LAGMA: LAtent Goal-guided Multi-Agent Reinforcement Learning

Hyungho Na [1] **Il-Chul Moon** [1 2]

Abstract

In cooperative multi-agent reinforcement learning (MARL), agents collaborate to achieve common goals, such as defeating enemies and scoring a goal. However, learning goal-reaching paths toward such a semantic goal takes a considerable amount of time in complex tasks and the trained model often fails to find such paths. To address this, we present **LA**tent **G**oal-guided **M**ulti-**A**gent reinforcement learning (**LAGMA**), which generates a goal-reaching trajectory in latent space and provides a latent goal-guided incentive to transitions toward this reference trajectory. LAGMA consists of three major components: (a) quantized latent space constructed via a modified VQ-VAE for efficient sample utilization, (b) goal-reaching trajectory generation via extended VQ codebook, and (c) latent goal-guided intrinsic reward generation to encourage transitions towards the sampled goal-reaching path. The proposed method is evaluated by StarCraft II with both dense and sparse reward settings and Google Research Football. Empirical results show further performance improvement over state-of-the-art baselines.

1. Introduction

Centralized training and decentralized execution (CTDE) paradigm (Oliehoek et al., 2008; Gupta et al., 2017) especially with value factorization framework (Sunehag et al., 2017; Rashid et al., 2018; Wang et al., 2020a) has shown its success on various cooperative multi-agent tasks (Lowe et al., 2017; Samvelyan et al., 2019). However, in more complex tasks with dense reward settings, such as super hard maps in StarCraft II Multi-agent Challenge (SMAC) (Samvelyan et al., 2019) or in sparse reward settings, as well as Google Research Football (GRF) (Kurach et al., 2020); learning optimal policy takes long time, and trained models even fail to achieve a common goal, such as destroying all enemies in SMAC or scoring a goal in GRF. Thus, researchers focus on sample efficiency to expedite training (Zheng et al., 2021) and encourage committed exploration (Mahajan et al., 2019; Yang et al., 2019; Wang et al., 2019).

[1]Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea. [2]summary.ai, Daejeon, Republic of Korea. Correspondence to: Hyungho Na <gudgh723@gmail.com>, Il-Chul Moon <icmoon@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

To enhance sample efficiency during training, state space abstraction has been introduced in both model-based (Jiang et al., 2015; Zhu et al., 2021; Hafner et al., 2020) and model-free settings (Grześ & Kudenko, 2008; Tang & Agrawal, 2020; Li et al., 2023). Such sample efficiency can be more important in sparse reward settings since trajectories in a replay buffer rarely experience positive reward signals. However, such methods have been studied within a single-agent task without expanding to multi-agent settings.

To encourage committed exploration, goal-conditioned reinforcement learning (GCRL) (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017) has been widely adopted in a single agent task, such as complex path finding with a sparse reward (Nasiriany et al., 2019; Zhang et al., 2020; Chane-Sane et al., 2021; Kim et al., 2023; Lee et al., 2023). However, GCRL concept has also been limitedly applied to multi-agent reinforcement learning (MARL) tasks since there are various difficulties: 1) a goal is not explicitly known, only a semantic goal can be found during training by reward signal; 2) partial observability and decentralized execution in MARL makes impossible to utilize path planning with global information during execution, only allowing such planning during centralized training; 3) most MARL tasks seek not the shortest path, but the coordinated trajectory, which renders single-agent path planning in GCRL be too simplistic in MARL tasks.

Motivated by methods employed in single-agent tasks, we consider a general cooperative MARL problem as finding trajectories toward semantic goals in latent space.

Contribution. This paper presents **LA**tent **G**oal-guided **M**ulti-**A**gent reinforcement learning (**LAGMA**). LAGMA generates a goal-reaching trajectory in latent space and provides a latent goal-guided incentive to transition toward this reference trajectory during centralized training.

- **Modified VQ-VAE for quantized embedding space construction:** As one measure of efficient sample utilization, we use Vector Quantized-Variational Autoencoder(VQ-VAE) (Van Den Oord et al., 2017) which projects states to a quantized vector space so that a common latent can be used as a representative for a wide range of embedding space. However, state distributions in high dimensional MARL tasks are quite limited to small feasible subspace unlike image generation tasks, whose inputs or states often utilize a full state space. In such a case, only a few quantized vectors are utilized throughout training when adopting the original VQ-VAE. To make quantized embedding vectors distributed properly over the embedding space of feasible states, we propose a modified learning framework for VQ-VAE with a novel *coverage loss*.

- **Goal-reaching trajectory generation with extended VQ codebook:** LAGMA constructs an extended VQ codebook to evaluate the states projected to a certain quantized vector and generate a goal-reaching trajectory based on this evaluation. Specifically, during training, we store various goal-reaching trajectories in a quantized latent space. Then, LAGMA uses them as a reference to follow during centralized training.
- **Latent goal-guided intrinsic reward generation:** To encourage coordinated exploration toward reference trajectories sampled from the extended VQ codebook, LAGMA presents a latent goal-guided intrinsic reward. The proposed latent goal-guided intrinsic reward aims to accurately estimate TD-target for transitions toward goal-reaching paths, and we provide both theoretical and empirical support.

2. Related Works

State space abstraction for RL State abstraction groups states with similar characteristics into a single cluster, and it has been effective in both model-based RL (Jiang et al., 2015; Zhu et al., 2021; Hafner et al., 2020) and model-free settings (Grześ & Kudenko, 2008; Tang & Agrawal, 2020). NECSA (Li et al., 2023) adopted the abstraction of grid-based state-action pair for episodic control and achieved state-of-the-art (SOTA) performance in a general single-RL task. This approach could relax the limitations of inefficient memory usage in the conventional episodic control, but this requires an additional dimensionality reduction technique, such as random projection (Dasgupta, 2013) in high-dimensional tasks. Recently, EMU (Na et al., 2024) presented a semantic embedding for efficient memory utilization, but it still resorts to the episodic buffer, which requires storing both the states and the embeddings. This additional memory usage could be burdensome in tasks with large state space. In contrast to previous research, we employ VQ-VAE for state embedding and estimate the overall value of abstracted states. In this manner, a sparse or delayed reward signal can be utilized by a broad range of states, particularly those in proximity. In addition, thanks to the discretized embeddings, the count-based estimation can be adopted to estimate the value of states projected to each discretized embedding. Then, we generate a reference or goal-reaching trajectory based on this evaluation in quantized vector space and provide an incentive for transitions that overlap with this reference.

Intrinsic incentive in RL In reinforcement learning, balancing exploration and exploitation during training is a paramount issue (Sutton & Barto, 2018). To encourage a proper exploration, researchers have presented various forms of methods in a single-agent case such as modified count-based methods (Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017), prediction error-based methods (Stadie et al., 2015; Pathak et al., 2017; Burda et al., 2018; Kim et al., 2018), and information gain-based methods (Mohamed & Jimenez Rezende, 2015; Houthooft et al., 2016). In most cases, an incentive for exploration is introduced as an additional reward to a TD target in Q-learning or a regularizer to overall loss functions. Recently, diverse approaches mentioned earlier have been adopted in the multi-agent environment to promote exploration (Mahajan et al., 2019; Wang et al., 2019; Jaques et al., 2019; Mguni et al., 2021). As an example, EMC (Zheng et al., 2021) utilizes episodic control (Lengyel & Dayan, 2007; Blundell et al., 2016) as regularization for the joint Q-learning, in addition to a curiosity-driven exploration by predicting individual Q-values. Learning with intrinsic rewards becomes more important in sparse reward settings. However, this intrinsic reward can adversely affect the overall policy learning if it is not properly annealed throughout the training. Instead of generating an additional reward signal solely encouraging exploration, LAGMA generates an intrinsic reward that guarantees a more accurate TD-target for Q-learning, yielding additional incentive toward a goal-reaching path.

Additional related works regarding goal-conditioned reinforcement learning (GCRL) and subtask-conditioned MARL are presented in Appendix C.

3. Preliminaries

Decentralized POMDP A general cooperative multi-agent task with n agents can be formalized as the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek & Amato, 2016). DecPOMDP consists of a tuple $G = \langle I, S, A, P, R, \Omega, O, n, \gamma \rangle$, where I is the finite set of n agents; $s \in S$ is the true state in the global state space S; A is the action space of each agent's action a_i forming the joint action $\boldsymbol{a} \in A^n$; $P(s'|s, \boldsymbol{a})$ is the state transition function determined by the environment; R is a reward function $r = R(s, \boldsymbol{a}, s') \in \mathbb{R}$; O is the observation function generating an individual observation from observation space Ω, i.e., $o_i \in \Omega$; and finally, $\gamma \in [0, 1)$ is a discount factor.

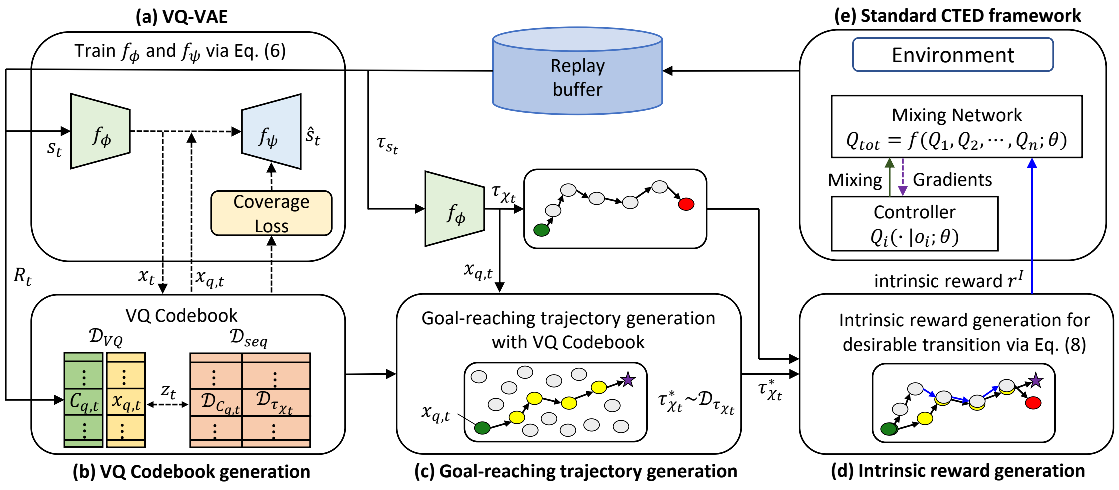

(a) VQ-VAE
Train f_ϕ and f_ψ via Eq. (6)
Coverage Loss
Replay buffer
(e) Standard CTED framework
Environment
Mixing Network $Q_{tot} = f(Q_1, Q_2, \cdots, Q_n; \theta)$
Mixing
Gradients
Controller $Q_i(\cdot|o_i;\theta)$
intrinsic reward r^I
VQ Codebook
Goal-reaching trajectory generation with VQ Codebook
Intrinsic reward generation for desirable transition via Eq. (8)
(b) VQ Codebook generation
(c) Goal-reaching trajectory generation
(d) Intrinsic reward generation

Figure 1: Overview of LAGMA framework.(a) VQ-VAE constructs quantized vector space with coverage loss, while (b) VQ codebook stores goal-reaching sequences from a given $x_{q,t}$. Then, (c) the goal-reaching trajectory is compared with the current batch trajectory to generate (d) intrinsic reward. MARL training is done by (e) the standard CTDE framework.

In a general cooperative MARL task, an agent acquires its local observation o_i at each timestep, and the agent selects an action $a_i \in A$ based on o_i. $P(s'|s, \boldsymbol{a})$ determines a next state s' for a given current state s and the joint action taken by agents $\boldsymbol{a}$. For a given tuple of $\{s, \boldsymbol{a}, s'\}$, R provides an identical common reward to all agents. To overcome the partial observability in DecPOMDP, each agent often utilizes a local action-observation history $\tau_i \in T \equiv (\Omega \times A)$ for its policy $\pi_i(a|\tau_i)$, where $\pi : T \times A \rightarrow [0, 1]$ (Hausknecht & Stone, 2015; Rashid et al., 2018). Additionally, we denote a group trajectory as $\boldsymbol{\tau} =< \tau_1, ..., \tau_n >$.

Centralized Training with Decentralized Execution (CTDE) In fully cooperative MARL tasks, under the CTDE paradigm, value factorization approaches have been introduced by (Sunehag et al., 2017; Rashid et al., 2018; Son et al., 2019; Rashid et al., 2020; Wang et al., 2020a) and achieved state-of-the-art performance in complex multi-agent tasks such as SMAC (Samvelyan et al., 2019). In value factorization approaches, the joint action-value function Q_θ^{tot} parameterized by θ is trained to minimize the following loss function.

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{a},r^{\text{ext}},\boldsymbol{\tau}'\in\mathcal{D}}\big[\big(r^{\text{ext}} + \gamma V_{\theta^-}^{tot}(\boldsymbol{\tau}') - Q_\theta^{tot}(\boldsymbol{\tau},\boldsymbol{a})\big)^2\big] \tag{1}$$

Here, $V_{\theta^-}^{tot}(\boldsymbol{\tau}') = \max_{\boldsymbol{a}'} Q_{\theta^-}^{tot}(\boldsymbol{\tau}', \boldsymbol{a}')$ by definition; $\mathcal{D}$ represents the replay buffer; r^{ext} is an external reward provided by the environment; $Q_{\theta^-}^{tot}$ is a target network parameterized by θ^- for double Q-learning(Hasselt, 2010; Van Hasselt et al., 2016); and Q_θ^{tot} and $Q_{\theta^-}^{tot}$ include both mixer and individual policy network.

Goal State and Goal-Reaching Trajectory In general cooperative multi-agent tasks, undiscounted reward sum, i.e., $R_0 = \Sigma_{t=0}^{T-1} r_t$, is maximized as $R_{\max}$ if agents achieve a semantic goal, such as defeating all enemies in SMAC or scoring a goal in GRF. Thus, we define goal states and the goal-reaching trajectory in cooperative MARL as follows.

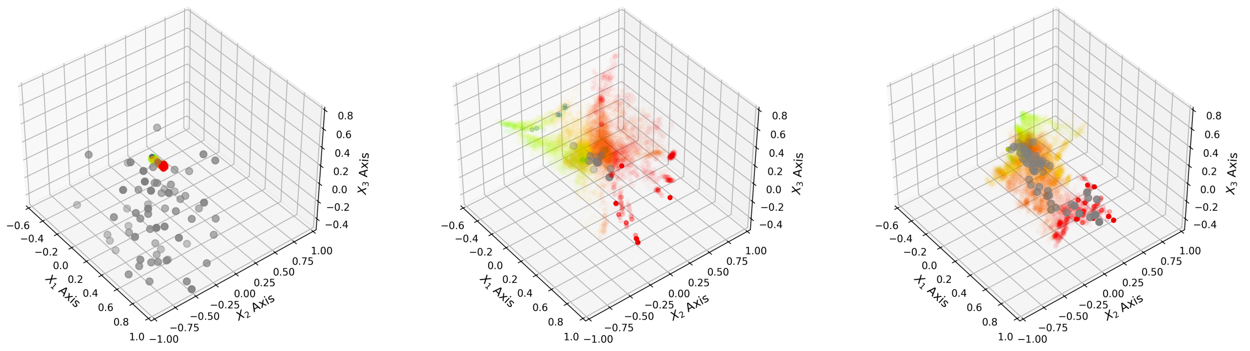
Definition 3.1. (Goal State and Goal-Reaching Trajectory) For a given task dependent $R_{\max}$ and an episodic sequence $\mathcal{T} := \{s_0, \boldsymbol{a_0}, r_0, s_1, \boldsymbol{a_1}, r_1, ..., s_T\}$, when $\Sigma_{t=0}^{T-1} r_t = R_{\max}$ for $r_t \in \mathcal{T}$, we define such an episodic sequence as a goal-reaching sequence and denote as $\mathcal{T}^*$. Then, for $\forall s_t \in \mathcal{T}^*$, $\tau_{s_t}^* := \{s_t, s_{t+1}, ...s_T\}$ is a goal-reaching trajectory and we define the final state of $\tau_{s_t}^*$ as a goal state denoted by s_T^*.

4. Methodology

This section introduces **LA**tent **G**oal-guided **M**ulti-**A**gent reinforcement learning (LAGMA) (Figure 1). We first explain how to construct a proper **(1) quantized embeddings via VQ-VAE**. To this end, we introduce a novel loss term called *coverage loss* to distribute quantized embedding vectors across the overall embedding space. Then, we elaborate on the details of **(2) goal-reaching trajectory generation** with extended VQ codebook. Finally, we propose **(3) a latent goal-guided intrinsic reward** which guarantees a better TD-target for policy learning and thus yields a better convergence on optimal policy.

4.1. State Embedding via Modified VQ-VAE

In this paper, we adopt VQ-VAE as a discretization bottleneck (Van Den Oord et al., 2017) to construct a discretized low-dimensional embedding space. Thus, we first define n_c-trainable embedding vectors (codes) $e_j \in \mathbb{R}^D$ in the codebook where $j = \{1, 2, ..., n_c\}$. An encoder network f_ϕ in VQ-VAE projects a global state s toward D-dimensional



(a) Training without $\mathcal{L}_{\text{cvr}}$ ($\lambda_{\text{cvr}} = 0.0$). (b) Training with $\mathcal{L}^{\text{all}}_{\text{cvr}}$ ($\lambda_{\text{cvr}} = 0.2$). (c) Training with $\mathcal{L}_{\text{cvr}}$ ($\lambda_{\text{cvr}} = 0.2$).

Figure 2: Visualization of embedding results via VQ-VAE. Under SMAC `5m_vs_6m` task, the size of codebook $n_c = 64$, the latent dimension $D = 8$; this illustrates embeddings at training time at T=1.0M. Colored dots represent χ, which is a state presentation before quantization, and gray dots are quantized vector representations belonging to VQ codebook derived from the state representations. Colors from red to purple (rainbow) represent from small to large timestep within episodes.

vector, $x = f_\phi(s) \in \mathbb{R}^D$. Instead of a direct usage of latent vector x, we use a discretized latent x_q by the *quantization process* which maps an embedding vector x to the nearest embedding vector in the codebook as follows.

$$x_q = e_z, \text{where } z = \text{argmin}_j ||x - e_j||_2 \tag{2}$$

Then, the quantized vector x_q is used as an input to a decoder f_ψ which reconstructs the original state s. To train an encoder, a decoder, and embedding vectors in the codebook, we consider the following objective similar to (Van Den Oord et al., 2017; Islam et al., 2022; Lee et al., 2023).

$$\begin{aligned}\mathcal{L}_{VQ}(\phi, \psi, \boldsymbol{e}) &= ||f_\psi([x = f_\phi(s)]_q) - s||_2^2 \\ &+ \lambda_{\text{vq}}||\text{sg}[f_\phi(s)] - x_q||_2^2 + \lambda_{\text{commit}}||f_\phi(s) - \text{sg}[x_q]||_2^2\end{aligned} \tag{3}$$

Here, $[\cdot]_q$ and $\text{sg}[\cdot]$ represent a quantization process and stop gradient, respecitvely. λ_{vq} and λ_{commit} are scale factor for correponsding terms. The first term in Eq. (2) is the reconstruction loss, while the second term represents VQ-objective which makes an embedding vector e move toward $x = f_\phi(s)$. The last term called a commitment loss enforces an encoder to generate $f_\phi(s)$ similar to x_q and prevents its output from growing significantly. To approximate the gradient signal for an encoder, we adopt a straight-through estimator (Bengio et al., 2013).

When adopting VQ-VAE for state embedding, we found that only a few quantized vectors e in the codebook are selected throughout an episode, which makes it hard to utilize such a method for meaningful state embedding. We presumed that the reason is the narrow projected embedding space from feasible states compared to a whole embedding space, i.e., $\mathbb{R}^D$. Thus, most randomly initialized quantized vectors e locate far from the latent space of states in the current replay buffer $\mathcal{D}$, denoted as $\chi = \{x \in \mathbb{R}^D : x = f_\phi(s), s \in \mathcal{D}\}$, leaving only a few e close to x within an episode. To resolve this issue, we introduce the *coverage loss* which minimizes the overall distance between the current embedding x and all vectors in the codebook, i.e., e_j for all $j = \{1, 2, ..., n_c\}$.

$$\mathcal{L}^{\text{all}}_{\text{cvr}}(\boldsymbol{e}) = \frac{1}{n_c}\sum_{j=1}^{n_c} ||\text{sg}[f_\phi(s)] - e_j||_2^2 \tag{4}$$

Although $\mathcal{L}^{\text{all}}_{\text{cvr}}$ could lead embedding vectors toward χ, all quantized vectors tend to locate the center of χ rather than densely covering whole χ space. Thus, we consider a *timestep dependent indexing* $\mathcal{J}(t)$ when computing the coverage loss. The purpose of introducing $\mathcal{J}(t)$ is to make only sequentially selected quantized vectors close to the current embedding x_t so that quantized embeddings are uniformly distributed across χ according to timesteps. Then, the final form of coverage loss can be expressed as follows.

$$\mathcal{L}_{\text{cvr}}(\boldsymbol{e}) = \frac{1}{|\mathcal{J}(t)|}\sum_{j \in \mathcal{J}(t)} ||\text{sg}[f_\phi(s)] - e_j||_2^2 \tag{5}$$

We defer the details of $\mathcal{J}(t)$ construction to Appendix E. By considering the coverage loss in Eq. (5), not only the nearest quantized vector but also all vectors in the codebook move towards overall latent space χ. In this way, χ can be well covered by quantized vectors in the codebook. Thus, we consider the overall learning objective as follows.

$$\mathcal{L}^{tot}_{VQ}(\phi, \psi, \boldsymbol{e}) = \mathcal{L}_{VQ}(\phi, \psi, \boldsymbol{e}) + \lambda_{\text{cvr}}\mathcal{L}_{\text{cvr}}(\boldsymbol{e}) \tag{6}$$

where λ_{cov} is a scale factor for $\mathcal{L}_{\text{cvr}}$.

Figure 2 presents the visualization of embeddings by principal component analysis (PCA) (Wold et al., 1987). In Figure

2, the training without $\mathcal{L}_{\text{cvr}}$ leads to quantized vectors that are distant from χ.

In addition, embedding space χ itself distributes around a few quantized vectors due to the commitment loss in Eq. (3). Considering $\mathcal{L}_{\text{cvr}}^{\text{all}}$ makes quantized vectors close to χ but they majorly locate around the center of χ rather than distributed properly. On the other hand, the proposed $\mathcal{L}_{\text{cvr}}$ results in well-distributed quantized vectors over χ space so that they can properly represent latent space of $s \in \mathcal{D}$. Figure 3 presents the occurrence of recalled quantized vectors for state embeddings in Fig. 2. We can see that training with λ_{cvr} guarantees quantized vectors well distributed across χ. Appendix E presents the training algorithm for the proposed VQ-VAE.

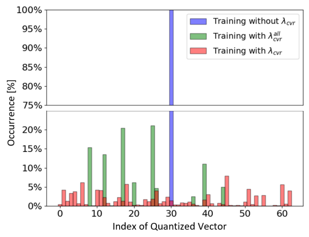


Figure 3: Histogram of recalled quantized vector.

4.2. Goal-Reaching Trajectory Generation with Extended VQ Codebook

After constructing quantized vectors in the codebook, we need to properly estimate the value of states projected to each quantized vector. Note that the estimated value of each quantized vector is used when generating an additional incentive to desired transitions, i.e., transition toward a goal-reaching trajectory. Thanks to the quantized vectors in the codebook, we can resort to count-based estimation for the value estimation of a given state. For a given s_t, a cumulative return from s_t denoted as $R_t = \Sigma_{i=t}^{T-1}\gamma^{i-t}r_i$, and $x_{q,t} = [x_t = f_\phi(s_t)]_q$, the value of $x_{q,t}$ can be computed via count-based estimation as

$$C_{q,t}(x_{q,t}) = \frac{1}{N_{x_{q,t}}} \sum_{j=1}^{N_{x_{q,t}}} R_t^j(x_{q,t}) \tag{7}$$

Here, $N_{x_{q,t}}$ is the visitation count on $x_{q,t}$. However, as an encoder network f_ϕ is updated during training, the match between a specific state s and $x = f_\phi(s)$ can break. Thus, it becomes hard to accurately estimate the value of s via the count-based visit on $x_{q,t}$. To resolve this, we adopt a moving average with a buffer size of m when computing $C_{q,t}(x_{q,t})$ and store the updated value in the extended codebook, $\mathcal{D}_{VQ}$. Appendix D presents structural details of $\mathcal{D}_{VQ}$.

After constructing $\mathcal{D}_{VQ}$, now we need to determine a goal-reaching trajectory $\tau_{s_t}^*$, defined in Definition 3.1, in the latent space. This trajectory is considered as a reference trajectory to incentivize desired transitions. Let the state sequence from s_t and its corresponding latent sequence projected by f_ϕ as $\tau_{s_t} = \{s_t, s_{t+1}, s_{t+2}, ..., s_T\}$ and $\tau_{x_t} = f_\phi(\tau_{s_t})$, respectively. Then, latent sequence after quantization process can be expressed as $\tau_{\chi_t} = [f_\phi(\tau_{x_t})]_q = \{x_{q,t}, x_{q,t+1}, x_{q,t+2}, ..., x_{q,T}\}$. To evaluate the value the of trajectory τ_{χ_t}, we use $C_{q,t}$ value in the codebook $\mathcal{D}_{VQ}$ of an initial quantized vector $x_{q,t}$ in τ_{χ_t}.

To encourage desired transitions contained in τ_{χ_t} with high $C_{q,t}$ value, we need to keep a sequence data of τ_{χ_t}. For a given starting node $x_{q,t}$, we keep top-k sequences in $\mathcal{D}_{seq}$ based on their $C_{q,t}$. Thus, $\mathcal{D}_{seq}$ consists of two parts; $\mathcal{D}_{\tau_{\chi_t}}$ stores top-k sequences of τ_{χ_t} and $\mathcal{D}_{C_{q,t}}$ stores their corresponding $C_{q,t}$ values. Updating algorithm for a sequence buffer $\mathcal{D}_{seq}$ and structural details of $\mathcal{D}_{seq}$ are presented in Appendix D.

As in Definition 3.1, the highest return in cooperative multi-agent tasks can only be achieved when the semantic goal is satisfied. Thus, once agents have achieved a common goal during training, goal-reaching trajectories starting from various initial positions are stored in $\mathcal{D}_{seq}$. After we construct $\mathcal{D}_{seq}$, a reference trajectory $\tau_{\chi_t}^*$ can be sampled out of $\mathcal{D}_{\tau_{\chi_t}}$. For a given initial position $x_{q,t}$ in the quantized latent space, we randomly sample a reference trajectory or goal-reaching trajectory from $\mathcal{D}_{seq}$.

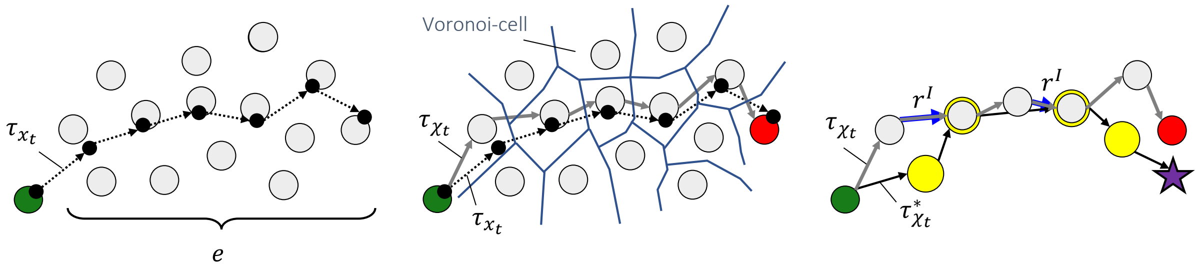
4.3. Intrinsic Reward Generation

With a goal-reaching trajectory $\tau_{\chi_t}^*$ from the current state, we can determine the desired transitions that lead to a goal-reaching path, simply by checking whether the quantized latent $x_{q,t}$ at each timestep t is in $\tau_{\chi_t}^*$. However, before quantized vectors e in the codebook well cover the latent distribution χ, only a few e vectors are selected and thus the same x_q will be repeatedly obtained. In such a case, staying in the same x_q will be encouraged by intrinsic reward if $x_q \in \tau_{\chi_t}^*$. To prevent this, we only provide an incentive to the desired transition toward $x_{q,t+1}$ such that $x_{q,t+1} \in \tau_{\chi_t}^*$ and $x_{q,t+1} \neq x_{q,t}$. A remaining problem is how much we incentivize such a desired transition. Instead of an arbitrary incentive, we want to design an additional reward to guarantee a better TD-target, to converge on optimal policy.

Proposition 4.1. *Provided that $\tau_{\chi_t}^*$ is a goal-reaching trajectory and $s' \in \tau_{\chi_t}^*$, an intrinsic reward $r^I(s') := \gamma(C_{q,t}(s') - \max_{\boldsymbol{a}'} Q_{\theta^-}(s', \boldsymbol{a}'))$ to the current TD-target $y = r(s, \boldsymbol{a}) + \gamma V_{\theta^-}(s')$ guarantees a true TD-target as $y^* = r(s, \boldsymbol{a}) + \gamma V^*(s')$, where $V^*(s')$ is a true value of s'.*

Proof. Please see Appendix A. □

According to Proposition 4.1, when $\tau_{\chi_t}^*$ is a goal-reaching trajectory and $s' \in \tau_{\chi_t}^*$, we can set a true TD-target by adding an intrinsic reward $r^I(s')$ to the current TD-target y, yielding a better convergence on an optimal policy. In the case when a reference trajectory $\tau_{\chi_t}^*$ is not a goal-reaching



(a) Trajectory embedding. (b) Trajectory in quantized latent space. (c) Intrinsic reward generation.

Figure 4: Intrinsic reward generation by comparing the current trajectory in quantized latent space (τ_{χ_t}) with a sampled goal-reaching trajectory ($\tau^*_{\chi_t}$).

trajectory, $r^I(s')$ incentivizes the transition toward the high-return trajectory experienced so far. Thus, we define a latent goal-guided intrinsic reward r^I as follows.

$$r^I_t(s_{t+1}) = \gamma(C_{q,t}(s_{t+1}) - \max_{\boldsymbol{a}'} Q_{\theta^-}(s_{t+1}, \boldsymbol{a}')), \quad \text{if } x_{q,t+1} \in \tau^*_{\chi_t} \text{ and } x_{q,t+1} \neq x_{q,t} \tag{8}$$

Note that $r^I_t(s_{t+1})$ is added to $y_t = r_t + \gamma V_{\theta^-}$ not y_{t+1}. In addition, we can make sure that r^I becomes non-negative so that an inaccurate estimate of $C_{q,t}(s')$ in the early training phase does not adversely affect the estimation of V_{θ^-}. Algorithm 1 summarizes the overall method for goal-reaching trajectory and an intrinsic reward generation. Figure 4 illustrates the schematic diagram of quantized trajectory embeddings τ_{χ_t} via VQ-VAE and intrinsic reward generation by comparing it with a goal-reaching trajectory, $\tau^*_{\chi_t}$.

Algorithm 1 Goal-reaching Trajectory and Intrinsic Reward Generation

```
**Given:** Sequences of the current batch $[\tau^i_{\chi_t}]^B_{i=1}$, a sequence buffer $\mathcal{D}_{seq}$, an update interval $n_{\text{freq}}$ for $\tau^*_{\chi_t}$, and VQ-VAE codebook $\mathcal{D}_{VQ}$
**for** $i = 1$ **to** $B$ **do**
  Compute $R^i_t$
  **for** $t = 0$ **to** $T$ **do**
    Get index $z_t \leftarrow \tau^i_{\chi_t}$
    **if** mod$(t, n_{\text{freq}})$ **then**
      Run **Algorithm 2** to update $\mathcal{D}^{z_t}_{seq}$ with $R^i_t$
      Sample a reference trajectory $\tau^*_{\chi_t}$ from $\mathcal{D}^{z_t}_{seq}$
    **else**
      **if** $z_t \in \tau^*_{\chi_t}$ and $z_t \neq z_{t-1}$ **then**
        Get $C_{q,t} \leftarrow \mathcal{D}^{z_t}_{VQ}.C_{q,t}$
        $(r^I_{t-1})^i \leftarrow \gamma\max(C_{q,t} - \max_{a'} Q_{\theta^-}(s_t, a'), 0)$
      **end if**
    **end if**
  **end for**
**end for**
```

4.4. Overall Learning Objective

This paper adopts a conventional CTDE paradigm (Oliehoek et al., 2008; Gupta et al., 2017), and thus any form of mixer structure can be used for value factorization. We use the mixer structure presented in QMIX (Rashid et al., 2018) similar to (Yang et al., 2022; Wang et al., 2021; Jeon et al., 2022) to construct the joint Q-value (Q^{tot}) from individual Q-functions. By adopting the latent goal-guided intrinsic reward r^I to Eq. (1), the overall loss function for the policy learning can be expressed as follows.

$$\mathcal{L}(\theta) = \left(r^{\text{ext}} + r^I + \gamma \max_{\boldsymbol{a}'} Q^{tot}_{\theta^-}(s', \boldsymbol{a}') - Q^{tot}_{\theta}(s, \boldsymbol{a})\right)^2 \tag{9}$$

Note that here r^I does not include any scale factor to control its magnitude. For an individual policy via Q-function, GRUs are adopted to encode a local action-observation history τ to overcome the partial observability in POMDP similar to most MARL approaches (Sunehag et al., 2017; Rashid et al., 2018; Son et al., 2019; Rashid et al., 2020; Wang et al., 2020a). However, in Eq. (9), we express the equation with s instead of τ for the conciseness and coherence with the mathematical derivation. The overall training algorithm for both VQ-VAE training and policy learning is presented in Appendix E.

5. Experiments

In this section, we present experiment settings and results to evaluate the proposed method. We have designed our experiments with the intention of addressing the following inquiries denoted as Q1-3.

- Q1. The performance of LAGMA in comparison to state-of-the-art MARL frameworks in both dense and sparse reward settings
- Q2. The impact of the proposed embedding method on overall performance
- Q3. The efficiency of latent goal-guided incentive compared to other reward design

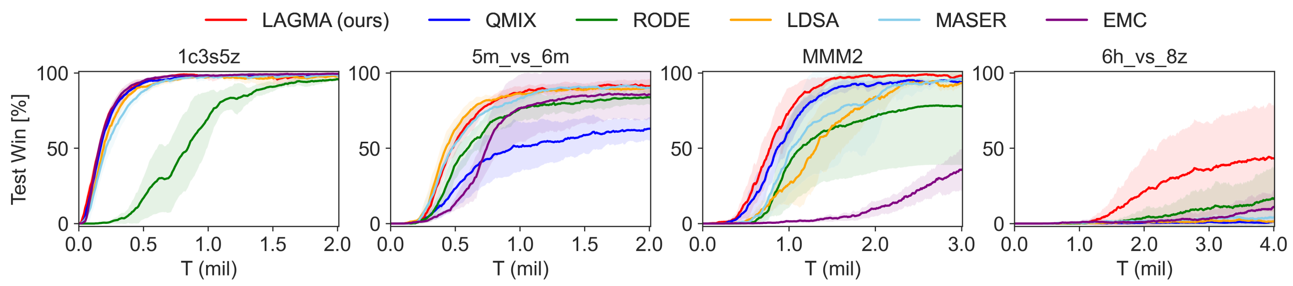


Figure 5: Performance comparison of LAGMA against baseline algorithms on two **easy and hard** SMAC maps: `1c3s5z`, `5m_vs_6m`, and two **super hard** SMAC maps: `MMM2`, `6h_vs_8z`. (Dense reward setting)

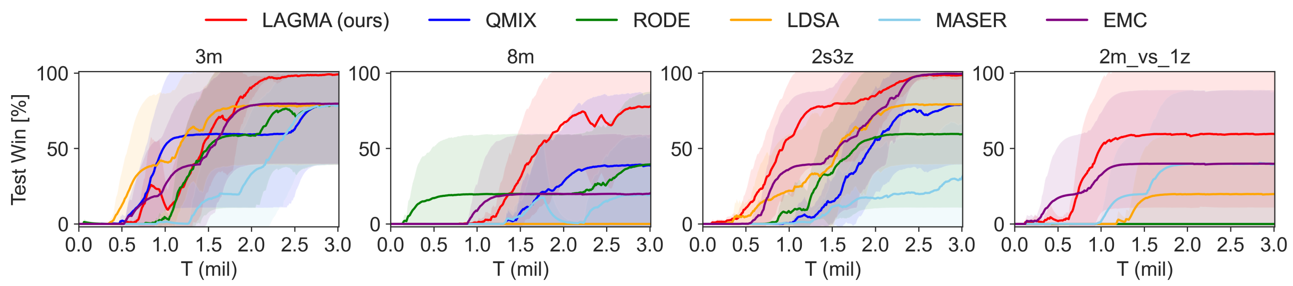


Figure 6: Performance comparison of LAGMA against baseline algorithms on four maps: `3m`, `8m`, `2s3z`, and `2m_vs_1z`. (Sparse reward setting)

We consider complex multi-agent tasks such as SMAC (Samvelyan et al., 2019) and GRF (Kurach et al., 2020) as benchmark problems. In addition, as baseline algorithm, we consider various baselines in MARL such as QMIX (Rashid et al., 2018), RODE (Wang et al., 2021) and LDSA (Yang et al., 2022) adopting a role or skill conditioned policy, MASER (Jeon et al., 2022) presenting agent-wise individual subgoals from replay buffer, and EMC (Zheng et al., 2021) adopting episodic control. Appendix B presents further details of experiment settings and implementations, and Appendix G illustrates the resource usage and the computational cost required for the implementation and training of LAGMA. In addition, additional generalizability tests of LAGMA are presented in Appendix F. Our code is available at: `https://github.com/aailabkaist/LAGMA`.

5.1. Performance evaluation on SMAC

Dense reward settings For dense reward settings, we follow the default setting presented in (Samvelyan et al., 2019). Figure 5 illustrates the overall performance of LAGMA. Thanks to quantized embedding and latent goal-guided incentive, LAGMA shows significant performance improvement compared to the backbone algorithm, i.e., QMIX, and other state-of-the-art (SOTA) baseline algorithms, especially in **super hard** SMAC maps.

Sparse reward settings For a sparse reward setting, we follow the reward design in MASER (Jeon et al., 2022). Appendix B enumerates the details of reward settings. Similar to dense reward settings, LAGMA shows the best performance in sparse reward settings thanks to the latent goal-guided incentive. Sparse reward hardly generates a reward signal in experience replay, thus training with the experience of the exact same state takes a long time to find the optimal policy. However, LAGMA considers the value of semantically similar states projected onto the same quantized vector during training, so its learning efficiency is significantly increased.

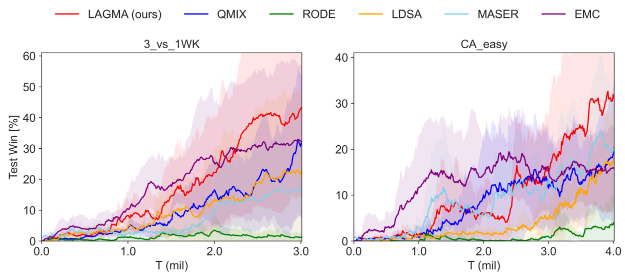


Figure 7: Performance comparison of LAGMA against baseline algorithms on two GRF maps: `3_vs_1WK` and `CounterAttack(CA)_easy`. (Sparse reward setting)

5.2. Performance evaluation on GRF

Here, we conduct experiments on additional sparse reward tasks in GRF to compare LAGMA with baseline algorithms. For experiments, we do not utilize any additional algorithm for sample efficiency such as prioritized experience replay (Schaul et al., 2015) for all algorithms.

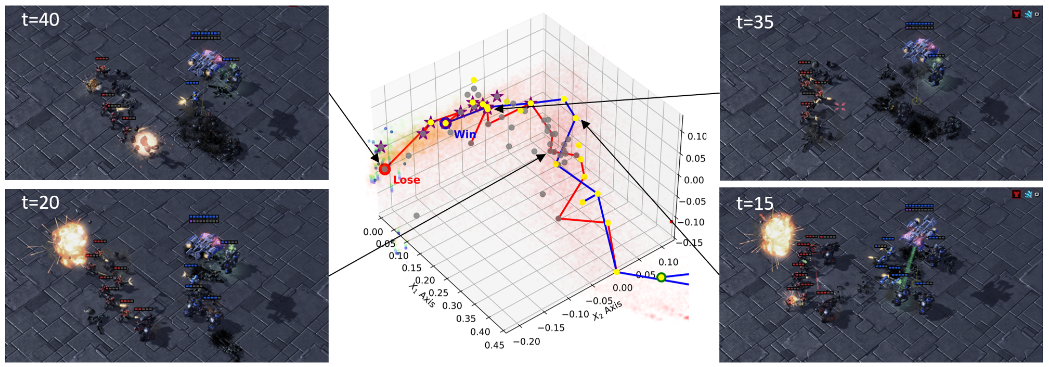


Figure 8: Qualitative analysis on SMAC `MMM2` (red teams are RL-agents). Purple stars represent quantized embeddings of goal states in replay buffer $\mathcal{D}$. Yellow dots indicate the quantized embeddings in a sampled goal-reaching trajectory starting from an initial state denoted by a green dot. Gray dots and transparent dots are the same as Figure 2. Blue and red dots indicate terminal embeddings of two trajectories, respectively.

EMC (Zheng et al., 2021) shows comparable performance by utilizing an episodic buffer, which benefits in generating a positive reward signal via additional episodic control term. However, LAGMA with a modified VQ codebook could guide a scoring policy without utilizing an additional episodic buffer as being required in EMC. Therefore, LAGMA achieves a similar or better performance with less memory requirement.

5.3. Ablation study

In this subsection, we conduct ablation studies to see the effect of the proposed embedding method and latent goal-guided incentive on overall performance. We compare LAGMA (ours) with ablated configurations such as LAGMA (CL-All) trained with $\lambda_{\text{cvr}}^{\text{all}}$ considering all quantized vectors at each timestep and LAGMA (No-CL) trained without coverage loss.

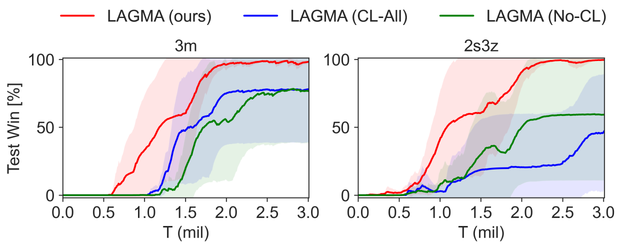


Figure 9: Ablation study considering the coverage loss (CL) on four SMAC maps: `3m` and `2s3z`. (Sparse reward setting)

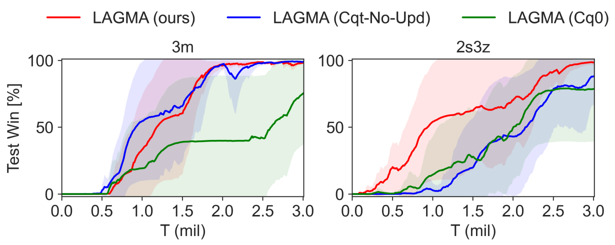


Figure 10: Performance comparison of goal-guided incentive with other reward design choices on two SMAC maps: `3m` and `2s3z`. (Sparse reward setting)

Figure 9 illustrates the effect of the proposed coverage loss in modified VQ-VAE on the overall performance. As shown in Fig. 9, the performance decreases when the model is trained without coverage loss or trained with $\lambda_{\text{cvr}}^{\text{all}}$ instead of λ_{cvr}. The results imply that, without the proposed coverage loss, quantized latent vectors may not cover χ properly and thus x_q can hardly represent the projected states. As a result, a goal-reaching trajectory that consists of a few quantized vectors yields no incentive signal in most transitions.

In addition, we conduct an ablation study on reward design. We consider a sum of undiscounted rewards, $C_{q0} = \Sigma_{t=0}^{T-1} r_t$, for trajectory value estimation instead of $C_{q,t}$, denoted as **LAMGA (Cq0)**. We also consider the LAMGA configuration with goal-reaching trajectory generation only at the initial state denoted by **LAMGA (Cqt-No-Upd)**. Figure 10 illustrates the results. Figure 10 implies that the reward design of $C_{q,t}$ shows a more stable performance than both **LAMGA (Cq0)** and **LAMGA (Cqt-No-Upd)**.

5.4. Qualitative analysis

In this section, we conduct a qualitative analysis to observe how the states in an episode are projected onto quantized vector space and receive latent goal-guided incentive compared to goal-reaching trajectory sampled from $\mathcal{D}_{seq}$. Figure 8 illustrates the quantized embedding sequences of two trajectories: one denoted by a blue line representing a battle-won trajectory and the other denoted by a red line representing a losing trajectory. In Fig. 8, a losing trajectory initially followed the optimal sequence denoted by yellow dots but began to bifurcate at $t = 20$ by losing Medivac and two more allies. Although the losing trajectory still passed through goal-reaching sequences during an episode, it ultimately reached a terminal state without a chance to defeat the enemies at $t = 40$, as indicated by the absence of a purple star. On the other hand, a trajectory that achieved victory followed the goal-reaching path and reached a goal state at the end, as indicated by purple stars. Since only transitions toward the sequences on the goal-reaching path are incentivized, LAGMA can efficiently learn a goal-reaching policy, i.e., the optimal policy in cooperative MARL.

6. Conclusions

This paper presents LAGMA, a framework to generate a goal-reaching trajectory in latent space and a latent goal-guided incentive to achieve a common goal in cooperative MARL. Thanks to the quantized embedding space, the experience of semantically similar states is shared by states projected onto the same quantized vector, yielding efficient training. The proposed latent goal-guided intrinsic reward encourages transitions toward a goal-reaching trajectory. Experiments and ablation studies validate the effectiveness of LAGMA.

Acknowledgements

This research was supported by AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data (IITP) funded by the Ministry of Science and ICT (2022-0-00077).

Impact Statement

This paper primarily focuses on advancing the field of Machine Learning through multi-agent reinforcement learning. While there could be various potential societal consequences of our work, none of which we believe must be specifically highlighted here.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J., Wierstra, D., and Hassabis, D. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

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.

Dasgupta, S. Experiments with random projection. *arXiv preprint arXiv:1301.3849*, 2013.

Ellis, B., Cook, J., Moalla, S., Samvelyan, M., Sun, M., Mahajan, A., Foerster, J., and Whiteson, S. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

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

Grześ, M. and Kudenko, D. Multigrid reinforcement learning with reward shaping. In *International Conference on Artificial Neural Networks*, pp. 357–366. Springer, 2008.

Gupta, J. K., Egorov, M., and Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*, pp. 66–83. Springer, 2017.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hasselt, H. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

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

Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

Islam, R., Zang, H., Goyal, A., Lamb, A., Kawaguchi, K., Li, X., Laroche, R., Bengio, Y., and Combes, R. T. D. Discrete factorial representations as an abstraction for goal conditioned reinforcement learning. *arXiv preprint arXiv:2211.00247*, 2022.

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P., Strouse, D., Leibo, J. Z., and De Freitas, N. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pp. 3040–3049. PMLR, 2019.

Jeon, J., Kim, W., Jung, W., and Sung, Y. Maser: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. In *International Conference on Machine Learning*, pp. 10041–10052. PMLR, 2022.

Jiang, N., Kulesza, A., and Singh, S. Abstraction selection in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 179–188. PMLR, 2015.

Kaelbling, L. P. Learning to achieve goals. In *IJCAI*, volume 2, pp. 1094–8. Citeseer, 1993.

Kim, H., Kim, J., Jeong, Y., Levine, S., and Song, H. O. Emi: Exploration with mutual information. *arXiv preprint arXiv:1810.01176*, 2018.

Kim, J., Seo, Y., Ahn, S., Son, K., and Shin, J. Imitating graph-based planning with goal-conditioned policies. *arXiv preprint arXiv:2303.11166*, 2023.

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, 2016.

Kurach, K., Raichuk, A., Stanczyk, P., Zajkc, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4501–4510, 2020.

Lee, S., Cho, D., Park, J., and Kim, H. J. Cqm: Curriculum reinforcement learning with a quantized world model. *arXiv preprint arXiv:2310.17330*, 2023.

Lengyel, M. and Dayan, P. Hippocampal contributions to control: the third way. *Advances in neural information processing systems*, 20, 2007.

Li, Z., Zhu, D., Hu, Y., Xie, X., Ma, L., Zheng, Y., Song, Y., Chen, Y., and Zhao, J. Neural episodic control with state abstraction. *arXiv preprint arXiv:2301.11490*, 2023.

Liu, Y., Li, Y., Xu, X., Dou, Y., and Liu, D. Heterogeneous skill learning for multi-agent tasks. *Advances in Neural Information Processing Systems*, 35:37011–37023, 2022.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

Mguni, D. H., Jafferjee, T., Wang, J., Perez-Nieves, N., Slumbers, O., Tong, F., Li, Y., Zhu, J., Yang, Y., and Wang, J. Ligs: Learnable intrinsic-reward generation selection for multi-agent learning. *arXiv preprint arXiv:2112.02618*, 2021.

Mohamed, S. and Jimenez Rezende, D. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

Na, H., Seo, Y., and Moon, I.-c. Efficient episodic memory utilization of cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2403.01112*, 2024.

Nasiriany, S., Pong, V., Lin, S., and Levine, S. Planning with goal-conditioned policies. *Advances in Neural Information Processing Systems*, 32, 2019.

Oliehoek, F. A. and Amato, C. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Oliehoek, F. A., Spaan, M. T., and Vlassis, N. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

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, 2018.

Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33: 10199–10210, 2020.

Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

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.

Stadie, B. C., Levine, S., and Abbeel, P. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

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.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Tang, Y. and Agrawal, S. Discretizing continuous action space for on-policy optimization. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pp. 5981–5988, 2020.

Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Wang, J., Ren, Z., Liu, T., Yu, Y., and Zhang, C. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020a.

Wang, T., Wang, J., Wu, Y., and Zhang, C. Influence-based multi-agent exploration. *arXiv preprint arXiv:1910.05512*, 2019.

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

Wang, T., Gupta, T., Mahajan, A., Peng, B., Whiteson, S., and Zhang, C. Rode: Learning roles to decompose multi-agent tasks. *In Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Wold, S., Esbensen, K., and Geladi, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

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

Yang, M., Zhao, J., Hu, X., Zhou, W., Zhu, J., and Li, H. Ldsa: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1698–1710, 2022.

Zhang, T., Guo, S., Tan, T., Hu, X., and Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21579–21590, 2020.

Zheng, L., Chen, J., Wang, J., He, J., Hu, Y., Chen, Y., Fan, C., Gao, Y., and Zhang, C. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *Advances in Neural Information Processing Systems*, 34: 3757–3769, 2021.

Zhu, D., Chen, J., Shang, W., Zhou, X., Grossklags, J., and Hassan, A. E. Deepmemory: model-based memorization analysis of deep neural language models. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1003–1015. IEEE, 2021.

A. Mathematical Proof

Here, we provide the omitted proof of Proposition 4.1.

Proof. Let $y = r(s, \boldsymbol{a}) + \gamma V_{\theta^-}$ be the current TD-target with the target network parameterized with θ^-. Adding an intrinsic reward $r^I(s')$ to y yields $y' = y + r^I(s')$. Now, we need to check whether y' accurately estimates $y^* = r + \gamma V^*(s')$.

$$\begin{aligned}
\mathbb{E}[y'] &= \mathbb{E}[r(s, \boldsymbol{a}) + \gamma V_{\theta^-} + r^I(s')] \\
&= \mathbb{E}[r(s, \boldsymbol{a}) + \gamma \max_{\boldsymbol{a'}} Q_{\theta^-}(s', \boldsymbol{a'}) + \gamma(C_{q,t}(s') - \max_{\boldsymbol{a'}} Q_{\theta^-}(s', \boldsymbol{a'}))] \\
&= \mathbb{E}[r(s, \boldsymbol{a}) + \gamma(C_{q,t}(s'))] \\
&= \mathbb{E}[r(s, \boldsymbol{a}) + \gamma(\mathbb{E}[\sum_{i=t+1}^{T-1} \gamma^{i-(t+1)} r_i])] \\
&= r(s, \boldsymbol{a}) + \gamma \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-2} r_{T-1}] \\
&= r(s, \boldsymbol{a}) + \gamma V^*(s')
\end{aligned} \tag{10}$$

The last equality in Eq. (10) holds since s' is on a goal-reaching trajectory, i.e., $s' \in \tau^*_{\chi_0}$ whose return is maximized, and $\mathbb{E}[r_{t+1} + \gamma r_{t+2} + \cdots]$ is an unbiased Monte-Carlo estimate of $V^*(s')$. □

B. Experiment Details

In this section, we present details of SMAC (Samvelyan et al., 2019) and GRF (Kurach et al., 2020), and we also list hyperparemeter settings of LAGMA for each task. Tables 1 and 2 present the dimensions of state and action spaces and the maximum episodic length.

Table 1: Dimension of the state space and the action space and the episodic length of SMAC

Task	Dimension of state space	Dimension of action space	Episodic length
3m	48	9	60
8m	168	14	120
2s3z	120	11	120
2m_vs_1z	26	7	150
1c3s5z	270	15	180
5m_vs_6m	98	12	70
MMM2	322	18	180
6h_vs_8z	140	14	150

Table 2: Dimension of the state space and the action space and the episodic length of GRF

Task	Dimension of state space	Dimension of action space	Episodic length
3_vs_1WK	26	19	150
CA_easy	30	19	150

In addition, Table 3 presents the task-dependent hyperparameter settings for all experiments. As seen from Table 3, we used similar hyperparameters across various tasks. For an update interval n_{freq} in Algorithm 1, we use the same value $n_{\text{freq}} = 5$ for all experiments. ϵ_T represents annealing time for exploration rate of ϵ-greedy, from 1.0 to 0.05.

After some parametric studies, adjusting hyperparameter for VQ-VAE training such as n^{cd}_{freq} and n^{vq}_{freq}, instead of varying λ values listed as λ_{vq}, λ_{commit}, and λ_{cvr}, provides more efficient way of searching parametric space. Thus, we primarily adjust n^{cd}_{freq} and n^{vq}_{freq} according to tasks, while keeping the ratio between λ values the same.

For hyperparameter settings, we recommend the efficient bounds for each hyperparameter based on our experiments as follows:

- Number of codebook, n_c: 64-512
- Update interval for VQ-VAE, n_{freq}^{vq}: 10-40 (under $n_{\text{freq}}^{cd} = 10$)
- Update interval for extended codebook, n_{freq}^{cd}: 10-40 (under $n_{\text{freq}}^{vq} = 10$)
- Number of reference trajectory, k: 10-30
- Scale factor of coverage loss, λ_{cvr}: 0.25-1.0 (under $\lambda_{\text{vq}} = 1.0$ and $\lambda_{\text{commit}} = 0.5$)

Note that larger values of n_c and k, and smaller values of n_{freq}^{vq} and n_{freq}^{cd} will increase the computational cost.

Table 3: Hyperparameter settings for experiments.

task		n_c	D	λ_{vq}	λ_{commit}	λ_{cvr}	ϵ_T	n_{freq}^{cd}	n_{freq}^{vq}
SMAC (sparse)	3m	256	8	2.0	1.0	1.0	50K	10	40
	8m	256	8	2.0	1.0	1.0	50K	20	10
	2s3z	256	8	2.0	1.0	1.0	50K	10	40
	2m_vs_1z	256	8	2.0	1.0	1.0	500K	20	10
SMAC (dense)	1c3s5z	64	8	1.0	0.5	0.5	50K	40	10
	5m_vs_6m	64	8	1.0	0.5	0.5	50K	40	10
	MMM2	64	8	1.0	0.5	0.5	50K	40	10
	6h_vs_8z	256	8	2.0	1.0	1.0	500K	40	10
GRF (sparse)	3_vs_1WK	256	8	2.0	1.0	1.0	50K	20	10
	CA_easy	256	8	2.0	1.0	1.0	50K	10	20

Table 4 presents the reward settings for SMAC (sparse) which follows the sparse reward settings from (Jeon et al., 2022).

Table 4: Reward settings for SMAC (sparse)

Condition	Sparse reward
All enemies die (Win)	+200
Each enemy dies	+10
Each ally dies	-5

C. Additional Related Works

Goal-conditioned RL vs. Subtask conditioned MARL In a single agent case, Goal-conditioned RL (GCRL) which aims to solve multiple tasks to reach given target-goals has been widely adopted in various tasks including tasks with a sparse reward (Kaelbling, 1993; Schaul et al., 2015; Andrychowicz et al., 2017). GCRL often utilizes the given goal as an additional input to action policy in addition to states (Schaul et al., 2015). Especially, goal-conditioned hierarchical reinforcement learning (HRL) (Kulkarni et al., 2016; Zhang et al., 2020; Chane-Sane et al., 2021) adopts hierarchical policy structure where an upper-tier policy determines subgoal or landmark and a lower-tier policy takes action based on both state and selected a subgoal or landmark.

As one technique, reaching to subgoals generates a reward signal via hindsight experience replay (Andrychowicz et al., 2017), and thus the goal-conditioned policy learn policy to reach the final goal with the help of these intermediate signals. Thus, many researchers (Nasiriany et al., 2019; Zhang et al., 2020; Chane-Sane et al., 2021; Kim et al., 2023; Lee et al., 2023) have studied on how to generate intermediate subgoals to reach final goals.

In the field of MARL, a subtask(Yang et al., 2022), role(Wang et al., 2020b; 2021) or skill(Yang et al., 2019; Liu et al., 2022) conditioned policy adopted in a hierarchical MARL structure has a structural commonality with a goal-conditioned RL in that lower-tier policy network use designated subtask by the upper-tier network as an additional input when determining individual action. In MARL tasks, such subtasks, roles, or skills are a bit different from subgoals in GCRL, as they are adopted to decompose action space for efficient training or for subtask-dependent coordination. Another major difference is that in a general MARL task, the final goal is not defined explicitly unlike a goal-conditioned RL. MASER (Jeon et al., 2022) adopts the subgoal generation scheme from goal-conditioned RL when it generates an intrinsic reward based on the Euclidean distance between actionable representations (Ghosh et al., 2018) of the current and subgoal observation. However, this signal does not guarantee the consistency with learning signal for the joint Q-function. In contrast to MASER, we adopt a latent goal-guided incentive during a centralized training phase based on whether visiting on the promising subgoals or goals in the latent space. Also, the generated incentive by LAGMA theoretically guarantees a better TD-target, yielding better convergence on the optimal policy.

D. Structure of Extended VQ Codebook

To compute $C_{q,t}$ via a moving average, data is stored in a FIFO (First in, First Out) style to the codebook $\mathcal{D}_{VQ}$, similar to a replay buffer $\mathcal{D}$. After computing $C_{q,t}(x_{q,t})$ with the current R_t, we update the value of $C_{q,t}(x_{q,t})$ in $\mathcal{D}_{VQ}$ as $\mathcal{D}_{VQ}^{z_t}.C_{q,t} \leftarrow C_{q,t}(x_{q,t})$ where z_t is an index of a quantized vector $x_{q,t}$.

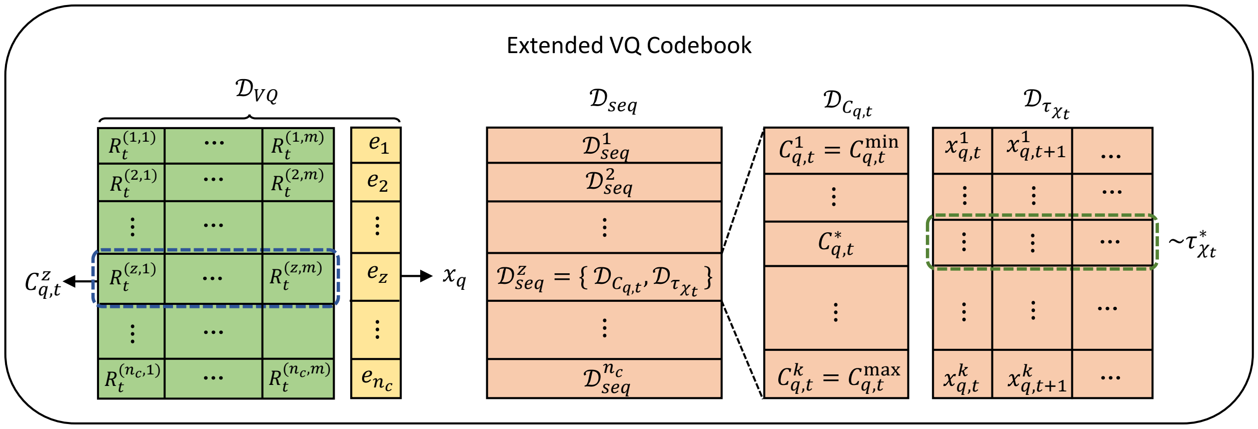

Extended VQ Codebook
$\mathcal{D}_{VQ}$
$R_t^{(1,1)}$ … $R_t^{(1,m)}$
$R_t^{(2,1)}$ … $R_t^{(2,m)}$
$R_t^{(z,1)}$ … $R_t^{(z,m)}$
$R_t^{(n_c,1)}$ … $R_t^{(n_c,m)}$
e_1 e_2 e_z e_{n_c}
$C_{q,t}^z$
x_q
$\mathcal{D}_{seq}$
$\mathcal{D}_{seq}^1$
$\mathcal{D}_{seq}^2$
$\mathcal{D}_{seq}^z = \{\mathcal{D}_{C_{q,t}}, \mathcal{D}_{\tau_{\chi t}}\}$
$\mathcal{D}_{seq}^{n_c}$
$\mathcal{D}_{C_{q,t}}$
$C_{q,t}^1 = C_{q,t}^{\min}$
$C_{q,t}^*$
$C_{q,t}^k = C_{q,t}^{\max}$
$\mathcal{D}_{\tau_{\chi t}}$
$x_{q,t}^1$ $x_{q,t+1}^1$
$x_{q,t}^k$ $x_{q,t+1}^k$
$\sim\tau_{\chi t}^*$

Figure 11: VQ codebook structure.

In Algorithm 2, `heap_push` and `heap_replace` adopt the conventional heap space management rule, with a computational complexity of $\mathcal{O}(\log k)$. The difference is that we additionally store the sequence information in $\mathcal{D}_{\tau_{\chi_t}}$ according to their $C_{q,t}$ values.

Algorithm 2 Update Sequence Buffer $\mathcal{D}_{seq}$

```
1: $\mathcal{D}_{seq}$ keep top $k$ trajectory sequences based on their $C_{q,t}$
2: **Input:** A total reward sum $R_t$ and a sequence $\tau_{\chi_t}$
3: Get an initial index $z_t \leftarrow \tau_{\chi_t}[0]$
4: Get $\mathcal{D}_{C_{q,t}}, \mathcal{D}_{\tau_{\chi_t}}$ from $\mathcal{D}_{seq}^{z_t}$
5: **if** $|\mathcal{D}_{C_{q,t}}| < k$ **then**
6:    heap_push($\mathcal{D}_{C_{q,t}}, \mathcal{D}_{\tau_{\chi_t}}, C_{q,t}, \tau_{\chi_t}$)
7: **else**
8:    $C_{q,t}^{\min} \leftarrow \mathcal{D}_{C_{q,t}}[0]$
9:    **if** $R_t > C_{q,t}^{\min}$ **then**
10:      heap_replace($\mathcal{D}_{C_{q,t}}, \mathcal{D}_{\tau_{\chi_t}}, R_t, \tau_{\chi_t}$)
11:    **end if**
12: **end if**
```

E. Implementation Details

In this section, we present further details of the implementation for LAGMA. Algorithm 3 presents the pseudo-code for a timestep dependent indexing $\mathcal{J}(t)$ used in Eq. (5). The purpose of a timestep dependent indexing $\mathcal{J}(t)$ is to distribute the quantized vectors throughout an episode. Thus, Algorithm 3 tries to uniformly distribute quantized vectors according to the maximum batch time of an episode. By considering the maximum batch time of each episode, Algorithm 3 can adaptively distribute quantized vectors.

Algorithm 3 Compute $\mathcal{J}(t)$

```
1: **Input:** For given the maximum batch time $T$, the number of codebook $n_c$, and the current timestep $t$
2: **if** $t == 0$ **then**
3:    $d = \lfloor n_c/T \rfloor$
4:    $r = n_c \bmod T$
5:    $i_s = d_n \times T$
6:    Keep the values of $d, r, i_s$ until the end of the episode
7: **end if**
8: **if** $d \geq 1$ **then**
9:    $\mathcal{J}(t) = d \times t : 1 : d \times (t+1)$
10:    **if** $t < r$ **then**
11:      Append $\mathcal{J}(t)$ with $i_s + t$
12:    **end if**
13: **else**
14:    $\mathcal{J}(t) = \lfloor t \times n_c/T \rfloor$
15: **end if**
16: return $\mathcal{J}(t)$
```

For given the maximum batch time $t_{\max}$ and the number of codebook n_c, `Line#4` and `Line#5` in Algorithm 3 compute the quotient and remainder, respectively. `Line#8` compute an array with increasing order starting from the index $d \times t$ to $d \times (t+1)$. `Line#10` additionally designate the remaining quantized vectors to the early time of an episode.

Algorithm 4 presents training algorithm to update encoder f_ϕ, decoder f_ψ, and quantized embeddings $\boldsymbol{e}$ in VQ-VAE. In Algorithm 4, we also present a separate update for $\mathcal{D}_{VQ}$, which estimates the value of each quantized vector in VQ-VAE. In addition, the overall training algorithm including training for VQ-VAE is presented in Algorithm 5.

Algorithm 4 Training algorithm for VQ-VAE and $\mathcal{D}_{VQ}$

```
**Parameter:** learning rate $\alpha$ and batch-size $B$
**Input:** $B$ sample trajectories $[\mathcal{T}]_{i=1}^{B}$ from replay buffer $\mathcal{D}$, the current episode number $n_{\text{epi}}$, an update interval $n_{\text{freq}}^{vq}$ for VQ-VAE and $n_{\text{freq}}^{cd}$ for $\mathcal{D}_{VQ}$ update interval.
**for** $t = 0$ **to** $T$ **do**
  **for** $i = 1$ **to** $B$ **do**
    **if** mod$(n_{\text{epi}}, n_{\text{freq}}^{cd})$ **then**
      Get $R_t^i$ and update $\mathcal{D}_{VQ}$ with Eq. (7).
    **end if**
    **if** mod$(n_{\text{epi}}, n_{\text{freq}}^{vq})$ **then**
      Get current state $s_t \sim [\mathcal{T}]_{i=1}$ and compute $\mathcal{J}(t)$ via Algorithm 3
      Compute $\mathcal{L}_{VQ}^{tot}$ via Eq. (6) with $f_\phi$, $f_\psi$, and $\boldsymbol{e}$.
    **end if**
  **end for**
**end for**
**if** mod$(n_{\text{epi}}, n_{\text{freq}}^{vq})$ **then**
  Update $\phi \leftarrow \phi - \alpha \frac{\partial \mathcal{L}_{VQ}^{tot}}{\partial \phi}$, $\psi \leftarrow \psi - \alpha \frac{\partial \mathcal{L}_{VQ}^{tot}}{\partial \psi}$, $\boldsymbol{e} \leftarrow \boldsymbol{e} - \alpha \frac{\partial \mathcal{L}_{VQ}^{tot}}{\partial \boldsymbol{e}}$
**end if**
```

Algorithm 5 Training algorithm for LAGMA.

```
1: **Parameter:** Batch size $B$ and the maximum training time $T_{env}$
2: **Input:** $Q_\theta^i$ is individual Q-network of $n$ agents, replay buffer $\mathcal{D}$, extended VQ codebook $\mathcal{D}_{VQ}$, and sequence buffer $\mathcal{D}_{seq}$
3: Initialize network parameters $\theta$, $\phi$, $\psi$, $\boldsymbol{e}$
4: **while** $t_{env} \leq T_{env}$ **do**
5:   Interact with the environment via $\epsilon$-greedy policy based on $[Q_\theta^i]_{i=1}^{n}$ and get a trajectory $\mathcal{T}$
6:   Append $\mathcal{T}$ to $\mathcal{D}$
7:   Get $B$ sample trajectories $[\mathcal{T}]_{i=1}^{B} \sim \mathcal{D}$
8:   For a given $[\mathcal{T}]_{i=1}^{B}$, run MARL training algorithm and Algorithm 1 to update $\theta$ with Eq. (9), and Algorithm 4 to update $\phi$, $\psi$, $\boldsymbol{e}$ with Eq. (6)
9: **end while**
```

F. Generalizability of LAGMA

F.1. Policy robustness test

To assess the robustness of policy learned by our model, we designed tasks with the same unit configuration but highly varied initial positions, ones that agents had not encountered during training, i.e., unseen maps. With these settings, opponent agents will also experience totally different relative positions and thus will make different decisions. We set the different initial positions for this evaluation as follows.

As illustrated in Figure 12, the initial position of each task is significantly moved from the nominal position experienced during the training phase.

For the comparison, we conduct the same experiment for other baselines, such as QMIX (Rashid et al., 2018) and LDSA (Yang et al., 2022). The model by each algorithm is trained for $T_{env} = 1M$ in nominal MMM2 map (denoted as Nominal) and then evaluated under various problem settings, such as NW(hard), NW, SW, and SW(hard). Each evaluation is conducted for 5 different seeds with 32 tests and Table 5 shows the mean and variance of winrate of each case.

In Table 5, LAGMA shows not only the best performance but also the robust performance in various problem settings. The fast learning of LAGMA is attributed to the latent goal-guided incentive, which generates accurate TD-target by utilizing values of semantically similar states projected to the same quantized vector. Because LAGMA utilizes the value of semantically similar states rather than the specific states when learning Q-values, different yet semantically similar states tend to have similar Q-values, yielding generalizable policies. In this manner, LAGMA would enable further exploration, rather

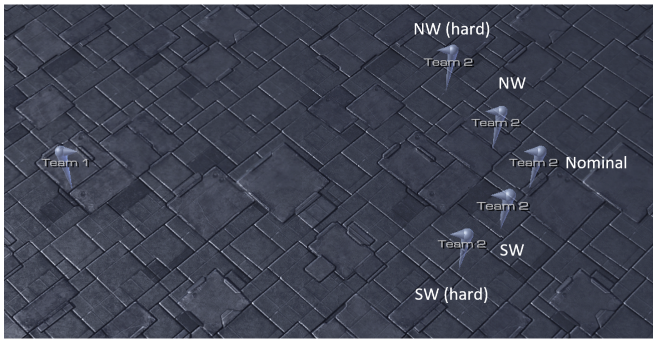


Figure 12: Problem settings for policy robustness test. Team 1 represents the initial position of RL agents, while Team 2 is the initial position of opponents.

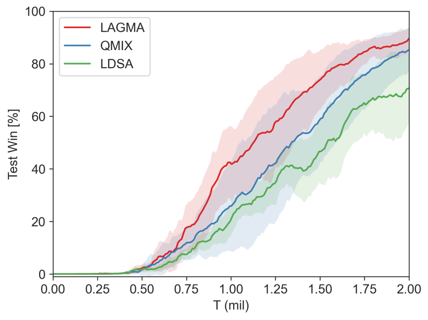
Table 5: Policy robustness test on SMAC MMM2 (super hard). All models are trained for $T_{env} = 1M$. The percentage (%) in parentheses represents the ratio compared to a nominal mean value.

	NW(hard)	NW	Nominal	SW	SW(hard)
LAGMA	0.275 ± 0.064 (28.2%)	0.500 ± 0.104 (51.3%)	0.975 ± 0.026	0.556 ± 0.051 (57.1%)	0.394 ± 0.042 (40.4%)
QMIX	0.050 ± 0.036 (13.1%)	0.138 ± 0.100 (36.1%)	0.381 ± 0.078	0.194 ± 0.092 (50.8%)	0.156 ± 0.058 (41.0%)
LDSA	0.000 ± 0.000 (0.0%)	0.081 ± 0.047 (18.3%)	0.444 ± 0.107	0.063 ± 0.049 (14.1%)	0.081 ± 0.028 (18.3%)

than solely enforcing exploitation of an identified state trajectory. Thus, even though the transition toward a goal-reaching trajectory is encouraged during training, the policy learned by LAGMA does not overfit to specific trajectories and exhibits robustness to unseen maps.

F.2. Scalability test

LAGMA can be adopted to large-scale problems without any modifications. VQ-VAE takes a global state as an input to project them into quantized latent space. Thus, in large-scale problems, only the input size will differ from tasks with a small number of agents. In addition, many MARL tasks include high-dimensional global input size as presented in Table 1 in the manuscript. To assess the scalablity of LAGMA, we conduct additional experiments in 27m_vs_30m SMAC task, whose state dimension is 1170. Figure 13 illustrates the performance of LAGMA. In Figure 13, LAGMA maintains efficient learning performance even when applied to large-scale problems, using identical hyperparameter settings as those for small-scale problems such as 5m_vs_6m.

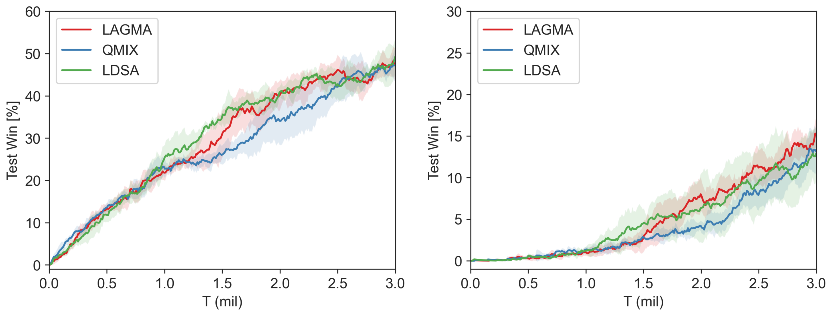


(a) Learning curve.

Figure 13: Performance on large-scale problem (27m_vs_30m SMAC).

F.3. Generalizability test to problems with diverse semantic goals

To show the generalizability of LAGMA further, we conducted additional experiments on another benchmark such as SMACv2 (Ellis et al., 2024), which includes diversity in initial positions and unit combinations within the identical task. Thus, from the perspective of latent space, SMACv2 tasks may encompass distinct multiple goals, even within the same task. For evaluation, we adopt the same hyperparameters as those for `3_vs_1WK` presented in Table 3 in the manuscript, except for $D = 4$ and $n_{\text{freq}}^{cd} = 40$.



(a) `protoss_5_vs_5` (b) `protoss_10_vs_11`

Figure 14: Performance evaluation on SMACv2.

In Figure 14, LAGMA shows comparable or better performance than baseline algorithms, but it does not exhibit distinctively strong performance, unlike other benchmark problems. We deem that this result stems from characteristics of the current LAGMA capturing reference trajectories towards a similar goal in the early training phase.

Multi-objective (or multiple goals) tasks may require a diverse reference trajectory generation. The current LAGMA only considers the return of a trajectory when storing reference trajectories in $\mathcal{D}_{seq}$. Thus, when trajectories toward different goals bifurcate from the same quantized vector, i.e., semantically similar states, they may not be captured by the current version of LAGMA algorithm if their return is relatively low compared to that of other reference trajectories already stored in $\mathcal{D}_{seq}$. Thus, LAGMA may not exhibit strong effectiveness in such tasks until various reference trajectories toward different goals are stored for a given quantized vector.

To improve, one may also consider the diversity of a trajectory when storing a reference trajectory in $\mathcal{D}_{seq}$. In addition, goal or strategy-dependent agent-wise execution would enhance coordination in such problem cases, but it may lead to delayed learning in easy tasks. The study regarding this trade-off would be an interesting direction for future research.

G. Computational cost analysis

G.1. Resource usage

The introduction of an extended VQ codebook in LAGMA requires additional memory usage to an overall MARL framework. Memory usage depends on the codebook number (n_c), the number of a reference trajectory (index sequence) to save (k) in sequence buffer $\mathcal{D}_{seq}$, its batch time length (T), the total number of data saved for moving average computation (m), and data type. Memory usage of $\mathcal{D}_{\tau_{\chi_t}}$ and $\mathcal{D}_{VQ}$ are computed as follows.

- $\mathcal{D}_{\tau_{\chi_t}}$: $\texttt{byte(dtype)} \times n_c \times k \times T$
- $\mathcal{D}_{VQ}$: $\texttt{byte(dtype)} \times n_c \times m$

For example, when $m = 100$, $n_c = 64, k = 30, T = T_{max} = 150$, i.e., the maximum timestep defined by the environment, and $\mathcal{D}_{\tau_{\chi_t}}$ and $\mathcal{D}_{VQ}$ use data type `int64` and `float32`, respectively, resource usages by introducing extended VQ codebook are computed as follows:

- $(\mathcal{D}_{\tau_{\chi_t}})_{\texttt{max}}$: $8(\texttt{int64}) \times 64 \times 30 \times 150 = \texttt{2.19MiB}$
- $\mathcal{D}_{VQ}$: $4(\texttt{float32}) \times 64 \times 100 = \texttt{25.6KiB}$

Here, $(\mathcal{D}_{\tau_{\chi_t}})_{\texttt{max}}$ value represents the possible maximum value and the actual value may vary based on the goal-reaching trajectory of each task. We can see that resource requirement due to the introduction of the extended codebook is marginal compared to that of the replay buffer and the GPU's memory capacity, such as `24GiB` in GeForce RTX3090. Note that any of these memory usages do not depend on the dimension of states since only the index (z) of the quantized vector (x_q) of a sequence is stored in $\mathcal{D}_{\tau_{\chi_t}}$.

G.2. Training time analysis

In LAGMA, we need to conduct an additional update for VQ-VAE and the extended codebook. Thus, the update frequency of VQ-VAE and the extended codebook would affect the overall training time. In the manuscript, we utilize the identical update interval $n_{\text{freq}}^{vq} = 10$, indicating training once every 10 MARL training iterations for both VQ-VAE and codebook update. Table 6 represents the overall training time taken by various algorithms for diverse tasks. GeForce RTX3090 is used for `5m_vs_6m` and GeForce RTX4090 for `8m(sparse)` and `MMM2`. In the case of `8m(sparse)` task, the training time varies according to whether the learned model finds policy achieving a common goal. Thus, the training time of the successful case is presented for each algorithm.

Table 6: Training time for each model in various SMAC maps (in hours).

Model	`5m_vs_6m` (2M)	`8m(sparse)` (3M)	`MMM2` (3M)
EMC	8.6	11.8	12.0
MASER	12.7	13.4	20.5
LDSA	5.6	11.0	9.8
LAGMA	10.5	12.6	17.7

Here, numbers in parenthesis represent the maximum training time (T_{env}) according to tasks. In Table 6, we can see that training of LAGMA does not take much time compared to existing baseline algorithms. Therefore, we can conclude that the introduction of VQ-VAE and the extended codebook in LAGMA imposes an acceptable computational burden, with only marginal increases in resource requirements.