TIGER-MARL: Enhancing Multi-Agent Reinforcement Learning with Temporal Information through Graph-based Embeddings and Representations

Nikunj Gupta[1*] NIKUNJ@USC.EDU
Ludwika Twardecka[1*] TWARDECK@USC.EDU
James Zachary Hare[2] JAMES.Z.HARE.CIV@ARMY.MIL
Jesse Milzman[2] JESSE.M.MILZMAN.CIV@ARMY.MIL
Rajgopal Kannan[2] RAJGOPAL.KANNAN.CIV@ARMY.MIL
Viktor Prasanna[1] PRASANNA@USC.EDU
[1] *University of Southern California*
[2] *DEVCOM Army Research Office*

Abstract

In this paper, we propose capturing and utilizing *Temporal Information through Graph-based Embeddings and Representations* or **TIGER** to enhance multi-agent reinforcement learning (MARL). We explicitly model how inter-agent coordination structures evolve over time. While most MARL approaches rely on static or per-step relational graphs, they overlook the temporal evolution of interactions that naturally arise as agents adapt, move, or reorganize cooperation strategies. Capturing such evolving dependencies is key to achieving robust and adaptive coordination. To this end, TIGER constructs dynamic temporal graphs of MARL agents, connecting their current and historical interactions. It then employs a temporal attention-based encoder to aggregate information across these structural and temporal neighborhoods, yielding time-aware agent embeddings that guide cooperative policy learning. Through extensive experiments on two coordination-intensive benchmarks, we show that TIGER consistently outperforms diverse value-decomposition and graph-based MARL baselines in task performance and sample efficiency. Furthermore, we conduct comprehensive ablation studies to isolate the impact of key design parameters in TIGER, revealing how structural and temporal factors can jointly shape effective policy learning in MARL. All codes can be found here: https://github.com/Nikunj-Gupta/tiger-marl.

Keywords: Multi-agent reinforcement learning, graph neural networks, cooperative decision-making

1. Introduction

Multi-Agent Reinforcement Learning (MARL) has emerged as a powerful paradigm across diverse domains such as autonomous driving (Shalev-Shwartz et al., 2016; Palanisamy, 2020), resource management (Guestrin et al., 2001), and swarm robotics (Orr and Dutta, 2023; Rizk et al., 2019). Its growing success in cooperative artificial intelligence reflects its ability to model shared decision-making among agents that collectively optimize a long-term objective in complex, interactive systems. As the field continues to advance, capturing inter-agent interactions has become central to achieving adaptive and effective coordination, whether through explicit communication, implicit information exchange, or integrating neighboring agents' signals into local decision-making.

[*] Equal contribution.

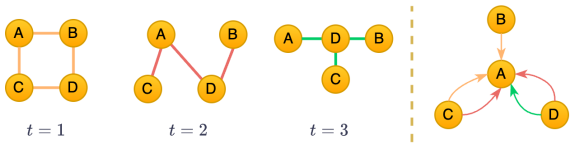


Figure 1: **Evolving coordination in MARL.** Temporal interaction patterns can reveal useful relational cues to enhance learning.

Irrespective of the application domain, agents in cooperative tasks must effectively reason about their interactions to agree on the best strategy at a given moment. To achieve that, both present and past context play an equally critical role. Fig. 1 illustrates this, showing that as agents progress through time, their connectivity and influence patterns shift. Restricting the model to current inter-agent connections limits its ability to capture how coordination structures evolve as agents adapt their roles and strategies in response to environmental dynamics. Modeling evolving relational dependencies can provide temporal cues that help agents anticipate, adapt, and collaborate more effectively. This becomes especially valuable in partially observable settings, where agents must infer latent coordination structures from limited observations.

Most MARL approaches either ignore inter-agent relationships or rely on static coordination graphs that capture dependencies only at a single timestep. To overcome this, we draw on advances in graph representation learning. Graph Neural Networks (GNNs) (Veličković et al., 2018b) have been widely used in MARL to model structural dependencies and integrate agent information (Liu et al., 2025). Extending to temporal dependencies, Temporal GNNs model how node and edge structures evolve over time, providing a natural way to represent dynamic coordination patterns. These have shown strong performance in domains such as social networks, financial forecasting, and recommendation systems (Feng et al., 2024; Barros et al., 2021). To the best of our knowledge, however, their application to MARL remains largely unexplored.

In this paper, we propose TIGER, a principled approach for modeling evolving agent interaction patterns in MARL. TIGER introduces a temporal graph construction mechanism that defines each agent's structural and temporal neighborhoods by explicitly linking current and historical interactions. It then employs an attention-based temporal graph encoder that aggregates information across these neighborhoods to produce time-aware agent embeddings. Through temporal attention, the encoder adaptively balances recent and past interactions, allowing agents to focus on the most relevant dynamics as coordination patterns evolve. TIGER serves as an expressive temporal reasoning module within MARL, allowing agents to anticipate shifting dependencies and act coherently under partial observability. We demonstrate that this approach substantially improves task performance and sample efficiency over strong value-factorization and graph-based MARL baselines across two diverse coordination-intensive benchmarks. Finally, we conduct detailed ablations to analyze how temporal depth and static connectivity influence coordination, offering insights into the trade-offs that govern effective static-temporal reasoning in MARL.

2. Related Works

MARL. Foundational value-factorization methods such as VDN (Sunehag et al., 2018) and QMIX (Rashid et al., 2018) decompose a global action-value function into individual utilities. Extensions including QTRAN (Son et al., 2019) and QPLEX (Wang et al., 2021) refined this decomposition to improve representational expressiveness. However, these approaches assume implicit or static coordination, where cooperation arises indirectly through reward structure rather than explicit modeling of agent dependencies. To overcome this limitation, graph-based MARL frameworks repre-

sent agents as nodes and their interactions as edges, enabling structured message passing (Li et al., 2021; Böhmer et al., 2020; Duan et al., 2024a,b; Gupta et al., 2025; Kok and Vlassis, 2006). DCG (Böhmer et al., 2020) introduced pairwise factorization on fixed graphs, DICG (Li et al., 2021) learned latent attention-based edges, and recent works such as CASEC (Wang et al., 2022) and GACG (Duan et al., 2024a) improved scalability and hierarchical reasoning. Mixer-style variants such as GraphMix (Naderializadeh et al., 2020) and QGNN (Kortvelesy and Prorok, 2022) combine graph convolution with value mixing. These approaches, however, employ static or per-step relational graphs, limiting adaptability in dynamic settings where coordination structures evolve over time. LTSCG (Duan et al., 2024b) is a closely related work that leverages historical trajectories to infer latent temporal graphs. TIGER, however, models temporal dependencies directly within the graph itself, jointly reasoning over structure and time.

Graph Representation Learning. GNNs have emerged as a powerful framework for learning over relational data by enabling message passing across structured entities (Wu et al., 2020). Works such as Graph Attention Networks (Veličković et al., 2018b) and Graph Convolutional Networks (Kipf, 2016), have been widely used to model local dependencies, assign adaptive edge weights, and capture higher-order relationships in dynamic systems. Temporal GNNs jointly model changes in both node attributes and edge structures over time. Notable architectures, such as TGAT (Xu et al., 2020), introduce continuous-time encoding and memory modules to efficiently represent long-term dependencies. Our work leverages these advances to design a temporal graph learning module that dynamically captures evolving coordination structures to enhance multi-agent policy learning.

3. Preliminaries

MARL. We formalize the problem as a decentralized partially observable Markov Decision Process (Oliehoek et al., 2016), defined by the tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{D}}, \mathcal{P}, \mathcal{R}, \{\mathcal{O}_i\}_{i\in\mathcal{D}}, \Omega, \gamma \rangle$, where $\mathcal{D}$ denotes the set of N agents. At timestep t, the environment is in a global state $s_t \in \mathcal{S}$, and each agent $i \in \mathcal{D}$ selects an action $a_{t,i} \in \mathcal{A}_i$, forming the joint action $\mathbf{a}_t = \langle a_{t,1}, \ldots, a_{t,N} \rangle$. The transition function $\mathcal{P}(s_{t+1}|s_t, \mathbf{a}_t)$ defines the probability of reaching s_{t+1}, and all agents share a global reward $\mathcal{R}(s_t, \mathbf{a}_t) \in \mathbb{R}$ that quantifies the cooperative objective. Each agent receives a local observation $o_{t,i} \in \mathcal{O}_i$ determined by the joint observation function $\Omega(s_t, \mathbf{a}_t) = \langle o_{t,1}, \ldots, o_{t,N} \rangle$. Since agents operate under partial observability, agent i maintains a local action-observation history $\tau_{t,i} = \{(o_{1,i}, a_{1,i}), \ldots, (o_{t,i}, a_{t-1,i})\}$ and acts according to its local policy $\pi_i(a_{t,i}|\tau_{t,i})$. The joint policy is given by $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_N \rangle$. The quality of a joint policy $\boldsymbol{\pi}$ is measured by the expected discounted return: $J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \mathbf{a}_t)\right]$, where $\gamma \in [0, 1)$ is the discount factor. The joint action-value function can be defined as: $Q^{\boldsymbol{\pi}}(s_t, \mathbf{a}_t) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{c=0}^{\infty} \gamma^c \mathcal{R}(s_{t+c}, \mathbf{a}_{t+c}) \mid s_t, \mathbf{a}_t]$. The goal in MARL is to find the optimal joint policy $\boldsymbol{\pi}^* = \langle \pi_i^* \rangle_{i\in\mathcal{D}}$ that maximizes this value function: $Q^*(s_t, \mathbf{a}_t) = \max_{\boldsymbol{\pi}} Q^{\boldsymbol{\pi}}(s_t, \mathbf{a}_t)$. Under the centralized training with decentralized execution (CTDE) paradigm (Amato, 2024), agents exploit global state and joint action information during training to learn coordinated policies, while acting independently using only local trajectories at execution. A common way to realize CTDE is by decomposing the joint value as

$$Q_{\text{tot}} = f_\psi\big(Q_1(\tau_{t,1}, a_{t,1}), \ldots, Q_N(\tau_{t,N}, a_{t,N})\big), \tag{1}$$

with f_ψ a learnable mixing function parameterized by ψ. Each agent acts greedily with respect to its local utility: $a_{t,i} = \arg\max_{a_i} Q_i(\tau_{t,i}, a_i)$.

Graph Attention Networks (GAT). GAT (Veličković et al., 2018a) provides a flexible way to model relational dependencies through attention-based message passing. In MARL, agents can be represented as nodes and their interactions as edges in a graph $G = (V, E)$, where $V = \{1, \ldots, N\}$ corresponds to the set of agents and $E \subseteq V \times V$ denotes the set of interaction links. Each node i has a feature or embedding vector $h_i \in \mathbb{R}^d$, which aggregates information from its neighbors $\mathcal{N}(i)$. For each neighboring agent $j \in \mathcal{N}(i)$, GAT computes an attention weight α_{ij} that quantifies the importance of agent j's information to agent i:

$$\alpha_{ij} = \frac{\exp\big(\text{LeakyReLU}\big(a^\top [Wh_i \| Wh_j]\big)\big)}{\sum_{k \in \mathcal{N}(i)} \exp\big(\text{LeakyReLU}\big(a^\top [Wh_i \| Wh_k]\big)\big)}, \tag{2}$$

where $W \in \mathbb{R}^{d \times d'}$ and $a \in \mathbb{R}^{2d'}$ are learnable parameters. The updated embedding for agent i is then computed as $h'_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} W h_j$.

Temporal Graph Attention Networks (TGAT). TGAT (Xu et al., 2020) extends to evolving graphs by introducing temporal attention, allowing attention weights to depend jointly on structural and temporal context. A temporal graph is defined as $G = \langle V, E, T \rangle$, where nodes V represent agents, edges $E = \{(u, v, t)\}$ are time-stamped interactions, and T is the continuous time domain. Each node v_i maintains a feature vector $x_i \in \mathbb{R}^{d_0}$ and a time-dependent embedding $h_i(t)$. TGAT constructs an entity-temporal feature matrix for each node v_i at time t:

$$Z_i(t) = [\, h_i(t) \| \phi(0);\ (h_j(t_j) \| \phi(t - t_j))_{v_j \in \mathcal{N}(v_i, t)} \,]^\top, \tag{3}$$

where $\phi(\cdot) \in \mathbb{R}^{d_T}$ encodes relative timestamps. Linear projections of $Z_i(t)$ yield the query, key, and value matrices: $q_i = Z_{i,0} W_q,\ K_i = Z_{i,1:} W_K,\ V_i = Z_{i,1:} W_V$, with $W_q, W_K, W_V \in \mathbb{R}^{(d_0 + d_T) \times d_h}$. Temporal attention over neighbors is then computed as

$$\alpha_{ij}(t) = \frac{\exp(q_i^\top K_{ij})}{\sum_{v_k \in \mathcal{N}(v_i, t)} \exp(q_i^\top K_{ik})}, \tag{4}$$

Temporally weighted aggregation produces the updated embedding: $h_i(t) = \sum_{v_j \in \mathcal{N}(v_i, t)} \alpha_{ij}(t)\, V_{ij}$. This formulation allows TGAT to selectively emphasize temporally and structurally relevant interactions, yielding time-aware node embeddings well-suited for dynamic relational reasoning.

4. Methodology

This section presents the architecture of TIGER for MARL. We first construct dynamic temporal graphs to capture both static and temporal inter-agent relationships. We then apply temporal attention to aggregate information from these graphs, obtaining agent embeddings that reflect evolving coordination. Finally, we integrate these with MARL to improve their performance.

Dynamic Temporal Graph Construction. To capture both current and evolving coordination structures among agents, TIGER constructs a temporal graph at each timestep that combines three distinct types of connections: (i) *static neighbors* representing current interactions, (ii) *self-history* edges that link an agent to its own past states, and (iii) *neighbor-history* edges connecting it to the past states of its current neighbors. As illustrated in Fig. 2, for each timestep t, we begin with a fully connected graph $G_t = \langle V_t, E_t \rangle$ over the N agents, where $V_t = \{v_1^t, \ldots, v_N^t\}$ and

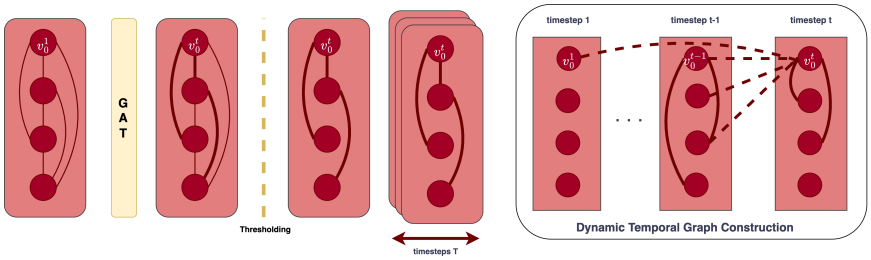


Figure 2: **Dynamic temporal graph construction in TIGER.** At each timestep t, GAT-pruned structural edges (*static neighbors*) are augmented with *self-* and *neighbor-history* links from the past $K_{\text{past\_self}}$ and $K_{\text{past\_nbr}}$ steps, capturing both current and evolving coordination among agents.

$|E_t| = N(N-1)/2$. A GAT layer assigns attention weights α_{ij}^t to all edges, indicating the relative importance of agent j to agent i at time t. To retain only the most significant interactions, we prune the graph by selecting the top $K_{\text{stat\_nbr}}$ fraction of edges with the highest attention scores, forming a sparse set of structural connections $E_t^{\text{static}} = K_{\text{stat\_nbr}} \times |E_t|$. The resulting static neighborhood of agent i at time t is thus: $\mathcal{N}_t(v_i^t) = \{\, v_j^t \mid e_{ij}^t \in E_t^{\text{static}} \,\}$. Next, we incorporate temporal information by augmenting this static graph with historical connections that capture how coordination evolves across time. Each agent is linked to its own past instances up to a fixed horizon $K_{\text{past\_self}}$: $\mathcal{N}^{\text{self}}(v_i^t) = \{\, v_i^{t-\Delta} \mid 1 \le \Delta \le K_{\text{past\_self}} \,\}$, and to the past states of its static neighbors up to $K_{\text{past\_nbr}}$ timesteps: $\mathcal{N}^{\text{nbr}}(v_i^t) = \{\, v_j^{t-\Delta} \mid v_j^t \in \mathcal{N}_t(v_i^t),\ 1 \le \Delta \le K_{\text{past\_nbr}} \,\}$. The complete temporal neighborhood of agent i at time t thus becomes: $\mathcal{N}(v_i^t) = \mathcal{N}_t(v_i^t) \cup \mathcal{N}^{\text{self}}(v_i^t) \cup \mathcal{N}^{\text{nbr}}(v_i^t)$. This formulation provides explicit control over how much historical information is integrated into the temporal graph through the parameters $K_{\text{stat\_nbr}}$, $K_{\text{past\_self}}$, and $K_{\text{past\_nbr}}$. Formally, we can define a *static-temporal trade-off* that constrains the size of an agent's temporal neighborhood as:

$$|\mathcal{N}(v_i^t)| = K_{\text{past\_self}} + \big(K_{\text{stat\_nbr}} \times |E_t|\big) + \big(K_{\text{stat\_nbr}} \times |E_t|\big) \times K_{\text{past\_nbr}}. \tag{5}$$

This governs how the temporal and structural components interact, determining how much of the agents' history is leveraged relative to current connectivity. In Section 6, we show resulting temporal graph enhances coordination performance. Through targeted ablations, we further examine how varying each component shapes learning dynamics. Analysis of computational complexity and graph size is provided in Appendix D.

Temporal Graph Encoding. Once the dynamic temporal graph is constructed, the next step is to derive time-aware agent feature representations. For this, TIGER includes a temporal encoder inspired by TGAT (Xu et al., 2020), given the proven effectiveness of attention mechanisms in TGNNs. This enables agents to dynamically incorporate information from recent and past interactions, focusing on the most relevant connections through temporal attention. For each agent v_i at time t, we construct an entity-temporal feature matrix (as introduced in Section 3) that concatenates its current embedding with the temporally encoded features of its neighbors: $Z_i(t) = \big[\, h_i^{(0)}(t) \| \phi(0);\ (h_j^{(0)}(\tau) \| \phi(t-\tau))_{v_j^\tau \in \mathcal{N}(v_i^t)} \big]^\top$, where $\phi : \mathbb{R}_{\ge 0} \to \mathbb{R}^{d_T}$ is a continuous-time encoder that maps relative timestamps to temporal embeddings. The rows of $Z_i(t)$ are linearly projected into a shared latent space using $W_q, W_K, W_V \in \mathbb{R}^{(d_0+d_T)\times d_h}$, producing the query, key, and value matrices: $q_i = Z_{i,0} W_q, \quad K_i = Z_{i,1:} W_K, \quad V_i = Z_{i,1:} W_V$. Temporal attention weights

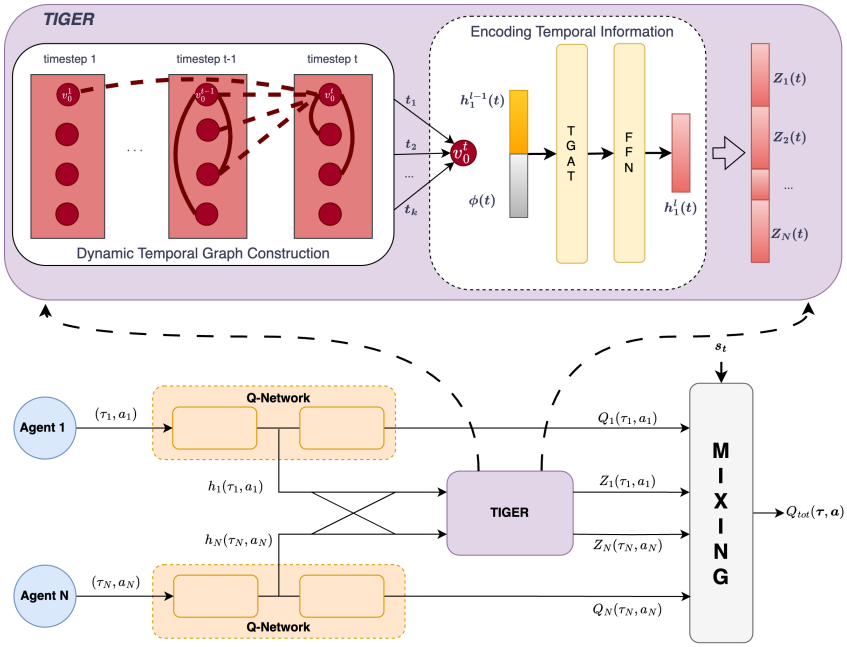


Figure 3: **Overview of TIGER.** At each timestep, TIGER constructs a dynamic temporal graph capturing both current and evolving inter-agent dependencies. Temporal attention then encodes these graphs into rich, weighted agent embeddings that integrate the most informative relational signals. These time-aware agent embeddings guide the learning of agent policies in MARL.

are then computed as $\alpha_{j\tau} = \frac{\exp(q_i^\top K_{j\tau})}{\sum_{v_k^\nu \in \mathcal{N}(v_i^t)} \exp(q_i^\top K_{k\nu})}$, assigning higher importance to temporally and structurally relevant neighbors. The agent aggregates the contextual information from its temporal neighborhood as $\tilde{h}_i(t) = \sum_{v_j^\tau \in \mathcal{N}(v_i^t)} \alpha_{j\tau} V_{j\tau}$. Finally, it fuses this temporal message with its current observation through a feed-forward transformation: $h_i(t) = \mathrm{ReLU}\big([\tilde{h}_i(t) \| o_{t,i}]W_0 + b_0\big)W_1 + b_1$, where W_0, W_1, b_0, and b_1 are learnable parameters. The resulting embeddings $h_i(t)$ encode both structural and temporal dependencies, enriching each agent's feature representation.

Integration with MARL. The updated embeddings $h_i(t)$ produced by the encoder serve as input features for each agent's value estimation in CTDE (see Section 3). By integrating information from the constructed temporal graph, these embeddings provide a latent representation that captures the evolving coordination context, enabling agents to make more informed decisions. Each agent i maintains a local utility head that estimates the value of taking action $a_{t,i}$ given its temporal embedding and the global state: $Q_i\big(s_t \| h_i(t), a_{t,i}\big)$. These local Q-values are combined through a mixing network f_ψ to form the joint action-value function,

$$Q_{\mathrm{tot}}(s_t, \mathbf{a}_t) = f_\psi\big(Q_1, \ldots, Q_N;\, s_t \| [h_1(t), \ldots, h_N(t)]\big), \tag{6}$$

which follows the standard value decomposition form of Eq. 1. This formulation allows temporal reasoning to enhance centralized value estimation during training without affecting decentralized execution at test time. During deployment, each agent selects its action independently using its updated embedding: $a_{t,i} = \arg\max_{a_i} Q_i(a_i \mid \tau_{t,i}, h_i(t))$. We demonstrate that this design improves task performance and sample efficiency in cooperative MARL tasks (Section 6).

5. Experimental Setup

Environments. We evaluate TIGER on two popular cooperative multi-agent tasks, Gather (Wang et al., 2022) and Tag (Lowe et al., 2017). These are coordination-intensive and test complementary aspects of temporal reasoning and adaptability in dynamic environments (see Appendix A).

- *Gather* extends Climb (Wei and Luke, 2016) to induce delayed cooperation under partial observability. Each agent selects among three actions $\{a_0, a_1, a_2\}$ for goals $\{g_1, g_2, g_3\}$, with one goal randomly chosen as optimal and known only to nearby agents. A team reward of $+10$ is given if all reach the optimal goal, $+5$ for a non-optimal one, and -5 if only a subset succeeds. This setup requires agents to infer and share latent information over time to coordinate effectively.
- *Tag* is a continuous-space particle-world task based on the Multi-Agent Particle Environment (Lowe et al., 2017), where a team of slower pursuers must collaboratively capture faster evaders in the presence of obstacles. Agents receive shared rewards upon successful captures, making cooperation essential for success. With constantly shifting spatial and relational agent dependencies, Tag challenges the need for real-time adaptation of coordination strategies.

Baselines. We compare TIGER against two representative algorithm families in cooperative MARL:

- *Mixers:* This family captures approaches that aggregate per-agent utilities through a joint value function without explicitly modeling coordination structure. We include VDN (Sunehag et al., 2018) as a minimal factorization baseline and QMIX (Rashid et al., 2018) as its monotonic generalization. We also evaluate against GraphMix (Naderializadeh et al., 2020) and QGNN (Kortvelesy and Prorok, 2022), which incorporate message passing among agents prior to value aggregation. Together, these baselines test whether TIGER complements or subsumes relational information implicitly captured by mixer-style architectures.
- *Graph-based methods:* This includes algorithms that construct and learn coordination graphs to represent inter-agent dependencies. We consider DICG (Li et al., 2021), which learns attention-weighted graphs; CASEC (Wang et al., 2022), which introduces sparsity through variance-based edge pruning; GACG (Duan et al., 2024a), which dynamically groups agents to capture higher-order relations; and LTSCG (Duan et al., 2024b), which infers temporal sparse graphs from historical observations. This diverse set enables us to isolate how explicit temporal graph updates in TIGER enhance MARL under evolving interaction patterns. More details are in Appendix C.

6. Experiments and Results

This section evaluates our approach on cooperative MARL tasks introduced in Section 5. Section 6.1 presents the main findings, highlighting the benefits of temporal relational reasoning through comparisons with strong mixer- and graph-based baselines. Section 6.2 then analyzes how specific design choices in TIGER influence these results through targeted ablations.

6.1. Main results

In this section, we demonstrate that incorporating temporal relational reasoning in MARL improves performance by comparing TIGER against several MARL baselines. We implement TIGER as a modular component that augments diverse MARL learners. Specifically, we integrate it into two baselines (QMIX and DICG) representing the two families identified in Section 5. This yields two TIGER variants: TIGER-MIX and TIGER-DICG. All methods are evaluated through two common metrics: (i) *task performance*, measured as test win rate in Gather and test return in Tag; and (ii)

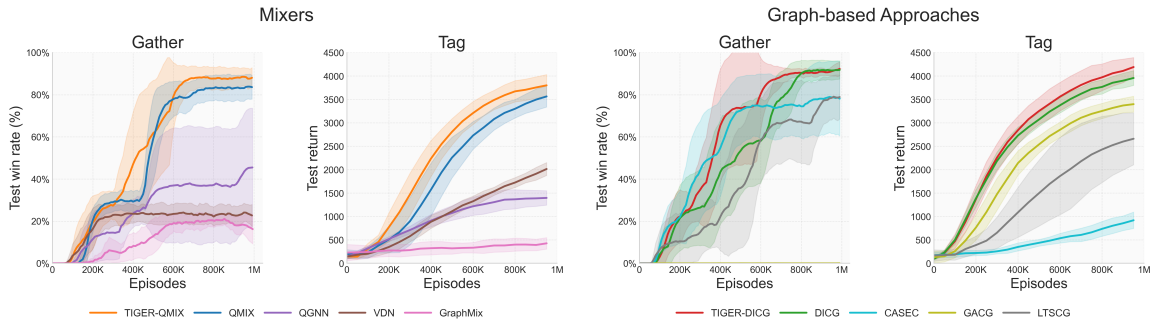
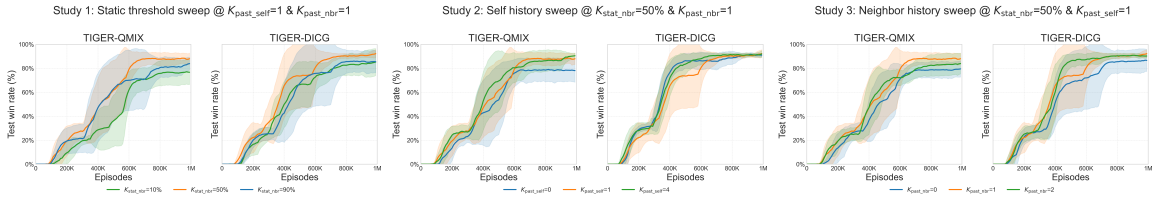


Figure 4: **Performance of TIGERon Gather and Tag.** Plots show task performance (win rate for Gather, return for Tag) over training steps, with standard deviation as shaded regions. (a) Mixer-based results showing faster convergence and stronger asymptotic performance with TIGER-MIX. (b) Graph-based results showing more stable learning and higher final returns with TIGER-DICG.

sample efficiency, defined as the number of steps required to reach convergence. Each result is averaged over five independent trials, with the mean and standard deviation reported. We instantiate $K_{\text{past\_nbr}} = 1$, $K_{\text{past\_self}} = 1$, and $K_{\text{static\_nbr}} = 50\%$ in Eq. 5, preserving a compact temporal neighborhood of size $N+1$. Further effects of varying these parameters are analyzed in Section 6.2.

Comparison with Mixers. We begin by evaluating TIGER-MIX, where TIGER augments QMIX's mixing network. Each agent's hidden state is embedded within TIGER's temporal graph, allowing information to propagate across recent interactions and yielding temporally enriched features for value decomposition. Across both Gather and Tag, TIGER-MIX consistently outperforms mixer baselines (Fig. 4a). The gains in performance and sample efficiency are most pronounced in Gather. Here, TIGER-MIX converges faster to a higher asymptotic win rate than all baselines, indicating the benefits of capturing evolving agent information. In Tag, TIGER-MIX maintains a clear margin in return and exhibits lower variance across runs. These results are particularly meaningful, as QMIX enforces a monotonic value factorization, GraphMix integrates attention-based graph mixing to model static dependencies, and QGNN employs multi-layer message passing to capture spatial relationships. Outperforming these baselines suggests that TIGER's temporal graph modeling captures dependencies that neither purely value-based nor static-graph mixers can express.

Comparison with Graph-based methods. We next evaluate TIGER-DICG, which extends DICG by enriching its coordination graphs with TIGER's temporal information modeling. Compared to DICG, CASEC, GACG, and LTSCG (Fig. 4b), TIGER-DICG consistently achieves higher task performance and better sample efficiency across both environments. On Gather, the improvement in win rate is substantial compared to CASEC, GACG, and LTSCG and modest in absolute value compared to DICG, but it occurs considerably earlier. The gains are more pronounced in *Tag*. TIGER-DICG converges faster to higher final returns. These results demonstrate that temporal reasoning complements graph-based MARL methods in cooperative settings. CASEC builds sparse coordination topologies based on payoff variance but lacks temporal continuity. GACG captures group-level dependencies while remaining static over time, and LTSCG constructs latent temporal graphs from historical trajectories, making its temporal reasoning indirect through inferred structures. Whereas TIGER-DICG directly integrates temporal edges into an evolving coordination graph.



(a) (b) (c)

Figure 5: **Ablation studies on TIGER.** We vary (a) static connectivity ($K_{\text{stat\_nbr}}$), (b) self-history depth ($K_{\text{past\_self}}$), and (c) neighbor-history depth ($K_{\text{past\_nbr}}$) to analyze how structural and temporal factors affect TIGER's performance.

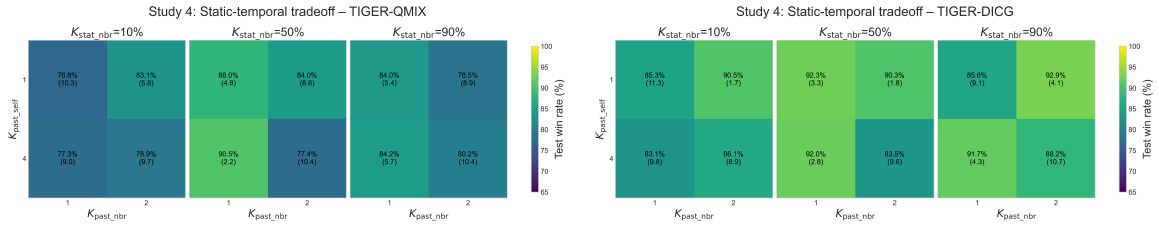
6.2. Ablations

In this section, we study how key design choices in TIGER ($K_{\text{stat\_nbr}}$, $K_{\text{past\_self}}$, and $K_{\text{past\_nbr}}$ in Eq. 5) affect its overall performance. While the main results validate TIGER across benchmarks, this section focuses on an in-depth analysis using Gather, which offers stable convergence and clear performance separability. Tag shows subtle but qualitatively similar trends (see Appendix B).

Study 1: Static neighbors $K_{\text{static\_nbr}}$ (Fig. 5a). Starting from the configuration introduced in Section 6.1 ($K_{\text{static\_nbr}} = 50\%$), we first reduce to a very sparse setting ($K_{\text{static\_nbr}} = 10\%$). Performance drops noticeably, indicating that without sufficient structural edges, TIGER's temporal graph has limited context to propagate information. This shows that TIGER's benefits rely on the added graph-based agent information integration rather than added network capacity in the module. Next, as we approach nearly complete graph ($K_{\text{static\_nbr}} = 90\%$), performance does not improve further or even drops slightly. We speculate that excessive or redundant edges introduce noisy or conflicting signals, and since $K_{\text{static\_nbr}}$ also influences temporal edge formation, oversaturation amplifies such effects. Overall, a moderate level of static connectivity gives the most effective performance.

Study 2: Self history $K_{\text{past\_self}}$ (Fig. 5b). To examine the influence of temporal recall over an agent's own past observations, we remove self-history edges ($K_{\text{past\_self}} = 0$). The performance of TIGER-MIX drops, showing that short-term self-history is vital, while TIGER-DICG remains relatively stable. We then increase the setting to a higher temporal depth. To maintain computational efficiency, we follow a logarithmic rule $K_{\text{past\_self}} = \lceil \ln(NT) \rceil$. This scales gracefully across environments, and ensures sublinear growth in self-history edges ($\mathcal{O}(NT)$). It yields values of 4 for Gather and 7 for Tag (see Appendix D). In Gather, $K_{\text{past\_self}} = 4$ offers only marginal benefits. This suggests that short-term past states already encode most relevant information, or other parameters (current or past neighbors) dominate or compensate for the contribution of deeper self-history.

Study 3: Neighbor history $K_{\text{past\_nbr}}$ (Fig. 5c). Upon removing past-neighbor links ($K_{\text{past\_nbr}} = 0$), the performance drops noticeably across both TIGER variants. This confirms that recalling recent neighbor interactions is crucial in TIGER. When extended to two-hop temporal neighbors, the improvement becomes marginal or even slightly drops, suggesting that most of the useful relational information is already captured by a single temporal hop. Thus, further increases are likely to provide limited new information. Also, increasing temporal neighbor depth introduces additional computational cost, scaling as $\mathcal{O}(N^2T)$, which makes larger hops impractical for many applications. Overall, short-range neighbor recall (one to two hops) provides sufficient temporal context for effective coordination while keeping computation manageable.



(a) (b)

Figure 6: **Static-temporal tradeoff in TIGER.** Plots report the final win rate at 1M training steps and standard deviation across runs in parentheses. Each configuration jointly varies static and temporal parameters to understand how they interact in shaping coordination. **(a)** TIGER-MIX peaks at moderate static connectivity (50%) and shallow temporal depth (one-hop) **(b)** TIGER-DICG remains robust across most settings, with 50% static threshold offering the best balance.

Study 4: Static-temporal tradeoff (Fig. 6). Finally, we jointly vary static and temporal parameters to understand how they interact in shaping coordination. Overall, although win rates fluctuate within a narrow band (≈76-91%), we observe static and temporal components complementing each other. When structural connectivity is sparse, adding temporal links improves stability (low standard deviation) and performance (high win rate), effectively recovering missing relational cues. However, in dense graphs, deeper temporal recall provides little or even negative benefit, likely due to redundant signals blurring coordination. Moderate connectivity (50%) with shallow temporal depth (one-hop) consistently provides the best balance. Notably, TIGER-DICG remains stable across configurations, while TIGER-QMIX shows higher sensitivity; expected as QMIX does not model any interactions without TIGER unlike DICG.

In this section, we examined how key design parameters in TIGER influence its performance. The results show that TIGER's improvements arise primarily from temporal reasoning rather than added capacity (Studies 2, 3), with self and neighbor histories providing informative temporal context. At the same time, the interaction between adjustable parameters highlights the complementary role of static and temporal components in shaping coordination dynamics (Studies 1, 4).

7. Conclusions

This work introduced TIGER, a temporal graph learning framework that enhances cooperative MARL by explicitly modeling how inter-agent relationships evolve over time. By constructing dynamic temporal graphs and encoding them through temporal attention, TIGER produces time-aware agent embeddings that integrate both structural and temporal context. Our experiments demonstrate that this temporal reasoning substantially improves coordination efficiency, convergence speed, and overall task performance across diverse MARL paradigms, including value-decomposition and graph-based methods. Extensive ablations further reveal that lightweight temporal recall and balanced structural connectivity are sufficient for stable, high-performing coordination. Future work could explore deploying TIGER in various domains such as intelligent traffic management, connected autonomous vehicles, and distributed robotic teams, where agents must coordinate under dynamic and partially observable conditions.

References

Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2409.03052*, 2024.

Claudio DT Barros, Matheus RF Mendonça, Alex B Vieira, and Artur Ziviani. A survey on embedding dynamic graphs. *ACM Computing Surveys (CSUR)*, 55(1):1–37, 2021.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *International Conference on Machine Learning*, pages 980–991. PMLR, 2020.

Wei Duan, Jie Lu, and Junyu Xuan. Group-aware coordination graph for multi-agent reinforcement learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, 2024a. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/434. URL https://doi.org/10.24963/ijcai.2024/434.

Wei Duan, Jie Lu, and Junyu Xuan. Inferring latent temporal sparse coordination graph for multiagent reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024b.

ZhengZhao Feng, Rui Wang, TianXing Wang, Mingli Song, Sai Wu, and Shuibing He. A comprehensive survey of dynamic graph neural networks: Models, frameworks, benchmarks, experiments and challenges, 2024. URL https://arxiv.org/abs/2405.00476.

Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. *Advances in neural information processing systems*, 14, 2001.

Nikunj Gupta, James Zachary Hare, Rajgopal Kannan, and Viktor Prasanna. Deep meta coordination graphs for multi-agent reinforcement learning. *arXiv preprint arXiv:2502.04028*, 2025.

TN Kipf. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Jelle R Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of machine learning research*, 7, 2006.

Ryan Kortvelesy and Amanda Prorok. Qgnn: Value function factorisation with graph neural networks. *arXiv preprint arXiv:2205.13005*, 2022.

Sheng Li, Jayesh K. Gupta, Peter Morales, Ross Allen, and Mykel J. Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 764–772, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Yifan Liu, Dalei Wu, and Yu Liang. Survey on graph-based reinforcement learning for networked coordination and control. *Automation*, 6(4):65, 2025.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Navid Naderializadeh, Fan H Hung, Sean Soleyman, and Deepak Khosla. Graph convolutional value decomposition in multi-agent reinforcement learning. *arXiv preprint arXiv:2010.04740*, 2020.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

James Orr and Ayan Dutta. Multi-agent deep reinforcement learning for multi-robot applications: A survey. *Sensors*, 23(7):3625, 2023.

Praveen Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

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

Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–31, 2019.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

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

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085–2087, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=rJXMpikCZ.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018b.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. {QPLEX}: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Rcmk0xxIQV.

Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. Context-aware sparse deep coordination graphs. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=wQfgfb8VKTn.

Ermo Wei and Sean Luke. Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research*, 17(84):1–42, 2016.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs, 2020. URL https://arxiv.org/abs/2002.07962.

Appendix

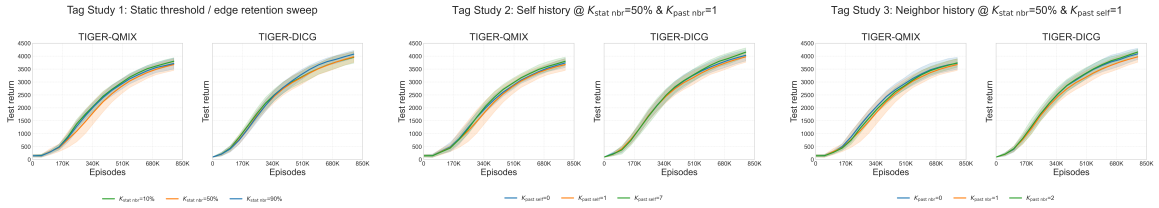
Appendix A. Additional details on the task settings

We elaborate on high-level environment overview from Section 5.

Gather (Wang et al., 2022). Gather is an extension of the classic Climb Game (Wei and Luke, 2016), where each agent chooses among three actions $A = \{a_0, a_1, a_2\}$. Action a_0 yields no immediate reward unless selected simultaneously by all agents, in which case it results in a high collective reward of $+10$. Actions a_1 and a_2 provide smaller individual rewards $(+5)$ independently. The task requires strong coordination among agents to consistently select the cooperative action. We adopt the precise setup defined by the Multi-Agent Coordination (MACO) benchmark (Wang et al., 2022) to facilitate comparison. Episodes last at most eight steps. Gather features a particularly sparse and delayed reward structure: agents must both identify a hidden target and synchronously occupy it before any payoff is received. This sparsity makes credit assignment challenging, as individual exploratory actions yield no immediate feedback, forcing agents to infer and align their teammates' intentions over multiple time steps in order to secure the collective $+10$ bonus.

Tag (Lowe et al., 2017). Tag is an environment based on the particle world scenario, in which a team of agents collaboratively chases multiple adversaries within a map containing randomly generated obstacles. The motion of each agent is discretized into five macroactions: *no-op*, move $\pm x$, or move $\pm y$. Episodes last up to 100 environment steps. Agents earn a global reward each time they successfully collide with an adversary. Specifically, every time an agent collides with the adversary, *all* agents instantly receive $+1$; simultaneous tags sum, and there are no penalties for failed attempts or wall impacts. Furthermore, adversaries have a speed advantage, and so successful captures typically require pursuers to assume complementary roles (for instance, blocking versus chasing), making the task a test of real-time coordination under dense yet still globally shared feedback. We used the experimental setup with ten agents pursuing three adversaries. In Tag, rewards are far more frequent but arise in a dynamic, partially observable setting: agents must implicitly divide roles - such as blocking escape routes versus direct pursuit - according to the evader's maneuvers and the obstacle layout. The dense feedback accelerates learning, yet optimal performance still hinges on adaptive cooperation and rapid role reassignment in response to adversary behavior.

Gather features a sparse and delayed reward structure, requiring agents to jointly identify and occupy a hidden target. This setup challenges coordination, as agents must infer each other's intentions without immediate feedback. Tag, however, provides dense rewards in a dynamic, partially observable environment, but still demands real-time cooperation as agents must implicitly assume complementary roles (e.g., blocking vs. chasing) based on adversary behavior and obstacles. These environments reflect distinct coordination demands; consistent gains across both demonstrate TIGER's effectiveness under varied credit assignment challenges (see Section 6).



(a) (b) (c)

Figure 7: **Ablation studies on TIGER.** (a) Static threshold sweep showing test win rate across neighbor ratios ($K_{\text{stat\_nbr}}$). (b) Self-history sweep varying temporal depth ($K_{\text{past\_self}}$). (c) Neighbor-history sweep varying neighbor temporal depth ($K_{\text{past\_nbr}}$)

Appendix B. Further Experiments and Ablations on Tag

We conduct ablation studies on the Tag environment following the same procedure as in Gather (see Section 6.2) to verify the consistency of TIGER's design choices under distinct coordination dynamics. Unlike Gather, which features delayed rewards and sparse interactions, Tag involves continuous spatial movement, dense encounters, and frequent feedback. We speculate that due to these differences in task structure, configuration, and metrics, the performance variations across TIGER's design parameters are less pronounced than in Gather, and definitive conclusions are therefore harder to establish. We also note here that sharp trends are not expected in this setting. The three parameters co-influence a complex, nonlinear learning process involving stochastic agent interactions, attention-driven aggregation, and partial observability. As a result, small changes in temporal depth or connectivity can lead to competing effects, enhanced information flow versus increased variance, which often average out across runs. Nonetheless, some general patterns emerge across both mixers and graph-based TIGER-variants in Tag:

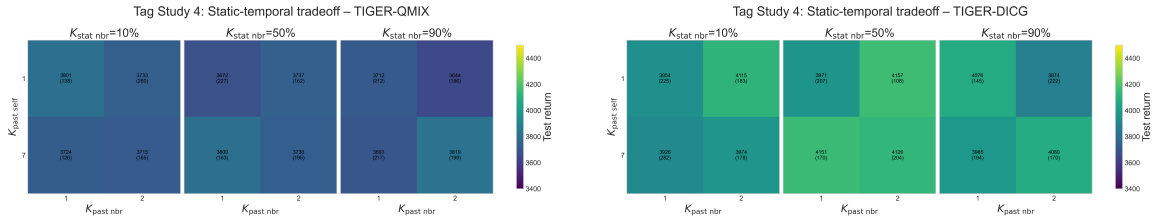
(i) performance remains relatively stable across most configurations, with test returns ranging between ∼3600-3800 for TIGER-MIX and ∼3900-4200 for TIGER-DICG.

(ii) moderate static connectivity ($K_{\text{stat\_nbr}}$=50%) consistently provides the best balance, yielding the highest or near-highest returns in both TIGER-MIX and TIGER-DICG.

(iii) deeper temporal recall generally improves coordination, especially in TIGER-DICG: returns rise for $K_{\text{past\_self}}$=7 and $K_{\text{past\_nbr}}$=2, suggesting that richer temporal context enhances adaptation in continuous multi-agent motion.

Below we further analyze the effects across each design factor, with results shown in Figs. 7 and 8. For this, we focus on the final absolute test returns to draw concise qualitative conclusions from the Tag ablation studies.

Static threshold $K_{\text{stat\_nbr}}$ (Fig. 7a). As in Gather, we begin by reducing static connectivity to a sparse setting ($K_{\text{stat\_nbr}}$=10%). For TIGER-MIX, this configuration leads to a mild improvement, suggesting that in Tag's dense feedback regime, fewer structural links help highlight the more relevant interactions. Conversely, TIGER-DICG attains improved results under denser connectivity ($K_{\text{stat\_nbr}}$=90%), indicating that it benefits from richer static relational structure for coordination. The intermediate $K_{\text{stat\_nbr}}$=50% configuration yields comparably stable but slightly lower returns, implying that extreme settings, either sparser or denser, tend to align better with each method. Overall, this study on Tag highlights complementary preferences: TIGER-MIX benefits from selective sparsity, while TIGER-DICG thrives in information-rich static relational settings.



(a) (b)

Figure 8: **Static–temporal tradeoff in TIGER.** (a) TIGER-MIX peaks at moderate static connectivity (50%) and shallow temporal depth (one-hop) (b) TIGER-DICG remains robust across most settings, with 50% static threshold offering the best balance.

Self history $K_{\text{past\_self}}$ (Fig. 7b). We next vary the temporal window capturing each agent's own past states. Short memory settings ($K_{\text{past\_self}} = 1$) yield slightly lower returns, indicating that limited recall may restrict agents from identifying motion patterns over time. When self-history is removed ($K_{\text{past\_self}} = 0$), performance remains stable, suggesting that agents can still coordinate effectively using current observations and neighbor context. Extending the temporal horizon ($K_{\text{past\_self}} = \ln(NT) = 7$) consistently improves both variants, highlighting that deeper self-histories help smooth individual trajectories and enhance adaptation to continuous movement. Compared to Gather, where temporal recall offered modest benefits, Tag shows more noticeable advantages from longer self-memory.

Neighbor history $K_{\text{past\_nbr}}$ (Fig. 7c). We next analyze the influence of temporal neighbor information by varying $K_{\text{past\_nbr}}$. With a shallow temporal window ($K_{\text{past\_nbr}} = 1$), performance slightly declines, indicating that limited relational memory may not fully capture the evolving positions and interactions of nearby agents. When neighbor history is omitted ($K_{\text{past\_nbr}} = 0$), learning remains stable, showing that agents can still coordinate effectively based on immediate context. Extending the look-back window to a deeper setting ($K_{\text{past\_nbr}} = 2$) consistently enhances performance for both TIGER-MIX and TIGER-DICG, demonstrating the benefit of richer temporal neighbor information. Compared to Gather, this effect is more pronounced in Tag, where fast, spatially dynamic interactions make relational history especially valuable for sustained coordination.

Static-temporal tradeoff (Fig. 8). Finally, we examine the joint influence of static connectivity and temporal depth. For TIGER-MIX, returns remain largely consistent across settings, showing limited sensitivity to either factor under Tag's dense feedback. In contrast, TIGER-DICG performs best at moderate static connectivity ($K_{\text{stat\_nbr}}$=50%) and deeper history windows ($K_{\text{past\_nbr}}$=2, $K_{\text{past\_self}}$=7), indicating that richer temporal context supports its relational reasoning. Overall, Tag favors models with deeper temporal recall, though the performance differences are smaller than in Gather, likely due to its denser and more continuously interactive feedback.

Concluding remarks. In summary, these ablation experiments provide some useful insights into the sensitivity of TIGER to its structural and temporal design parameters. While the relative differences between configurations are modest, TIGER consistently outperforms all non-temporal baselines, reaffirming that temporal reasoning drives its performance gains. The observed variations across static connectivity, self-history, and neighbor-history further indicate that each component contributes to coordination quality, confirming that the temporal module captures evolving rela-

tional cues rather than acting as a redundant extension. Given the inherent complexity of multi-agent coordination, where learning dynamics, environmental stochasticity, and intertwined dependencies are difficult to isolate, these empirical findings invite a deeper theoretical or analytical discussion of temporal graph learning in MARL, which we leave as a promising direction for future work.

Appendix C. Implementation details

C.1. More details on MARL baselines

- **VDN** (Sunehag et al., 2018): VDNs decompose the joint Q-function into the sum of per-agent Q-values, enabling decentralized policies. VDN serves as a foundational baseline without explicit modeling of inter-agent dependencies.
- **QMIX** (Rashid et al., 2018): QMIX extends VDN by using a mixing network to combine individual agent Q-values under a monotonicity constraint, facilitating more expressive joint policies while maintaining decentralized execution.
- **QGNN** (Kortvelesy and Prorok, 2022): Graph Neural Network-based Q-value factorization employs message passing to encode agent interactions explicitly. We employ QGNN configurations based on standard implementations provided in the original paper, adjusting the parameters minimally to ensure compatibility with our scenarios.
- **GraphMix** (Naderializadeh et al., 2020): GraphMix extends the QMIX architecture by integrating static graph neural networks to improve relational reasoning among agents. We utilize the standard GraphMix settings with the default fully connected configuration as suggested in the corresponding literature.
- **DICG** (Li et al., 2021): Deep Implicit Coordination Graphs utilize attention mechanisms to dynamically generate weighted, fully connected coordination graphs among agents. Information is passed using learned attention weights. We strictly follow the original DICG methodology, employing QMIX as the underlying CTDE framework.
- **CASEC** (Wang et al., 2022): Context-Aware Sparse Deep Coordination Graphs selectively prune edges from fully connected coordination graphs based on variance estimation of payoffs. We configure CASEC using the variance-based edge pruning (as defined in the original paper) and set the sparsity loss weighting parameter $\lambda_{sparse} = 0.3$.
- **GACG** (Duan et al., 2024a): Group-Aware Coordination Graphs learn dynamic groupings among agents to capture higher-order relationships. We adopt recommended hyperparameters from the original implementation to effectively model agent group dependencies.
- **LTSCG** (Duan et al., 2024b): Latent Temporal Sparse Coordination Graphs infer sparse temporal graphs by combining edge probability learning with historical interaction encoding. We use the official codebase and training protocols.

C.2. MARL training setup

All agents share the same network configuration to ensure a fair comparison across baselines. Each agent uses a two-layer GRU (64 hidden units) to encode its local observation and produce an individual Q-vector. Mixer-based algorithms employ a two-layer hypernetwork with ReLU activations to compute aggregation weights, whereas graph-based learners treat agent embeddings as nodes and perform one round of message passing with 32-dimensional channels. Training settings are kept consistent across all methods to isolate the effect of temporal graph reasoning. ϵ-greedy explo-

ration is linearly annealed from 1.0 to 0.05 over the first 200K timesteps. Optimization uses Adam (5×10^{-4} learning rate), discount factor $\gamma = 0.99$, and TD(λ) with $\lambda = 0.8$. The replay buffer stores the most recent 5000 episodes, from which 32 mini-batches are sampled per update. Target networks are synchronized every 200 learner steps, and gradients are clipped to a maximum norm of 10 to ensure stability.

Appendix D. Discussion on the computational complexity

Size of Temporal Graph. Let N denote the number of agents and T the episode horizon. If the interaction graph is time-unrolled, it yields NT nodes in total. The number of edges decomposes into three main components: (i) **Static neighbors:** within each timestep, every agent can connect to up to $N-1$ others, resulting in $\mathcal{O}(N^2)$ edges per step; (ii) **Self-history:** each agent connects to its own past, producing $\mathcal{O}(NT)$ edges at time t and $\mathcal{O}(NT^2)$ over an episode; (iii) **Past neighbors:** agents connect to their neighbors' past states, yielding $\mathcal{O}(N^2T)$ edges at time t and $\mathcal{O}(N^2T^2)$ over the full episode. Hence, the total temporal graph complexity across the episode is dominated by

$$E_{\text{total}} = \mathcal{O}(N^2T^2),$$

indicating quadratic growth in both the number of agents and the horizon length.

Examples. For Gather with N=5 agents and T=8 timesteps, the temporal graph contains $NT = 40$ nodes. The static edges are $\frac{5\times4}{2} \times 8 = 80$ across the episode, while self-history and past-neighbor edges accumulate to approximately $5 \times \frac{8\times7}{2} = 140$ and $\frac{5\times4}{2} \times (8 \times 7) = 560$ respectively. Simailarly, for Tag, with N=13 and T=100, there are 1,300 nodes, 78 static edges, and roughly 64,350 and 772,200 edges from self- and neighbor-history respectively. This demonstrates how long horizons and dense interactions can produce rapid edge growth when temporal graphs are reconstructed at each timestep.

Controlling graph size. As shown in the above examples, even with relatively small agent counts and horizons, temporal graph construction can quickly grow in scale. TIGER introduces control parameters that directly mitigate this blow-up: $K_{\text{stat\_nbr}}$ limits static neighbor connectivity, $K_{\text{past\_self}}$ bounds each agent's self-history window, and $K_{\text{past\_nbr}}$ restricts how many neighbor histories are considered. These choices cap the effective complexity to a tractable subset of the full temporal graph, reducing redundant edges while retaining relevant relational structure. The scalability and flexibility of TIGER's temporal graph construction are particularly relevant to real-world MARL domains where agent coordination evolves over time. In autonomous driving or drone swarm control, where decisions unfold over long horizons and coordination structures change rapidly, deeper temporal recall ($K_{\text{past\_self}}$, $K_{\text{past\_nbr}}$) allows agents to leverage extended behavioral history and adapt to dynamic formations. In contrast, short-horizon or event-triggered domains such as multi-robot assembly or distributed sensing can benefit from lightweight configurations with shallow temporal windows and sparse connectivity ($K_{\text{stat\_nbr}}$), reducing computation without compromising coordination fidelity. Thus, the ability to regulate graph density and temporal depth enables TIGER to scale computationally across domains with different coordination requirements, maintaining efficiency in resource-constrained systems while preserving expressivity in complex environments.