REINFORCEMENT LEARNING UNDER A MULTI-AGENT PREDICTIVE STATE REPRESENTATION MODEL: METHOD AND THEORY

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper proposes a new algorithm for learning the optimal policies under a novel multi-agent predictive state representation reinforcement learning model. Compared to the state-of-the-art methods, the most striking feature of our approach is the introduction of a dynamic interaction graph to the model, which allows us to represent each agent's predictive state by considering the behaviors of its "neighborhood" agents. Methodologically, we develop an online algorithm that simultaneously learns the predictive state representation and agent policies. Theoretically, we provide an upper bound of the L_2-norm of the learned predictive state representation. Empirically, to demonstrate the efficacy of the proposed method, we provide thorough numerical results on both a MAMuJoCo robotic learning experiment and a multi-agent particle learning environment.

1 INTRODUCTION

Real-world Multi-agent Systems are considered partially observable since agents do not have a complete perception of the environment states. Learning in this environment is challenging since these partially observable observations bring the noise. The formal model for solving learning in partially observable environments is the partially observable Markov decision process (POMDP) (Kaelbling et al., 1998). However, in the setting where the system model is not known a priori or the model introduced by the domain expert is biased, it is always beneficial and safer to learn system models and develop policies based on the models.

The predictive state representation (PSR) (Littman et al., 2001) is a representation of the dynamic system state by a vector of predictions of tests conditioned on a history. The tests are sequences of actions and observations, which are true if and only if all the observations occur, given that all the actions taken and the histories are sequences of actions and observations recorded before. Compared to POMDP, PSR records a sequence of actions and observations and focuses on observable quantities rather than the system's dynamics.Therefore, compared with POMDP (Hamilton et al.) which tries to learn the hidden-state representation, PSR may be easier to learn. Also, a previous study has demonstrated that PSR is more compact than POMDP (Littman et al., 2001).

Due to the above benefits of PSR, it has been used as the basis of some model-based single-agent reinforcement learning (Boots et al., 2011; James & Singh, 2004; Hefny et al., 2018a). Moreover, the predictive states that act as sufficient statistics for the states of multi-agent dynamic systems could potentially help multi-agent decision-making in the complicated, partially observable domains. However, there is no method for developing multi-agent predictive state representation (MAPSR) for multi-agent reinforcement learning (MARL). Due to this reason, we focused on building an algorithm for learning the optimal policies by representing the multi-agent dynamic systems with the MAPSR.

An existing algorithm by Chen et al. (2020) provides a spectral framework for learning MAPSR using a tensor, with its dimensions represent the agents and joint history. A tensor decomposition is implemented to learn the model parameters. This type of algorithm avoids explicit estimation of the latent state and can get an unbiased estimation if an infinite amount of data is available. However, it is not easy to incorporate prior information such as sparsity or structure because each new source of information leads to new moments and a different and more complex set of equations to solve. Hefny et al. (2015) represent the distributions in PSR via observable statistics and learn PSR by supervised learning, alleviating the above problem. Later, Hefny et al. (2018a) extend this method to develop a

new one called predictive state representation with a controlled model that can represent the system model where the agents can affect the environment. Due to the trend of deep RL, Hefny et al. (2018b) develop an end-to-end policy learning method by using the PSR with controlled model as the belief state to replace the partially observable observation to train RL. Both the PSR and RL parameters are updated by using a combined loss objective. The PSR part maintains a sufficient representation of distributions under the partially observable environment, the deep RL algorithm brings efficiency for learning.

The environment of MARL can be partially observable and always non-stationary (one agent learning while the other agents' actions can influence the environment) Bowling & Veloso (2002). The predictive states that act as sufficient statistics for the states of multi-agent dynamic systems could potentially help multi-agent decision-making in a complicated, partially observable domain. PSR with controlled model can model the prediction of systems states affected by actions; if we design a state representation model for MARL environment that can predict the future observations of agents upon exerting future actions of other agents and conditioned on past histories, then the non-stationary environment caused by other agents learning can be modeled. Due to the above reason, we seek to develop a MARL-based PSR model and learning algorithm, which we call MAPSR. It could potentially ameliorate both the problem of partially observable observations and non-stationarity by other agents' interaction for multi-agent decision making.

The paper's first goal is to develop a PSR model for MARL environment. In MARL settings, the distribution of observed observations is dependent on its actions and relevant to other agents and histories. So far, many works (Ryu et al., 2019; Liu et al., 2019b; Niu et al., 2021) that take other agents' actions into consideration have been shown to achieve competitive performance. In order to consider the interactive relationship under the MAPSR, we introduce a dynamic interaction graph to characterize the "neighborhoods" of agents. Such a graph topology, which is often easy to obtain nowadays, enables us to encode the adjacency relationship of agents into our MAPSR model. We build our model under three common graphs, namely static complete graphs, static non-complete graphs, and dynamic graphs, covering as many real-world scenarios as possible. Under the existence of such a dynamic interaction graph, we build predictions of its observations by considering its neighbor's actions for every agent. We call this primitive interactive predictions. Then an agent's PSR will be formulated as a linear combination of those primitive interactive predictions. Due to the decentralized formulation, our model allows implementing the learning of the MAPSR on a single agent level while maintaining other agents' interactions through the encoding of the interactive graph topology.

As set as our second goal, we analytically upper bound the L_2-norm difference between the underlying true and learned predictive state representations, providing theoretical guarantees for the performance of our learning algorithm under the MAPSR model in all three graph settings.

Our paper's third goal is to develop an algorithm that integrates the MAPSR model to the MARL. We develop an online algorithm that simultaneously learns the MAPSR, and the agents' policies end-to-end with a fully differentiable neural network structure. Our algorithm provides a new model-based MARL, and it has the flexibility to be extended to most of the offline MARL algorithms. We focus on the continuous action space, which is more practical and common to real-world applications such as robotic control.

Finally, we test our algorithm through systematic numerical experiments on MAMujoCo robotic learning experiments (de Witt et al., 2020) and multi-agent particle learning environments (Ryu et al., 2020), and compare our proposed method against two baselines as detailed in Section 6. The results demonstrate the efficiency of our method.

2 Related Works

Partially Observable Environment. Real-world agents often experience situations that the observed signals are aliased and do not fully determine their state in the system. This is particularly true for multiple agents environments where agents have partial observability due to limited communication Oliehoek & Amato (2016). In accommodation to the partially observable environment, POMDP has been adopted by Kaelbling et al. (1998), and the algorithms for determining an optimal policy have shifted to using the probability distributions over the state-space instead of exact state space. The most common POMDP policy learning assumes the agent has access to a priori knowledge of the system. The access to such prior knowledge has a premise that the agent has considerable domain

knowledge Kaelbling et al. (1998). However, it is also expected that the real-world agents do not know the domain to learn the system model to learn a planning policy further.

Overview of PSR. Littman et al. (2001); James & Singh (2004) introduced the PSR over an expressive and robust framework for modeling dynamical systems and defined PSR as a representation of state by using a vector of predictions of fully observable quantities (tests) conditioned on past events (histories). A predictive model is constructed directly from execution traces in the PSR framework, utilizing minimal prior information about the domain. The PSR paradigm subsumes POMDPs as a special case Littman et al. (2001). PSR is considered much more compact than POMDP Aberdeen et al. (2007). The spectral learning method has been proved to show success for learning the PSR Boots et al. (2011); Jiang et al. (2018). There are other classes of dynamical system learning algorithms that are based on likelihood-based optimization or sampling approaches Frigola et al. (2013), but they are prone to poor local optima. The spectral learning represents the estimated state by sufficient statistics of future observations and estimates the model parameters by method of moments. However, this line of algorithms is hard to incorporate prior information Hefny et al. (2015). Thus, Hefny et al. (2015; 2018a) introduce the supervised learning method to learn PSR and proves its convergence. Although many works study PSR in discrete action space Hsu et al. (2012); Siddiqi et al. (2010); Boots et al. (2011). Boots et al. (2013) proposes Hilbert space embedding (HSE)-PSR to deal with continuous actions. Hefny et al. (2018a) uses an approximation of HSE-PSR by Random Fourier transform (RFF) and built a more principled generalization of PSR to deal with high dimensions. However, all of these studies are aimed at the single agent.

PSR and RL. The predictive states estimated by the PSR are considered as states in a fully observable Markov Decision Process so that the value function is learned on these states. This line of work has been done in the single-agent environment Boots et al. (2011); Hefny et al. (2018b). Especially, Hefny et al. (2018b) proposes the recurrent predictive state in the RNN network. Moreover, the learning PSR and policy functions are connected with the end-to-end training.

MAPSR. MAPSR model is formulated by Chen et al. (2020) and the model parameters are learnt by tensor decomposition. This formulation is aligned with the spectral algorithm, so it also has the same issue as the single-agent case. MARL algorithms have been developed to learn the Markov Game environment Lyu & Amato (2020); Son et al. (2019); Zhang et al. (2019); Rashid et al. (2018); Foerster et al. (2017). The partially observable states are pervasive in this environment, bringing non-stationarity to the learning algorithm Bowling & Veloso (2002) so that the learning needs to consider the non-stationarity. Recent works propose more sophisticated deep MARL algorithms for multi-agent problems under the paradigm of centralized training with decentralized execution Sunehag et al. (2017); Foerster et al. (2017); Rashid et al. (2018); Zhou et al. (2020). Other than their methods, our approach also accounts for the importance of macroscopic measures of dynamic underline systems. Also, given with a predictive state as belief state, rather than the partially observable observations could further help the agent learning. Like the single-agent case, we hope our method could further improve these algorithms.

3 Model Description of MAPSR

3.1 Review of Single Agent PSR

Predictive state representation (PSR). A prediction of a state is defined as the conditional probability of seeing a test's observations in sequence given that actions of the test are taken in sequence from a history Littman et al. (2001). Givn a finite observation space $\mathcal{O}$ ($o \in \mathcal{O}$) and action space $\mathcal{A}$ ($a \in \mathcal{A}$). A **test** of length k at time t, is defined as a sequence of action-observation pairs that starts at time t and ends at time $t+k-1$, $\{(o_l, a_l)\}_{l=t}^{t+k-1} = \{o_t, a_t, o_{t+1}, a_{t+1}, \cdots, o_l, a_l, \cdots, o_{t+k-1}, a_{t+k-1}\}$. A **history**, at time t is a sequence of action-observation pairs that starts from the beginning of time and ends at time $t-1$, $\{(o_l, a_l)\}_{l=1}^{t-1} = \{o_1, a_1, \cdots, o_l, a_l, \cdots, o_{t-1}, a_{t-1}\}$.

Hilbert Space Embedding Predictive State Controlled Model. In this work, we are interested in extending PSR to decision makings of controlled systems with continuous actions. So we use the model introduced by Hefny et al. (2018a). In this model, predictive state Q_t satisfies $Q_t \psi_t^a = \mathbb{E}[\psi_t^o | \psi_t^a; \psi_t^h]$ and extended predictive state P_t satisfies $P_t \xi_t^a = \mathbb{E}[\xi_t^o | \xi_t^a; \psi_t^h]$, where $\psi_t^o := \phi_O(\{o_l\}_{l=t}^{t+k-1})$ and $\psi_t^a := \phi_A(\{a_l\}_{l=t}^{t+k-1})$ are feature maps by kernels k_O, k_A over future observation and action features. The extended predictive state compared to predictive state adds one more pair of $\{(a_{t+k}, o_{t+k})\}$ to the prediction. The ξ_t^a and ξ_t^o are the corresponding extended feature

maps, which satisfy $\xi_t^o = \psi_t^o \otimes \phi_t^o$ and $\xi_t^a = \psi_t^a \otimes \phi_t^a$. Here $\phi_t^o := \phi_o(o_t)$ and $\phi_t^a := \phi_a(a_t)$ are the shorthands for one time feature map by k_o and k_a. Also, we use the $\psi_t^h := \psi^h(\{(o_l, a_l)\}_{l=1}^{t-1})$ to define a set of features extracted from previous observations and actions (typically from a fixed length window ending at $t-1$).

3.2 MAPSR Formulation

For n-agent systems, we denote joint observations as $\{o_{t,i}\}_{i=1}^n$ and joint actions as $\{a_{t,i}\}_{i=1}^n$ at any time t. The joint test is then a collection of tests for all agents $\{(o_{l,1}, a_{l,1}), \cdots, (o_{l,n}, a_{l,n})\}_{l=t}^{t+k-1}$ and the joint history is a collection of histories for all agents $\{(o_{l,1}, a_{l,1}), \cdots, (o_{l,n}, a_{l,n})\}_{l=1}^{t-1}$. A prediction of multi-agent state is thus the conditional probability of a joint test given a joint history.

Our Formulation of MAPSR. We bring the HSE-PSR model to the multi-agent settings. For any agent i in an n-agent system, given kernels $\{k_{O_i}\}_{i=1}^n$,$\{k_{A_i}\}_{i=1}^n$,$\{k_{o_i}\}_{i=1}^n$,$\{k_{a_i}\}_{i=1}^n$, we use $\psi_{t,i}^o$, $\psi_{t,i}^a$, $\xi_{t,i}^o$, $\xi_{t,i}^a$, $\phi_{t,i}^o$, $\phi_{t,i}^a$ and $\psi_{t,i}^h$ to denote its corresponding feature maps, same as the previous definitions for single agent. The superscripts o, a, h are local observations, actions, and histories for agent i. And we use the $Q_{t,i}$ and $P_{t,i}$ to denote its predictive state and extended predictive state.

3.3 The Estimation for MAPSR Model Components

As stated by Littman et al. (2001), a complete PSR model can build a recursive rule to update itself. In other words, if we are given Q_t, we can get Q_{t+1} via the Bayes' rules. We use a supervised learning method proposed by Hefny et al. (2018a) with three types of models to update the predictive states: For any agent i, $P_{t,i} = W_i(Q_{t,i})$, $Q_{t+1,i} = F_i(P_{t,i}, o_{t,i}, a_{t,i})$, and $o_{t,i} = Z_i(Q_{t,i}, a_{t,i})$. Typically, W_i, Z_i are learnable linear maps, and F_i is non-linear but known in advance. Since the model components $Q_{t,i}$ and $P_{t,i}$ are operators so they have to be learned as well. The W_i, Z_i can be learned from regression if we know the estimation of $Q_{t,i}$ and $P_{t,i}$, see Appendix A.3.2 for details of estimation of W_i, Z_i. We focus on the details of estimation of $Q_{t,i}, P_{t,i}$ to section 4.1.1.

4 Dynamic Interaction Graph For MAPSR Model

Many works have considered graph representation of the multi-agent network Liu et al. (2020); Ryu et al. (2020). In general, the relationship between agents is characterized as an undirected graph. We introduce a dynamic interaction graph to represent the MAPSR by considering the interaction between agents.

Definition 1. *Let $G = (V, E)$, including the set V of nodes and set E of the edges. Each node represents the agent entry, and the edge represents the relationship between the two adjacent agents.*

Here we suppose the graph structure is given, that means the number of nodes (n), the number of edges (m), the edge weights, and the maximum number of degrees (k) are available to us. We think that this kind of presupposition is very reasonable because in the real world, for example, there are multiple robots; we can quickly get the geographic position of the robots through sensors, then calculate the structure of the graph formed by them.

4.1 Static Complete Graph

We starts with a static complete graph G_c, where the relationship between nodes are invariant to time change. A complete graph has $m = n(n-1)/2$ edges, where $m = |E|$. For each agent, we then represent its PSR and extended PSR by considering other agents' interactions by

$$Q_{t,i} \quad := g\left(\{Q_{t,i,j}\}_{j=1}^n\right) = \textstyle\sum_j Q_{t,i,j} \tag{1}$$

To consider the interactive behavior between agents, we introduce two additional notations $Q_{t,i,j}$ and $P_{t,i,j}$. Let $\{(i,j)\}_{i,j=1}^n$ represents a pair of agents on a n multi-agent system. The $Q_{t,i,j}$ is a primitive predictive state of i's observation ($\psi_{t,i}^o$) by intervening agent j's action ($\psi_{t,j}^a$) and observing agent i's observation history and agent j's action history ($\psi_{t,i,j}^h$). And $P_{t,i,j}$ is the extended counterpart. If $i == j$, then it becomes an exact single agent scenario. Similarly, we use the same approach to represent $P_{t,i}$. Each agent's PSR $Q_{t,i}$ and extended PSR $P_{t,i}$ are modeled by fully considering all other available agents.

Based on equation 1, in practice, estimating $Q_{t,i}$ and $P_{t,i}$ denoted as $\widehat{Q}_{t,i}$ and $\widehat{P}_{t,i}$ requires us to get $\widehat{Q}_{t,i,j}$ and $\widehat{P}_{t,i,j}$ at first.

4.1.1 ESTIMATION OF $Q_{t,i,j}$, $P_{t,i,j}$

To estimate the P_t and Q_t for single agent, Hefny et al. (2018a) use the supervised learning method. They show that $\widehat{Q}_t = M_{\psi_t^o|\psi_t^a;\psi_t^h} = C_{\psi_t^o\psi_t^a|\psi_t^h}(C_{\psi_t^a\psi_t^a|\phi_t^h}+\lambda I)^{-1}$, where $M_{A|B;c}$ is a linear operator that satisfies $\mathbb{E}[A|B=b;C=c] = M_{A|B;c}b$, and $C_{XY} := \mathbb{E}[\phi(X)\otimes\phi(Y)]$ is the uncentered covariance operator, and $C_{XY|z}$ is covariance of X and Y given $Z=z$. They estimate $C_{\psi_t^o\psi_t^a|\psi_t^h}$, $C_{\psi_t^a\psi_t^a|\phi_t^h}$ by sampling data, then get the estimation $\widehat{Q}_t$. The same procedure is used to estimate $\widehat{P}_t$ by replacing the features ψ_t^o and ψ_t^a with their extend counterparts ξ_t^o and ξ_t^a.

Similar to the single-agent case above, the representation of $Q_{t,i,j}$ can be achieved as

$$\widehat{Q}_{t,i,j} = M_{\psi_{t,i}^o|\psi_{t,j}^a;\psi_{t,i,j}^h} = C_{\psi_{t,i}^o\psi_{t,j}^a|\psi_{t,i,j}^h}\left(C_{\psi_{t,j}^a\psi_{t,j}^a|\phi_{t,i,j}^h}+\lambda I\right)^{-1} \tag{2}$$

To estimate $C_{\psi_{t,i}^o\psi_{t,j}^a|\psi_{t,i,j}^h}$ and $C_{\psi_{t,j}^a\psi_{t,j}^a|\psi_{t,i,j}^h}$, we learn two linear maps $T_{i,j}$ and $U_{i,j}$ such that $T_{i,j}\left(\psi_{t,i,j}^h\right) \approx C_{\psi_{t,i}^o\psi_{t,j}^a|\psi_{t,i,j}^h}$ and $U_{i,j}\left(\psi_{t,i,j}^h\right) \approx C_{\psi_{t,j}^a\psi_{t,j}^a|\psi_{t,i,j}^h}$. The training examples for $T_{i,j}$ and $U_{i,j}$ consist of pairs $(\psi_{t,i,j}^h, \psi_{t,i}^o\otimes\psi_{t,j}^a)$ and $(\psi_{t,i,j}^h, \psi_{t,j}^a\otimes\psi_{t,j}^a)$. The learning of $\widehat{P}_{t,i,j}$ can be done in a similar way. More details can be found in Appendix A.3.1

After calculation of $\widehat{Q}_{t,i,j}$ and $\widehat{P}_{t,i,j}$, under the static complete graph setting where the interaction is considered, defined in equation 1, we can get the estimate of $\widehat{Q}_{t,i}$ and $\widehat{P}_{t,i}$.

4.1.2 THEORETICAL GUARANTEE OF ESTIMATION OF Q_i, P_i UNDER THE STATIC COMPLETE GRAPH

Theoretically, we show that the difference between Q_i and its estimator $\widehat{Q}_i$ is bounded with high probability.

Theorem 1. *Let π_Θ be a data collection policy and $\mathcal{H}$ is the range of π_Θ on joint histories. If Equation 1 and 2 used, then for all $h\in\mathcal{H}$ and any $\epsilon\in(0,1)$, such that $N > \frac{t_{A_j}^2\log(2d_{A_j}/\epsilon)}{v\left(C_{\psi_j^a}\right)}$ where N is the number of time points we collect sample, then $\left\|\widehat{Q}_i - Q_i\right\|$ is bounded as below with probability at least $1-3\epsilon$.*

$$\left\|\widehat{Q}_i - Q_i\right\| \le n\Delta \tag{3}$$

where

$$\Delta = \sqrt{\frac{u\left(C_{\psi_i^o|\psi_{i,j}^h}\right)}{v\left(C_{\psi_j^a|\psi_{i,j}^h}\right)^3}\cdot\frac{\|\Delta_1\|^2+2u\left(C_{\psi_j^a|\psi_{i,j}^h}\right)\|\Delta_1\|+\lambda}{v\left(C_{\psi_j^a|\psi_{i,j}^h}\right)(1-\gamma)+\lambda}} + \frac{\left\|C_{\psi_i^o\psi_j^a|\psi_{i,j}^h}\right\|\|\Delta_1\|+\|\Delta_2\|\left\|C_{\psi_j^a|\psi_{i,j}^h}\right\|+\|\Delta_2\|\|\Delta_1\|}{v\left(C_{\psi_j^a|\psi_{i,j}^h}\right)^2(1-\gamma)^2+\lambda}$$

Here Δ_1, Δ_2 are two other relevant bounds, we provide them in Appendix E.1. $u(\cdot), v(\cdot)$ denote the largest, smallest eigenvalue of a matrix. And $\gamma = \frac{t_{A_j}^2\log(2d_{A_j}/\epsilon)}{v\left(C_{\psi_j^a}\right)N} < 1$ is a constant depends on the magnitude of the norm of ψ_j^a (we assume $\left\|\psi_j^a\right\| \le t_{A_j}$), the dimension of ψ_j^a (d_{A_j}), the (uncentered) covariances ($C_{\psi_j^a} := \mathbb{E}\left[\psi_j^a\psi_j^{aT}\right]$) and the sample size ($N$).

We consider a sensor collects measurements (samples) from finite time points with the same gap between two time points. Theorem 1 says we need at least N sample for the bound in equation 3 to be valid. We give the proofs in Appendix E. It is not hard to obtain a bound for P_i using the same approach. We omit that.

4.2 EXTENSION TO STATIC NON-COMPLETE GRAPH

In large-scale multi-agent systems, the number of agents is large, and not all agents need to interact with each other. A static non-completed graph can perfectly represent such a situation. In a given static non-complete graph G_s, we know its maximum number of degree k, and we use the binary $n\times n$ matrix with each entry as $I_{i,j}$ to indicate the interaction between two agents. Then the MAPSR for each agent will be

$$Q_{t,i} := g\left(\{Q_{t,i,j}\}_{j=1}^n\right) = \sum_j I_{i,j}Q_{t,i,j}. \tag{4}$$

Lemma 1. *Under the same environment depicted in Theorem 1 and given a G_s with k maximum number of the degree to represent agents, then the bound in Theorem 1 can be rewritten as* $\left\|\widehat{Q}_i - Q_i\right\| \leq k\Delta$.

The conclusion of Lemma 1 is evident since we replace the n with the k neighbors, the total error bound is also decreased approximately as $\frac{k}{n}$.

4.3 Dynamic Graph

Real-world multi-agents can also formulate a time-dependent dynamic graph G_d, rather than a static graph. A dynamic graph has its structure dynamically changing with time. In other words, the edges can be inserted or deleted across time. The dynamic graph brings more challenges to the representation as to the interaction relationship among agents changes constantly.

Braha & Bar-Yam (2009), Ma et al. (2017) and Zhao et al. (2010) consider a dynamic graph as a set of ordered static graphs. For each time point, we are given a static graph such that $G_d = \{G_{d1}, G_{d2}, \cdots, G_{dt}\}$, and a time-dependent given binary matrix with $I_{t,i,j}$ indicating the interaction between two agents at each time. Then we have the agent-wise PSR as

$$Q_{t,i} := g\left(\{Q_{t,i,j}\}_{j=1}^{n}\right) = \sum_j I_{t,i,j} Q_{t,i,j}. \tag{5}$$

Compared to (4), the coefficient $I_{t,i,j}$ is time-dependent, which brings a challenge to our theoretical bound. We consider a dynamic graph experiences a trajectory path, assume every node has a chance at least p to interact by connection with another node at any time point. For a node i, if we take the union set of the nodes interacted with i over the path, then the union set could form a static complete graph, in other words, if i connects j, then we can obtain a valid sample to estimate $Q_{i,j}$ as the complete static graph does, if not, then we skip to the next time point. For the complete static graph, we need the trajectory to run at least N time points to collect enough data to estimate our conditional operator accurately $Q_{i,j}$. Furthermore, The total number of time points needed by i until the Nth interaction with j follows a negative binomial distribution $\mathcal{NB}(N, p)$. On average, we need $\frac{N}{p}$ time points before we see i, j completely connecting N times. Now we consider node i could interact with every node in a set of nodes ($J : |J| = n - 1$) for N number of time points. We assume the interaction between two nodes ($i, j \in J$) does not affect their interaction with other nodes. Thus we have a set of independent negative binomial random variables $\{J_l\}_{l=1}^{n-1} \sim \mathcal{NB}(N, p)$ to characterize the interaction of i with $j \in J$. So we are interested in the expectation of the maximum of $J_1, \ldots, J_{n-1}$, a statistics that tells us the expected maximum number of time points of collection of measurements needed for the node i to be able to connect with every node $j \in J$ for at least N number of time points. We denote it as $J_{\{1,\ldots,n-1\}}$ and we have

$$\mathbb{E}\{J_{\{1,\ldots,n-1\}}\} = \mathbb{E}\{\max(J_1, \cdots, J_{n-1})\} = \sum_{N \geq 0} \left(q^N + Npq^{N-1} + \cdots + \binom{N}{N-1} p^{N-1} q\right)^{n-1}. \tag{6}$$

Lemma 2. *Under the same environment depicted in 1 and given a dynamic graph G_d with every node has a chance at least p to interact with another node in one-time point, let N be the number of time points we collect data of measurements in order to get the bound in Theorem 1, if Equation 5 and 2 used, then we need at least*

$$N' = \left(N - \frac{1}{2}\right) + K(q, n, N) - \frac{\gamma}{\log_{1/q}(1/q)} + F\left[K(q, n, N)\right] + \mathcal{O}(1), \tag{7}$$

total number of time points, where $q = 1 - p$,
$K(q, n, N) := \log_{1/q}(n-1) + (N-1)\log_{1/q}\left[\log_{1/q}(n-1)\right] + (N-1)\log_{1/q} p - \log_{1/q}(N-1)!$,
F is a periodic C^∞-function of period 1 and mean value 0 whose Fourier-coefficients are given by $\hat{F}(k) = -\frac{1}{\log\left(\frac{1}{q}\right)}\Gamma\left(-\frac{2k\pi i}{\log\left(\frac{1}{q}\right)}\right)$ for $k \in \mathbb{Z} \setminus \{0\}$. Then $\left\|\widehat{Q}_i - Q_i\right\|$ achieves the same bound as in Theorem 1 with probability at least $1 - 3\epsilon$, in other words, $\left\|\widehat{Q}_i - Q_i\right\| \leq n\Delta$. (7) is an asymptotic expansion of the right-hand of (6). We give the proof in Appendix E.3.

Lemma 2 gives the worst-case bound for our estimation under the dynamic graph. The result tells us that if we need 1 more sample of measurement for our algorithm to converge on the complete static graph, we need roughly $\log_{1/q}\log_{1/q}(n-1)$ more samples on the dynamic graph. So as long as we allow enough learning time, the algorithm can converge with high probability.

Complexity with Increasing Agents: For the complete static graph, we need evaluate $\widehat{Q}_{i,j}$ and $\widehat{P}_{i,j}$ every time point, which requires $\mathcal{O}(n^2)$ operations and space. Overall, with an increased number of agents, our MAPSR has a polynomial $\mathcal{O}(n^2)$ scaled complexity, which is feasible for learning in a large number of agents environment, and is more efficient compared to the centralized MAPSR Chen et al. (2020) theoretically, which is combinatorially sample complex, the analysis is shown in Appendix A.2. For a non-complete static graph with k maximum number of degrees for $k \ll n$, which is more common in the real world, because a very far-away robot will not likely affect the targeted robot, the operation will be significantly decreased to $\mathcal{O}(k^2)$.

After we get the $\widehat{Q}_{t,i}$ and $\widehat{P}_{t,i}$, we build the update rule by using W_i, F_i, Z_i, as described in section 3.3.

5 Decision-Making Framework With MAPSR Model

Previous work on single-agent proposed an end-to-end training algorithm Hefny et al. (2018a) for PSR model and policy learning. Here we design an algorithm for multi-agent settings and incorporate our MARPSR model contains the interactive graph component.

We propose an online learning algorithm to learn the MAPSR and agent policies simultaneously. Our algorithm is shown in Appendix B.1. We call the algorithm **MAPSRL**. The algorithm can be divided into two parts. Firstly, we have a state tracking component, which lets the agents explore the environment and learn an initial predictive state and its parameters W_i and Z_i. The agents first enter the initialization phase (Line 1-4), where we initialize MAPSR by executing an exploration policy to collect data. The agents explore the environment and learn the MAPSR parameters under the given interactive graph. Secondly, we also have a policy component, in which, by starting from the initialized MAPSR and a random policy, the agents experience a T length trajectory under the current policy function that maps predictive states $\{Q_{t,i}\}_{i=1}^n$ to actions $\{a_{t,i}\}_{i=1}^n$ (line 5-18). The agents then update to the next predictive state as introduced in subsection 3.3. Agents also save trajectories (actions, observations, predictive states, next predictive states, and rewards) over the path. Finally, a MARL algorithm conducts the policy learning component. We use a diagram in Appendix B.2 to illustrate this process. We update the MARL and MAPSR model parameter $\mathbf{\Theta} = \{\Theta_{\text{MAPSR}}, \Theta_{\text{MARL}}\}$ (line 19) by minimizing the following objective function.

$$\begin{aligned} L(\Theta) = \quad & \alpha_1 l_1(\Theta_{\text{MARL}}) + \alpha_2 l_2\left(\Theta_{\text{MAPSR,MARL}}\right) \\ = \quad & \alpha_1 l_1(\Theta_{\text{MARL}}) + \alpha_2 \frac{1}{n} \sum_{t=0}^{T} \sum_{i=1}^{n} E_{p(\tau_i|\Theta)} \left[\left\| Z_i \left(F_i(W_i(g\left(Q_{t-1,i,j}\right))) \otimes \psi_{t,i}^a \right) - \psi_{t,i}^o \right\|^2 \right] \end{aligned} \tag{8}$$

Here the $l_1(\Theta_{\text{MARL}})$ is the loss for MARL, and $l_2(\Theta_{\text{MAPSR,MARL}})$ is the MSE between prediction and actual observation. And $p(\tau_i|\Theta)$ is the distribution over trajectories induced by the policy and MAPSR. $\Theta_{\text{MAPSR}} = \{W_i, Z_i\}_{i=1}^n$ denotes MAPSR's parameters, and Θ_{MARL} denotes the MARL part. Usually, in a continuous environment, each agent's policy will be parameterized by its actor network that outputs the mean and diagonal covariance of a Gaussian distribution over the continuous action space. We could also give a parameter sharing actor network that maps individual PSR to parameters of a Gaussian distribution over the individual action space if agents are homogeneous.

Now we devise our MAPSR based MARL algorithm in detail. Our algorithm is on the actor-critic framework. We first develop an algorithm under partially observable environments, where each agent has its own critic and actor independently, the gradient of the policy is written as $\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|Q_i)(r_i + \gamma V^{\pi_i}(Q_{t+1,i}) - V^{\pi_i}(Q_i))]$, and the independent critic is updated by minimizing the loss $\mathcal{L} = \mathbb{E}[(V(Q_i) - y_i)^2]$. We replace the partially observation o with PSR Q. We call it **MAPSRL-1**. As we know, independent actor-critic (IAC) Foerster et al. (2017) would not work well since the environment is not stationary under multi-agent setting, the MADDPG by Lowe et al. (2017) solves a non-stationary environment by considering other agents actions; however, every agent needs a separate critic that has the global information. Instead, we use a centralized critic like methods COMA Foerster et al. (2017), LICA Zhou et al. (2020) to save space. Our centralized critic minimizes the loss $\mathcal{L} = \mathbb{E}[(V^{\pi}(Q_1, \cdots, Q_n) - y)^2]$. Unlike MADDPG using the deterministic policy, the gradient is taken on the policy parameter space, which is reflected in LICA Zhou et al. (2020). So our policy gradient is written as $\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|Q_i) \nabla_{\theta_i} V^{\pi}(Q_1^{\pi_{\theta_1}}, \cdots, Q_n^{\pi_{\theta_n}})]$, Also, to solve the non-stationary environments when each agent is learning, it uses the joint predictive states as input. The benefits are 1. PSR (Q) is a function of action (a), bringing agents' action information into the policy gradient, 2. Also, our interactive MAPSR brings the effect of other agents' actions to the policy gradient, so it does not learn a deterministic policy like MADDPG rather a policy that

affects the predictions of future observations considering other agents' actions 3. Without the need to input other agents' actions, which saves the input space since the predictive state is a conditional operator that considers other agents' actions. We call it **MAPSRL-2**. We put our integration details in Appendix B.3.

6 EXPERIMENTS

Environments. We evaluate the performance of our MAPSRL on a collection of MARL tasks under some OpenAI Gym MAMujoco environments de Witt et al. (2020), such as multi-agent swimmer, hopper, and ant. Each robotic agent is represented as a body graph, where vertices (joints) are connected by adjacent edges (body segments) as shown in Appendix Figure 5. Each agent controls its joints based on the local information observed. All tasks are learned under the partially observable environments by manually hiding some observations for each agent. The goals of the multi-agent systems are aligned with their corresponding single-agent ones. However, different from the single-agent system, the agents in the multi-agent system need to collaborate to reach their goals. For the interactive graph, for simplicity, we considered a complete static graph, where we assume every agent is connected with all other agents in the graph. We defer the details of the robotic agents to Appendix C.1 and the details of the experimental setup to Appendix C.2.

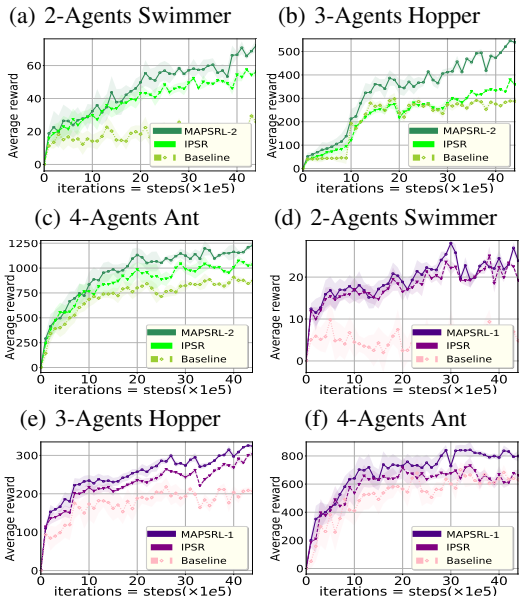

Figure 1: Performance of our method under MAMujoco partially observable environments, the ablation study– IPSR, in which we do not consider the agent interaction, and the baseline, which is not using MAPSR. (a)-(c):MAPSRL-2 (It uses centralized critic, and uses gradients of value function with respect to policy parameter, its baseline is MADDPG), (d)-(f):MAPSRL-1 (Its baseline is IAC). We run 10 times and the shaded area is the 95% confidence interval

Baselines and Evaluation. We run 50 iterations for each experiment and collect $M = 100$ trajectories in every iteration with a maximum of 1000 steps in every trajectory. After each iteration, we compute the average return $R = \frac{1}{M}\sum_{i=1}^{n}\sum_{b=1}^{M}\sum_{t=1}^{T_b} r_{i,b}^{t}$ on a batch of M trajectories, where, T_b is the length of the bth trajectory. We repeat this process using ten different random seeds and report the average and a standard deviation. To verify the effectiveness of interactive graph, we introduce a baseline called Independent PSR learning (IPSR); in this model, we do not consider the graph, so we formulate n independent single PSR without considering their interactions, which means there are no $Q_{i,j}$ any more but only Q_i. The architecture of MAPSRL-1 and MAPSRL-2 remains the same. To verify the advantage of PSR, we introduce another baseline (MARL) where we take out the MAPSR entirely, so it matches with the MARL run on a partially observable environment. For MAPSRL-1, its MARL baseline is IAC Foerster et al. (2017), and for MAPSRL-2, is MADDPG Lowe et al. (2017).

Results And Discussion Figure 1 illustrates the empirical average return vs. the number of interactions with the environment measured in time steps. Our MAPSRL methods consistently outperform IPSR and get the highest rewards under partially observable environments, which justifies the representation power of the interactive graph to assist agents in learning in the non-stationary environment with limited observation when the MARL algorithm has a defect (IAC). Moreover, it can also boost the performance of existed good MARL algorithm (MADDPG). To further verify the effect of learning the PSR part, we also plotted the predicted trajectory to verify the MAPSR's performance for predicting the observations in Figure 2. We plotted the predicted observations vs. actual observations in iterations 1, and 40, respectively, for MAPSRL-2. We plotted a row $\times$ columns figure, with each row representing the observation feature and each column representing each agent. As we can see from the figures, the first iteration of the learning does not predict the actual observation very well; it has some mismatches. However, as learning progresses, the predictions

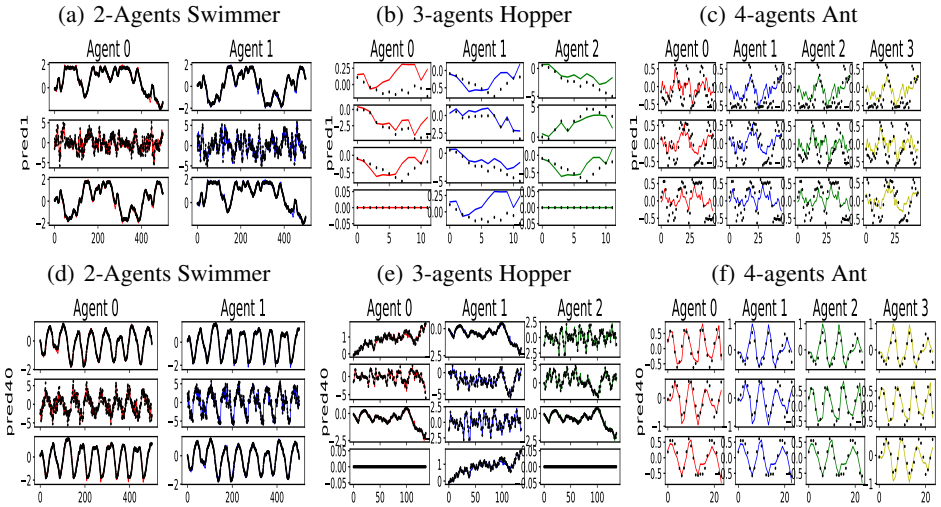

Figure 2: Predicted Trajectories (colored) vs Actual Observations (black). (a)-(c) First iteration; (d) - (f) Iteration 40. The X axis represents the part of steps encountered for one trajectory under that iteration, and the Y axis represents the numerical value of the observation, i.e. (a) has three rows to represent its three coordinates of its observation. We also provide iteration 10, 20 in Appendix D.1

get increasingly more accurate. Note that the actual trajectory is changing according to the current policy, and the current policy is optimized based on the further accurate learning of MAPSR.

To enrich our algorithm environment, we also test our algorithm (MAPSRL-2) in multi-agent particle environments, using the benchmark by Lowe et al. (2017), please check Appendix D.2 for experiment details. We test our algorithm for large n cases. In this environment, the agents can have cooperative goals such that all agents must maximize a shared return and conflicting competitive goals. We set up the environments where agents can only perform physical actions but not communication; however, to achieve the goals, agents need explicit communication about others' locations to achieve the best reward. These partially observable environments give us the motivation to test our method. We report the rewards in Figure 3. We see that MAPSRL outperforms IPSR and baseline, in terms of the convergence speed and final attained rewards, with a different number of agents. The predictive states convey the information that can help communication between agents in limited communication and observation environments.

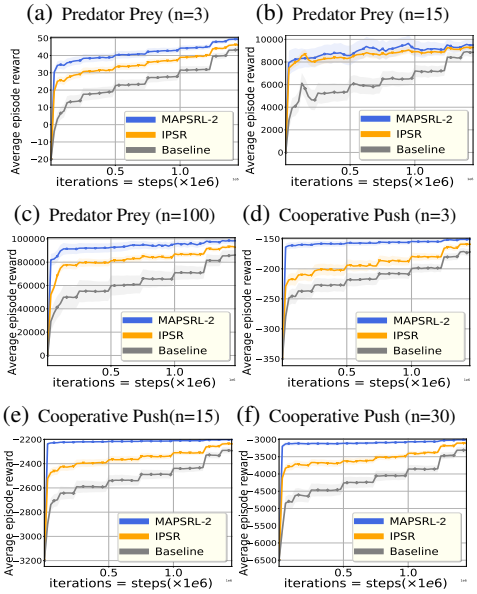

Figure 3: Performance of our method (MAPSRL-2) under multi-agent particle partially observable environments. We use a different number of agents in predator-prey and cooperative push environments. We run ten times, and the shaded area is the 95% confidence interval.

7 Conclusion

We propose a MAPSR model, extending ideas from single-agent predictive state representations to a multi-agent scenario, during the process, we introduce the dynamic interactive graph to model agents' interactions. Furthermore, we provide the theoretical guarantees of the MAPSR model. Finally, a learning algorithm that supports gradient-based deep MARL methods is developed. Our method provides a model-based MARL framework under a partially observable environment. The experiments proved that our model assumption is valid by observing the highest return while reducing the observations' prediction error over trajectories.

ETHICS STATEMENT

The authors of this paper claimed that we have read the Code of Ethics, adhered to it, and explicitly acknowledged this during the submission process. We claimed that we did not perform any of the following studies that involve human subjects, practices to data set releases, potentially harmful insights, methodologies and applications, potential conflicts of interest and sponsorship, discrimination/bias/fairness concerns, privacy and security issues, legal compliance, and research integrity issues.

REPRODUCIBILITY STATEMENT

We have an appendix section after the main paper to include the model supplements, algorithm details, method diagram, experiment settings, and proofs for our main theorems. We give a link directs to our codebase in the supplementary material.

REFERENCES

Douglas Aberdeen, Olivier Buffet, and Owen Thomas. Policy-Gradients for PSRs and POMDPs. pp. 8, 2007.

Byron Boots, Sajid M Siddiqi, and Geoffrey J Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954–966, June 2011. ISSN 0278-3649. doi: 10.1177/0278364911404092.

Byron Boots, Arthur Gretton, and Geoffrey J Gordon. Hilbert Space Embeddings of Predictive State Representations. pp. 10, 2013.

Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, April 2002. ISSN 0004-3702. doi: 10.1016/S0004-3702(02)00121-2.

D. Braha and Yaneer Bar-Yam. Time-Dependent Complex Networks: Dynamic Centrality, Dynamic Motifs, and Cycles of Social Interactions. In Thilo Gross and Hiroki Sayama (eds.), *Adaptive Networks: Theory, Models and Applications*, Understanding Complex Systems, pp. 39–50. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-642-01284-6. doi: 10.1007/978-3-642-01284-6_3.

Bilian Chen, Biyang Ma, Yifeng Zeng, Langcai Cao, and Jing Tang. Tensor Decomposition for Multi-agent Predictive State Representation. *arXiv:2005.13706 [cs]*, May 2020.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual Multi-Agent Policy Gradients. *arXiv:1705.08926 [cs]*, December 2017.

Roger Frigola, Fredrik Lindsten, Thomas B. Schön, and Carl E. Rasmussen. Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC. *arXiv:1306.2861 [cs, stat]*, December 2013.

Peter J. Grabner and Helmut Prodinger. Maximum Statistics of N Random Variables Distributed by the Negative Binomial Distribution. *Combinatorics, Probability and Computing*, 6(2):179–183, June 1997. ISSN 1469-2163, 0963-5483. doi: 10.1017/S0963548397002939.

William Hamilton, Mahdi Milani Fard, and Joelle Pineau. Efficient Learning and Planning with Compressed Predictive States. pp. 45.

Ahmed Hefny, Carlton Downey, and Geoffrey Gordon. Supervised Learning for Dynamical System Learning. *arXiv:1505.05310 [cs, stat]*, November 2015.

Ahmed Hefny, Carlton Downey, and Geoffrey J. Gordon. An Efficient, Expressive and Local Minima-free Method for Learning Controlled Dynamical Systems. *arXiv:1702.03537 [stat]*, February 2018a.

Ahmed Hefny, Zita Marinho, Wen Sun, Siddhartha Srinivasa, and Geoffrey Gordon. Recurrent Predictive State Policy Networks. *arXiv:1803.01489 [cs, stat]*, March 2018b.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. A Spectral Algorithm for Learning Hidden Markov Models. *arXiv:0811.4413 [cs]*, July 2012.

Michael R. James and Satinder Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, pp. 53, New York, NY, USA, July 2004. Association for Computing Machinery. ISBN 978-1-58113-838-2. doi: 10.1145/1015330.1015359.

Nan Jiang, Alex Kulesza, and Satinder Singh. Completing State Representations using Spectral Learning. pp. 10, 2018.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, May 1998. ISSN 00043702. doi: 10.1016/S0004-3702(98)00023-X.

Michael L Littman, Richard S Sutton, and Satinder Singh. Predictive Representations of State. pp. 14, 2001.

Iou-Jen Liu, Raymond A. Yeh, and Alexander G. Schwing. PIC: Permutation Invariant Critic for Multi-Agent Deep Reinforcement Learning. *arXiv:1911.00025 [cs, stat]*, October 2019a.

Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-Agent Game Abstraction via Graph Attention Neural Network. *arXiv:1911.10715 [cs]*, November 2019b.

Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-Agent Game Abstraction via Graph Attention Neural Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7211–7218, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i05.6211.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *arXiv:1706.02275 [cs]*, June 2017.

Xueguang Lyu and Christopher Amato. Likelihood Quantile Networks for Coordinating Multi-Agent Reinforcement Learning. *arXiv:1812.06319 [cs, stat]*, June 2020.

Shuai Ma, Renjun Hu, Luoshu Wang, Xuelian Lin, and Jinpeng Huai. Fast Computation of Dense Temporal Subgraphs. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 361–372, April 2017. doi: 10.1109/ICDE.2017.95.

Yaru Niu, Rohan Paleja, and Matthew Gombolay. Multi-Agent Graph-Attention Communication and Teaming. pp. 10, 2021.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer International Publishing, Cham, 2016. ISBN 978-3-319-28927-4 978-3-319-28929-8. doi: 10.1007/978-3-319-28929-8.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv:1803.11485 [cs, stat]*, June 2018.

Heechang Ryu, Hayong Shin, and Jinkyoo Park. Multi-Agent Actor-Critic with Hierarchical Graph Attention Network. *arXiv:1909.12557 [cs, stat]*, November 2019.

Heechang Ryu, Hayong Shin, and Jinkyoo Park. Multi-Agent Actor-Critic with Hierarchical Graph Attention Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7236–7243, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i05.6214.

Sajid Siddiqi, Byron Boots, and Geoffrey Gordon. Reduced-Rank Hidden Markov Models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 741–748. JMLR Workshop and Conference Proceedings, March 2010.

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. *arXiv:1905.05408 [cs, stat]*, May 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *arXiv:1706.05296 [cs]*, June 2017.

Ming Tan. MInudletip-AengdeennttRvesi.nCfoorocpemereantitvLe eAargneinntgs:. pp. 8, 1993.

Joel A. Tropp. An Introduction to Matrix Concentration Inequalities. *arXiv:1501.01571 [cs, math, stat]*, January 2015.

Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Deep Multi-Agent Reinforcement Learning for Decentralized Continuous Cooperative Control. *arXiv:2003.06709 [cs, stat]*, December 2020.

Zhi Zhang, Jiachen Yang, and Hongyuan Zha. Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. *arXiv:1909.10651 [cs, stat]*, September 2019.

Qiankun Zhao, Yuan Tian, Qi He, Nuria Oliver, Ruoming Jin, and Wang-Chien Lee. Communication motifs: A tool to characterize social communications. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pp. 1645–1648, New York, NY, USA, October 2010. Association for Computing Machinery. ISBN 978-1-4503-0099-5. doi: 10.1145/1871437.1871694.

Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning Implicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning. *arXiv:2007.02529 [cs, stat]*, October 2020.

APPENDICES TO REINFORCEMENT LEARNING UNDER A MULTI-AGENT PREDICTIVE STATE REPRESENTATION MODEL: METHOD AND THEORY

A METHOD DETAILS

A.1 TENSOR DECOMPOSITION OF MAPSR

Existed Formulation. Chen et al. (2020) use a $n+1$ multi-dimensional tensor called system dynamics tensor $\mathcal{D} \in \mathbb{R}^{|\mathcal{T}_1| \times \cdots \times |\mathcal{T}_n| \times |\mathcal{H}|}$ to represent the system dynamics of the MAPSR, with $|\mathcal{T}_1|$ representing the cardinality of the tests for agent 1, n denoting the number of agents, and the $|\mathcal{H}|$ being the cardinality of the joint history. Each element of the tensor is a probability of a joint test given joint histories. Given the system dynamics tensor, finding the latent predictive state can be transferred into finding a minimal linearly independent set from the system dynamics tensor, and it can be solved by spectral method such as tensor decomposition.

As mentioned earlier, this formulation can not satisfy our needs. In **Proposition 1** of Appendix A.2, we also give the sample complexity analysis such that the sample size needed to formulate $\mathcal{D}$ scales exponentially with the length of tests and number of agents.

Here we give a summary of tensor decomposition of their method. Given a system dynamic tensor $\mathcal{D}$ such that:

$$\mathcal{D} \approx [\lambda; D^1, \cdots, D^n, F] = \sum_{r=1}^{R} \lambda_r D_r^1 \circ \cdots \circ D_r^n F_r \tag{9}$$

Here $\circ$ is the outer product. $D^1, \cdots, D^n, F$ are matrices. The factor matrices D^1, D^n, F consist of the vectors, i.e., $D^1 = [D_{:1}^1 D_{:2}^1 \cdots D_{:R}^1] \in \mathbb{R}^{|\mathcal{T}_1| \times R}$. A colon is used to indicate all elements of a mode, thus, the Rth column of D^1 is denoted by $D_{:R}^1$. For any $i_1 \in \{1, \ldots, |\mathcal{T}_1|\}$, $i_n \in \{1, \ldots, |\mathcal{T}_n|\}$, and $k \in \{1, \ldots, |\mathcal{H}|\}$, after the decomposition, we get

$$\mathcal{D}_{i_1, \cdots, i_n, k} = \sum_{r=1}^{R} \lambda_r D_{i_1 r}^1 \cdots D_{i_n r}^n F_{kr}, \tag{10}$$

where $\{i_1, \cdots, i_n, k\}$ are index corresponding to the specific dimension of the $\mathcal{D}$. The last dimension of the tensor $\mathcal{D}$ is compressed in a matrix F, and its row vector $x_k = [x_k(1) \ldots x_k(R)] \in \mathcal{R}^{1 \times R}, k \in \{1, \ldots, |\mathcal{H}|\}$ is a summary of joint history and can be considered as a compressed version of the system predictive state vector $p(\mathbf{Q}|\mathbf{h}_k)$, the joint history $\mathbf{h}_k \in \mathcal{H}(k \in [1, |\mathcal{H}|])$ at time step $s = |\mathbf{h}_k|$, where $\mathbf{Q}$ is the core joint test set. The whole fibers listed in the set $\mathbf{Q}$ form a basis of the space spanned by the mode-(n+1) fibers of tensor $\mathcal{D}$. Thus, by constructing the vector $m = (\lambda * D_{i_1:}^1 * \cdots D_{i_n:}^n)^T$, where $*$ is Hadamard product. $D_{i_1:}^1$ denotes the i_1-th row vector of D^1. Then we could rewrite the previous equation as

$$\mathcal{D}_{i_1, \cdots, i_n, k} = \sum_{r=1}^{R} m(r) x_k(r) = x_k (\lambda * D_{i_1:}^1 * \cdots D_{i_n:}^n)^T = x_k m \tag{11}$$

$m(r)$ is a scalar that $m(r) = \lambda_r D_{i_1 r}^1 \cdots D_{i_n r}^n$. And x_k is the system state vector and m is the prediction parameter, both of them are obtained by the tensor decomposition.

A.2 ANALYSIS OF SAMPLE COMPLEXITY FOR FORMULATING THE SYSTEM DYNAMIC TENSOR

The paper Chen et al. (2020) does not analyze the sample complexity to construct $\mathcal{D}$, which is a $n+1$ multi-dimensional tensor $\mathcal{D} \in \mathbb{R}^{|\mathcal{T}_1| \times \cdots \times |\mathcal{T}_n| \times |\mathcal{H}|}$. We give this analysis. The $\mathcal{D}_{i_1, \cdots, i_n, k} := \mathbf{Pr}(t_{1 i_1}, \cdots, t_{n i_n} | \mathbf{h}_k)$ is an element of that tensor $\mathcal{D}$ such that $t_{1 i_1}$ is the i_1-th test of agent 1, similarly, the $t_{n i_n}$ is the i_n-th test of agent n, and $\mathbf{h}_k$ is the joint history.

Proposition 1. *In a n-agents system, assume every agent has the same observation and action space $|\mathcal{O}|$, $|\mathcal{A}|$, for a length-k test, to formulate a complete system dynamics tensor $\mathcal{D}$ defined in equation 10. Assume each entry of the tensor needs S samples to give a sufficient estimation using Monte-Carlo roll-out method, then the total sample size is at least $(|\mathcal{O}||\mathcal{A}|)^{kn} S$, if the agents are homogeneous, in other words, they are permutation invariant such that identity does not matter, then the total sample size is $\binom{|\mathcal{O}||\mathcal{A}|^k + n - 1}{n} S$.*

Proof. At one time step, for any agent, it has $|\mathcal{O}||\mathcal{A}|$ different combinations for the joint test, then for a length of k tests, it follows that $(|\mathcal{O}||\mathcal{A}|)^k$ number of different choices. Then the tensor $\mathcal{D}$ would need $(|\mathcal{O}||\mathcal{A}|)^{kn}$ elements to cover all the possible length k tests. So the total sample size is $(|\mathcal{O}||\mathcal{A}|)^{kn}S$. If the agents are homogeneous, then the ordering does not matter, for each agent, we have $(|\mathcal{O}||\mathcal{A}|)^k$ number of different choices for length k test, so the total choices for n agents are $\binom{|\mathcal{O}||\mathcal{A}|^k+n-1}{n}$, then we need $\binom{|\mathcal{O}||\mathcal{A}|^k+n-1}{n}S$ samples. □

The sample complexity is exponentially scaled with the number of agents and length of tests.

A.3 Learning for MAPSR in Details

Here we introduce the details about learning MAPSR, which supplements the section 3.3 and 4.1.1.

A.3.1 Learning For $\widehat{Q}_{t,i,j}$ and $\widehat{P}_{t,i,j}$

To calculate $\widehat{Q}_{t,i,j}$, we introduce two sets of linear operators $T_{i,j}$ and $U_{i,j}$, such that $T_{i,j}(\psi^h_{t,i,j}) \approx C_{\psi^o_{t,i}\psi^a_{t,j}|\psi^h_{t,i,j}}$ and $U_{i,j}(\psi^h_{t,ij}) \approx C_{\psi^a_{t,j}\psi^a_{t,j}|\psi^h_{t,i,j}}$. We estimate them by using two ridge regressions:

$$\arg\min_{T_{i,j}} \sum_{t=1}^{T} \mathcal{L}(T_{i,j}(\psi^h_{t,i,j})), \psi^o_{t,i} \otimes \psi^a_{t,j}) + R(T_{i,j}) \tag{12}$$

$$\arg\min_{U_{i,j}} \sum_{t=1}^{T} \mathcal{L}(U_{i,j}(\psi^h_{t,i,j}), \psi^a_{t,j} \otimes \psi^a_{t,j}) + R(U_{i,j}) \tag{13}$$

$\mathcal{L}$ represents the ridge regression loss, and R represents the regularizer. After learning $T_{i,j}$ and $U_{i,j}$, we can get the estimate of $C_{\psi^o_{t,i}\psi^a_{t,j}|\psi^h_{t,i,j}}$ and $C_{\psi^a_{t,j}\psi^a_{t,j}|\psi^h_{t,i,j}}$. Then we get the $\widehat{Q}_{t,i,j}$ by equation 2. Similarly, we use extended features to obtain $\widehat{P}_{t,i,j}$.

A.3.2 Extension, Filtering and Prediction Functions

Filtering. To obtain $\widehat{Q}_{t+1,i}$ from $\widehat{Q}_{t,i}$, $\widehat{P}_{t,i}$, we use filtering. We denote F_i as the filtering function. We describe the filtering process as below. From $\{o_i, a_i\}_{i=1}^n$, we obtain the embedding $\{\phi^o_{t,i}, \phi^a_{t,i}\}_{i=1}^n$. We then compute the observation covariance

$$C_{o_{t,i},o_{t,i}|h_{t,i},a_{t,i}} = M_{\phi^o_{t,i}\otimes\phi^o_{t,i}|\phi^a_{t,i};\psi^h_{t,i}}\phi^a_{t,i}. \tag{14}$$

We then multiply the extended state by inverse observation covariance to change predicting $\phi^o_{t,i}$ into conditioning on $\phi^o_{t,i}$.

$$M_{\psi^o_{t+1,i}|\psi^a_{t+1,i},\phi^o_{t,i},\phi^a_{t,i};\psi^h_{t,i}} = M_{\psi^o_{t+1,i}\otimes\phi^o_{t,i}|\psi^a_{t+1,i},\phi^a_{t,i};\psi^h_{t,i}} \times_{\phi^o_{t,i}} (C_{o_{t,i},o_{t,i}|h_{t,i},a_{t,i}} + \lambda I)^{-1}. \tag{15}$$

$\times$ here is to denote $n-mode$ (matrix or vector) product, $\times_{\phi^o_{t,i}}$ means multiplying the tensor by a matrix (or vector) in mode $\phi^o_{t,i}$.

We condition on $\phi^o_{t,i}$ and $\phi^a_{t,i}$ to obtain shifted state.

$$Q_{t+1,i} := M_{\psi^o_{t+1,i}|\psi^a_{t+1,i};\phi^o_{t,i},\phi^a_{t,i},\psi^h_{t,i}} = M_{\psi^o_{t+1,i}|\psi^a_{t+1,i},\phi^o_{t,i},\phi^a_{t,i};\psi^h_{t,i}} \times_{\phi^o_{t,i}} \phi^o_{t,i} \times_{\phi^a_{t,i}} \phi^a_{t,i}. \tag{16}$$

Based on the updating rule, $Q_{t+1,i} = F_i(P_{t,i}, o_{t,i}, a_{t,i})$, and $P_{t,i} = W_i(Q_{t,i})$, we write the filtering equation.

$$\begin{aligned} Q_{t+1,i} &= F_i(P_{t,i}, o_{t,i}, a_{t,i}) \\ &:= M_{\psi^o_{t+1,i}\otimes\phi^o_{t,i}|\psi^a_{t+1,i},\phi^a_{t,i};\psi^h_{t,i}} \times_{\phi^o_{t,i}} (M_{\phi^o_{t,i}\otimes\phi^o_{t,i}|\phi^a_{t,i};\psi^h_{t,i}}\phi^a_{t,i} + \lambda I)^{-1} \times_{\phi^o_{t,i}} \phi^o_{t,i} \times_{\phi^a_{t,i}} \phi^a_{t,i}, \end{aligned}$$

where $P_{t,i} := M_{\psi^o_{t+1,i}\otimes\phi^o_{t,i}|\psi^a_{t+1,i},\phi^a_{t,i};\psi^h_{t,i}}$. F_i usually is known because it is obtained through the above calculation using known quantities, however, W_i, Z_i must have to be learned by using regressions.

Extension. The W_i can be learned by kernel regression if we know the $\widehat{Q}_{t,i}$ and $\widehat{P}_{t,i}$. Previous work Hefny et al. (2018a) demonstrated the kernel regression model for learning single-agent PSR, here we extend to the MAPSR. We set the model parameter W_i.

We optimized a ridge regression problem for W_i.

$$\arg\min_{W_i} \sum_{t=1}^{T} \mathcal{L}(W_i(\widehat{Q}_{t,i}), \widehat{P}_{t,i}) + R(W_i)$$

Prediction. We can also get the prediction about the next one time observation o_t by the regression function such as:

$$\widehat{o}_{t,i} := \mathbb{E}(o_{t,i}|Q_{t,i}, a_{t,i}) = Z_i(Q_{t,i} \otimes \psi^a_{t,i})$$

We solve the prediction regression function Z_i by another ridge regression:

$$\arg\min_{Z_i} \sum_{t=1}^{T} \mathcal{L}(Z_i(\widehat{Q}_{t,i} \otimes \psi^a_{t,i}), \psi^o_{t,i}) + R(Z_i)$$

B Algorithm and Integrating with MARL method

B.1 Algorithm: MAPSRL

We give the details of MAPSRL in Algorithm 1.

B.2 MAPSRL in Diagram

We also use a diagram (Fig 4) to depict the algorithm. The algorithm runs k iterations; each iteration first uses the policy to roll out data and uses the regressions to obtain the PSR parameters. Then it executes the policy phase by using the PSR as the input of policy to generate action and using the current PSR parameters to update the predictive state. At the end of the policy phase, it updates both the PSR and policy parameters. The policy parameters and the linear relationship captured by W_i and Z_i are parameterized by the neural network, section C.3 has the network architectures. The loss is a composite loss that includes loss from actor-critic and the loss from the predictive state representation. The next iteration will re-learn the PSR parameters using the newly generated data based on the current policy obtained from the previous iteration; then, it does a soft update to update the PSR parameters with ones obtained at the previous iteration.

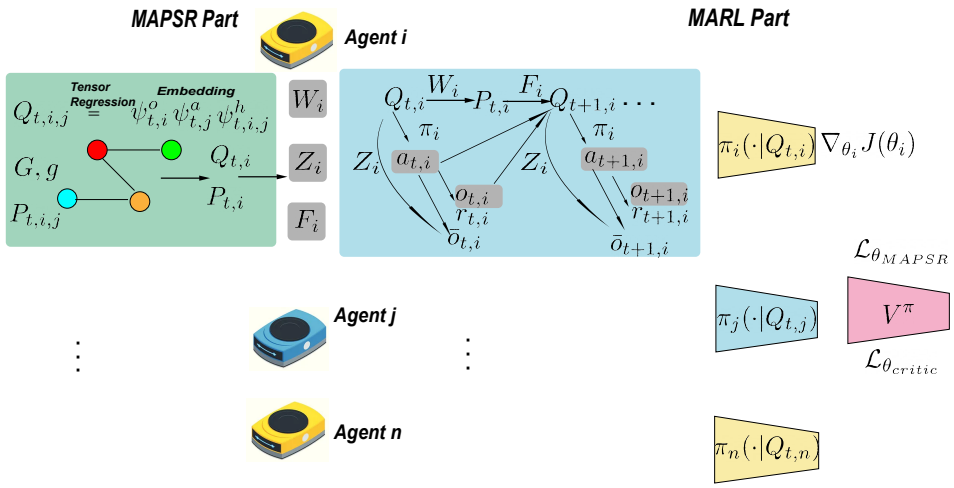

Figure 4: MAPSR architecture combining MAPSR and MARL. The left part is the MAPSR model, which corresponds to the state tracking phase in Algorithm 1. The right part is the MARL, which corresponds to the policy phase. We use the actor critic framework, which contains a centralized critic and many decentralized actors. Both actor and critic are parameterized by neural networks. Section B.3 and section C.3 give a detailed verbal description about this architecture.

r MAPSR based MARL algorit

Algorithm 1 MAPSRL

Input: Learning rate η, a graph G, a static complete graph G_c or static non-compete graph G_s or dynamic graph G_t
Initialize MARL Policy Θ_{MARL} randomly
for $k = 1, 2, 3, \cdots$ iterations **do**
  State Tracking Phase
  Sample $b = 1, 2, 3, \cdots M$ batch of initial trajectories: $\{(o_t^b, a_t^b)\}_{b=1}^M$ from existed policy obtained from previous iteration $k-1$: $\{\pi_i^{k-1}\}_{i=1}^n$
  Let $\pi_{\theta_{MARL}} = \{\pi_i^{k-1}\}_{i=1}^n$ if available or the initial policy;
  Given G, calculate $\widehat{Q}_i$ and $\widehat{P}_i$, and obtain the initial W_i, Z_i, F_i:
    (1).Regression $Q_{t,i,j} = T_{i,j} \circ U_{i,j}(h_{t,i,j})$ to get the $\widehat{Q}_{t,i,j}$ and $\widehat{P}_{t,i,j}$
    (2).Given the graph G, using equation 1, 4, or 5 to obtain $\widehat{Q}_{t,i}$, $\widehat{P}_{t,i}$
    (3).Given the $\widehat{Q}_{t,i}$, $\widehat{P}_{t,i}$, to obtain W_i, Z_i, F_i

  Policy Phase
  Initialize MAPSR parameters $\Theta_{MAPSR} = \{Q_{0,i}, W_i, Z_i\}_{i=1}^n$ from state tracking phase and previous iteration by a soft-update:
  $W_i = \beta W_i + (1-\beta) W_i^{k-1}$, $Z_i = \beta Z_i + (1-\beta) Z_i^{k-1}$, $F_i = F_i$
  for $b = 1, 2, 3, \cdots M$ batch of trajectories from $\{\pi_i^{k-1}\}_{i=1}^n$ **do**
    Reset episode: $a_{0,i}^b, o_{0,i}^b$
    for $t = 0, 1, 2, \cdots T$ roll-in in each trajectory **do**
      for Each agent i **do**
        Get observation $o_{t,i}^b$ and reward $r_{t,i}^b$
        Extension $P_{t,i}^b = W_i(Q_{t,i}^b)$
        Filtering $Q_{t+1,i}^b = F_i(Q_{t,i}^b, a_{t,i}^b, o_{t,i}^b, W_i)$
        Execute $a_{t+1,i}^b \sim \pi_i^{k-1}(Q_{t+1,i}^b)$
        Predict $\widehat{o}_{t,i}^b = Z_i(O_{t,i}^b, a_{t,i}^b)$
        Collect $o_{t,i}^b$, $\widehat{o}_{t,i}^b$, $a_{t,i}^b$, $r_{t,i}^b$, $Q_{t,i}^b$, $Q_{t+1,i}^b$
      end for
    end for
  end for
  Update Θ using $D = \{\{\{o_{t,i}^b, \widehat{o}_{t,i}^b, a_{t,i}^b, r_{t,i}^b, Q_{t,i}^b\}_{i=1}^n\}_{t=1}^T\}_{b=1}^M$:
  $\Theta \leftarrow Update(\Theta^{k-1}, D, \eta)$ as in Equation (8), get W_i^k,Z_i^k, and π_i^k, the linear operator W_i, Z_i, and nonlinear operator π_i are all parameterized by neural network.
end for
Output: Return $\Theta = (\Theta_{MAPSR}, \Theta_{MARL})$

B.3 Integrating Two Common MARL Algorithms Into MAPSR Model

Here we provide a brief intuitive introduction about how we connect existed MARL algorithms with MAPSR. Please look at Fig 4 for demonstration. $\psi_{t,i}^o$, $\psi_{t,j}^a$, and $\psi_{t,i,j}^h$ are embedding of $(o_{t:t+k-1,i}, a_{t:t+k-1,j})$, and $(o_{1:t-1,i}, a_{1:t-1,j})$. We also have embedding for extended part, labeled as ξ. $Q_{t,i,j}$ and $P_{t,i,j}$ are estimated by tensor regression using the embedding vectors. G represents the given graph, the estimation of $Q_{t,i}$ and $P_{t,i}$ are based on graph G and equation 1, equation 4 and equation 5 for reference. Please also go to subsection 3.2 and section 4.1.1 for a verbal description. The policy network uses the predictive state as input to return the action. The agent takes the action to get the observation. The filter F_i takes predictive state, action, and observation as inputs to get the next predictive state. We shows a centralized critic and gives the description in paragraph B.3.2. The loss is composed into two parts, $\mathcal{L}_{\theta_{MAPSR}} = \|Z_i\{F_i\left[W_i(Q_{t-1,i}), a_{t-1,i}, o_{t-1,i}\right], a_{t,i}\} - o_{t,i}\|^2$, and $\mathcal{L}_{\theta_{critic}}$, which we gives detailed explanation in equation 20. Please also go to section 5 and Algorithm B.1 for more details about the framework.

B.3.1 MAPSRL-1

It is based on the IAC Foerster et al. (2017) which directly applies the single-agent policy gradient to have each agent learn independently, with the idea behind independent Q-learning Tan (1993), with actor-critic in place of Q-learning.

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|Q_i)(r_i + \gamma V^{\pi_i}(Q_{t+1,i}) - V^{\pi_i}(Q_i))] \tag{17}$$

to train an actor-critic pair for each agent, resulting in actors $\pi_{\theta_i}(a_i|Q_i)$ and critics $V^{\pi_i}(Q_i)$. Here we use the Q_i to replace the o_i in the algorithm. If the agents are homogeneous, we can share the critic and actor network. We have neural network loss as

$$\mathcal{L} = \mathbb{E}[(V^{\pi_i}(Q_i) - y_i)^2] \qquad y_i = r_i + \gamma \widehat{V}^{\pi_i}(Q_{t+1,i}) \tag{18}$$

Here $\widehat{V}^{\pi_i}$ is the target value function.

B.3.2 MAPSRL-2

The algorithm borrows the idea from MADDPG Lowe et al. (2017) and LICAZhou et al. (2020) using the gradient of the action value function with respect to policy to direct the policy gradient update.

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|Q_i) \nabla_{\theta_i} V^{\pi}(Q_1^{\pi_{\theta_1}}, \cdots, Q_n^{\pi_{\theta_n}})] \tag{19}$$

Unlike MADDPG using the deterministic policy, the gradient is taken on the policy parameter space, which is reflected in LICA Zhou et al. (2020). Our method also uses the centralized critic to save space. Also, in order to solve the non-stationary environments when each agent is learning, it uses the joint predictive states as input without considering other agents' actions, which saves the input space since the predictive state is a conditional operator that considers other agents' actions.

The critic loss is defined as below:

$$\begin{aligned} \mathcal{L}_{\theta_{critic}} = & \quad \mathbb{E}[(V^{\pi}(Q_1, \cdots, Q_n) - y)^2] \\ y = & \quad \textstyle\sum_i r_i + \gamma \widehat{V}^{\pi}(Q_{t+1,1}^{\pi_{\theta_1}}, \cdots, Q_{t+1,n}^{\pi_{\theta_n}}) \end{aligned} \tag{20}$$

Even though the original MADDPG has the current actions as input to the value function so that the environment is stationary, we take out this operation because the predictive state already considers the effect actions can bring.

C Environment and Experiment

C.1 MAMujoCo Environment Setup

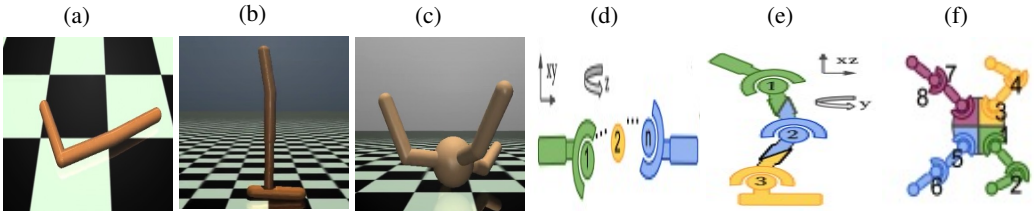

Figure 5: The illustration of three environments swimmer, hopper, ant, and their corresponding MAMuJoCo version. (a) Single swimmer; (b) Single hopper; (c) Single ant; (d) n-agents swimmer; (e) 3-agents hopper; (f) 4-agents ant.

Here, we give the details of setting up our multi-agent environment - MAMuJoCo. As introduced in section 6, the many agents are constructed by separating a existed single agent into parts, and each agent will only control a part of the whole agent (Figure 5).

Partially Observable space: MAMuJoCo is a simulated robotic environment, the partially observable property is achieved by only allowing partial information to the agents. For all environments, only the angles of the agent's joints are visible to the network; the velocities are hidden.

Action: Each agent's action space in MAMuJoCo is given by the joint action space overall motors controllable by that agent.

Observation: For each agent i, observations are constructed by inferring which body segments and joints are observable by an agent i. Each agent can always observe all joints within its sub-graph. A

Table 1: Configurations for MAMuJoCo environment

Environments	
Swimmer	
k	0
R	$\sum_i(\frac{\Delta d_i}{\Delta t}) + 0.0001r$
r	$r = -\left\|\mathbf{a}\right\|_2^2$ is a regularizer for joint action $\mathbf{a}$
Hopper	
k	2
R	$\sum_i(\frac{\Delta d_i}{\Delta t}) + 0.001r + 1.0$
r	$r = -\left\|\mathbf{a}\right\|_2^2$ is a regularizer for joint action $\mathbf{a}$
Ant	
k	0
R	$\sum_i(\frac{\Delta d_i}{\Delta t}) + 5 \cdot 1e(-3)\left\|\text{external contact forces}\right\|_2^2 + 0.0001r$
r	$r = -\left\|\mathbf{a}\right\|_2^2$ is a regularizer for joint action $\mathbf{a}$

configurable parameter $k \geq 0$ determines the maximum graph distance to the agent's subgraph at which joints are observable. For example, $k = 0$ means agents can only observe their own joints and body parts, while $k = 1$ means it can observe its adjacent joints, which has 1 graph distance to the agent. The agent observation is then given by a fixed order concatenation of each observable graph element's representation vector. Depending on the environment and configuration, representation vectors may include attributes such as position, velocity, and external body forces. In addition to joint and body segment-specific observation categories, agents can also be configured to observe the robot's central torso's position and velocity attributes.

C.2 Experiment setup

We select three experiments from MAMuJoCo and give a detailed description of the experiments' setup. In all environments, the agent has the goal to maximize the velocity of the first coordinate for the team. We use k to denote the maximum observation distances to the subgraph. We use Δd to denote the first coordinate position difference between a time difference Δt. Finally, we use R to denote the reward function. Table 1 has the configuration details for these parameters.

C.3 Neural Network Architecture

We implement all algorithms using deep neural networks as function approximators. We ensure that all policy, value, and action-value functions have the same neural network architecture among all algorithms to the extent each algorithm allows for a fair comparison.

For our experiments with continuous action spaces, a Gaussian distribution with a diagonal covariance matrix is used. The policy network maps from the input feature to a Gaussian distribution vector μ. Moreover, $\mu = [\text{mean}, \text{std}]$, where mean is a vector specifies the action means, and std vector specifies the standard deviation.

In the implementation of all actor-critic method, all the actor-network is parameterized by a multi-layer perceptron (MLP) with two hidden layers of size 400 and 300 respectively and ReLU activation, which takes in the individual agent's predictive state and outputs the mean and covariance of a Gaussian policy. The critic network is also an MLP with two hidden layers with 400 and 300 units, respectively. For MAPSRL-1, the critic network is used to approximate per-agent utilities, which receives each agent's predictive state as input. For IAC, same as MAPSRL-1, it receives agent local observation and individual action as input.

In MAPSRL-2, there is a shared critic network that approximates all agents utilities, which receives all agents' predictive states as input. In MADDPG, the critic receives the global state and the joint action of all agents as input. The global state consists of the complete state information from the original OpenAI Gym environment. Each decentralized actor (i.e., policy) network takes in each agent's observation and outputs the agent's action vector.

Table 2: Model parameters for MAMuJoCo environment

	Environments		
	Swimmer	Hopper	Ant
n	2	3	4
μ	0	0	0
σ	0.1	0.1	0.1
γ	0.99	0.99	0.99
Soft target network	0.001	0.001	0.001
α_1	0.7	0.65	0.7
α_2	0.3	0.35	0.3
β	0.6	0.45	0.5
η learning rate of Adam	0.001	0.001	0.001
λ ridge regression regularization	0.01	0.01	0.01
Total iterations	50	50	50
Number of trajectories	100	100	100
Maximum number of steps per trajectory	1000	1000	1000
Length of test window	8	12	10
Length of history window	8	12	10

C.4 Model Parameters

The hyper parameters for the the MAMuJoCo environments are in Table 2.

D Supplement Experiments

D.1 MAMujoCo Supplement Experimental Results

We plotted the predictive observations compared to actual observations in Figure 2 at the beginning of the learning process (iteration 1) and end of the learning (iteration 40). We also show the results of iteration ten and iteration 20 in Figure 6. By comparing with iterations 1 and iterations 40 in Figure 2, we see that the iteration 40 has the smallest difference between predictive observation and true observation, and the difference gets increased as the iteration goes to the earlier stage of the learning process. So the predictive accuracy is improved incrementally with the learning progresses.

D.2 Multi-Agent Particle Environment

We also test MAPSR into another environment, multi-agent particle environment Lowe et al. (2017). The agents are displaced into a 2-dimensional coordinate. This environment does not assume that all agents have identical action and observation spaces. We run experiments using a different number of agents on two environments, the predator-prey, and cooperative-push. We use the same configuration in Liu et al. (2019a). The network has the same design as MAMuJoCo in section C.3, except we use MLP with two hidden layers with the same 128 units respectively, so we do not repeat the description. In Table 3, we report the hyperparameters for multi-agent particle environments.

E Detailed Proofs

We give the proof of Theorem 1 in Appendix E.1, and we first introduce the following Lemmas to prepare the proof.

Definition 2. *Let $X_1 \dots X_k$ be independent random variables of dimensionality $d_{X_1} \dots d_{X_k}$ such that $\|X_k\| < t_{x_k}$. Let $\{(x_{1_j} \dots, x_{k_j})\}_{j=1}^N$ be the N i.i.d samples from distribution of $X_1, \dots, X_k$, the $C_{X_1} := \mathbb{E}[X_1 X_1^T]$ and $\widehat{C}_{X_1} = \frac{1}{N}\sum_{j=1}^N x_{1_j} x_{1_j}^T$, and $C_{X_1 X_2} := \mathbb{E}[X_1 X_2^T]$ and $\widehat{C}_{X_1,X_2} = \frac{1}{N}\sum_{j=1}^N x_{1_j} x_{2_j}^T$. Also, we use the $\cdot), v(\cdot)$ to denote the largest, smallest eigenvalue of a matrix.*

Lemma 3 (Tropp (2015))**.** *Let a_j be a finite sequence of independent random, Hermitian matrices with dimension d. Assume that $0 \le v(a_j)$ and $u(a_j) \le L$ for each j. Let $S = \sum_j a_j$, then for any*

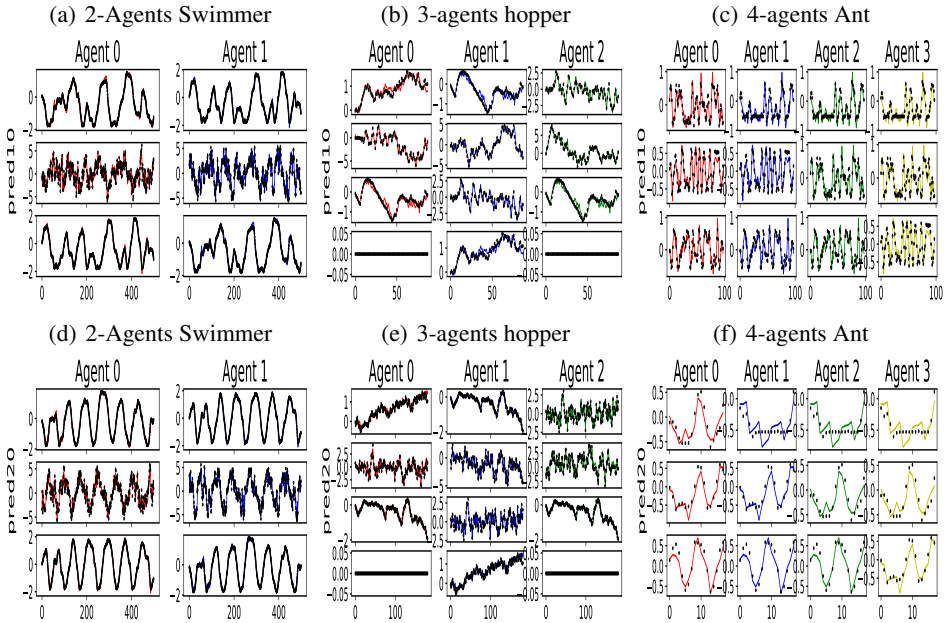

Figure 6: Additional Experiments Results, Predicted Trajectories vs Actual Observations for Multi-Agent Environments. (a) - (c) Iteration 10; (d) - (f) Iteration 20; This is the supplement for Figure 2

Table 3: Model parameters for multi-agent particle environment

	Environments					
	Predator-prey			Cooperative-push		
	n=3	n=15	n=100	n=3	n=15	n=30
γ	0.99	0.99	0.99	0.99	0.99	0.99
Soft target network	0.001	0.001	0.001	0.001	0.001	0.001
α_1	0.8	0.75	0.5	0.8	0.75	0.7
α_2	0.2	0.25	0.5	0.2	0.25	0.3
β	0.45	0.5	0.45	0.55	0.5	0.5
η learning rate of Adam	0.05	0.05	0.05	0.05	0.05	0.05
λ ridge regression regularization	0.01	0.01	0.01	0.01	0.01	0.01
Total iterations	30	30	30	30	30	30
Number of trajectories	100	100	100	100	100	100
Maximum number of steps per trajectory	500	500	500	500	500	500
Length of test window	5	5	5	5	5	5
Length of history window	5	5	5	5	5	5

$\eta \in [0, 1]$, *it follows that*

$$\mathbf{Pr}(v(S) \leq (1-\eta)v(\mathbb{E}[S])) \leq d\Big[\frac{e^{-\eta}}{(1-\eta)^{1-\eta}}\Big]^{v(\mathbb{E}[S])/L} \leq 2de^{-\eta v(\mathbb{E}[S])/L} \tag{21}$$

Corollary 1. *Let X be a random variable, for any $\epsilon \in (0, 1)$ such that $N > \frac{t_x^2 \log(2d_X/\epsilon)}{v(C_X)}$ the following holds with probability at least $1 - \epsilon$*

$$v(\widehat{C}_X) > \frac{t_x^2 \log(2d_X/\epsilon)}{v(C_X)N}$$

In other words, if N large enough, then $\widehat{C}_X$ and C_X will be close enough.

Proof. Define $S_j = 1/N x_j x_j^T$. Then it follows that $u(S_j) \leq L = t_x^2/N$ and define $\epsilon := 2d_X e^{-\sigma N v(C_X)/t_x^2}$, which implies that $\sigma = \frac{t_x^2 \log(2d_X/\epsilon)}{v(C_X)N}$. Then it follows from Matrix Chernoff Inequality in Lemma 3 that $P_r(v(\widehat{C}_X) \leq (1-\sigma)v(C_X)) \leq \epsilon$ □

Lemma 4 (Tropp (2015)). *A finite sequence $\{a_j\}$ of independent, random matrices with common dimensions $a \times b$, and assume that $\mathbb{E}[a_j] = 0$ and $\|a_j\| \leq L$ for each j, let $S = \sum_j a_j$ as a random matrix. Let $Var(S)$ be the variance statistics such that $Var(S) = \max\{\left\|\mathbb{E}[SS^T]\right\|, \left\|\mathbb{E}[S^T S]\right\|\}$, then*

$$\mathbf{Pr}(\|S\| > c) \leq (a+b)e^{\frac{-c^2/2}{Var(S)+Lc/3}} \tag{22}$$

Corollary 2. *With at least probability $1-\epsilon$ that*

$$\left\|\widehat{C}_{YX} - C_{YX}\right\| \leq \sqrt{\frac{2\log(d_Y+d_X)/\epsilon Var}{N}} + \frac{2\log((d_Y+d_X)/\epsilon)L}{3N}$$

where $L = t_y t_x + \|C_{YX}\| \leq 2t_y t_x$ and $Var = \max\{t_y^2 \|C_X\|, t_x^2 \|C_Y\|\} + \|C_{YX}\|^2 \leq 2t_y^2 t_x^2$

Proof. Let X, Y be two random variables, and let a finite sequence $\{a_j\}$ of independent random matrices to satisfy $a_j = y_j x_j^T - C_{YX}$. So the a_j will have dimensions $d_X \times d_Y$. Let random matrix $S = \sum_j a_j$. It follows that $\mathbb{E}[a_j] = 0$ and $\|a_j\| = \left\|y_j x_j^T - C_{YX}\right\| \leq \|y_k\| \|x_k\| + \|C_{YX}\| \leq t_y t_x + \|C_{YX}\|$

$$\begin{aligned}
\left\|\mathbb{E}[SS^T]\right\| &= \left\|\sum_{i,j}\left(\mathbb{E}[y_i x_i^T x_j y_j^T] - C_{YX}C_{XY}\right)\right\| \\
&= \left\|\sum_i\left(\mathbb{E}[\|x_i\|^2 y_i y_i^T] - C_{YX}C_{XY}\right) + \sum_{i\neq j}\left(\mathbb{E}[y_i x_i^T]\mathbb{E}[x_j y_j^T] - C_{YX}C_{XY}\right)\right\| \\
&= \left\|\sum_i(\mathbb{E}[\|x_i\|^2 y_i y_i^T] - C_{YX}C_{XY})\right\| \\
&\leq N(t_x^2 \|C_Y\| + \|C_{YX}\|^2)
\end{aligned}$$

Similarly, $\left\|\mathbb{E}[SS^T]\right\| \leq N(t_y^2 \|C_X\| + \|C_{YX}\|^2)$. By applying lemma 4, we have $\epsilon = \mathbf{Pr}(\|S\| \geq Nc) \leq (d_X+d_Y)e^{\left(\frac{-Nc^2/2}{Var+Lc/3}\right)}$ and therefore, it implies that

$$\begin{aligned}
c &\leq \frac{\log((d_X+d_Y)/\epsilon)L}{3N} + \sqrt{\frac{(\log(d_X+d_Y/\epsilon))^2 L^2}{9N^2} + \frac{2\log((d_X+d_Y)/\epsilon)Var}{N}} \\
&\leq \frac{2\log((d_X+d_Y)/\epsilon)L}{3N} + \sqrt{\frac{2\log((d_X+d_Y)/\epsilon)Var}{N}}
\end{aligned}$$

□

Corollary 3. *For random variable X with dimensionality d_X and $\|X\| \leq t_x$, with probability $1-\epsilon$, it follows that*

$$\left\|C_X^{-1/2}(\widehat{C}_X) - C_X\right\| \leq 2t_x\sqrt{\frac{2\log(2d_X/\epsilon)}{N} + \frac{2\log(2d_X/\epsilon)L}{3N}}$$

where $L = \frac{t_x^2}{\sqrt{v(C_X)}} + t_x$

Proof. The proof is similarly to the the proof of corollary 2, define $a_j = \sum_X^{-1/2} x_j x_j^T - C_X^{1/2}$, $S = \sum a_j$then it follows that $\mathbb{E}[a_j] = 0$ and $\|a_j\| \leq \frac{t_x^2}{\sqrt{v(C_X)}} + t_x$

$$\left\|\mathbb{E}[S^T S]\right\| = \left\|\mathbb{E}[SS^T]\right\| \leq N(t_x^2 + \|C_X\|^2) \leq 2Nt_x^2 \tag{23}$$

Applying lemma 4 to get

$$\epsilon = \mathbf{Pr}(\|S\| \geq Nc) \leq 2d_X e^{\frac{-Nc^2/2}{2t_x^2+Lc/3}} \tag{24}$$

it follows that

$$c \leq \frac{2\log(2d_X/\epsilon)L}{3N} + 2t_x\sqrt{\frac{\log(2d_X/\epsilon)}{N}} \tag{25}$$

□

Lemma 5. *For two random variables X, Y, let $\widehat{C}_{YX} = C_{YX} + \Delta_{YX}$, and $\widehat{C}_X = C_X + \Delta_X$ where $\mathbb{E}[\Delta_{YX}]$ and $\mathbb{E}[\Delta_X]$ are not necessarily zero and $\widehat{C}_X$ is symmetric positive semidefinite. Define $A = C_{YX}C_X^{-1}$ and $\widehat{A} = \widehat{C}_{YX}(\widehat{C}_X + \lambda)^{-1}$. Then it follows that:*

$$\left\|\widehat{A} - A\right\| \leq \sqrt{\frac{u\left(C_Y\right)}{v(C_X)}}\left(\frac{\sqrt{v(C_X)}\left\|C_X^{-1/2}\right\|\Delta_X + \lambda}{v(\widehat{C}_X) + \lambda}\right) + \frac{\|\Delta_{YX}\|}{v(\widehat{C}_X) + \lambda}$$

Proof.

$$\widehat{A} - A = C_{YX}((C_X + \Delta_X + \lambda I)^{-1} - C_X^{-1}) + \Delta_{YX}(C_X + \Delta_X + \lambda I)^{-1} = M_1 + M_2$$

It follows that

$$\|M_2\| \leq \frac{\Delta_{YX}}{v(\widehat{C}_X) + \lambda}$$

For M_1, by using facts $U^{-1} - V^{-1} = U^{-1}(V - U)V^{-1}$ and $C_{YX} = C_Y^{1/2}PC_X^{1/2}$, where P is a correlation matrix with $\|P\| \leq 1$,

$$\begin{aligned} M_1 &= -C_{YX}C_X^{-1}(\Delta_X + \lambda I)(C_X + \Delta_X + \lambda I)^{-1} \\ &= -C_Y^{1/2}PC_X^{-1/2}(\Delta_X + \lambda I)(C_X + \Delta_X + \lambda I)^{-1} \end{aligned}$$

$$\begin{aligned} \|M_1\| &\leq \sqrt{u\left(C_Y\right)}\frac{\left\|C_X^{-1/2}\Delta_X\right\| + \lambda\left\|C_X^{-1/2}\right\|}{v(\widehat{C}_X) + \lambda} \\ &= \sqrt{\frac{u(C_Y)}{v(C_X)}}\frac{\sqrt{v(C_X)}\left\|C_X^{-1/2}\Delta_X\right\| + \lambda}{v(\widehat{C}_X) + \lambda} \end{aligned}$$

□

Corollary 4. *Let $\{(x_k, y_k)\}_{k=1}^N$ be i.i.d samples from two random variables X, Y with dimensions d_X and d_Y and (uncentered) covariances C_X and C_Y. Assume $\|X\| \leq t_x$ and $\|Y\| \leq t_y$. Define $A = C_{YX}C_X^{-1}$ and $\widehat{A} = \widehat{C}_{YX}(\widehat{C}_X + \lambda)^{-1}$. For any $\epsilon \in (0, 1)$ such that $N > \frac{t_x^2 \log(2d_X/\epsilon)}{v(C_X)}$ the following holds with probability at least $1 - 3\epsilon$*

$$\left\|\widehat{A} - A\right\| \leq \sqrt{\frac{u(C_Y)}{v(C_X)}}\left(\frac{\sqrt{v(C_X)}\alpha + \lambda}{v(C_X)(1-\gamma) + \lambda}\right) + \frac{\beta}{v(C_X)(1-\gamma) + \lambda}$$

where

$$\alpha = 2t_x\sqrt{\frac{2\log(2d_X/\epsilon)}{N}} + \frac{2\log(2d_X/\epsilon)}{3N}\left(\frac{c_x^2}{\sqrt{v(C_X)}} + t_x\right)$$

$$\beta = 2t_y t_x\sqrt{\frac{\log(d_Y + d_X)/\epsilon}{N}} + \frac{3t_y t_x \log((d_Y + d_X)/\epsilon)}{3N}$$

$$\gamma = \frac{t_x^2 \log(2d_X/\epsilon)}{v(C_X)N}$$

Proof. It follows by applying Corollaries 1,2,3 to Lemma 5. By union bound, each condition has probability $1 - \epsilon$, so the total events are bounded

$$\mathbf{Pr}(\text{bounds satisfied}) := 1 - \mathbf{Pr}(\bigcup_{i=1}^{3} A_i) \geq 1 - \sum_{i=1}^{3}\mathbf{Pr}(A_i) = 1 - 3\epsilon$$

□

Lemma 6. *For two random variables X, Y, let $\widehat{C}_{YX} = C_{YX} + \Delta_{YX}$, and $\widehat{C}_X = C_X + \Delta_X$ where $\mathbb{E}[\Delta_{YX}]$ and $\mathbb{E}[\Delta_X]$ are not necessarily zero and $\widehat{C}_X$ is symmetric but not positive semidefinite.*

Define $A = C_{YX}C_X^{-1}$ *and* $\widehat{A} = \widehat{C}_{YX}\widehat{C}_X(\widehat{C}_X^2 + \lambda)^{-1}$. *Then it follows that:*

$$\left\|\widehat{A} - A\right\| \leq \sqrt{\frac{u(C_Y)}{v(C_X)^3}}\frac{\|\Delta_x\|^2 + 2u(C_X)\|\Delta_X\| + \lambda}{v(\widehat{C}_X) + \lambda} + \frac{\|C_{YX}\|\|\Delta_X\| + \|\Delta_{YX}\|\|C_X\| + \|\Delta_{YX}\|\|\Delta_X\|}{v(\widehat{C}_X)^2 + \lambda}$$

Proof.

$$\begin{aligned}\widehat{A} - A &= (C_{YX} + \Delta_{yx})(C_X + \Delta_X)((C_X + \Delta_X)^2 + \lambda I)^{-1} - C_{YX}C_XC_X^{-2} \\ &= C_{YX}C_X(((C_X + \Delta_X)^2 + \lambda I)^{-1} - C_X^{-2}) + (C_{YX}\Delta_X + \Delta_{YX}C_X + \Delta_{YX}\Delta_X)((C_X + \Delta_X)^2 + \lambda I) \\ &= M_1 + M_2\end{aligned}$$

For M_1, by using facts $U^{-1} - V^{-1} = U^{-1}(V - U)V^{-1}$ and $C_{YX} = C_Y^{1/2}PC_X^{1/2}$, where P is a correlation matrix with $\|P\| \leq 1$, it follows that

$$M_1 = -C_Y^{1/2}PC_X^{-3/2}(\Delta_X^2 + C_X\Delta_X + \Delta_XC_X + \lambda I)((C_X + \Delta_X)^2 + \lambda I)^{-1}$$

Therefore,

$$\|M_1\| \leq \sqrt{\frac{u(C_Y)}{v(C_X)^3}}\frac{\|\Delta_x\|^2 + 2u(C_X)\|\Delta_X\| + \lambda}{v(\widehat{C}_X) + \lambda}$$

$$\|M_2\| \leq \frac{\|C_{YX}\|\|\Delta_X\| + \|C_{YX}\|\|C_X\| + \|C_{YX}\|\|\Delta_X\|}{v(\widehat{C}_X)^2 + \lambda}$$

□

Corollary 5. *Let* $\{(x_k, y_k)\}_{k=1}^N$ *be i.i.d samples from two random variables* X, Y *with dimensions* d_X *and* d_Y *and (uncentered) covariances* C_X *and* C_Y. *The* $\mathbb{E}[\Delta_{YX}]$ *and* $\mathbb{E}[\Delta_X]$ *is not necessarily zero and* C_x *is symmetric but not necessarily positive semidefinite. Assume* $\|X\| \leq t_x$ *and* $\|Y\| \leq t_y$. *Define* $A = C_{YX}C_X^{-1}$ *and* $\widehat{A} = \widehat{C}_{YX}\widehat{C}_X(\widehat{C}_X^2 + \lambda)^{-1}$. *For any* $\epsilon \in (0, 1)$ *such that* $N > \frac{t_x^2\log(2d_X/\epsilon)}{v(C_X)}$ *the following holds with probability at least* $1 - 3\epsilon$

$$\left\|\widehat{A} - A\right\| \leq \sqrt{\frac{u(C_Y)}{v(C_X)^3}}\frac{\|\Delta_x\|^2 + 2u(C_X)\|\Delta_X\| + \lambda}{v(C_X)(1-\gamma) + \lambda} + \frac{\|C_{YX}\|\|\Delta_X\| + \|\Delta_{YX}\|\|C_X\| + \|\Delta_{YX}\|\|\Delta_X\|}{v(C_X)^2(1-\gamma)^2 + \lambda}$$

where

$$\gamma = \frac{t_x^2\log(2d_X/\epsilon)}{v(C_X)N}$$

Proof. It follows by applying Corollaries 1,2,3 to Lemma 6. Also by union bound, so the total events are bounded $1 - 3\epsilon$

□

Theorem 2. ***Grabner 1997 (3.3)*** ***Grabner & Prodinger (1997)*** *Consider there are n independent copies* $X_1, \ldots X_n$ *i.i.d negative binomial random variables, with parameters defined as* $\mathcal{NB}(b, p)$, *and our goal is to calculate the expectation of the maximum of these N random variables* $\mathbb{E}_n = \mathbb{E}\{\max(X_1, \ldots, X_n)\}$ *then we have following asymptotic solution*

$$\begin{aligned}\mathbb{E}_n &= \log_{\frac{1}{q}}(n) + (b-1)\log_{\frac{1}{q}}\log_{\frac{1}{q}}(n) + (b-1)\log_{\frac{1}{q}}p + (b-1) - \log_{\frac{1}{q}}(b-1)! + \frac{1}{2} + \frac{\gamma}{\log_{\frac{1}{q}}(1/q)} \\ &\quad + F(\log_{\frac{1}{q}}(n) + (b-1)\log_{\frac{1}{q}}\log_{\frac{1}{q}}(n) + (b-1)\log_{\frac{1}{q}}p - \log_{\frac{1}{q}}(b-1)!) + o(1) \qquad (26)\end{aligned}$$

(where F is a periodic $C^\infty -$ *function of period 1 and mean value 0 whose Fourier-coefficients are given by* $\hat{F}(k) = -\frac{1}{\log(\frac{1}{q})}\Gamma(-\frac{2k\pi i}{\log(\frac{1}{q})})$ *for* $k \in \mathbb{Z} \setminus \{0\}$, *and* $q = 1 - p$*)*

We omit the proof, interested readers could go to Grabner & Prodinger (1997) for details.

E.1 Proof of Theorem 1

We first prove the bound for $\left\|\widehat{Q}_{i,j} - Q_{i,j}\right\|$.

Proposition 2. *Let π_Θ be a data collection policy and $\mathcal{H}$ is the range of π_Θ on joint histories. If Equation 2 used, then for all $h \in \mathcal{H}$ and any $\epsilon \in (0,1)$, $\left\|\widehat{Q}_{i,j}(\psi^h) - Q_{i,j}(\psi^h)\right\|$ is bounded as below with probability at least $1-3\epsilon$.*

$$\left\|\widehat{Q}_{i,j} - Q_{i,j}\right\| \leq \sqrt{\frac{u\left(C_{\psi_i^o|\psi_{i,j}^h}\right)}{v\left(C_{\psi_j^a|\psi_{i,j}^h}\right)^3}\frac{\|\Delta_1\|^2 + 2u\left(C_{\psi_j^a|\psi_{i,j}^h}\right)\|\Delta_1\| + \lambda}{v\left(C_{\psi_j^a|\psi_{i,j}^h}\right)(1-\gamma)+\lambda}} + \frac{\left\|C_{\psi_i^o\psi_j^a|\psi_{i,j}^h}\right\|\|\Delta_1\| + \|\Delta_2\|\left\|C_{\psi_j^a|\psi_{i,j}^h}\right\| + \|\Delta_2\|\|\Delta_1\|}{v\left(C_{\psi_j^a|\psi_{i,j}^h}\right)^2(1-\gamma)^2+\lambda}$$

where Δ_1 follows the bound 29 and Δ_2 follows the bound 27, and $\gamma = \frac{t_{A_j}^2 \log(2d_{A_j}/\epsilon)}{v\left(C_{\psi_j^a}\right)N}$

Proof. Let $T_{i,j}$ is the tensor such that $C_{\psi_i^o\psi_j^a|\psi_{i,j}^h} = T_{i,j} \times_h \psi_{i,j}^h$, and $U_{i,j}$ is the tensor such that $C_{\psi_j^a|\psi_{i,j}^h} = U_{i,j} \times_h \psi_{i,j}^h$ and for simplicity without loss meaning, we use $C_{\psi_j^a|\psi_{i,j}^h}$ to denote $C_{\psi_j^a\psi_j^a|\psi_{i,j}^h}$. Then we have

$$\left\|\widehat{C}_{\psi_i^o\psi_j^a|\psi_{i,j}^h} - C_{\psi_i^o\psi_j^a|\psi_{i,j}^h}\right\| \leq \left\|\widehat{T}_{i,j} - T_{i,j}\right\|\|\psi_{i,j}^h\|$$
$$\left\|\widehat{C}_{\psi_j^a|\psi_{i,j}^h} - C_{\psi_j^a|\psi_{i,j}^h}\right\| \leq \left\|\widehat{U}_{i,j} - U_{i,j}\right\|\|\psi_{i,j}^h\|$$

We finish the above proof by proofing the $\|T_{i,j} - T_{i,j}\|$ and $\|U_{i,j} - U_{i,j}\|$ are bounded by using Corollary 4.

$$\left\|\widehat{T}_{i,j} - T_{i,j}\right\|\|\psi_{i,j}^h\| \leq t_h\sqrt{\frac{u\left(C_{\psi_i^o,\psi_j^a}\right)}{v\left(C_{\psi_{i,j}^h}\right)}}\left(\frac{\sqrt{v\left(C_{\psi_{i,j}^h}\right)}\alpha + \lambda}{v\left(C_{\psi_{i,j}^h}\right)(1-\gamma)+\lambda}\right) + \frac{\beta}{v\left(C_{\psi_{i,j}^h}\right)(1-\gamma)+\lambda}, \tag{27}$$

where

$$\begin{aligned}
\alpha &= 2t_h\sqrt{\frac{2\log(2d_h/\epsilon)}{N}} + \frac{2\log(2d_h/\epsilon)}{3N}\left(\frac{t_h^2}{\sqrt{v\left(C_{\psi_{i,j}^h}\right)}} + t_h\right), \\
\beta &= 2t_{O_i}t_{A_j}t_h\sqrt{\frac{\log(d_{O_i}d_{A_j}+d_h)/\epsilon}{N}} + \frac{4t_{O_i}t_{A_j}t_h\log((d_{O_i}d_{A_j}+d_h)/\epsilon)}{3N}, \\
\gamma &= \frac{t_h^2\log(2d_h/\epsilon)}{v\left(C_{\psi_{i,j}^h}\right)N};
\end{aligned} \tag{28}$$

and

$$\left\|\widehat{U}_{i,j} - U_{i,j}\right\|\|\psi_{i,j}^h\| \leq t_h\sqrt{\frac{u\left(C_{\psi_j^a}\right)}{v\left(C_{\psi_{i,j}^h}\right)}}\left(\frac{\sqrt{v\left(C_{\psi_{i,j}^h}\right)}\alpha + \lambda}{v\left(C_{\psi_{i,j}^h}\right)(1-\gamma)+\lambda}\right) + \frac{\beta}{v\left(C_{\psi_{i,j}^h}\right)(1-\gamma)+\lambda}, \tag{29}$$

where

$$\begin{aligned}
\alpha &= 2t_h\sqrt{\frac{2\log(2d_h/\epsilon)}{N}} + \frac{2\log(2d_h/\epsilon)}{3N}\left(\frac{t_h^2}{\sqrt{v\left(C_{\psi^h_{i,j}}\right)}} + t_h\right), \\
\beta &= 2t_{A_j}t_h\sqrt{\frac{\log(d_{A_j}+d_h)/\epsilon}{N}} + \frac{4t_{A_j}t_h\log((d_{A_j}+d_h)/\epsilon)}{3N}, \\
\gamma &= \frac{t_h^2\log(2d_h/\epsilon)}{v\left(C_{\psi^h_{i,j}}\right)N}.
\end{aligned} \tag{30}$$

Then using the equation 2 and corollary 5 to obtain the bound for $Q_{i,j}$

$$\begin{aligned}
\left\|\widehat{Q}_{i,j} - Q_{i,j}\right\| \leq & \sqrt{\frac{u\left(C_{\psi^o_j|\psi^h_{i,j}}\right)}{v\left(C_{\psi^a_j|\psi^h_{i,j}}\right)^3}\frac{\|\Delta_1\|^2 + 2u\left(C_{\psi^a_j|\psi^h_{i,j}}\right)\|\Delta_1\| + \lambda}{v\left(C_{\psi^a_j|\psi^h_{i,j}}\right)(1-\gamma)+\lambda}} \\
& + \frac{\left\|C_{\psi^o_i\psi^a_j|\psi^h_{i,j}}\right\|\|\Delta_1\| + \|\Delta_2\|\left\|C_{\psi^a_j|\psi^h_{i,j}}\right\| + \|\Delta_2\|\|\Delta_1\|}{v\left(C_{\psi^a_j|\psi^h_{i,j}}\right)^2(1-\gamma)^2+\lambda},
\end{aligned}$$

where Δ_1 follows the bound 29 and Δ_2 follows the bound 27, and $\gamma = \frac{t^2_{A_j}\log(2d_{A_j}/\epsilon)}{v\left(C_{\psi^a_j}\right)N}$

□

Now we start to prove our theorem 1.

Proof. The equation 1 says:

$$Q_{t,i} := g\left(\{Q_{t,i,j}\}_{j=1}^n\right) = \sum_j Q_{t,i,j},$$

here we assume the agents are homogeneous, in other words, each pair of $\{(o_i, a_i)\}_{i=1}^n$ coming from the same spaces $\mathcal{O}, \mathcal{A}$. They are permutation invariant and their identities do not matter. Thus, the bound of $Q_{i,j}$ is invariant to agents. Under the assumption of static fully complete graph, for any agent $Q_i = \sum_{j=1}^n Q_{i,j}$, thus

$$\begin{aligned}
\left\|\widehat{Q}_i - Q_i\right\| &= \left\|\sum_j \widehat{Q}_{i,j} - \sum_j Q_{i,j}\right\| \\
&\leq n\left\|\widehat{Q}_{i,j} - Q_{i,j}\right\|
\end{aligned} \tag{31}$$

□

E.2 Proof of Lemma 1

Here we prove Lemma 1.

Proof. Equation 4 says

$$Q_{t,i} := g\left(\{Q_{t,i,j}\}_{j=1}^n\right) = \sum_j I_{i,j}Q_{t,i,j},$$

where $I_{i,j}$ is an indicator function to denote if two agents are connected. Under the assumption that the static non-complete graph has maximum of number of degrees k and the agents are homogeneous, we have $\sum_j I_{i,j} <= k$. Thus,

$$\begin{aligned}
\left\|\widehat{Q}_i - Q_i\right\| &= \left\|\sum_j I_{i,j}\widehat{Q}_{i,j} - \sum_j I_{i,j}Q_{i,j}\right\| \\
&\leq k\left\|\widehat{Q}_{i,j} - Q_{i,j}\right\|
\end{aligned} \tag{32}$$

□

E.3 INFORMAL PROOF OF LEMMA 2

Here we make some intuitions for the proof of Lemma 2. As we already give the proof sketch in our main paper. Lemma 2 is a direct application of Theorem 2. Here our random variable $J_1, \cdots, J_n \sim \mathcal{NB}(r, p)$, where $\{(J_i)\}_{i=1}^n$ represents the number of time points node i needs before it meets node j number of r times.

For the complete static graph, we need at least N sample for the bound in equation 3 to be valid; in other words, we need the trajectory to run at least N time points to collect enough data to estimate our conditional operator accurately $Q_{i,j}$.

For the dynamic graph, each time t, the two nodes are randomly connected with probability p, if it connects, then we can obtain a valid sample to estimate $Q_{i,j}$; if not, then we skip to the next time step. For the dynamic graph node i, if we take the union set of the nodes i connected over the trajectory path, then the union set could form a static complete graph. We say the two graphs are equivalent. The number of time points needed by i until the Nth connection with j for each of the pairs (i, j) follows the same distribution $J \sim \mathcal{NB}(N, p)$. Then we are interested in the expectation of the maximum of $J_1, \ldots, J_{n-1}$, we denote it as $J_{\{1,\ldots,n-1\}}$.

$\mathbb{E}\{J_{\{1,\ldots,n-1\}}\}$ means on average, how many time points (N') we need for all nodes other than i at least meets N times with i. Obviously, this $N' >= N$ since $p \in [0, 1]$. The calculation of this expectation is solved by Grabner & Prodinger (1997). And we also put their result in Theorem 2.