LEARNING NEARLY DECOMPOSABLE VALUE FUNCTIONS VIA COMMUNICATION MINIMIZATION

Tonghan Wang*, Jianhao Wang*, Chongyi Zheng & Chongjie Zhang
Institute for Interdisciplinary Information Sciences
Tsinghua University
Beijing, China
wangth18@mails.tsinghua.edu.cn, chongyeezheng@gmail.com
jxwuyi@gmail.com, chongjie@tsinghua.edu.cn

ABSTRACT

Reinforcement learning encounters major challenges in multi-agent settings, such as scalability and non-stationarity. Recently, value function factorization learning emerges as a promising way to address these challenges in collaborative multi-agent systems. However, existing methods have been focusing on learning fully decentralized value function, which are not efficient for tasks requiring communication. To address this limitation, this paper presents a novel framework for learning *nearly decomposable value functions with communication*, with which agents act on their own most of the time but occasionally send messages to other agents in order for effective coordination. This framework hybridizes value function factorization learning and communication learning by introducing two information-theoretic regularizers. These regularizers are maximizing mutual information between decentralized Q functions and communication messages while minimizing the entropy of messages between agents. We show how to optimize these regularizers in a way that is easily integrated with existing value function factorization methods such as QMIX. Finally, we demonstrate that, on the StarCraft unit micromanagement benchmark, our framework significantly outperforms baseline methods and allows to cut off more than 80% communication without sacrificing the performance. The video of our experiments is available at https://sites.google.com/view/ndvf.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) are finding applications in many real-world domains, such as autonomous vehicle teams (Cao et al., 2012), intelligent warehouse systems (Nowé et al., 2012), and sensor networks (Zhang & Lesser, 2011). To help address these problems, recent years have made a great progress in MARL methods (Lowe et al., 2017; Foerster et al., 2018; Rashid et al., 2018; Jaques et al., 2019). Among these successes, the paradigm of centralized training with decentralized execution has attracted much attention for its scalability and the ability to deal with non-stationarity.

Value function decomposition methods provide a promising way to exploit such paradigm. They learn a decentralized Q function for each agent and use a mixing network to combine these local Q values into a global action value. In previous works, VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), and QTRAN (Son et al., 2019) have progressively enlarged the family of functions that can be represented by the mixing network. Despite their increasing ability in terms of value factorization representation, existing methods have been focusing on learning full decomposition, where each agent acts upon its local observations. However, many multi-agent tasks in the real world are not fully decomposable – agents sometimes require information from other agents in order to effectively coordinate their behaviors. This is because partial observability and stochasticity in a multi-agent environment can exacerbate an agent's uncertainty of other agents' states and actions during decentralized execution, which may result in catastrophic miscoordination.

*Equal Contribution.

To address this limitation, this paper presents a scalable multi-agent learning framework for learning *nearly decomposable value functions with communication*, with which agents act on their own most of the time but occasionally send messages to other agents in order for effective coordination. This framework hybridizes value function factorization learning and communication learning by introducing an information-theoretic regularizer for maximizing mutual information between decentralized Q functions and communication messages. Messages are parameterized in a stochastic embedding space. To minimize communication, we introduce an additional information-theoretic regularizer for minimizing the entropy of messages between agents. With these two regularizers, our framework implicitly learn when, what, and with whom to communicate and also ensure communication to be both *expressive* (i.e., effectively reducing the uncertainty of agents' action-value functions) and *succinct* (i.e., only sending useful and necessary information). To optimize these regularizers, we draw inspiration from variational inference and derive a tractable lower bound objective, which is easily integrated with existing value function factorization methods such as QMIX.

We demonstrate the effectiveness of our learning framework on the StarCraft II[1] unit micromanagement benchmark used in Foerster et al. (2017; 2018); Rashid et al. (2018); Samvelyan et al. (2019b). Empirical results show that it significantly outperforms baseline methods and allows to cut off more than 80% communication without sacrificing the performance. We also observe that agents can effectively learn to coordinate their actions at the cost of sending one or two bits of messages even in complex StarCraft II tasks.

2 BACKGROUND

In our work, we consider a fully cooperative multi-agent task that can be modelled by a Dec-POMDP (Oliehoek et al., 2016) $G = \langle I, S, A, P, R, \Omega, O, n, \gamma \rangle$, where $I \equiv \{1, 2, ..., n\}$ is the finite set of agents. $s \in S$ is the true state of the environment from which each agent i draw an individual partial observation $o_i \in \Omega$ according to the observation function $O(s, i)$. Each agent has a an action-observation history $\tau_i \in \mathrm{T} \equiv (\Omega \times A)^*$. At each timestep, each agent selects an action $a_i \in A$, forming a joint action $\boldsymbol{a} \in A^n$, resulting in a shared reward $r = R(s, \boldsymbol{a})$ for each agent and the next state s' according to the transition function $P(s'|s, a)$. The joint policy $\boldsymbol{\pi}$ induces a joint action-value function: $Q^{\boldsymbol{\pi}}_{tot}(s, \boldsymbol{a}) = \mathbb{E}_{s_{0:\infty}, \boldsymbol{a}_{0:\infty}}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0{=}s, \boldsymbol{a}_0{=}\boldsymbol{a}, \boldsymbol{\pi}]$, where τ is the joint action-observation history and $\gamma \in [0, 1)$ is the discount factor.

Learning the optimal action-value function encounters challenges in multi-agent settings. On the one hand, to properly coordinate actions of agents, learning a centralized action value function Q_{tot} seems a good choice. However, such a function is difficult to learn when the number of agents is large. On the other hand, directly learning decentralized action-value function Q_i for each agent alleviates the scalability problem (Tan, 1993; Tampuu et al., 2017). Nevertheless, such independent learning method largely neglects interactions among agents, which often results in miscoordination and inferior performance.

In between, value function factorization method provides a promising way to attenuate such dilemma by representing Q_{tot} as a mixing of decentralized Q_i conditioned on local information. Such methods has shown their effectiveness on complex task (Samvelyan et al., 2019b).

However, current value function factorization methods have mainly focusing on full decomposition. Such decomposition reduces the complexity of learning Q_{tot} by first learning independent Q_i and putting the burden of coordinating actions on the mixing networks whose input is all Q_i's and output Q_{tot}. For many tasks with partial observability and stochastic dynamics, mixing networks are not sufficient to learn coordinated actions, regardless of how powerful its representation ability is. The reason is that full decomposition cuts off all dependencies among decentralized action-value functions and agents will be uncertain about states and actions of other agents. Such uncertainty will increase as time goes by and can result in severe miscordination and arbitrarily worse performance during decentralized execution.

[1] StarCraft and StarCraft II are trademarks of Blizzard Entertainment™.

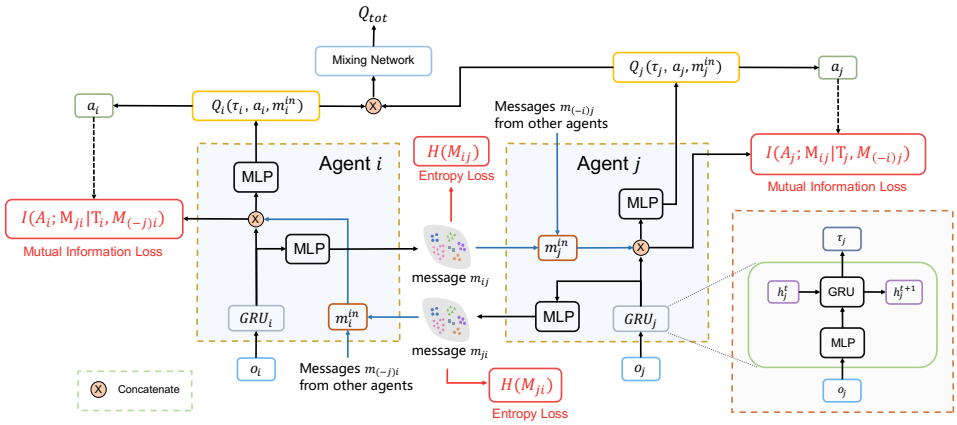


Figure 1: Schematics of our approach. The message encoder generates an embedding distribution that is sampled and concatenated with the current local history to serve as an input to the local action-value function. We introduce two regularizers, entropy loss and mutual information loss, to learn expressive and succinct messages.

3 METHODOLOTY

In this section, we propose to learn nearly decomposable value functions with communication, a new framework to overcome the miscoordination issue of full factorization methods. Our framework adopts the centralized training and decentralized execution paradigm. During the centralized training, we assume the learning algorithm has access to all agents' individual observation-action histories and global state s. During the decentralized execution, each agent acts based on its learned action-value function $Q_i(\tau_i, a_i, m_i^{in})$ conditioning on its local action-observation history τ_i and received messages m_i^{in}. Message communication is also centrally learned and executed in a decentralized manner. Our learning framework is shown in Fig. 1.

In our learning framework, individual action-value functions condition on local action-observation history and, at certain timesteps, messages from few other agents. We learn such near decomposable structure via learning minimized communication. We thus expect the communication to have the following properties:

i) Expressiveness: The message passed to one agent should effectively reduce the uncertainty in its action-value function.

ii) Succinctness: Agents are expected to send messages as short as possible to the agents who need it and when necessary.

To learn such a communicating strategy, we draw inspiration from variational inference for its proven ability in learning structure from data and endow a stochastic latent message space, which we also refer to as "message embedding". Messages are encoded as multi-variate Gaussian distributions with diagonal covariance in this latent space. Specifically, an encoder $f_m(m_i|\tau_i; \boldsymbol{\theta}_c)$ conditions on individual observation-action history is learned and shared among agents. At each timestep, this encoder outputs a message distribution M_i for each agent i, from which a message m_i^t is resampled. We adopt a point-to-point communication paradigm and $m_i^t = \langle m_{i1}^t, m_{i2}^t, ..., m_{in}^t \rangle$ where message m_{ij}, sampled from the distribution M_{ij}, is the message intended from agent i to agent j.

We impose constraints, which will be discussed in detail in the next section, on the latent message embedding to enable agent deciding locally which bits in a message should be sent (from 0 to all) according to their utility in terms of helping other agents make decision. Agent j will receive an input message m_j^{in} that has been cut, on which it conditions the local action-value function $Q_j(\tau_j, a_j, m_j^{in})$. All the individual Q values are then feed into a mixing network such as in QMIX (Rashid et al., 2018).

Apart from the constraints on the message embedding, all the components (individual action-value function, message encoder, and mixing network) are trained end-to-end by the downstream TD loss.

Thus, our overall objective is to minimize

$$\mathcal{L}(\theta) = \mathcal{L}_{TD}(\theta) + \mathcal{L}_c(\boldsymbol{\theta}_c), \tag{1}$$

where $\mathcal{L}_{TD} = \left[r + \gamma \max_{a'} Q_{tot}(s', \boldsymbol{a'}; \theta^-) - Q_{tot}(s, \boldsymbol{a}; \theta)\right]^2$ (θ^- are the parameters of a target network as in DQN) is the TD loss and θ is all parameters in the model. We now discuss how to get $\mathcal{L}_c(\boldsymbol{\theta}_c)$ to regularize the message embedding.

3.1 Minimized Communication Objective and Variational Bound

Introducing a latent variable facilitates the representation of message, but it does not mean that the messages can reduce uncertainty in action-value functions of other agents. To achieve this goal, we maximize the mutual information between message M_{ij} and optimal action policy of agent j, A_j, $I_{\boldsymbol{\theta}_c}(A_j; M_{ij}|\mathrm{T}_j, M_{(-i)j})$. However, if this is the only objective, the encoder can easily learn to cheat by giving messages under different histories a representation in different regions in the latent space, rendering cutting off useless message difficult. A natural constraint to avoid such representation is the entropy of the messages. Therefore, our objective for message of agent i is to maximize:

$$J_{\boldsymbol{\theta}_c}[M_i|\mathrm{T}_j, M_{(-i)j}] = \sum_{j=1}^{n} I_{\boldsymbol{\theta}_c}(A_j; M_{ij}|\mathrm{T}_j, M_{(-i)j}) - \beta H_{\boldsymbol{\theta}_c}(M_{ij}), \tag{2}$$

where β is a scaling factor trading expressiveness and succinctness and M_{ij} is the message distribution intended from i to j.

This objective is appealing because it agree exactly with the desiderata that we impose on the message embedding. However, applying this objective to our message embedding needs extra efforts because mutual information and entropy becomes intractable. By introducing a variational approximator, a popular technique from variational toolkit (Alemi et al., 2017), we can construct a lower bound on the mutual information term in Eq. 2:

$$\begin{aligned} &I_{\boldsymbol{\theta}_c}(A_j; M_{ij}|\mathrm{T}_j, M_{(-i)j}) \\ &\geq \mathbb{E}_{\mathbf{T}\sim\mathcal{D}, M_j^{in}\sim f_m(\mathbf{T};\boldsymbol{\theta}_c)}\left[-\mathcal{CE}\left[p(A_j|\mathbf{T}) \| q_\xi(A_j|\mathrm{T}_j, M_j^{in})\right]\right], \end{aligned} \tag{3}$$

where T_j is local action-observation history of agent j and $\mathbf{T} = \langle \mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_n \rangle$ is the joint local history sampled from the replay buffer $\mathcal{D}$, $q_\xi(A_j|\mathrm{T}_j, M_j^{in})$ is the variational posterior estimator with parameters ξ. $\mathcal{CE}$ is the cross entropy. We share ξ among agents to accelerate learning.

Next we discuss how to minimize the term $\beta H(M_{ij})$. Directly minimize the entropy of Gaussian distribution can cause the variance to collapse to 0. To deal with this possible numeric issue, we set the covariance to unit matrix and minimize $H(M_{ij}) - H(M_{ij}|\mathrm{T}_i)$ instead, where T_i is the random variable of local history of agent i. This is equivalent because $H(M_{ij}|\mathrm{T}_i)$ is a multivariate Gaussian random variable whose entropy is a constant ($\log(\det(2\pi e\mathbf{\Sigma}))/2$, where Σ is the unit matrix). Then we have:

$$H(M_{ij}) - H(M_{ij}|\mathrm{T}_i) = \int p(m_{ij}|\tau_i) p(\tau_i) \log \frac{p(m_{ij}|\tau_i)}{p(m_{ij})} dm_{ij} d\tau_i. \tag{4}$$

We use the same technique as in the mutual information term – introducing an variational inference distribution $r(m_{ij})$ to approximate $p(m_{ij})$, and we can get:

$$\begin{aligned} H(M_{ij}) - H(M_{ij}|\mathrm{T}_i) &\leq \int p(m_{ij}|\tau_i) p(\tau_i) \log \frac{p(m_{ij}|\tau_i)}{r(m_{ij})} dm_{ij} d\tau_i \\ &= \mathbb{E}_{\mathrm{T}_i\sim D}\left[D_{\mathrm{KL}}(p(M_{ij}|\mathrm{T}_i) \| r(M_{ij}))\right]. \end{aligned} \tag{5}$$

This bound holds for any distribution $r(M_{ij})$. To facilitate cutting off messages, we use unit Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. Combining Eq. 3 and 5, we get a tractable variational lower bound of our objective in Eq. 2:

$$\begin{aligned} &J_{\boldsymbol{\theta}_c}[M_i|\mathrm{T}_j, M_{(-i)j}] \\ &\geq \mathbb{E}_{\mathbf{T}\sim\mathcal{D}, M_j^{in}\sim f_m(\mathbf{T};\boldsymbol{\theta}_c)}\left[-\mathcal{CE}\left[p(A_j|\mathbf{T}) \| q_\xi(A_j|\mathrm{T}_j, M_j^{in})\right] - \beta D_{\mathrm{KL}}(p(M_{ij}|\mathrm{T}_i) \| r(M_{ij}))\right]. \end{aligned} \tag{6}$$

We optimize this bound to get a expressive and succinct message embedding. Specifically, we minimize:

$$\mathcal{L}(\boldsymbol{\theta}_c) = \mathbb{E}_{\mathbf{T}\sim\mathcal{D}, M_j^{in}\sim f_m(\mathbf{T};\boldsymbol{\theta}_c)}\left[\mathcal{CE}\left[p(A_j|\mathbf{T})\|q_\xi(A_j|\mathrm{T}_j, M_j^{in})\right] + \beta D_{\mathrm{KL}}(p(M_{ij}|\mathrm{T}_i)\|r(M_{ij}))\right]. \tag{7}$$

Intuitively, the first term, which we call the expressiveness loss, guarantees that messages sent can reduce the uncertainty in action-value functions of other agents. The second term, called succinctness loss, forces messages to get close to unit Gaussian distribution. Since we set covariance of message embedding to unit matrix, this term actually pushes the mean of messages to the origin of the latent space. Use this two losses leading to an embedding where useless messages distributes all near the origin, while messages that can contains information important for decision processes of other agents occupy other spaces.

Note that loss in Eq. 7 are used to update parameters in message encoder. In the meantime, all the components (individual action-value function, message encoder, and mixing network) are trained end-to-end by the TD loss as in QMIX. Thus, the message encoder $f_m(M_i|\mathrm{T}_i;\boldsymbol{\theta}_c)$ is updated guided by two gradients: stochastic gradient given by Eq. 6 and the gradient associated with TD loss.

3.2 Cutting Off Messages

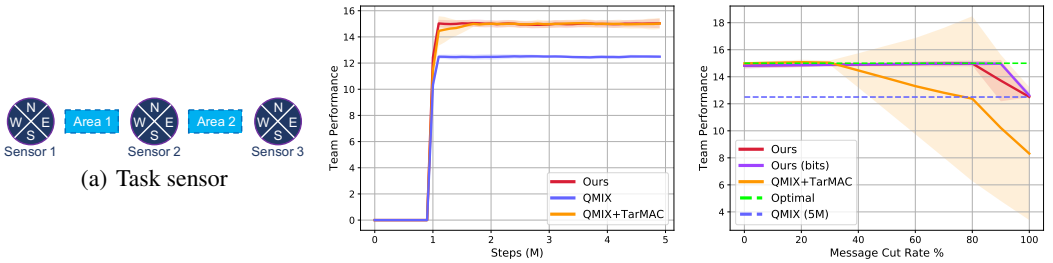
Our objective compresses messages which can not reduce the uncertainty in action-value functions of other agents to the origin of the latent message space. This naturally gives us a hint to drop out meaningless messages – we can order the message distributions according to their means and drop accordingly. Note that since we adopt diagonal covariance for our message embedding, bits in a message are independent. We thus make decision bit by bit and send messages with various length. In this way, such method not only finds out when to communicate (agent keep silence when all bits are cut), but also what to send (how many bits are sent and their values). Implementation details are discussed in Appendix B.

4 Related Works

Deep multi-agent reinforcement learning has witnessed vigorous progress in recent years. COMA (Foerster et al., 2018), MADDPG (Lowe et al., 2017), and PR2 (Wen et al., 2019) explores multi-agent policy gradients and respectively address the problem of credit assignment, learning in mixed environments and recursive reasoning. Another line of research focuses on value-based multi-agent RL, among which value-function factorization is the most popular method. Three representative examples: VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), and QTRAN (Son et al., 2019) gradually increase the representation ability of the mixing network. In particular, QMIX (Rashid et al., 2018) stands out as a scalable and robust algorithm and achieves state-of-the-art results on StarCraft unit micromanagement benchmark (Samvelyan et al., 2019b).

Communication is a hot topic in multi-agent reinforcement learning. End-to-end learning with differentiable communication channel is a popular approach now. Sukhbaatar et al. (2016); Hoshen (2017); Jiang & Lu (2018); Singh et al. (2019); Das et al. (2019) focus on learning decentralized communication protocol and address the problem of when and who to communicate. Foerster et al. (2016); Das et al. (2017); Lazaridou et al. (2017); Mordatch & Abbeel (2018) study the emergence of natural language in the context of multi-agent learning. IC3Net (Singh et al., 2019) learns gate to control the agents to only communicate with their teammates in mixed multi-agent environment. Zhang & Lesser (2013); Kim et al. (2019) study action coordination under limited communication channel and thus are also related to our works. The difference lies in that they do not explicitly minimize communication. Social influence (Jaques et al., 2019) and InfoBot (Goyal et al., 2019) penalize message that has no effect on policies of other agents.

The most related work to this paper is TarMAC (Das et al., 2019), where attention mechanism is used to differentiate the importance of incoming messages. In comparison, we use variation inference to decide the content of messages and whether a message should be sent under the guidance of global reward signals. We compare our method with TarMAC and a baseline combining TarMAC and QMIX in our methods.



(a) Task sensor (b) Performance comparison (c) Performance vs message drop rate

Figure 2: (a) Task **sensor**; (b) Performance comparison on sensor; (c) Performance comparison when different percentages of messages are dropped. We measure the drop rate of our method on two ways: count by number of message (our method) or count by bit (our method (bits)). QMIX (5M) is the performance of QMIX after training for 5M time steps.

StarCraft unit micromanagement has attracted lots of research interests for its high degree of control complexity and environmental stochasticity. Usunier et al. (2017) and Peng et al. (2017) study this problem from a centralized perspective. In order to facilitate decentralized control, we use the setup introduced by Samvelyan et al. (2019b), which is the same in Foerster et al. (2017; 2018) and Rashid et al. (2018).

5 Experimental Results

In this section, we show our experiments to answer the following questions: (i) Is miscoordination problem of full value function factorization methods widespread? (ii) Can our method learn the minimized message required by the task? (iii) Can the learnt message distribution reduce uncertainty in value functions of other agents? (iv) How does our method differ from communication via attention mechanism? (v) How do β influence the communication protocol? We will first show three simple examples to clarify our idea and then provide performance analysis on StarCraftII unit micromanagement task. For evaluation, all experiments are carried out with 3 random seeds and results are shown with 95% confidence interval. Videos of our experiments on StarCraft II are available online[2].

5.1 Baselines and Ablations

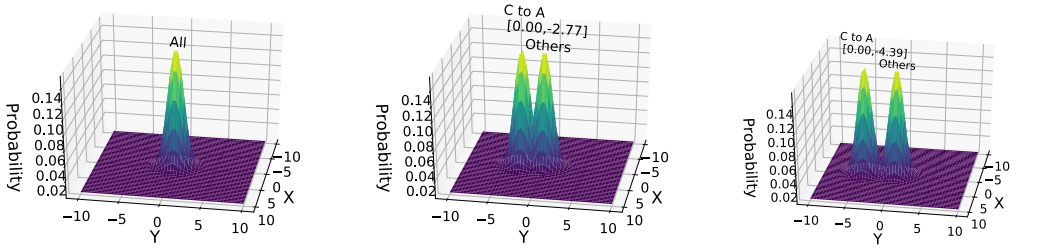
We compare our method with various baselines and ablations (i) QMIX (Rashid et al., 2018). We use QMIX as representation for full factorization method; (ii) TarMAC (Das et al., 2019). TarMAC is a state-of-the-art attentional communication method. Comparison with it can illustrate the difference between our method and attention mechanism. (iii) TarMAC+QMIX. Directly compare with TarMAC is perhaps unfair because it is not designed for tasks with shared rewards. Therefore, we integrate the communication component of TarMAC into QMIX and compare with this strong baseline.

5.2 Didactic Examples

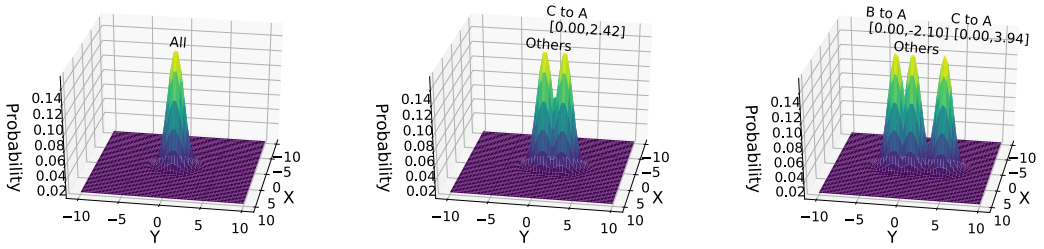
We first demonstrate our idea on a set of didactic examples.

Sensor network is a frequently used testbed in multi-agent learning field (Kumar et al., 2011; Zhang & Lesser, 2011). We use a 3-chain sensor configuration in task **sensor**. Each sensor is controlled by one agent and they are rewarded for successfully locating targets, which requires two sensors to scan the same area simultaneously when the target appears. At each timestep, target 1 appears in area 1 with possibility 1 and locating it induces a team reward of 20; target 2 appears with probability 0.5 in area 2 and agents are rewarded 30 for locating it. Agents can observe whether a target is present

[2] https://sites.google.com/view/ndvf

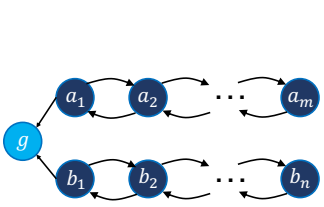


(a) Target 2 present, β=1 (b) Target 2 present, β=10^{-3} (c) Target 2 present, β=10^{-5}

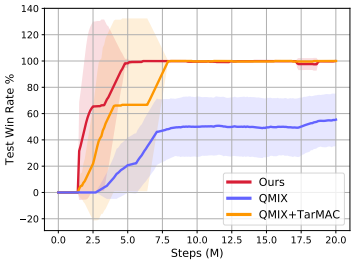


(d) Target 2 absent, β=1 (e) Target 2 absent, β=10^{-3} (f) Target 2 absent, β=10^{-5}

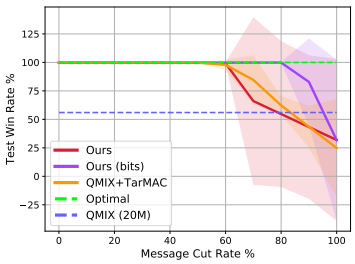
Figure 3: Message distribution learnt by our method under different values of β. (Message is cut by bit, if its $\mu < 2.0$). When $\beta = 10^{-3}$, our method learns the strategy where agent 3 sends a bit to agent 1 to indicate whether target 2 appears while messages intended between any other pairs of agents is unit Gaussian, which is the minimized communication strategy.



(a) Task hallway

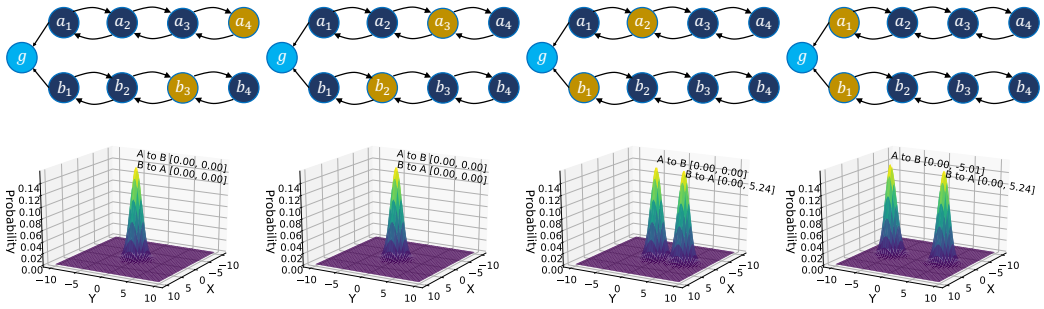


(b) Performance comparison



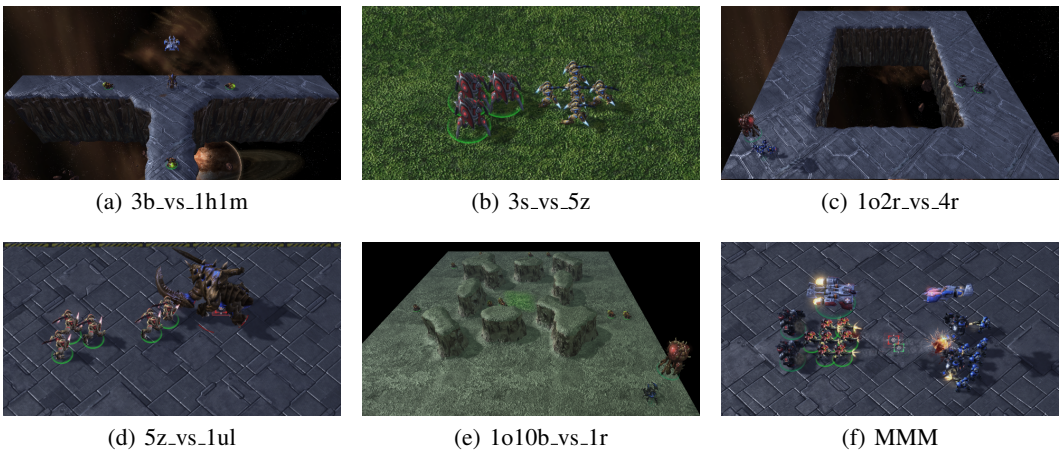
(c) Performance vs message drop rate

Figure 4: Results on **hallway**. (a, b) Task hallway and performance comparison. (c) Similar to Fig. 2(c), we show performance comparison when different percentages of messages are dropped.



(a) t=1 (b) t=2 (c) t=3 (d) t=4

Figure 5: Message embedding learnt by our method. 0 means that bit is below the cutting threshold (μ=3) and is not sent.



(a) 3b_vs_1h1m (b) 3s_vs_5z (c) 1o2r_vs_4r

(d) 5z_vs_1ul (e) 1o10b_vs_1r (f) MMM

Figure 6: Snapshots of the StarCraft II scenarios that we consider.

in nearby areas and need to take one of five actions: scanning `north`, `east`, `south`, `west`, and `noop`. Every scan induces a cost of -5.

In the optimal policy, when the target 2 appears, sensor 1 should turn itself off while sensor 2 and 3 are expected to scan area 2 to get the reward. And when target 2 is absent, sensor 1 and 2 need to cooperatively scan area 1 while sensor 3 take `noop`.

Sensor is representative of a class of tasks where the uncertainty about the true state causes policies learned by full value function factorization method to be sub-optimal – sensor 1 has to know whether the target is present in area 2 to make decision. However, mixing network of QMIX cannot provide such information. As a result, QMIX learners converge to a sub-optimal policy which gets a team reward of 12.5 on average per step (see Fig. 2(b)).

We are particularly interested in whether our method can learn the minimized communication strategy. Fig. 3(a) - 3(f) shows the latent message space learned by our method. Using our method and set β to 10^{-3}, agent 3 learns to send a bit to tell agent 1 whether target 2 appears. In the meantime, the latent message distribution between any other pair of agents is close unit Gaussian and thus is dropped. This results indicates that our method has learnt the minimized conditional graph and can explain why our method can still perform optimally when 80% messages are cut off (Fig. 2(c)). When β becomes too large (1.0), all the message bits are pushed below cutting threshold (Fig. 3(a) and 3(d)); and when β is too small (10^{-5}), our method puts more attention on reducing uncertainty in Q-functions in stead of compressing messages, so both agent 3 and agent 2 pass message to agent 1 (Fig. 3(c) and 3(f)).

The second example, called **Hallway** (Fig. 4(a)) is a Dec-POMDP with two agents randomly starting at states a_1 to a_m and b_1 to b_n respectively. Agents can observe its own position and chooses to move left, move right, or keep still at each timestep. Agents win if they arrive at state g simultaneously. Otherwise, if any agent arrives at g earlier than the other, they receive no reward and the next episode begins. Winning reward is set to 10 and the horizon is set to $max(m, n) + 10$ to avoid infinite loop.

Hallway aims to show that the miscoordination problem of full factorization method can be severe in multi-step scenarios. We show results in Fig. 4(b) where m=n=4 and our method outperforms all the baselines. We are again particularly interested in the message embedding space learnt by our method. We show an episode in Fig 5. Two agents begin at a_4 and b_3 respectively. They first move left silently (t=1 and t=2) until agent B arrives b_1. On arriving b_1, it send a bit whose value is 5.24 to A. After sending this bit, B stays at b_1 until it receives a bit from A indicating that A has arrived at a_1. After that, they move left together and win. This is the minimized communication strategy and our method can therefore still win in 100% episodes when 80% communicating bits are dropped 4(c).

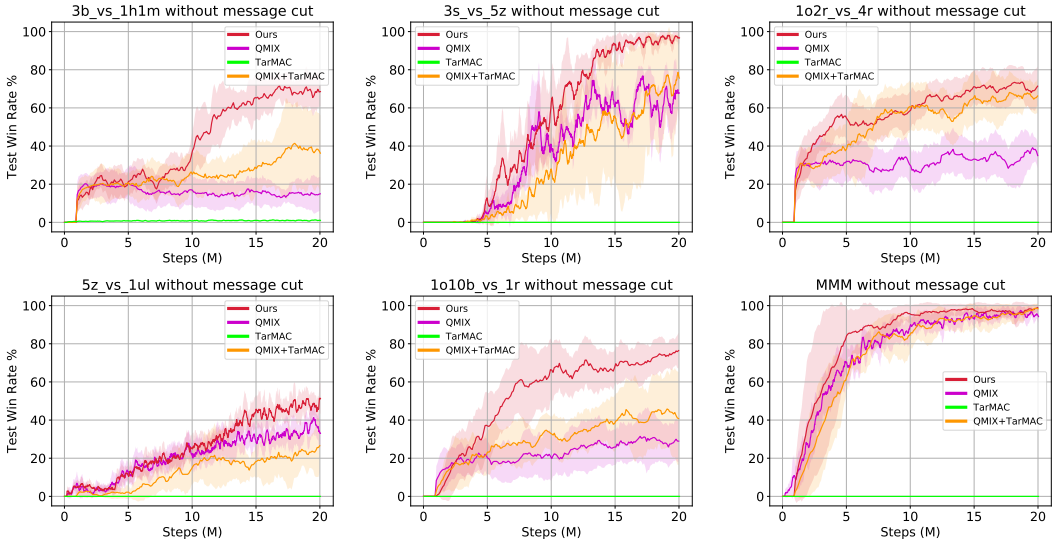


Figure 7: Learning curves of our method and baselines on all scenarios during 20M timesteps when no message is cut for our method and QMIX+TarMAC.

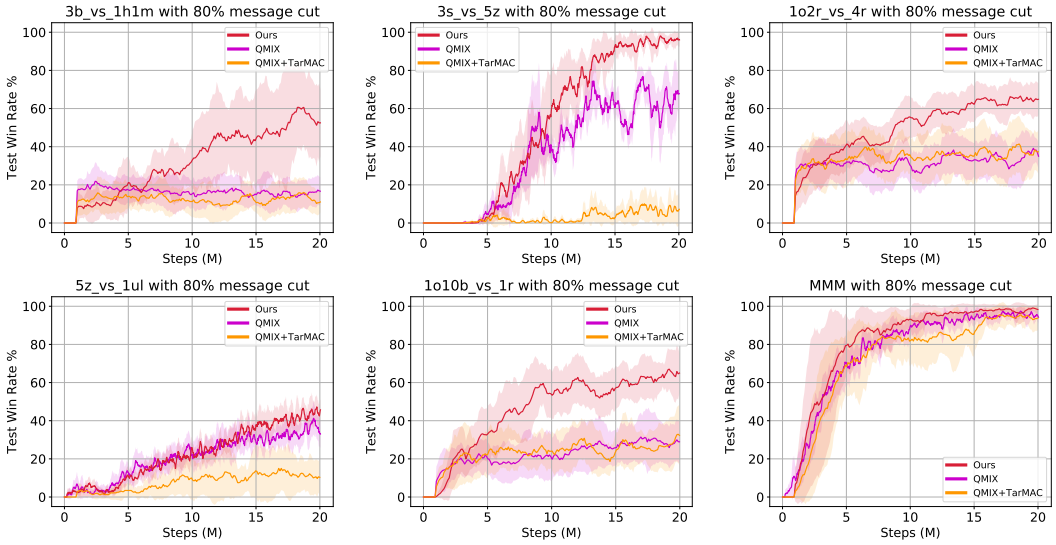


Figure 8: Performance of our method and TarMAC+QMIX when 80% messages are cut off. We also plot the learning curves of QMIX for comparison.

The third task, **independent search**, aims to demonstrate that our method can learn not to communicate in scenarios where agents are independent. Task description and results analysis are deferred to Appendix C.1.

5.3 Maximum Value Function Factorization in StarCraftII

To demonstrate that the miscoordination problem is widespread in multi-agent scenarios, we apply our method and ablations to decentralized StarCraft II micromanagement benchmark. We use the setup introduced by Samvelyan et al. (2019b) and consider combat scenarios. We describe the settings in detail in Appendix C.2.

We further increase the difficulty of action coordination in these tasks by i) reducing the sight range of the agents from 9 to 2; ii) introducing challenging maps that has complex terrain or with highly

random spawning positions for units. We compare and analysis our method on the six maps shown in Fig. 6: 3b_vs_1h1m, 3s_vs_5z, 1o2r_vs_4r, 5z_vs_1ul, MMM, and 1o10b_vs_1r. 3s_vs_5z and MMM are the same as in Samvelyan et al. (2019b) but with a narrow sight range. Detailed descriptions of these scenarios are provided in Appendix C.2.

5.3.1 PERFORMANCE COMPARISON

Across all StarCraft II scenarios, we use the same set of hyper-parameters: the length of message m_{ij} is set to 3 and β is set to 10^{-3}. To evaluate the learned policy, we pause training every 100k environment steps and run 48 testing episode. We show the performance when no message is cut of our method and baselines in Fig. 7.

Superior performance of methods with communication over QMIX demonstrates that the misco-ordination problem of full factorization method is widespread, especially in scenarios with high stochasticity, such as 1o2r_vs_4r, 3b_vs_1h1m, and 1o10b_vs_1r. Notably, our method outperforms attentional communication mechanism by a large margin. Since both of these two methods use the TD-error to update parameters, these results highlights the role of constraints that we impose on our message embedding, especially the expressiveness requirement. In all scenarios, performance of pure TarMAC struggles because it cannot deal with global rewards.

5.3.2 MESSAGE CUT OFF

Results shown in Fig. 7 is the performance when the coordination graph is a complete graph. In this paper, we are interested in learning nearly decomposable structure – agents only need to coordinate their actions with few agents at a time. Therefore, we cut off 80% messages according to the mean of distribution when testing and show the results in Fig. 8. The results indicates that we can omit more than 80% communication without significantly affect performance. We show the strategies learnt by our method in supplementary videos.

For comparison, we cut off messages in QMIX+TarMAC whose weights are 80% smallest and find that the performance of QMIX+TarMAC drops significantly (Fig. 8). These results indicate that our method is more powerful in terms of message cut off compared to attentional communication method.

We further cut all the messages in models learnt by our methods and show the development of testing performance in Fig. 10. The win rates decrease dramatically, proving that the superior performance of our method when 80% message is dropped comes from expressive and succinct communication protocols in stead of implicit coordination strategies are learnt.

6 CLOSING REMARKS

In this paper, we presented a novel multi-agent learning framework within the paradigm of centralized training with decentralized execution. This framework fuses value function factorization learning and communication learning, and efficiently learns nearly domcomposable value functions for agents to act independently most of the time and communicate when it is necessary for coordination. We introduce two information-theoretical regularizers to minimize overall communication while maximizing the message information for coordination. Empirical results in challenging StarCraft II tasks show that our method significantly outperforms baseline methods and allows to reduce communication by more than 80% without sacrificing the performance. We also observe that nearly minimal messages (e.g., with one or two bits) are learned to communicate between agents in order to ensure effective coordination.

REFERENCES

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1): 427–438, 2012.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2951–2960, 2017.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pp. 1538–1546, 2019.

Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.

Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1146–1155. JMLR. org, 2017.

Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Sergey Levine, and Yoshua Bengio. Infobot: Transfer and exploration via the information bottleneck. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*, pp. 2701–2711, 2017.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pp. 3040–3049, 2019.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pp. 7254–7264, 2018.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Akshat Kumar, Shlomo Zilberstein, and Marc Toussaint. Scalable multiagent planning using probabilistic inference. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, pp. 441–470. Springer, 2012.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4292–4301, 2018.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019a.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019b.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5887–5896, 2019.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

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

Nicolas Usunier, Gabriel Synnaeve, Zeming Lin, and Soumith Chintala. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Chongjie Zhang and Victor Lesser. Coordinated multi-agent reinforcement learning in networked distributed pomdps. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Chongjie Zhang and Victor Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 1101–1108. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

APPENDIX

A VARIATIONAL BOUND ON MUTUAL INFORMATION

In order to let messages effectively reduce the uncertainty in action-value functions of other agents, we propose to maximize mutual information between A_j and M_{ij}. We borrow ideas from variational inference literature and derive a lower bound on this mutual information regularizer.

Theorem 1. *A lower bound of mutual information $I_{\boldsymbol{\theta}_c}(A_j; M_{ij}|\mathrm{T}_j, M_{(-i)j})$ is*

$$\mathbb{E}_{\mathbf{T}\sim\mathcal{D}, M_j^{in}\sim f_m(\mathbf{T};\boldsymbol{\theta}_c)}\left[-\mathcal{CE}\left[p(A_j|\mathbf{T})\|q_\xi(A_j|\mathrm{T}_j, M_j^{in})\right]\right], \tag{8}$$

where T_j is local action-observation history of agent j and $\mathbf{T} = \langle \mathrm{T}_1, \mathrm{T}_2, \ldots, \mathrm{T}_n\rangle$ is the joint local history sampled from the replay buffer $\mathcal{D}$, $q_\xi(A_j|\mathrm{T}_j, M_j^{in})$ is the variational posterior estimator with parameters ξ.

Proof.

$$I_{\boldsymbol{\theta}_c}(A_j; M_{ij}|T_j, M_{(-i)j}) \tag{9}$$

$$= \int p(a_j, \tau_j, m_j^{in}) \log \frac{p(a_j, m_{ij}|\tau_j, m_{(-i)j})}{p(a_j|\tau_j, m_{(-i)j})p(m_{ij}|\tau_j, m_{(-i)j})} da_j d\tau_j dm_j^{in} \tag{10}$$

$$= \int p(a_j, \tau_j, m_j^{in}) \log \frac{p(a_j|\tau_j, m_j^{in})}{p(a_j|\tau_j, m_{(-i)j})} da_j d\tau_j dm_j^{in}, \tag{11}$$

where $p(a_j|\tau_j, m_j^{in})$ is defined by our encoder $f_m(m_i|\tau_i; \boldsymbol{\theta}_c)$ and Markov Chain:

$$p(a_j|\tau_j, m_j^{in}) \tag{12}$$

$$= \int p(\tau_{-j}, a_j|\tau_j, m_j^{in}) d\tau_{-j} \tag{13}$$

$$= \int p(\tau_{-j}|\tau_j, m_j^{in}) p(a_j|\boldsymbol{\tau}) d\tau_{-j} \quad \left(\text{According to } \left[a_j \perp m_j^{in}|\boldsymbol{\tau}\right]\right) \tag{14}$$

$$= \int \frac{p(\boldsymbol{\tau}) p(m_j^{in}|\boldsymbol{\tau}) p(a_j|\boldsymbol{\tau})}{p(\tau_j, m_j^{in})} d\tau_{-j}. \tag{15}$$

We introduce $q_\xi(a_j|\tau_j, m_j^{in})$ as a variational approximation to $p(a_j|\tau_j, m_j^{in})$. Since

$$D_{\mathrm{KL}}(p(a_j|\tau_j, m_j^{in})\|q_\xi(a_j|\tau_j, m_j^{in}) \geq 0, \tag{16}$$

we have

$$\int p(a_j|\tau_j, m_j^{in}) \log p(a_j|\tau_j, m_j^{in}) da_j \tag{17}$$

$$\geq \int p(a_j|\tau_j, m_j^{in}) \log q_\xi(a_j|\tau_j, m_j^{in}) da_j. \tag{18}$$

Thus, for the mutual information term:

$$I_{\boldsymbol{\theta}_c}(A_j; M_{ij}|T_j, M_{(-i)j}) \tag{19}$$

$$\geq \int p(a_j, \tau_j, m_j^{in}) \log \frac{q_\xi(a_j|\tau_j, m_j^{in})}{p(a_j|\tau_j, m_{(-i)j})} da_j d\tau_j dm_j^{in} \tag{20}$$

$$= \int p(a_j, \tau_j, m_j^{in}) \log q_\xi(a_j|\tau_j, m_j^{in}) da_j d\tau_j dm_j^{in} \tag{21}$$

$$- \int p(a_j, \tau_j, m_j^{in}) \log p(a_j|\tau_j, m_{(-i)j}) da_j d\tau_j dm_j^{in} \tag{22}$$

$$= \int p(\boldsymbol{\tau}) p(m_j^{in}|\boldsymbol{\tau}) p(a_j|\boldsymbol{\tau}, m_j^{in}) \log q_\xi(a_j|\tau_j, m_j^{in}) da_j d\boldsymbol{\tau} dm_j^{in} \tag{23}$$

Table 1: The number of agents n, the scaling weight β and the communication bandwidth c_len for different tasks.

	Tracker	Hallway	3b vs 1h1m	3s vs 5z	1o2r vs 4r	5z vs 1ul	MMM	1o10b vs 1r
n	3	2	3	3	3	5	10	11
β	1e-3	1e-3	1e-5	1e-5	1e-5	1e-3	1e-5	1e-5
c_len	4	2	6	6	6	12	27	30

$$
\begin{aligned}
&- \int p(a_j, \tau_j, m_{(-i)j}) \log p(a_j|\tau_j, m_{(-i)j}) da_j d\tau_j dm_{(-i)j} && (24)\\
&= \int p(\boldsymbol{\tau}) p(m_j^{in}|\boldsymbol{\tau}) p(a_j|\boldsymbol{\tau}) \log q_\xi(a_j|\tau_j, m_j^{in}) da_j d\boldsymbol{\tau} dm_j^{in} \quad \left(\text{According to } \left[a_j \perp m_j^{in}|\boldsymbol{\tau}\right]\right) && (25)\\
&\quad + H_{\boldsymbol{\theta}_c}(A_j|\mathrm{T}_j, M_{(-i)j}) && (26)\\
&= \mathbb{E}_{\mathbf{T}\sim\mathcal{D}, M_j^{in}\sim f_m(\mathbf{T};\boldsymbol{\theta}_c)} \left[\int p(a_j|\mathbf{T}) \log q_\xi(a_j|\mathrm{T}_j, M_j^{in}) da_j\right] && (27)\\
&\quad + H_{\boldsymbol{\theta}_c}(A_j|\mathrm{T}_j, M_{(-i)j}) && (28)\\
&= \mathbb{E}_{\mathbf{T}\sim\mathcal{D}, M_j^{in}\sim f_m(\mathbf{T};\boldsymbol{\theta}_c)} \left[-\mathcal{CE}\left[p(A_j|\mathbf{T}) \| q_\xi(A_j|\mathrm{T}_j, M_j^{in})\right]\right] && (29)\\
&\quad + H_{\boldsymbol{\theta}_c}(A_j|\mathrm{T}_j, M_{(-i)j}). && (30)
\end{aligned}
$$

Because $H_{\boldsymbol{\theta}_c}(A_j|\mathrm{T}_j, M_{(-i)j}) \geq 0$, we get the lower bound in Theorem 1.

□

B Implementation Details

B.1 Details of Message Dropping

In our methods, not only the number of messages, but also the length of messages are minimized. In other words, we send messages with varying length in the communication channel. However, messages at the recipient side are feed into action-value function approximator, which requires inputs with fixed length. To solve this problem, we send a mask indicating which bits are sent along with the messages. To save channel width, masks are regraded as a binary number, so each of them only consumes a negligible $\log$-scale space compared to the length of message.

B.2 Network Architecture, Hyperparameters, and Infrastructure

We base our framework on PyMARL implementation of QMIX (Samvelyan et al., 2019a) and use its default parameters to carry out all the experiments. We train our models on an NVIDIA RTX 2080TI GPU using experience sampled from 16 parallel environments. We have collected a total of 20 million time step data for each task. Specific values of the number of agents n, the scaling weight β and the communication bandwidth c_len of each agent can be found in Table 1.

C Experimental Results

C.1 Didactic example: independent search

In **independent search**, two agents are finding landmarks in two independent 5×5 rooms for 100 time steps (see Fig. 9). Agent is rewarded 1 every time step they step on the landmark in its room.

Independent search is an example where agents are totally independent. This task aims to demonstrate that our method can learn not to communicate in independent scenarios. We show the demonstrative plot in Fig. 9 and team performance comparison in Table 2.

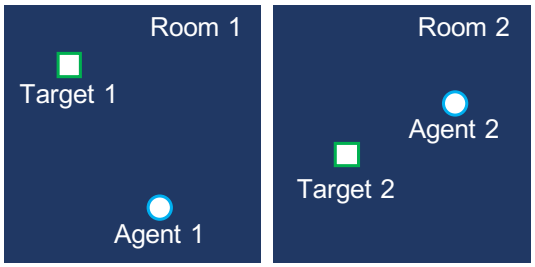


Figure 9: Task **Independent-search**. Two agents are both reward- and transition-independent.

Table 2: Team reward gained on average in an episode on task independent-search.

	Ours	QMIX	TarMAC	TarMAC + QMIX
No message is cut	96.0	96.0	96.0	96.0
100% messages are cut	96.0	—	—	96.0

C.2 StarCraft II

We first describe the scenarios that we consider in details. We consider combat scenarios where the enemy units are controlled by StarCraft II built-in AI (difficulty level is set to `medium`) and each of the ally unit is controlled by a learning agent. Units of two groups can be asymmetric and the initial placement is random. At each timestep, each agent takes one action from the discrete action space consists of the following actions: `noop`, `move`[`direction`], `attack`[`enemy_id`], and `stop`. Under the control of these actions, agents move and attack in a continuous map. A global reward that is equal to the total damage dealt on the enemy units is given at each timestep. Killing each enemy unit and winning a combat induces an extra bonus of 10 and 200 respectively.

3b_vs_1h1m: 3 Banelings spawning randomly on the map try to kill a Hydralisk assisted by a Medivac. 3 Banelings together can just blow up the Hydralisk. Therefore, they should not give the Hydralisk rest time during which the Medivec can restore its health. Banelings have to align their attacking time to get the winning reward. This scenario is designed to test whether our method can learn communication protocol to coordinate actions.

3s_vs_5z: 3 Stalkers encounter 5 Zealots on a map. Zealots can deal high damage but are much slower so that Stalkers have to take advantage of this to beat Zealots using a technique called *kiting* – Stalkers should alternatively attack the Zealots and flee for a distance. Kiting requires the knowledge of exact positions of enemies. Since we narrow down the sight range of units, 3 Stalkers has to coordinate and learn to communicate necessary messages to win.

1o2r_vs_4r: An Overseer has found 4 Reapers. Ally units of the Observer, 2 Roaches, need to get there and kill the Reapers. At the beginning of each episode, the Overseer and Reapers spawn at a random point on the map while the Roaches are initialized at another random point. Since the sight range of Roaches is limited, only the Overseer knows the position of the enemy. Therefore, a learning algorithm has to learn to communicate the target position to effectively win the combat.

5z_vs_1ul: 5 Zealots try to kill a powerful Ultralisk. A sophisticated micro-trick demanding right positioning and attack timing has to be learnt to win.

MMM: Symmetric teams consist of 7 Marines, 2 Marauders and 1 Medivac spawn at two fixed points and the enemy team are tasked to attack the ally team. This task can demonstrate the scalability of our method.

1o10b_vs_1r: In a map full of cliffs, an Overseer detects a Roach, and its teammates, 10 banelings need to kill this Roach to get winning reward. The overseer and the Roach spawn at a random point while the Banelings spawn randomly all round the map. In the minimized communication strategy, Banelings can keep silence and the Overseer needs to encode its position and send it to Banelings. We use this task to test the performance of our method in complex scenarios.

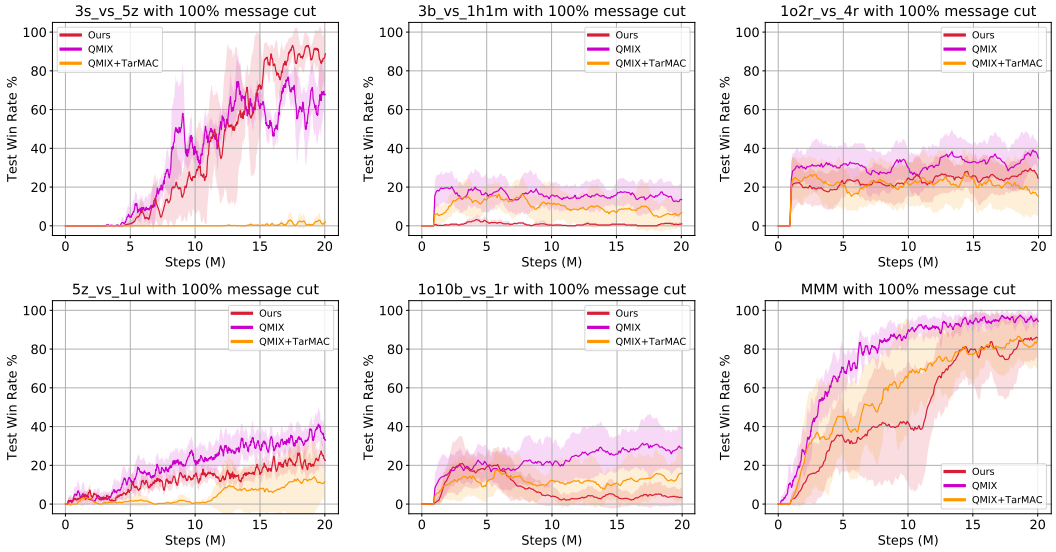


Figure 10: Performance comparison of our method and TarMAC+QMIX when 100% messages are cut off. We also plot the learning curves of QMIX for comparison. As expected, performance of our method drops

Performance of our methods when all messages are cut is shown in Fig. 10