HAVEN: Hierarchical Cooperative Multi-Agent Reinforcement Learning with Dual Coordination Mechanism

Zhiwei Xu, Yunpeng Bai, Bin Zhang, Dapeng Li, Guoliang Fan

Institute of Automation, Chinese Academy of Sciences
School of Artificial Intelligence, University of Chinese Academy of Sciences
Beijing, China
{xuzhiwei2019, baiyuanpeng2020, zhangbin2020, lidapeng2020, guoliang.fan}@ia.ac.cn

Abstract

Recently, some challenging tasks in multi-agent systems have been solved by some hierarchical reinforcement learning methods. Inspired by the intra-level and inter-level coordination in the human nervous system, we propose a novel value decomposition framework HAVEN based on hierarchical reinforcement learning for fully cooperative multi-agent problems. To address the instability arising from the concurrent optimization of policies between various levels and agents, we introduce the dual coordination mechanism of inter-level and inter-agent strategies by designing reward functions in a two-level hierarchy. HAVEN does not require domain knowledge and pre-training, and can be applied to any value decomposition variant. Our method achieves desirable results on different decentralized partially observable Markov decision process domains and outperforms other popular multi-agent hierarchical reinforcement learning algorithms.

Introduction

There has been a growing interest in multi-agent reinforcement learning (MARL) in the last few years, which plays a vital role in various tasks such as traffic control (Kuyer et al. 2008) and recommendation systems (Choi et al. 2018). Most MARL algorithms follow the paradigm known as centralized training with decentralized execution (CTDE) (Lowe et al. 2017). Each agent can utilize all available information during training but can only make decisions based on local observations. According to this principle, MARL algorithms can be divided into several categories, including those based on centralized critics and decentralized actors (Lowe et al. 2017; Foerster et al. 2018; Iqbal and Sha 2019), communication (Sukhbaatar, Szlam, and Fergus 2016; Foerster et al. 2016; Peng et al. 2017), and value decomposition (Sunehag et al. 2018; Rashid et al. 2018; Son et al. 2019; Xu et al. 2022). In a fully cooperative scenario, value decomposition methods can significantly alleviate the credit assignment issue. Numerous value decomposition variants with significant performance have been proposed recently.

However, most previous studies on multi-agent cooperative tasks do not consider hierarchical structures. Hierarchical reinforcement learning (HRL) is a computational approach that learns to operate on different levels of tem-
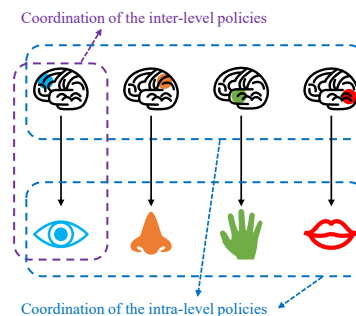
Figure 1: Inter-level and intra-level coordination in the human nervous system. We can consider the upper half of the figure as high-level policies in hierarchical reinforcement learning, and the lower half as low-level policies.

poral abstraction. Traditional HRL methods include hierarchical abstraction machine (HAM) (Parr and Russell 1997), MAXQ (Dietterich 2000), option (Sutton, Precup, and Singh 1999; Precup and Sutton 2000), and feudal architecture (Dayan and Hinton 1992). With the emergence of deep learning, HRL has gradually evolved into two distinct branches: subgoal-based (Vezhnevets et al. 2017; Nachum et al. 2018) and option-based methods (Bacon, Harb, and Precup 2017; Harb et al. 2018). Both of them have been employed in many single-agent applications.

As depicted in Figure 1, various brain regions are responsible for various organs. We view different regions of the cerebral cortex as different high-level strategies. When an individual performs a complex action, coordination between high-level strategies is required. Similarly, we see the peripheral nervous system distributed in various organs as different low-level strategies. They also need to coordinate with each other (we ignore that coordination is mainly done through the spinal cord). Besides, high-level strategies need to guide their respective low-level strategies. Therefore, when viewing the human body as a multi-agent system, we find that inter-level and intra-level coordination are crucial for solving fully cooperative tasks.

Inspired by the human nervous system, we propose a new framework for multi-agent cooperation problems, **H**ier**A**rchical **V**alue d**E**compositio**N** (HAVEN), a hierarchically structured value decomposition method. HAVEN de-

velops a two-level QMIX-style strategy and uses the advantage function of the high-level policy as part of the low-level reward. In this way, the coordination of inter-level and inter-agent policies is guaranteed. There is also no need to pre-train the low-level policies. Simultaneously, because the action space of the high-level policies is preset while maintaining the generality, domain knowledge is not required for the training process of the entire framework. Besides, it should be noted that HAVEN can be extended to any value decomposition variant. In summary, HAVEN is an end-to-end and knowledge-free framework.

Our contributions include two aspects:

- We present the HAVEN framework with a dual coordination mechanism of inter-level and inter-agent to solve the decentralized partially observable Markov decision process (Dec-POMDP) problems.
- Empirical evaluations in two testbeds, including StarCraft II (Samvelyan et al. 2019) and Google Research Football (Kurach et al. 2020), also demonstrate that our method outperforms previous algorithms by a substantial margin.

Preliminaries

Hierarchical Reinforcement Learning

Hierarchical reinforcement learning is a structured framework intended to tackle complex problems by learning to make decisions over different levels of temporal abstraction. Since most of the related work is two hierarchy levels, we focus on the two-level structures. We call the whole hierarchical system the joint policy π^{joint}, composed of the high-level policy π^h and low-level policy π^l. In the option-based hierarchical methods (Bacon, Harb, and Precup 2017; Harb et al. 2018), the action space of the high-level policy π^h is discrete, and a low-level policy π^l will be selected from a finite set of ones. For subgoal generation (Vezhnevets et al. 2017; Nachum et al. 2018; Li et al. 2021), the output space of the high-level policy π^h is often continuous. We need to calculate the intrinsic reward to guide the low-level policy π^l to make decisions based on the goals generated by π^h. High-level strategies and low-level strategies often operate at two different time scales. One straightforward form (Zhang, Yu, and Xu 2021) is that π^h runs every k step to determine the low-level policies or subgoals in the next k steps. Another more complicated approach (Rafati and Noelle 2019b) judges whether the subgoal is reached. If π^l achieves the subgoal, π^h makes a new decision and selects the next subgoal. Besides, we can set the termination function (Bacon, Harb, and Precup 2017; Harb et al. 2018) to control whether or not π^h makes a new decision.

Value Decomposition Methods in Dec-POMDPs

In this paper, we consider a fully cooperative multi-agent task that can be modelled by a Dec-POMDP (Oliehoek and Amato 2016), which can be represented by the tuple $G = \langle S, U, A, P, r, Z, O, n, \gamma \rangle$. At each time step, each agent $a \in A := \{1, \ldots, n\}$ selects the corresponding action $u^a \in U$ with only having access to the local observation $z^a \in Z$ obtained by $O(s, a) : S \times A \to Z$, where

$s \in S$ is the real state of the environment. The joint action of all agents is defined as $\boldsymbol{u} \in \boldsymbol{U}$. The environmental dynamics, also known as the state transition function, is written as $P(s' \mid s, \boldsymbol{u}) : S \times \boldsymbol{U} \times S \to [0, 1]$. In Dec-POMDPs, all agents share a reward function: $r(s, \boldsymbol{u}) : S \times \boldsymbol{U} \to \mathbb{R}$. $\gamma \in [0, 1)$ is the discount factor. The goal of the multi-agent reinforcement learning problem in the Dec-POMDP is to maximize the discounted return $\sum_{j}^{\infty} \gamma^j r_{t+j}$.

An essential concept for multi-agent value decomposition methods is decomposability. Specifically, the overall and individual interests in the multi-agent system are consistent. This assumption can be formulated as Individual-Global-Max (IGM) (Son et al. 2019), which assumes that the optimality of each agent $\arg\max_{u^a} Q_a(\tau^a, u^a)$ is consistent with the optimality of all agents $\arg\max_{u^a} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u})$. The equation that describes IGM is as follows:

$$\arg\max_{u^a} Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}) = \arg\max_{u^a} Q_a(\tau^a, u^a), \quad \forall a \in A,$$

where $\boldsymbol{\tau} \in T^n$ represents the joint action-observation histories of all agents, Q_{tot} is the global action-value function, and Q_a is the individual one. Many variants of value decomposition have been developed and HAVEN can be applied to these methods.

Method

This section introduces the proposed novel multi-agent hierarchical reinforcement learning framework HAVEN. We first describe the entire process of HAVEN for interacting with the environment and then elaborate on its structure and implementation. Finally, we give the loss functions.

The HAVEN Work Process

In multi-agent systems, each agent a has a high-level policy $\pi^{h,a}$ and a low-level policy $\pi^{l,a}$. And the corresponding action spaces are the macro action space $u^{h,a} \in U^h$ and the primitive action space $u^{l,a} \in U^l$. We define the macro action space U^h as N one-hot variables in this paper so that the output space of $\pi^{h,a}$ is discrete. $\boldsymbol{\pi}^h = \{\pi^{h,1}, \ldots, \pi^{h,n}\}$ represents the high-level joint policy of all agents, and $\boldsymbol{\pi}^l = \{\pi^{l,1}, \ldots, \pi^{l,n}\}$ denotes the low-level one.

HAVEN uses a two-timescale framework, faster for the low-level policy, and slower by a factor of k for the high-level policy. So we define T and t as the time scales of
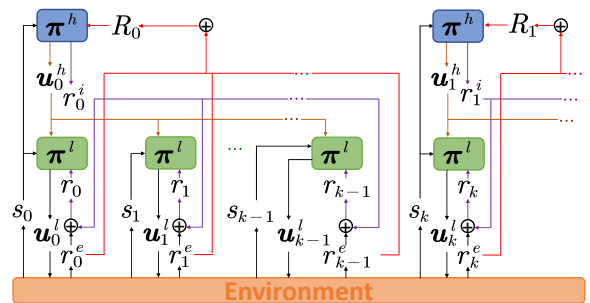

Figure 2: The workflow of HAVEN over an episode. The purple lines and the red lines represent the calculation processes of the reward function of $\boldsymbol{\pi}^l$ and $\boldsymbol{\pi}^h$, respectively.

the high-level policy and the low-level one, respectively. We carry out $\boldsymbol{\pi}^h$ every k steps at the slow time scale. After $\boldsymbol{\pi}^h$ selects the joint macro action $\boldsymbol{u}^h$, $\boldsymbol{\pi}^l$ will select the joint primitive action $\boldsymbol{u}^l$ depending on the local observation $\boldsymbol{z}$ for k steps. In Dec-POMDPs, all agents share a reward function given by environments and we denote it as the external reward r^e of $\boldsymbol{\pi}^l$. We also set the high-level reward function to be shared, defined as $R_T = \sum_{i=0}^{k-1} r^e_{T \cdot k + i}$. We denote the replay buffers of the both level policies as $\mathcal{D}^l$ and $\mathcal{D}^h$ respectively, and the stored trajectories correspond to $\langle s_t, \boldsymbol{z}_t, \boldsymbol{u}^h_{\lfloor t/k \rfloor}, \boldsymbol{u}^l_t, r^e_t \rangle$ and $\langle s_T, \boldsymbol{z}_T, \boldsymbol{u}^h_T, R_T \rangle$.

For the sake of the concurrent optimization of policies at both levels, we adopt the advantage function of $\boldsymbol{\pi}^h$ as the intrinsic reward of $\boldsymbol{\pi}^l$. An intuitive interpretation of the intrinsic reward is that the high-level advantage function can give low-level policies the temporal abstraction of next k steps and guide them to learn skills. When $\boldsymbol{\pi}^h$ performs the joint action $\boldsymbol{u}^h_T$ in state s_T, we set the advantage function for $\boldsymbol{u}^h_T$ as $A_h(s_T, \boldsymbol{u}^h_T)$. Then for $\boldsymbol{\pi}^l$, the advantage function A_h is evenly divided among k steps to get the intrinsic reward of each low-level step, which can be expressed as:

$$r^i_t = \frac{A_h(s_T, \boldsymbol{u}^h_T)}{k}, \qquad T \cdot k \le t < (T+1) \cdot k. \quad (1)$$

r^i links the strategies of different levels together. R and r^e act as joint reward functions between all agents. They respectively represent the coordination of the inter-level and inter-agent policies. Furthermore, according to Eq. (1), the advantage-based intrinsic reward r^i does not change in a k-step time interval. Equal rewards over a period of time can cause lower-level policies to suffer from temporal credit assignment problems. Agents need to know the actual feedback of the chosen action at each low-level step. Therefore, we get the linear combination of the external reward and the intrinsic reward simply to obtain the reward function $r = r^e + r^i$ of the low-level joint policy $\boldsymbol{\pi}^l$. The whole workflow of HAVEN is shown in Figure 2. It should be noted that under the off-policy setting, the intrinsic reward is calculated during training, which means that r^i is recalculated every time after sampling previous transitions from the replay buffer. So r^i corresponding to each trajectory is not fixed. The calculation methodology of the intrinsic reward represented by the purple line in Figure 2 is only for the convenience of illustration.

The HAVEN Framework

To better address the credit assignment problem in Dec-POMDPs, policies at both levels in HAVEN are QMIX-style architectures, including a shared Agent Net and a Mixing Net. For the high-level policy, at the T-th high-level step, each agent a chooses the macro action $u^{h,a}_T = \varepsilon\text{-greedy}\left(Q^h_a(\tau^{h,a}_T, u)\right)$ on the condition of the local observation z^a_T and the previous macro action $u^{h,a}_{T-1}$. After all agents select the macro action through the Macro Agent Net composed of DRQN (Hausknecht and Stone 2015), the corresponding individual macro action-values will be fused by the Macro Mixing Net to obtain the global
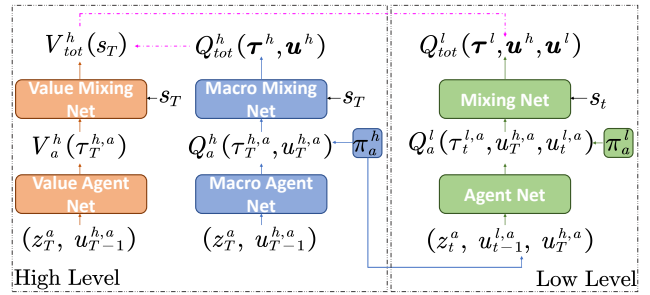

Figure 3: The overall HAVEN architecture. The left part is the high-level policy which includes $V^h(\cdot)$ and $Q^h(\cdot)$ two functions. And the right part is low-level policy which is the vanilla value decomposition architecture. The pink dashed arrows indicate that the update of one function is conditioned on another function.

macro action-value $Q^h_{tot}(\boldsymbol{\tau}^h, \boldsymbol{u}^h)$. The specific structure of the Mixing Net is determined by the basic value decomposition method. Similarly, we obtain the joint primitive action $\boldsymbol{u}^l_t$ and the global low-level action-value function $Q^l_{tot}(\boldsymbol{\tau}^l, \boldsymbol{u}^h, \boldsymbol{u}^l)$ through the low-level value decomposition structure. However, the difference is that the input of the low-level Agent Net contains the macro action $u^{h,a}_T$ given by the Macro Agent Net.

At the T-th high-level step, the advantage function can be defined as:

$$A_h(s_T, \boldsymbol{u}^h_T) = \mathbb{E}_{s_{T+1} \sim (\boldsymbol{\pi}^h, \boldsymbol{\pi}^l)} \left[R_T + \gamma V^h(s_{T+1}) - V^h(s_T) \right], \quad (2)$$

where $V^h(\cdot)$ represents the value function, which is usually estimated by the on-policy methods and gives the expected return if agents start in a certain state and always act according to the current policy $\boldsymbol{\pi}^{joint}$. Therefore, we need to add an additional neural network structure to estimate $V^h(\cdot)$. Enlightened by VDAC (Su, Adams, and Beling 2021), we calculate the local state value $V^h_a(\tau^{h,a}_T)$ of all agents and then feed them into the Value Mixing Net to finally get the global state-value function $V^h_{tot}(s_T)$. The additional neural network to approximate $V^h(\cdot)$ is indispensable. We tried the advantage function $A(s, \boldsymbol{u}) = Q_{tot}(s, \boldsymbol{u}) - \max Q_{tot}(s, \cdot)$, but it didn't work well. The possible reason is that $Q^h_{tot}(s, \boldsymbol{u}^h)$ with respect to the non-optimal action $\boldsymbol{u}^h$ is inaccurately estimated. The state-value function $V^h(s)$ does not depend on the action and therefore is trained using more experiences than a action-value function that is only updated if a specific action is selected. The overall framework of HAVEN is depicted in Figure 3. Note that although two sets of neural networks have been added, the parameters of the entire framework did not increase linearly with the number of agents due to the parameter sharing mechanism. The number of parameters is similar to other multi-agent hierarchical reinforcement learning methods.

Loss Functions

HAVEN can realize the concurrent optimization of inter-level policies, similar to the monotonic improvement of joint

policy in HAAR (Li et al. 2019). Nevertheless, the most obvious difference is that HAVEN is an off-policy value-based method so that it can achieve higher sample efficiency. The objective function which is used by general on-policy methods for learning the state-value function in Eq. 2 is given by the following equation:

$$V^h(s_T) = (1-\alpha)V^h(s_T) + \alpha(R_T + \gamma V^h(s_{T+1})), \quad (3)$$

where α is the learning rate. Eq. 3 obviously can not be directly applied to the off-policy reinforcement learning algorithm. The reason is that under the off-policy settings, the state value in Eq. 3 is estimated from the past policy and not from the target policy π^{joint} we need to optimize. To address this issue, we have modified the update formula of the state-value function. The point is that the state-value estimates is obtained under the target policy. The state value of the $T + 1$ step can be represented by the action value $\max_{u^h} Q_{tot}^h(\cdot)$ under the target policy. Enlightened by the loss function in QVMAX (Wiering and Hasselt 2009), we provide the off-policy objective function by using the $\max$ operator:

$$V^h(s_T) = (1-\alpha)V^h(s_T) + \alpha(R_T + \gamma \max_{u_{T+1}^h} Q_{tot}^h(\tau_{T+1}^h, u_{T+1}^h)).$$
$$(4)$$

R_T in Eq. 4 is obtained under the past behavior policy instead of the target policy π^{joint}. So the resulting algorithm uses a one-step trajectory of the behavior policy, which means that the state value function $V^h(\cdot)$ under the joint policy π^{joint} calculated by Eq. 4 is still different from the true state value. However, compared with Eq. 3 in which R_T and $V^h(\cdot)$ are all estimated from the past policy, our proposed modified objective function is closer to the true value under the off-policy settings.

We take the initial state value η as the optimization goal, which means we need to find a suitable joint policy π^{joint} to maximize it. The equation that describes $\eta(\pi^{joint})$ is as follows:

$$\eta(\pi^{joint}) = \mathbb{E}_{s_0^h}\left[V^h\left(s_0^h\right)\right]$$
$$= \mathbb{E}_{s_0^h, u_0^h, \dots \sim \pi^{joint}}\left[\sum_T \gamma_h^T R\left(s_T^h\right)\right]. \quad (5)$$

We can easily get that, in the case of fixed low-level policy π^l, optimizing high-level policy π^h leads to improvement in the joint policy π^{joint}. For the optimization of the low-level policy, we need to justify it theoretically. We assume that the high-level policy is fixed when we optimize the low-level policy, and use $\tilde{\pi}^{joint}$ and $\tilde{\pi}^l$ to represent the updated joint policy and the updated low-level policy. We can obtain the optimization target of $\tilde{\pi}^{joint}$ and $\tilde{\pi}^l$:

$$\eta(\tilde{\pi}^{joint}) = \eta(\pi^{joint})$$
$$+ \mathbb{E}_{(s_T^h, u_T^h) \sim \tilde{\pi}^{joint}}\left[\sum_T \gamma_h^T A_h\left(s_T^h, u_T^h\right)\right], \quad (6)$$

$$\eta\left(\tilde{\pi}^l\right) \approx \eta(\pi^{joint})$$
$$+ \left[1 + \frac{1-\gamma_l^k}{k(1-\gamma_l)}\right]\mathbb{E}_{\tau_h \sim (\tilde{\pi}_l, \pi_h)}\left[\sum_T \gamma_h^T A_h\left(s_T^h, u_T^h\right)\right]. \quad (7)$$

The proof of Eq. (7) can be found in Appendix A. When the two-level discount factors γ_l and γ_h are close to 1 and k is not large, the optimization goals of the joint policy and that of the low-level policy have both the term $\mathbb{E}_{\tau_h \sim (\tilde{\pi}_l, \pi_h)}\left[\sum_T \gamma_h^T A_h\left(s_T^h, u_T^h\right)\right]$. Meantime, since the updated policy has nothing to do with the original joint policy π^{joint}, the optimization goals of the two can be further simplified:

$$\max_{\tilde{\pi}^{joint}} \eta(\tilde{\pi}^{joint}) = \max_{\tilde{\pi}^{joint}} \mathbb{E}_{(s_t^h, u_t^h) \sim \tilde{\pi}}\left[\sum_T \gamma_h^T A_h\left(s_T^h, u_T^h\right)\right],$$

$$\max_{\tilde{\pi}^l} \eta(\tilde{\pi}^l) =$$

$$\max_{\tilde{\pi}^l}\left[1 + \frac{1-\gamma_l^k}{k(1-\gamma_l)}\right]\mathbb{E}_{\tau_h \sim (\tilde{\pi}^l, \pi^h)}\left[\sum_T \gamma_h^T A_h\left(s_T^h, u_T^h\right)\right].$$

$1 + \frac{1-\gamma_l^k}{k(1-\gamma_l)}$ is obviously a positive value, so when we maximize Eq. (7), $\eta(\tilde{\pi}^{joint})$ in Eq. (6) increases. To sum up, the joint policy π^{joint} is monotonically optimized when we monotonically optimize π^h and π^l. So the above updating scheme with the intrinsic reward avoids instability of the concurrent optimization of inter-level policies. In addition to the intrinsic reward r^i that focuses on the coordination of inter-level policies, the reward function of π^l includes the external rewards r^e, which can improve cooperation between agents through the value decomposition mechanism of the low-level policy and alleviate temporal credit assignment problems mentioned above.

From the above explanation, we can get the loss function of the three sets of neural networks: the high-level state-value network, the high-level action-value network, and the low-level action-value network. θ, ϕ, and ψ represent their parameters, respectively. Thus, the following loss function is obtained:

$$\mathcal{L}_V^h(\theta) = \left(R_T + \gamma_h \max_{u_{T+1}^h} Q_{tot}^h\left(\tau_{T+1}^h, u_{T+1}^h \mid \phi\right)\right.$$
$$\left. - V^h\left(s_T \mid \theta\right)\right)^2, \quad (8)$$

$$\mathcal{L}_Q^h(\phi) = \left(R_T + \gamma_h \max_{u_{T+1}^h} Q_{tot}^h\left(\tau_{T+1}^h, u_{T+1}^h \mid \phi^-\right)\right.$$
$$\left. - Q_{tot}^h\left(\tau_T^h, u_T^h \mid \phi\right)\right)^2, \quad (9)$$

$$\mathcal{L}_Q^l(\psi) = \left(r_t + \gamma_l \max_{u_{t+1}^l} Q_{tot}^l\left(\tau_{t+1}^l, u_{\lfloor(t+1)/k\rfloor}^h, u_{t+1}^l \mid \psi^-\right)\right.$$
$$\left. - Q_{tot}^l\left(\tau_t^l, u_{\lfloor t/k\rfloor}^h, u_t^l \mid \psi\right)\right)^2, \quad (10)$$

where ϕ^- and ψ^- refer to the parameters of the high-level and low-level action-value target network, respectively. It is worth mentioning that the optimization of the three networks is independent of each other. The implementation details and algorithmic description of HAVEN can be found in Appendix B.

Related Work

Several approaches to single-agent HRL have been proposed. One is the method based on options, which abstracts

frequently reused sub-policies into actions of the high-level policy. This approach often causes options to degenerate into primitive actions. A somewhat different approach identifies a set of representations (usually the subset of the state space or the hidden variable space) that make for useful subgoals. The output space of the high-level policy is set to the subgoal space, and the low-level policy outputs the primitive actions depending on the subgoals output by the high-level policy. Although this approach is quite similar to human decision-making, it is frequently challenging to put into practice because of the size of the subgoal space. We can also speed up the reinforcement learning process by manually setting subgoals (Rafati and Noelle 2019a; Song et al. 2019) or intrinsic rewards (Vezhnevets et al. 2017), but this inevitably introduces the domain knowledge. In addition, some methods have been proposed to solve the instability caused by the simultaneous learning of policies at both levels. For example, HAAR (Li et al. 2019) calculates advantage-based auxiliary rewards, and CHER (Kreidieh et al. 2019) collaboratively optimizes goal-assignment and goal-achievement policies from a multi-agent perspective.

In recent years, hierarchical structures have gradually been used in multi-agent reinforcement learning. Feudal Multi-agent Hierarchies (FMH) (Ahilan and Dayan 2019) applies a feudal architecture to the multi-agent environments. However, its primary flaw is that it cannot be applied to a fully-cooperative setting, in which all agents optimize a shared reward function. To address the sparse and delayed reward problem in the cooperative multi-agent situation, the hierarchical deep multi-agent reinforcement learning methods with temporal abstraction (Tang et al. 2018) such as Hierarchical QMIX and Hierarchical Communication Network were proposed. Nevertheless, the significant limitation of these methods is that the high-level action space is set manually. Hierarchical learning with skill discovery (HSD) (Yang, Borovikov, and Zha 2020) makes the skills output by the macro policy more diversified through supervised learning, which helps agents learn useful skills. However, the parameter tuning process in HSD is tedious because of a large number of hyperparameters. RODE (Wang et al. 2021) explicitly divides the action space by clustering actions. Each action subspace corresponds to a kind of "role". It is a novel idea except for the high cost of clustering.

Our proposed HAVEN is **NOT** a method that simply replaces the intrinsic reward in HSD with a HAAR-like advantage-based objective. The high-level Q-function in HSD is a QMIX architecture but the low-level one is learned by independent Q-learning. Due to the belief that low-level policies would continue to experience credit assignment issues, we suggest bi-level QMIX-style structures for HAVEN. Furthermore, the low-level reward is defined as a combination of the team reward and the intrinsic reward in HSD, where the latter is a probability value. It is difficult for the skill discovery mechanism to work if the scales of the team reward and the intrinsic reward are not essentially equivalent. However, the scales of r^e and r^i in the low-level reward in HAVEN must be the same. In HSD, the intrinsic reward is calculated when interacting with the environment and stored directly in the replay buffer. While in HAVEN the

intrinsic reward is calculated at training time, so it does not become obsolete as the policies change.

HAAR is an on-policy Actor-Critic algorithm. To improve the sample efficiency, we modify its loss function and successfully apply the advantage-based intrinsic reward to off-policy multi-agent value decomposition methods. HAAR needs to pre-train low-level skills while HAVEN does not. Simultaneously, the low-level reward in HAVEN is not just the high-level advantage function like HAAR, we also introduce external reward to alleviate the temporal credit assignment problem caused by equal intrinsic rewards in a k-step time interval. And we theoretically prove that the addition of external reward does not affect the monotonic improvement of the inter-level policies in Appendix A.

Experiments

In this section, we test our method on the StarCraft II micromanagement benchmark and the Google Research Football environment. Then by carrying out ablation studies, we show that each module that constitutes HAVEN is not redundant. We also investigate the influence of different hyperparameter settings. Finally, we visualize both level policies, which sheds further light on the role of the dual coordination mechanism. Details of the experimental setup can be found in Appendix C.

Performance on StarCraft II

We first evaluate the performance of HAVEN in the SMAC testbed and compare it with other popular baselines. SMAC is a multi-agent reinforcement learning environment based on the real-time strategy game StarCraft II. There are numerous Dec-POMDP micromanagement tasks in SMAC. The version of StarCraft II is SC2.4.6.2.69232 which is the same as some literature (Rashid et al. 2018, 2020), not the easier SC2.4.10. Performance is not always comparable between versions. To verify the validity of our method, we choose the most common method QMIX as the basic algorithm of HAVEN. Of course, HAVEN can also be built based on other value decomposition algorithms.

The implementation of HAVEN and other benchmarks in our experiment is based on Pymarl (Samvelyan et al. 2019). To make the empirical results more convincing, we compare HAVEN with state-of-the-art value decomposition approaches, including QMIX, QTRAN (Son et al. 2019), Weighted QMIX (Rashid et al. 2020), MAVEN (Mahajan et al. 2019), and following hierarchical multi-agent reinforcement learning methods:

- **RODE** (Wang et al. 2021) which decomposes joint action spaces based on the high-level role selector. RODE did some changes to the SMAC environment but did not carry over these changes to the baselines in the original paper. We test it under the original environment settings and show its real performance.

- **HSD** (Yang, Borovikov, and Zha 2020) which is a hierarchical multi-agent method with unsupervised skill discovery for strategic teamwork.

The hyperparameters of the basic algorithm in HAVEN and those of other baseline algorithms are consistent with the
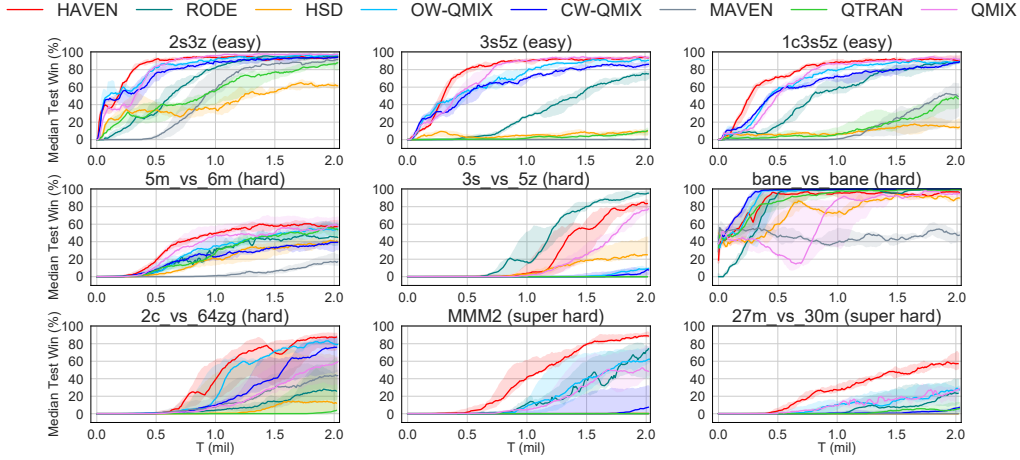
Figure 4: Performance comparison with baselines in different scenarios.
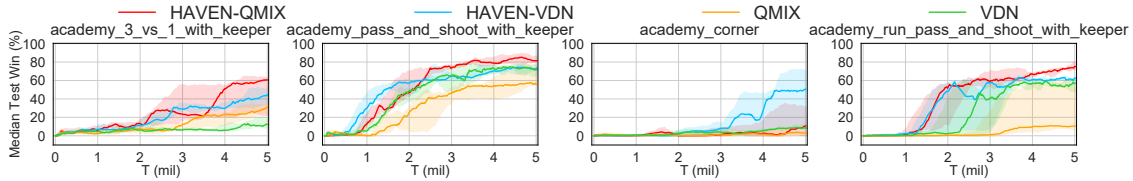

Figure 5: Comparison of our approach against baseline algorithms on Google Research Football.

original work. For the high-level time scale k and the number of macro actions N, we set them to 3 and 8, respectively.

Figure 4 shows the performance comparison between HAVEN based on QMIX and other baselines in different scenarios. The solid lines represent the median win rates, and the 25-75% percentiles are shaded. The performance of our method is significantly better than its basic algorithm QMIX and many other baselines. The role-based learning of RODE does not bring much performance improvement in our fair comparison. HSD also performs poorly and we guess it is because in the original paper HSD was only evaluated in a fully observable game. So the skill discovery mechanism in HSD may not work in a partially observable environment, even though we used DRQN. The superiority of HAVEN is more obvious in hard scenario *2c_vs_64zg*, and super hard scenarios *MMM2* and *27m_vs_30m*. In addition, HAVEN can still achieve high sample efficiency in all easy scenarios, which is difficult for some other complex value decomposition methods. The dual coordination mechanism in HAVEN significantly improves the sample efficiency, and we will analyze it further in the visualization.

Performance on Google Research Football

The Google Research Football environment is a reinforcement learning experimental platform focused on training agents to play football. We study the effectiveness and generalization of HAVEN framework in the Football Academy, which has many mini-scenarios. We tried two different basic algorithms, VDN and QMIX. These methods applied with

HAVEN are denoted as HAVEN-VDN and HAVEN-QMIX, respectively. We evaluate them and the vanilla algorithms on some official maps of Google Research Football. All other experimental settings are the same as those on SMAC.

Figure 5 shows the learning curve on four different maps. In all scenarios, both VDN and QMIX, their performance is worse than HAVEN-VDN and HAVEN-QMIX. The overall experiments show that HAVEN can be applied to different Dec-POMDP domains and extended to different value decomposition algorithms.

Ablation Studies

We also carry out some ablation studies and discuss the influence of different values of k and N. The ablation experiments include setting the low-level reward function to (1) only the intrinsic reward r^i or (2) only the external reward r^e, and (3) using the general bootstrap update formula which is described by Eq. (3) as the update formula of $V^h(\cdot)$ instead of the off-policy update mode in Eq. (4). We propose three ablations in which one of the above three components is different from the original HAVEN while the other parts remain unchanged. The above three alterations of HAVEN are called HAVEN-I, HAVEN-E, and HAVEN-B. Recall that the high-level policy is executed every k steps and the number of macro actions is N. For the k and N, we choose different values of the two to explore how they influence the performance.

Ablation experiments are carried out on two typical scenarios in SMAC, *2c_vs_64zg* and *MMM2*. From Fig-
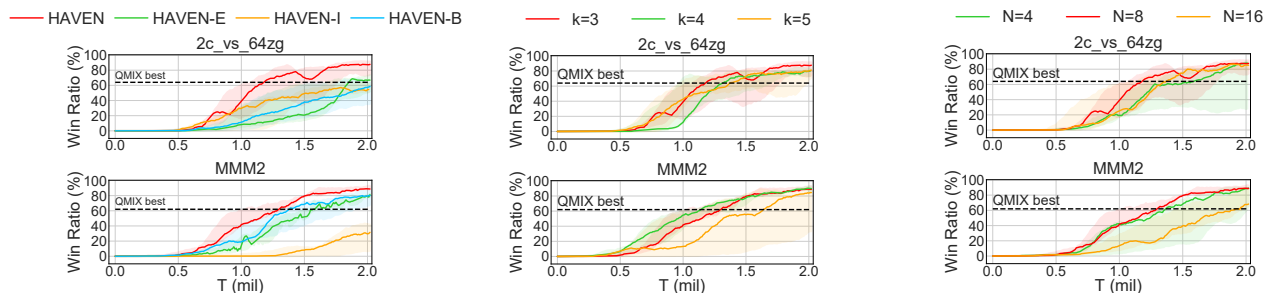
Figure 6: Left: Win rates for HAVEN and ablations. Middle: Influence of the k for HAVEN. Right: HAVEN with different N.

ure 6, it can be seen that no matter which ablations, its performance is significantly worse than that of vanilla HAVEN. Especially in scenario *MMM2*, the performance of HAVEN-I that only contains the intrinsic reward for the low-level policy is far worse than other ablations and the original framework, which means that the understanding the relationship between primitive actions and rewards greatly influences learning. So the introduction of external rewards in the low-level reward function can alleviate the above temporal credit assignment problem. Meanwhile, HAVEN-E does not perform well in *2c_vs_64zg* because of the lack of the long-term intrinsic reward calculated from high-level value functions, which can guide the learning of low-level skills. Through the above analysis, both the intrinsic reward and the external reward in the low-level reward are indispensable. Comparing the performance of HAVEN-B and original HAVEN, we can also conclude that the inaccurate state-value function estimated in an on-policy manner is harmful to learning. As mentioned above, these three components all contribute to HAVEN.

We show the influence of various hyperparameter settings on the performance of HAVEN in Figure 6. First, we discuss how the k influences the performance. The results show a trend that HAVEN performs worse as the k increases, and this phenomenon is more pronounced in *MMM2*. The experimental results match the assumption made in the previous section: k cannot be very large. Regarding the setting of the number of macro actions N, we found that N should not be too large either. We hypothesize that this is because a large N value will enlarge the high-level action space and the low-level state space.

Visualization

According to previous work, it is straightforward to know that the intra-level policies implemented by the value decomposition structure are coordinated with each other and we visualize them in Appendix D. Therefore we focus on the visualization of inter-level policies. Figure 7 shows the 2D t-SNE (van der Maaten and Hinton 2008) embedding of states of the corresponding macro action selected by agents. The density of points in a particular area reflects agents' preference for the macro action in the corresponding state. Each point is colored according to primitive actions. We set the movement-related primitive actions as warm colors and
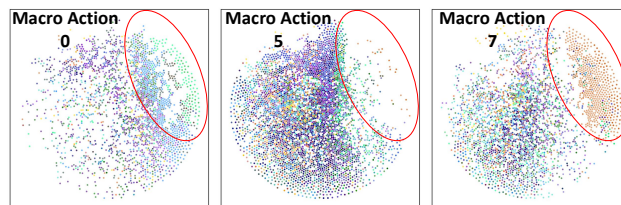

Figure 7: The 2D t-SNE embedding of states of the corresponding macro action. The colors of the points correspond to primitive actions.

attack-related actions as cool colors. It can be clearly seen that the regions in the red circles corresponding to the three different macro actions present three completely different situations. When in the states indicated by the red circle area, agents choosing Macro Action 0 are more inclined to attack, and those choosing Macro Action 7 are more likely to move. Agents rarely choose Macro Action 5 in these states. The coordination mechanism in HAVEN for inter-level policies guides lower-level policies to learn different skills for different macro actions.

Conclusion

This paper proposes a novel hierarchical off-policy value decomposition framework HAVEN, which is simple yet effective and can be applied to any value decomposition variant. The dual coordination mechanism for the simultaneous learning of inter-level and inter-agent policies also provides a solid theoretical foundation for the excellent performance of HAVEN. HAVEN does not need to set high-level action spaces manually and carry out pre-training. The experimental results show that HAVEN is robust to both easy and hard scenarios. We believe that our proposed HAVEN framework provides a general and efficient approach for multi-agent hierarchical reinforcement learning.

In our future research, we intend to concentrate on how to select the number of macro actions and whether the trained low-level policies can be transferred to other tasks. Further study of the issues would be of interest.

Acknowledgments

References

Ahilan, S.; and Dayan, P. 2019. Feudal Multi-Agent Hierarchies for Cooperative Reinforcement Learning. *ArXiv*, abs/1901.08492.

Bacon, P.; Harb, J.; and Precup, D. 2017. The Option-Critic Architecture. In Singh, S. P.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 1726–1734. AAAI Press.

Choi, S.; Ha, H.; Hwang, U.; Kim, C.; Ha, J.-W.; and Yoon, S. 2018. Reinforcement Learning based Recommender System using Biclustering Technique. *ArXiv*, abs/1801.05532.

Dayan, P.; and Hinton, G. E. 1992. Feudal Reinforcement Learning. In *NIPS*.

Dietterich, T. G. 2000. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *ArXiv*, cs.LG/9905014.

Foerster, J. N.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2137–2145.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual Multi-Agent Policy Gradients. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 2974–2982. AAAI Press.

Harb, J.; Bacon, P.; Klissarov, M.; and Precup, D. 2018. When Waiting Is Not an Option: Learning Options With a Deliberation Cost. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3165–3172. AAAI Press.

Hausknecht, M.; and Stone, P. 2015. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI Fall Symposia*.

Iqbal, S.; and Sha, F. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 2961–2970. PMLR.

Kreidieh, A. R.; Parajuli, S.; Lichtlé, N.; You, Y.; Nasr, R.; and Bayen, A. 2019. Inter-Level Cooperation in Hierarchical Reinforcement Learning. *ArXiv*, abs/1912.02368.

Kurach, K.; Raichuk, A.; Stanczyk, P.; Zajac, M.; Bachem, O.; Espeholt, L.; Riquelme, C.; Vincent, D.; Michalski, M.; Bousquet, O.; and Gelly, S. 2020. Google Research Football: A Novel Reinforcement Learning Environment. In *AAAI*.

Kuyer, L.; Whiteson, S.; Bakker, B.; and Vlassis, N. 2008. Multiagent Reinforcement Learning for Urban Traffic Control Using Coordination Graphs. In *ECML/PKDD*.

Li, S.; Wang, R.; Tang, M.; and Zhang, C. 2019. Hierarchical Reinforcement Learning with Advantage-Based Auxiliary Rewards. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 1407–1417.

Li, S.; Zheng, L.; Wang, J.; and Zhang, C. 2021. Learning Subgoal Representations with Slow Dynamics. In *ICLR*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 6379–6390.

Mahajan, A.; Rashid, T.; Samvelyan, M.; and Whiteson, S. 2019. MAVEN: Multi-Agent Variational Exploration. In Wallach, H. M.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E. B.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 7611–7622.

Nachum, O.; Gu, S.; Lee, H.; and Levine, S. 2018. Data-Efficient Hierarchical Reinforcement Learning. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 3307–3317.

Oliehoek, F.; and Amato, C. 2016. A Concise Introduction to Decentralized POMDPs. In *SpringerBriefs in Intelligent Systems*.

Parr, R. E.; and Russell, S. J. 1997. Reinforcement Learning with Hierarchies of Machines. In *NIPS*.

Peng, P.; Wen, Y.; Yang, Y.; Yuan, Q.; Tang, Z.; Long, H.; and Wang, J. 2017. Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games. *arXiv: Artificial Intelligence*.

Precup, D.; and Sutton, R. 2000. Temporal abstraction in reinforcement learning. In *ICML 2000*.

Rafati, J.; and Noelle, D. C. 2019a. Learning Representations in Model-Free Hierarchical Reinforcement Learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 10009–10010. AAAI Press.

Rafati, J.; and Noelle, D. C. 2019b. Unsupervised Methods For Subgoal Discovery During Intrinsic Motivation in Model-Free Hierarchical Reinforcement Learning. In *KEG@AAAI*.

Rashid, T.; Farquhar, G.; Peng, B.; and Whiteson, S. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv: Learning*.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J. N.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, 4292–4301. PMLR.

Samvelyan, M.; Rashid, T.; Witt, C. S. D.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J. N.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. In *AAMAS*.

Son, K.; Kim, D.; Kang, W. J.; Hostallero, D.; and Yi, Y. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, 5887–5896. PMLR.

Song, S.; Weng, J.; Su, H.; Yan, D.; Zou, H.; and Zhu, J. 2019. Playing FPS Games With Environment-Aware Hierarchical Reinforcement Learning. In Kraus, S., ed., *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 3475–3482. ijcai.org.

Su, J.; Adams, S. C.; and Beling, P. 2021. Value-Decomposition Multi-Agent Actor-Critics. In *AAAI*.

Sukhbaatar, S.; Szlam, A.; and Fergus, R. 2016. Learning Multiagent Communication with Backpropagation. In Lee, D. D.; Sugiyama, M.; von Luxburg, U.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2244–2252.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; and Graepel, T. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *ArXiv*, abs/1706.05296.

Sutton, R.; Precup, D.; and Singh, S. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell.*, 112: 181–211.

Tang, H.; Hao, J.; Lv, T.; Chen, Y.; Zhang, Z.; Jia, H.; Ren, C.; Zheng, Y.; Meng, Z.; Fan, C.; and Wang, L. 2018. Hierarchical Deep Multiagent Reinforcement Learning with Temporal Abstraction. *arXiv: Learning*.

van der Maaten, L.; and Hinton, G. E. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9: 2579–2605.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. FeUdal Networks for Hierarchical Reinforcement Learning. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 3540–3549. PMLR.

Wang, T.; Gupta, T.; Mahajan, A.; Peng, B.; Whiteson, S.; and Zhang, C. 2021. RODE: Learning Roles to Decompose Multi-Agent Tasks. *ArXiv*, abs/2010.01523.

Wiering, M. A.; and Hasselt, H. V. 2009. The QV family compared to other reinforcement learning algorithms. *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 101–108.

Xu, Z.; Bai, Y.; Li, D.; Zhang, B.; and Fan, G. 2022. SIDE: State Inference for Partially Observable Cooperative Multi-Agent Reinforcement Learning. In *AAMAS*.

Yang, J.; Borovikov, I.; and Zha, H. 2020. Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery. In *AAMAS*.

Zhang, J.; Yu, H.; and Xu, W. 2021. Hierarchical Reinforcement Learning By Discovering Intrinsic Options. *ArXiv*, abs/2101.06521.