Neurocomputing 513 (2022) 341–350

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Common belief multi-agent reinforcement learning based on variational recurrent models

Xianjie Zhang [a], Yu Liu [a,*], Hangyu Mao [b], Chao Yu [c]

[a] Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian University of Technology, Dalian 116620, China
[b] Huawei Noah's Ark Lab, No. 3 Xinxi Road, Haidian District, Beijing 100085, China
[c] School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 30 January 2022
Revised 12 September 2022
Accepted 24 September 2022
Available online 29 September 2022

Keywords:
Deep reinforcement learning
Multi-agent systems
Variational recurrent models
Value-based method

ABSTRACT

The tacit cooperation among human teams benefits from the fact that consensus can be reached on a task through common belief. Similar to human social groups, agents in distributed learning systems can also rely on common belief to achieve cooperation under the condition of limited communication. In this paper, we show the role of common belief among agents in completing cooperative tasks, by proposing the Common Belief Multi-Agent (CBMA) reinforcement learning method. CBMA is a novel value-based method that infers the belief between agents with a variational model and models the environment with a variational recurrent neural network. We validate CBMA on two grid-world games as well as the StarCraft II micromanagement benchmark. Experimental results show that the learned common belief by CBMA can improve performance in both discrete and continuous state settings.

1. Introduction

As a key method to solve many practical decision-making problems, cooperative multi-agent reinforcement learning (MARL) has recently made huge progress in a number of domains, such as autonomous vehicle teams [1], network packet delivery [2,3], and distributed logistics [4]. In these real-world domains, each decision-making agent is usually constrained by its local observation and communication constraints, which makes it a challenging issue to achieve efficient cooperation among the agents [5].

To cope with such problems, a series of centralised training and decentralised execution (CTDE) methods have been proposed. MADDPG [6] and COMA [7] learned the centralised critic and decentralised actors within the actor-critic framework. QMIX [8] used a mixing network to factorize the value functions. Other prominent progress includes but is not limited to studies such as [9–14]. However, all these methods only use centralised critic to coordinate *during training*, and lack a coordination mechanism among agents *during execution*.

Therefore, a large number of studies resorted to communication mechanisms [15–20] to enable coordination among agents during the execution process. These works are normally built upon the assumption that agents can share some kind of private information using explicit communication protocols or emergent symbols. In the emergence of human cooperation, however, individuals are usually able to perceive their situations in a consistent way even without communication, which is a kind of natural consciousness [21]. Inspired by this, some studies have introduced the concept of cognitive recognition or common knowledge into multi-agent systems [22,23], exhibiting promising performance in facilitating cooperation among the agents. However, due to agents' local observations of the environment, it is difficult for the agents to form consistent beliefs that can be shared among them for the achievement of cooperation.

In this paper, we propose the Common Belief Multi-Agent (CBMA) method, which is a novel value-based RL method that infers the common beliefs among the agents under the constraints of local observations. CBMA enables agents to infer their latent beliefs through local observations and make consistent latent beliefs using a KL-divergence metric. In order to deal with the historical data while at the same time inferring consistent latent beliefs, our approach is based on a variational recurrent neural network framework that combines a recurrent neural network (RNN) within the variational model [24,25]. In the training phase, all the observations are available to obtain a common belief for each agent, while only local observation and the latent variable inferred by itself can be accessed by each agent's controller in the execution phase. Therefore, CBMA is still under the CTDE regime, but is communication-free and fully distributed in the sense that no

* Corresponding author.
E-mail addresses: zhangxianjie@mail.dlut.edu.cn (X. Zhang), yuliu@dlut.edu.cn (Y. Liu), maohangyu1@huawei.com (H. Mao), yuchao3@mail.sysu.edu.cn (C. Yu).

https://doi.org/10.1016/j.neucom.2022.09.144
0925-2312/

explicit information should be exchanged among the agents and only local information is required during the execution process.

We evaluate our methods on the grid world with discrete states and the StarCraft II Micromanagement with continuous states. Experimental results demonstrate that CBMA can achieve state-of-the-art performance compared to the existing MARL methods.

2. Related work

2.1. MARL

As a critical way to solve the cooperative decision-making problem, MARL has a long history of research [26,27]. Recently, a widely adopted MARL solution is to use the CTDE framework [6,28] to realize coordinated learning among the agents. MADDPG [6] learns a critic for each agent based on the global information, but only uses local information when executing. This form of critic-networks lead to a sharp increase in critic input dimensions. MAAC [29] uses the attention mechanism in the critic network to solve the problem of instability in the environment. COMA [7] also applies a centralised critic to keep each agent informed about the relative contribution to the whole system with counterfactual reward. Another type of CTDE methods decompose the global value function into some factored local value functions. VDN [30] directly sums the independent functions, and QMIX [8] ensures a monotonic mixing of each agent's utilities depending on the global state. To solve the constraints of Individual-Global-Max (IGM) [9] under joint action-value functions, QTRAN [9] transforms the original joint action-value function into individual action-value functions. In addition, the attention mechanism is introduced in Weighted QMIX [31] and AI-Qmix [32] to achieve better performance in complex problems, and MAVEN [10] adds a mutual information loss to encourage the diversity of trajectories with a shared latent space, which solves the problem of ε-greedy exploration in the QMIX algorithm. Although all these methods use centralised critics to realize the cooperation among the agents, they do not consider any coordination mechanisms during execution.

2.2. Variational recurrent models

Variational recurrent models are another line of work that are used to infer the ground truth and help to make decisions, mainly including variational autoencoder (VAE), recurrent neural network (RNN) and so on. The VAE [33,34] can learn a generative model of the non-sequential data, and then performs inference. The VAE trains the two parts (the encoder and decoder) in a joint way, in which we can obtain the belief z to effectively aggregate the available information. However, VAE is usually limited to relatively simple state transition structures. Since most of the data in the real world is serialized, such as natural speech and behavior trajectory, learning the sequence model is the critical technology to solve the long-standing machine learning challenge [25]. The simplest way to fully mine the important information in local observations in the sequence model is to use RNN [8,35] to process sequences of inputs. Nevertheless, in the face of data with strong and complex correlations between output variables at different time steps, the correlation of these variables cannot be adequately modeled in the standard RNN. Therefore, Chung et al. proposed a stochastic sequential model based on VAE to avoid the above problem by learning effective belief representations [25]. Researchers like Krishnan et al., Archer et al., Karl et al., Fraccaro et al. and Doerr et al. also explored different methods to learn effective state beliefs [36–40].

2.3. Belief state in RL

Natural sequential data exists widely in RL [41], so there are many works on single-agent reinforcement learning to solve partially observable problems by unifying RL and stochastic sequential models [42,43,24,44,45,22]. In partially observable Markov decision process (POMDP), we can use the sequence attributing of historical data $\mathbf{x} = (x_1, \ldots, x_T)$ to establish sequential variational model for inferring the filtering distributions $p(z_t|x_1, \ldots, x_t)$, the belief state in RL. These methods use VAE models to obtain latent variables as inputs to their controllers. They are different from each other in the factorization of the generation and inference models. Benefited from this advantage of the variational RNN framework and the belief state z, it is feasible to handle dynamic local observations in MARL. Inspired by the consistent belief in human society, MACKRL [23] proposed a hierarchical structure of common knowledge learning algorithm for agents. Owing to the existence of a large number of possible trajectories, computing the common-knowledge beliefs in MACKRL can be computationally expensive. NCC-MARL [22] proposed the neighborhood cognition consistency. In contrast, our method only needs to calculate one latent belief per agent, which greatly reduces the amount of computation.

3. Preliminaries

The problem of multi-agent cooperation can be formulated as a decentralised partially observable Markov decision process (Dec-POMDP) [46] denoted by a tuple $\mathbf{T} = \langle S, O, A, R, P, Z, \gamma, \mathbb{N} \rangle$. Within $\mathbf{T}$, $o_i \in O$ contains partial information from the global state, $s \in S$, and $a_i \in A$ is the action of agent i. We assume that each agent $i \in \mathbb{N} := \{1, \ldots, N\}$ holds a partially observable state, o_i, which is obtained through an observation function, $Z(s, i) : S \times \mathbb{N} \to O$. The joint action for the N agents is represented by $\mathbf{a} := (a_i)_{i \in \mathbb{N}} \in A^N$. The state transition is $P(s'|s, a_1, \ldots, a_N) : S \times A^N \times S \to [0, 1]$. At each transition, the environment provides a reward according to the reward function $R(s, a) : S \times A^N \to \mathbb{R}$. The agents aim to learn a joint policy $\boldsymbol{\pi} = \prod_i \pi_i(a_i|o_i)$ that optimizes the shared utility $G = \sum_t \gamma^t r_t$, where G is an expected discounted future reward and γ is the discount factor. A joint action-value function $Q^{\pi}_{\text{tot}}(s, \mathbf{a}) = \mathbb{E}[G|S = s, \boldsymbol{\pi}]$ is derived by the joint policy $\boldsymbol{\pi}$.

3.1. Q-learning

The Q-learning algorithm is a popular value-based RL method which can be written as a bootstrapping formula $Q(s, a) = \mathbb{E}_{(s,a,r,s')}[r(s, a) + \gamma \max_{a'} Q(s', a')]$. In deep Q-networks (DQNs) [47], the action-value function is parameterised by a neural network, θ, which is updated to minimize the Temporal-Difference (TD) error:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}(y' - Q(s, a; \theta)) \tag{1}$$

where the target value y' can be calculated through the target network, $y' = r + \gamma \max_{a'} Q(s', a'; \bar{\theta})$ with the target network parameter $\bar{\theta}$ updated by copying from θ. When this method is used in multi-agent systems, a direct learning method is to learn the decentralised action-value function Q_i [48,49] independently. The drawback of this method is that it does not consider the environmental instability caused by the simultaneous actions of other agents.

3.2. Individual-global-max (IGM) condition

In the CTDE [50] framework, the centralised training algorithm can acquire global state s and observation-action trajectories of all

agents. Some studies [8,9] applied the factorization of global Q-function based on the concept of IGM condition [9]. In multi-agent Q-learning, a fundamental requirement of IGM is that the optimal joint action induced from the optimal centralised action value function is equivalent to the collection of individual optimal actions of agents [51]. Therefore, IGM can be formalized as follows:

$$\arg\max_{\mathbf{a}} Q_{\text{tot}} = \begin{pmatrix} \arg\max_{a^1} Q_1 \\ \vdots \\ \arg\max_{a^n} Q_n \end{pmatrix} \tag{2}$$

where Q_{tot} is a function with joint action value, and Q_i satisfys IGM for Q_{tot}. In other words, Q_i is a factor of Q_{tot}. The most popular methods are the factorized VDN method with an additivity operation and the QMIX algorithm with a monotonicity restriction.

Algorithm 1 CBMA

Initialize RL networks (controller MLP network θ_Q; mixing network θ_m)
Initialize variational model (RNN network; encoder and decoder networks)
Initialize exploration $\varepsilon = 1$; Total steps $t = 0$; Replay buffer $\mathscr{B} = \{\}$
Initialize common belief variation networks training frequency δ; First stage training times t_{repeat_times}
Initialize *Train_first_stage_flag* = *True*; Number of execution episode *Enough_episode_num*
for each episodic iteration **do**
  Initialize an episode
  for step not terminated **do**
    Calculate the value of Q_i function of all agents, get ε-greedy actions
    ε is annealed from 1.0 to 0.05
  end for
  Record (o, s, r) in replay buffer $\mathscr{B}$
  $t = t + episode\_steps$
  for each gradient step **do**
    if mini-batch can be sample from buffer **then**
      Train RL controller
      Sample mini-batch from replay buffer B
      Update controller MLP and mixing network (Eqution (15)):
      $L_{Q_{\text{mix}}}(\xi) = \mathbb{E}_{\text{batch}}\left[\left(y^{\text{tot}} - Q_{\text{tot}}(\mathbf{o}, \mathbf{d}, \mathbf{a}, s; \xi)\right)^2\right]$
      if $t >$ *Enough_episode_num* and *Train_first_stage_flag* = *True* **then**
        repeat
          Sample mini-batch from $\mathscr{B}$
          Train first stage common belief variational network (Eqution (16)): L_{VR}^1
        until $t >$ *Enough_episode_num* $+ t_{repeat\_times}$
        Train_first_stage_flag = *False*
      end if
      if $random(0,1) < \delta$ **then**
        Sample mini-batch from $\mathscr{B}$
        Train second stage common belief variational network (Eqution (16)): L_{VR}^2
      end if
    end if
  end for
end for

4. Method

In this section, we introduce our MARL algorithm CBMA in detail. The key idea is to infer a latent variable z_i for each agent during the execution phase using an RNN-based variational model. Similar to human common beliefs, these variables are kept consistent in the process of execution in order to ensure coordination among the agents.

4.1. The variational model in dec-POMDP

In VAE, the generative model is trained with the form $p(x,z) = p(z)p(x|z)$, where $p(z)$ is the prior distribution of the latent variable z and $p(x|z)$ is the likelihood function which generates data x given the latent variable z. To solve the intractable true posterior $p(z|x)$, the distribution $p(z|x)$ is approximated by the tractable distribution $q(z|x)$. The VAE is trained by maximizing the evidence lower bound (ELBO) of log-likelihood.

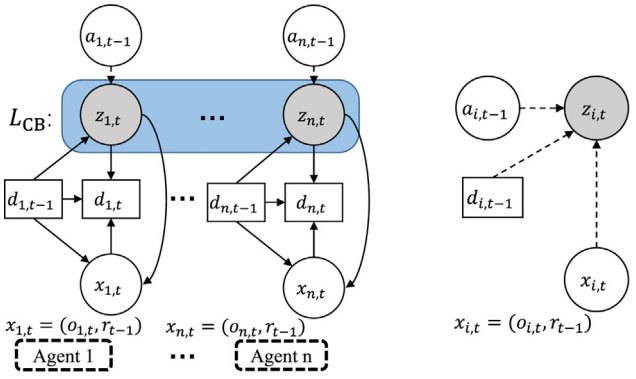
$$\log p(x) = D_{\text{KL}}(q(z|x)||p(z|x)) + \text{ELBO} \tag{3}$$

$$\text{ELBO} = -D_{\text{KL}}(q(z|x)||p(z)) + \mathbb{E}_q(\log p(x|z)) \tag{4}$$

where $D_{\text{KL}}(q(z|x)||p(z|x))$ is always positive, and it is only necessary to maximize ELBO to get $q(z|x)$ to approximate the true posterior distribution $p(z|x)$. Maximizing the ELBO is equivalent to minimizing the loss function, $L^{\text{elbo}} = -\text{ELBO}$. The loss function about ELBO is shown as below.

$$L^{\text{elbo}} = -\left[\mathbb{E}_q(\log p(x|z)) - D_{\text{KL}}(q(z|x)||p(z))\right] \tag{5}$$

VRNN [25] is an extension of VAE and can be used to process sequential data. We generalize VRNN to multiagent sequential decision making settings. Consider Dec-POMDP, where there are N agents. For each agent, the local state-transition (i.e., an agent performing a local action $a_{i,t}$ under local observation $o_{i,t}$ transfers to the next moment of local observation $o_{i,t+1}$) can be modeled approximately by a graphical model. The reward r is combined with the partial observation o_i of the agent i as an entire $x_i = (o_i, r)$ for convenience. Fig. 1 gives the overall graphical model



(a) Generative model. (b) Inference model.

Fig. 1. The graphical model of CBMA, where squares are deterministic variables, and circles represent stochastic variables. The white circles are observed variables, while the gray ones are latent variables. (a) is the generative model, where we can get a latent variable $z_{i,t}$ from each generative model and calculate the common belief loss between them. Within generative model $p_\theta(z_{i,t}|d_{i,t-1})p_\theta(x_{i,t}|z_{i,t-1})$, the variable $z_{i,t}$ generates from the conditional prior distribution $p_\theta(z_{i,t}|d_{i,t-1})$ and the variable $x_{i,t}$ from the conditional distribution $p_\theta(x_{i,t}|z_{i,t-1})$. (b) is the inference model of each agent i. Note, $d_{i,t}$ is the hidden layer vector of recurrent neural network.

of the multiagent VAE model, CBMA, where each agent i has observable variables x_i and corresponding latent variables z_i. Moreover, CBMA has the following components:

- The initial distribution and the transition of **prior** distribution are $p_\theta(z_{i,1})$ and $p_\theta(z_{i,t}|z_{i,t-1}, x_{i,t-1}, a_{i,1:t-1})$, respectively. For convenience, the transition distribution for agent i at time step t is marked as $p_{i,t}^{\theta_{\text{prior}}}$.
- In the **generative model**, the future observation is predicted by its internal state, $p_\theta(x_{i,t}|z_{i,1:t}, x_{i,1:t-1})$, where the generative model is marked as $p_{i,t}^{\theta_{\text{dec}}}$.
- The latent variable $z_{i,t}$ is approximated by the **inference model**, $q_\phi(z_{i,t}|z_{i,1:t-1}, x_{i,1:t}, a_{i,1:t-1})$, which is denoted as $q_{i,t}^{\phi_{\text{enc}}}$.

Based on Eq. (5) and the notations defined above, the loss function of ELBO for agent i can be rewritten follow.

$$L_i^{\text{elbo}} = -\mathbb{E}_{(o_i,a_i)}\left[\mathbb{E}_{q_\phi}\left(\log p_{i,t}^{\theta_{\text{dec}}}\right) - D_{\text{KL}}\left(q_{i,t}^{\phi_{\text{enc}}}||p_{i,t}^{\theta_{\text{prior}}}\right)\right] \tag{6}$$

Similar to the notations in VAE [33], we also name $q_{i,t}^{\phi_{\text{enc}}}$ the encoder and $p_{i,t}^{\theta_{\text{dec}}}$ the decoder, respectively. Note that, $\mathbb{E}_{(o_i,a_i)}$ represents a mini-batch version of the loss function.

The variational model is combined with a gated recurrent unit (GRU) [52] network in order to form a highly flexible function approximator. Each agent's hidden layer variable can be updated by the following iterative equation:

$$d_{i,t} = \text{GRU}(d_{i,t-1}; z_{i,t}, x_{i,t}) \tag{7}$$

where $d_{i,t}$ is the hidden layer variable.

4.2. Reparameterizable approximation

The Generative Model. The variational model is conditioned on the hidden layer variable $d_{i,t}$ of the GRU. We can obtain a prior distribution based on historical information, $z_{i,t} \sim \mathcal{N}(\mu_{i,t}^\theta, \left(\sigma_{i,t}^\theta\right)^2\mathbf{I})$, with a diagonal covariance structure $\mathbf{I}$. We use the Gaussian distributions as the outputs. The parameterized diagonal Gaussian distribution is as follows:

$$\left[\mu_{i,t}^\theta, \left(\sigma_{i,t}^\theta\right)^2\right] = \theta^{\text{prior}}(d_{i,t-1}, a_{i,t-1}) \tag{8}$$

where $\mu_{i,t}^\theta$ and $(\sigma_{i,t}^\theta)^2$ are the prior distribution of parameterization, and $\theta^{\text{prior}}(\cdot)$ is the neural network parameter of a prior distribution. The future observations can be predicted by the generative model given the internal states, $x_{i,t}|z_{i,t} \sim \mathcal{N}(\mu_{i,t}^x, \left(\sigma_{i,t}^x\right)^2\mathbf{I})$:

$$\left[\mu_{i,t}^x, \left(\sigma_{i,t}^x\right)^2\right] = \theta^p(z_{i,t}, d_{i,t-1}) \tag{9}$$

where $\mu_{i,t}^x$ and $(\sigma_{i,t}^x)^2$ are the predicted observations distribution of parameterization, and $\theta^p(\cdot)$ is the neural network parameters.

The Inference Model. The common belief $z_{i,t}$ of agent i can be parameterized as a Gaussian distribution $z_{i,t}|x_{i,t} \sim \mathcal{N}(\mu_{i,t}^\phi, (\sigma_{i,t}^\phi)^2\mathbf{I})$, given the variable $x_{i,t}$:

$$\left[\mu_{i,t}^\phi, (\sigma_{i,t}^\phi)^2\right] = \phi^q(x_{i,t}, d_{i,t-1}, a_{i,t-1}) \tag{10}$$

where $\mu_{i,t}^\phi$ and $(\sigma_{i,t}^\phi)^2$ represent the parameterized Gaussian posterior distributions, and $\phi^q(\cdot)$ is the neural network parameter of posterior distributions.

Given the above models, the two items in Eq. 6 can be calculated. The first item is the generative model, $\mathbb{E}_{q_\phi}(\log p_{i,t}^{\theta_{\text{dec}}})$, which is a negative *reconstruction error* in autoencoder parlance. In other words, $\mathbb{E}_{q_\phi}(\log p_{i,t}^{\theta_{\text{dec}}})$ is equivalent to the negative of the distance between the reconstructed variable and the original variable. For each agent i, we have the following formula:

$$\mathbb{E}_{q_\phi}(\log p_{i,t}^{\theta_{\text{dec}}}) \simeq -\left|\tilde{x}_{i,t} - x_{i,t}\right| \tag{11}$$

where $x_{i,t}$ is the raw state and reward sampled from the replay buffer [47], and $\tilde{x}_{i,t}$ is reconstructed variable based on the historical information and latent variable, $\tilde{x}_{i,t} = \theta^p(z_{i,t}, d_{i,t-1})$. Based on Eq. (9), we assume that $\tilde{x}_{i,t}$ belongs to a parameterized Gaussian distribution, with the mean $\mu_{i,t}^x$ and variance $(\sigma_{i,t}^x)^2$. We can obtain the parameterized $\tilde{x}_{i,t} = \mu_{i,t}^x + \sigma_{i,t}^x\epsilon$, where ϵ is an auxiliary noise variable $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.

The second item is the KL-divergence. Similarly, $q_{i,t}^{\phi_{\text{enc}}}$ and $p_{i,t}^{\theta_{\text{prior}}}$ can be parameterized as Gaussian distributions, and the KL-divergence between these two distributions is given as follows:

$$\begin{aligned} & D_{\text{KL}}\left(q_{i,t}^{\phi_{\text{enc}}}||p_{i,t}^{\theta_{\text{prior}}}\right) \\ = \ & \log\frac{\sigma_{i,t}^\theta}{\sigma_{i,t}^\phi} + \frac{\left(\sigma_{i,t}^\phi\right)^2}{2\left(\sigma_{i,t}^\theta\right)^2} + \frac{\left(\mu_{i,t}^\phi - \mu_{i,t}^\theta\right)^2}{2\left(\sigma_{i,t}^\theta\right)^2} - \frac{1}{2} \end{aligned} \tag{12}$$

Our generative model and inference model are trained jointly to maximize the ELBO with respect to their parameters. The detailed training algorithm is shown in the Algorithm 1.

Common Belief. In the previous subsections, we use a variational distribution to approximate the prior distribution for each agent. This allows each agent to infer a meaningful latent variable. However, for multiple agents, simply inferring their latent variables alone might not benefit the establishment of cooperation. One possible solution is that each agent infers a latent variable about the global state s. Given a prior distribution $p(z_t|s_t)$, we can use $D_{\text{KL}}(q_{i,t}^{\phi_{\text{enc}}}||p(z_t|s_t))$ to infer the common beliefs. However, the global state is often inaccessible to agents due to partial observations. To circumvent this problem, we let the variational distributions between agents approximate each other, in order to realize the belief consistency among the agents:

$$\begin{aligned} L_{\text{CB}} &= \mathbb{E}_{(o_i,a_i)}\sum_{i=1}^{N}\sum_{j=1}^{N} D_{\text{KL}}\left(q_{i,t}^{\phi_{\text{enc}}}||q_{j,t}^{\phi_{\text{enc}}}\right) \\ &= \mathbb{E}_{(o_i,a_i)}\left[\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\log\frac{\sigma_{j,t}^\phi}{\sigma_{i,t}^\phi} + \frac{\left(\sigma_{i,t}^\phi\right)^2 + \left(\mu_{i,t}^\phi - \mu_{j,t}^\phi\right)^2}{2\left(\sigma_{j,t}^\phi\right)^2}\right)\right] \end{aligned} \tag{13}$$

The above loss L_{CB} will keep agent i' belief $q_{i,t}^{\phi_{\text{enc}}}$ as close as to agent j' belief $q_{j,t}^{\phi_{\text{enc}}}$, which means that agent i and j will achieve common belief as expected.

4.3. The training and execution process

The training of CBMA is based on the QMIX family of algorithms, by minimizing the TD loss and variational model loss in parallel. Pseudocode is given in Algorithm 1. The overall architecture of our algorithm is shown in Fig. 5.

The TD Loss. In our model, the CBMA uses a monotonic approximation mixing network, which is originated from QMIX. QMIX guarantees that the global argmax executed on Q_{tot} produces the same result as a set of individual argmax operations executed on each Q_a, formalized as Eq. (2). The QMIX algorithm enforces monotonicity through a constraint on the relationship between Q_{tot} and each Q_a, and this monotonic function is sufficient but not necessary to satisfy Eq. (2). Monotonicity is shown as follow.

$$\frac{\partial Q_{\text{tot}}}{\partial Q_a} \geqslant 0, \forall a \in A \tag{14}$$

As shown in Fig. 2(a), the mixing network processes the input from the agents and outputs the Q_{tot}. To enforce the monotonicity of Eq. (14) the weights of the hypernetworks (excluding bias) are restricted to be non-negative. The non-negative weights allow the hypernetworks to approximate any monotonic function arbitrarily close [53]. The weights of layers of the mixing network are generated by using the state s as an input to the hypernetworks. The hypernetworks consists of a linear layer and an absolute activation function, which ensures that the generated hypernetworks weights are non-negative.

We update each agent's Q-network θ_Q and mixing network θ_m using the gradient of TD loss. As shown in Fig. 2(a), the input is $(o_{i,t}, r_{i,t-1}, a_{i,t-1}, z_{i,t-1})$ for every agent's Q-network Q_i, and then each individual utility is fed into the mixing network, which is used to compute the global action-value $Q_{\text{tot}} = \theta_m(Q_1, \ldots, Q_N, s)$. We also describe the computational process of Q_i in detail in Fig. 5. The variable d_i contains information of the latent variable z_i, which potentially allows the agents to have convergent behavioral strategies. To allow the agents to choose specific policy, we encode the local observations o_i of the agents directly concatenating with d_i as the input to Q_i. For convenience, we write the equation for calculating Q_{tot} as $Q_{\text{tot}}(\mathbf{o}, \mathbf{d}, \mathbf{a}, s; \xi)$. Here, $\mathbf{o} = \{o_i\}_{i=1}^N$ contains partially observable information of each agent, and $\mathbf{d} = \{d_i\}_{i=1}^N$ are the internal states calculated by the variational model. The networks are updated by the following equation:

$$L_{Q_{\text{mix}}}(\xi) = \mathbb{E}_{\text{batch}}\left[\left(y^{\text{tot}} - Q_{\text{tot}}(\mathbf{o}, \mathbf{d}, \mathbf{a}, s; \xi)\right)^2\right] \tag{15}$$

where $y^{\text{tot}} = r + \gamma \max_{\mathbf{a}'} Q_{\text{tot}}(\mathbf{o}', \mathbf{d}', \mathbf{a}', s'; \xi^-)$, agent's Q network and the mixing network parameters are denoted as $\xi = (\theta_Q, \theta_m)$ and ξ^- is the parameters of target networks.

The Common Belief Variational Model Loss. The combination of the variational formula (Eq. (6)) and the common belief loss (Eq. (13)) is as follow:

$$L_{\text{VR}} = -\sum_{i=1}^{N} L_i^{\text{elbo}} + \alpha L_{\text{CB}} \tag{16}$$

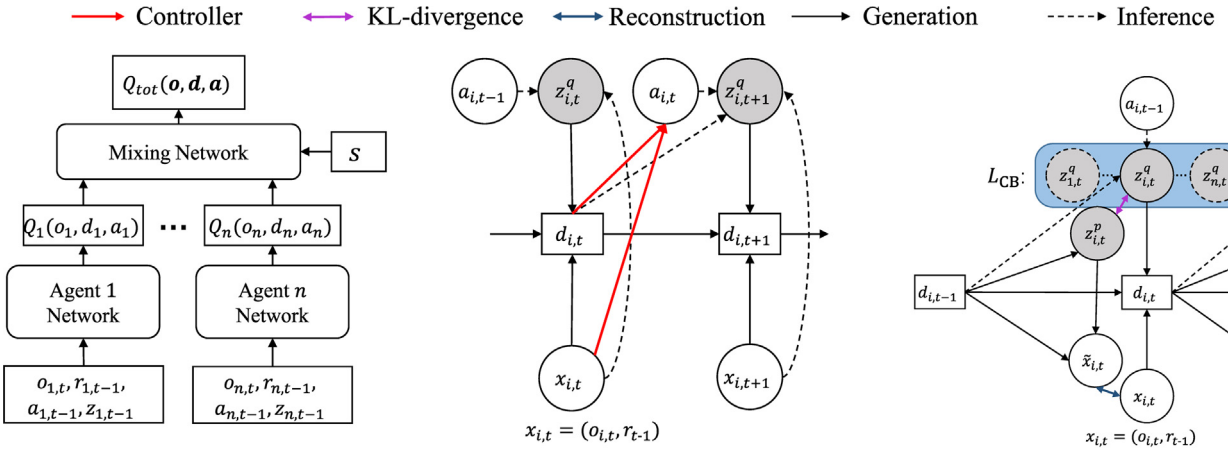
where L_{VR} is the loss for training the common belief variational network. The hyperparameter α is used for balancing the weight of L_{CB}.

As shown in Fig. 2(b), in the execution phase, our model first computes the hidden layer state $d_{i,t}$ in the RNN network. Then, the latent variable of the common beliefs $z_{i,t}$ is calculated. Afterwards, the controller selects an action by $a_i = \arg\max_{a_i} Q_i(o_i, d_i, a_i)$. Fig. 2(c) is the training phase. We sample a mini-batch to train our variational model and RL model. When training the variational model, the generative model and inference model combined with the RNN network are trained to optimize the loss in Eq. (13). When training RL, only the multilayer perceptron (MLP) network of the controller and the mixing network are trained using the loss function in Eq. (15).

The variational network is unstable in the early training stage due to the fact that the agents' policies in the initial stage are more exploratory and most of the trajectories are random. We design two variational networks for solving this problem. The first network, after collecting enough data, conducts enough rounds of training, and then is fixed without further training. The second network is trained with RL until the end of training. The Q-network receives the raw observations and the state variables from both variational networks. In the ablation study section, we show that using two networks can lead to better results compared to using a single variational network. The training algorithm for CBMA is provided in Algorithm 1 and Fig. 5.

5. Experiment

In this section, we examine CBMA on grid world games and the StarCraft II micromanagement (SMAC) tasks [54]. We compare the proposed method with the state-of-the-art MARL algorithms: IQL [49], VDN [30], QMIX [8], QTRAN [9], COMA [7], and Q-DPP [55]. In our algorithm (CBMA) we use the parameter settings as shown in Table 1 and Table 2. In the StarCraft environment, the parameters of our algorithm CBMA are consistent with those in the original QMIX, except for the RNN dimension. Since our model involves the concatenation of multiple variables, we reduced the RNN dimension from 64 to 32. In the Grid experiment, we first optimize the baseline algorithms and find the best parameters. Then the parameters belonging to the common belief module are adjusted based on the parameters of the baseline algorithms. All experiments are conducted based on control variables.



(a) Mixing network structure. **(b) Excecution.** **(c) Training.**

Fig. 2. Computation graph of CBMA. (a) is the QMIX architecture. (b) is the execution phase for agent i. (c) is the training phase. Q_i: Q-value function of agent i; Q_{tot}: the centralised action-value function; s: the global state; d_i: the hidden layer state of agent i; x_i: the variable contains partial observation o_i and reward r, $x_i = (o_i, r)$; z_i: the latent variable and the belief of agent i; L_{CB}: the loss function for the forming of common belief.

Table 1
The hyperparameters used in games.

Hyperparameters	Grid	StarCraft
Learning rate of LR (Controller MLP network and Mixing network)	0.0005	0.0005
Learning rate of common belief variation networks	0.0008	0.0005
RNN Hidden dimension	64	32
Activation function	Relu/ Tanh	Relu/ Tanh
Buffer size	5000	5000
Batch size	32	32
Epsilon init	1.0	1.0
Epsilon end	0.05	0.05
Discount factor γ	0.99	0.99
Common belief α	0.01	0.01

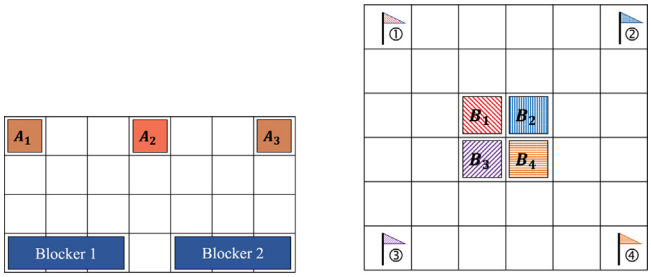
Table 2
The hyperparameters in Algorithm 1 for different games.

	Game Name	Training frequency δ	First stage training times t_{repeat_times}
Grid	Blocker 7×4	0.3	5000
	Navigate 6×6	0.3	5000
	Navigate 8×8	0.3	5000
StarCraft	StarCraft 3 m_vs_3m	0.3	5000
	StarCraft 5 m_vs_6m	0.12	1000
	StarCraft 8 m_vs_9m	0.3	5000

5.1. Discrete state-space games

Blocker Games. In the blocker task [56,57], agents must learn to reach the bottom of the playing field. Meanwhile the blockers aim to stop them with certain strategies. Three agents and two blockers are placed in a 7×4 board, as shown in Fig. 4(a). Blocker 1 and blocker 2 are responsible for column $1-4$ and $4-7$, respectively. The state space consists of the location of each agent and blocker. The action space of each agent includes going up, down, left and right, or staying in place, while the blocker moves left or right in a certain strategy to stop these agents. Each episode has a maximum of 20 steps until one of the agents reaches the ultimate zone. Before reaching the ultimate zone, the team's reward is -1 for each step. When one of the agents succeeds, the episode ends. The maximum cumulative reward for this game is -3.

Coordinated Navigation Games. In the navigation games, four agents aim to learn a collaborative strategy to reach four corresponding landmarks. This type of tasks has the same discrete action settings as the blocker. We set up two tasks with different sizes, i.e., 6×6 and 8×8. For these two tasks, a reward of -1 is given per step before all agents arrive at their terminals. As shown in Fig. 4(b), the agent and flag are identified by a number (i.e., 1–4). The game ends only when all agents reach the corresponding flags. The maximum reward of 6×6 and 8×8 tasks are -4 and -6, respectively.



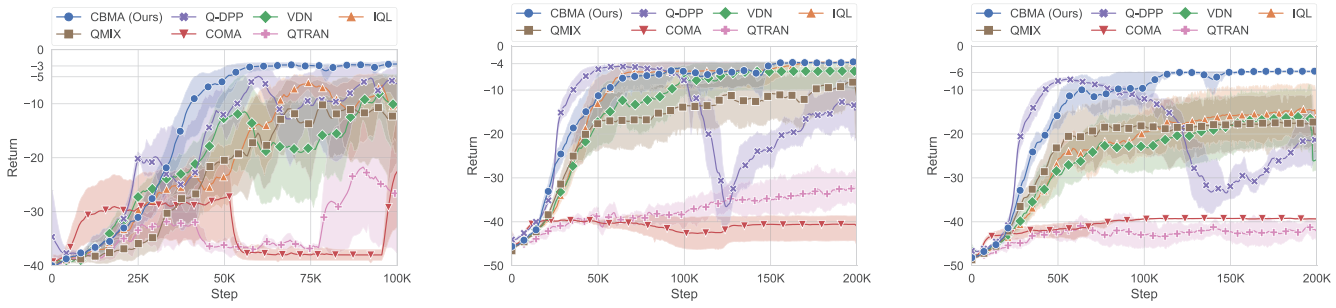
(a) Blocker 7×4. (b) Navigation 6×6.

Fig. 4. Grid games.

Results in Discrete Games. The results are shown in Fig. 3. In the blocker game, some agents need to learn to cheat the blockers, so that the other agents can take this opportunity to reach the bottom smoothly. This requires common belief among agents to achieve such a tacit agreement. We can see from Fig. 3(a) that our algorithm can learn a more stable policy compared to the baselines. CBMA can converge to the highest reward -3 at a relatively fast speed. The results of navigation games can be found in Fig. 3(b) and 3(c). CBMA can achieve satisfactory results in the two environments. As the size of the game increases from 6×6 to 8×8, CBMA performs significantly better than the baseline algorithms. In the small-scale 6×6 game, it is surprising that IQL achieves the same result as CBMA. This might be due to the little exploration space in such small-scale games, where learning without coordination can also quickly achieve the optimal strategies through random exploration. Furthermore, although the state-of-the-art algorithm Q-DPP can quickly reach the highest point in the early stages of training, a stable strategy cannot be obtained in subsequent training and the reward becomes lower later.

5.2. Continuous state-space games

The SMAC tasks. To prove that CBMA can solve more complex collaborative tasks, we conduct experiments in the challenging SMAC tasks [54]. Each agent has its own discrete actions including move[direction], attack[enemy_id], stop, and noop. The test warriors are the Marines, which are ranged-attack units. Here, we report the result for three maps 3 m_vs_3m, 5 m_vs_6m and 8 m_vs_9m. The 3 m_vs_3m is a symmetrical game, where both the Ally and the Enemy groups include 3 Marines. The other two



(a) Blocker 7×4 reward. (b) Navigation 6×6 reward. (c) Navigation 8×8 reward.

Fig. 3. Performance over time on different grid games.

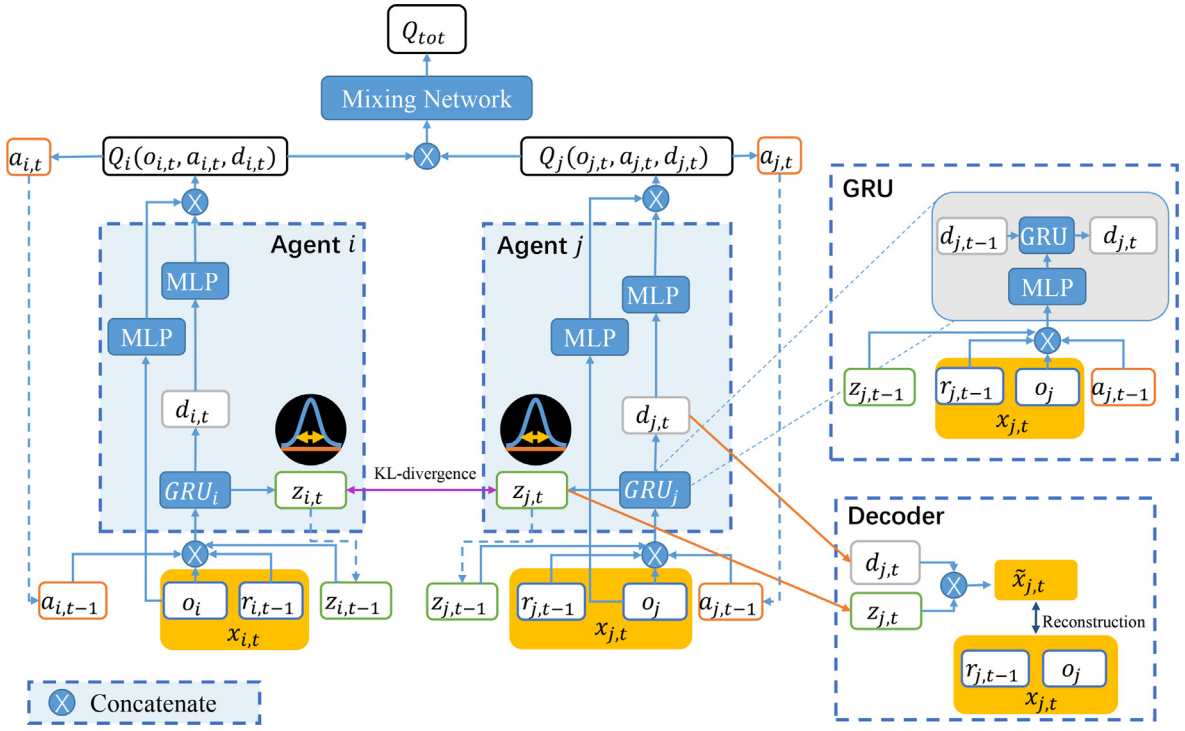


Fig. 5. The overall CBMA architecture. The figure shows that the process of inferring the latent variables z_i and z_j for agent i and j, respectively, which is also applicable in the case of more than two agents. During the training process, the inference network is trained by setting the KL-divergence between z_i and z_j as a loss function so that the distributions between them approximate each other. During the execution process, we do not calculate the KL-divergence between agents, and the algorithm just uses the inference network to get the respective latent variable z_i. Our algorithm guarantees the decentralized execution property.

are asymmetrical games verifying the effect of our algorithm in more complex modes, including: 5 Marines for user versus 6 Marines for enemy (5 m_vs_6m), and 8 Marines for user versus 9 Marines for enemy (8 m_vs_9m).

Results in SMAC. From Fig. 6, we can see that CBMA and other state-of-the-art algorithms can achieve similar optimal performance in easy map (i.e., 3 m_vs_3m), but their performance gap becomes wider in more challenging maps. Importantly, CBMA outperforms QMIX under the same training parameters, which fully verifies the benefits of inferring common beliefs in CBMA.

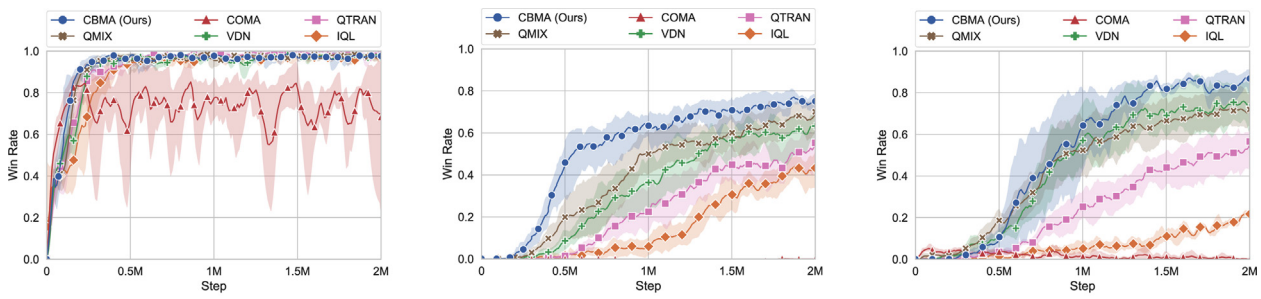
5.3. Ablation studies

We test several ablation models on the 5 m_vs_6m map to demonstrate the effectiveness of our contributions. We first consider the influence of common belief on the final performance. We remove the loss function L_{CB} that forms common beliefs among the agents, namely "Without Common Belief", which means that the variational model is only used for state representation. Moreover, we also evaluate the performance using only one variational network (denoted as "One Stage"). Finally, we test a method to predict the global state s by the decoder model (denoted as "Common State"), where the common belief loss function becomes $L_{\text{VR}} = -\sum_{i=1}^{N} L_i^{\text{elbo}}$, and the generative model in ELBO is rewritten as follows:

$$\mathbb{E}_{q_\phi}\left[\log p_\theta\left(s_t | z_{i,1:t}, x_{i,1:t-1}\right)\right] \simeq \sum_{i=1}^{N} \left(f_\omega(x_i) - s\right)^2 \tag{17}$$

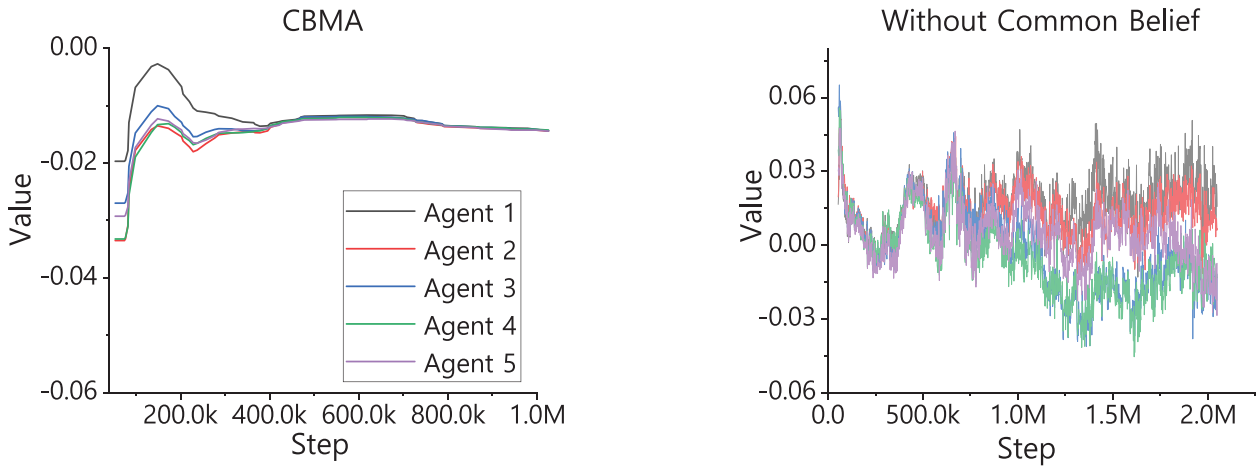
where f_ω is the MLP network.

As can be seen from Fig. 8, agents with common belief can achieve significantly better performance than those without common beliefs, and the performance of a single variational network is much lower than the two-variational network CBMA method, showing that the double network design can overcome the instability of the variational model. Moreover, predicting the global state by the decoder model would hinder the establishment of common beliefs among the agents, leading to inferior performance against the original CBMA method.



(a) 3m_vs_3m. (b) 5m_vs_6m. (c) 8m_vs_9m.

Fig. 6. Comparison of our method against baseline algorithms on the SMAC.



(a) CBMA. (b) Without Common Belief.

Fig. 7. Agents' beliefs $\{z_i\}_{i=1}^{N}$ in 5 m_vs_6m. These values are obtained by sampling the Gaussian distribution.

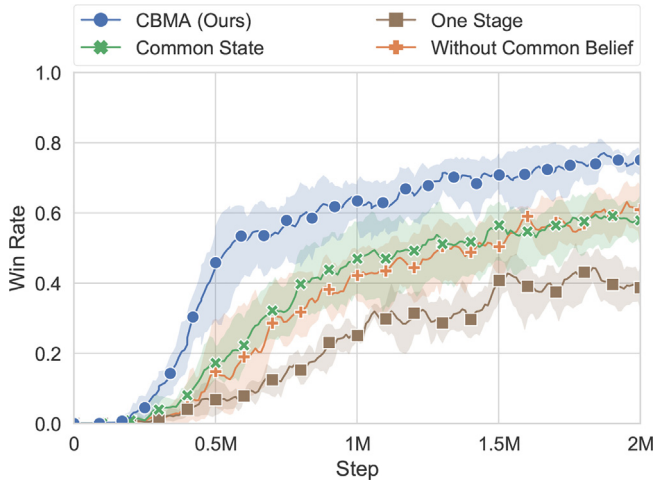


Fig. 8. Ablation reward in 5 m_vs_6m.

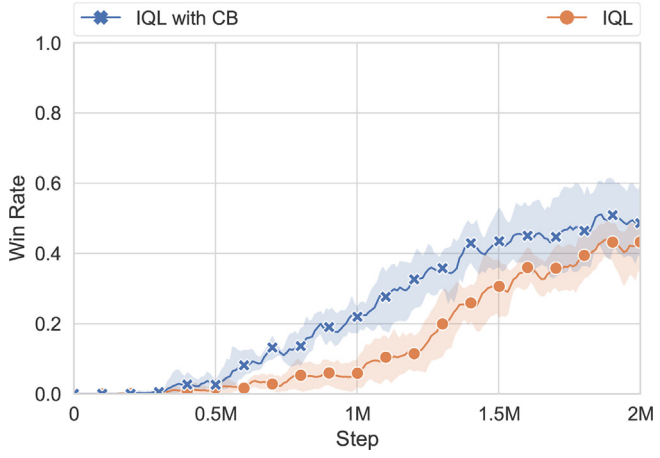


Fig. 9. IQL with CB modul in 5 m_vs_6m.

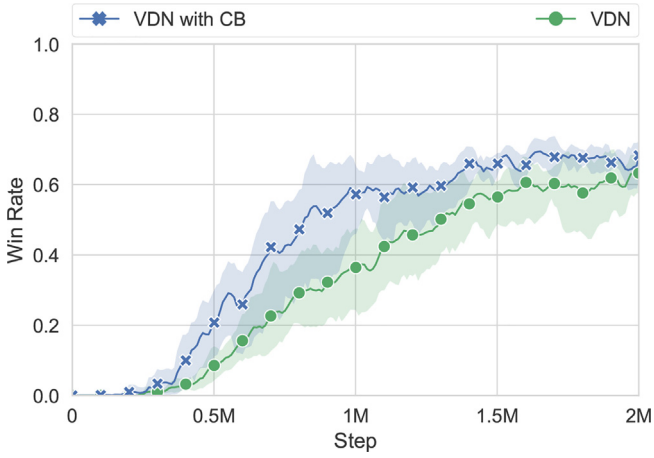


Fig. 10. VDN with CB model in 5 m_vs_6m.

At last, to further prove the effectiveness of common belief (CB), we combine CB module with classic algorithms VDN and IQL. As shown in Fig. 9 and Fig. 10, the convergence speed of the "VDN with CB" and "IQL with CB" algorithms with CB module is faster than that of the classic MARL algorithm VDN and IQL. The performance of the algorithm with CB module is also higher than that of the classic MARL algorithm.

5.4. Further analysis of common belief

To further understand the influence of common belief, we analyze the differences between the beliefs obtained by "Without Common Belief" and CBMA. In Fig. 7, we plot the belief z_i of each agent during the 5 m_vs_6m training process. As shown in the figure, the beliefs of CBMA tend to be more stable with the progress of training, compared with "Without Common Belief". We can see that in the initial stage of 0–500 K, the belief curves between CBMA agents have quickly converged. In contrast, "Without Common Belief" has no consistency tendency in the whole training process. Compared with CBMA's, beliefs are obviously unstable in "Without Common Belief". Meanwhile, in the ablation experiment in Fig. 8, we also find that the CBMA's reward is significantly higher than "Without Common Belief", which proves that common belief can promote the cooperation between agents.

6. Conclusion

In this paper, we propose the CBMA, a novel deep MARL method with decentralised policies and centralised training setting to solve

the belief inference problem for multi-agents systems. Although CBMA is a fully decentralised policy algorithm, agents can still use local observation to obtain consistent beliefs with the help of RNN-based variational model. Benefiting from the variational model, CBMA can handle the dynamic input of the raw sequence observations. In two grid games and the challenging StarCraft II micromanagement, CBMA outperformed the baselines and achieved state-of-the-art utility utilizing common belief. Our experimental analysis demonstrates the flexibility of our framework and the ability to form consistent belief among agents.

In future work, we will consider the dynamic agents to generate local common belief (rather than all agents with a unified belief). We are also interested in exploring how agents can reach consensus through language in the context of communication in order to reduce communication costs in the current communication-based methods.

CRediT authorship contribution statement

Xianjie Zhang: Conceptualization, Methodology, Software, Validation, Writing - original draft. **Yu Liu:** Supervision, Resources, Funding acquisition, Writing - review & editing. **Hangyu Mao:** Methodology, Writing - review & editing. **Chao Yu:** Methodology, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant: 61672128) and the Fundamental Research Fund for Central University (Grant: DUT20TD107). The contact author is Yu Liu.

References

[1] Y. Cao, W. Yu, W. Ren, G. Chen, An overview of recent progress in the study of distributed multi-agent coordination, IEEE Trans. Industr. Inf. 9 (1) (2013) 427–438.
[2] D. Ye, M. Zhang, Y. Yang, A multi-agent framework for packet routing in wireless sensor networks, Sensors 15 (5) (2015) 10026–10047, https://doi.org/10.3390/s150510026.
[3] H. Mao, Z. Gong, Z. Xiao, Reward design in cooperative multi-agent reinforcement learning for packet routing, 2020, arXiv preprint arXiv:2003.03433.
[4] W. Ying, S. Dayong, Multi-agent framework for third party logistics in e-commerce, Expert Syst. Appl. 29 (2) (2005) 431–436, https://doi.org/10.1016/j.eswa.2005.04.039.
[5] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, Y. Ni, Learning agent communication under limited bandwidth by message pruning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 5142–5149.
[6] R. Lowe, Y. Wu, A. Tamar, J. Harb, O.P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Advances in Neural Information Processing Systems (NeurIPS), 2017.
[7] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual multi-agent policy gradients, in: Conference on Artificial Intelligence (AAAI), 2018, pp. 2974–2982.
[8] T. Rashid, M. Samvelyan, C.S. De Witt, G. Farquhar, J. Foerster, S. Whiteson, Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, in: International Conference on Machine Learning (ICML), 2018.
[9] K. Son, D. Kim, W.J. Kang, D.E. Hostallero, Y. Yi, Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning, International Conference on Machine Learning (ICML) 97 (2019) 5887–5896.
[10] A. Mahajan, T. Rashid, M. Samvelyan, S. Whiteson, MAVEN: multi-agent variational exploration, Advances in Neural Information Processing Systems (NeurIPS), 2019.
[11] T. Wang, H. Dong, V.R. Lesser, C. Zhang, ROMA: multi-agent reinforcement learning with emergent roles, in: International Conference on Machine Learning (ICML), 2020, pp. 9876–9886.
[12] H. Wu, J. Zhang, Z. Wang, Y. Lin, H. Li, Sub-avg: Overestimation reduction for cooperative multi-agent reinforcement learning, Neurocomputing 474 (2022) 94–106, https://doi.org/10.1016/j.neucom.2021.12.039.
[13] F. Zhang, J. Li, Z. Li, A td3-based multi-agent deep reinforcement learning method in mixed cooperation-competition environment, Neurocomputing 411 (2020) 206–215, https://doi.org/10.1016/j.neucom.2020.05.097.
[14] N.D. Nguyen, T. Nguyen, S. Nahavandi, Multi-agent behavioral control system using deep reinforcement learning, Neurocomputing 359 (2019) 58–68, https://doi.org/10.1016/j.neucom.2019.05.062.
[15] J.N. Foerster, Y.M. Assael, N. De Freitas, S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, 2016, arXiv preprint arXiv:1605.06676.
[16] J. Jiang, Z. Lu, Learning attentional communication for multi-agent cooperation, arXiv preprint arXiv:1805.07733 (2018).
[17] A. Singh, T. Jain, S. Sukhbaatar, Learning when to communicate at scale in multiagent cooperative and competitive tasks, arXiv preprint arXiv:1812.09755 (2018).
[18] D. Simões, N. Lau, L. Paulo Reis, Multi-agent actor centralized-critic with communication, Neurocomputing 390 (2020) 40–56, https://doi.org/10.1016/j.neucom.2020.01.079.
[19] H. Mao, Z. Gong, Y. Ni, Z. Xiao, Accnet: Actor-coordinator-critic net for learning-to-communicate with deep multi-agent reinforcement learning, 2017, arXiv preprint arXiv:1706.03235.
[20] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, Y. Ni, Learning multi-agent communication with double attentional deep reinforcement learning, Auton. Agent. Multi-Agent Syst. 34 (1) (2020) 1–34.
[21] B. Corgnet, A.M. Espín, R. Hernán-González, The cognitive basis of social behavior: cognitive reflection overrides antisocial but not always prosocial motives, Front. Behav. Neurosci. 9 (2015) 287, https://doi.org/10.3389/fnbeh.2015.00287.
[22] H. Mao, W. Liu, J. Hao, J. Luo, D. Li, Z. Zhang, J. Wang, Z. Xiao, Neighborhood cognition consistent multi-agent reinforcement learning, in: Conference on Artificial Intelligence (AAAI), 2020.
[23] C. Schroeder de Witt, J. Foerster, G. Farquhar, P. Torr, W. Boehmer, S. Whiteson, Multi-agent common knowledge reinforcement learning, Advances in Neural Information Processing Systems (NeurIPS), 2019.
[24] A.X. Lee, A. Nagabandi, P. Abbeel, S. Levine, Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems (NeurIPS), 2020.
[25] J. Chung, K. Kastner, L. Dinh, K. Goel, A.C. Courville, Y. Bengio, A recurrent latent variable model for sequential data, Advances in Neural Information Processing Systems (NeurIPS), 2015.
[26] L. Busoniu, R. Babuska, B.D. Schutter, D. Ernst, Reinforcement Learning and Dynamic Programming Using Function Approximators, 1st Edition., CRC Press Inc, USA, 2010.
[27] E. Yang, D. Gu, Multiagent reinforcement learning for multi-robot systems: A survey, Technical Report CSM 04 (06 2004).
[28] X. Zhang, Y. Liu, X. Xu, Q. Huang, H. Mao, A. Carie, Structural relational inference actor-critic for multi-agent reinforcement learning, Neurocomputing 459 (2021) 383–394, https://doi.org/10.1016/j.neucom.2021.07.014.
[29] S. Iqbal, F. Sha, Actor-attention-critic for multi-agent reinforcement learning, in: International Conference on Machine Learning (ICML), 2019.
[30] P. Sunehag, G. Lever, A. Gruslys, W.M. Czarnecki, V.F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J.Z. Leibo, K. Tuyls, T. Graepel, Value-decomposition networks for cooperative multi-agent learning based on team reward, in: International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2018, pp. 2085–2087.
[31] T. Rashid, G. Farquhar, B. Peng, S. Whiteson, Weighted QMIX: expanding monotonic value function factorisation for deep multi-agent reinforcement learning, Advances in Neural Information Processing Systems (NeurIPS), 2020.
[32] S. Iqbal, C.A.S. de Witt, B. Peng, W. Böhmer, S. Whiteson, F. Sha, Ai-qmix: Attention and imagination for dynamic multi-agent reinforcement learning, 2020, arXiv preprint arXiv:2006.04222.
[33] D.P. Kingma, M. Welling, Auto-encoding variational bayes, in: International Conference on Learning Representations (ICLR), 2014.
[34] D.J. Rezende, S. Mohamed, D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in: International Conference on Machine Learning (ICML), 2014, pp. 1278–1286.
[35] M.J. Hausknecht, P. Stone, Deep recurrent q-learning for partially observable mdps, in: Conference on Artificial Intelligence (AAAI), Fall Symposia, 2015.
[36] R.G. Krishnan, U. Shalit, D. Sontag, Deep kalman filters, arXiv preprint arXiv:1511.05121 (2015).
[37] E. Archer, I.M. Park, L. Buesing, J. Cunningham, L. Paninski, Black box variational inference for state space models, arXiv preprint arXiv:1511.07367 (2015).
[38] M. Karl, M. Soelch, J. Bayer, P. van der Smagt, , Deep variational bayes filters: Unsupervised learning of state space models from raw data, in: International Conference on Learning Representations (ICLR), 2017.
[39] M. Fraccaro, S.K. Sønderby, U. Paquet, O. Winther, Sequential neural models with stochastic layers, in: D.D. Lee, M. Sugiyama, U. von Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems (NeurIPS), 2016.
[40] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, T. Sebastian, Probabilistic recurrent state-space models, in: International Conference on Machine Learning (ICML), 2018.

[41] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[42] L.P. Kaelbling, M.L. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artif. Intell. 101 (1) (1998) 99–134.
[43] M. Igl, L.M. Zintgraf, T.A. Le, F. Wood, S. Whiteson, Deep variational reinforcement learning for pomdps, in: International Conference on Machine Learning (ICML), 2018.
[44] D. Han, K. Doya, J. Tani, Variational recurrent models for solving partially observable control tasks, in: International Conference on Learning Representations (ICLR), 2020.
[45] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, T. Weber, Temporal difference variational auto-encoder, arXiv preprint arXiv:1806.03107 (2018).
[46] D.S. Bernstein, R. Givan, N. Immerman, S. Zilberstein, The complexity of decentralized control of markov decision processes, Math. Oper. Res. 27 (4) (2002) 819–840, https://doi.org/10.1287/moor.27.4.819.297.
[47] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533, https://doi.org/10.1038/nature14236.
[48] M. Tan, Multi-agent reinforcement learning: Independent vs. cooperative agents, in: International Conference on Machine Learning (ICML), 1993.
[49] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, R. Vicente, Multiagent cooperation and competition with deep reinforcement learning, PLoS One 12 (4) (04 2017).
[50] F.A. Oliehoek, M.T. Spaan, N. Vlassis, Optimal and approximate q-value functions for decentralized pomdps, J. Artif. Intell. Res. 32 (2008) 289–353.
[51] J. Wang, Z. Ren, T. Liu, Y. Yu, C. Zhang, QPLEX: duplex dueling multi-agent q-learning, in: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021, OpenReview.net, 2021. URL: https://openreview.net/forum?id=Rcmk0xxIQV.
[52] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724–1734.
[53] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, R. Garcia, Incorporating functional knowledge in neural networks, J. Mach. Learn. Res. 10 (42) (2009) 1239–1262.
[54] M. Samvelyan, T. Rashid, C.S. de Witt, G. Farquhar, N. Nardelli, T.G. Rudner, C.-M. Hung, P.H. Torr, J. Foerster, S. Whiteson, The starcraft multi-agent challenge, 2019, arXiv preprint arXiv:1902.04043.
[55] Y. Yang, Y. Wen, J. Wang, L. Chen, K. Shao, D. Mguni, W. Zhang, Multi-agent determinantal q-learning, in: International Conference on Machine Learning (ICML), 2020.
[56] B.A. Sallans, Reinforcement learning for factored markov decision processes, Ph.D. thesis, University of Toronto, CAN, 2002, aAINQ69120.
[57] B. Sallans, G.E. Hinton, Reinforcement learning with factored states and actions, J. Mach. Learn. Res. 5 (2004) 1063–1088.

Xianjie Zhang is currently pursuing the Ph.D. degree with the School of Software Technology, Dalian University of Technology, China. His research interest includes reinforcement learning, multi-agent systems, and their application to taxi fleet optimization.

Yu Liu is currently a full Professor in Software School and Head of Institute of Machine intelligence and Informetrics, Dalian University of Technology, China. He received the Ph.D. degree from Xi'an Jiaotong University, China in 2006 and worked as a postdoctoral research fellow at Department of Computer Science and Technology, Tsinghua Unversity from 2006 to 2008. Professor Liu also worked at Harvard University as a visiting research scientist in 2012. His research interests include swarm intelligence, evolutionary computation, computational intelligence, and Web data mining.

Hangyu Mao received the PhD degree from Peking University, in July 2020. He is currently a researcher with the Huawei Noah's Ark Lab. His research interests include Multi-agent System, Reinforcement Learning, and various intelligent decision-making and reasoning issues. He is a member of the AAAI.

Chao Yu received the Ph.D. degree in computer science from the University of Wollongong, Australia, in 2014. He is currently an Associate Professor with the School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China. He has published more than 70 papers in prestigious journals, such as the IEEE ITS, IEEE TNNLS, IEEE TCYB, and IEEE TVT. His research interests include multi-agent systems, reinforcement learning, and their wide applications in autonomous driving, swarm systems, robotic control, and intelligent healthcare.