Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

APRES: Adaptive Policy-Aware Prioritized Experience Sharing for Scalable Decentralized Multi-Agent Reinforcement Learning

Arefeh Esmaili
Tarbiat Modares University of Technology
Mehdy Roayaei
mroayaei@modares.ac.ir

Tarbiat Modares University of Technology

Article

Keywords: Cooperative Multi-Agent Reinforcement Learning, Decentralized Experience Sharing, Heterogeneous Policy, Sample Efficiency, Scalability, Diversity

Posted Date: November 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11362220/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

APRES: Adaptive Policy-Aware Prioritized Experience Sharing for Scalable Decentralized Multi-Agent Reinforcement Learning

Arefeh Esmaili, Mahdi Roayaei[1]

Department of Electrical and Computer Engineering, Tarbiat Modares University of Technology, Tehran, Iran

Abstract

Decentralized cooperative Multi-Agent Reinforcement Learning (MARL) faces sever non-stationarity and sample inefficiency, particularly in sparse-reward environments where informative experiences may be discovered by only a subset of agents. Experience sharing can improve exploration by allowing agents to reuse knowledge acquired by their peers; however, indiscriminate sharing may introduce redundant or conflicting information and become increasingly problematic as the number of agents grows. Existing approaches generally do not jointly consider (1) which experiences are useful to a receiver, (2) whom to learn from, and (3) When does sharing become more beneficial?

To address this challenge, we present Adaptive Policy-aware Prioritized Experience Sharing (APRES), a lightweight and fully decentralized experience-sharing framework that addresses these three dimensions. First, APRES employs receiver-aware advantage prioritization, using the receiving agent's critic to identify shared experiences which are most informative relative to its current knowledge. Second, it introduces policy-aware weighting, based on clipped importance-sampling ratios to regulate the contribution of experiences generated by behaviorally different peers while limiting unstable gradient amplification. Third, an adaptive temporal decay schedule progressively reduces the contribution of shared experience as training advances, balancing early-stage collaborative exploration with late-stage policy specialization.

Experiments on sparse-reward cooperative MARL benchmarks show that our approach achieves higher performance while training on less shared experiences, improves sample efficiency, and scales significantly better than state-of-the-art experience-sharing baselines.

Keywords:

Cooperative Multi-Agent Reinforcement Learning, Decentralized Experience Sharing, Heterogeneous Policy, Sample Efficiency, Scalability, Diversity.

[1] Corresponding author.

mroayaei@modares.ac.ir (M.Roayaei), arefehesmaili@modares.ac.ir (A. Esmaili), Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran, Tel: +982182883939, P.O. Box 14115-111

1.Introduction

Cooperative MARL has emerged as a promising paradigm for coordinating multiple autonomous agents in complex real-world applications, including robotic warehouse management [1], autonomous vehicles [2], drone fleet operations [3], and traffic management [4]. In these applications, agents often operate in decentralized settings where each agent makes decisions based on its own local and partial observations while learning concurrently with other agents. Although decentralized learning avoids the need for a centralized controller and enables agents to operate autonomously, it introduces fundamental challenges for effective cooperation. In particular, the concurrent adaptation of multiple agents makes the learning environment non-stationary from the perspective of each individual agent, while partial observability and independent exploration can lead to inefficient acquisition of useful experience. These challenges become particularly severe in cooperative tasks with sparse rewards, where informative transitions are infrequent and successful behaviors may require coordinated exploration among agents.

Experience sharing among agents provides a promising approach to alleviating these challenges by enabling agents to reuse knowledge acquired through their peers' interactions. Instead of relying solely on locally collected experiences, agents can exploit informative transitions discovered by others, reducing redundant exploration and accelerating the propagation of useful behaviors across the team. This is particularly beneficial in sparse-reward environments, where informative experiences are rare and may be discovered by only a subset of agents. By incorporating shared experiences into policy learning, agents can exploit a broader pool of environmental interactions while maintaining decentralized execution [5].

However, existing experience-sharing methods in cooperative MARL do not explicitly account for the varying usefulness of shared experience across agents and throughout training. As a result, irrelevant or less informative experience may be transferred, potentially degrading learning performance. The value of shared experience is inherently dependent on three interrelated factors: the experience itself, the sending agent, and the stage of learning. First, existing prioritization mechanisms primarily rely on sender-centric criteria, such as TD error or return magnitude, without considering whether an experience is informative for the receiving agent. Second, most approaches do not explicitly account shared experience dependency to agent, although experiences generated by behaviorally diverse agents may either complement or interfere with the receiver's learning. Third, existing frameworks generally employ static sharing strategies and overlook the time-dependent nature of collaboration: intensive sharing can be beneficial during early exploration, whereas excessive sharing may hinder policy specialization and convergence at later stages. The central limitation is therefore not only whether agents share experience, but also consider (1) which experiences are useful to a receiver, (2) whom to learn from, and (3) When does sharing become more beneficial?

To overcome these challenges, this paper presents a lightweight, fully decentralized experience-sharing framework that explicitly addresses the three fundamental questions of experience sharing: **which experience to use**, **from whom to learn**, and **when to share experience.** The framework requires neither auxiliary neural architectures nor additional trainable parameters. Specifically, our primary contributions are:

- **Absolute Advantage-Guided Filtering (Which):** We introduce a receiving-agent prioritization mechanism based on the magnitude of absolute advantage. By evaluating incoming peer experiences using the receiving agent's own critic, the method selectively retains experiences that are most informative relative to the receiver's current knowledge, capturing both critical failures and unexpected successes.
- **Policy Weighting (Whom):** We introduce a policy-aware weighting mechanism based on clipped importance-sampling ratios to regulate the contribution of peer experiences and mitigate excessively large off-policy weights as policies diverge.
- **Adaptive Temporal Decay Scheduling (When):** We introduce an adaptive temporal decay mechanism that reduces the influence of shared experience as training progresses, balancing early collaborative exploration with later policy specialization.
- **Scalable decentralized experience sharing**: We demonstrate empirically that the combination of these mechanisms improves the efficiency and scalability of decentralized experience sharing, particularly as the number of cooperative agents increase.

The remainder of this paper is organized as follows: Section 2 presents related work, while section 3 introduces the necessary background. Section 4 details the proposed framework and its mathematical formulation. Section 5 presents experimental evaluation, benchmark comparisons, and ablation studies. Finally, Section 6 concludes the paper and discusses directions for future work.

2. Related work

2.1 Decentralized Cooperative MARL

Cooperative MARL requires multiple agents to learn coordinated behaviors while jointly interacting with a shared environment. As the number of agents increases, the joint action space grows rapidly, while concurrent policy updates make the environment non-stationary from the perspective of each individual agent. These challenges are further exacerbated under partial observability, where each agent has access only to local information. Consequently, achieving effective coordination while maintaining sample efficiency and scalability remains a central challenge in cooperative MARL [6].

Existing cooperative MARL methods can be broadly distinguished according to the degree of centralization during training and execution. *Independent learning*, such as Independent Q-Learning (IQL), allows each agent to learn independently while treating other agents as part of the environment. Although this approach is fully decentralized and computationally simple, simultaneous adaptation of the agents can lead to non-stationary learning dynamics and inefficient exploration [7].

At the opposite end of the spectrum, *Centralized Training and Centralized Execution* (CTCE) methods jointly determine the behavior of multiple agents using a centralized model. SABLE [8], for example, employs a transformer-based sequence model that jointly processes information from multiple agents to facilitate coordinated decision-making. Although centralized architectures can exploit global information and simplify coordination, their computational and communication

requirements may increase with the number of agents, which can limit their applicability to large-scale multi-agent systems.

A compromise is provided by *Centralized Training with Decentralized Execution* (CTDE), in which centralized information is available during training while agents retain decentralized policies during execution. Representative approaches include MADDPG [9] and MAPPO [10], which exploit centralized training information to improve learning while maintaining decentralized execution. Value-decomposition methods, such as VDN [11] and QMIX [12], similarly exploit centralized training information by representing a joint value function through agent-level value functions. Although these approaches can improve coordination and mitigate some difficulties of independent learning, their reliance on centralized training information introduces additional computational, communication, or structural dependencies that can become increasingly important as the number of agents grows.

In contrast, *Decentralized Training and Decentralized Execution* (DTDE) requires each agent to learn using locally available information and its own training process. This setting provides a high degree of autonomy and avoids dependence on centralized critics, global state representations, or centralized coordination mechanisms. However, decentralization also limits each agent's access to experiences discovered by its peers. Agents may therefore repeatedly explore similar regions of the environment, while useful behaviors discovered by one agent may remain unavailable to others. This creates a fundamental tension between decentralized scalability and limited knowledge reuse, motivating inter-agent knowledge and experience sharing as a complementary mechanism for improving cooperative learning [13].

2.2 Experience Sharing in Cooperative MARL

Inter-agent knowledge sharing has been investigated as a means of improving coordination and sample efficiency in cooperative MARL. Existing approaches can broadly be categorized according to the type of knowledge exchanged, including policies or parameters, gradients, and interaction experiences. Policy and parameter sharing enables agents to directly reuse learned representations or decision-making knowledge, whereas experience sharing allows agents to retain their individual policies while learning from interaction data collected by their peers [14–24]. This distinction is particularly important in decentralized cooperative MARL, where preserving agent-specific policies can support behavioral diversity and the emergence of complementary behaviors.

A fundamental approach to experience sharing is to make interaction data collected by one agent available to other agents for learning. Shared experience may consist of complete trajectories or subset of trajectories, including observations, actions, rewards, and subsequent observations. Such mechanisms allow agents to reuse environmental interactions beyond their own trajectories, reducing redundant exploration and potentially improving sample efficiency [25–28]. Experience sharing can therefore be particularly useful in sparse-reward environments, where informative transitions may be discovered by only a subset of agents.

Shared Experience Actor-Critic (SEAC) provides a representative approach to decentralized experience sharing and is particularly relevant to our work. Each agent can benefit from peer-

generated experience while maintaining its own independently parameterized policy. In its evaluation, SEAC demonstrated improved learning speed and final performance over independent learning and shared-policy baselines, particularly in sparse-reward cooperative environments [29]. Despite these benefits, the conventional SEAC formulation treats peer experience uniformly once it is made available to the receiving agent. In particular, it does not explicitly determine whether an individual shared experience is informative for the receiving agent, whether experiences from different peers should contribute differently, or whether the contribution of shared experience should vary throughout training. These limitations become increasingly relevant as the number of agents grows, because indiscriminate reuse of peer experiences can introduce redundant information, increase the variance of off-policy updates, and reduce the ability of agents to develop complementary behaviors.

This motivates a more structured view of decentralized experience sharing along three complementary dimensions: which experiences should be used, from whom should they be accepted, and when should their contribution be reduced or increased during training. The following sections review these three dimensions separately.

2.3 Which Experienced Should be used?

Experience sharing enables agents to learn from interaction data collected by their peers, but exposing an agent to all available shared experiences does not necessarily improve learning. Recent research has therefore moved beyond uniform experience sharing toward selective and adaptive knowledge exchange. For example, experience-prioritization methods use learning signals such as TD error or trajectory quality to identify potentially informative experiences [30–36], while other approaches regulate information exchange through selective communication, attention, or gated mechanisms [37,38,39]. However, the usefulness of shared experience should be evaluated from the receiver's perspective rather than solely according to sender-side criteria. This motivates receiver-side filtering, in which each agent independently determines whether a received experience provides sufficient additional learning value, while preserving decentralized experience sharing without requiring the sender to preselect experiences.

2.4 From Whom Should Experience Be Accepted?

Another important dimension of decentralized experience sharing concerns from whom a receiving agent should accept or use shared experiences. Early communication-based approaches investigated selective knowledge exchange in multi-agent reinforcement learning using tabular representations [40,41]. Memory-augmented methods further use memory and attention mechanisms to selectively integrate peer information with an agent's own experience [42].

Source selection can be based on spatial proximity, where communication is restricted to nearby agents [43–48], or on behavioral and semantic similarity, such as similar rewards or objectives [49]. Agent embedding [22] and role-based abstractions [50] can be used to identify similar agents and form groups or clusters for knowledge exchange [51,52]. Other approaches operate on aggregated or structured state representations to identify relevant experience sources [53]. Other approaches consider task-specific relationships, including behavioral similarity, value estimates, and goal alignment [31,54]. Also, experience sharing can also be asymmetric, where knowledge

is preferentially transferred from more reliable or knowledgeable agents to less reliable learners. In teacher–student approaches, teacher agents can be selected according to policy confidence or low policy entropy, directing experience from more certain to less certain agents [55].

Despite these advances, existing source-selection strategies mainly rely on proximity, similarity, task relationships, or source reliability, rather than directly measuring whether a sender's experiences provide complementary information to the receiving agent. Such criteria may therefore result in redundant experience sharing, motivating receiver-aware source selection based on the usefulness of peer experiences from the receiver's perspective.

2.5 When sharing is beneficial

Another dimension of decentralized experience sharing concerns when experiences should be exchanged. Rather than continuously sharing experiences, several approaches trigger communication when an experience is likely to provide useful information to another agent. [56,57,58] combine TD-error prioritization, information gain and Q-values respectively, with state visitation frequency to identify informative and infrequently encountered experiences. These approaches shift experience sharing from continuous exchange toward event-triggered communication, where sharing is initiated when information is sufficiently novel or informative. However, they primarily determine when to share based on novelty, visitation frequency, or sender-side information value, without considering how the usefulness of shared experiences changes throughout the learning process. Experience sharing may be more beneficial during early exploration, while excessive sharing later can introduce redundancy and interfere with policy specialization. This motivates an adaptive temporal mechanism that regulates shared experience according to the training stage.

2.6 Motivation for the proposed method

Motivated by the limitations mentioned in previous subsections, we propose a lightweight and fully decentralized framework for receiver-aware and adaptive experience sharing that jointly addresses three key questions: which experiences should be used, from whom should experience be accepted, and when should sharing occur. Rather than treating the value of an experience or the suitability of a sender as independent of the receiving agent, the proposed framework considers the relationship between shared experiences and the receiver's current learning behavior. This enables experience exchange to be selectively used when it is expected to provide complementary information, with the objective of improving sample efficiency and coordination while preserving the development of specialized and complementary agent behaviors.

3. Problem Formulation and Theoretical Foundations

3.1 Decentralized Partially Observable Markov Decision Process

Cooperative MARL can be formulated as a decentralized partially observable Markov decision process (Dec-POMDP), in which multiple agents interact with a common environment while each agent has access only to its local observation. A cooperative Dec-POMDP can be represented by

the tuple $(\mathcal{N}, \mathcal{S}, \{\mathcal{O}_i\}_{i=1}^N, \{\mathcal{A}_i\}_{i=1}^N, \mathcal{T}, \mathcal{R}, \gamma)$ where $N = |\mathcal{N}|$ is the number of agents, $\mathcal{S}$ is the global state space, O_i represents the local observation space for agent i, respectively, and $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots . \mathcal{A}_N$ is the joint action space. The transition function $\mathcal{T}(s'|$ s, a) defines the probability of transitioning from state s to s' after executing the joint action $a = (a_1, a_2, \ldots, a_N)$. The cooperative reward function is denoted by R(s, a, s') $\in \mathbb{R}$, and $\gamma \in [0,1)$ is the discount factor.

At time t, agent i receives only its local observation $o_i^t \in \mathcal{O}_i$ and selects an action according to its local policy $a_i^t \sim \pi_i(.|o_i^t; \phi_i)$ where ϕ_i denotes the parameters of the actor of agent i. The joint policy is therefore represented as:

$$\pi(a \mid o) = \prod_{i=1}^{N} \pi_i(a_i^t | o_i^t; \phi_i) \tag{1}$$

The objective of cooperative learning is to maximize the expected discounted team return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{H-1} \gamma^t r^t \right] \tag{2}$$

where $r^t = R(s_t, a_t, s_{t+1})$ and H denotes the episode horizon.

In this work, we focus on homogeneous cooperative agents that have compatible observation and action spaces but maintain separate actor and critic networks [55,59–61]. Each agent therefore learns its own policy from its locally collected experience. This corresponds to a DTDE setting: during execution, each agent selects actions using only its local observation and its own policy, without access to a centralized critic, global state, or centralized controller (Figure 1).

Although learning is decentralized, agents may exchange interaction experiences during training. Thus, the knowledge available to an agent is not restricted to experiences that it personally collected. This peer-to-peer experience exchange forms the basis for the shared-experience learning framework considered in the following sections.

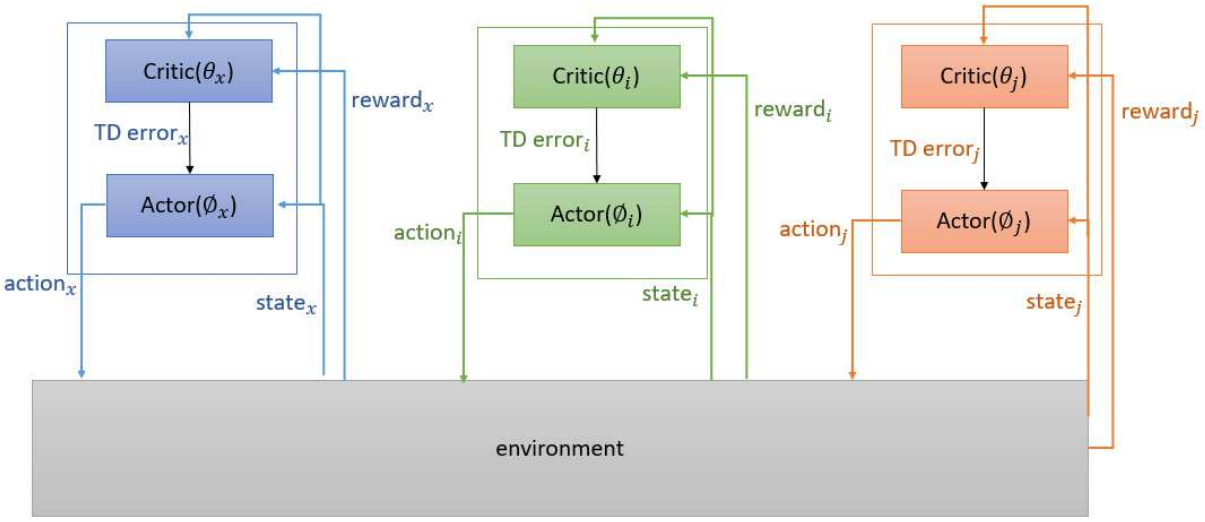


Figure 1: Overview of the decentralized training and decentralized execution (DTDE) framework. Each agent independently interacts with the environment using its own actor–critic networks.

3.2 Importance Sampling

Actor-critic

Actor-critic methods combine a policy network (actor) with a value network (critic). The actor determines the probability of selecting an action, while the critic estimates the expected return from the current observation. The actor represents the policy $\pi(a_i \mid o_i; \phi_i)$, while the critic estimates the value function $\mathrm{V}(o_i; \theta_i)$, where ϕ_i and θ_i denote the parameters of the actor and critic networks, respectively. For agent i, the actor and critic losses can be generally expressed as in Eqs. (3) and (4):

$$\mathcal{L}_{actor}(\emptyset_i) = -\log \pi(a_i^t | o_i^t; \emptyset_i)\sigma_i^t \quad (3)$$

$$\mathcal{L}_{critic}(\theta_i) = ||V(o_i^t; \theta_i) - y_i^t||^2 \quad (4)$$

where y_i^t is the target return and σ_i^t represents the corresponding learning baseline signal. The exact return estimation can be implemented using one-step or multi-step returns.

Importance Sampling

RL algorithms are broadly categorized into on-policy and off-policy methods according to the relationship between the data-generating policy (the *behavioral policy* ($\beta(a^t|r^t)$) and the policy being optimized (the *target policy* ($\pi(a^t|o^t)$)). In on-policy learning, policy updates are restricted to trajectories sampled from the current target policy ($\pi = \beta$). In contrast, off-policy methods can leverage trajectories collected by different behavioral policies. However, because the distribution of sampled transitions differs from that of the target policy, a distribution mismatch arises. To compensate for this discrepancy, importance sampling is commonly employed to reweight the contribution of off-policy transitions during optimization [62]. The importance-sampling ratio is defined as:

$$Importance\ Sampling\ (IS) = \frac{\pi(a_i^t|o_i^t;\emptyset_i)}{\beta(a_i^t|o_i^t;\emptyset_i)} \tag{5}$$

Importance sampling indicates the extent to which a sample generated by one distribution (the behavior policy β) should be weighted to estimate quantities under another distribution (the target policy π).

Shared experience actor-critic

As it mentioned before, this formulation provides a natural theoretical foundation for experience sharing in cooperative MARL. From the perspective of a receiving agent i, trajectories generated by another agent j can be regarded as off-policy experiences generated by a different behavioral policy β_t. Importance sampling therefore enables agent i to reuse the experiences of its peers while compensating for the discrepancy between peers' behavioral policies and its own target policy. In this setting, the current policy of agent j is behavioral policy, so $\beta = \pi_j$ and the current policy of agent i is target policy, so $\pi = \pi_i$. Therefore, importance-sampling ratio in MARL is defined as:

$$\rho_{ij}^t = \frac{\pi\left(a_j^t\middle|o_j^t;\emptyset_i\right)}{\pi\left(a_j^t\middle|o_j^t;\emptyset_j\right)} \tag{6}$$

Based on this principle in MARL, the actor and critic loss function for the receiving agent i can be extended to incorporate both its own on-policy trajectories and the off-policy trajectories received from other agents by the help of importance sampling ratio. So, policy loss in MARL as mentioned in [29] extend to:

$$\mathcal{L}(\emptyset_i) = -\mathbb{E}_{(o_i^t,a_i^t)\sim D_i}[\mathcal{L}_{actor}(\emptyset_i)] - \sum_{j\neq i}\mathbb{E}_{\left(o_j^t,a_j^t\right)\sim D_j}[\rho_{ij}^t\mathcal{L}_{actor}\left(\emptyset_j\right)] \tag{7}$$

And value loss in MARL extend to:

$$\mathcal{L}(\theta_i) = \frac{1}{2}\mathbb{E}_{(o_i^t,a_i^t)\sim D_i}[\mathcal{L}_{critic}(\theta_i)] + \frac{1}{2}\mathbb{E}_{\left(o_j^t,a_j^t\right)\sim D_j}[\sum_{j\neq i}\rho_{ij}^t\mathcal{L}_{critic}\left(\theta_j\right)] \tag{8}$$

By augmenting each agent's local actor-critic updates with importance-weighted experience computed from the trajectories of other agents and allowing agents to learn from other agent, this idea improves exploration efficiency and accelerates convergence, particularly in sparse-reward environments where informative transitions are rare. In practice, agents may explore different regions of the environment, develop distinct policies, and encounter experiences with varying relevance.

3.3 Problem Definition

Given a receiver i, a set of peer trajectories generated by agents $j \neq i$, determine which received experiences should be used, from which peers their contributions should be trusted, and how strongly shared experience should influence learning at the current training stage. To address these challenges, we propose Adaptive Policy-aware Prioritized Experience Sharing (APRES).

4. Proposed method

APRES extends the shared-experience learning paradigm through three complementary mechanisms: (i) Advantage-Guided Experience Prioritization for selecting informative experience, (ii) Policy-Aware Weighting for regulating the contribution of agents, and (iii) Temporal Decay Scheduling. The notations and parameters used throughout the proposed method are summarized in Table 1.

Table 1-Parameters descriptions

Parameter	Explanation
$\pi_{\emptyset_i}$	Actor network of agent i parameterized by $\emptyset_i$
Q_{θ_i}	Critic network of agent i parameterized by θ_i
$\mathcal{L}$	Loss function
a_i^t	Action selected by agent i in time step t
o_i^t	Observation of agent i in time step t
r_i^t	Reward received by agent i in time step t
σ_i^t	Learning baseline signal
y_i^t	Target return
D_i	Local replay buffer of agent i
τ_i	Trajectories Collected by agent i
ρ_{ij}^t	Importance sampling ration between agent i, j in time step t
N	Number of agents
G_i^t	Return of agent i in time step t
A_i^t	Advantage of agent i in time step t
$D^*{}_i$	Selected local replay buffer of agent i
δ_{ij}^t	Clipped importance sampling ratio between agent i, j in time step t
λ^t	Adaptive temporal decay

4.1 Advantage-Guided Experience Prioritization (Which Experience)

Existing prioritization schemes are predominantly sender-centric, relying on metrics such as TD error or cumulative return computed by the sender. However, the usefulness of a transition is inherently receiver-dependent: a trajectory that is highly informative for one agent may provide little new information to another agent whose critic already accurately predicts its outcome. Therefore, experience selection should be performed from the perspective of the receiving agent.

Consider a receiving agent i and a sending agent j. Let $\tau_j = \{(o_j^t, a_j^t, r_j^t, o_j^{t+1})\}$ denote a trajectory collected by agent j. Instead of uniformly incorporating every shared experience in τ_j, agent i

evaluates the usefulness of each incoming experience using its own critic. Specifically, the receiving agent estimates the absolute advantage on the sender's experience with respect to its current value function:

$$G_j^t = r_j^t + \gamma r_j^{t+1} + \cdots + \gamma^{T-t} V\left(o_j^T; \theta_j\right) \quad (9)$$

$$A_j^t = |G_j^t - V\left(o_j^t; \theta_i\right)| \quad (10)$$

where G_j^t denotes the multi-step return computed from the sender agent's experience and $V\left(o_j^t; \theta_i\right)$ is the value function of the receiving agent calculated on sender's shared experience. This formulation measures the discrepancy between the outcome achieved in the shared experience and the receiving agent's current expectation for the corresponding state. A larger absolute difference suggests greater potential learning value for the receiving agent. Consequently, the same experience may have different learning value for different receiving agents. Since agents are homogeneous and cooperate under a shared reward structure, the sender's return serves as an informative estimate of the long-term outcome associated with the shared experience from the receiver's perspective.

Specifically:

- **Large absolute advantage:** indicates that the experience strongly deviates from the receiver's current expectations and therefore carries high learning value. Agent learns from both success and failure.
- **Small absolute advantage:** indicates that the experience is already well explained by the receiver's critic and provides limited additional information.

APRES ranks shared experiences according to their absolute advantage magnitude and selects the highest-priority experiences for experience sharing. To adapt the amount of shared experience to the learning stage, APRES employs a linear decay in the experience-selection ratio, decreasing it from 100% at the beginning of training to 50% in the later stages shown in Eq. (11). This design is motivated by the different knowledge requirements at different stages of learning. During the early stages, agents have limited task knowledge and their learned policies are still developing; therefore, retaining all shared experiences promotes broad knowledge transfer and prevents potentially useful experiences from being prematurely discarded. As training progresses and the agents acquire more task-dependent knowledge, APRES progressively increases the selectivity of experience sharing by retaining experiences with larger learning signals, as indicated by their advantage magnitude.

The selection ratio is deliberately bounded below by 50% rather than decreasing to zero. A zero-selection ratio would eliminate experience sharing and consequently approach independent learning, which is outside the objective of APRES. Maintaining a minimum selection ratio of 50% ensures that substantial cross-agent experience sharing is preserved throughout training while reducing the transmission of lower-priority experiences in later stages.$D^*{}_j$ is the subset of local replay buffer D_j which is define as Eq. 11:

$$D^*{}_j = top_k(A_j^t\ , k = max(0.5, \min(1 - 0.5 * \tfrac{t}{T}))) \quad (11)$$

where t denotes the current optimization step and T denotes the total number of training updates. This selective prioritization suppresses redundant experiences while allowing the receiving agent to concentrate on the most informative shared experience.

4.2 Policy-Aware Weighting (From Whom)

Large difference between the behavioral policy and the target policy can increase the variance of importance-sampling estimates. In the context of shared experience, the importance-sampling ratio compares the probability assigned by the receiving agent to the probability assigned by the sending agent. When the receiving policy assigns a substantially higher probability to the shared action than the sending policy, the resulting importance-sampling ratio can become large, causing explosion in gradient update calculated in loss function of actor and critic. This effect can become increasingly problematic as the number of learning agents increase, since each agent may have a substantially different policy.

To mitigate the influence of excessively large importance-sampling ratios, APRES applies an upper-bound truncation to the ratio. Specifically, the importance-sampling ratio is constrained to a maximum value of one. As it shown in Eq. (13), when the receiving policy assigns a probability greater than or equal to sending policy on shared experience, the ratio is clipped to one, preventing the importance sampling ratio from being amplified. Conversely, when the receiving policy assigns a lower probability than the sending policy, the resulting ratio remains below one and importance sampling get power factor of two which down-weights the contribution of peer shared experience. APRES do this by multiplying importance sampling with suggested coefficient shown in Eq. (12):

$$\delta_{ij}^{t} = e^{-|\log \pi_j - \log \pi_i|} * \rho_{ij}^{t} \tag{12}$$

$$\delta_{ij}^{t} = \begin{cases} \left(\dfrac{\pi\left(a_j^t \middle| o_j^t; \emptyset_i\right)}{\pi\left(a_j^t \middle| o_j^t; \emptyset_j\right)}\right)^2, & \pi\left(a_j^t \middle| o_j^t; \emptyset_i\right) < \pi\left(a_j^t \middle| o_j^t; \emptyset_j\right) \\ 1, & \pi\left(a_j^t \middle| o_j^t; \emptyset_i\right) \geq \pi\left(a_j^t \middle| o_j^t; \emptyset_j\right) \end{cases} \tag{13}$$

$$0 < \delta_{ij}^{t} \leq 1 \tag{14}$$

Importance-Sampling Ratio Greater Than One: When the importance-sampling ratio is greater than 1, it means that the receiving agent assigns a higher probability to the shared action than the sending agent. In APRES, such ratios are clipped to 1 to prevent the shared experience from being excessively amplified**.** The clipped value indicates that the receiving agent has relatively high acceptance of the shared experience and trust more in sending agent policy, while avoiding an increase in its contribution beyond the original scale.

Importance-Sampling Ratio Less Than One: When the importance-sampling ratio is less than 1, it means that the receiving agent assigns a lower probability to the shared action than the sending agent. APRES retains this ratio below one, thereby reducing the contribution of the corresponding

shared experience to the receiver's learning update. A smaller ratio therefore indicates lower acceptance of the shared experience and trust less in sending agent policy and results in a proportionally reduced contribution to the learning update.

4.3 Adaptive Temporal Decay Scheduling: When

The usefulness of shared experience may also vary throughout training. During the early stages of learning, agents face severe uncertainty and non-stationarity, making shared experience particularly valuable for accelerating exploration and discovering sparse rewards. As training progresses, however, agents develop increasingly specialized behaviors and more accurate value estimates. Excessive experience sharing at later stages may therefore introduce unnecessary interference and reduce the ability of individual agents to refine their own policies.

APRES therefore introduces a temporal coefficient that controls the overall contribution of shared experience as shown in formula (15):

$$\lambda^t = max(\alpha_{min},\ 1 - \frac{t}{T}) \tag{15}$$

where t denotes the current optimization step and T denotes the total number of training updates $\alpha_{min} = \text{round}(\frac{1}{N},1)$ establishes a residual sharing floor with adaptive to N which is number of agents, where λ^t prevents the contribution of shared experience from disappearing completely. So, shared experience has its strongest influence at the beginning of training and its contribution is gradually reduced as training progresses. Consequently, APRES dynamically balances cooperation and specialization: it promotes intensive experience sharing when exploration is crucial and gradually shifts the learning process toward independent policy refinement as training converges.

4.4 Unified APRES Objective and Loss Formulation

APRES integrates the three proposed components (receiver-side advantage prioritization, policy-aware weighting, and adaptive temporal decay scheduling) into a unified decentralized learning objective. Given a shared experience received from agent j, agent i first determines whether the experience belongs to the prioritized subset, then evaluates its policy trust, and finally modulates its contribution according to the current training stage.

The actor and critic objectives of the receiving agent are obtained by incorporating the unified weighting coefficient into its losses. The resulting actor and critic losses are shown in Eqs. (16) and (17), respectively:

$$\begin{aligned}\mathcal{L}(\emptyset_i) = &-\mathbb{E}_{(o_i^t,a_i^t)\sim D_i}[\log \pi(a_i^t|o_i^t;\emptyset_i)\,(A_i^t)] \\ &- \lambda^t \sum_{j\neq i} \mathbb{E}_{\left(o_j^t,a_j^t\right)\sim {D^*}_j}[\delta_{ij}^t \log \pi\left(a_j^t\middle|o_j^t;\emptyset_i\right)\left(A_j^t\right)]\end{aligned} \tag{16}$$

$$\mathcal{L}(\theta_i) = \frac{1}{2}\mathbb{E}_{(o_i^t,a_i^t)\sim D_i}[||A_i^t||^2] + \lambda^t \sum_{j\neq i}\frac{1}{2}\mathbb{E}_{\left(o_j^t,a_j^t\right)\sim {D^*}_j}[\delta_{ij}^t||A_j^t||^2] \tag{17}$$

Pseudo code of APRES is depicted in **Algorithm 1**.

Algorithm 1: Pseudo code for APRES

Input: Number of agents N
Output: Trained policies $\{\pi_i\}$ for $i = 1 \ldots N$
Initialize $actor_i$ network $\emptyset_i$, $critic_i$ network θ_i randomly for each $agent_i$ $i \in \{1, \ldots, N\}$ Initialize local replay buffers D_i for each agent.
For episode= 1 to M do:
    // Phase 1: which experience to select.
    Each agent i explore environment individually and collects trajectories τ_i using current policies π_i.
    Store τ_i in local buffer D_i.
    Send D_i to other agents. Also, receive D_j to from other agents.
    Select top k from D_j considering absolute advantage using Eq. (11).
    // Phase 2: from whom to accept knowledge
    Clip importance sampling to up weighted more trusted agent and hinder gradient explosion using Eq. (12).
    // Phase 3: when to use shared experience
    Calculate Adaptive temporal decay to make exploration first then policy specialize in the end using Eq. (15).
    Update actor and critic loss by aggregated knowledge using Eqs. (16) and (17).
End For
Return trained policies $\{\pi_i\}$ for $i = 1 \ldots N$

Unlike many existing approaches that rely on graph neural networks, attention mechanisms, explicit communication protocols, or auxiliary modules to estimate inter-agent relevance, APRES introduces a lightweight and fully decentralized experience-sharing strategy that requires only advantage estimation, policy trust evaluation, and adaptive scheduling. Consequently, APRES improves the quality of shared experiences while preserving the simplicity, scalability, and decentralized nature of real application.

5. Experimental Evaluation

5.1Configuration

All methods will be described in this section were evaluated on identical hardware: NVIDIA GeForce RTX 3060 GPU (12 GB), an Intel(R) Core (TM) i7-4790K CPU, and 16 GB RAM. The operating system was windows 10 pro, and all experiments used Python 3.8 with PyTorch 1.8.1 To ensure a fair comparison, we fixed the hyper parameters to those used by [29] : a learning rate of 3e-4, a batch size of 24, the Adam optimizer, a discount factor $\gamma = 0.99$ and a two-layer MLP with 64 hidden units for both actor and critic network. No additional tuning was performed for any baseline.

Code availability

The source code of our algorithm is publicly available in a GitHub repository: https://github.com/mroayaei/APRES.

5.2 Environment

RWARE

We evaluate the proposed methods in the multi-robot warehouse environment, RWARE [29], using standard benchmark configuration. In RWARE, multiple robots navigate a grid-based warehouse to transport requested shelves to designated goal locations. In Figure 2, agents (robot) are shown in yellow; when carrying a shelf, an agent changes to red. Agents can rotate, move forward, and load or unload shelves (depicted as purple squares). Robots carrying shelves must navigate the warehouse corridors and avoid collisions with other robots.

RWARE is a partially observable environment. Each agent observes a local 3*3 area around itself, together with its absolute position, current orientation, carrying status, and local information about other agents and shelves within their observation range, including shelf identifiers and whether a shelf is currently requested (highlighted in green).

At each time step, R shelves are requested to be carried to one of the goal positions denoted as G colored in black. An agent receives a reward of 1.0 only when it successfully delivers a requested shelf to a goal. After delivery, the completed request is replaced by a new one, maintaining a constant demand level. Therefore, RWARE presents a sparse-reward and delayed-feedback challenge, since agents receive feedback only after completing the entire delivery task. An additional challenge arises from the requirement that agents must return delivered shelves to valid empty storage locations before picking up another shelf. Since storage positions may become occupied by other agents during navigation, agents must dynamically search for available unloading locations, significantly increasing coordination difficulty.

RWARE provides four warehouse sizes: tiny (10*11), small (10*20), medium (16*20), and large (16*29). The number of agents is denoted by (N), while the number of simultaneously requested shelves is denoted by (R). In the default setting, (R=N). The easy setting uses (R=N/2), whereas the hard setting uses (R=2N).

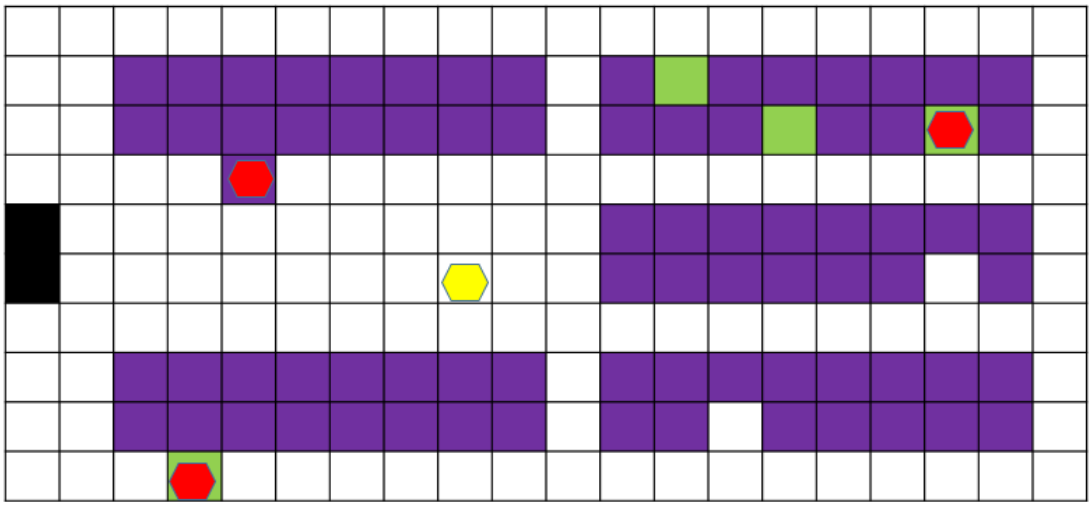


Figure 2: RWARE-small (20*10)-4ag environment [29]

5.3 Ablation study

In this section we investigate three proposed components of APRES individually and see effect of each of them separately in RWARE environment compare to Shared Experience Actor-Critic (SEAC) method. SEAC is a full sharing experience method that represents one of the strongest DTDE baselines evaluated on the RWARE benchmark and achieves state-of-the-art performance among fully decentralized methods. Therefore, it provides an appropriate reference point for evaluating the effectiveness of APRES.

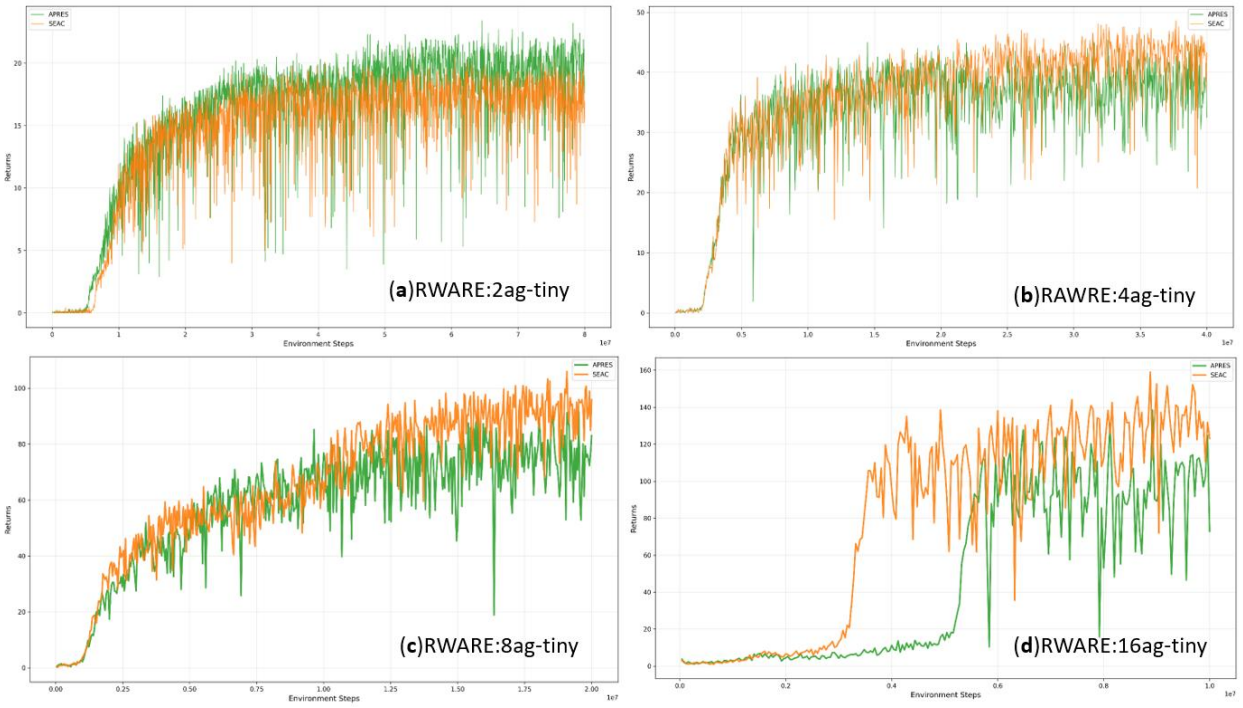


Figure 3-Comparison of which to share components of APRES with SEAC on RWARE environments

As it shown in Figure 3, linearly decreasing in the amount of data agent trained on normally decrease performance of APRES, because it learns on less shared experience but in combination of all three components choosing which experience to learn on improve APRES performance, that mean not only decrease computational overhead in receiver side but also help improving result in combined model and help scalability, too.

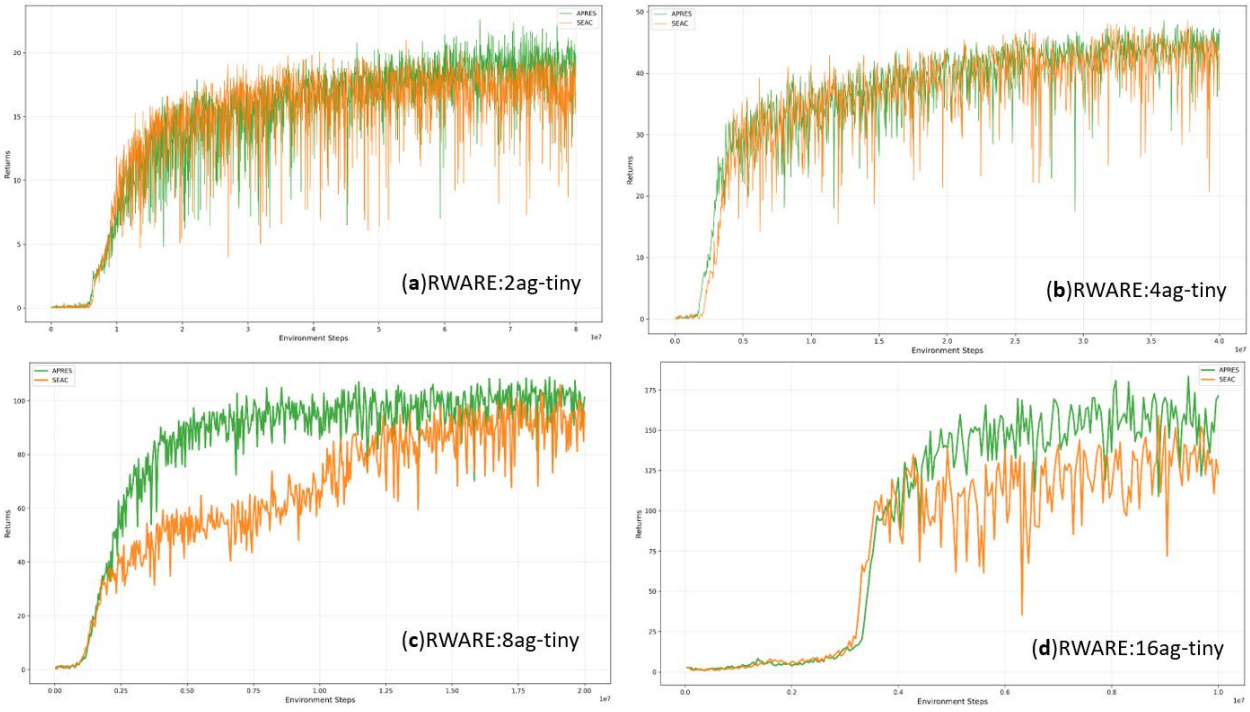


Figure 4- Comparison of from whom to learn components of APRES with SEAC on RWARE environments

As shown in Figure 4, choosing whom to learn by putting trust value on receiving agent itself in proportion to sender instead of trusting all agent uniformly that SEAC did in their method help faster convergence and better result in APRES method in all environment.

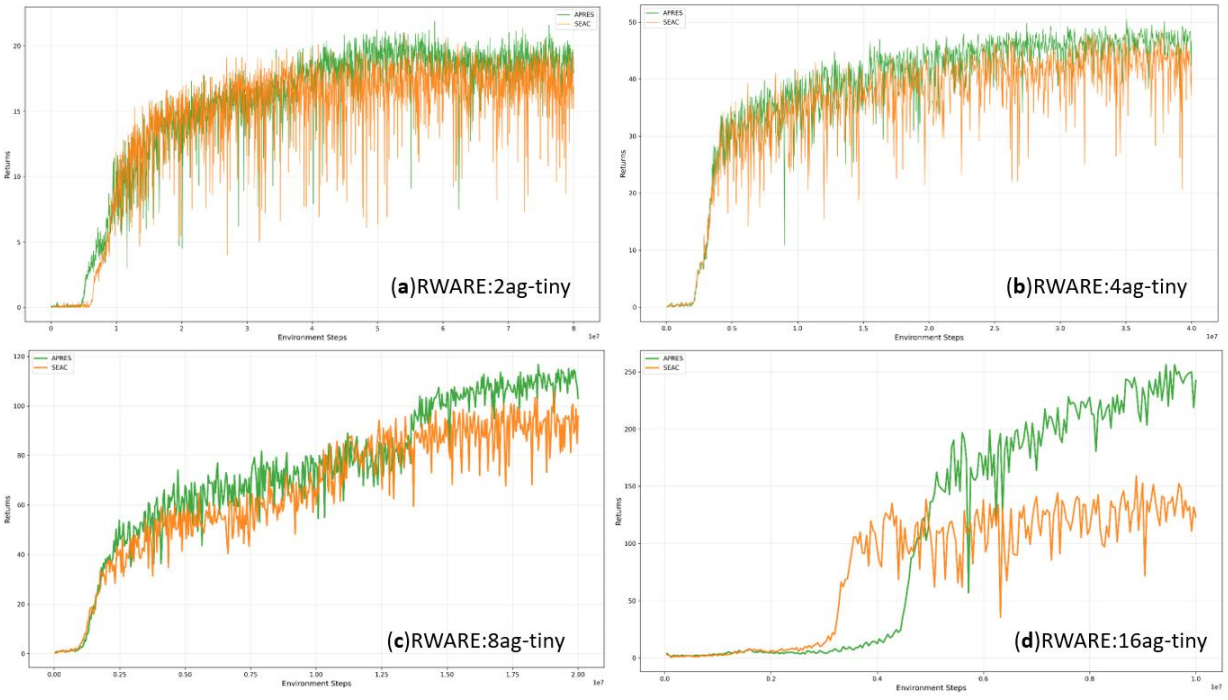


Figure 5-Comparison of when to share components of APRES with SEAC on RWARE environments

As it shown in Figure 5, APRES linearly decrease the amount of sharing from full sharing in early stage for exploring more when agent does not have enough knowledge to near independent learning for having discriminate policy in late stage of learning focus on when to share experience, result better in all four environment than SEAC that suppose full sharing experience all the time.

5.4 Benchmarking on Cooperative MARL

In this section, we compare our proposed method (APRES) against both fully centralized and fully decentralized MARL approaches. As reported by [63], most MARL algorithms struggle in the RWARE environment due to its sparse reward structure and partial observability.

Recent and state of the art methods include independent learning approaches without sharing mechanism, such as Independent Q-Learning (IQL) and Independent Advantage Actor-Critic (IA2C) and Independent Proximal Policy Optimization (IPPO) ; centralized training with decentralized execution (CTDE) approaches, including Multi-Agent Deep Deterministic Policy Gradient (MADDPG), Counterfactual Multi-Agent Policy Gradients (COMA), and Multi-Agent Advantage Actor-Critic (MAA2C); as well as value-decomposition methods based on joint action-value factorization, namely Value-Decomposition Networks (VDN) and Monotonic Value Function Factorizations for Deep Multi-Agent Reinforcement Learning (QMIX). We additionally compare against SABLE [8], a recent transformer-based centralized MARL architecture.

SABLE [8]: SABLE is fully centralized which means it use centralized training centralized execution (CTCE) in the form of transformer. SABLE replaces the attention mechanism in multi agent attention (MAT) with an RL-adapted version of retention mechanism derived from Retentive Networks [64], enabling efficient sequence modeling for multi-agent coordination.

VDN [11]: VDN learns a linear decomposition of the joint Q-value. Each agent independently estimates its local Q-value, while the global joint Q-value is represented as the sum of all individual Q-values.

QMIX [12]: QMIX extends VDN by introducing a non-linear mixing network to combine individual agent Q-values into a joint action-value function. The mixing architecture is constrained to preserve monotonicity, ensuring that the optimal joint action corresponds to the combination of individually optimal actions.

MADDPG [9]: It is a multi-agent extension of the DDPG [30]. Each actor operates using local observations, while a centralized critic is trained using the joint observations and actions of all agents to estimate the joint state-action value function.

COMA [65]: COMA introduces a counterfactual advantage function for multi-agent policy gradients. The advantage is computed as the difference between the value of the executed joint action and a counterfactual baseline obtained by marginalizing over one agent's action while keeping other agents' actions fixed.

MAA2C: AA2C is the multi-agent extension of A2C implemented by [63]. It employs a centralized value function to reduce variance in policy gradient estimation and is sometimes referred to as Central-V in MARL literature.

MAPPO [10]: It extends PPO to cooperative multi-agent settings. During training, it employs a centralized critic with access to global state information. MAPPO has demonstrated strong empirical performance across many cooperative MARL benchmarks.

IQL, IA2C, IPPO: IQL, IA2C and IPPO apply conventional single-agent Q-learning, A2C and PPO independently to each agent without explicit coordinator or information sharing mechanisms implemented by [63,66].

Table 2 present the highest episode returns obtained by the evaluated algorithms on the RWARE benchmark. Due to computational resource limitations, the results for all methods except APRES were taken from the benchmark studies reported in [63]; therefore, only two experimental configurations are included.

Table 2: Max return for all 11 algorithms tested on RWARE. Bold values indicate the highest score per task.

Framework	Task/Algorithm	RWARE-2ag-tiny	RWARE-4ag-tiny
CTCE	SABLE	$\mathbf{22.11 \pm 0.79}$	46.82 ± 1.42
CTDE	VDN	0.06 ± 0.08	0.34 ± 0.17
	QMIX	0.06 ± 0.08	0.16 ± 0.15
	MADDPG	0.24 ± 0.21	0.4 ± 0.24
	COMA	1.46 ± 0.34	1.48 ± 0.48
	MAA2C	6.16 ± 1.94	31.56 ± 4.29
	MAPPO	17.48 ± 4.52	38.74 ± 2.99
DTDE	IQL	0.06 ± 0.08	0.44 ± 0.10
	IA2C	5.56 ± 1.68	18.02 ± 5.87
	IPPO	8.70 ± 4.04	14.10 ± 5.20
	APRES	20.1 ± 0.32	$\mathbf{48.9 \pm 1.29}$

APRES achieves competitive performance across both tasks, substantially outperforming all decentralized baselines and matching or exceeding the performance of several centralized methods. APRES achieves performance very close to SABLE, a state-of-the-art fully centralized transformer-based method, despite operating entirely without centralized critics or global coordination mechanisms.

The results further demonstrate that many widely used MARL algorithms struggle in sparse-reward environments such as RWARE, even when equipped with centralized training components. In contrast, APRES achieves strong performance while remaining fully decentralized. These findings suggest that properly experience sharing can provide more effective coordination signals than centralized critics under highly sparse and partially observable conditions, while preserving scalability.

5.5 APRES vs SEAC

In this section we compare all three combined components investigated in ablation study with SEAC method. Also, we evaluate scalability of APRES by increasing number of agents presented in Figure 6-Figure 9 and changing in size of environment presented in Figure 10-Figure 11

compare with SEAC. Table 3, shown mean episode returns of SEAC and APRES in train and evaluation phase on different configurations of RWARE.

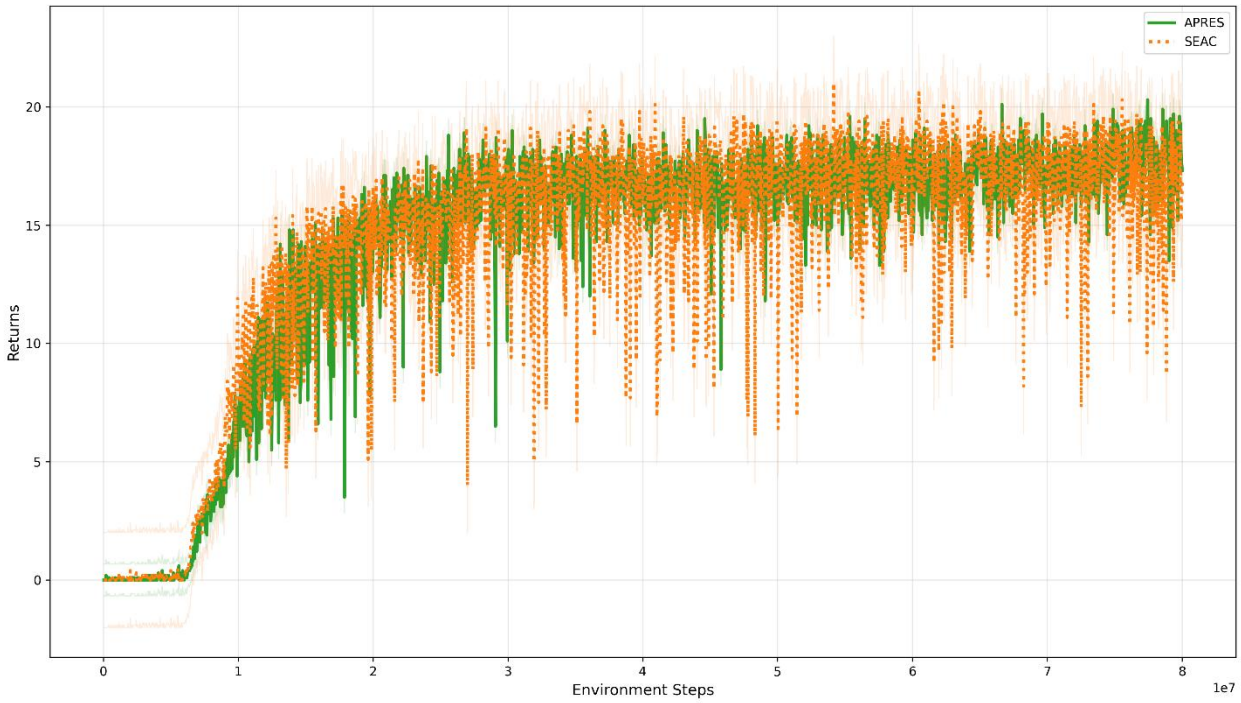


Figure 6-Comparison of APRES and SEAC on 2ag-tiny-RWARE

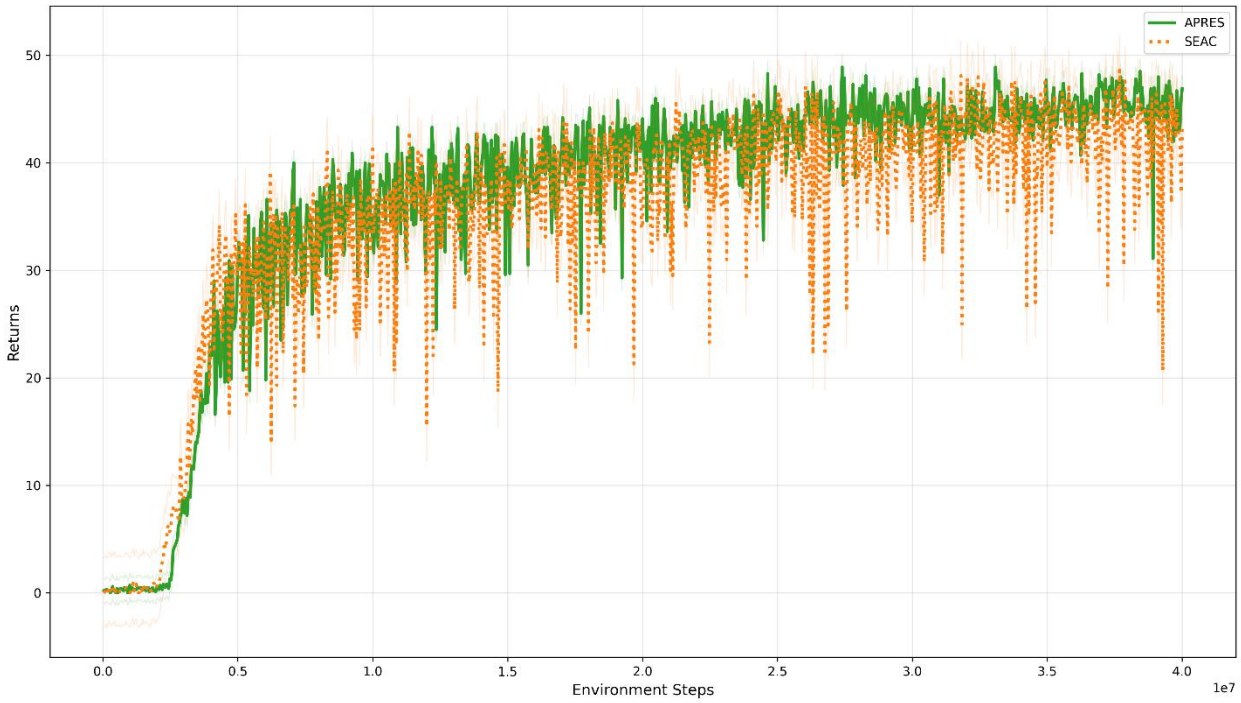


Figure 7-Comparison of APRES and SEAC on 4ag-tiny-RWARE

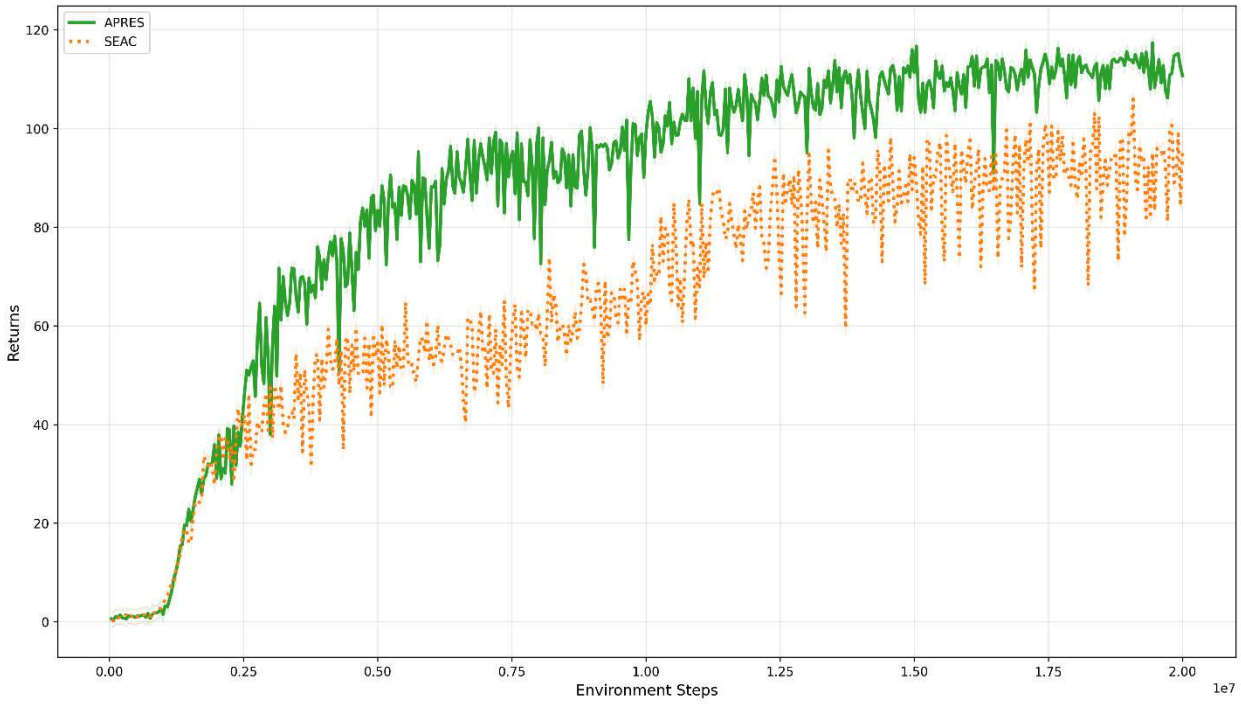


Figure 8-Comparison of APRES and SEAC on 8ag-tiny-RWARE

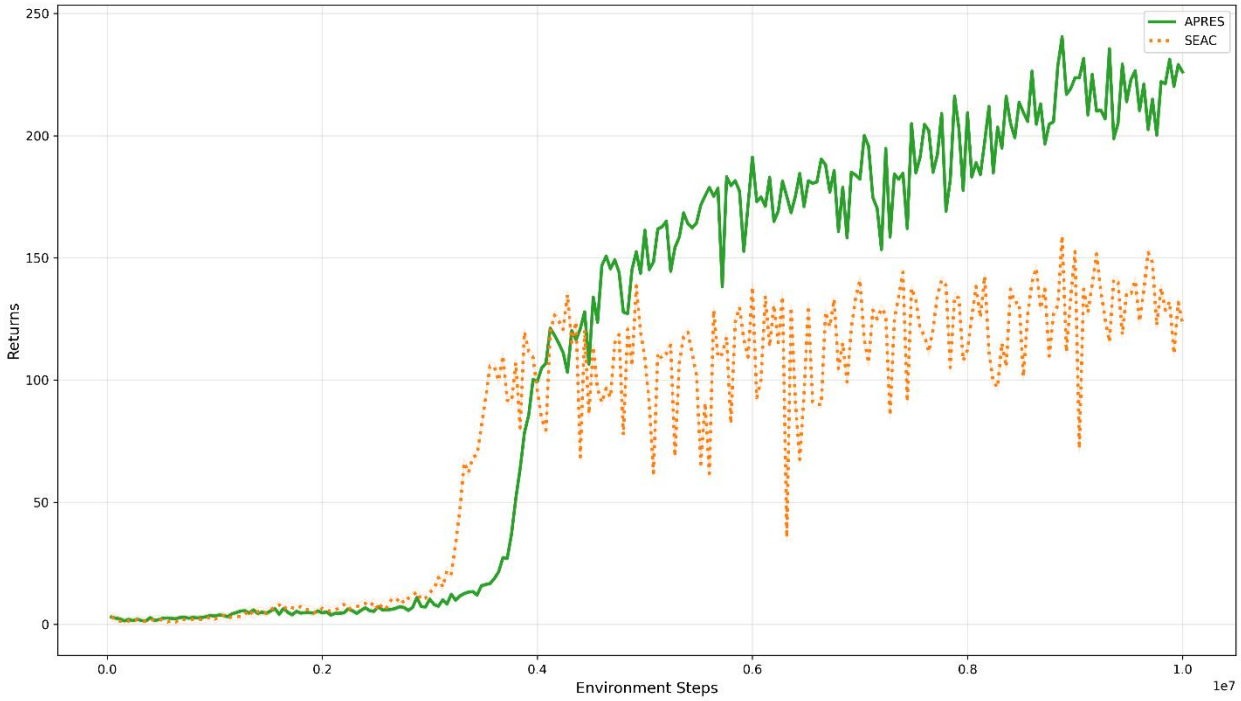


Figure 9-Comparison of APRES and SEAC on 16ag-tiny-RWARE

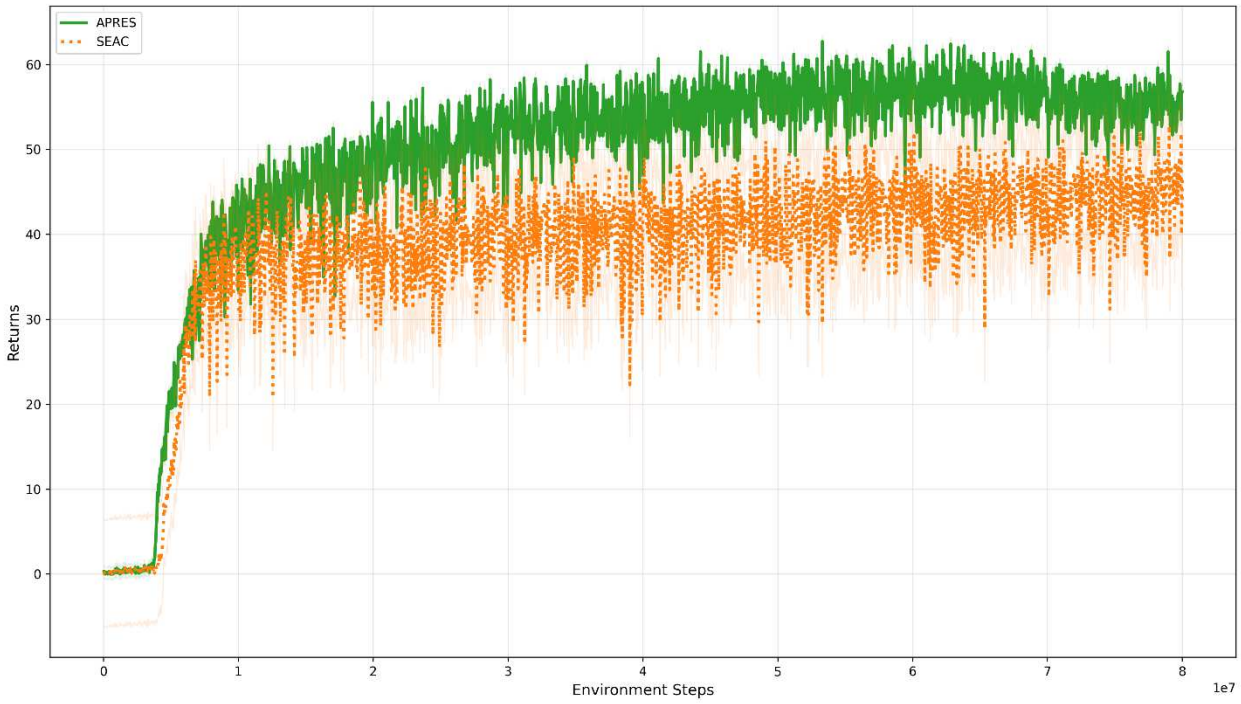


Figure 10- Comparison of APRES and SEAC on 8ag-small-RWARE

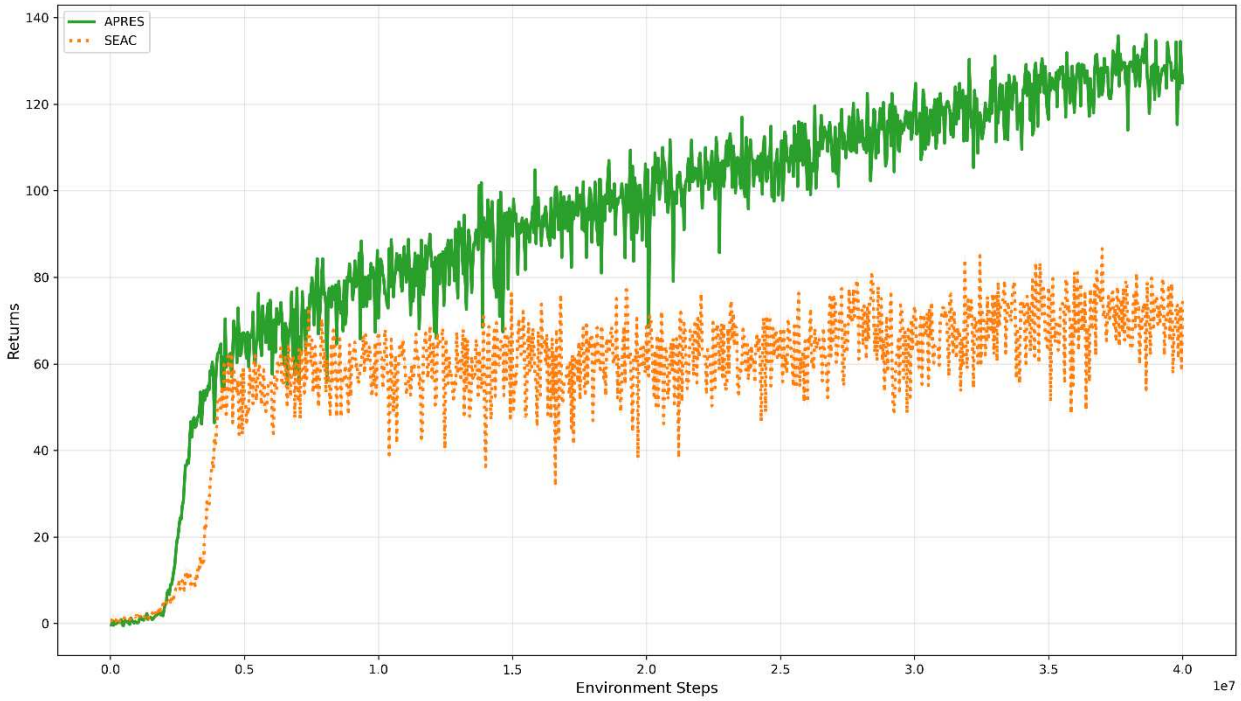


Figure 11- Comparison of APRES and SEAC on 16ag-small-RWARE

Table 3: Mean episode returns of SEAC and APRES in train and evaluation phase on RWARE. Bold values indicate the highest score per task.

Task/Algorithm	SEAC-train	SEAC-evaluation	APRES-train	APRES-evaluation

RWARE-2ag-tiny	16.9±2.01	**17.625±2.16**	**17.33±0.67**	17.25±0.89
RWARE-4ag-tiny	43.3±3.23	45±1.39	**46.9±1.13**	**45.125±1.66**
RWARE-8ag-tiny	95.9±1.37	79±13.24	**110.7±1.5**	**111±6.64**
RWARE-16ag-tiny	123.1±1.56	104.5±2.7	**226.1±0.9**	**236.125±0.3**
RWARE-8ag-small	46.4±6.3	39.75±9.75	**56.8±0.7**	**53.625±1.09**
RWARE-16ag-small	74.6±0.75	75.125±0.72	**125.7±0.7**	**123.125±0.85**

As it shown in Table 3, APES get higher result both in train and evaluation phase compare to SEAC, which show APRES effectiveness combined component that carefully select which experience should be used and from whom should experience accepted and when sharing is beneficial, instead of full sharing happen in SEAC. Also, APRES help scalability, as the number of agents and the complexity of the environment increase, the gap between the APRES and SEAC widens.

6.Conclusion

In cooperative Multi-Agent Reinforcement Learning (MARL), sharing experiences across agents offers a compelling strategy to accelerate learning and enhance joint exploration in sparse-reward environments. However, conventional shared-experience frameworks treat incoming peer data uniformly—ignoring experience utility, behavioral diversity, and evolving learning dynamics. This indiscriminate integration induces high-variance gradient updates, suppresses role specialization, and causes policy homogenization. To address these fundamental limitations, this paper presented **Adaptive Policy-aware Prioritized Experience Sharing (APRES),** a lightweight framework that refines cross-agent experience sharing along three core operational dimensions:

1. **Which Experience to Select (Absolute Advantage Prioritization):** By evaluating incoming shared experiences through the receiver's critic using absolute advantage magnitude, APRES isolates high-informative breakthroughs and critical failures while filtering out routine, uninformative shared experiences.
2. **From Whom to Accept Knowledge (Policy-Aware Weighting):** APRES establishes a behavioral-level trust gate that use clipped importance sampling ratios, smoothly dampening updates from strongly diverged peer policies to preserve optimization stability.
3. **When Sharing is Beneficial (Adaptive Temporal Decay Scheduling):** Through a piecewise adaptive decay schedule, APRES smoothly transitions agents from dense initial exploration to late-stage policy specialization while maintaining a residual knowledge-exchange baseline.

Unlike existing selective sharing approaches that rely on parameter-heavy graph neural networks, centralized attention mechanisms, or explicit communication networks, APRES performs experience selection and weighting strictly using quantities already computed during decentralized actor-critic optimization. Consequently, APRES preserves the decentralized execution paradigm,

scalability, and minimal computational overhead of standard actor-critic algorithms while delivering significantly higher result, sample efficiency and policy stability. Extending APRES to heterogeneous multi-agent settings is also an important direction for improving its generality. In addition, due to limited computational resources and the high training cost of decentralized MARL, the current evaluation was restricted to a single representative environment. Broader validation across diverse and large-scale domains such as autonomous driving, network resource allocation, and smart grid systems remains an important avenue for future study.

Data Availability

The Robotic Warehouse (RWARE) environment used during the current study is publicly available in the RWARE repository, at https://github.com/semitable/robotic-warehouse.

7. References

1. Esmaili, A. & Roayaei, M. UAV-Based warehouse management using multi-agent RL: Applications, challenges, and solutions. in *Applications of Machine Learning in UAV Networks* 263–306 (IGI Global Scientific Publishing, 2024).
2. Zhou, W. *et al.* Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic. *Auton. Intell. Syst.* **2**, 5 (2022).
3. Poudel, S. & Moh, S. Maml-integrated multi-agent reinforcement learning for adaptive coalition-based uav coordination in disaster scenarios. *Internet Things* 101930 (2026).
4. Xiong, Q., Chi, D., Luan, Q. & Shen, J. Coordination in Traffic Signal Control: A Mechanism-Oriented Survey of Multi-Agent Learning Approaches. *IET Intell. Transp. Syst.* **20**, e70296 (2026).
5. Zhao, C., Zhao, J., Du, Z. & Lu, K. ACUTE: Attentional communication framework for multi-agent reinforcement learning in partially communicable scenarios. *Electronics* **11**, 4204 (2022).
6. Ibrahim, A. M., Yau, K.-L. A., Chong, Y.-W. & Wu, C. Applications of multi-agent deep reinforcement learning: Models and algorithms. *Appl. Sci.* **11**, 10870 (2021).
7. Nekoei, H. *et al.* Dealing with non-stationarity in decentralized cooperative multi-agent deep reinforcement learning via multi-timescale learning. in *Conference on Lifelong Learning Agents* 376–398 (PMLR, 2023).
8. Mahjoub, O. *et al.* Sable: a performant, efficient and scalable sequence model for marl. *ArXiv Prepr. ArXiv241001706* (2024).
9. Lowe, R. *et al.* Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **30**, (2017).
10. Yu, C. *et al.* The surprising effectiveness of ppo in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **35**, 24611–24624 (2022).
11. Sunehag, P. *et al.* Value-decomposition networks for cooperative multi-agent learning. *ArXiv Prepr. ArXiv170605296* (2017).
12. Rashid, T. *et al.* Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **21**, 1–51 (2020).
13. Ba, Y. *et al.* Cautiously-optimistic knowledge sharing for cooperative multi-agent reinforcement learning. in *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 38 17299–17307 (2024).
14. Chen, J., Feng, J., Gao, W. & Wei, K. Decentralized natural policy gradient with variance reduction for collaborative multi-agent reinforcement learning. *J. Mach. Learn. Res.* **25**, 1–49 (2024).

15. Chu, X. & Ye, H. Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning. *ArXiv Prepr. ArXiv171000336* (2017).
16. Ding, Z., Huang, T. & Lu, Z. Learning individually inferred communication for multi-agent cooperation. *Adv. Neural Inf. Process. Syst.* **33**, 22069–22079 (2020).
17. Ferreira, T., Clua, E. & Kohwalter, T. C. Centralized critic per knowledge for cooperative multi-agent game environments. in *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)* 39–48 (IEEE, 2021).
18. Gupta, J. K., Egorov, M. & Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. in *International conference on autonomous agents and multiagent systems* 66–83 (Springer, 2017).
19. Huang, J., Huang, Y., Zhao, Y., Meng, W. & Xu, J. CoCoL: A Communication Efficient Decentralized Collaborative Learning Method for Multi-Robot Systems. in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 6219–6226 (IEEE, 2025).
20. Jiang, J., Dun, C., Huang, T. & Lu, Z. Graph convolutional reinforcement learning. *ArXiv Prepr. ArXiv181009202* (2018).
21. Kim, W. & Sung, Y. Parameter sharing with network pruning for scalable multi-agent deep reinforcement learning. *ArXiv Prepr. ArXiv230300912* (2023).
22. Li, D., Lou, N., Zhang, B., Xu, Z. & Fan, G. Adaptive parameter sharing for multi-agent reinforcement learning. in *ICASSP 2024-2024 IEEE International conference on acoustics, speech and signal processing (ICASSP)* 6035–6039 (IEEE, 2024).
23. Xu, J., Zhu, S., Soh, Y. C. & Xie, L. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. in *2015 54th IEEE Conference on Decision and Control (CDC)* 2055–2060 (IEEE, 2015).
24. Yang, N., Ding, B., Shi, P. & Feng, D. Improving scalability of multi-agent reinforcement learning with parameters sharing. in *2022 IEEE International Conference on Joint Cloud Computing (JCC)* 37–42 (IEEE, 2022).
25. Aslam, S., Khan, D. & Chang, K. The shared experience actor–critic (SEAC) approach for allocating radio resources and mitigating resource collisions in 5G-NR-V2X mode 2 under aperiodic traffic conditions. *Sensors* **24**, 6769 (2024).
26. Kilinc, O. & Montana, G. Multi-agent deep reinforcement learning with extremely noisy observations. *ArXiv Prepr. ArXiv181200922* (2018).
27. Sukhbaatar, S., Fergus, R., & others. Learning multiagent communication with backpropagation. *Adv. Neural Inf. Process. Syst.* **29**, (2016).
28. Suttle, W. *et al.* A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning. *IFAC-Pap.* **53**, 1549–1554 (2020).
29. Christianos, F., Schäfer, L. & Albrecht, S. Shared experience actor-critic for multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **33**, 10707–10717 (2020).
30. Barth-Maron, G. *et al.* Distributed distributional deterministic policy gradients. *ArXiv Prepr. ArXiv180408617* (2018).
31. Dahal, M. & Vaezi, M. Selective experience sharing in reinforcement learning enhances interference management. *IEEE Commun. Lett.* **29**, 615–619 (2025).
32. Gerstgrasser, M., Danino, T. & Keren, S. Selectively sharing experiences improves multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **36**, 59543–59565 (2023).

33. Louati, F., Ktata, F. B. & Amous, I. Boosting multi-agent reinforcement learning with pruned prioritized experience replay and collaborative learning. Case study: intrusion detection system. *J. Ambient Intell. Humaniz. Comput.* **17**, 851–862 (2026).
34. Souza, L. O., de Oliveira Ramos, G. & Ralha, C. G. Experience sharing between cooperative reinforcement learning agents. in *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)* 963–970 (IEEE, 2019).
35. Tian, Q., Kuang, K., Liu, F. & Wang, B. Learning from good trajectories in offline multi-agent reinforcement learning. in *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 37 11672–11680 (2023).
36. Zhu, H. *et al.* Sharing control knowledge among heterogeneous intersections: a distributed arterial traffic signal coordination method using multi-agent reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* **26**, 2760–2776 (2025).
37. Das, A. *et al.* Tarmac: Targeted multi-agent communication. in *International Conference on machine learning* 1538–1546 (PMLR, 2019).
38. Liu, Y.-C. *et al.* Who2com: Collaborative perception via learnable handshake communication. in *2020 IEEE International Conference on Robotics and Automation (ICRA)* 6876–6883 (IEEE, 2020).
39. Singh, A., Jain, T. & Sukhbaatar, S. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *ArXiv Prepr. ArXiv181209755* (2018).
40. Kasai, T., Tenmoto, H. & Kamiya, A. Learning of communication codes in multi-agent reinforcement learning problem. in *2008 IEEE conference on soft computing in industrial applications* 1–6 (IEEE, 2008).
41. Varshavskaya, P., Kaelbling, L. P. & Rus, D. Efficient distributed reinforcement learning through agreement. in *Distributed Autonomous Robotic Systems 8* 367–378 (Springer, 2009).
42. Rangwala, M. & Williams, R. Learning multi-agent communication through structured attentive reasoning. *Adv. Neural Inf. Process. Syst.* **33**, 10088–10098 (2020).
43. Chu, T., Chinchali, S. & Katti, S. Multi-agent reinforcement learning for networked system control. *ArXiv Prepr. ArXiv200401339* (2020).
44. Gao, Y., Wang, W. & Yu, N. Consensus multi-agent reinforcement learning for volt-var control in power distribution networks. *IEEE Trans. Smart Grid* **12**, 3594–3604 (2021).
45. Jiang, J. & Lu, Z. Learning attentional communication for multi-agent cooperation. *Adv. Neural Inf. Process. Syst.* **31**, (2018).
46. Wang, Z., Yang, K., Li, L., Lu, Y. & Tao, Y. Traffic signal priority control based on shared experience multi-agent deep reinforcement learning. *IET Intell. Transp. Syst.* **17**, 1363–1379 (2023).
47. Zhang, K., Yang, Z. & Basar, T. Networked multi-agent reinforcement learning in continuous spaces. in *2018 IEEE conference on decision and control (CDC)* 2771–2776 (IEEE, 2018).
48. Zhang, K., Yang, Z., Liu, H., Zhang, T. & Basar, T. Fully decentralized multi-agent reinforcement learning with networked agents. in *International conference on machine learning* 5872–5881 (PMLR, 2018).
49. Yi, Y., Li, G., Wang, Y. & Lu, Z. Learning to share in networked multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **35**, 15119–15131 (2022).
50. Wang, J. *et al.* Dyps: Dynamic parameter sharing in multi-agent reinforcement learning for spatio-temporal resource allocation. in *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining* 3128–3139 (2024).

51. Christianos, F., Papoudakis, G., Rahman, M. A. & Albrecht, S. V. Scaling multi-agent reinforcement learning with selective parameter sharing. in *International conference on machine learning* 1989–1998 (PMLR, 2021).
52. Inokuchi, K. & Sugawara, T. Multi-Agent Reinforcement Learning with Clustering and Experience Sharing. *Procedia Comput. Sci.* **246**, 1329–1337 (2024).
53. Daniel, J. *et al.* Multi-agent reinforcement learning with selective state-space models. *ArXiv Prepr. ArXiv241019382* (2024).
54. Du, H., Thudumu, S., Nguyen, H., Vasa, R. & Mouzakis, K. Contextual knowledge sharing in multi-agent reinforcement learning with decentralized communication and coordination. *ArXiv Prepr. ArXiv250115695* (2025).
55. Wu, J., Wei, S., Feng, X. & Yu, H. EntShare: Adaptive entropy balancing for multi-agent reinforcement learning with selective knowledge sharing. *Future Gener. Comput. Syst.* 108663 (2026).
56. Jiang, H., Ding, Z. & Lu, Z. Settling decentralized multi-agent coordinated exploration by novelty sharing. in *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 38 17444–17452 (2024).
57. Kota, M. S., Fan, T., Poojary, H., Teklehaimanot, N. & Balwani, A. NOVELTY-GATED EXPERIENCE SHARING FOR MULTI-AGENT REINFORCEMENT LEARNING.
58. Zhu, C., Leung, H.-F., Hu, S. & Cai, Y. A q-values sharing framework for multi-agent reinforcement learning under budget constraint. *ACM Trans. Auton. Adapt. Syst. TAAS* **15**, 1–28 (2021).
59. Amato, C., Chowdhary, G., Geramifard, A., Üre, N. K. & Kochenderfer, M. J. Decentralized control of partially observable Markov decision processes. in *52nd IEEE Conference on Decision and Control* 2398–2405 (IEEE, 2013).
60. Amato, C. & Oliehoek, F. Scalable planning and learning for multiagent POMDPs. in *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 29 (2015).
61. Hansen, E. A., Bernstein, D. S. & Zilberstein, S. Dynamic programming for partially observable stochastic games. in *AAAI* vol. 4 709–715 (2004).
62. Precup, D., Sutton, R. S. & Singh, S. Eligibility traces for off-policy policy evaluation. (2000).
63. Papoudakis, G., Christianos, F., Schäfer, L. & Albrecht, S. V. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *ArXiv Prepr. ArXiv200607869* (2020).
64. Sun, Y. *et al.* Retentive network: A successor to transformer for large language models. *ArXiv Prepr. ArXiv230708621* (2023).
65. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N. & Whiteson, S. Counterfactual multi-agent policy gradients. in *Proceedings of the AAAI conference on artificial intelligence* vol. 32 (2018).
66. De Witt, C. S. *et al.* Is independent learning all you need in the starcraft multi-agent challenge? *ArXiv Prepr. ArXiv201109533* (2020).

Legend

1. Figure legends

Figure 1. Overview of the decentralized training and decentralized execution (DTDE) framework. Each agent independently interacts with the shared environment using its own actor–critic networks. The actor selects actions based on the agent's observations, while the critic estimates value-related quantities used to update the policy. Agents learn independently without a centralized critic.

Figure 2. The RWARE-small warehouse environment with four agents. The grid-based environment represents robots navigating warehouse aisles to collect and deliver requested shelves. Agents are shown in yellow when not carrying a shelf and in red when carrying one. Purple cells represent shelf-storage locations, green indicates a requested shelf, and black cells indicate goal locations.

Figure 3. Ablation study of the APRES experience-selection (WHICH) component on RWARE. Learning curves compare SEAC with APRES configurations that vary the selection of shared experiences. Panels (a–d) correspond to RWARE-2ag-tiny, RWARE-4ag-tiny, RWARE-8ag-tiny, and RWARE-16ag-tiny, respectively. The horizontal axis represents environment steps, and the vertical axis represents episode return. The comparison examines how selecting shared experiences affects learning performance across different numbers of agents.

Figure 4. Ablation study of the APRES policy-aware experience-weighting (WHOM) component on RWARE. Learning curves compare SEAC with APRES configurations that vary how the contribution of experiences received from other agents is weighted. Panels (a–d) correspond to RWARE-2ag-tiny, RWARE-4ag-tiny, RWARE-8ag-tiny, and RWARE-16ag-tiny, respectively. The horizontal axis represents environment steps, and the vertical axis represents episode return.

Figure 5. Ablation study of the APRES adaptive temporal-decay (WHEN) component on RWARE. Learning curves compare SEAC with APRES configurations that vary the temporal contribution of shared experiences during training. Panels (a–d) correspond to RWARE-2ag-tiny, RWARE-4ag-tiny, RWARE-8ag-tiny, and RWARE-16ag-tiny, respectively. The horizontal axis represents environment steps, and the vertical axis represents episode return.

Figure 6. Training performance of APRES and SEAC on RWARE-2ag-tiny. Episode return is plotted against environment steps. The curves compare the learning performance of APRES and SEAC in the two-agent tiny-warehouse configuration.

Figure 7. Training performance of APRES and SEAC on RWARE-4ag-tiny. Episode return is plotted against environment steps. The curves compare the learning performance of APRES and SEAC in the four-agent tiny-warehouse configuration.

Figure 8. Training performance of APRES and SEAC on RWARE-8ag-tiny. Episode return is plotted against environment steps. The curves compare the learning performance of APRES and SEAC in the eight-agent tiny-warehouse configuration.

Figure 9. Training performance of APRES and SEAC on RWARE-16ag-tiny. Episode return is plotted against environment steps. The curves compare the learning performance of APRES and SEAC in the sixteen-agent tiny-warehouse configuration.

Figure 10. Training performance of APRES and SEAC on RWARE-8ag-small. Episode return is plotted against environment steps. The curves compare the learning performance of APRES and SEAC in the eight-agent small-warehouse configuration.

Figure 11. Training performance of APRES and SEAC on RWARE-16ag-small. Episode return is plotted against environment steps. The curves compare the learning performance of APRES and SEAC in the sixteen-agent small-warehouse configuration.

2. Table legends

Table 1. Notation used in the APRES framework. The table summarizes the symbols and their meanings used to describe the agents' actor–critic networks, observations, actions, rewards, returns, advantages, trajectories, importance-sampling ratios, selected experiences, and adaptive temporal-decay coefficient.

Table 2. Maximum episode returns of 11 multi-agent reinforcement learning algorithms on RWARE. Results are reported for RWARE-2ag-tiny and RWARE-4ag-tiny. Algorithms are grouped according to their training and execution framework: centralized training with centralized execution (CTCE), centralized training with decentralized execution (CTDE), and decentralized training with decentralized execution (DTDE). Results for methods other than APRES are taken from the cited benchmark study, whereas APRES results are obtained in this study. Bold values indicate the highest return for each task.

Table 3. Training and evaluation episode returns of SEAC and APRES across RWARE configurations. The table reports the training and evaluation returns for the two algorithms on tiny and small warehouse environments with different numbers of agents. Bold values indicate the highest score for each task and evaluation category.

Author Contributions

A.E. conceived the study, developed the proposed APRES framework, implemented the methodology, conducted the experiments, analyzed the results, and prepared the initial manuscript. M.R. contributed to the research conceptualization, supervised the study, provided critical feedback on the methodology and interpretation of the results, and revised the manuscript. Both authors reviewed and approved the final version of the manuscript and agree to be accountable for their respective contributions and for the integrity of the work.

Funding

This work is based upon research funded by Iran National Science Foundation (INSF) under project No.4031031

Additional Information

- **Competing interests**

The authors declare no competing interests.