Fast Teammate Adaptation in the Presence of Sudden Policy Change (Supplementary Material)

Ziqian Zhang[1,*], **Lei Yuan**[1,2*], **Lihe Li**[1], **Ke Xue**[1], **Chengxing Jia**[1,2], **Cong Guan**[1], **Chao Qian**[1], **Yang Yu**[1,2†]

[1] National Key Laboratory for Novel Software Technology, Nanjing University
[2] Polixir Technologies
`191240076@smail.nju.edu.cn`, `{yuanl, xuek, jiacx, guanc}@lamda.nju.edu.cn`,
`lilhzq76@gmail.com`, `{qianc, yuy}@nju.edu.cn`

A RELATED WORK

Cooperative Multi-agent Reinforcement Learning Many real-world problems are made up of multiple interactive agents, which could usually be modeled as a multi-agent system [Dorri et al., 2018]. Among the multitudinous solutions, Multi-Agent Reinforcement Learning (MARL) [Zhang et al., 2021] has made great success profit from the powerful problem-solving ability of deep reinforcement learning [Wang et al., 2020]. Further, when the agents hold a shared goal, this problem refers to cooperative MARL [Oroojlooy and Hajinezhad, 2022], showing great progress in diverse domains like path finding [Sartoretti et al., 2019], active voltage control [Wang et al., 2021], and dynamic algorithm configuration [Xue et al., 2022a], etc. Many methods are proposed to facilitate coordination among agents, including policy-based ones (e.g., MADDPG [Lowe et al., 2017], MAPPO [Yu et al., 2022]), value-based series like VDN [Sunehag et al., 2018], QMIX [Rashid et al., 2018], or other techniques like transformer [Wen et al., 2022] and many variants [Gorsane et al., 2022], demonstrating remarkable coordination ability in a wide range of tasks like SMAC [Samvelyan et al., 2019], Hanabi [Yu et al., 2022], GRF [Wen et al., 2022]. Besides the mentioned approaches and the corresponding variants, many other methods are also proposed to investigate the cooperative MARL from other aspects, including casual inference among agents [Grimbly et al., 2021], policy deployment in an offline way for real-world application [Yang et al., 2021], communication [Zhu et al., 2022] for partial observability, model learning for sample efficiency improvement [Wang et al., 2022], policy robustness when perturbations occur [Guo et al., 2022], training paradigm like CTDE (centralized training with decentralized execution) [Lyu et al., 2021], testbed design for continual coordination validation [Nekoei et al., 2021], and ad hoc teamwork [Mirsky et al., 2022], etc.

Non-stationary is a longstanding topic in single-agent reinforcement learning (SARL) [Padakandla et al., 2019, Padakandla, 2020], where the environment dynamic (e.g., transition and reward functions) of a learning system may change over time. For SARL, most existing works focus on inter-episode non-stationarity, where decision processes are non-stationary across episodes, including multi-task setting [Varghese and Mahmoud, 2020], continual reinforcement learning [Khetarpal et al., 2022], meta reinforcement learning [Beck et al., 2023], etc., these problems can be formulated as a contextual MDP [Hallak et al., 2015], and could be solved by techniques like task embeddings learning. Other works also consider intra-episode non-stationarity, where an agent may suffer from dynamic drifting within one single episode [Kumar et al., 2021, Ren et al., 2022, Chen et al., 2022, Luo et al., 2022, Dastider and Lin, 2022, Feng et al., 2022]. Specifically, HDP-C-MDP [Ren et al., 2022] assumes the latent context to be finite and Markovian, and adapts a sticky Hierarchical Dirichlet Process (HDP) prior for model learning; while FANS-RL [Feng et al., 2022] assumes the latent context is Markovian and the environment can be modeled as a factored MDP; ESCP [Luo et al., 2022] considers the sudden changes one agent may encounter and obtains a robust policy via learning an auxiliary context recognition model. Experiments show that in environments with both in-distribution and out-of-distribution parameter changes, ESCP can not only better recover the environment encoding, but also adapt more rapidly to the post-change environment ; SeCBAD [Chen et al., 2022] further assumes the environment context usually stays stable for a stochastic period and then changes in an abrupt and unpredictable manner.

Differing from the SARL setting, non-stationarity is an inherent challenge for MARL, as the agent's policy may be instability

[*] The first two authors contributed equally.
[†] Corresponding Author

caused by the concurrent learning of multiple policies of other agents [Papoudakis et al., 2019]. Previous works mainly focus on solving the non-stationary in the training phase, using techniques like modeling others [Albrecht and Stone, 2018], meta policy adaptation [Kim et al., 2021], experience sharing [Christianos et al., 2020]. Other works concentrate on non-stationarity across episodes, like multi-task training [Qin et al., 2022], adversarial training [Xue et al., 2022b, Sun et al., 2023], training policy for zero-shot coordination [Hu et al., 2020a], other works also investigate the policy robust of a multi-agent system when perturbations happen in a different way, including the uncertainty in local observation [Lin et al., 2020], model function [Zhang et al., 2020], action making [Hu et al., 2020b] and message sending [Xue et al., 2022b, Sun et al., 2023] Although these approaches make progress somewhat, they leave the non-stationary caused by teammates' policy sudden change to a less involved but urgent-needed point.

B DETAILS ABOUT DERIVATION

B.1 DETAILS ABOUT CRP AND DERIVATION OF CLUSTER ASSIGNMENT

Chinese restaurant process (CRP) [Blei and Frazier, 2010] is a discrete-time stochastic process that defines a prior distribution over the cluster structures, which can be described simply as follows. A customer comes into a Chinese restaurant, he chooses to sit down alone at a new table with a probability proportional to a concentration parameter α or sits with other customers with a probability proportional to the number of customers sitting on the occupied table. Customers sitting at the same table will be assigned to the same cluster. Concretely, suppose that K customers sit in the restaurant currently. Let z_i be an indicator variable that tells which table that i^{th} sits on, and n_m denote the number of customers sitting at the m^{th} table, and M be the total number of non-empty tables. Note that $\sum_{m=1}^{M} n_m = K$. The probability that the $K+1^{\text{th}}$ customer sits at the m^{th} table is:

$$P(z_{K+1} = m|\alpha) = \frac{n_m}{K+\alpha}, \quad m = 1, ..., M. \tag{1}$$

There is some probability that the customer decides to sit at a new table and if the label of the new table is $M+1$, then:

$$P(z_{K+1} = M+1|\alpha) = \frac{\alpha}{K+\alpha}. \tag{2}$$

Taken together, the two equations characterize the CRP.

The cluster assignment of the k^{th} generated teammate group $P(v_k^{(m)}|\tau_k^S, \tau_k^A)$ can be decomposed:

$$\begin{aligned} & P(v_k^{(m)}|\tau_k^S, \tau_k^A) \\ & \quad = \frac{P(v_k^{(m}, \tau_k^S, \tau_k^A)}{P(\tau_k^S, \tau_k^A)} \\ & \quad = \frac{P(\tau_k^S, \tau_k^A|v_k^{(m)})P(v_k^{(m)})}{P(\tau_k^S, \tau_k^A)} \\ & \quad = \frac{P(\tau_k^A|\tau_k^S, v_k^{(m)})P(\tau_k^S|v_k^{(m)})P(v_k^{(m)})}{P(\tau_k^S, \tau_k^A)} \\ & \quad \propto P(\tau_k^A|\tau_k^S, v_k^{(m)})P(\tau_k^S|v_k^{(m)})P(v_k^{(m)}). \end{aligned} \tag{3}$$

As τ_k^S is a set of states that is not determined by the behavioral type of the teammates if neglecting the correlation in time dimensionality. $P(\tau_k^S|v_k^{(m)})$ can be considered as a constant. Accordingly, we would derive that $P(v_k^{(m)}|\tau_k^S, \tau_k^A) \propto P(v_k^{(m)})P(\tau_k^A|\tau_k^S; v_k^{(m)})$.

B.2 THE FULL DERIVATION OF $\mathcal{L}_{\text{GCE}}$

To guide the context encoder to identify and track the sudden change rapidly, ESCP [Luo et al., 2022] proposes the following optimization objective:

$$\mathcal{L}_{\text{GCE}} = \sum_{m=1}^{M} \mathbb{E}[||z_t^m - \mathbb{E}[z_t^m]||_2^2] + ||\mathbb{E}[z_t^m] - u^m||_2^2, \tag{4}$$

where z_t^m is the representation that context encoder embeds in the m^{th} environment, u^m is the oracle latent context vector, and M is the number of environments. For a better understanding, we would explain the meanings of symbols based on our setting in the following. So, z_t^m is the latent context vector when paired with teammates belonging to the m^{th} cluster, and u^m is the oracle behavior type.

Since we have no access to the oracle u^m, a set of surrogates that possesses large diversity is required to be separable and representative. Meanwhile, u^m is an intermediate variable used to guide $\mathbb{E}[z_t^m]$, so we could directly maximize the diversity of $\{\mathbb{E}[z_t^m]\}_{m=1}^M$ by maximizing the determinant of a relational matrix $R_{\{\mathbb{E}[z_t^m]\}}$. Each element of the relational matrix is:

$$R_{\{\mathbb{E}[z_t^m]\}}(i,j) = \exp(-\kappa ||\mathbb{E}[z_t^i] - \mathbb{E}[z_t^j]||_2^2), \tag{5}$$

where κ is the radius hyperparameter of the radius basis function applied to calculate the distance of two vectors. The objective function can now be written as:

$$\mathcal{L}_{\text{GCE}} = \sum_{m=1}^{M} \mathbb{E}[||z_t^m - \mathbb{E}[z_t^m]||_2^2] - \log\det(R_{\{\mathbb{E}[z_t^m]\}}). \tag{6}$$

To stabilize the training process, ESCP substitutes $\mathbb{E}[z_t^m]$ with $\bar{z}^m$, which is the moving average of all past context vectors. $\{\bar{z}^m\}$ will be updated after sampling a new batch of z_t^m:

$$\bar{z}^m = \eta \text{sg}(\bar{z}^m) + (1-\eta)\mathbb{E}[z_t^m], \tag{7}$$

where sg$(\cdot)$ denotes stopping gradient, and η is a hyperparameter controlling the moving average horizon.

B.3 VARIATIONAL BOUND OF TEAMMATES CONTEXT APPROXIMATION

In order to make context vector $e_t^{m,i}$ generated by local trajectory encoder f_{ϕ_i} informatively consistent with global context z_t^m encoded by g_θ, we propose to maximize the mutual information between $e_t^{m,i}$ and z_t^m conditioned on the agent i's local trajectory $\tau_t^{m,i}$. We draw the idea from variational inference [Alemi et al., 2017] and derive a lower bound of this mutual information term.

Theorem 1. *Let $\mathcal{I}(e_t^{m,i}; z_t^m|\tau_t^{m,i})$ be the mutual information between the local context $e_t^{m,i}$ of agent i and global context z_t^m conditioned on agent i's local trajectory $\tau_t^{m,i}$. The lower bound is given by*

$$\mathbb{E}_{\mathcal{D}}[\log q_\xi(e_t^{m,i}|z_t^m, \tau_t^{m,i})] + \mathcal{H}(e_t^{m,i}|\tau_t^{m,i}). \tag{8}$$

Here m is the cluster id of the teammates cooperating with controlled agents to finish the task in this episode.

Proof. By a variational distribution $q_\xi(e_t^{m,i}|z_t^m, \tau_t^{m,i})$ parameterized by ξ, we have

$$\begin{aligned}
&\mathcal{I}(e_t^{m,i}; z_t^m|\tau_t^{m,i})\\
=&\mathbb{E}_{\mathcal{D}}\Big[\log \frac{p(e_t^{m,i}; z_t^m|\tau_t^{m,i})}{p(e_t^{m,i}|\tau_t^{m,i})p(z_t^m|\tau_t^{m,i})}\Big]\\
=&\mathbb{E}_{\mathcal{D}}\left[\log \frac{p(e_t^{m,i}|z_t^m; \tau_t^{m,i})}{p(e_t^{m,i}|\tau_t^{m,i})}\right]\\
=&\mathbb{E}_{\mathcal{D}}\left[\log \frac{q_\xi(e_t^{m,i}|z_t^m, \tau_t^{m,i})}{p(e_t^{m,i}|\tau_t^{m,i})}\right] +\\
&D_{\text{KL}}(p(e_t^{m,i}|z_t^m, \tau_t^{m,i})||q_\xi(e_t^{m,i}|z_t^m, \tau_t^{m,i}))\\
\geq&\mathbb{E}_{\mathcal{D}}\left[\log \frac{q_\xi(e_t^{m,i}|z_t^m, \tau_t^{m,i})}{p(e_t^{m,i}|\tau_t^{m,i})}\right]\\
=&\mathbb{E}_{\mathcal{D}}[\log q_\xi(e_t^{m,i}|z_t^m, \tau_t^{m,i})] + \mathcal{H}(e_t^{m,i}|\tau_t^{m,i}).
\end{aligned} \tag{9}$$

□

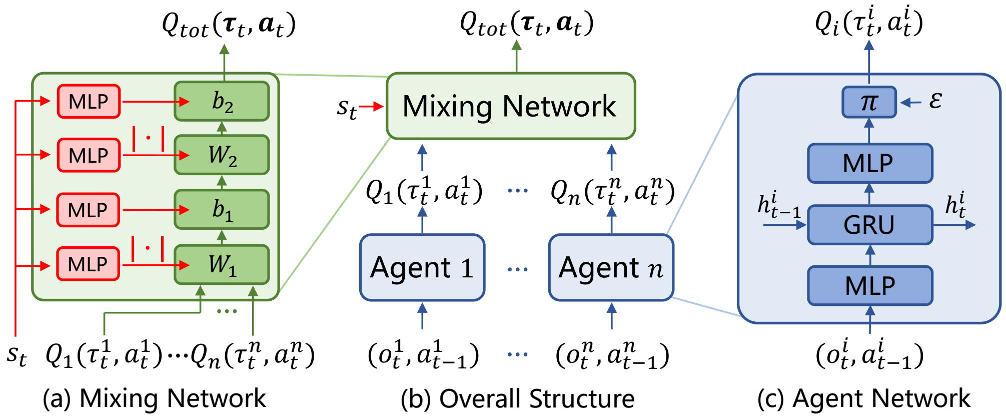


Figure 1: The overall structure of QMIX. (a) The detailed structure of the mixing network, whose weights and biases are generated from a hyper-net (red) which takes the global state as the input. (b) QMIX is composed of a mixing network and several agent networks. (c) The detailed structure of the individual agent network.

C DETAILS ABOUT BASELINES AND BENCHMARKS

C.1 BASELINES USED

QMIX [Rashid et al., 2018]: As we investigate the integrative abilities of Fastap in the manuscript, here we introduce the value-based method QMIX [Rashid et al., 2018] used in this paper. Our proposed framework Fastap follows the *Centralized Training with Decentralized Execution* (CTDE) paradigm used in value-based MARL methods, as well as the Individual-Global-Max (IGM) [Son et al., 2019] principle, which asserts the consistency between joint and local greedy action selections by the joint value function $Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a})$ and individual value functions $\left[Q_i(\tau^i, a^i)\right]_{i=1}^n$:

$$\begin{aligned} &\forall \boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}, \underset{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}}{\arg\max} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \\ &\left(\underset{a^1 \in \mathcal{A}}{\arg\max} Q_1\left(\tau^1, a^1\right), \ldots, \underset{a^n \in \mathcal{A}}{\arg\max} Q_n\left(\tau^n, a^n\right) \right). \end{aligned} \tag{10}$$

QMIX extends VDN by factorizing the global value function $Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})$ as a monotonic combination of the agents' local value functions $\left[Q_i(\tau^i, a^i)\right]_{i=1}^n$:

$$\forall i \in \mathcal{N}, \frac{\partial Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q_i\left(\tau^i, a^i\right)} > 0. \tag{11}$$

We mainly implement Fastap on QMIX for its proven performance in various papers, and its overall structure is shown in Fig. 1. QMIX uses a hyper-net conditioned on the global state to generate the weights and biases of the local Q-values and uses the absolute value operation to keep the weights positive to guarantee monotonicity.

PEARL [Rakelly et al., 2019]: This baseline comes from single-agent and meta-learning settings. It aims to represent the environments according to some hidden representations. Concretely, PEARL utilizes the transition data as context to infer the feature of the environment, which is modeled by a product of Gaussians. When it is applied to MARL tasks, the PEARL module is adopted and optimized for local context encoders of each individual controllable agent.

ESCP [Luo et al., 2022]: As a single-agent reinforcement learning algorithm that aims to recognize and adapt to new environments rapidly when encountering a sudden change in environments, the optimization objective Eqn. 4 is applied to optimize a context encoder. To cater to the framework and specific tasks in MARL, the history is not truncated, and each controllable agent is equipped with a local encoder.

LIAM [Papoudakis et al., 2021a]: A method equips each agent with an encoder-decoder structure to predict other agents' observations $\boldsymbol{o}_t^{-1}$ and actions $\boldsymbol{a}_t^{-1}$ at current timestep based on its own local observation history $\tau_t = \{o_{0:t}\}$. The encoder and decoder are optimized to minimize the mean square error of observations plus the cross-entropy error of actions. To fit in the MARL setting in our work, local context encoders of controllable agents will be asked to predict the teammates' observations and actions based on their local trajectories. The mean value of their loss is used to optimize the encoders.

ODITS [Gu et al., 2022]: Unlike the previous two methods that predict the actual behaviors of teammate agents, ODITS improves zero-shot coordination performance in an end-to-end fashion. Two variational encoders are adopted to improve the coordination capability. The global encoder takes in the global state trajectory as input and outputs a Gaussian distribution. A vector z is sampled and fed into hyper-network that maps the ad hoc agent's local utility Q_i into global utility Q_{tot} to approach the global discounted return. The local encoder has a similar structure and the sampled e is fed into the ad hoc agent's policy network. The encoders are updated by maximizing the return, together with the mutual information of the two context vectors conditioned on the local transition data in an end-to-end manner. As ODITS considers only a single ad hoc agent, we also equip each controllable agent with a local trajectory encoder and maximize the mean of mutual information loss to fit in our MARL's setting.

C.2 RELEVANT ENVIRONMENTS

Level-Based Foraging (LBF) [Papoudakis et al., 2021b]: LBF is a mixed cooperative-competitive partially observable grid-world game that requires highly coordinated agents to complete the task of collecting the foods. The agents and the foods are assigned with random levels and positions at the beginning of an episode. The action space of each agent consists of the movement in four directions, loading food next to it and a "no-op" action, but the foods are immobile during an entire episode. A group of agents can collect the food if the summation of their levels is no less than the level of the food and receive a normalized reward correlated to the level of the food. The main goal of the agents is to maximize the global return by cooperating with each other to collect the foods in a limited time.

To test the performance of different algorithms in this setting, we consider a scenario with four (at most) agents with different levels and three foods with the minimum levels $l \geq \sum_{i=1}^{3} sorted(levels)[i]$ in a 6×6 grid world. Agents have a limited vision with a range of 1 (3×3 grids around the agent), and the episode is under a limited horizon of 25. In our Open Dec-POMDP setting, two agents are controllable and will stay in the environment for the whole episode. The number of teammates might be 1 or 2, and the policy network will change as well. The rewards that the agents receive are the quotient of the level of the food they collect divided by the summation of all the food levels, as follows:

$$r^i = \frac{\text{Food\_with\_Level\_i}}{\sum_j \text{Food\_with\_Level\_j}}. \tag{12}$$

Predator-prey (PP) [Lowe et al., 2017]: This is a predator-prey environment. Good agents (preys) are faster and receive a negative reward for being hit by adversaries (predators) (-10 for each collision). Predators are slower and are rewarded for hitting good agents (+10 for each collision). Obstacles block the way. By default, there is 1 prey, 3 predators, and 2 obstacles. In our Open Dec-POMDP setting, two predators are controllable and will stay in the environment for the whole episode. The other predator is the uncontrollable teammate whose policy changes suddenly.

Cooperative navigation (CN) [Lowe et al., 2017]: In this task, four agents are trained to move to four landmarks while avoiding collisions with each other. All agents receive their velocity, position, and relative position to all other agents and landmarks. The action space of each agent contains five discrete movement actions. Agents are rewarded with the sum of negative minimum distances from each landmark to any agent, and an additional term is added to punish collisions among agents. In our Open Dec-POMDP setting, two agents are controllable and will stay in the environment for the whole episode. The number of teammates might be 1 or 2, and the policy network will change as well.

StarCraft II Micromanagement Benchmark (SMAC) [Samvelyan et al., 2019]: SMAC is a combat scenario of StarCraft II unit micromanagement tasks. We consider a partial observation setting, where an agent can only see a circular area around it with a radius equal to the sight range, which is set to 9. We train the ally units with reinforcement learning algorithms to beat enemy units controlled by the built-in AI. At the beginning of each episode, allies and enemies are generated at specific regions on the map. Every agent takes action from the discrete action space at each timestep, including the following actions: no-op, move [direction], attack [enemy id], and stop. Under the control of these actions, agents can move and attack in continuous maps. MARL agents will get a global reward equal to the total damage done to enemy units at each timestep. Killing each enemy unit and winning the combat (killing all the enemies) will bring additional bonuses of 10 and 200, respectively. Here we create a map named 10m_vs_11m, where 10 allies and 14 enemies are divided into

2 groups separately, and they are spawned at different points to gather together and enforce attacks on the same group of enemies to win this task. Specifically, we control 7 allies to cooperate with 3 other teammates to finish the task, where the number of teammates keeps unchangeable during an episode.

D THE ARCHITECTURE, INFRASTRUCTURE, AND HYPERPARAMETERS CHOICES OF FASTAP

Since Fastap is built on top of QMIX in the main experiments, we here present detailed descriptions of specific settings in this section, including network architecture, the overall flow, and the selected hyperparameters for different environments.

D.1 NETWORK ARCHITECTURE

In this section, we would give details about the following networks: (1) encoder E_{ω_1} and decoder D_{ω_2} in CRP process, (2) trajectory encoder g_θ, f_{ϕ_i}, and agent networks, and (3) variational distribution q_ξ and teammates modeling decoder h_{ψ_i}.

The 8-layer transformer encoder E_{ω_1} takes global trajectory $\tau = (s_0, \boldsymbol{a}_0, ..., s_T)$ as inputs and outputs 16-dimensional behavioral embeddings v. The RNN-based decoder D_{ω_2}, consisting of a GRU cell whose hidden dimension is 16, takes $\tau_t^X = (s_0, ..., s_t)$ and v as input and reconstructs the action $\boldsymbol{a}_t$.

For the global and local trajectory encoder g_θ and f_{ϕ_i}, we design it as a 2-layer MLP and GRU, and the hidden dimension is 64. Then a linear layer transforms the embeddings into mean values and standard deviations of a Gaussian distribution. The context vector will be sampled from the distribution. The global context z_t and state s_t will be concatenated and input into the hypernetwork. As for the local context e_t^i, it, together with local trajectory τ_t^i, will be input into the agent i's individual Q network, having a GRU cell with a dimension of 64 to encode historical information and two fully connected layers, to compute the local Q values $Q^i(\tau_t^i, e_t^i, \cdot)$. The local Q values will be fed into the mixing network to calculate TD loss finally.

To maximize the mutual information between local and global context vectors conditioned on the agent i's local trajectory, a variational distribution network q_ξ is used to approximate the conditional distribution. Concretely, q_ξ is a 3-layer MLP with a hidden dimension of 64, and it outputs a Gaussian distribution where the predicted local context vector will be sampled. The agent modeling decoder h_{ψ_i} is divided into two components including $h_{\psi_i}^o$ and $h_{\psi_i}^a$, where each one is a 3-layer MLP. Mean squared loss and maximum likelihood loss are calculated to optimize the objective, respectively.

D.2 THE OVERALL FLOW OF FASTAP

To illustrate the overall flow of Fastap, we first show the CRP-based infinite mixture procedure in Alg. 1. A teammate group can be generated via any MARL algorithm, and we store the small batch of trajectories into a replay buffer $\mathcal{D}_k$ (Line 2~3). The encoder and decoder are trained to force the learned representation to precisely capture the behavioral information and precisely estimate the predictive likelihood (Line 4). Afterward, the CRP prior and predictive likelihood are calculated to determine the assignment of the newly generated teammate group m^* (Line 5~7). Then, we update the existing cluster or instantiate a new cluster based on the assignment (Line 8~17).

The training process of Fastap is also shown in Alg. 2. During the trajectory sampling stage, we first sample a teammate group from the cluster and fix it in this episode. The teammate group pairs with the controllable agents and they make decisions together (Line 3~12). To train the agent policy networks and the context encoders, the moving average values of context vectors are updated and the optimization objectives are calculated (Line 14~22). Besides, we present the testing process in Alg. 3, where teammates might change suddenly. A sudden change distribution $\mathcal{U}$ controls the waiting time that determines the changing frequency (Line 5~12).

Our implementation of Fastap is based on the EPymarl[1] [Papoudakis et al., 2021b] codebase with StarCraft 2.4.6.2.69223 and uses its default hyper-parameter settings (e.g., $\gamma = 0.99$). The selection of other additional hyperparameters for different environments is listed in Tab.1.

[1]`https://github.com/oxwhirl/epymarl`

Algorithm 1 Fastap: CRP-based infinite mixture procedure

Input: concentration param α, num of teammate groups generated in one iteration L, number of teammate groups generated so far K, number of clusters instantiated so far M, encoder E_{ω_1}, decoder D_{ω_2}.

```
1:  for $k = K+1, .., K+L$ do
2:      Generate the $k^{\text{th}}$ teammate group.
3:      Sample small batch of trajectories $\tau_k$ of the $k^{\text{th}}$ teammate group and store them into $\mathcal{D}_k$.
4:      Train $E_{\omega_1}$ and $D_{\omega_2}$ according to $\mathcal{L}_{\text{model}}$ in Eqn. 4.
5:      Calculate the CRP prior $P(v_k^{(m)}), m = 1, 2, ..., M+1$ according to Eqn. 2.
6:      Calculate the predictive likelihood $P(\tau_k^Y | \tau_k^X; v_k^{(m)}), m = 1, 2, ..., M+1$ according to Eqn. 3.
7:      $m^* = \arg\max_m P(v_k^{(m)}) P(\tau_k^Y | \tau_k^X; v_k^{(m)})$.
8:      if $m^* \leq M$ then
9:          Assign the $k^{\text{th}}$ teammate group to the $m^*$ cluster.
10:         Update the cluster center $\bar{v}^{m^*} = \frac{n^{(m^*)} \bar{v}^{m^*} + v_k}{n^{(m^*)} + 1}$.
11:         Update the counter of the cluster $m$: $n^{(m^*)} = n^{(m^*)} + 1$.
12:     else
13:         Initialize the $M + 1^{\text{th}}$ cluster with the $k^{\text{th}}$ teammate group.
14:         Initialize the cluster center $\bar{v}^{M+1} = v_k$.
15:         Initialize the counter of the cluster $M+1$: $n^{(M+1)} = 1$.
16:         Update $M = M + 1$.
17:     end if
18: end for
19: Update $K = K + L$.
```

Algorithm 2 Fastap: training process

Input: controllable agent policy networks $\{\pi^i\}_{i=1}^n$, global trajectory encoder g_θ, local trajectory encoders $\{f_{\phi_i}\}_{i=1}^n$, teammate group clusters $\mathcal{C}$, number of clusters instantiated so far M, episode length T, number of sampled episodes $sample\_num$, environment env.

```
1:  Initialize moving average $\bar{z}^m = \mathbf{0}, m = 1, ..., M$.
2:  Initialize moving average $\bar{e}^{m,i} = \mathbf{0}, m = 1, ..., M; i = 1, .., n$.
3:  for $l = 1, ..., sample\_num$ do
4:      sample teammate group from $\mathcal{C}$ belonging to the $m^{\text{th}}$ cluser.
5:      $s_0^m = env.start()$.
6:      for $t = 0, ..., T$ do
7:          $e_t^{m,i} = f_{\phi_i}(\tau_t^{m,i}), \quad i = 1, ..., n$.
8:          $a_t^{m,i} = \pi^i(\tau_t^{m,i}, e_t^{m,i}), \quad i = 1, ..., n$.
9:          $\boldsymbol{a}_t^m = (a_t^{m,i})_{i=1}^n$. // controllable agents decision-making
10:         $\bar{\boldsymbol{a}}_t^m = \bar{\boldsymbol{\pi}}^m(\bar{\boldsymbol{\tau}}_t^m)$. // uncontrollable teammates decision-making
11:         $s_{t+1}^m, r_t^m = env.step(\langle \boldsymbol{a}_t^m, \bar{\boldsymbol{a}}_t^m \rangle)$.
12:     end for
13:     Add trajectory to the replay buffer $\mathcal{D}$.
14:     for $m = 1, .., M$ do
15:         Sample $bs$ trajectories from $\mathcal{D}$.
16:         Calculate estimated Q-values and context vectors $z_t^m = g_\theta(\tau_t^m), e_t^{m,i} = f_{\phi_i}(\tau_t^{m,i}), \quad t = 0, ..., T$.
17:         Update $\bar{z}^m = \eta \text{sg}(\bar{z}^m) + (1 - \eta) \text{mean}(z_t^m)$.
18:         Update $\bar{e}^{m,i} = \eta \text{sg}(\bar{e}^{m,i}) + (1 - \eta) \text{mean}(e_t^{m,i})$.
19:         Optimize agent Q networks according to $\mathcal{L}_{\text{TD}}$.
20:     end for
21:     Optimize $g_\theta$ according to $\mathcal{L}_{\text{ADAP}}$ in Eqn. 6.
22:     Optimize $\{f_{\phi_i}\}_{i=1}^n$ according to $\mathcal{L}_{\text{DEC}}$ in Eqn. 11.
23: end for
```

Algorithm 3 Fastap: testing process

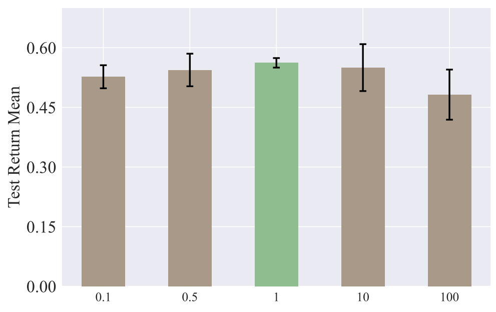
```
Input: controllable agent policy networks $\{\pi^i\}_{i=1}^n$, local trajectory encoders $\{f_{\phi_i}\}_{i=1}^n$, episode length $T$, number of test episodes $test\_num$, environment $env$, sudden change distribution $\mathcal{U}$, teammates set $\bar{\mathcal{N}}$.
1: for $l = 1, ..., test\_num$ do
2:   Sample teammate policy $\bar{\boldsymbol{\pi}}$ from $\bar{\mathcal{N}}$.
3:   $s_0 = env.start()$.
4:   for $t = 0, ..., T$ do
5:     if $t = 0$ then
6:       Sample waiting time $u_0 \sim \mathcal{U}$.
7:     else
8:       Update waiting time $u_t = u_{t-1} - 1$.
9:       if $u_t \leq 0$ then
10:        Re-sample $u_t \sim \mathcal{U}$.
11:        Re-sample teammate policy $\bar{\boldsymbol{\pi}}$ from $\bar{\mathcal{N}}$.
12:      end if
13:    end if
14:    $e_t^i = f_{\phi_i}(\tau_t^i), \quad i = 1, ..., n$.
15:    $a_t^i = \pi^i(\tau_t^i, e_t^i), \quad i = 1, ..., n$.
16:    $\boldsymbol{a}_t = (a_t^i)_{i=1}^n$. // controllable agents decision-making
17:    $\bar{\boldsymbol{a}}_t = \bar{\boldsymbol{\pi}}(\bar{\boldsymbol{\tau}}_t)$. // uncontrollable teammates decision-making
18:    $s_{t+1}, r_t, done = env.step(\langle \boldsymbol{a}_t, \bar{\boldsymbol{a}}_t \rangle)$.
19:  end for
20: end for
```

Environment / Hyperparameter	Level-Based Foraging	Predator-prey	Cooperative navigation	10m_vs_14m
concentration hyperparameter α	0.5	2.5	2.5	0.5
number of teammate groups generated in one iteration L	4	1	1	2
radius hyperparameter κ	80	80	80	80
moving average hyperparameter η	0.01	0.01	0.01	0.01
α_{GCE}	1	0.4	0.4	10
α_{LCE}	1	0.4	0.4	10
α_{MI}	0.001	0.001	0.001	0.001
α_{REC}	0.1	0.2	0.2	0.2
dimension of local context vector e	4	16	4	8
dimension of global context vector z	6	20	6	16

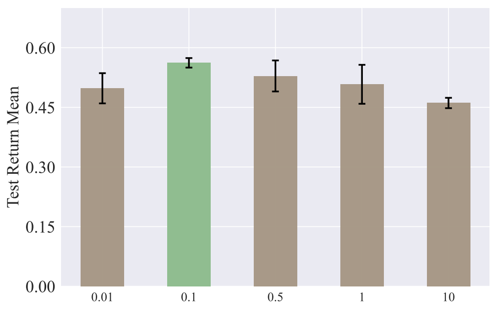
Table 1: Hyperparameters in the experiments.

E MORE SENSITIVE STUDIES

Here we further conduct more experiments on benchmark LBF to investigate how another two hyperparameters $\alpha_{\text{LCE}}, \alpha_{\text{REC}}$ influence the coordination ability. The results can be seen in Fig. 2, we can find that $\alpha_{\text{LCE}} = 1, \alpha_{\text{REC}} = 0.1$ are the corresponding best choices in a similar way as in the manuscript.



(a) Sensitivity of α_{LCE}



(b) Sensitivity of α_{REC}

Figure 2: More Sensitivity Studies on LBF.

References

Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *ICLR*, 2017.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *preprint arXiv:2301.08028*, 2023.

David M. Blei and Peter I. Frazier. Distance dependent chinese restaurant processes. In *ICML*, pages 87–94, 2010.

Xiaoyu Chen, Xiangming Zhu, Yufeng Zheng, Pushi Zhang, Li Zhao, Wenxue Cheng, Peng CHENG, Yongqiang Xiong, Tao Qin, Jianyu Chen, and Tie-Yan Liu. An adaptive deep RL method for non-stationary environments with piecewise stable context. In *NeurIPS*, 2022.

Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *NeurIPS*, 2020.

Apan Dastider and Mingjie Lin. Non-parametric stochastic policy gradient with strategic retreat for non-stationary environment. In *CASE*, pages 1377–1384. IEEE, 2022.

Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593, 2018.

Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. Factored adaptation for non-stationary reinforcement learning. In *NeurIPS*, 2022.

Rihab Gorsane, Omayma Mahjoub, Ruan John de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative MARL. In *NeurIPS*, 2022.

St John Grimbly, Jonathan Shock, and Arnu Pretorius. Causal multi-agent reinforcement learning: Review and open problems. *preprint arXiv:2111.06721*, 2021.

Pengjie Gu, Mengchen Zhao, Jianye Hao, and Bo An. Online ad hoc teamwork under partial observability. In *ICLR*, 2022.

Jun Guo, Yonghong Chen, Yihang Hao, Zixin Yin, Yin Yu, and Simin Li. Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning. *preprint arXiv:2204.07932*, 2022.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. *preprint arXiv:1502.02259*, 2015.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob N. Foerster. "other-play" for zero-shot coordination. In *ICML*, pages 4399–4410, 2020a.

Yizheng Hu, Kun Shao, Dong Li, HAO Jianye, Wulong Liu, Yaodong Yang, Jun Wang, and Zhanxing Zhu. Robust multi-agent reinforcement learning driven by correlated equilibrium. 2020b.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

Dong-Ki Kim, Miao Liu, Matthew Riemer, Chuangchuang Sun, Marwa Abdulhai, Golnaz Habibi, Sebastian Lopez-Cot, Gerald Tesauro, and Jonathan P. How. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In *ICML*, pages 5541–5550, 2021.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: rapid motor adaptation for legged robots. In Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh, editors, *Robotics: Science and Systems XVII*, 2021.

Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. On the robustness of cooperative multi-agent reinforcement learning. In *SPW*, pages 62–68, 2020.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NIPS*, pages 6379–6390, 2017.

Fan-Ming Luo, Shengyi Jiang, Yang Yu, Zongzhang Zhang, and Yi-Feng Zhang. Adapt to environment sudden changes by learning a context sensitive policy. In *AAAI*, pages 7637–7646, 2022.

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. In *AAMAS*, pages 844–852, 2021.

Reuth Mirsky, Ignacio Carlucho, Arrasy Rahman, Elliot Fosong, William Macke, Mohan Sridharan, Peter Stone, and Stefano V Albrecht. A survey of ad hoc teamwork: Definitions, methods, and open problems. *preprint arXiv:2202.10450*, 2022.

Hadi Nekoei, Akilesh Badrinaaraayanan, Aaron C. Courville, and Sarath Chandar. Continuous coordination as a realistic scenario for lifelong learning. In *ICML*, pages 8016–8024, 2021.

Afshin Oroojlooy and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, pages 1–46, 2022.

Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54:1 – 25, 2020.

Sindhu Padakandla, Prabuchandran K. J., and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, pages 1–17, 2019.

Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V Albrecht. Dealing with non-stationarity in multi-agent deep reinforcement learning. *preprint arXiv:1906.04737*, 2019.

Georgios Papoudakis, Filippos Christianos, and Stefano Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *NeurIPS*, pages 19210–19222, 2021a.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *NeurIPS*, 2021b.

Rongjun Qin, Feng Chen, Tonghan Wang, Lei Yuan, Xiaoran Wu, Zongzhang Zhang, Chongjie Zhang, and Yang Yu. Multi-agent policy transfer via task relationship modeling. *preprint arXiv:2203.04482*, 2022.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, pages 5331–5340, 2019.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, pages 4295–4304, 2018.

Hang Ren, Aivar Sootla, Taher Jafferjee, Junxiao Shen, Jun Wang, and Haitham Bou-Ammar. Reinforcement learning in presence of discrete markovian context evolution. In *ICLR*, 2022.

Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The Starcraft multi-agent challenge. In *AAMAS*, pages 2186–2188, 2019.

Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, pages 5887–5896, 2019.

Yanchao Sun, Ruijie Zheng, Parisa Hassanzadeh, Yongyuan Liang, Soheil Feizi, Sumitra Ganesh, and Furong Huang. Certifiably robust policy learning against adversarial multi-agent communication. In *ICLR*, 2023.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, pages 2085–2087, 2018.

Nelson Vithayathil Varghese and Qusay H. Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 2020.

Haonan Wang, Ning Liu, Yiyun Zhang, Dawei Feng, Feng Huang, Dongsheng Li, and Yiming Zhang. Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 21:1726 – 1744, 2020.

Jianhong Wang, Wangkun Xu, Yunjie Gu, Wenbin Song, and Tim C. Green. Multi-agent reinforcement learning for active voltage control on power distribution networks. In *NeurIPS*, pages 3271–3284, 2021.

Xihuai Wang, Zhicheng Zhang, and Weinan Zhang. Model-based multi-agent reinforcement learning: Recent progress and prospects. *preprint arXiv:2203.10603*, 2022.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *NeurIPS*, 2022.

Ke Xue, Jiacheng Xu, Lei Yuan, Miqing Li, Chao Qian, Zongzhang Zhang, and Yang Yu. Multi-agent dynamic algorithm configuration. In *NeurIPS*, 2022a.

Wanqi Xue, Wei Qiu, Bo An, Zinovi Rabinovich, Svetlana Obraztsova, and Chai Kiat Yeo. Mis-spoke or mis-lead: Achieving robustness in multi-agent communicative reinforcement learning. In *AAMAS*, pages 1418–1426, 2022b.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. In *NeurIPS*, pages 10299–10312, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *NeurIPS*, 2022.

Kaiqing Zhang, Tao Sun, Yunzhe Tao, Sahika Genc, Sunil Mallya, and Tamer Basar. Robust multi-agent reinforcement learning with model uncertainty. In *NeurIPS*, pages 10571–10583, 2020.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent reinforcement learning with communication. *preprint arXiv:2203.08975*, 2022.