Communicating Unexpectedness for Out-of-Distribution Multi-Agent Reinforcement Learning

Min Whoo Lee, Kibeom Kim, Soo Wung Shin, Minsu Lee, Byoung-Tak Zhang

Biointelligence Laboratory, Seoul National University
1, Gwanak-ro, Gwanak-gu
Seoul 08826 Republic of Korea
{mwlee,kbkim}@bi.snu.ac.kr, ps4southwest@gmail.com, {mslee,btzhang}@bi.snu.ac.kr

Abstract

Applying multi-agent reinforcement learning methods to re-alistic settings is challenging as it may require the agents to quickly adapt to unexpected situations that are rarely or never encountered in training. Recent methods for general-ization to such out-of-distribution settings are limited to more specific, restricted instances of distribution shifts. To tackle adaptation to distribution shifts, we propose Unexpected En-coding Scheme, a novel decentralized multi-agent reinforce-ment learning algorithm where agents communicate "unex-pectedness," the aspects of the environment that are surpris-ing. In addition to a message yielded by the original reward-driven communication, each agent predicts the next observa-tion based on previous experience, measures the discrepancy between the prediction and the actually encountered observa-tion, and encodes this discrepancy as a message. Experiments on multi-robot warehouse environment support that our pro-posed method adapts robustly to dynamically changing train-ing environments as well as out-of-distribution environment.

1 Introduction

Recent development of multi-agent reinforcement learn-ing (MARL) (Zhang, Yang, and Başar 2021) has shown promises in domains such as games (Schrittwieser et al. 2020; Vinyals et al. 2019; Berner et al. 2019), unmanned aerial vehicles (Zhou et al. 2021), and smart grids (Zhang et al. 2022). Nonetheless, extension of current MARL meth-ods to reality still calls for multiple challenges. First, the ac-tual environments are very likely different from the environ-ments on which the agents are trained, as illustrated in Fig-ure 1. In reality, agents will have to operate robustly even in those situations that they have never encountered within the distribution of the training environments, in order to avoid inefficient or even unsafe behaviours. Meanwhile, given that one agent manages to adapt to some unexpected situation, it is important that other agents learn from this experience even without making these same mistakes. Second, many realistic settings are partially observable and decentralized. An individual agent is often only able to observe the vicin-ity, and the learning difficulty is exacerbated by the presence of other agents, who may alter their behaviours and environ-mental state dynamically.
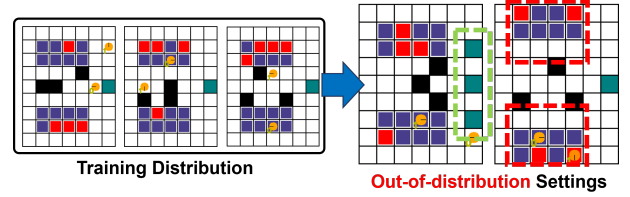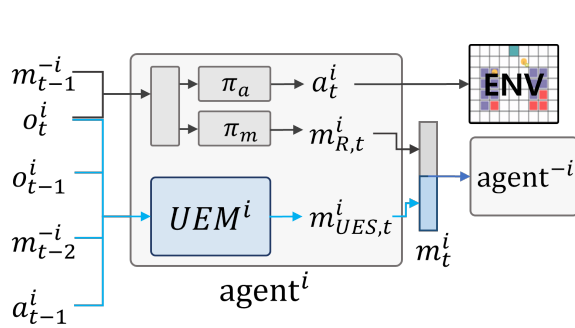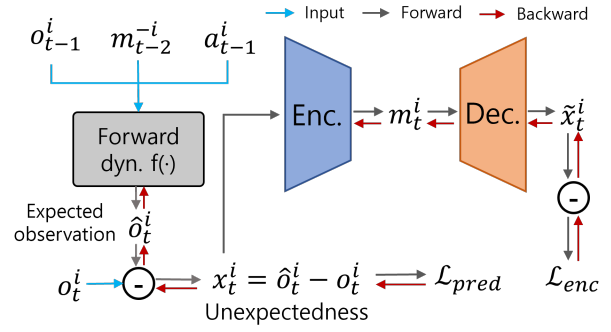
Figure 1: Conceptual diagram of the problem description. The green dotted box indicates the Goal-Shift setting, and the red dotted box indicates the Shelf-Shift setting.

These challenges call for novel use of communication in MARL. In other words, while adopting RL to enable online adaptation of agents during unexpected, out-of-distribution situations in deployment, we also seek an effective inter-agent communication. Communication is a unique feature of multi-agent systems that allows agents to share infor-mation that may aid cooperative or distributed task-solving (Shoham and Leyton-Brown 2008). An appropriate use of agent-to-agent communication may resolve the aforemen-tioned issues by sharing (1) environmental state informa-tion that is invisible to other agents, and (2) experience of unexpected situations that are not encountered during train-ing. Thus, our research objective is to develop an MARL architecture that effectively uses communication to let the agents adapt well in out-of-distribution downstream tasks. Since agents will not have access to other agents' observa-tions or actions in deployment, our method is designed as decentralized training, allowing the agents to adapt.

Recently, research has been extensively conducted on end-to-end learning of communication protocols in multi-agent systems (Zhu, Dastani, and Wang 2022; Brandizzi 2023). While vast amount of work has been done on in-vestigating non-stationarity issue of multi-agent system that arises from evolving agent behaviors (Papoudakis et al. 2019; Hernandez-Leal et al. 2017), studies on generaliz-ing multi-agent system to out-of-distribution environments are relatively scarce. Particularly, several studies (Liu et al. 2021; Shao et al. 2022) demonstrate the capability of gen-eralizing to unseen number of agents and environmental ob-jects. Nonetheless, these works assume that all environmen-tal factors can be identified by objects and constrain the ob-

(a) Overall architecture   (b) Architecture of Unexpectedness Encoding Module (UEM)

Figure 2: Overview of the Unexpectedness Encoding Scheme with Reward (UES+R).

servation to a strict format. It seems reasonable, on the other hand, that sudden environmental changes may occur in a more general manner that may not be restricted to identification of objects. A recent work (Abu et al. 2021) that is most closely related to ours introduces the notion of "confusion" to communicate unexpectedness of the environment. However, the definition of confusion relies on immediate one-step reward, which is susceptible to noises in rewards and lacks consideration of long-term value.

We propose Unexpectedness Encoding Scheme with Reward (**UES+R**), a novel agent-to-agent communication scheme in decentralized MARL setting that enhances robustness to distribution shifts. On top of the base communication scheme that is trained to maximize reward, we additionally devise Unexpectedness Encoding Scheme. The key idea is to predict the next observation via forward dynamics, and measure the difference between the prediction and the actual next observation. Such discrepancy vector is formulated as the *unexpectedness*, and an autoencoder is used to encode this vector as *unexpectedness encoding*. This is fused with reward-driven message, in order to ensure task-relevant information to be shared among agents. Experiments on the multi-agent warehouse environments support that UES+R not only boosts training performance in dynamically changing settings, but also promotes robust adaptation in environments with distribution shift. Notably, performance of this decentralized training method is on par with a centralized training method that has access to all agents' observations.

2 Unexpectedness Encoding for Communicating in Out-of-Distribution Environments

2.1 Overview

We consider Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek, Spaan, and Vlassis 2008), a variant of Markov game (Littman 1994) that accounts for partial observability. At time step t, agent i receives local observation o_t^i from observation space $\mathcal{O}^i$ and messages from other agents m_{t-1}^{-i}, and chooses an action a_t^i from action space $\mathcal{A}^i$. In such partially observable setting, the local observations o_t^i do not fully reflect the envi-

ronment's true state $s_t \in \mathcal{S}$. This necessitates an effective usage of messages to share individual agents' knowledge of surroundings and coordinate their behaviors. The detailed preliminaries including Markov game and Dec-POMDP are covered in Appendix A.1.

On top of using message to communicate task-relevant information among the agents, we also intend to use message as a feature that expresses how much and in what sense the environment differs from anticipation. Consequently, if an out-of-distribution observation is encountered, the agent may encode this as message m_t^i and communicate to the other agents, thus informing them about the surprising change in the environment. In such case, the entire multi-agent system may be able to utilize the received message to learn about and prepare against such distribution shift situations. To implement the aforementioned purpose, we propose a novel communication scheme called Unexpectedness Encoding Scheme with Reward (UES+R). The architecture and data flow for each agent $i \in \mathcal{N}$ are outlined in Figure 2.

2.2 Unexpectedness Encoding Scheme

First of all, we describe Unexpectedness Encoding Scheme (UES), the main addition of our proposed architecture. This is illustrated in Figure 2b. To formulate the difference between anticipated and actual situations, we utilize a forward dynamics module $f(\cdot)$ that carries out the anticipation. UES introduces an Unexpectedness Encoding Module (UEM) that uses the information from time step $(t-1)$ in order to predict the observation at time t in hindsight. To be specific, at time step t, UEM receives the agent i's observation o_{t-1}^i, action a_{t-1}^i, and other agents' messages m_{t-2}^{-i} at previous time steps. Based on these inputs, the module performs forward dynamics $f(\cdot)$ and calculates the agent's prediction of the next observation o_t^i in hindsight, denoted as $\hat{o}_t^i := f(o_{t-1}^i, m_{t-2}^{-i}, a_{t-1}^i)$. We remind that the messages chosen at time step $(t-2)$ are received by the agent at time step $(t-1)$, hence the usage of m_{t-2}^{-i}. The discrepancy between the prediction $\hat{o}_t^i$ and the actual observation o_t^i is referred to as *unexpectedness*[1], denoted as $x_t^i = \hat{o}_t^i - o_t^i$.

---

[1]To be precise, as described in Appendix A.2, linear projection is used on the observations, rather than directly calculating the dif-

Then, in order to communicate the unexpected information relevant to distribution shift, the unexpectedness vector x_t^i is encoded to yield an *unexpectedness encoding*. This is the message output $m_{UES,t}^i$ of the UEM. The encoding is executed by an autoencoder, used for both dimension reduction and learning to encode the unlabeled data. In summary, the encoding process is represented by Equation 1.

$$m_{UES,t}^i = \text{Enc}\left(f(o_{t-1}^i, m_{t-2}^{-i}, a_{t-1}^i) - o_t^i\right) \quad (1)$$

The forward dynamics model $f(\cdot)$ is trained by backpropagating the prediction loss $\mathcal{L}_{pred}$ in Equation 2, which is the l^2-norm of the unexpectedness x_t^i. The autoencoder is trained with the loss $\mathcal{L}_{enc}$ in Equation 3, which is backpropagated only up to the encoder. $\tilde{x}_t^i$ denotes the reconstruction output of the autoencoder.

$$\mathcal{L}_{pred} := \left\|\hat{o}_t^i - o_t^i\right\|_2 = \left\|f(o_{t-1}^i, m_{t-2}^{-i}, a_{t-1}^i) - o_t^i\right\|_2 \quad (2)$$

$$\mathcal{L}_{enc} := \left\|\tilde{x}_t^i - x_t^i\right\|_2 = \left\|\text{Dec}(\text{Enc}(x_t^i)) - x_t^i\right\|_2 \quad (3)$$

2.3 Incorporating Extrinsic Reward

Intuitively, the message $m_{UES,t}^i$ yielded by UES can be deemed by the receiving agents as information that is useful for promptly learning about the environmental changes. Nonetheless, since UES is driven purely by observation prediction error and not by environmental rewards, the unexpectedness encoding may not focus on including information that is indeed relevant to the task. For the agents to conduct the task, communication guided by reward is needed.

For this, we consider a simple setting for reward-driven communication[2]. A message $m_{R,t}^i$ is assumed to be a binary bit vector $[u_1, u_2, \cdots, u_K]$ of length K, such that $u_k \in \{0, 1\}$ for $i \in \{1, 2, \cdots, K\}$. Just as how a reinforcement learning agent is trained to choose its action via an objective that reflects the environment reward, the agent is trained to choose each bit u_k of the message via the same RL objective. In other words, each message bit is treated as a separate action channel and reinforced according to reward likewise.

The objective is determined by the specific algorithm that is used to train the agents. For instance, we conduct our experiments using Advantage Actor-Critic (A2C) (Mnih et al. 2016) whose loss gradients are outlined below for agent i:

$$\nabla_\theta \mathcal{L}(\theta) = \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t, m_{R,t}^i | o_t; \theta)(R_t - V(o_t; \phi)),$$
$$(4)$$

$$\nabla_\phi \mathcal{L}_v(\phi) = \sum_{t=1}^{T} \nabla_\phi (R_t - V(o_t; \phi))^2, \quad (5)$$

where θ is actor parameters, ϕ is value parameters, V is an individual agent-wise value function, and $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t$ is sum of rewards. For simplicity, we omitted the the superscript i, which indicates the agent index, from $o_t^i, a_t^i, \theta^i, \phi^i$, and π^i.

---

ference between raw observations.

[2]Such reward-driven communication is a common setting, an example of which is in (Jaques et al. 2019) for actor-critic methods.
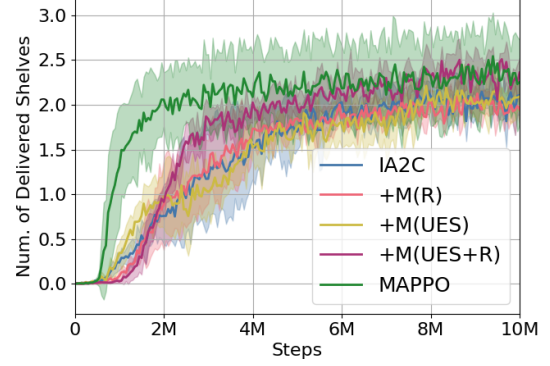

Figure 3: Learning curves on training distribution. Mean and standard deviation across 5 runs are plotted. Note that **M(UES+R)** is our main method, and MAPPO is intended to indicate the upper bound of performance.

To combine the benefit of UES in adapting to environmental changes and the usefulness of reward-driven communication in fulfilling the task, we fuse the messages from both schemes by concatenating them. We refer to this combined scheme as **UES+R**, highlighting this as our main proposed method.

3 Experiments

3.1 Benchmark Environment

We adopt Multi-Robot Warehouse (RWARE) environment (Papoudakis et al. 2020) as our benchmark for our algorithm. This environment, illustrated by snapshots in Figure 1, simulates grid-based robotic warehouses in reality, composed of two agents (orange circles) and many shelves containing items (blue/red squares). Briefly described, the agents must deliver the requested shelves (red squares) to the goal location (teal tile), and subsequently return the shelf back to the original place. To make the problem dynamically changing, three impassable obstacles (black tiles) are randomly positioned every 1K steps. One challenging factor of this environment is that a long sequence of correct actions must be conducted to receive a reward, which means that reward is highly sparse. Additionally, each agent can only observe adjacent tiles, making RWARE a Dec-POMDP. The detailed environment setting is described in Appendix A.3.

3.2 Out-of-Distribution Settings for Few-Shot Transfer Learning

The environment described in Section 3.1 outlines the *training distribution*. Our objective is to investigate the robustness of our method and the baselines to out-of-distribution environments that are not encountered during training. For this, the few-shot transfer learning capability of the methods are measured. In detail, the agents are initially trained on the training distribution for 10M steps. Subsequently, the trained agents are placed in previously unseen distributions of environments, and fine-tuning is conducted for 10 batches of

Table 1: Performances of baselines and the proposed methods. Mean and standard deviation of number of delivered shelves across 5 runs are recorded. MAPPO is a centralized training method, intended to show an upper bound performance.

Algorithm	Training	Goal-Shift	Shelf-Shift
IA2C	2.16±0.12	2.04±0.13	1.34±0.04
+M(R)	2.12±0.30	2.04±0.14	1.33±0.12
+M(UES)	2.21±0.20	2.04±0.25	1.25±0.09
+M(UES+R)	**2.51±0.07**	**2.29±0.07**	**1.45±0.07**
MAPPO (upper bound)	2.54±0.49	2.45±0.50	1.31±0.22

episodes. The performance is recorded as the cumulative average number of successfully delivered shelves per episode.

We emphasize that the unseen distributions are designed such that the environment varies in a manner that has never been encountered during training. We devise two such distributions as shown in Figure 1: (1) **Goal-Shift**, with more goal tiles than in training, and (2) **Shelf-Shift**, with shelves' positions shifted towards the walls. The former can be considered a beneficial distribution shift that imitates a realistic situation where new outlets are constructed for better throughput. Agents that are robust to such distribution shifts of the environment will quickly adapt to and exploit the newly added goal tiles. In contrast, Shelf-Shift increases the difficulty of the task, since the shelves adjacent to the wall are blocked by the surrounding shelves and are highly difficult to carry out.

3.3 Baselines

The experiments were conducted with the following algorithms: (1) Independent Advantage Actor-Critic (**IA2C**), where A2C (Mnih et al. 2016) was used to train agents individually without any message-sharing; (2) **IA2C+M(R)**, an ablative baseline where messages were trained via environmental reward r_{t+1}^i; (3) **IA2C+M(UES)**, another ablative baseline where messages are constructed only via UES; (4) **IA2C+M(UES+R)**, the proposed main method that fuses messages from the UES and reward; and (5) Multi-Agent Proximal Policy Optimization (**MAPPO**) (Yu et al. 2021), a multi-agent version of Proximal Policy Optimization algorithm (Schulman et al. 2017) that has shown one of the highest performances in this task (Papoudakis et al. 2020). It should be noted that all methods conduct decentralized training except for MAPPO, which is a centralized training method that has access to all agents' observations during training. Rather, MAPPO can be considered as an upper bound of the performance for decentralized training methods. We also clarify that the length of combined messages from **IA2C+M(UES+R)** is made to match the length of message from IA2C+M(R) and that from IA2C+M(UES) for fair comparison. Additional details of these baselines are explained in Appendix A.2.

3.4 Results

The training of all five methods were conducted on the training distribution with the aforementioned methods for 10M steps. Five repeated trials were conducted with different random seeds, and the learning curves and the highest performances are recorded in Figure 3 and Table 1, respectively. It is fascinating that in the training distribution, the communication methods IA2C+M(R) and IA2C+M(UES) do not significantly surpass IA2C that does not communicate, which is evident from comparison of the learning curves. This indicates that each of the two schemes alone does not lead to messages that beneficially contributes to solving the given task. However, the complementarity of the two message schemes is shown, as **IA2C+M(UES+R)** converges to a superior performance that eventually attains the performance of MAPPO, the centralized-training upper bound method.

Furthermore, we evaluated the robustness of the trained agents to distribution shifts as outlined in Section 3.2. The post-adaptation performances of all methods were recorded in Table 1. The results showed a trend similar to one in the training distribution. In other words, merely using reward or UES individually led to performances on par with or even worse than IA2C without messages, but a simultaneous use of both schemes led to noticeable improvement in robustness. It is noteworthy that in Shelf-Shift experiment, MAPPO, which we expected to be an upper bound, performed similarly to IA2C, a decentralized method. In comparison, our proposed **IA2C+M(UES+R)** achieves the highest performance. This indicates that simply sharing all agents' observations and conducting centralized training may not remedy the overfitting to the training distribution. Rather, the key solution may be the communication scheme that discreetly selects the messages that specifically inform how the environment has changed – addressed by UES – and what aspect of this new environment is relevant to task-solving – addressed by reward-driven message. We believe the result supports that our method **IA2C+M(UES+R)** achieves both.

4 Conclusion

We proposed UES+R, a novel MARL scheme that communicates unexpectedness to adapt to environmental distribution shifts. This scheme measures the difference between predicted observation and actual observation, and encodes this discrepancy as the message that is received by other agents. This message is combined with message that is trained to maximize reward. Experiments on multi-robot warehouse environment indicate that our proposed method leads to robust adaptation to dynamic, out-of-distribution environment.

Nonetheless, several limitations lie within this study. For instance, we assume the setting that messages are broadcasted to all agents, which is infeasible when number of agents are high. A method to compress or encode these into smaller representations may be needed, such as an attentional method in (Jiang and Lu 2018). Also, interpreting the message contents in relation to actual environmental shift is an important topic for future work.

Acknowledgments

References

Abu, O.; Gerstgrasser, M.; Rosenschein, J.; and Keren, S. 2021. Promoting Resilience in Multi-Agent Reinforcement Learning via Confusion-Based Communication. *arXiv preprint arXiv:2111.06614*.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Brandizzi, N. 2023. Towards More Human-like AI Communication: A Review of Emergent Communication Research. *arXiv preprint arXiv:2308.02541*.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Hernandez-Leal, P.; Kaisers, M.; Baarslag, T.; and de Cote, E. M. 2017. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*.

Jaques, N.; Lazaridou, A.; Hughes, E.; Gulcehre, C.; Ortega, P.; Strouse, D.; Leibo, J. Z.; and De Freitas, N. 2019. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, 3040–3049. PMLR.

Jiang, J.; and Lu, Z. 2018. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, 157–163. Elsevier.

Liu, B.; Liu, Q.; Stone, P.; Garg, A.; Zhu, Y.; and Anandkumar, A. 2021. Coach-player multi-agent reinforcement learning for dynamic team composition. In *International Conference on Machine Learning*, 6860–6870. PMLR.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353.

Papoudakis, G.; Christianos, F.; Rahman, A.; and Albrecht, S. V. 2019. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*.

Papoudakis, G.; Christianos, F.; Schäfer, L.; and Albrecht, S. V. 2020. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shao, J.; Lou, Z.; Zhang, H.; Jiang, Y.; He, S.; and Ji, X. 2022. Self-Organized Group for Cooperative Multi-agent Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35: 5711–5723.

Shoham, Y.; and Leyton-Brown, K. 2008. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; and Wu, Y. 2021. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. *arXiv preprint arXiv:2103.01955*.

Zhang, K.; Yang, Z.; and Başar, T. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321–384.

Zhang, Y.; Yang, Q.; An, D.; Li, D.; and Wu, Z. 2022. Multistep multiagent reinforcement learning for optimal energy schedule strategy of charging stations in smart grid. *IEEE Transactions on Cybernetics*.

Zhou, W.; Liu, Z.; Li, J.; Xu, X.; and Shen, L. 2021. Multitarget tracking for unmanned aerial vehicle swarms using deep reinforcement learning. *Neurocomputing*, 466: 285–297.

Zhu, C.; Dastani, M.; and Wang, S. 2022. A survey of multiagent reinforcement learning with communication. *arXiv preprint arXiv:2203.08975*.

A Appendix

A.1 Preliminaries

A common formulation of multi-agent reinforcement learning problem is Markov game (Littman 1994), which can be considered an extension of single-agent Markov Decision Process (MDP) to multi-agent setting. Markov game is defined as a tuple $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i \in \mathcal{N}}, \mathcal{P}, \{\mathcal{R}^i\}_{i \in \mathcal{N}}, \gamma)$, where $\mathcal{N} = \{1, ..., N\}$ is the set of agent indices with N agents,

$\mathcal{S}$ is state space, $\mathcal{A}^i$ is action space of agent i, and γ is discount factor. For convenience, we denote joint action space as $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$. Also, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$ is transition probability, $\mathcal{R}^i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is reward function of agent i. Every agent i has access to state $s_t \in \mathcal{S}$, based on which the agent can choose action a_t^i according to its policy $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$. Then, the agent receives the next state $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, \mathbf{a}_t)$ and reward $r_{t+1}^i = \mathcal{R}^i(s_t, \mathbf{a}_t)$, where $\mathbf{a}_t = (a_t^1, ..., a_t^N)$ is joint action at time step t.

The aim of each agent is to select a policy that maximizes individual value function

$$V^{\pi^i, \pi^{-i}}(s) = \mathbb{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} \mathcal{R}^i(S_{t'}, \mathbf{A}_{t'}) \Big| S_t = s_t \right], \quad (6)$$

where $\pi(\mathbf{a}_t|s_t) = \prod_{i=1}^{N} \pi^i(a_t^i|s_t)$ is combination of all agents' policies, and π^{-i} is combination of all agents' policies excluding the policy of agent i. In Markov games, Nash Equilibrium is widely used as the notion of optimality and is defined as joint policy $(\pi^{1,*}, \cdots, \pi^{N,*})$ where, for each $i \in \mathcal{N}$,

$$\forall s \in \mathcal{S}, \pi^i, \quad V^{\pi^{i,*}, \pi^{-i,*}}(s) \geq V^{\pi^i, \pi^{-i,*}}(s). \quad (7)$$

Intuitively, this means that every agent i is discouraged from choosing some other policy $\pi^i \neq \pi^{i,*}$ given the fixed policies of other agents $\pi^{-i,*}$.

We consider Decentralized Partially Observable Markov Decision Process (Dec-POMDP), a variant of Markov game that accounts for partial observability (Oliehoek, Spaan, and Vlassis 2008). Its definition is similar to that of Markov game, with the addition of the set of agent observations $\{\mathcal{O}^i\}_{i\in\mathcal{N}}$ and observation functions $\Omega^i : \mathcal{S} \to \mathcal{O}^i$ that maps state to individual agent's observation. An agent i has no access to the true global state s_t, and instead receives an observation $o_t^i := \Omega^i(s_t)$.

To enable agent-to-agent communication, we further allow a finite set of messages $\{\mathcal{M}^i\}_{i\in\mathcal{N}}$, from which each agent i can choose $m_t^i \in \mathcal{M}^i$ to communicate to other agents. In our problem setting, we assume that all agents' messages $m_t = (m_t^1, \cdots, m_t^N)$ at time step t are broadcasted to all agents and are concatenated to the next observations of each agent $i \in \mathcal{N}$ as $\tilde{o}_{t+1}^i = [o_{t+1}^i, m_t]$ at the next time step.

A.2 Implementation Details

Common Details of All Baselines. This section describes the common settings that all baseline methods implemented on top of IA2C (Mnih et al. 2016) follow. Actor function of each agent is a neural network that contains a hidden linear layer of size 64 followed by a Gated-Recurrent Unit (Cho et al. 2014) of size 64. The GRU is followed by another linear layer that outputs the logits for the actions, after which Softmax function is applied to yield the action probabilities. Critic function contains a single linear layer of size 64. Critic function learns to predict the n-step action value, where $n = 5$. The input to both the actor function and the critic function contains the current state and the messages of other agents.

Batch size of 10 is used. Adam optimizer (Kingma and Ba 2014) is used with $\alpha = 0.99$ and $\epsilon = 10^{-5}$. Soft updates are applied to target critic networks with $\tau = 0.01$. Rectified linear unit (ReLU) nonlinearity is applied to the output of every MLP or GRU.

IA2C. The experiments with IA2C can be considered a control group that does not communicate any message. Learning rate of 0.0005 is used for both the actor and the critic. The objective function contains an entropy term with coefficient of 0.01.

IA2C+M(R). This baseline serves as a naive decentralized method where the agents' communications are only driven by environmental rewards. Learning rate of 0.0005 is used for both the actor and the critic. The objective function contains an entropy term with coefficient of 0.01. Message length of 10 is used, and each message bit is discretized to values of 0 or 1. Each message bit is treated as a separate action channel, also trained with A2C.

IA2C+M(UES). Learning rate of 0.001 is used for both the actor and the critic. Message length of 10 is used, where each message element is a continuous value between 0 and 1. The objective function contains an entropy term with coefficient of 0.01.

To conduct dimensionality reduction on the observation, a random linear projection $g(\cdot)$ was applied on both the previous observation o_{t-1}^i and the current observation o_t^i. The linear projection is implemented as a linear layer of output size 64, and ReLU is applied.

The forward dynamics module $f(\cdot)$ in Section 2.2 receives the local observation embedding $g(o_{t-1}^i)$, the action a_{t-1}^i, and the messages of other agents m_{t-2}^{-i}. This module is a 2-layered MLP, with both layers having an output size of 64. Hence, to be precise, the unexpectedness x_t^i in Equations 1, 2, and 3 are calculated as $x_t^i = g(\hat{o}_t^i) - g(o_t^i)$.

The encoder is a linear layer with output size of 10, which is the length of the message, and the decoder is a linear layer with output size of 64 for reconstructing x_t^i.

IA2C+M(UES+R). The UEM structure is identical to the one used in IA2C+M(UES), except for the message size. The message yielded via reward scheme is of length 5, and the message yielded via UES is of length 5, which leads to a concatenated message length of 10. This matches the message length of other baselines, for fair comparison.

Learning rate of 0.0005 is used for both the actor and the critic. The objective function contains an entropy term with coefficient of 0.05.

A.3 Benchmark Environment Details

The environment is an adapted version of Multi-Robot Warehouse benchmark proposed in (Papoudakis et al. 2020). Refer to the snapshots in Figure 1. The five possible actions for each agent are {MoveForward, RotateLeft, RotateRight, Pickup/PutDown, NoOp}. Among the shelves (blue squares), four are requested (red squares) to be delivered to the goal location (teal tiles). An agent must then reach the tile containing a requested shelf, pick up the shelf, and carry

it all the way to the goal location to receive a reward of +0.5. The agent must also return the shelf to its original location, which grants an additional +0.5 reward. Each episode is 50 time steps long, and agent positions are randomized at the start of every episode.

Each tile can be occupied by at most one agent. If two agents try to move into the same tile, one agent is arbitrarily chosen to successfully move into the tile, and the other remains stationary.

An agent can move to a tile with a shelf and pick it up using the "Pickup/Putdown" action. If the agent is currently holding a shelf, it can put down the shelf via the same action. However, to avoid the shelves from being placed in narrow spaces such as corridors and blocking other shelves from being carried (one tile can only contain at most one shelf), the shelf can only be placed on tiles where the shelves are spawned. The remaining tiles where the shelves cannot be placed on are called "highways".

An agent can only observe the information of immediately surrounding tiles, within the 3×3 square centered on the agent. This information includes whether each tile is occupied by an agent, which direction that agent is facing, whether the tile contains a shelf, whether that shelf is requested, and whether the tile contains an impassable obstacle. Also, the agent observes its own grid position, the direction it is facing, whether it is carrying a shelf or not, and whether the current tile is a "highway".

To encourage each agent to carry requested shelves on their own while not specifically hindering the other agent from carrying the shelves, the reward function is designed such that the agent that successfully carries the requested shelf to the goal or returns the shelf to the original place receives the full reward of +0.5, while the other agent receives +0.125 reward. This modification was adopted to simplify the credit assignment problem of the given high-difficulty sparse-reward environment.