions of this paper are described as follows:

- We proposed a new exploration method, influence-seeking exploration, for considering influences that one agent receives from multiple agents at the same time in MARL.
- We proposed an approximation method based on the concept of variance propagation to reduce computational cost when calculating the influence among multiple agents.

The rest of this paper is organized as follows. In Section II, we summarize the related work on value-based methods and exploration strategies in MARL. We describe the previous research to be used as a baseline for the proposed scheme in Section III. The proposed scheme is detailed in Section IV. In Section V, the simulation results are presented followed by the concluding remarks in Section VI.

II. RELATED WORK

A. PRIMARY STUDY IN RL AND MARL

Reinforcement learning (RL), in which an agent learns a sequence of actions that maximizes its reward in a given

TABLE 3. Comparison between highly relevant studies and the proposed method. A task that is solved under a monotonic constraint, which was introduced in [4], is called a monotonic task. A task that can be solved when the monotonic constraint is slightly relaxed is called a low non-monotonic task, otherwise, it is called a high non-monotonic task. MGS indicates a Multi-domain Gaussian Squeeze task.

Algorithm	Method	Advantage	Disadvantage	Test environment	Type of RL
QMIX [4]	Monotonic value factorization	High performance in monotonic tasks	Low performance in non-monotonic tasks	SMAC [11]	Value-based
QTRAN [5]	Non-monotonic value factorization	High performance in non-monotonic tasks	Slow convergence and low scalability	MGS [5], MPP [5]	Value-based
WQMIX [6]	Weighted projection in value factorization	High performance in low non-monotonic tasks	Low performance in high non-monotonic tasks and inefficient exploration	SMAC [11], MPP [5]	Value-based
QPLEX [21]	Duplex dueling architecture in value factorization	Fast convergence and high scalability	Low performance in high non-monotonic tasks and inefficient exploration	SMAC [11], MPP [5]	Value-based
MAVEN [26]	Entropy-based exploration	Exploration for diversity of samples	No consideration for influences among agents	SMAC [11]	Value-based
EDTI [29]	Influence-based exploration as intrinsic reward	Efficient exploration considering influences between two agents	Focusing on pairwise interaction	Push-box [29], Island [29]	Policy-based
ISE	Influence-based exploration as behavior policy	Efficient exploration considering influences among multiple agents	Only considering interaction with agents	SMAC [11], MPP [5]	Value-based

environment, is a field of machine learning that has been continuously studied with the development of computing devices. In particular, since the advent of Deep Q-network [13], a huge number of RL studies have been conducted [14]–[17], and various related fields have also begun to receive attention. Recently, not only single-agent RL but also MARL has attracted considerable interest considering real-world applications and scalability of RL algorithms, and thus numerous MARL algorithms have been developed.

To obtain decentralized policies that enable collaboration, recent MARL studies are based on the centralized training and decentralized execution (CTDE) paradigm. This paradigm is that the agents share their observations during the training for learning cooperative behavior, and the trained behavior is decomposed to be applicable only with observations of individual agents. The CTDE paradigm was successfully applied in [10] and these methods have been advanced for complex cooperative and competitive scenarios [18]–[20].

B. VALUE FACTORIZATION METHOD

For a cooperative setting in MARL, value-based methods which train each agent by value decomposition of joint action-values have also been significantly studied and exhibit state-of-the-art performances. Value Decomposition Networks (VDN) [3] was proposed, which train the joint action-value function as the sum of decomposed action-value functions. To improve the model structure of VDN, QMIX [4] proposes a mixing neural network to connect the joint action-value function to decomposed action-value functions called an individual utility function using a monotonic constraint. QPLEX [21] introduces a new framework using a duplex dueling architecture to encode Individual-Global-Max (IGM) conditions in the form of a neural network. For non-monotonic multi-agent tasks, QTRAN [5] balances between the suboptimality and decentralization by introducing relaxed L2 penalties in the loss function of RL. WQMIX [6] improves QMIX by weighting the loss function for sub-optimal actions to overcome the limitation of restricted function class due to the monotonic constraint. Recently, distributional RL and risk-sensitive RL, which have previously been successful in single-agent scenarios, have been extended to multi-agent scenarios [22], [23]. In addition, role-based learning [24] was proposed for decomposing multi-agent tasks.

C. EXPLORATION FOR MULTI-AGENT SYSTEM

Although value-based methods in MARL achieve notable progress, the studies of relevant exploration methods are not much in terms of handling large action spaces and interactions among agents at the same time. Entropy-based exploration algorithms for value-based methods [25], [26] accelerate exploration for large action spaces, but interactions among agents cannot be considered in these methods. Another popular method for exploration is curiosity-based exploration [27], [28]. In these studies, authors addressed the problem by giving a small memory of recently taken actions, and previously unseen actions are preferably selected. This method enables an effective exploration of a wide range of actions; however, it is still not the fundamental solution for considering interaction among agents.

To consider interactions among agents, intrinsic rewards for influence-based exploration were proposed in multi-agent systems [29]. This is one conspicuous relevant research, which models the influence of one agent on the transition dynamics of other agents based on the mutual information and *value of interaction*. However, there are two major differences between the proposed method and [29]: the difference in application method and the number of agents that can be handled. [29] focuses on finding the influence between each pair of agents, which has limited application to numerous agents. On the other hand, the influence by more than two agents can be calculated in our method based on an analytical approximation. Also, [29] uses the form of intrinsic reward, our idea is directly applied to the general action-selection scheme, ϵ-greedy policy. Finally, as a summary of the related works, we show Table 3 comparing the main references and this study.

III. BACKGROUND

A. DEC-POMDP

A fully cooperative multi-agent task can be formulated as *decentralized partially observable Markov decision process* (Dec-POMDP) [30], which consists of a tuple $\mathcal{G} =< \mathcal{S}, \mathcal{U}, P, r, \mathcal{Z}, O, N, \gamma \ >$, where $s \in \mathcal{S}$ is the true state of the environment. At each time step, each agent $i \in \mathcal{N} := 1, \ldots, N$ selects an action $u_i \in \mathcal{U}$, forming a joint action vector, $\mathbf{u} \in \mathcal{U}^N$. All state transition dynamics are conducted according to the state transition function $P(s'|s, \mathbf{u}) : \mathcal{S} \times \mathcal{U}^{\mathcal{N}} \longmapsto [0, 1]$, where $s' \in \mathcal{S}$ is the successor state from the current state s. All agents share the same joint reward function $r(s, \mathbf{u}) : \mathcal{S} \times \mathcal{U}^{\mathcal{N}} \longmapsto \mathbb{R}$, and $\gamma \in [0, 1)$ is the discount factor. All agents have their individual and partial observations $z \in \mathcal{Z}$ according to observation function $O(s, i) : \mathcal{S} \times \mathcal{N} \longmapsto \mathcal{Z}$. The action-observation history for an agent i is $\tau_i \in \mathcal{T} := (\mathcal{Z} \times \mathcal{U})^*$ on which it conditions its policy $\pi(u_i|\tau_i) : \mathcal{T} \times \mathcal{U} \longmapsto [0, 1]$.

B. IGM CONDITION

Value decomposition for value-based methods in MARL should satisfy the Individual-Global-Max (IGM) condition described in (1):

$$\underset{\mathbf{u}}{\operatorname{argmax}}\, Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u_1} Q_1(\tau_1, u_1) \\ \vdots \\ \operatorname{argmax}_{u_N} Q_N(\tau_N, u_N) \end{pmatrix}, \tag{1}$$

where Q_{jt} is the joint action-value function to estimate the expectation of discounted cumulative rewards, $\boldsymbol{\tau}$ is the joint action-observation history, $\mathbf{u}$ is the joint action, and Q_i is the individual utility function which acts as an action-value function for each agent. With this condition, Q_{jt} can be factorized by Q_i, which indicates that the optimal joint actions of all agents can be obtained using the collection of individual optimal actions of each agent.

C. QMIX

The monotonic constraint for the joint action-value function was introduced as the sufficient condition for IGM in QMIX [4] as follows:

$$\frac{\partial Q_{jt}}{\partial Q_i} \geq 0. \tag{2}$$

Q_i is the function of the individual action-observation history and the current action, $Q_i = g(\tau_i, u_i)$, and Q_{jt} is the function of the individual utilities and states, $Q_{jt} = f_m(Q_i, s)$. g and f_m are also called agent network and mixing network, respectively. The agent network is constructed as Deep Recurrent Q-Networks (DRQN) [31]. The mixing network is a feed-forward network that receives the output of the agent network as input. In QMIX, separate hypernetworks that take states as input produce the weights of the mixing network which are constrained by using non-negative weights to satisfy Eq 2.

QMIX is learned by minimizing the squared temporal difference (TD) loss of joint action-value function:

$$\sum_{k=1}^{b} (Q_{jt}(\boldsymbol{\tau}, \mathbf{u}, s) - y_k)^2, \tag{3}$$

where b is the batch size of the data sampled from the replay buffer and $y_k = r + \gamma \max_{\mathbf{u}'} Q_{jt}(\boldsymbol{\tau}', \mathbf{u}', s')$ is treated as a fixed target. QMIX is a practical and powerful method satisfying the IGM condition; however, it sometimes cannot find the optimal policy with the restricted function class due to the monotonic constraint.

D. WQMIX

To overcome the limitation of QMIX, the unrestricted joint action-value function $\hat{Q}^*$ is additionally trained in WQMIX [6]. Q_{jt} has the restricted function class because of the monotonic constraint, whereas $\hat{Q}^*$ has no constraint. $\hat{Q}^*$ also has agent networks and a mixing network as QMIX, and the structure of agent network is same as QMIX. The mixing network is also a feed-forward network using the state and the output of agent networks as input; however, separate hypernetworks are not used. Based on Q_{jt} and $\hat{Q}^*$, each joint action is weighted according to its importance. As the weighting methods in WQMIX, two types of weighting were proposed: centrally-weighting and optimistically-weighting. Centrally weighting (CW) is worked by approximating the maximal joint action over the unrestricted action-value function $\hat{Q}^*$. Unlike CW, Optimistically weighting (OW) does not use any approximation and explicitly weight the joint actions based on Q_{jt}. Detailed equations for CW and OW are described in (4) and (5), respectively.

$$w(s, \mathbf{u}) = \begin{cases} 1 & \hat{y}_k > \hat{Q}^*(\boldsymbol{\tau}, \hat{\mathbf{u}}^*, s) \text{ or } \mathbf{u} = \hat{\mathbf{u}}^*, \\ \alpha & \text{otherwise} \end{cases} \tag{4}$$

$$w(s, \mathbf{u}) = \begin{cases} 1 & \hat{y}_k > Q_{jt}(\boldsymbol{\tau}, \mathbf{u}, s) \\ \alpha & \text{otherwise} \end{cases} \tag{5}$$

where w is the weighting factor, α is the weight value for suboptimal action, $\hat{\mathbf{u}}^* = \operatorname{argmax}_{\mathbf{u}} Q_{jt}(\boldsymbol{\tau}, \mathbf{u}, s)$, and $\hat{y}_k := r + \gamma \hat{Q}^*(s', \boldsymbol{\tau}', \operatorname{argmax}_{\mathbf{u}'} Q_{jt}(\boldsymbol{\tau}', \mathbf{u}', s'))$ is treated as a fixed target. The unrestricted joint action-value function $\hat{Q}^*$ is trained using TD loss, and the restricted joint action-value function Q_{jt} is trained with CW or OW as follows:

$$\sum_{k=1}^{b} (\hat{Q}^*(\boldsymbol{\tau}, \mathbf{u}, s) - \hat{y}_k)^2, \tag{6}$$

$$\sum_{k=1}^{b} w(s, \mathbf{u})(Q_{jt}(\boldsymbol{\tau}, \mathbf{u}, s) - \hat{y}_k)^2. \tag{7}$$

QMIX with CW is called Centrally-Weighted QMIX (CW-QMIX) and QMIX with OW is called Optimistically-Weighted QMIX (OW-QMIX).

Algorithm 1 Calculation of Influence

```
1: Input: agent ID $i$, individual utility function $Q_i$, joint action-value function $Q_{jt}$, variance of individual utilities $v_i$
2: Initialize covariance matrix $\Sigma_i^0$ with $v_i$ according to (14)
3: for each layer of mixing network $l = 1$ to $L$ do
4:     Compute covariance matrix in $l$-th layer $\Sigma_i^l$ according to (13)
5: end for
6: Set Influence $I_i = \Sigma_i^L$
7: return $I_i$
```

Algorithm 2 Influence-Seeking Exploration

```
1: Input replay buffer $\mathcal{D}$, influence $I_i$, individual utility function $Q_i$, weight value $\beta$
2: while step < step$_{\max}$ do
3:     $t = 0$, $s^0$ = initial state
4:     while $s^t \neq$ terminal state and $t <$ episode limit do
5:         for each agent $i$ do
6:             Get current observation $z_i^t$
7:             Set $\tau_i^t = \tau_i^{t-1} \cup \{(z_i^t, u_i^{t-1})\}$
8:             Compute $\psi_i$ according to (16)
9:             Compute $u_i^t$ according to (17)
10:        end for
11:        Get reward $r^t$ and next state $s^{t+1}$
12:        Store $\{s^t, \mathbf{u}^t, r^t, s^{t+1}\}$ in $\mathcal{D}$
13:        $t = t + 1$, step = step + 1
14:    end while
15: end while
```

E. QTRAN

Another sufficient condition for IGM was proposed and proved in QTRAN [5]:

$$\sum_{i=1}^{N} Q_i(\tau_i, u_i) - Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) + V_{jt}(\boldsymbol{\tau}) = \begin{cases} 0 & \mathbf{u} = \bar{\mathbf{u}} \\ \geq 0 & \mathbf{u} \neq \bar{\mathbf{u}}, \end{cases} \tag{8}$$

where

$$V_{jt}(\boldsymbol{\tau}) = \max_{\mathbf{u}} Q_{jt}(\boldsymbol{\tau}, \mathbf{u}) - \sum_{i=1}^{N} Q_i(\tau_i, \bar{u}_i)$$
$$\bar{\mathbf{u}} = [\bar{u}_i]_{i=1}^N, \ \bar{u}_i = \operatorname*{argmax}_{u_i} Q_i(\tau_i, u_i).$$

QTRAN has larger expressiveness with the constraint described above, and thus QTRAN can solve the task which has a high degree of cooperation.

The methods described above basically use the ϵ-greedy exploration with their trained individual utility functions. In complex scenarios, it often fails even when performing many iterations. This means that the exploration is not efficiently performed in a large action space with constant interactions among agents.

IV. INFLUENCE-SEEKING EXPLORATION

In this section, we describe the definition and calculation of influence and its application to exploration strategy.

A. INFLUENCE BY OTHER AGENTS

1) DEFINITION OF INFLUENCE

As the first step, influences by other agents need to be defined. In general, the fact that a factor has a large influence on a certain event means that the result of the event can be greatly changed by that factor. From the point of view of MARL, since the outcome of an event is considered as the expectation of return, we define influences by other agents as the variance of the expectation of return according to the actions of other agents. For the variance of expected return, joint action-values for different actions of other agents are calculated based on the joint action-value function which is trained for estimating expected return over a joint action. Consequently, the influence based on the variance can be estimated as follows:

$$\mu_i = \frac{1}{M_{-i}} \sum_{m=1}^{M_{-i}} Q_{jt}(u_i, \mathbf{u}_{-i,m}), \tag{9}$$

$$I_i = \frac{1}{M_{-i}} \sum_{m=1}^{M_{-i}} \left(Q_{jt}(u_i, \mathbf{u}_{-i,m}) - \mu_i\right)^2, \tag{10}$$

where I_i indicates the influence of agent i affected by other agents, μ_i indicates the averaged joint action-values for different actions of other agents, $\mathbf{u}_{-i}$ indicates a joint action except actions of agent i, and M_{-i} indicates the number of all $\mathbf{u_{-i}}$. This calculation can be performed by sampling different joint action values; however, an excessive number of samples are required to calculate the variance accurately for a large number of agents and actions, which demands a high computational cost.

2) APPROXIMATION FOR CALCULATING INFLUENCE

To reduce computational overhead, sampling-free estimation based on approximated variance propagation [32] is employed. For the approximated variance propagation, a covariance matrix of inputs and Jacobian of the estimated function should be calculated or modeled to transform a covariance of inputs to an approximated output variance. For example, the propagation in a bivariate function $f(x, y)$ is performed as follows:

$$\sigma_f^2 \approx \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \sigma_y^2 + 2\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\sigma_{xy}, \tag{11}$$

where σ_f^2 is the variance of the output, σ_x^2 is the variance of x, σ_y^2 is the variance of y, and σ_{xy} is the covariance between x and y. Equation 11 can be reformulated into a matrix form as follows:

$$\sigma_f^2 \approx \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}, \tag{12}$$

where the innermost matrix is known as the covariance matrix and the outermost matrix as the Jacobian matrix.

To apply the approximated variance propagation for a deep neural network, the propagation would be performed layer by layer, which requires the covariance and Jacobian matrix at each layer as follows:

$$\Sigma_i^l \approx J^{l-1} \Sigma_i^{l-1} (J^{l-1})^T, \tag{13}$$

where Σ_i^l indicates the covariance matrix of l-th layer for agent i, and J^{l-1} is the Jacobian of the function mapping between $l-1$-th layer and l-th layer. At each layer, the non-linearity of activation functions should be handled for

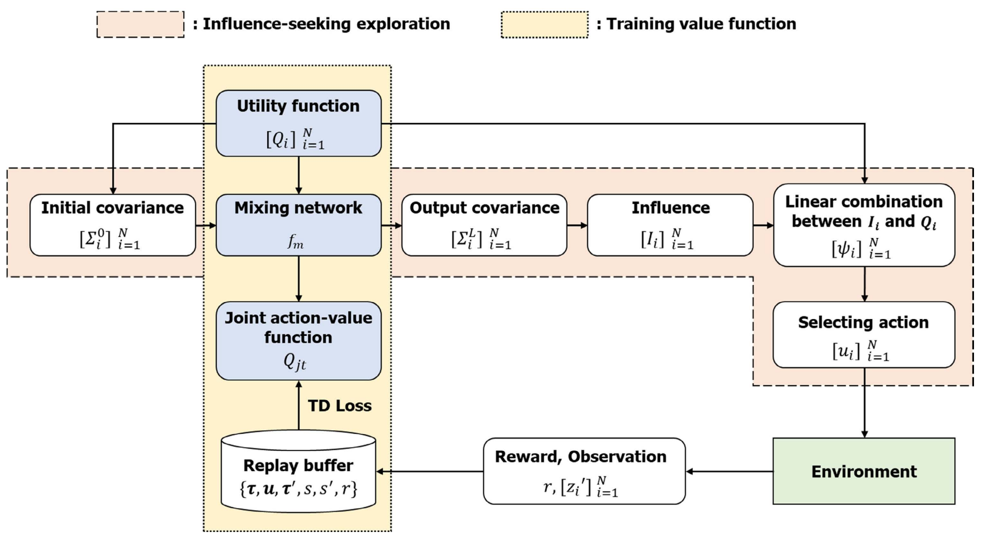


FIGURE 2. A diagram of training based on ISE. The red box indicates the procedure of ISE, and the yellow box indicates the training of value function. Based on ISE, agents choose actions to get a reward and observations. Actions, observations, and rewards are stored in the replay buffer. The joint action-value function and utility functions are trained using the data sampled from the replay buffer based on TD loss.

Jacobian. In [32], the first-order Taylor expansion of various activation functions was presented to approximate the non-linearity, and thus we also employ those approximations. In short, without calculation of (10) by sampling, an output variance can be estimated using (13) with the Jacobian and the input covariance.

Our approach requires additional matrix multiplications in each layer of the neural network as in (13) to calculate influences among agents. Therefore, the time complexity of our work than others is $O(n^3LT)$ because the time complexity of naive matrix multiplication is $O(n^3)$, where n is the number of agents, L is the number of layers, and T is the total number of training steps. In terms of memory, the memory complexity of the matrix multiplication is $O(n^2)$. The additional matrix multiplications in (13) are performed in every training step, but the calculations in the current step do not need to use in the next training step. Therefore, the memory complexity of our work than others is $O(n^2L)$.

Actions are inputs of the joint action-value function, and thus the covariance matrix of actions is required to calculate. However, the variance of actions is difficult to calculate on many test-beds that are handled in value-based RL since the actions are set to be categorical variables (e.g. attacking action in a battle game). To be applicable even in these cases, we assume that Q_{jt} is the function of Q_i, $Q_{jt} = f_m(Q_i)$, where f_m is the mixing network. Then, the categorical actions are mapped to the individual utility function Q_i as numerical values. After that, the variance of individual utilities is used as the input covariance:

$$\Sigma_i^0 = \begin{bmatrix} \nu_1 & 0 & \dots & 0 \\ 0 & \nu_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \nu_N \end{bmatrix}, \tag{14}$$

where Σ_i^0 indicates the covariance matrix of inputs for agent i, and ν_i is the variance of Q_i. Note that ν_i is set to be zero when computing Σ_i^0 because we need to calculate only the influence by other agents, excluding the target agent i as defined in (10). Σ_i^0 is formed as a diagonal matrix since it is reasonable to set that inputs are independent at the first layer (input layer). In contrast, Σ_i^l when $l > 0$ is not a diagonal matrix since the linear combinations of inputs at the previous layer are used as inputs of the next layer.

With calculating the covariance matrix, the Jacobian matrix is also calculated at each layer of the mixing network f_m as in 12 and [32]. To sum up, the output variance Σ_i^L is estimated by propagating the covariance of inputs Σ_i^0 through the function f_m using (13). The estimated variance according to this procedure is applied as the influence I, and its calculation for the mixing network with L layers is described in Algorithm 1. After the estimation of influences by other agents, the influences are applied to exploration.

B. APPLICATION TO EXPLORATION METHOD

In exploration methods of existing value-based MARL algorithms, each agent uses independent action-selections over its utility function based on ϵ-greedy as follows:

$$u_i = \begin{cases} \operatorname{argmax} Q_i(\tau_i, u_i) & \text{with prob. } 1-\epsilon \\ \text{a random action} & \text{with prob. } \epsilon \end{cases}. \tag{15}$$

Instead of only using the individual utility, the linear combination of the individual utility and the influence is used in the proposed exploration as follows:

$$\psi_i = Q_i + \beta I_i, \tag{16}$$

where I_i is the influence of agent i, and β is the weight value balancing between the influence and the utility function. As a result, an action is selected randomly with a probability of ϵ, and an action that has the largest value for the linear combination of the influence and the utility function is selected with a probability of $1 - \epsilon$ as follows:

$$u_i = \begin{cases} \operatorname{argmax} \psi_i(\tau_i, u_i) & \text{with prob. } 1-\epsilon \\ \text{a random action} & \text{with prob. } \epsilon \end{cases}. \tag{17}$$

We named the proposed exploration method with the influence as Influence-Seeking Exploration (ISE) which is explained in Algorithm 2. Agents select actions by ISE to interact the environment and store the experience samples in the replay buffer. The data sampled from the replay buffer is used to train the joint action-value function Q_{jt}. Finally, overall procedure including training value function and ISE, which is explained in Section IV, is summarized in Fig. 2.

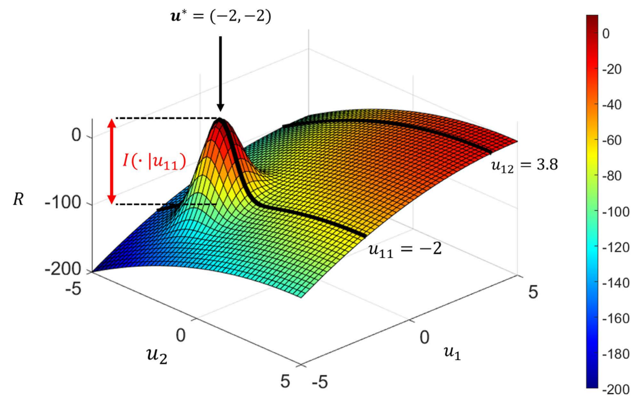


FIGURE 3. **An example for explaining the influence. *R* means return values and u_{ij} indicates the *j*-th action of agent *i*. The influence for action u_{11} of agent 1 is defined as the variance of expected return with different action set of agent 2, which is marked as the red arrow.**

C. EXPECTED EFFECT OF ISE IN MARL

To describe why ISE works in MARL, additional explanations with a graphical example are presented. The advantage of using the proposed influence is related to *relative overgeneralization* (RO). RO is that a sub-optimal Nash Equilibrium is preferred over an optimal Nash Equilibrium in the joint action space for MARL. As exemplified in Fig. 3, $u_{12}(= 3.8)$ could be preferred in terms of u_1 considering arbitrary actions of the other agent, u_2, even though the joint optimal action $\mathbf{u}^*$, $(-2, -2)$, includes u_{11}. This is the simple example to explain RO in the system with only two agents. To explore the optimal action with overcoming RO, we need the measure to quantify the variation of expected returns. As described in IV-A, the variance of joint action-values is defined as the influence. The influence could be used as an indicator to make u_{11} more favorable, which is marked as the red arrow in Fig. 3. As the variation of expected return increases, it becomes more difficult to find the optimal policy because RO is more likely to occur. Accordingly, the influence should be considered to help solve RO, especially in an environment with the high variation of expected returns.

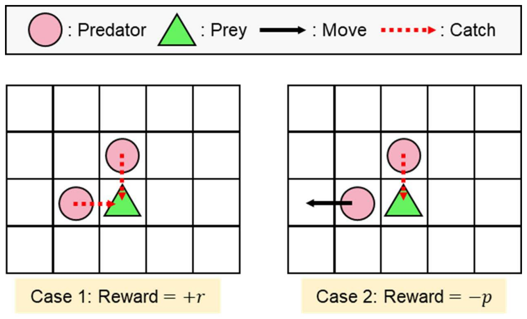


FIGURE 4. **Reward structure in modified predator-prey (MPP) task. The red circle and green triangle indicate a predator and a prey, respectively. The black solid arrow and red dotted arrow mean moving and catching action, respectively. If two predators catch a prey simultaneously as in Case 1, the reward is *r* (*r* is positive). If a single predator only catches a prey as in Case 2, the penalty is $-p$ (*p* is positive).**

TABLE 5. **Detail information of agents in SMAC.**

Name	Shield points	Hit points
Stalkers (s)	80	80
Zealots (z)	50	100
Colossi (c)	150	200
Zerglings (zg)	N/A	35
Hydralisks (h)	N/A	90
Banelings	N/A	30

V. EXPERIMENT

In this section, we present our experimental setup and results to verity the validity of the proposed method. All experiments were performed using our computational power system (GPU = RTX 2080 Ti, CPU = Intel(R) Xeon(R) Gold 5115 @ 2.40GHz, RAM = 128GB). All simulations and algorithms in this work were implemented in the PyMARL framework[1]with Ubuntu version 16.04 [11].

A. EXPERIMENTAL SETUP

1) EXPERIMENTAL ENVIRONMENT

Experiments were performed in two popular test-beds: StarCraft Multi-Agent Challenge (SMAC) and partially-observable modified predator-prey (MPP) tasks. SMAC was created by focusing on battles of small units in StarCraft II of Blizzard. Allied units trained using the developed algorithm battles with enemy units operated by the built-in AI, and the winning means that allied units defeat all enemy units. We tested six different scenarios with difficulty levels of easy, hard, and super-hard defined in [11]. The detailed information about agents and scenarios in SMAC is presented in Table 4 and Table 5, respectively. For SMAC tasks, 32 test episodes were implemented, and the winning percentage in test episodes, i.e., test win rate., was measured. The average test win rate of five independent runs is reported as the performance. Additional experimental details for SMAC were followed on the training setup in [6], [11].

[1]The code of PyMARL is available at https://github.com/oxwhirl/pymarl

TABLE 4. Description of test scenarios in SMAC [11].

Name	Ally units	Enemy units	Type
2s3z	2 Stalkers and 3 Zealots	2 Stalkers and 3 Zealots	Heterogeneous and symmetric
3s5z	3 Stalkers and 5 Zealots	3 Stalkers and 5 Zealots	Heterogeneous and symmetric
2c_vs_64zg	2 Colossi	64 Zerglings	Micro-trick: positioning
3s_vs_5z	3 Stalkers	5 Zealots	Micro-trick: kiting
Bane_vs_Bane	20 Zerglings and 4 Banelings	20 Zerglings and 4 Banelings	Micro-trick: positioning
6h_vs_8z	6 Hydralisks	8 Zealots	Micro-trick: focus-fire

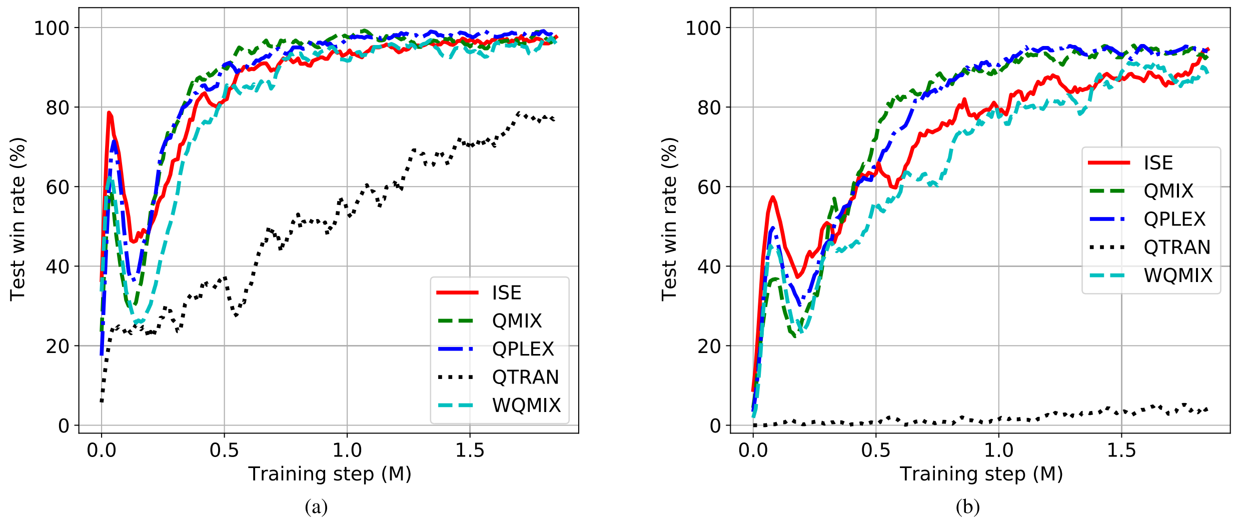


FIGURE 5. Average test win rates in easy scenarios: a) *2s3z*, b) *3s5z*. Learning curves of all baselines (QMIX, QPLEX, QTRAN, and WQMIX) and the proposed method (ISE) are shown and compared. In these easy scenarios, ISE does not significantly affect the results. This is because the optimal policy can be found quickly with a simple exploration scheme based on ϵ-greedy policy in these scenarios.

TABLE 6. Common parameter settings for all methods.

Type of parameter	Value
Discount factor	0.99
Learning rate	0.0005
Target update period	200
Minibatch size	32
Replay buffer size	5000
Test interval	10000
Initial epsilon	1
Final epsilon	0.05
Final exploration step	50000

TABLE 7. Hyper-parameters applied differently depending on the task.

Method	Hyper-parameter	Task	
		SMAC	MPP
OW-QMIX, ISE	α	0.5	0.1
ISE	β	1.5	3.0

The MPP task [5] was created based on the predator-prey task in which predators get rewards when catching preys. Predators are agents to be trained, and the state of predators is the position in grids. Predator can select six actions: move left, move right, move up, move down, catch, and stay. The difference from the original predator-prey is that if more than two predators catch a prey, then they get a reward; however, if only one predator catches a prey, it gets a penalty as shown in Fig. 4. For the experiment in the MPP task, the reward r was set to 10, and the punishment p when only a single predator catch a prey was set as four cases: -2, -4, -6, and -8. For testing, 16 episodes are repeated, and the average return of five independent runs was measured. Detailed experimental settings for MPP tasks were based on the training setup in [33].

2) EXPERIMENTAL SETTING FOR ALGORITHMS

We compared our approach with state-of-the-art MARL algorithms: QMIX [4], QTRAN [5], WQMIX [6], and QPLEX [21]. In WQMIX, optimistically-weighting is used for overall good performance. The proposed method is implemented based on WQMIX which has agent networks, a mixing network of Q_{jt}, and a mixing network of $\hat{Q}^*$. All agent networks have a DRQN structure with a recurrent layer of a GRU [34]. The dimension of the hidden state in this GRU is 64, and a fully-connected (FC) layer exists before and after this GRU layer. The mixing network in Q_{jt} is formed by a single hidden layer with 32 nodes using ELU non-linear activation function. The weights of the mixing network in Q_{jt} are produced by the hypernetworks. The hypernetwork is composed of a single hidden layer of 32 units with

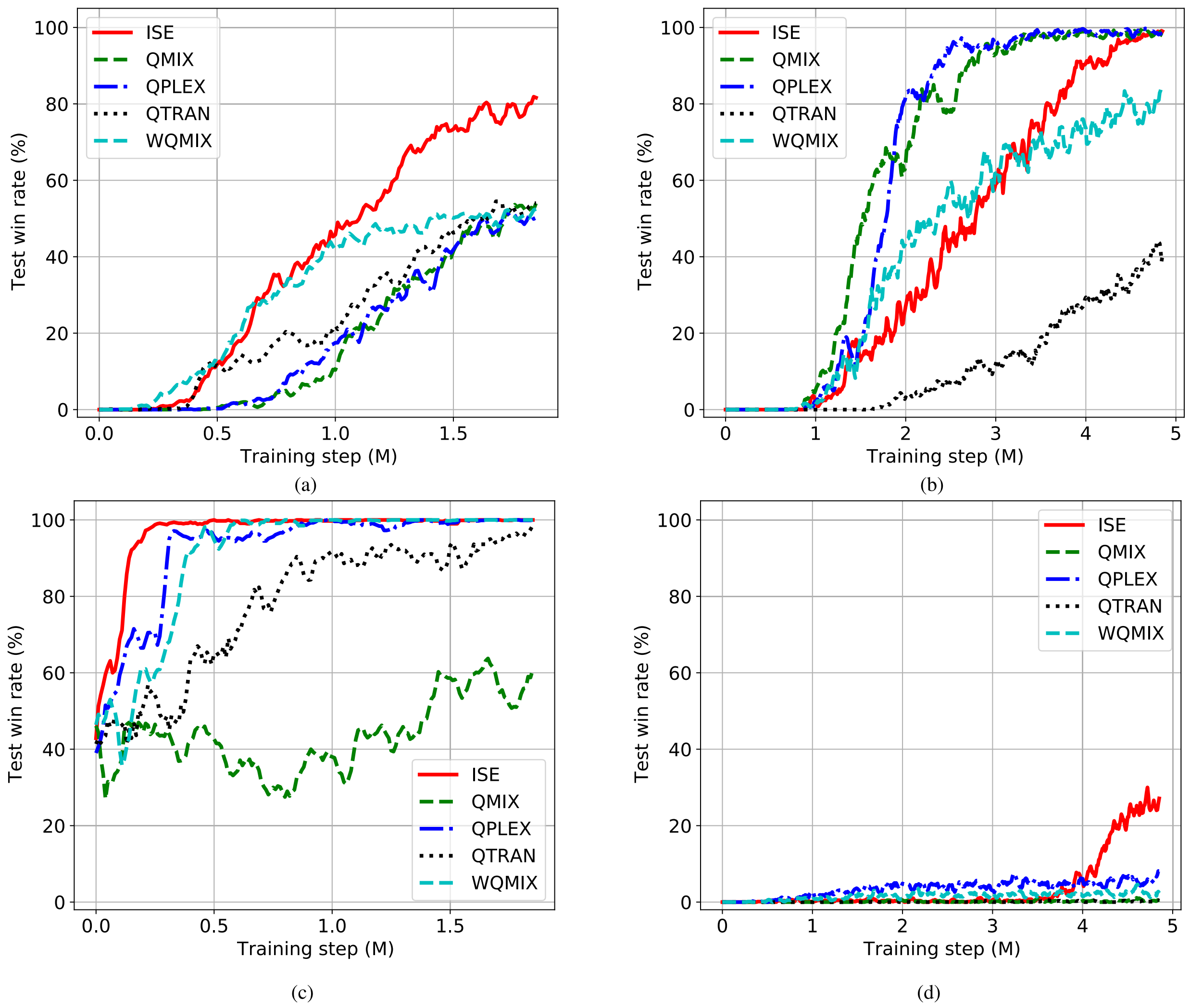


FIGURE 6. Average test win rates in hard and super-hard scenarios: a) *2c_vs_64zg*, b) *3s_vs_5z*, c) *Bane_vs_Bane*, d) *6h_vs_8z*. Learning curves of all baselines (QMIX, QPLEX, QTRAN, and WQMIX) and the proposed method (ISE) are shown and compared. In more difficult scenarios, ISE improves the performance significantly because of efficient searching in exploration.

a ReLU non-linearity. The mixing network in $\hat{Q}^*$ has a feed-forward structure with three hidden layers of 256 dimensions and ReLU activation functions. Each run takes approximately 24 hours with slight differences depending on the scenarios.

Although our approach requires additional matrix multiplications to calculate influences, our computation time is not significantly different from existing MARL methods in the test environment such as SMAC and MPP tasks. This is because the number of agents in the test environment is at most 20 to 30, and the calculation time of the added process in the proposed method is not relatively large compared to the calculation time of the main algorithm part. For these reasons, this paper focuses more on performance improvement rather than computational complexity. Experimental settings commonly applied to all methods are shown in Table 6, and hyper-parameters changing with tasks are described in Table 7.

B. EXPERIMENTAL RESULTS

1) STARCRAFT MULTI-AGENT CHALLENGE

Fig. 5 and 6 show the learning curve of the proposed method and baselines, and Table 8 describes the final average performance which is the maximum average across the testing intervals within the last 250k of training. The proposed method, ISE, improves the performance in the difficult scenarios shown in Fig. 6. In *3s_vs_5z* scenario, the convergence speed of QMIX is fast since the searching space is not quite huge due to the small number of agents and actions; however, the final win rate of ISE is higher than that of QMIX. Although the original purpose of this study is to propose a solution to difficult scenarios that cannot be solved by the existing methods, the final win rate of ISE is similar or slightly greater than those of baselines even in easy scenarios shown in Fig. 5. From this, we can claim that the proposed method can be used in general regardless of the difficulty of scenarios.

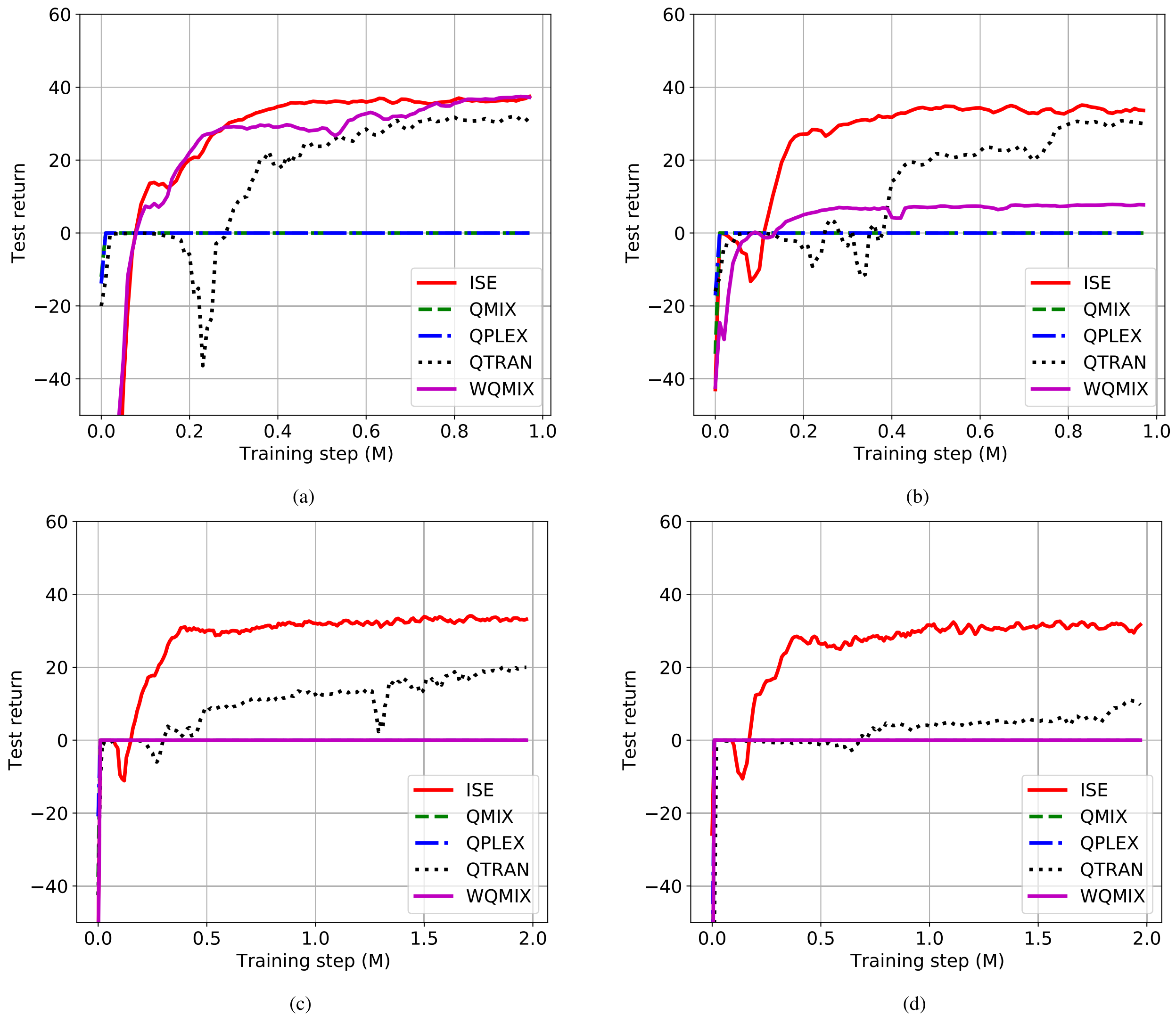


FIGURE 7. **Average test return in modified predator prey: a) $p = -2$, b) $p = -4$, c) $p = -6$, d) $p = -8$. p indicates the punishment of capturing by a single predator. Learning curves of all baselines (QMIX, QPLEX, QTRAN, and WQMIX) and the proposed method (ISE) are shown and compared. It is difficult to find an optimal policy because agents avoid capturing preys because of large p. Even in these scenarios, ISE succeeds in finding the optimal policy.**

Despite these performance improvements, the proposed method does not increase the training time significantly in SMAC as shown in Table 9. This is because the number of agents in the environment is at most 20 to 30, and the calculation time of the added process in the proposed method is not relatively long compared to the calculation time of the main algorithm part.

2) MODIFIED PREDATOR-PREY

Fig. 7 shows the learning curve of the proposed method and baselines, and Table 10 describes the final average return. In the MPP tasks, as the punishment increases, it becomes more prone to RO and eventually more difficult to discover the optimal policy. With the punishment of -2, the punishment is relatively small compared to the reward value, the weighting in WQMIX is sufficient to converge the optimal policy. However, when the punishment is -4 or less, WQMIX cannot search for the desired behavior, and the effect of ISE is clearly visible in these cases. Even though the average return of QTRAN increased with the training

TABLE 8. **Final average test win rate (%) of six different scenarios in SMAC. The bold text indicates the best performance in each scenario.**

Scenario	ISE	QMIX	QPLEX	QTRAN	WQMIX
2s3z	97.6	97.3	**98.0**	78.0	96.0
3s5z	**94.5**	92.9	94.0	4.13	88.4
2c_vs_64zg	**81.6**	54.0	50.3	54.0	52.5
3s_vs_5z	**99.0**	97.9	98.4	40.1	83.9
Bane_vs_Bane	**100**	60.9	**100**	98.6	**100**
6h_vs_8z	**27.1**	0.63	8.13	0.38	2.38
Average	**83.3**	67.3	74.8	45.9	70.5

TABLE 9. Computation time (h) required by baselines and the proposed influence-seeking exploration for different scenarios in SMAC.

Scenario	ISE	QMIX	QPLEX	QTRAN	WQMIX
2s3z	9.02	6.82	7.73	6.75	8.28
3s5z	11.3	8.33	9.23	8.17	9.85
2c_vs_64zg	15.4	13.1	11.7	18.2	12.7
3s_vs_5z	24.1	14.7	17.3	29.5	19.3
Bane_vs_Bane	29.1	15.3	22.7	19.7	21.6
6h_vs_8z	19.3	19.7	25.1	17.3	19.6
Average	18.0	13.0	15.7	16.6	15.2

TABLE 10. Final average test return of four different cases in MPP. The bold text indicates the best performance in each scenario.

Value of punishment	ISE	QMIX	QPLEX	QTRAN	WQMIX
P = -2	**37.4**	0.0	0.0	30.3	37.2
P = -4	**33.6**	0.0	0.0	30.0	7.73
P = -6	**33.1**	0.0	0.0	20.0	0.0
P = -8	**31.7**	0.0	0.0	9.71	0.0
Average	**34.0**	0.0	0.0	22.5	11.2

TABLE 11. Computation time (h) required by baselines and the proposed influence-seeking exploration for different scenarios in MPP.

Value of punishment	ISE	QMIX	QPLEX	QTRAN	WQMIX
P = -2	4.38	3.32	3.70	3.00	4.18
P = -4	4.78	3.85	3.67	3.60	4.42
P = -6	4.88	3.70	3.77	3.67	4.32
P = -8	4.98	3.87	3.70	3.63	4.20
Average	4.76	3.69	3.71	3.48	4.28

step, the convergence speed is considerably slow due to its constraints which apply decentralization based on relaxed L2 penalties. From these results, we verified that ISE helps to find the optimal policy and accelerates convergence speed even in environments with a high chance to occur RO as explained in IV-C.

To show the computational cost of the proposed method, we compare the computation time of baselines and the proposed method in MPP tasks as shown in Table 11. It was confirmed that there was no significant difference in calculation time for the same reason as in the SMAC environment.

Finally, to consider an extension to other multi-agent scenarios, we analyze how many agents can be handled considering computation time and time complexity for training. In general, a computational power system requires 1 second to perform 10^8 operations. The parameter of our model is that L is set to be 3 and the maximum of T is set to be 5 million. For each run, the existing algorithms take approximately 16 hours which is estimated based on the results in SMAC. Considering the computation time and time complexity, the maximum number of agents that can be trained within the limited computation time is calculated as shown in Table 12. The larger the training time, the higher the maximum number of agents it can handle. This estimation can be different from actual calculations; however, this interpretation is meaningful because it gives the scale information of systems that the proposed algorithm can handle.

TABLE 12. The maximum number of agents that can be handled within the limited computation time. We set 16 hours as 100% for computation time, and thus +100% means 32 hours. The maximum number of agents (N_{max}) is calculated as $\left(\frac{t_A \times 3600 \times 10^8}{LT}\right)^{1/3}$.

Allowed computation time (t_A)	+10%	+20%	+50%	+100%
Maximum number of agents (N_{max})	33	42	57	72

VI. CONCLUSION

In this paper, we studied the problem of multi-agent exploration and proposed the influence-seeking exploration method which encourages agents to significantly explore action spaces influenced by others. The influence was defined as the variance of joint action-values with different actions of other agents, and then the approximated variance propagation was employed for the feasible application under the assumption that the joint action-value is the function of individual utilities. For applying the influence as an exploration scheme, the linear combination of the individual utility and the proposed influence is utilized in collecting experience samples.

We verified that the proposed method improves performance by approximately 10% in StarCraft multi-agent challenge and 50% in modified predator-prey tasks. Moreover, our approach can accelerate the convergence of the value-based methods. From the improved results, we verified that the proposed exploration method could help in finding samples to achieve the optimal policy even if RO is easy to occur in multi-agent systems.

The restriction of the proposed method is that it only considers the influence by other agents even though the source of influence exists more such as stochasticity of the environment. In future work, we plan to combine the influences from the two sources: other agents and the environment. For this work, the characteristics of these two influences need to be analyzed, and the way of incorporating them into the exploration method should be updated.

REFERENCES

[1] O. Vinyals *et al.*, "StarCraft II: A new challenge for reinforcement learning," 2017, *arXiv:1708.04782*.

[2] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*.

[3] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 1–17.

[4] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4295–4304.

[5] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5887–5896.

[6] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 1–12.

[7] J. Hu, S. A. Harding, H. Wu, S. Hu, and S. W. Liao, "QR-MIX: Distributional value function factorisation for cooperative multi-agent reinforcement learning," *arXiv:2009.04197*.

[8] H. M. R. U. Rehman, B.-W. On, D. D. Ningombam, S. Yi, and G. S. Choi, "QSOD: Hybrid policy gradient for deep multi-agent reinforcement learning," *IEEE Access*, vol. 9, pp. 129728–129741, 2021.

[9] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multiagent policy gradients," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.

[10] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[11] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C. M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The StarCraft multi-agent challenge," in *Proc. 18th Int. Conf. Auton. Agents Multiagent Syst.*, 2019, pp. 2186–2188.

[12] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Auton. Robots*, vol. 8, no. 3, pp. 345–383, Jun. 2000.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[16] D. Silver, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 387–395.

[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," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[18] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 4213–4220.

[19] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.

[20] H. Ryu, H. Shin, and J. Park, "Multi-agent actor-critic with hierarchical graph attention network," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2020, pp. 7236–7243.

[21] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–11.

[22] W.-F. Sun, C.-K. Lee, and C.-Y. Lee, "DFAC framework: Factorizing the value function via quantile mixture for multi-agent distributional Q-learning," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 9945–9954.

[23] W. Qiu, X. Wang, R. Yu, X. He, R. Wang, B. An, S. Obraztsova, and Z. Rabinovich, "RMIX: Learning risk-sensitive policies for cooperative reinforcement learning agents," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 1–14.

[24] T. Wang, T. Gupta, B. Peng, A. Mahajan, S. Whiteson, and C. Zhang, "RODE: Learning roles to decompose multi-agent tasks," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–14.

[25] E. Wei, D. Wicke, D. Freelan, and S. Luke, "Multiagent soft Q-learning," 2018, *arXiv:1804.09817*.

[26] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "MAVEN: Multiagent variational exploration," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 7611–7622.

[27] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 2778–2787.

[28] M. Szemenyei and P. Reizinger, "Attention-based curiosity in multi-agent reinforcement learning environments," in *Proc. Int. Conf. Control, Artif. Intell., Robot. Optim. (ICCAIRO)*, May 2019, pp. 176–181.

[29] T. Wang, J. Wang, Y. Wu, and C. Zhang, "Influence based multi-agent exploration," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–13.

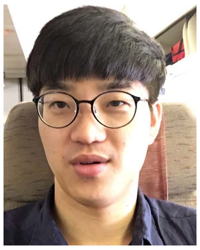
[30] F. A. Oliehoek and C. Amato, "The decentralized POMDP framework," in *A Concise Introduction to Decentralized POMDPs*, 1st ed. Cham, Switzerland: Springer, 2016, pp. 11–32.

[31] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPS," in *Proc. AAAI Fall Symp. Sequential Decis. Making Intell. Agents*, 2015, pp. 1–7.

[32] J. Postels, F. Ferroni, H. Coskun, N. Navab, and F. Tombari, "Sampling-free epistemic uncertainty estimation using approximated variance propagation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2931–2940.

[33] W. Böhmer, V. Kurin, and S. Whiteson, "Deep coordination graphs," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 980–991.

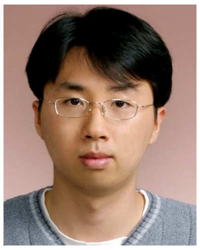
[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Workshop Deep Learn.*, 2014, pp. 1–9.



BYUNGHYUN YOO received the Ph.D. degree in mechanical engineering from the Korea Advanced Institute of Science and Technology (KAIST), in 2019. Since 2019, he has been with the Electronics and Telecommunications Research Institute (ETRI), South Korea, where he is currently a Senior Researcher with the Integrated Intelligence Research Section and the Intelligence Information Research Division. His research interests include the areas of multi-agent reinforcement learning (MARL) and artificial general intelligence (AGI).

DEVARANI DEVI NINGOMBAM received the M.Tech. degree in communication engineering from the National Institute of Technology Karnataka (NITK), Surathkal, India, in 2015, and the Ph.D. degree in computer engineering from Chosun University, South Korea, in 2019. She was a Postdoctoral Researcher at the Future Technology Research Laboratory and Planning Division, and the Artificial Intelligence Research Laboratory and Smart Data Research Section, Electronics and Telecommunications Research Institute (ETRI), South Korea, in 2019 and 2021, respectively. She is currently an Assistant Professor with the Department of Computer Science and Engineering, GITAM University, Visakhapatnam, India. Her current research interests include artificial intelligence, multi-agent reinforcement learning (MARL), the Internet of Things (IoT), device-to-device (D2D) communications, and wireless networks.

SUNGWON YI received the M.S. and Ph.D. degrees in computer science and engineering from The Pennsylvania State University, in 2004 and 2005, respectively. Before joining The Pennsylvania State University, he worked at LG-CNS, as a System Engineer. Since 2005, he has been with the ETRI, South Korea, where he is currently a Researcher with the Future Technology Research Laboratory and the Planning Division. His research interests include the areas of network security, storage systems, mobile computing, and machine learning. He has been served on the Technical Program Committee for the IEEE GLOBECOM and ICC, since 2005.



HYUN WOO KIM received the B.S. and M.S. degrees in electrical engineering from Seoul National University (SNU), Seoul, South Korea, in 2001 and 2003, respectively. Since 2003, he has been with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, where he is currently a Principal Member of Engineering Staff. His research interests include speech signal processing, meta-learning, and machine learning.

EUISOK CHUNG received the B.S. degree in computer science from Soongsil University, Seoul, Republic of Korea, in 1997, and the M.S. degree in computer science from Yonsei University, Seoul, in 1999. Since 1999, he has been working with the ETRI, Daejeon, Republic of Korea. His current research interests include natural language processing, machine learning, and spoken dialogue systems.

RAN HAN received the B.S., M.S., and Ph.D. degrees in electrical and electronics engineering from Yonsei University, Seoul, South Korea, in 2008, 2010, and 2015, respectively. She was a Senior Engineer at Samsung Electronics, Suwon, South Korea, from 2015 to 2017. In 2017, she joined the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, where she is currently a Senior Researcher of the Artificial Intelligence Group. Her research interests include digital signal processing, pattern recognition, and machine learning.

HWA JEON SONG received the B.S., M.S., and Ph.D. degrees in electronics engineering from Pusan National University, Republic of Korea, in 1993, 1995, and 2005, respectively. From 1995 to 2001, he was a Researcher with Hyundai Motor Company. Since 2010, he has been a Principal Researcher with the Electronics and Telecommunications Research Institute (ETRI). His research interests include speech recognition, multi-modal representation, and artificial general intelligence (AGI).