MASER: Multi-Agent Reinforcement Learning with Subgoals Generated from Experience Replay Buffer

Jeewon Jeon [1] Woojun Kim [1] Whiyoung Jung [1] Youngchul Sung [1]

Abstract

In this paper, we consider cooperative multi-agent reinforcement learning (MARL) with sparse reward. To tackle this problem, we propose a novel method named MASER: MARL with subgoals generated from experience replay buffer. Under the widely-used assumption of centralized training with decentralized execution and consistent Q-value decomposition for MARL, MASER automatically generates proper subgoals for multiple agents from the experience replay buffer by considering both individual Q-value and total Q-value. Then, MASER designs individual intrinsic reward for each agent based on actionable representation relevant to Q-learning so that the agents reach their subgoals while maximizing the joint action value. Numerical results show that MASER significantly outperforms StarCraft II micromanagement benchmark compared to other state-of-the-art MARL algorithms.

1. Introduction

Deep multi-agent reinforcement learning (MARL) is an active research field to handle many real-world problems such as games, traffic control, and connected self-driving cars, which can be modeled as multi-agent systems (Li et al., 2019; Andriotis & Papakonstantinou, 2019). Since the introduction of the simplest approach of independent Q-learning (IQL) (Tan, 1993), many advanced MARL algorithms have been developed recently, e.g., VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2018b), QTRAN (Son et al., 2019), COMA (Foerster et al., 2018b), MADDPG (Lowe et al., 2017), Social Influence (Jaques et al., 2018), Intention Sharing (Kim et al., 2020b). Although these recent algorithms effectively handle the increased dimensions of joint state and action spaces associated with multiple agents and alleviate the burden of inter-agent communication (OroojlooyJadid & Hajinezhad, 2019; Rashid et al., 2018a), they were primarily developed under the assumption of dense reward. However, many multi-agent problems are modeled as sparse-reward environments in which a non-zero reward is not given to agents' actions in every time step but given only when certain conditions are met. For example, in multi-agent robot soccer an actual non-zero reward occurs only when scoring a goal, and individual actions such as passes, dribbles or tackles do not receive non-zero rewards. Thus, non-zero rewards come very seldom in such sparse-reward environments. Reinforcement learning in sparse-reward environments is difficult in general and is more challenging in multi-agent environments since a global reward is shared among multiple agents and the combination of multiple agents' actions jointly affects the state evolution in multi-agent systems.

A common approach to tackle sparse-reward MARL is learning with efficient exploration (Liu et al., 2021; Christianos et al., 2020; Ryu et al., 2020; Mahajan et al., 2019; Kim et al., 2020a), which has been shown effective in some tasks. However, with exploration only, it is difficult to learn which action induces rare non-zero rewards. As an alternative to direct learning of assigning credit to trajectory, which is an essential difficulty in sparse environments, subgoal-based methods are recently emerging as an effective approach to sparse-reward MARL (Tang et al., 2018; Wang et al., 2020a;b), inspired by the methods using subgoals in single-agent systems (Kulkarni et al., 2016; Andrychowicz et al., 2017; Zhang et al., 2020; Chane-Sane et al., 2021). In subgoal-based methods, typically subgoals for maximizing the total episodic reward are determined or designed and learning is performed to reach the subgoals to attain the final goal. The key elements here are how to determine or design subgoals and how to learn to reach the subgoals. One way to determine subgoals is to select from a set of pre-defined subgoals designed based on domain knowledge for a given task (Tang et al., 2018; Kulkarni et al., 2016). However, such an approach is task-specific and not general enough to be applicable to a variety of different tasks. Thus, *automatic subgoal generation* is desirable for subgoal-based approaches to be applicable to various tasks without being dependent on task-

[1] School of Electrical Engineering, KAIST, Daejeon, South Korea. Correspondence to: Youngchul Sung <ycsung@kaist.ac.kr>, Woojun Kim <woojun.kim@kaist.ac.kr>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

specific domain knowledge. Automatic subgoal generation has been considered in single-agent systems (Andrychowicz et al., 2017; Zhang et al., 2020; Chane-Sane et al., 2021). In multi-agent systems, however, automatic subgoal generation has not been investigated much yet. In addition to subgoal generation, subgoal-based methods require a way to reach the subgoals, and reaching subgoals is typically attained by employing intrinsic rewards. In single-agent systems, reaching the subgoals through intrinsic rewards is relatively easy because a single intrinsic reward is designed in single-agent systems (Andrychowicz et al., 2017). In multi-agent systems, on the other hand, designing intrinsic rewards is challenging because we need to design multiple intrinsic rewards for multiple agents considering different contributions of different agents to achieve the final goal.

In order to deal with these difficulties in subgoal-based MARL, we propose a novel method named MASER: Multi-Agent reinforcement learning with Subgoals generated from Experience Replay buffer. Under the widely-used assumption of centralized training with decentralized execution (CTDE) (OroojlooyJadid & Hajinezhad, 2019; Rashid et al., 2018a) and consistent Q-value decomposition for MARL (Rashid et al., 2018b), MASER automatically generates proper subgoals for multiple agents from the experience replay buffer by compromising the individual Q-values and the total Q-value. MASER designs local and global intrinsic rewards based on actionable representation (Ghosh et al., 2018) relevant to Q-learning so that the agents can reach their subgoals while maximizing the joint action value. Numerical results show that MASER significantly outperforms existing algorithms in MARL with sparse reward.

2. Related Works

MARL with Value Decomposition: Value decomposition is one of the key techniques for MARL with finite action space, enabling CTDE, e.g., VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2018b), QTRAN (Son et al., 2019). In QMIX, the joint action-value function Q^{tot} is decomposed into local Q-value functions associated with individual agents and then the overall Q-value is obtained through a mixing network, where some constraint is applied to guarantee consistency between the local Q-values and the overall Q-value. In this paper, we use a mixing network similar to that introduced in QMIX.

Subgoal Assignment: There exist previous works on subgoal-based MARL with sparse reward. HDMARL (Tang et al., 2018) proposed a hierarchical deep MARL scheme to solve sparse-reward tasks with temporal abstraction. HDMARL selects one from a set of predefined subgoals designed based on domain knowledge and hence such subgoal design is not easily applicable to different tasks. HSD (Yang et al., 2019) trains cooperative decentralized policies at a high level to select skills, and the agents take actions conditioned by chosen skills from high-level.

In the single-agent case, Zhang et al. (2020) generated subgoal by restricting the goal space based on adjacency constraint and Chane-Sane et al. (2021) defined a subgoal set composed of the midpoints between the current state and the goal state and selected a subgoal from the subgoal set. Andrychowicz et al. (2017) proposed using replay buffer to determine subgoals. However, this method simply chooses a state randomly in a failed trajectory or the final state of a failed trajectory as the subgoal. On the other hand, MASER selects multiple effective subgoals for multiple agents from experience episodes by considering their local and global Q-values so that the selected subgoals provide more credit to the final goal achievement.

ROMA (Wang et al., 2020a) and RODE (Wang et al., 2020b) give roles to agents to decompose the overall task, where role is a bit different concept from subgoal. In particular, in ROMA, the role is learned by the agents and the agents with similar roles perform similar sub-tasks to increase the overall task efficiency. In contrast to the roles in ROMA, the subgoals in our approach are determined from the entire trajectory and guide the multiple agents to their own goals. In addition, MASER takes advantage of using centralized training for determining subgoals, but ROMA determines the roles based on local observations of the agents.

MARL with Individual Intrinsic Rewards: Individual intrinsic reward generation for multiple agents in MARL has been considered. By assigning individual reward for each agent, agents are stimulated differently. LIIR (Du et al., 2019) directly learns individual intrinsic reward for each agent to update a proxy critic to obtain each agent's policy, while intrinsic reward learning is updated to maximize extrinsic reward sum. Similarly to LIIR, GIIR (Wu et al., 2021) also learns individual intrinsic reward for each agent for multi-agent Q-learning. Unlike this direct intrinsic reward learning, our method generates individual intrinsic reward based on generated subgoals to attain the final goal.

Representation Learning in RL: Representation learning in RL deals with the transformation of state or observation (i.e., partial state) into a form that captures information relevant to control (Huang et al., 2019; Wu et al., 2018). Actionable representation for control (ARC) provides an effective way to capture the elements in the observation that are necessary for decision making (Ghosh et al., 2018). For this, Ghosh et al. (2018) defined the actionable distance between two goal states s_1 and s_2 via an intermediate state s as the symmetrized KL divergence between two policies conditioned on two states: $\pi(a|s, s_1)$ and $\pi(a|s, s_2)$. This basically quantifies the difference in actions to reach two states s_1 and s_2 from s. When the required actions are totally different, the actionable distance is large. When the required actions are similar, on the other hand, the actionable distance is small. Then, ARC is learned based on the actionable distance. We also use actionable representation to obtain the intrinsic reward from partial states chosen as subgoals. In our case, to be compatible with multi-agent Q-learning, we define our actionable distance using the local Q-function. This new actionable distance has a very simple form and is well suited to Q-learning.

3. Background

Decentralized Partially Observable Markov Decision Process: A cooperative multi-agent task with N agents can be described as a decentralized partially observable Markov decision process (Dec-POMDP) (Oliehoek, 2012). A Dec-POMDP consists of a tuple $G = \langle S, U, P, Z, O, r, \gamma, N \rangle$, where S is the state space, U is the action space, P is the transition probability, Z is the observation space, O is the observation function, r is the reward function, γ is the discount factor and N is the number of agents. The action space is assumed to be finite in this paper. At each time step, each agent $i \in I := \{1, \dots, N\}$ chooses an action $u^i \in U$ and the joint action $\mathbf{u} := (u^1, \cdots, u^N) \in U^N =: \mathbf{U}$ is formed based on the actions of all agents. Then, the joint action causes a transition in the environment state according to the state transition probability $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$, where s is the global state of the environment and s' is the next global state. Due to partial observability, Agent i makes an individual observation $o^i \in Z$ following the observation function $O(s, i) : S \times I \to Z$ at each time step. The action u^i of Agent i is selected according to its policy $\pi^i(u^i|\tau^i)$, where τ^i is the observation-action history of Agent i up to the current time step. The policies of all agents form a joint policy $\pi = (\pi^1, \cdots, \pi^N)$, and the goal is to maximize the expected global return $\mathbb{E}[G_0]$ by optimizing π, where $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the discounted return and r_t is the global reward at time step t according to the reward function r, shared by all agents. The joint action-value function of π is defined as $Q^\pi(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}}[G_t|s_t, \mathbf{u}_t]$.

4. Method

In this section, we present MASER solving MARL with sparse reward effectively based on subgoals. We assume the Centralized Training and Decentralized Execution (CTDE) paradigm, which is widely assumed in MARL (Oroojlooy-Jadid & Hajinezhad, 2019; Rashid et al., 2018a). We also assume a finite action space U and episodic off-policy learning in which each episode is composed of T_e time steps. The overall architecture of MASER is shown in Fig. 1. MASER is based on Q-learning and Q-decomposition similar to QMIX (Rashid et al., 2018b). That is, the overall Q-function network for global return is composed of N local utility networks computing N local Q-values $Q^1, \cdots, Q^N$ (one for each agent) and a mixing network computing the global Q-value denoted as Q^{tot}. At time step t, Agent i's local utility network receives the pair of its local observation o^i_t and action u^i_{t-1} to learn the local Q-value Q^i. Here, the local utility network incorporates a RNN-type structure to deal with partial observability in o^i_t. Then, the N local Q-values $Q^1, \cdots, Q^N$ are fed into the mixing network to generate the total Q-value Q^{tot} by enforcing the consistency constraint $\frac{\partial Q^{tot}}{\partial Q^i} \geq 0, \forall i$. In the execution phase after centralized training, the mixing network is removed and Agent i acts based on the local policy derived from the local action-value function Q^i in a decentralized manner. We implement the mixing network and the local utility networks by using deep neural networks, and θ and θ_i $(i = 1, \cdots, N)$ denote the network parameters of the mixing network and the i-th utility network, respectively.

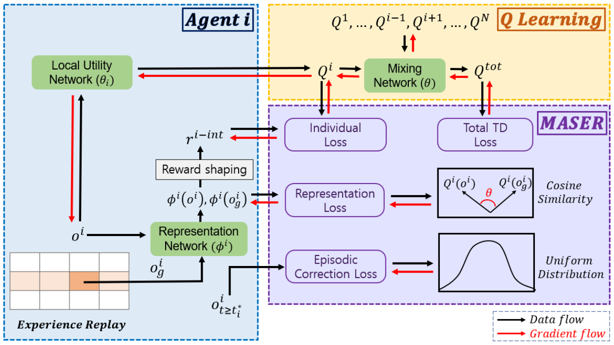


Figure 1. Overflow of MASER: (Blue box) The blue box shows how MASER assigns subgoals to the agents. MASER finds out a subgoal for each agent from the replay buffer and computes individual intrinsic rewards from subgoals by using actionable representation learning. (Purple box) The purple box shows the learning loss components of MASER. From the subgoals from the blue box, it computes total TD loss, individual TD loss, episodic correction loss, and representation learning loss. (Yellow box) The yellow box shows the Q-learning based on the mixing network.

On the basis of the aforementioned Q-decomposition, MASER performs blockwise operation, where the block length is equal to the episode length T_e, and adopts the following three key ingredients to efficiently learn in environments with sparse reward:

i) Generating and assigning subgoals: MASER finds subgoals for agents from the experience replay buffer. This eliminates the necessity of predesigning good subgoals based on domain knowledge.

ii) Giving individual rewards: MASER designs individual rewards for local agents to reach their subgoals while maximizing the joint return.

iii) Actionable distance relevant to Q-learning: To determine the intrinsic reward based on the Euclidean distance

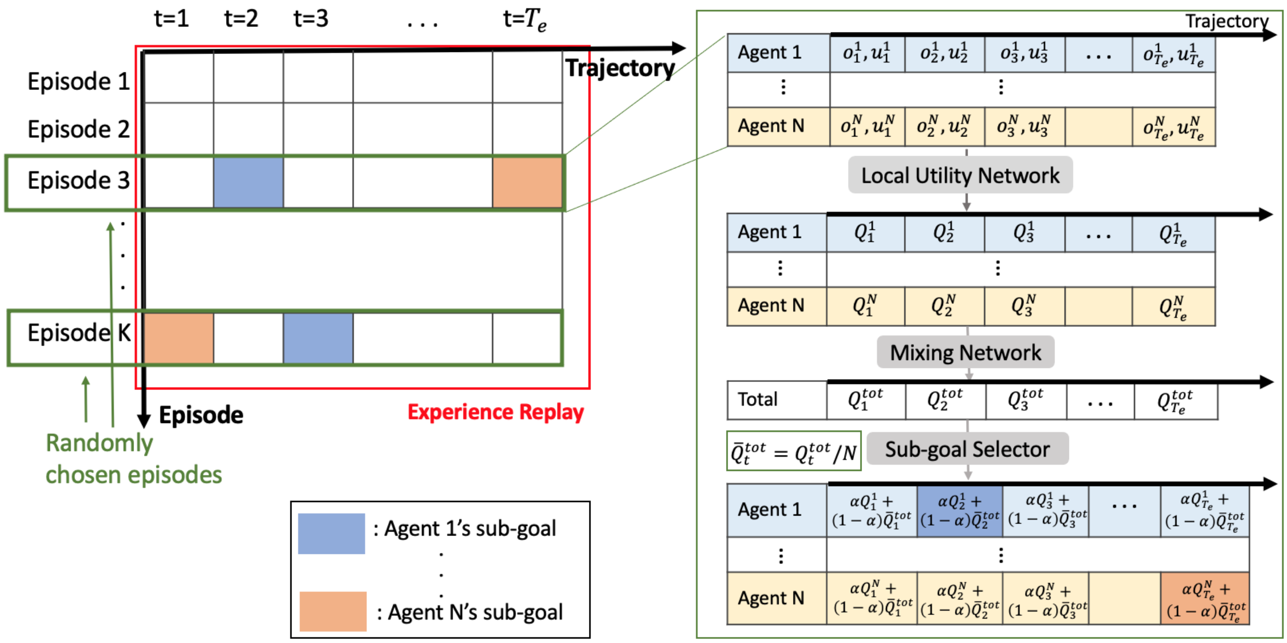


Figure 2. Diagram of subgoal determination from replay buffer. (Left) MASER chooses M episodes randomly from the replay buffer and each episode of length T_e has sequential sample information of action and observation of all agents and extrinsic reward from $t = 1$ to $t = T_e$. (Right) The right side shows how to select a subgoal for each agent based on the current Q-function estimate.

in the transformed domain, MASER uses representational transform based on an actionable distance relevant to Q-learning derived from Amari 0-divergence (Amari, 2016).

4.1. Generating Subgoals

As aforementioned, MASER performs blockwise operation, where the block length is equal to the episode length T_e.

Suppose that the current block index is b, and the current mixing network and local utility network parameters are respectively given by $\theta[b]$ and $\theta_i[b]$ $(i = 1, \cdots, N)$ with the corresponding Q-functions $Q^{tot}_{\theta[b]}$ and $Q^i_{\theta_i[b]}$ $(i = 1, \cdots, N)$ in the beginning of block b. Suppose also that the replay buffer stores the episodes up to now, as seen in Fig. 2.

In order to generate subgoals for block b, MASER randomly selects M episodes from the replay buffer. Each selected episode consists of $\{(o^i_1, u^i_1, \cdots, o^i_{T_e}, u^i_{T_e})^N_{i=1}, (r^{ex}_1, \cdots, r^{ex}_{T_e})\}$, where r^{ex}_t is the extrinsic global reward at time step t. Basically, our choice of the subgoal for each agent is the observation (i.e., partial state) that yields maximum value (i.e., future sum reward) from the selected episode. In order to compute the value of each observation in each of the selected episodes, we exploit the current learned local and global Q-functions $Q^i_{\theta_i[b]}$ and $Q^{tot}_{\theta[b]}$.

For explanation, let us consider the m-th selected episode. The subgoal for Agent i for block b from the m-th selected episode is determined as

$$t^i_\star = \underset{t:1\le t\le T_e}{\arg\max} \Big[\alpha Q^i_{\theta_i[b]}\big(o^i_t, \underset{u^i}{\arg\max}\, Q^i_{\theta_i[b]}(o^i_t, u^i)\big) + (1-\alpha)\bar{Q}^{tot}_{\theta[b]}(\mathbf{o}_t, \mathbf{u}_t)\Big]$$

$$o^i_g = o^i_{t^i_\star}, \qquad (1)$$

where $\alpha \in [0, 1]$, $\bar{Q}^{tot}_{\theta[b]}(\mathbf{o}_t, \mathbf{u}_t) = \frac{1}{N} Q^{tot}_{\theta[b]}(\mathbf{o}_t, \mathbf{u}_t)$, $\mathbf{o}_t = (o^1_t, \cdots, o^N_t)$, $\mathbf{u}_t = (u^1_t, \cdots, u^N_t)$, and $\mathbf{o}_t$ and $\mathbf{u}_t$ are from the m-th selected episode (for notational simplicity, we omit the episode index m). Note that the subgoal for Agent i for block b is not the observation that greedily maximizes the local Q-value Q^i based on the current Q-function estimate but is the observation that maximizes the sum of the local Q-value Q^i and the global Q-value Q^{tot} based on the current Q-function estimate. The weighting between the two is determined by α. When $\alpha = 0$, we have $t^1_\star = \cdots = t^N_\star$ and only the total Q-value is considered with neglection of individual Q-value variation. When $\alpha = 1$, on the other hand, the subgoal for each agent is the greedy local partial state of each agent, and $t^i_\star \neq t^j_\star$ for $i \neq j$ in general in this case. When $0 < \alpha < 1$, both local and global values are considered to determine each agent's subgoal. An ablation study on this aspect is provided in Section 5. Fig. 2 illustrates the subgoal generation process, where $Q^i_{\theta_i[b]}(o^i_t, u^i_t)$

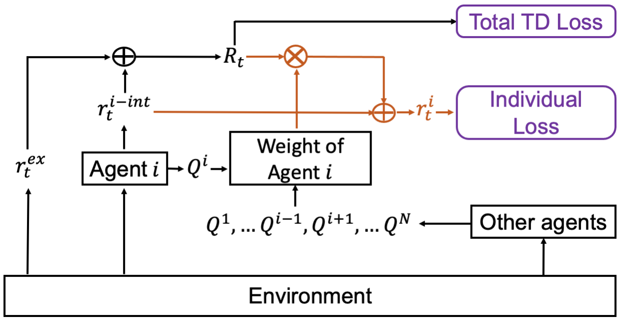


Figure 3. Overall reward design diagram

and $Q^{tot}_{\theta[b]}(\mathbf{o}_t, \mathbf{u}_t)$ are denoted simply as Q^i_t and Q^{tot}_t, respectively. In Fig. 2, o^1_2 is selected as agent 1's subgoal and is $o^N_{T_e}$ is selected as agent N's subgoal for instance.

4.2. Overall Reward Design

With the determined individual subgoals for all agents for block b from Section 4.1, we now design individual and global rewards to reach subgoals while maximizing the joint action value. We define the individual intrinsic reward for Agent i at time step t as the negative Euclidean distance between the current partial state o^i_t at time step t and the subgoal partial state o^i_g for current block b after representational transform, i.e., (Ghosh et al., 2018)

$$r^{i-int}_t = -||\phi^i(o^i_t) - \phi^i(o^i_g)||_2, \quad t = 1, \cdots, T_e, \qquad (2)$$

where $\phi^i(\cdot)$ is an actionable representational transform, which will be described in Section 4.3 in detail. By defining intrinsic reward as eq. (2) and adding it to the extrinsic reward, each agent tries to reach its subgoal while maximizing the extrinsic return. Here, we reshape the reward to capture individual contributions to the overall reward. For this, we first define a proxy reward as the sum of the extrinsic reward from the environment and the intrinsic rewards:

$$R_t = r^{ex}_t + \lambda \frac{1}{N} \sum_{i=1}^{N} r^{i-int}_t \quad \text{for some } \lambda > 0. \qquad (3)$$

This proxy reward R_t is used to update the mixing network parameter θ. In the proxy reward R_t for updating the mixing network parameter θ, the intrinsic rewards are added so as to make the mixing parameter update nontrivial at each time step since the extrinsic reward is sparse. Then, the individual reward r^i_t for Agent i used for updating Agent i's utility network parameter θ_i is defined as

$$r^i_t = \underbrace{\frac{\exp(\max_{u^i} Q^i_{\theta_i[b]}(o^i_t, u^i))}{\sum_{j=1}^{N} \exp(\max_{u^j} Q^j_{\theta_j[b]}(o^j_t, u^j))}}_{=:\ \text{softmax}\left(\max_{u^i} Q^i_{\theta_i[b]}(o^i_t, u^i)\right)} \cdot R_t + \lambda r^{i-int}_t, \qquad (4)$$

where $\text{softmax}(\max_{u^i} Q^i_{\theta_i[b]}(o^i_t, u^i))$ estimates the contribution of Agent i to the overall reward based on the current Q-function estimate. Note that the first term in the right-hand side (RHS) of eq. (4) represents the estimated contribution of Agent i to the overall extrinsic reward r^{ex}_t if there was no intrinsic reward term in R_t in eq. (3). The intrinsic term r^{i-int}_t is explicitly added again in r^i_t to incorporate achieving the individual subgoal. In this way, both mixing and local utility network parameters are learned to achieve the subgoals as well as maximize the overall extrinsic reward. The overall reward design is described in Fig. 3. This reward design process is done for each of the M selected episodes for current block b.

4.3. Q-function-based Representation Learning

As seen in eq. (2), we define the individual intrinsic reward r^{i-int}_t for Agent i at time step t as the negative Euclidean distance between the current partial state o^i_t at time step t and the subgoal partial state o^i_g after actionable representation transform ϕ^i. An actionable representation can be learned through an actionable distance, which is defined as the difference in actions to reach two different goal states (Ghosh et al., 2018). For our Q-learning and Q-decomposition based MARL, we exploit the Q-functions to design an actionable distance between two subgoal states. That is, we consider a soft local policy $\pi^i(\cdot|o^i_t)$ by raising the vector $Q^i_{\theta_i[b]}(o^i_t, \cdot)$ to exponent $e^{\beta(\cdot)}$ and normalizing it, and use the Amari 0-divergence between two distributions p_1 and p_2, given by (Amari, 2016)

$$D^{(0)}(p_1, p_2) = 4\left(1 - \sum_u \sqrt{p_1(u)}\sqrt{p_2(u)}\right), \qquad (5)$$

which is basically the square of Hellinger distance between the two distributions. The Amari 0-divergence is attractive since it induces a symmetric self-dual Euclidean geometry on the space of distributions (a divergence induces a Riemannian metric, dual connections, and corresponding geometry (Amari, 2016)). Furthermore, the nontrivial second term of the RHS of eq. (5) has the nice form of inner product. But, this approach requires the determination of an additional temperature parameter β. So, instead of raising $Q^i_{\theta_i[b]}(o^i_t, \cdot)$ to exponent, normalizing it and taking square root, we directly normalize $Q^i_{\theta_i[b]}(o^i_t, \cdot)$ and put this into inner product. Thus, we define our actionable distance between two subgoal states o^i_t and o^i_g for Q-learning simply as

$$D_Q(o^i_t, o^i_g) = 1 - \frac{< Q^i_{\theta_i[b]}(o^i_t, \cdot), Q^i_{\theta_i[b]}(o^i_g, \cdot) >}{||Q^i_{\theta_i[b]}(o^i_t, \cdot)|| \times ||Q^i_{\theta_i[b]}(o^i_g, \cdot)||}, \qquad (6)$$

where $Q^i_{\theta_i[b]}(o^i_t, \cdot)$ denotes the vector with the elements of the Q-values for all possible actions for o^i_t with dimension

$|U|$, and $< \cdot, \cdot >$ represents the inner product between two vectors. For distance D_Q, we only need to compute the cosine similarity between two vectors. Then, we define the loss for learning the representation ϕ^i based on the actionable distance D_Q as

$$L_D(\phi^i) = \mathbb{E}_{o_t^i}\Big[||\phi^i(o_t^i) - \phi^i(o_g^i)||_2 - D_Q(o_t^i, o_g^i)\Big]^2, \quad (7)$$

where $|| \cdot ||_2$ denotes L_2-norm. By minimizing the loss $L_D(\phi^i)$, the representational transform ϕ^i is learned so that the post-transform Euclidean distance between two partial states becomes similar to one minus cosine-similarity between the two partial states' Q-value vectors.

4.4. Overall Learning

With the designed proxy reward R_t and individual rewards r_t^i for block b, local Q and total Q-learning is performed sequentially from time step $t = 1$ to $t = T_e$ for block b based on the selected episodes for block b. The loss function for the i-th local utility network θ_i is given by

$$L_i(\theta_i) = \left[r_t^i + \gamma \text{max}_{u^i} Q^i_{\theta_i^-}(o^i_{t+1}, u^i) - Q^i_{\theta_i}(o_t^i, u_t^i)\right]^2,$$

where θ_i^- is the parameter of the i-th local target network (Mnih et al., 2015), and (o_t^i, u_t^i) is the t-th sample of Agent i in the m-th episode selected from the replay buffer at block b (again the episode index m is omitted for notational simplicity). The loss function for the mixing network parameter θ is given by

$$L_{TD}(\theta) = \left[R_t + \gamma \text{max}_{\mathbf{u}'} Q^{tot}_{\theta^-}(s_{t+1}, \mathbf{u}') - Q^{tot}_{\theta}(s_t, \mathbf{u}_t)\right]^2,$$

where θ^- is the parameter of the target network for the mixing network.

Episodic Correction: Consider the i-th agent's sample history in the m-th selected episode: $(o_1^i, u_1^i, o_2^i, u_2^i, \cdots, o_{T_e}^i, u_{T_e}^i)$. Based on the current Q-function estimates $Q^i_{\theta_i[b]}$ and $Q^{tot}_{\theta[b]}$, as a subgoal, we picked the partial state within the sequence $(o_1^i, u_1^i, o_2^i, u_2^i, \cdots, o_{T_e}^i, u_{T_e}^i)$ that maximizes the weighted sum of local Q-value and total Q-value, where the maximizing time step is $t^i_\star$, as seen in eq. (1), and the proxy reward R_t and individual rewards r_t^i for block b were designed based on this. Thus, based on the current Q-function estimates $Q^i_{\theta_i[b]}$ and $Q^{tot}_{\theta[b]}$, the subsequence $(o_1^i, u_1^i, o_2^i, u_2^i, \cdots, o^i_{t^i_\star - 1}, u^i_{t^i_\star - 1})$ is a proper sequence which increases the Q-value towards the maximum at $t^i_\star$. On the other hand, the subsequence $(o^i_{t^i_\star}, u^i_{t^i_\star}, o^i_{t^i_\star + 1}, u^i_{t^i_\star + 1}, \cdots, o^i_{T_e}, u^i_{T_e})$ is an improper sequence according to the current Q-function estimates $Q^i_{\theta_i[b]}$ and $Q^{tot}_{\theta[b]}$ because this subsequence deteriorates the Q-value away from the maximum. So, it is highly likely that for o_t^i in this latter subsequence, the (known) action u_t^i is a bad action making transition to a partial state o^i_{t+1} even away from the maximum. Therefore, we want to try diverse actions other than the curren action at o_t^i in this latter subsequence. For this exploration purpose, we apply maximum entropy principle to the timesteps after $t^i_\star$. That is, we additionally incorporate the loss function $L_{E,t}(\theta_i)$ for the local utility network parameter θ_i, given by

$$L_{E,t}(\theta_i) = D_{KL}\left(\hat{\pi}_t^i(\theta_i)(\cdot|o_t^i)||\pi_U(\cdot|o_t^i)\right), t = t^i_\star, \cdots, T_e, \quad (8)$$

where $D_{KL}(\cdot||\cdot)$ is the KL divergence (KLD), π_U is the uniform distribution over the action space U, and $\hat{\pi}_t^i(\theta_i)$ is a softened Q-distribution defined as

$$\hat{\pi}_t^i(\theta_i)(u|o_t^i) = \frac{\exp(Q^i_{\theta_i}(o_t^i, u^i))}{\sum_{v^i}\exp(Q^i_{\theta_i}(o_t^i, v^i))}, \quad \forall u \in U. \quad (9)$$

Note that minimizing KLD between $\hat{\pi}_t^i(\theta_i)(\cdot|o_t^i)$ and the uniform distribution is equivalent to maximizing the entropy of $\hat{\pi}_t^i(\theta_i)(\cdot|o_t^i)$. Note that this extra-loss is due to our way of choosing subgoal based on eq. (1). Here, in eq. (9) we do not consider a temperature parameter in exponentiation for simplicity. Instead, a weighting factor for this exploration loss is included in the total loss, as seen below.

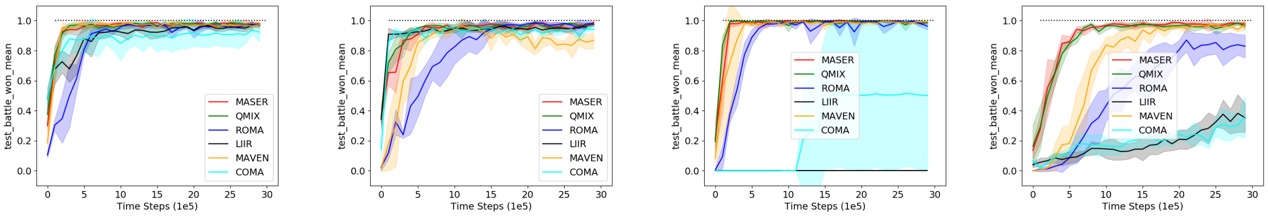
The overall learning loss for each of the m-th selected episode is given by the sum of total TD loss, individual TD losses, episodic correction losses, and representation losses:

$$\begin{aligned} L(\theta, \theta_i, \phi^i) &= L_{TD}(\theta) \quad (10)\\ &+ \sum_{i=1}^{N}\Big[\lambda_I L_i(\theta_i) + \lambda_E \sum_{t=t^i_\star}^{T_e} L_{E,t}(\theta_i) + \lambda_D L_D(\phi^i)\Big], \end{aligned}$$

where λ_I, λ_E and λ_D are the weights for the losses. Finally, the overall learning losses of all M episodes are added and gradient descent to update the parameters is performed based on the summed learning loss. Once the parameter update is done for block b, a new episode is sampled from an ϵ-greedy policy induced from the updated Q-network parameters and is stored into the replay buffer. Then, the learning process for block $b + 1$ starts.

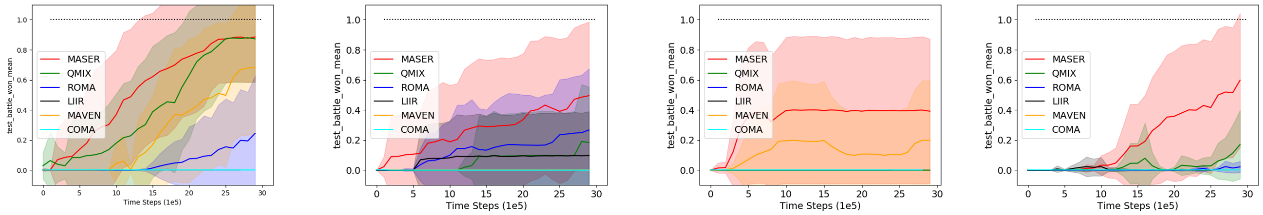
5. Experiments

To evaluate MASER, we considered the widely-used StarCraft II micromanagement benchmark (SMAC) environment. We conducted experiments on four maps: 3 marines_vs_3 marines, 8 marines_vs_8 marines, 2 marines_vs_1 zealot, and 2 stalkers+3 zealots_vs_2 stalkers+3 zealots (denoted as 3m, 8m, 2m_vs_1z, 2s3z) in SMAC with dense and sparse reward and compared the performance with state-of-the-art MARL algorithms: QMIX (Rashid et al., 2018b), ROMA (Wang et al., 2020a), LIIR (Du et al., 2019), MAVEN (Mahajan et al., 2019), and COMA (Foerster et al., 2018a).



(a) 3m (dense)
(b) 8m (dense)
(c) 2m_vs_1z (dense)
(d) 2s3z (dense)

Figure 4. Performance of the considered algorithms on the conventional dense-reward setting



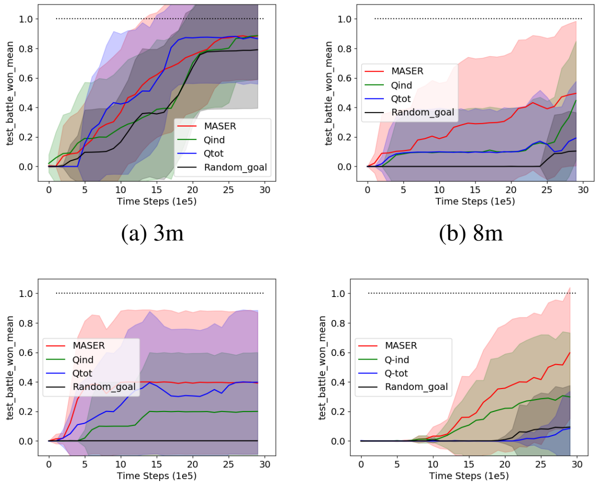
(a) 3m (sparse)
(b) 8m (sparse)
(c) 2m_vs_1z (sparse)
(d) 2s3z (sparse)

Figure 5. Performance of the considered algorithms on the sparse-reward setting

Table 1. Reward Setting

	Dense reward	Sparse reward
All enemies die (Win)	+200	+200
One enemy dies	+10	+10
One ally dies	-5	-5
Enemy's health	-Enemy's remaining health	-
Ally's health	+Ally's remaining health	-
Other elements	+/- with other components	-

The reward setting for the dense and sparse reward cases are shown in Table 1. The dense reward setting is the conventional reward setting in which proper reward is given not only for major events but also for subsidiary side events. To make a sparse-reward StarCraft II setting, we sparsified the rewards of the dense setting as follows: In the sparse-reward setting, the task is made more difficulty by sparsifying the reward function. That is, a reward is given only when one or all enemies die or when one ally dies with no additional reward for state information such as enemy's and ally's health. For evaluation, in the case of dense reward, we carried out experiments with 4 different random seeds because the dense reward case has lower variance than the sparse reward case. On the other hand, we carried out experiments with 10 different random seeds in the sparse case. The result is summarized as the mean of 'test battle won mean' which shows the performance in the SMAC environment, and the shaded area in each figure represents one standard deviation. The hyperparameter setting is available in Appendix A.



(a) 3m
(b) 8m
(c) 2m_vs_1z
(d) 2s3z

Figure 6. Performance with respect to different subgoal generation

Performance on StarCraft II Micromanagement: We first ran all the considered algorithms on the conventional dense-reward setting and the result is shown in Fig. 4. It is seen that most of the algorithms work well in 3m, 8m, 2m_vs_1z, and some algorithms are worse than others in 2s3z. We then ran the algorithms on the difficult sparse-reward setting and the result is shown in Fig. 5. It is seen that MASER significantly outperforms other state-of-the-art algorithms: roll-based ROMA, intrinsic reward learning LIIR, exploration-based MAVEN in the difficult sparse-

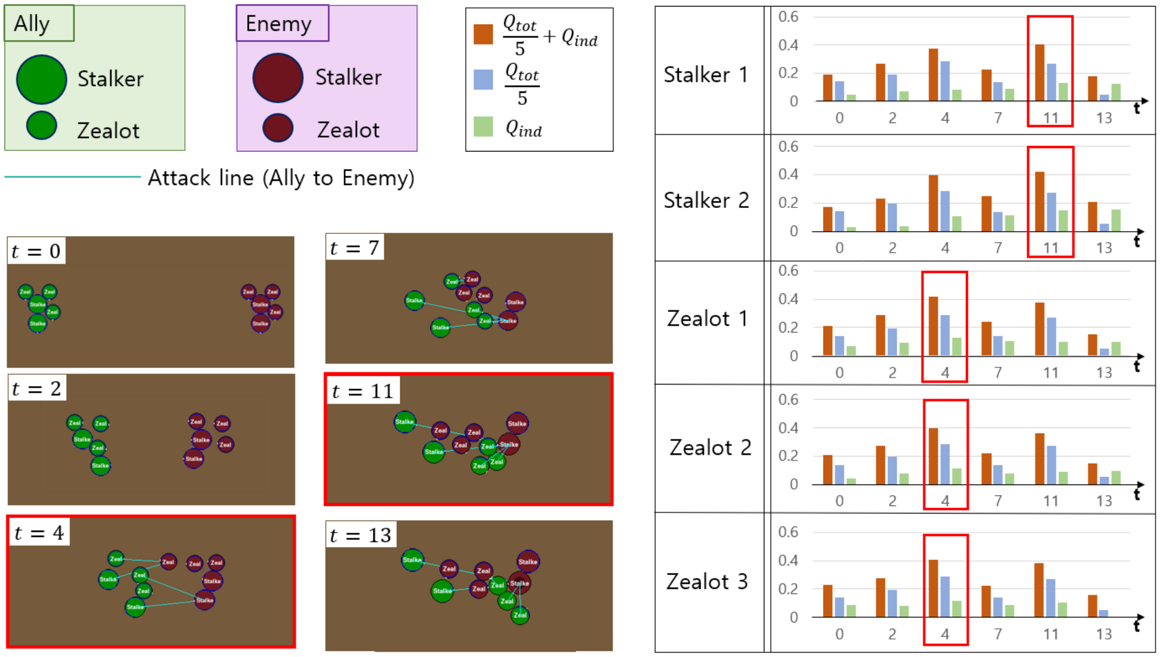
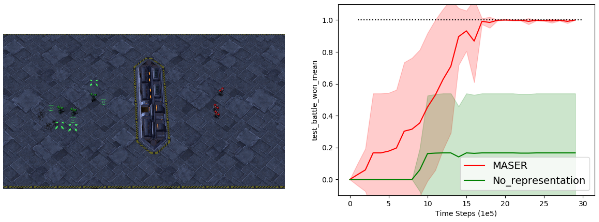


Figure 7. Selected subgoals for agents: Just before fully learning the task (2s3z). The value that each color indicates in the bar graphs is described in the upper middle region.

reward multi-agent setting. In the relatively easy sparse 3m task, QMIX and MAVEN can also learn the task, as seen in Fig. 5a. In the more difficult sparse 8m and 2m_vs_1z tasks, however, other algorithms except MASER have difficulty in learning the tasks, as seen in Figs. 5b and 5c. We tested MASER on a heterogeneous environment 2s3z. It is seen that MASER significantly outperforms other algorithms.

Subgoal Generation: In order to see the effectiveness of our subgoal design, we compared MASER with the method of choosing random subgoals for agents from the replay buffer. We also compared with the case in which the subgoal is selected based only on individual Q-value Q^i or only on total Q-value Q^{tot} (MASER considers both individual and total Q-values by setting $\alpha = 0.5$ in eq. (1).) The comparison result on several StarCraft II micromanagement tasks are shown in Fig. 6. It is seen that considering Q-values performs better than the random selection, and MASER considering both individual and total Q-values performs best. We recognize from Fig. 6 that considering individual Q-values is important. To see the reason, we investigated how subgoals for agents for the 2s3z task were selected by MASER. In 2s3z, domain knowledge tells us that it is advantageous that stalkers with weak health perform long-distance attacks and zealots with high health perform short-distance attacks. Fig. 7 shows the selected subgoals for all five agents just before fully learning the task (around 250M timesteps). The selected subgoals for stalkers are the partial state at $t^i_\star = 11$ and the selected subgoals for zealots are the partial states at $t^i_\star = 4$ from the chosen episode from the buffer (for $t^i_\star$, please see eq. (1)). The corresponding situations are shown in the left figures. In the figure at $t = 4$, we can see that zealots with high health are ahead of stalkers with weak health, protecting stalkers and preparing for short-distance attacks. In the figure at $t = 11$, we can see that stalkers are helping zealots from a long distance while multiple zealots are attacking a single enemy stalker. Both situations are advantageous in 2s3z from our domain knowledge. We can observe that by considering both individual and total Q-values, MASER leads to certain role assignments like in ROMA, while achieving the global goal together.

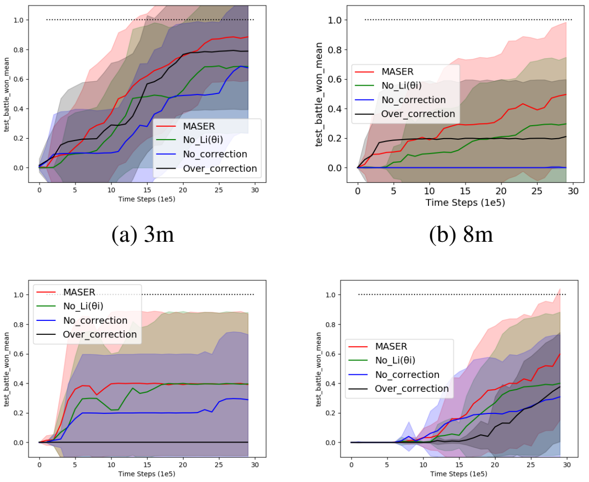


(a) 3m-cliff environment

(b) 3m-cliff

Figure 8. Impact of Q-function-based actionable representation

Q-function-based Actionable Distance: We tested our representation learning based on the proposed actionable distance using Q-function. For this, we created an environ-



(a) 3m

(b) 8m

(c) 2m_vs_1z

(d) 2s3z

Figure 9. Performance regarding individual loss and episodic correction

ment, given by 3m with a cliff with sparse reward, as shown in Fig. 8a, which has a blocking cliff. Fig. 8b shows the performance of MASER in this environment, averaged over 6 different random seeds. Fig. 8b shows the result. It is seen that MASER with the intrinsic reward in eq. (2) learning the representation ϕ^i based on Q-function effectively learns the task, while the baseline with no representation learning mostly fails in learning.

Further Ablation Study: For further ablation study, we considered the following :

- No loss function $L_i(\theta_i)$ for individual Q update in eq. (10): In this case, the parameters of the local utility networks are updated solely by back propagation from $L_{TD}(\theta)$. This case is denoted as 'No_Li(θi) in Fig.9.

- No episodic correction mentioned in Sec. 4.4. This case is denoted as 'No correction' in Fig. 9.

- Episodic correction from the start of the block: In this case, the reward design is the same, but the loss in eq. (8) starts from $t = 1$ not $t^i_\star$. This case is denoted as 'over correction' in Fig. 9.

The results on StarCraft II micromanagement tasks '3m', '8m', '2m_vs_1z' and '2s3z' with these components ablated are shown in Fig. 9. It is seen that each component of MASER contributes to the overall good performance of MASER.

The source code of the proposed algorithm is available at https://github.com/Jiwonjeon9603/MASER.

6. Conclusion

In this paper, we have introduced a novel method of assigning subgoals from the experience replay buffer for sparse-reward multi-agent RL environments. We have chosen the subgoals for multiple agents as the partial states in episodes selected from the replay buffer, maximizing the mixed individual and total Q-values. This choice enables the agents to learn behavior optimizing both individual roles and the global goal. Using the chosen subgoals, we have generated intrinsic reward based on representation learning using a novel Q-function-based actionable distance and designed an overall reward architecture for sparse-reward multi-agent RL. Finally, we have demonstrated the effectiveness of the proposed MARL learning scheme with StarCraft II micromanagement tasks, and experiment results show its good performance in the considered sparse-reward MARL tasks.

Acknowledgments

This work was conducted by Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) (UD190031RD).

References

Amari, S.-i. *Information geometry and its applications*, volume 194. Springer, 2016.

Andriotis, C. and Papakonstantinou, K. Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering & System Safety*, 191:106483, 2019.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

Chane-Sane, E., Schmid, C., and Laptev, I. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pp. 1430–1440. PMLR, 2021.

Christianos, F., Schäfer, L., and Albrecht, S. V. Shared experience actor-critic for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.07169*, 2020.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Du, Y., Han, L., Fang, M., Liu, J., Dai, T., and Tao, D. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. 2019.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018a.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018b.

Ghosh, D., Gupta, A., and Levine, S. Learning actionable representations with goal-conditioned policies. *arXiv preprint arXiv:1811.07819*, 2018.

Hausknecht, M. and Stone, P. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

Huang, Z., Liu, F., and Su, H. Mapping state space using landmarks for universal goal reaching. *arXiv preprint arXiv:1908.05451*, 2019.

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P. A., Strouse, D., Leibo, J. Z., and De Freitas, N. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. *arXiv preprint arXiv:1810.08647*, 2018.

Kim, W., Jung, W., Cho, M., and Sung, Y. A maximum mutual information framework for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.02732*, 2020a.

Kim, W., Park, J., and Sung, Y. Communication in multi-agent reinforcement learning: Intention sharing. In *International Conference on Learning Representations*, 2020b.

Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29: 3675–3683, 2016.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., and Ye, J. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference*, pp. 983–994, 2019.

Liu, I.-J., Jain, U., Yeh, R. A., and Schwing, A. Cooperative exploration for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6826–6836. PMLR, 2021.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. *arXiv preprint arXiv:1910.07483*, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Oliehoek, F. A. Decentralized pomdps. In *Reinforcement Learning*, pp. 471–503. Springer, 2012.

OroojlooyJadid, A. and Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018a.

Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018b.

Ryu, H., Shin, H., and Park, J. Remax: Relational representation for multi-agent exploration. *arXiv preprint arXiv:2008.05214*, 2020.

Son, K., Kim, D., Kang, W. J., Hostallero, D. E., and Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5887–5896. PMLR, 2019.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Tang, H., Hao, J., Lv, T., Chen, Y., Zhang, Z., Jia, H., Ren, C., Zheng, Y., Meng, Z., Fan, C., et al. Hierarchical deep multiagent reinforcement learning with temporal abstraction. *arXiv preprint arXiv:1809.09332*, 2018.

Wang, T., Dong, H., Lesser, V., and Zhang, C. Roma: Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020a.

Wang, T., Gupta, T., Mahajan, A., Peng, B., Whiteson, S., and Zhang, C. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020b.

Wu, H., Li, H., Zhang, J., Wang, Z., and Zhang, J. Generating individual intrinsic reward for cooperative multi-agent reinforcement learning. *International Journal of Advanced Robotic Systems*, 18(5):17298814211044946, 2021.

Wu, Y., Tucker, G., and Nachum, O. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

Yang, J., Borovikov, I., and Zha, H. Hierarchical cooperative multi-agent reinforcement learning with skill discovery. *arXiv preprint arXiv:1912.03558*, 2019.

Zhang, T., Guo, S., Tan, T., Hu, X., and Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *arXiv preprint arXiv:2006.11485*, 2020.

A. Details in Implementation

We built MASER on top of QMIX (Rashid et al., 2018b). Our code is based on Pytorch and we used NVIDIA-TITAN Xp. The values of hyper-parameters are shown in Table 2.

In MASER, the most recent 5000 episodes are stored in the replay buffer, and the mini-batch size is 32. MASER updates the target network every 200 episodes. We used RMSProp for the optimizer and the learning rate of the optimizer is 0.0005. The discounted factor of expected reward (i.e. return) is 0.99. And the value of epsilon for epsilon-greedy Q-learning starts at 1.0 and ends at 0.05 with 50000 anneal time. During the training, the maximum time step is 3005000 in all experiments. We carried out 10 random seeds for the sparse environments, 4 random seeds for the dense environment, and 6 random seeds for the '3m with cliff' environment.

The agent network is a DRQN (Hausknecht & Stone, 2015) which consists of GRU (Chung et al., 2014) for the recurrent layer with a 64-dimensional hidden layer, and 64-dimensional fully-connected layer before and after the GRU layer. For the mixing network, the embedding dimension is 32. For the representation network, we used a neural network with 1 hidden layer with ReLU activation function and 1 output layer with a linear function. The dimension of an input layer, hidden layer, and output layer was the observation dimension, 128, and the action dimension, respectively.

The hyper-parameters for MASER are as follows. For '8m environment', α is 0.5, λ is 0.03, and λ_I, λ_D and λ_E are 0.0007, 0.0003 and 0.0003, respectively. All experiments except for '8m environment', α is 0.5, λ is 0.03, and λ_I, λ_D, λ_E are all 0.001 respectively.

Table 2. Hyper-parameters of MASER and five baselines for SMAC tasks.

	MASER	QMIX	ROMA	LIIR	MAVEN	COMA
Replay buffer size	5000	5000	5000	32	5000	8
Mini-batch size	32	32	32	32	32	8
Optimizer	RMSProp	RMSProp	RMSProp	RMSProp	RMSProp	RMSProp
Agent Runner	episode	episode	parallel	parallel	episode	parallel
Learning rate	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
Discounted factor	0.99	0.99	0.99	0.99	0.99	0.99
Epsilon anneal step	50000	50000	50	50000	50000	100000
Target update interval	200	200	200	200	200	200
Mixing network dimension	32	32	32	-	32	-