Robust Individualistic Learning in Many-Agent Systems

Keyang He[1], Prashant Doshi[1], and Bikramjit Banerjee[2]

[1] THINC Lab, School of Computing, University of Georgia, 415 Boyd Research and Education Center, Athens, GA 30602, USA
{keyang, pdoshi}@uga.edu
http://thinc.cs.uga.edu/

[2] School of Computing Sciences and Computer Engineering, University of Southern Mississippi, 118 College Drive #5106, Hattiesburg, MS 39406, USA
Bikramjit.Banerjee@usm.edu
https://sites.usm.edu/banerjee

Abstract. A recent multiagent reinforcement learning method, the interactive advantage actor critic (IA2C), engages in individual agent training coupled with decentralized execution by predicting the other agents' actions from possibly noisy observations. This paradigm differs from the prevailing methods that engage in centralized training, which involves knowing varied information about the agents during the learning. Such epistemic commitments may not be feasible in live adversarial settings and other agents in practice could have learned differently. Against explicitly modeling others, in this paper we let IA2C utilize a specific encoder-decoder architecture, which allows it to learn a latent representation of the hidden state and other agents' actions that does not encounter some of the challenges of explicit modeling. Our experiments in two domains – each populated by many agents – reveal that the latent IA2C not only learns better quality behaviors but also improves sample efficiency by reducing variance and converging faster. We add further realism by introducing open versions of these domains where the agent population may change over time, and evaluate on these instances under the added uncertainty with good results.

1 Introduction

Recent multiagent RL (MARL) techniques either generalize policy optimization or the actor-critic (AC) architecture to multiagent settings while predominantly following the centralized training and decentralized execution (CTDE) paradigm. These techniques maintain a joint-action value or advantage function whereas each actor learns individual policies that map an agent's own observations to its actions. For example, multiagent proximal policy optimization (MA-PPO) [29] updates a centralized value function with action-observation inputs from all agents and utilizes this value function to learn improved policies for each agent. AC based methods such as MADDPG [13], LOLA [4], and COMA [5] all subscribe to CTDE with the agents exchanging varying types of information to facilitate a centralized critic. *However, such information exchanges may not be feasible in live competitive or adversarial settings and the learning agent could be sharing its world with others who may have learned differently.* This motivates an individualistic or decentralized training and decentralized execution (DTDE), as

adopted by AC methods such as IA2C [6] and independent Q-learning techniques such as I2Q [10]. IA2C takes a subjective perspective (egocentric) to the learning by modeling other agents. Such decentralized approaches also better align with Russell's [21] and Shoham's [23] well-known AI stance of designing an optimal learning agent effective in its environment.

A recent method, labeled IA2C^{++} [7] scales DTDE to *many-agent* systems (fifty and more) using Dirichlet-multinomial models to predict other agents' actions. It updates the Dirichlet-multinomial model using private observations of the agent but this is done outside the critic network. While IA2C^{++} is shown to converge to good quality policies in mixed-motive settings, our experiments reveal that the method suffers from high variance, particularly in domains where some states admit a single optimal action which the Dirichlet-multinomial is unable to predict with probability one and also because the model's predictions change drastically when the sample size is initially low.

Against this explicit modeling, we present a new method for DTDE learning that uses an encoder-decoder architecture to effectively simulate the Dirichlet-multinomial model and its update using a neural pipeline, yielding an end-to-end neural network architecture which facilitates training. Latent in the encoder-decoder is a representation of the hidden state, its update, and the other agents' mean-field action, which is input to the advantage function in the AC. Use of this latent variable in the critic has a regularizing effect, leading to lower variance and improved sample complexity, which has been corroborated in recent work [12,28].

We evaluate this new method in two domains, Org [6] and MAgent [30], with up to one hundred agents. Org models a typical business organization featuring a mix of cooperation for achieving the overall improvement of the organization and individual employee competition. In this paper, we let Org be an *open* many-agent system where the supervisory agents may add or remove employees thereby exhibiting agent openness. MAgent is a multiagent testbed focusing on battlefield simulation of two groups of agents, for which we also introduce an open variant. In several instances of these domains, we find that the latent IA2C with its individualistic perspective converges to significantly better quality policies compared to baselines, exhibits less variance and is significantly more sample efficient than relevant baselines. Consequently, the key contribution of this paper is a state-of-the-art RL method which aligns with the DTDE paradigm and is designed to generalize and scale to open contexts shared with many agents.

2 Background: Individualistic Learning in Multiagent Contexts

Non-stationary environments are a primary challenge for MARL. In cooperative settings, agents may directly share their policies with others. If this is not feasible, then inferring other agents' actions based on their past action histories is a common workaround. However, where communication between agents is not possible and other agents' actions are not perfectly observed, frequentist inference methods such as maximum likelihood estimation struggle to provide accurate predictions. On the other hand, methods such as IA2C engage in an individualistic perspective to RL training, which makes it better suited for mixed cooperative-competitive environments compared to CTDE methods [5,13,4],

which require information exchanges that may not be feasible among live competitive agents. We briefly review IA2C and discuss how it performs RL in many-agent settings.

2.1 Interactive A2C with Explicit Models

Interactive advantage actor-critic (IA2C) [6] is designed for egocentric RL in partially observable Markovian settings shared with other agents. Each agent in IA2C has its own critic and actor, the critic mapping individual observations to joint action values, $Q_0(\mathbf{o}, a_0, \mathbf{a}_{-0})$, in terms of the agent's own reward function and the actor mapping individual observations to individual action probabilities, $\pi_{0,\boldsymbol{\theta}}(a_0|\mathbf{o})$, $\boldsymbol{\theta}$ is its parameters. Observation stacking is used to manage the partial observability of the state. IA2C estimates advantages as

$$A_0(\mathbf{o}, a_0, \hat{\mathbf{a}}_{-0}) = avg[r + \gamma Q_0(\mathbf{o}', a_0', \hat{\mathbf{a}}'_{-0}) - Q_0(\mathbf{o}, a_0, \hat{\mathbf{a}}_{-0})]$$

while the actor's gradient is estimated as

$$avg[\nabla_{\boldsymbol{\theta}} \log \pi_{0,\boldsymbol{\theta}}(a_0|\mathbf{o})\ A_0(\mathbf{o}, a_0, \hat{\mathbf{a}}_{-0})]$$

where r, o' and a_0' are samples from the trajectory, $\hat{\mathbf{a}}_{-0}$ and $\hat{\mathbf{a}}'_{-0}$ are *predicted* actions of the other agents, and the avg is taken over sampled trajectories. IA2C does not require access to other agents' actions and/or gradients. Rather, agents hypothesize other agents' possible models and maintain belief distributions over the set. Let $\mathbf{a}_{-0} = \langle a_1 \ldots, a_N \rangle$ for N other agents. Given the ego-agent 0's prior belief b_0, action a_0, as well as its public and private observations o_0', ω_0', the agent updates its belief over agent j's model, $m_j' = \langle \pi_j', h_j' \rangle$,

$$b_0'(m_j'|b_0, a_0, o_0', \omega_0') \propto \sum_{\mathbf{a}_{-0}} \Bigg(\prod_{k=1, k\neq j}^{N} \sum_{m_k \in M_k} b_0(m_k) Pr(a_k|m_k) \sum_{m_j \in M_j} b_0(m_j)$$

$$\times\ Pr(a_j|m_j)\ \delta_K(\pi_j', \pi_j)\ \delta_K(APPEND(h_j, \langle a_j, o' \rangle), h_j') \Bigg)\ W_0(a_0, \mathbf{a}_{-0}, \omega_0') \quad (1)$$

where public and private observations are noisy observations of the state and other agents' actions; m_j denotes agent j's model: π_j is j's policy and h_j is its action-observation history required to predict j's action from its policy. W_0 is the private observation function that maps joint actions to the subject agent's private observations. δ_K is the Kronecker delta function and APPEND returns a string with the second argument appended to its first. The belief update is performed by a separate belief filter not integrated into the critic's network.

In experiments including on SMAC [22], IA2C shows robust performance in noisy environments and converges significantly faster to the optimal policy when compared to several CTDE methods that require communication among agents or rely on frequentist inference methods [8].

2.2 IA2C for Many-Agent Contexts

Another key challenge for MARL is the exponential growth of the joint action space with the number of agents. Often, many-agent environments naturally exhibit *action anonymity*, which implies that environment dynamics and rewards depend on the *action configuration* (notated henceforth as $\mathcal{C}$), which is the count distribution of actions in the population. Many-agent IA2C (IA2C^{++}) [7] – and other popular methods such as mean-field RL – utilizes this key insight of action anonymity thereby using configurations which scale polynomially with the number of agents rather than using joint action vectors that scale exponentially.

IA2C^{++} adapts Eq. 1 which updates agent 0's belief over one other agent's possible models to using action configurations instead of joint actions.

$$b_0'(m_j'|b_0,a_0,o_0',\omega_0') \propto \sum_{m_j\in M_j} b_0(m_j) \sum_{a_j} Pr(a_j|m_j) \sum_{\mathcal{C}\in\boldsymbol{\mathcal{C}}^{\mathbf{a}-\mathbf{o}}} Pr(\mathcal{C}|b_0(M_1),b_0(M_2),$$
$$\ldots,b_0(M_N))\; W_0(a_0,\mathcal{C},\omega_0')\; \delta_K(\pi_j,\pi_j')\; \delta_K(APPEND(h_j,\langle a_j,o'\rangle),h_j'). \tag{2}$$

The term $Pr(\mathcal{C}|b_0(M_1),\ldots,b_0(M_N))$ is the probability of configuration $\mathcal{C}$ in the distribution over the set of configurations $\boldsymbol{\mathcal{C}}^{\mathbf{a}-\mathbf{o}}$. The distribution is obtained using the dynamic programming procedure of Jiang *et al.* [9]. The algorithm takes as input N beliefs each of size $|M_j|$ compared to a single large belief of exponential size $|M_j|^N$.

The method above requires a pre-defined model set for each other agent. IA2C^{++}DM removes that assumption by using a Dirichlet-multinomial model to approximate action distributions of the entire agent population, which can then yield the mean action. Suppose the action space for the homogeneous agent population is $\{a_1,a_2,\ldots,a_{|A|}\}$, and for each agent i, the probability of picking action a_k is θ_k. Let $\boldsymbol{\theta}=(\theta_1,\theta_2,\ldots,\theta_{|A|})$ and $\boldsymbol{\theta}$ has a Dirichlet-multinomial prior distribution with parameter $\boldsymbol{\alpha}$ if

$$Pr(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_k \alpha_k)}{\Pi_n \Gamma(\alpha_k)} \Pi_k \theta_k^{\alpha_k-1} \tag{3}$$

where $\alpha_k>0$ for all k, $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\ldots,\alpha_N)$, and $\sum_k\theta_k=1$. We can write this as $\boldsymbol{\theta}\sim Dir(\boldsymbol{\alpha};\boldsymbol{\theta})$. Then, probability of an action configuration $\mathcal{C}$ is expressed as:

$$Pr(\mathcal{C}|\boldsymbol{\theta}) = Pr(\#a_1,\#a_2,\ldots,\#a_{|A|}|\boldsymbol{\theta}) = \Pi_{k=1}^{|A|}\theta_k^{\#a_k} \tag{4}$$

where $\#a_k$ is the number of agents selecting the kth action. After executing action a_0 and receiving (noisy) private observation $\omega_0'=(\#a_1,\#a_2,\ldots,\#a_{|A|})$, the subject agent's Dirichlet-multinomial distribution can be updated by noting that the posterior $Pr(\boldsymbol{\theta}|a_0,\omega_0')$ is proportional to:

$$\propto \sum_{\mathcal{C}} W_0(a_0,\mathcal{C},\omega_0')Pr(\mathcal{C}|\boldsymbol{\theta})\cdot Dir(\boldsymbol{\alpha};\boldsymbol{\theta}) = \sum_{\mathcal{C}} W_0(a_0,\mathcal{C},\omega_0')Dir(\boldsymbol{\alpha}+\mathcal{C};\boldsymbol{\theta}) \tag{5}$$
$$\approx Dir(\boldsymbol{\alpha}+\mathcal{C}';\boldsymbol{\theta}) \tag{6}$$

Equation 5 makes a simplifying modeling assumption that the distribution over true action counts of other agents, $\mathcal{C}$, is conditionally independent of a_0 given $\boldsymbol{\theta}$. For the sake

of tractability, the last step is approximated as a single component, $Dir(\boldsymbol{\alpha} + \mathcal{C}'; \boldsymbol{\theta})$. $\mathcal{C}'$ can be calculated in several alternative ways as discussed by He et al. [7], and in this paper we use the *rectified* method.

In IA2C^{++}, the critic network estimates advantage as:

$$A_0(\mathbf{o}, a_0, \mathcal{C}^{\mathbf{a}_{-\mathbf{o}}}) = avg[r + \gamma\, Q_0(\mathbf{o}', a_0', \mathcal{C}^{\mathbf{a}'_{-\mathbf{o}}}) - Q_0(\mathbf{o}, a_0, \mathcal{C}^{\mathbf{a}_{-\mathbf{o}}})]$$

while the actor network's gradient is:

$$avg\,[\nabla_{\boldsymbol{\theta}}\, log\, \pi_{0,\boldsymbol{\theta}}(a_0|\boldsymbol{o})\; A_0(\mathbf{o}, a_0, \mathcal{C}^{\mathbf{a}_{-\mathbf{o}}})] \tag{7}$$

where r, o', and a_0' are samples, $\mathbf{a}_{-\mathbf{o}}$ and $\mathbf{a}'_{-\mathbf{o}}$ are the predicted joint actions of the other agents for the current and next step, respectively, replaced by their corresponding configurations, and avg is taken over the sampled trajectories. The predicted action for the next time step is sampled from the updated Dirichlet distribution.

Empirically, IA2C^{++}DM benefits from large agent populations with the Dirichlet-multinomial model accurately and efficiently predicting the population configuration. The noised private observations stymie the convergence of previous many-agent RL methods with their eventual learned policies being far from optimal.

3 Latent Interactive A2C

We present **l**atent **i**nteractive A2C (LIA2C), which utilizes an encoder-decoder to model the dynamism of the hidden state and the predicted actions of the agent population as latent variables, for RL in partially observable multiagent settings.

3.1 Encoder-Decoder for Implicit Modeling

We aim to replace the explicit belief filter module in the critic of IA2C^{++} with an encoder-decoder combine. Figure 1 depicts the neural network architecture of this encoder-decoder.

At each time step t, the inputs to the encoder network are the public observation o^t, private observation ω^t, self action a^t, and action distribution of the agent population θ^t, which is initially uniform. The encoder generates a latent embedding z^t based on the inputs and label. The latent embedding z^t is sent to the actor-critic network for RL and the decoder network for reconstruction. Given z^t, the decoder generates the public observation o^{t+1} of the next time step through the observation reconstruction head f_d^o, and the updated action distribution of the agent population θ^{t+1} through the model reconstruction head f_d^θ. The loss function of the encoder-decoder network is defined as,

$$\begin{aligned}\mathcal{L}(o^t, \omega^t, a^t, \theta^t) =& \frac{1}{H}\sum_{t=1}^{H}[(f_d^o(z^t) - o^{t+1})^2 - \log Pr(f_d^\theta(z^t)|\omega^t, a^t) \\ &+ \mathcal{D}_{KL}(Dir(\mathcal{C}; f_d^\theta(z^t))||Dir(\alpha + \mathcal{C}'; \theta))] \\ =& \frac{1}{H}\sum\nolimits_{t=1}^{H}[(f_d^o(z^t) - o^{t+1})^2 - \log Dir(\alpha + \mathcal{C}'; f_d^\theta(z^t)) \\ &+ \mathcal{D}_{KL}(Dir(\mathcal{C}; f_d^\theta(z^t))||Dir(\alpha + \mathcal{C}'; \theta))].\end{aligned} \tag{8}$$

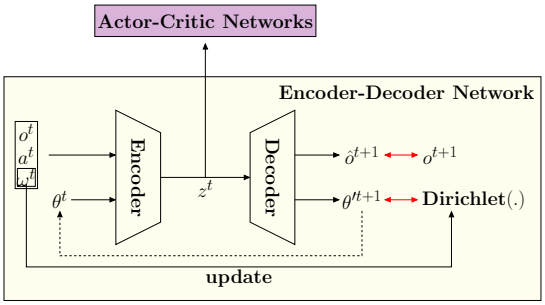


Fig. 1: The encoder-decoder network for modeling underlying state and agent population. Here, ω^t and o^t are the current private and public observations, a^t is the self action, θ^t is the predicted action distribution by the decoder from previous time step. $\hat{o}^{t+1}$ is the predicted next public observation, θ'^{t+1} is the updated action distribution of the agent population. Notice that the original Dirichlet also relies on ω^t for its update.

The first term of the loss function is the mean squared error between the reconstructed next observation $f_d^o(z^t) = \hat{o}_{t+1}$ and the true next observation o_{t+1}. Minimization of this loss term trains the parameters of both the encoder and the decoder to produce observation estimates close to o^{t+1} at the observation reconstruction head, affecting the latent representation z^t en route. The second loss term is the log posterior of the Dirichlet-multinomial model (see Eq. 6) that we want to maximize, hence it is negated. $\mathcal{C}'$ is calculated by the rectified method as mentioned in Section 2.2. The last loss term is the KL-divergence between the reconstructed model and the model obtained by maintaining a Dirichlet distribution using private observation history. Here θ in the second term under KL-divergence is sampled from the distribution $Pr(\boldsymbol{\theta}|\boldsymbol{\alpha} + \boldsymbol{\mathcal{C}'})$ following Eq. 3. The $\mathcal{C}$ in the first term under KL-divergence is sampled from $Pr(\mathcal{C}|f_d^\theta(z^t))$ following Eq. 4. Minimization of this loss term trains the parameters of both the encoder and the decoder to produce an action distribution θ'^{t+1} at the model reconstruction head that is close to the true Dirichlet-multinomial model, also affecting the latent representation z^t en route. Via the last two loss terms, *the network can be viewed as learning both to match the external belief update algorithm driven by the Dirichlet-multinomial model (Eq. 3–6) as well as to maximize the resulting posterior*. The latent representation z^t is learned as a by-product of this training.

Both these observations are important distinctions from previous methods that also model other agents using an encoder-decoder such as LIAM [17] and LILI [27]. Intuitively, LIA2C's network *learns to update belief* over hidden variables, thus creating a latent representation z^t in the belief space and does not require observation stacking. This representation serves as an appropriate augmentation for our decentralized critic.

What would be the consequence if the network did not learn to match the external belief update algorithm and instead learned its own internal version of such an algorithm? The corresponding loss function would not contain the last loss term, and the posterior maximization (second term) would be the sole driver of this aspect of learning. We conjecture that this version will be slower to converge. Additionally, it may learn similar policies as existing baselines that also do not explicitly model the belief update. We

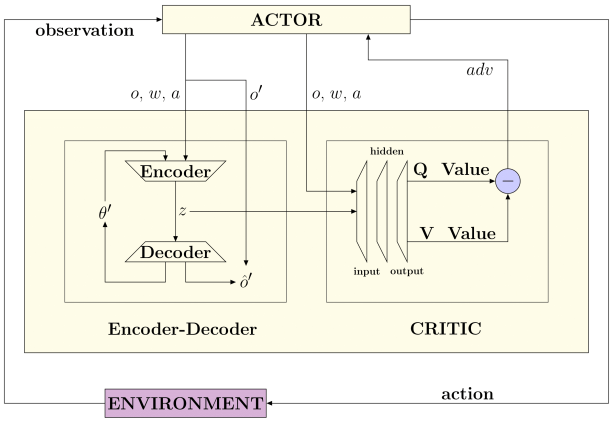


Fig. 2: The encoder-decoder network in LIA2C updates the model of the agent population. The critic's advantage function now depends on the latent embedding, self action, and observations instead of the configuration as in IA2C^{++}.

include this version as an ablation in our experiments, called LIA2C-w/oKLD, to evaluate these conjectures.

Note that the Dirichlet distributions are only needed for training. During execution, it is not required to maintain a Dirichlet distribution as actors do not use z^t or anything else that this network produces. Hence, both the critic and this encoder-decoder network can be discarded at execution time. During training, however, the encoder-decoder introduces an additional source of non-stationarity (apart from the inherent non-stationarity of decentralized training) since a mapping from θ^t to θ'^{t+1}—and hence, z^t—depends on the agents' time-varying policies.

3.2 Actor-Critic Network for DTDE RL

We augment the input of our critic network to include the latent embedding, z^t, from the encoder-decoder network. In particular, the Q-function of agent 0 is now $Q_0(o, a_0, z)$ (cf. the Q-function in Section 2.1). The advantage function of the critic is defined as,

$$A_0(o, a_0, z) = avg\left[r + \gamma\, Q_0(o', a'_0, z') - Q_0(o, a_0, z)\right] \tag{9}$$

where r, o', ω'_0, and a'_0 are samples, z and z' are latent embeddings of the encoder-decoder network, and avg is taken over the sampled trajectories. The agent population's predicted action for the next time step is obtained by sampling from the updated decoder model. Note that the gradients from critic update are not backpropagated to the encoder-decoder. This reduces the impact of the additional non-stationarity (explained at the end of the previous subsection) on the Q-values.

It might be desirable to augment the actor network's inputs to include the latent embedding as it would bring useful information about the hidden state to bear on the actor's computation. However, we avoid this for two reasons. The first reason is technical difficulty: to run the actor to output a^t, we cannot use z^t as one of its inputs because the

encoder itself requires a^t as one of its inputs. Using z^{t-1} as an actor input instead would introduce a one-step lag. The second reason is historical precedent. As Lee et al. [[12]] argue in their paper, "... the policy is not conditioned on the latent state, as this can lead to over-optimistic behavior since the algorithm would learn Q-values for policies that have perfect access to the latent state. Instead, the learned policy in our algorithm is conditioned directly on the past observations and actions. This has the additional benefit that the learned policy can be executed at run time without requiring inference of the latent state." This observation applies to LIA2C as well (but is surprisingly not followed by past methods LILI or LIAM). Consequently, we leave the actor network to recommend an action based solely on the public observation o. The latent state still has an impact on the actor's updates via the advantage term, as its gradient is

$$avg\ [\nabla_\theta \log \pi_\theta(a_0|o)\ A_0(o, a_0, z)]. \tag{10}$$

Figure 2 demonstrates the overall network architecture of LIA2C. The actor receives observation from the environment and sends self actions to the encoder-decoder network. The encoder-decoder network updates the approximated model θ' based on the actor's action and observations from environment. The latent embedding of the encoder-decoder network is sent to the critic for advantage computation. The critic network updates its parameters based on the latent embedding and environment reward, and sends the advantage value to the actor for its gradient update (Eq. 10).

4 Experiments

We implemented LIA2C in Python and evaluate its learning performance on multiple instances of two domains in reference to significant baselines. [3]

4.1 Open Organization Domain

A typical business organization (Org) features a mix of cooperation among the employees for improving the overall financial health of the organization and individual employee competition with each other for the predetermined bonus pool. Org [6] offers substantially more realistic challenges than previous MARL evaluation domains. One of these is that domains generally assume a closed system. In real-life organizations, employees may leave the organization due to retirement or job-hopping. The organization hires employees to fill job vacancies, or lays off employees due to business shrinkage. To simulate this, we introduce the *Open Org* domain in this paper that generalizes Org to an open multiagent system (see Fig. 3(a) for a visualization). Open Org contains two types of agents: *employee* and *manager*. Employees have the option to leave the organization. A manager can hire new employees or fire existing employees based on Org's current financial health level. Unlike existing open system domains where external agents enter the system periodically and/or randomly [18], in Open Org, the system openness is fully controlled by internal agents. The goal of the agents is to optimize their mixed-motive payoff under the uncertainty of agent openness.

[3] Our code, parameters, and domain specs are available at https://github.com/thinclab/IA2C/tree/LIA2C

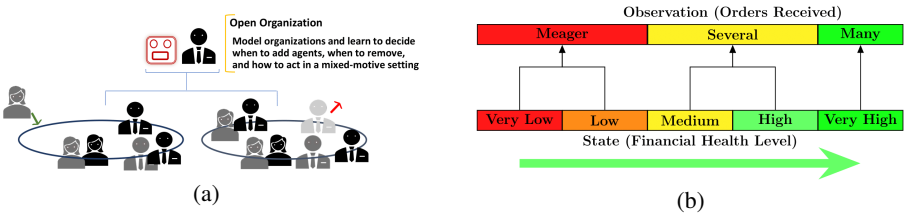


Fig. 3: (a) In Open Org, a manager can remove or add an employee to a group while any employee may resign. (b) States, observations, and their relationship in Org.

States and Observations Five states represent the organization's financial health levels as shown in Fig. 3(b) [6]: very low (s_{vl}), low (s_l), medium (s_m), high (s_h), and very high (s_{vh}). States are not observable by agents. Instead, agents receive observations of the amount of orders received by the organization that relates to states, and this information is *public*. Meager (o_e) would be observed when the organization is in either s_{vl} or s_l. Several (o_s) is observed when the organization is in either s_m or s_h. Many (o_m) is observed when the organization is in s_{vh}. Agents also receive *private* observations of other agent's actions. Private observations reveal other agent's true action with probability $1-\delta$, where $0 \leq \delta < 1$.

Action and State Transition Employees have four possible actions: self, balance, group, and resign. Self, as the label suggests, gives a high individual reward and no group reward; group gives a medium group reward and no individual reward; balance action gives a low individual reward and a low group reward. The resign action removes the employee from the organization. In this paper, we let Org contain a single manager only. The manager has six possible actions: self, balance, group, fire, and hire. Self, balance, and group are similar to the employee's actions. The fire action removes the most recently hired employee whereas hire adds a new employee to the organization.

Joint actions are determined by the number of agents picking self compared to those picking group, which transitions the state. More agents picking self yields self as the joint action, which lowers Org's financial health (by one level). More agents picking group leads to group as the joint action, which improves the organization's financial health (by one level). If the same number of agents pick both, the organization's financial health remains unchanged. The employee's action resign and manager's actions fire and hire do not impact the financial health of the organization. In addition, agents only receive noisy observations of others' actions. *To maximize its overall return, an agent needs to trade off individual benefits with group welfare to optimize long-term payoff.*

Rewards The employee's reward function is a sum of the agent's *individual* reward, $R_0^t \triangleq R_0(s^t, a_i^t)$, and *group* reward, $R_G^t \triangleq R_G(s^t, \boldsymbol{a}^t)$. The manager's reward function consists of three components: individual, group, and cost:

$$R_m^t = \sigma(\sum\nolimits_{n=1}^{N} R_n^t) + R_G^t - C^t$$

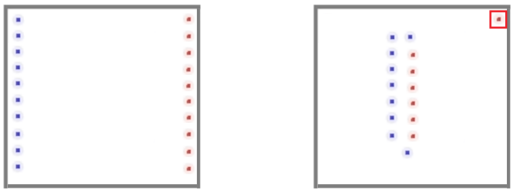
where σ is a Sigmoid function that simulates diminishing returns from the individual rewards of too many employees. $\sum_n R_n$ is the sum of all agents' individual rewards; R_G is the group reward determined by joint action; and $C = c \times \#$ of employees hired in this time step, where c is the cost of hiring one employee.

Agent Openness in Organization Any employee agent may leave the system at will or can be removed, and a new employee may enter the system. New agents may have no knowledge about the domain and start with exploration, or they may be pre-trained with domain knowledge. An *open* multiagent system is thus more challenging than a closed one in terms of non-stationarity.

In the extant open agent domains, the internal agents do not have complete control over the system openness. For example, Radulescu et al. [18] presents an open system simulation of highway driving. High speed vehicles randomly enter the system, while internal vehicles leave the system if they are too far away from the ego vehicle. Another example is the firefighting domain introduced previously [1]. Internal firefighters leave the system if they run out of fire suppressants. Only the firefighters who left the system previously can subsequently re-enter. When the internal agents have minimal control over the openness, rational decision making entails modeling the openness and predicting its impact. In contrast, in Open Org, internal agents can actively manage the system openness in order to reach optimality.

4.2 Evaluation Domains and Baselines

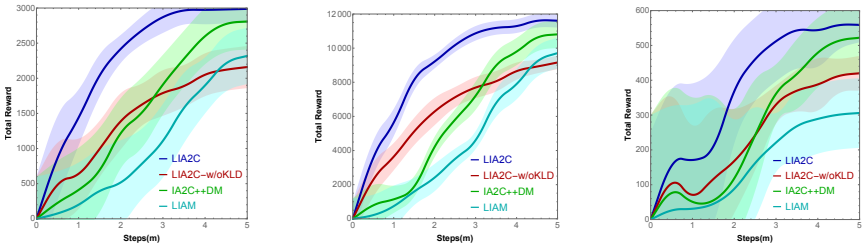
Our first experimental domain is Org; we use the original (closed) Org and our new Open Org described in Section 4.1. Our topology of Org is fully connected, where all agents share one neighborhood. For the Open Org domain, the cost c for hiring one employee is set to 1, and the individual reward for resign, hire, and fire are all set to 0. The environment initiates with one manager and one employee.



(a) initial position (b) re-spawned red agent

Fig. 4: (a) Agents from blue team are deployed on the left side, while agents from red team are deployed on the right side. (b) A red agent re-spawned after retreat.

Our second experimental domain is the Battlefield in the well-known MAgent environment [30] with 10 agents from each team as shown in Fig. 4. It is a grid world and each agent possesses a field of view centered on itself, with a radius of 5 cells. Agents can attack adversaries in adjacent cells. Public observation o_0 encompasses (x,y)



(a) 30 agents Org (b) 100 agents Org (c) Open Org

Fig. 5: LIA2C is more sample efficient as it requires much fewer samples to converge on average than baselines. This is in part because LIA2C's variance is lower than IA2C^{++}DM and LIAM. The improved performance is observed for both (a) 30 employees, and (b) as Org scales up to 100 employee agents. (c) Open Org starts with one manager and one employee. LIA2C and IA2C^{++}DM policies hire 8 employees each, which leads to the maximum payoff for the manager. All results are obtained over 5 runs.

coordinates of all agents within this observed area. Each agent receives the true (x, y) of all agents in its observed area with a 70% chance otherwise receives coordinates that are adjacent to the true positions. The reward function for an agent is mixed and consists of two parts: individual reward and group reward. An agent receives -1 for moving or attacking and 10 for eliminating an opponent. An agent gets a group reward of 2 if its team has more alive members than the opponent team in each time step. Otherwise, the agent receives -1 as the group reward. We introduce system openness to the Battlefield by adding a new action retreat. All blue agents are spawned on the left of the field and all red agents are spawned on the right of the field. If the agent is not eliminated at the current time step, the retreat action removes the agent from the current location and re-spawns the agent on the left or right depending on its color.

Baselines Our baseline methods include:
- an ablation of LIA2C without the KL-divergence term in the loss function (8) to clarify the value of the process model update, labeled as LIA2C-w/oKLD;
- prior approach IA2C^{++}DM described in Section 2.2; and
- a recent method LIAM [17] for modeling other agents that also uses an encoder-decoder.

Implementation details and hyperparameters are specified in Appendix A in the supplementary material. IA2C has previously been evaluated against CTDE techniques [8] so we do not include any here.

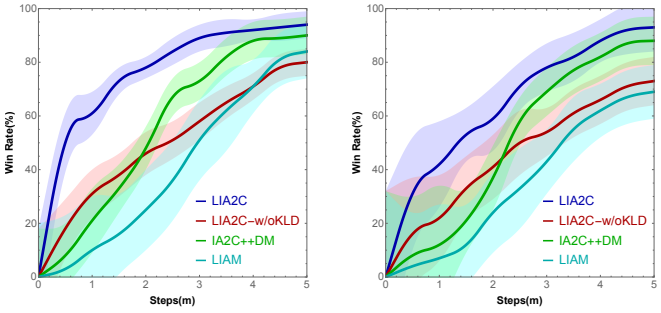
4.3 Comparative Performance

Org and Open Org We show the learning performance over 5 runs on the original Org with N=30, 100 agents in Fig. 5. Here, LIA2C converges to the optimal policy in roughly 3 million steps (total across all episodes) whereas IA2C^{++}DM takes more than 4.5 million steps to converge. For N=100, the latter converges to a policy whose value is significantly less (Wilcoxon signed rank test, n=5, p=0.008). The ablation LIA2C-w/oKLD fails to converge to the optimal policy in 5 million steps. LIAM converges to

suboptimal policies only indicating that it may not be robust to *noisy* observations of others' actions. Equally important, the average variance of LIA2C is 32% lower than IA2C^{++}DM and 44% less than LIAM. Indeed, we peeked into the action (Q-) values for the top two methods and the reduced variance of LIA2C is evident, as shown in Appendix C. This improved value performance is attributed to the joint latent embedding of the update of the hidden state and predictive action distributions as discussed later.

However, LIA2C took 9 ± 1 and 13 ± 1 hours to converge in the 30- and 100-agent Org contexts, respectively, on an Intel 64-bit i7 (4 cores, 3.6 GHz each) PC with Linux and 64GB memory. These are about an hour more in both cases than that of IA2C^{++}DM and the increase is due to the learning burden of the additional encoder-decoder component.

In the Open Org, the manager's individual reward component $\sigma(\cdot)$ is discounted by 0.9^{E-1}, where E is the number of employees hired. The manager's cost C for hiring an employee is 1 per time step. Figure 5c shows cumulative rewards for the different methods in Open Org. Notice the higher variances from all methods in Open Org. Clearly, the system openness has exacerbated the non-stationarity of the environment. Whereas LIA2C converges in roughly 3.5 million steps, IA2C^{++}DM converges after 5 million. Both LIA2C and IA2C^{++}DM learn to ultimately hire 8 employees, which is the optimal number of employees that should be hired according to the reward function. An employee's policy remains the same as learned in the original Org. LIA2C-w/oKLD and LIAM were not robust to the openness and failed to reach satisfactory policies with LIAM achieving just 50-60% of the returns of LIA2C.



(a) MAgent Battlefield (b) Open MAgent Battliefield

Fig. 6: (a) Within 2 million steps, LIA2C learns a policy that exhibits about 80% win rate, while baseline methods' show win rates less than 50%. (b) All methods take more steps to learn good policies due to the additional uncertainty introduced by system openness; LIA2C and IA2C^{+}+DM remain robust to it.

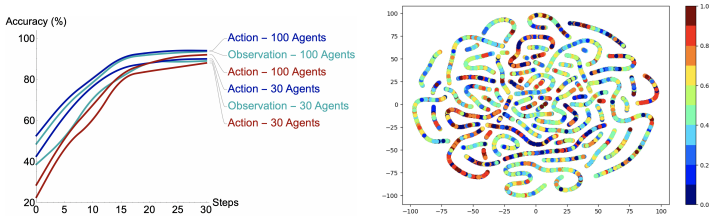
Battlefield and Open Battlefield In the MAgent Battlefield domains, all methods are tested against red agents pre-trained by independent A2C. Figures 6a and 6b show the averaged win rates of the methods over 5 runs for the closed and open variants, respectively.

For the closed domain, LIA2C learns high-quality policies (with win rates of greater than 80%) relatively early after 2 million steps while IA2C^{++}DM takes nearly twice as many to reach similar performance. LIAM converges to policies that are close to LIA2C and IA2C^{++}DM, but the variance is higher. On average, LIA2C's variance is 29% lower than IA2C^{++}DM and 41% lower than LIAM's.

Letting the agents retreat rather than get eliminated challenges LIA2C which takes nearly 4 million steps to converge to high-quality policies, and IA2C^{++}DM is able to learn policies that are very close to LIA2C. However, we noticed a drop in the performance of LIAM in the open MAgent Battlefield. As such, LIAM appears to struggle with learning accurate latent representations under openness and noisy observations.

4.4 Model's Predictive Performance

Prediction accuracy To better understand LIA2C's improved performance, we evaluate the novel encoder-decoder on its predictive accuracy as we increase the number of agents N. Figure 7a compares the prediction accuracy of agent population actions in terms of the predicted configuration (blue) and public observations (cyan) from the encoder-decoder in LIA2C, with prediction accuracy of these actions by the Dirichlet model in baseline IA2C^{++}DM (red). Note that the latter uses observation stacking and does not explicitly predict the underlying state. We observe that the encoder-decoder model reaches accurate predictions in few steps that are comparable to the explicit modeling in the baseline, and these improve with N.



(a) Prediction accuracy (b) t-SNE plot

Fig. 7: (a) Next-step action configuration and public observation prediction accuracy across time steps for increasing N. Blue and cyan lines are the prediction accuracy of LIA2C, and red lines are the prediction accuracy of IA2C^{++}DM. Accuracy of both models gets close, which validates the encoder-decoder architecture. (b) t-SNE plot of the encoder-decoder latent embeddings. Different colors represent different distributions of θ and different points represent different time steps.

Embedding visualization We visualize the latent embeddings learned by the encoder-decoder using a t-distributed stochastic neighbor embedding (t-SNE), a statistical method for visualizing the high-dimensional embeddings by giving each data point a location in a lower dimensional map [15]. Figure 7b shows the two-dimensional projection of the latent embeddings and Appendix B has additional informative visualizations. We observe that points with similar colors are connected or nearby, indicating a smooth transition of the belief over others' action distribution.

5 Related Work

CTDE has been a common paradigm in recent MARL research [13,5,29,26] and there is now better awareness of its limitations [14]. A thread within CTDE toward scaling to several agents has been value-decomposition to alleviate joint-action based value representation. For example, Sunehag *et al.* [24], Rashid *et al.* [20] decompose joint Q-functions as a function of individual agents' local Q-functions. In contrast, the DTDE paradigm is under investigated and LIA2C focuses on subjectively learning representations of the interactive state as well as latent state dynamics. A different thread under DTDE is independent Q-learning where I2Q [10] improves on IQL [25] by learning idealized transition functions that marginalize the actions of others under the strong assumption that both the transitions and policies are deterministic. This line of work does not aim to *intentionally* model the other interacting agents as we do.

Encoder-decoder in MARL The use of unsupervised learning techniques to learn low-dimensional representations of the environment has led to promising improvements in policy quality and sample efficiency. Recent works such as SIDE [28] and LIAM [17] focus on learning embedded variable models which represent jointly the state and environment dynamics, but not the state update, whereas LILI [27] learns the other agent's hidden strategy. Furthermore, LILI's use of maximum likelihood estimation to learn the latent variable cannot disentangle models of more than one other agent thereby limiting to N=2 and is better suited for perfect observability of the state and other's action.

Learning in open agent systems The number of agents in real-world environments may not be fixed – agents can leave or enter [3]. Such openness exacerbates the non-stationarity of the environment. Note that open agent systems generalize classic ad hoc teamwork [16] by letting agents exit the system and re-enter, and the agent interactions can be competitive. Past work [1,2] manages the uncertainty due to openness by modeling neighbors and learning to predict their future presence and behavior. Transfer learning approaches can also be used for RL in multiagent open systems with Radulescu *et al.* [18] offering a CTDE method by learning a single-agent policy and then transfering knowledge to a multi-agent setting. The policy is shared by direct parameter sharing and hence only applicable in homogeneous populations. A recent method called graph-based policy learning (GPL) [19] builds on graph neural networks (GNN) to learn agent models and joint-action value models in *fully observable* open multiagent systems. GPL learns joint action-values that model the effects of other agents' actions on the subject agent's returns, along with a GNN-based model to predict the teammates' actions. However, GPL may not scale to contexts containing more than a handful of agents.

6 Concluding Remarks

DTDE offers an alternative to CTDE RL that is better suited in certain contexts. This paper presented a new individual agent learning method called LIA2C that models the *updates of both the hidden physical state and others' predicted actions* as latent

while simultaneously utilizing the embedding for A2C in many-agent settings. LIA2C is a clear improvement on the previous best DTDE method that explicitly models the agent population's actions using a Dirichlet-multinomial and updates it directly using noisy private observations. Evaluations on several instances of two challenging domains populated by up to 100 agents demonstrates for the first time convergence robustness not only to noisy observations but also to agent openness where agents may exit and new ones may enter the system. Favorable comparisons with an ablation lend correctness to the loss function that facilitates learning.

This study opens up exciting future directions. The observed higher variance in open domains indicates a need for renewed attention to non-stationarity and the AI stance [23], long masked by a focus on CTDE. Indeed, volatility introduced by openness may very well be amenable to latent space modeling.

Acknowledgment: The authors acknowledge helpful comments and suggestions from anonymous reviewers. This work was supported in part by NSF grant #IIS-2312657. We thank Adam Eck, Leen-Kiat Soh, and seminar participants at the Universities of Waterloo, Canada and Northumbria, England for feedback.

References

1. Chandrasekaran, M., Eck, A., Doshi, P., Soh, L.: Individual planning in open and typed agent systems. In: Uncertainty in Artificial Intelligence (2016)
2. Eck, A., Shah, M., Doshi, P., Soh, L.K.: Scalable decision-theoretic planning in open and typed multiagent systems. In: Association for the Advancement of Artificial Intelligence (AAAI) (2020)
3. Eck, A., Soh, L.K., Doshi, P.: Decision making in open agent systems. AI Magazine **44**(4), 508–523 (2023)
4. Foerster, J., Chen, R.Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., Mordatch, I.: Learning with opponent-learning awareness. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). p. 122–130 (2018)
5. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: Association for the Advancement of Artificial Intelligence (AAAI) (2018)
6. He, K., Banerjee, B., Doshi, P.: Cooperative-competitive reinforcement learning with history-dependent rewards. In: Autonomous Agents and Multiagent Systems (AAMAS) (2021)
7. He, K., Doshi, P., Banerjee, B.: Reinforcement learning in many-agent settings under partial observability. In: Uncertainty in Artificial Intelligence (UAI) (2022)
8. He, K., Doshi, P., Banerjee, B.: Modeling and reinforcement learning in partially observable many-agent systems. Autonomous Agents and Multi-Agent Systems **38**(1), 10.1007/s10458–024–09640–1 (2024)
9. Jiang, A.X., Leyton-Brown, K., Bhat, N.A.: Action-graph games. Games and Economic Behavior **71**(1), 141–173 (2011)
10. Jiang, J., Lu, Z.: I2Q: A Fully Decentralized Q-Learning Algorithm . In: Proceedings of the Neural Information Processing System (NeurIPS). NeurIPS (2022)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) International Conference on Learning Representations (ICLR) (2015)
12. Lee, A.X., Nagabandi, A., Abbeel, P., Levine, S.: Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In: Neural Information Processing System (NeurIPS) (2020)

13. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Neural Information Processing Systems (NIPS) (2017)
14. Lyu, X., Xiao, Y., Daley, B., Amato, C.: Contrasting centralized and decentralized critics in multi-agent reinforcement learning. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. p. 844–852 (2021)
15. Maaten, L.V.D., Hinton, G.: Visualization data using t-sne. Journal of Machine Learning Research p. 2579–2605 (11 2008)
16. Mirsky, R., Carlucho, I., Rahman, A., Fosong, E., Macke, W., Sridharan, M., Stone, P., Albrecht, S.V.: A survey of ad hoc teamwork research. In: Baumeister, D., Rothe, J. (eds.) European Conference on Multi-Agent Systems. pp. 275–293 (2022)
17. Papoudakis, G., Christianos, F., Albrecht, S.V.: Agent modelling under partial observability for deep reinforcement learning. In: Neural Information Processing System (NeurIPS) (2021)
18. Radulescu, R., Legrand, M., Efthymiadis, K., Roijers, D.: Deep multi-agent reinforcement learning in a homogeneous open population. Artificial Intelligence p. 90–105 (November 2018)
19. Rahman, A., Hopner, N., Christianos, F., Albrecht, S.V.: Towards open ad hoc teamwork using graph-based policy learning. In: International Conference on Machine Learning (ICML) (2021)
20. Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning (ICML) (2018)
21. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (Third Edition). Prentice Hall (2011)
22. Samvelyan, M., Rashid, T., de Witt, C.S., Farquhar, G., Nardelli, N., Rudner, T.G.J., Hung, C.M., Torr, P.H.S., Foerster, J.N., Whiteson, S.: The starcraft multi-agent challenge. In: Neural Information Processing Systems (NeurIPS) (2019)
23. Shoham, Y., Powers, R., Grenager, T.: If multi-agent learning is the answer, what is the question? Artificial Intelligence **171**(7), 365–377 (2007). https://doi.org/https://doi.org/10.1016/j.artint.2006.02.006
24. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., Graepel, T.: Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Autonomous Agents and Multiagent Systems (AAMAS). p. 2085–2087 (2018)
25. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning (1993)
26. Wang, J., Ye, D., Lu, Z.: More centralized training, still decentralized execution: Multi-agent conditional policy factorization. In: Proceedings of the International Conference on Learning Representations (ICLR). ICLR (2023)
27. Xie, A., Losey, D., Tolsma, R., Finn, C., Sadigh, D.: Learning latent representations to influence multi-agent interactions. In: Conference on Robot Learning (CoRL). pp. 575–588. PMLR (2021)
28. Xu, Z., Bai, Y., Li, D., Zhang, B., Fan, G.: Side: State inference for partially observable cooperative multi-agent reinforcement learning. In: Autonomous Agents and Multiagent Systems (AAMAS) (2022)
29. Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of ppo in cooperative multi-agent games. In: Neural Information Processing Systems (NeurIPS) (2022)
30. Zheng, L., Yang, J., Cai, H., Zhou, M., Zhang, W., Wang, J., Yu, Y.: Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In: Association for the Advancement of Artificial Intelligence (AAAI) (2018)

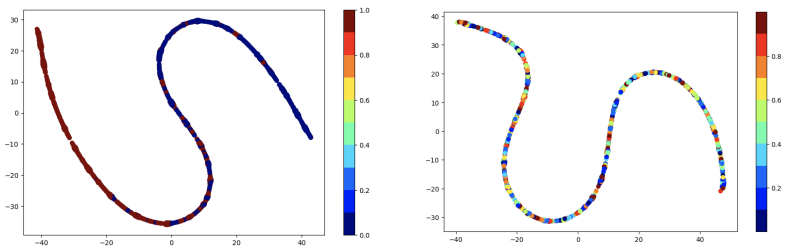
Technical Appendix

A Hyperparameters and Implementation Details

The encoder of LIA2C consists of one LSTM layer and a fully connected hidden layer with the ReLU activation function. The size of the latent layer is 32. The decoder's output head for θ is passed through a softmax activation function, and the observation reconstruction head is passed through a ReLU activation function. Both actor and critic networks consist of an input layer, two hidden layers, and an output layer. The first hidden layer uses $tanh$ activation function, and the second hidden layer uses a ReLU activation function. The optimizer used for the encoder-decoder and IA2C networks is Adam [11]. The learning rates are 3×10^{-4} for the encoder-decoder, 1×10^{-3} for the actor, and 1×10^{-4} for the critic. The batch size is set to 32. We use the hyperparameter values specified in [7,17] for the baselines IA2C^{++}DM and LIAM. The inputs to LIAM are concatenated public and private observations as well as self actions. In all experiments, public and private observation noise is set to 0.2. Episode length is set to 30.

B Additional t-SNE of the Latent Space

We show additional two-dimensional t-SNE projections of the latent embeddings for Org. Figure 8a shows these for a fixed policy of the agent, which picks Self for observation o_{many}, obtained when the underlying states could either be s_{high} or $s_{very\ high}$. Group is picked for all other observations. The two ends of the curved line in the plot correspond to the lowest ($s_{very\ low}$) and highest ($s_{very\ high}$) states, respectively. Only actions Self (denoted in blue) or Group (red) are picked in these states. The red points and blue points are mixed at the middle of the line, representing the belief transition in corresponding states.



(a) (b)

Fig. 8: (a) 2D representation of latent embeddings using t-SNE for a fixed policy of the subject agent. (b) 2D representation of latent embeddings using t-SNE for a uniform random policy.

Figure 8b shows the t-SNE plot of latent embeddings for a uniform random policy in Org. Differently colored points are uniformly distributed on the curved line, which are interpreted as corresponding to a uniformly distributed belief.

o	$\hat{\boldsymbol{a}}_{-0}$	a	**LIA2C**	**IA2C^{++}DM**
o_{meager}	self	self	3624 ± 221	3284 ± 495
o_{meager}	self	balance	3768 ± 218	2982 ± 511
o_{meager}	self	group	3828 ± 196	3136 ± 449
o_{meager}	balance	self	3850 ± 246	3252 ± 637
o_{meager}	balance	balance	3943 ± 234	3410 ± 594
o_{meager}	balance	group	4409 ± 252	3685 ± 573
o_{meager}	group	self	4637 ± 271	3806 ± 580
o_{meager}	group	balance	4585 ± 206	3591 ± 452
o_{meager}	group	group	4904 ± 232	4013 ± 515

Table 1: Mean and standard deviations of Q-values of LIA2C and IA2C^{++}DM in a critical state. o is the public observation, $\hat{\boldsymbol{a}}_{-0}$ is the predicted joint action of others, a is the self action. Q-values are the mean of 5 example runs.

C Comparing Q-Values

Recall that LIA2C shows significant improvement on IA2C^{++}DM's converged values for N=100 in (closed) Org. We investigated this result further and compare the action values $Q_0(o, a_0, \omega_0, z)$ of LIA2C and $Q_0(o, a_0, \mathcal{C}^{\mathbf{a}_{-0}})$ of IA2C^{++}DM. Table 1 shows the Q-values when the observation is o_{meager}, which is a critical state in Org.

Observe that (a) the mean Q-values of LIA2C are all much higher than those of IA2C^{++}DM, and (b) the Q-values of IA2C^{++}DM clearly demonstrate much higher standard deviations compared to LIA2C. As such, LIA2C exhibits better stability and convergent points.

D Learning Time Comparison

Finally, in this section of the Appendix, we measured the clock time in hours required by LIA2C and IA2C^{++}DM to converge in various contexts for the two domains Org and MAgent. Both methods either consume a similar amount of time to converge or IA2C^{++}DM is faster by an hour in some contexts, as shown in Table 2.

Method	**Domain**	**Convergence Time**
LIA2C	30-Agent Org	9 ± 1 hrs
IA2C^{++}DM	30-Agent Org	8 ± 1 hrs
LIA2C	100-Agent Org	13 ± 1 hrs
IA2C^{++}DM	100-Agent Org	13 ± 1 hrs
LIA2C	100-Agent Battle	13 ± 1 hrs
IA2C^{++}DM	100-Agent Battle	12 ± 2 hrs

Table 2: All experiments were conducted on a standard Linux PC with Intel 64-bit i7 processor (4 cores, 3.6 GHz) and 64 GB memory.