CONTINUAL GRAPH MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In Continual Multi-Agent Reinforcement Learning (CMARL), agents learn cooperative policies across sequences of tasks, aiming to adapt effectively to new tasks while preserving the ability to solve previously encountered ones. In many applications, tasks differ in their underlying *structure*, which can represent, for example, distinct operational conditions or target configurations (e.g., different network topologies in power grids or arrangements in formation control). Existing CMARL methods lack dedicated mechanisms to leverage this structural information when learning new tasks, failing to promote transfer and mitigate forgetting. To fill this gap, we propose Continual Graph Multi-Agent Reinforcement Learning (CGMARL), a novel framework for CMARL problems in which task sequences are mapped into a series of attributed graphs, each modeling a task-specific structure. In CGMARL, each graph determines the environment dynamics (next states and/or rewards) and the number of agents for the corresponding task. Then, we present Graph-based Formation (GRAFO), the first CGMARL benchmark, and show how forgetting arises in this setting. Finally, to address this limitation, we propose Frozen Graph Encoder (FROG), a method that relies on a frozen graph backbone to preserve past structural information in graph-based CMARL policies. Experiments on GRAFO show that pairing FROG with existing CL methods substantially improves performance on multiple CGMARL scenarios.

1 INTRODUCTION

In cooperative Multi-Agent Reinforcement Learning (MARL; Littman, 1994), multiple RL agents learn to coordinate their behaviors to accomplish a shared task. This paradigm has proven effective across a variety of domains, such as autonomous driving (Candela et al., 2022), traffic signal control (Kolat et al., 2023), or video games (Vinyals et al., 2019). However, its applicability remains limited: most existing methods train RL agents on a single problem, whereas real-world scenarios typically involve sequences of related tasks that are often not known in advance. Continual RL (CRL; Khetarpal et al., 2022; Abel et al., 2023) formalizes this challenge by considering RL environments in which the agent never stops learning from an unknown sequence of tasks and cannot access previously observed ones. In particular, the agent aims to adapt to new tasks while preserving performance on previously learned ones (McCloskey & Cohen, 1989; Parisi et al., 2019), i.e., to transfer knowledge across tasks (*plasticity*) while avoiding catastrophic forgetting (*stability*). The multi-agent extension of the CRL paradigm, namely, Continual MARL (CMARL; Pan et al., 2025; Tomilin et al., 2026), poses additional challenges. Specifically, learning new tasks while mitigating the forgetting of past ones is complicated by multiple interacting learners (Yuan et al., 2024). This introduces a tightly coupled adaptation problem: agents must adapt to external changes in the environment, while accounting for changes in other agents' behaviors due to policy updates.

In many real-world domains, such as autonomous mobility-on-demand (Gammelli et al., 2021) or power grid control (Yoon et al., 2021; Hassouna et al., 2026), the multi-agent problem is characterized by an underlying *structure* (Liu et al., 2025), which encodes dependencies across agents (e.g., physical or relational). Importantly, this structure determines the environment's dynamics, i.e., it shapes state transitions and rewards. For example, in a traffic light network where each agent controls a specific intersection, local rewards and observations depend on incoming/outgoing vehicles from/to *connected* agents. Crucially, agents often face sequences of MARL tasks that differ in their underlying structures (Chen et al., 2026), requiring them to adapt their strategy to distinct structure-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

dependent dynamics. These structures may be entirely unrelated, such as flying formation problems with distinct target configurations, or may share similarities, e.g., adding or removing intersections and roads in a traffic light network. However, the current CMARL framework is not suited to address such sequential structure-based MARL problems: task shifts are typically attributed to variations in environment dynamics or team size (Pan et al., 2025), whereas the impact of task-specific problem structures is largely overlooked. Moreover, standard CL methods lack dedicated mechanisms to model *structural* information (Zhou & Cao, 2021), making them ineffective at preserving past structural knowledge when adapting to new tasks. For example, regularization-based methods (Kirkpatrick et al., 2017) mitigate forgetting by preserving important features or parameters, but fundamentally ignore the task's structure. Given these limitations, we argue that *graphs* provide a natural and effective approach for capturing task-specific structures by representing agents as nodes and encoding pairwise (agent-agent) dependencies through edges.

In this work, we introduce Continual Graph Multi-Agent Reinforcement Learning (CGMARL), a novel framework for continual structure-based MARL problems. In CGMARL, task sequences are characterized by a series of attributed graphs, each modeling a task-specific structure and determining the environment dynamics and the number of agents. We first show how distinct graph distributions induce CGMARL problems relevant to real-world scenarios. Then, we present the first CGMARL benchmark, named Graph-based Formation (GRAFO), and show the failure modes of graph-based policies in CGMARL. Finally, we propose Frozen Graph Encoder (FROG), a simple approach tailored for CGMARL problems that relies on decoupling graph representation learning from action prediction. Specifically, FROG leverages node representations generated with a frozen graph backbone to *provably* eliminate forgetting in the graph representation learning module caused by structural changes. At the same time, the action prediction module can be paired with standard CL approaches to further mitigate forgetting. To the best of our knowledge, this is the first work that formalizes continual graph-based MARL problems and can pave the way for novel methodologies tailored to CGMARL.

Main Contributions: 1) We introduce the Continual Graph Multi-Agent Reinforcement Learning (CGMARL) framework, in which continual structure-based MARL problems are modeled as a series of attributed graphs. **2)** We present Graph-based Formation (GRAFO), a novel graph-based navigation benchmark that allows us to design arbitrary CGMARL task sequences. **3)** We provide empirical evidence that, in CGMARL, standard graph-based methods fail to generalize across different graphs and strongly suffer from forgetting. **4)** We propose Frozen Graph Encoder (FROG), a simple technique that relies on a frozen graph backbone, and showcase its effectiveness across multiple CGMARL sequences.

2 PRELIMINARIES

Cooperative MARL: Cooperative MARL problems can be modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP; Oliehoek et al., 2016), i.e., as a tuple:

$$\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{O}} = \{\mathcal{O}^i\}_{i\in\mathcal{N}}, \boldsymbol{\mathcal{A}} = \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \mathcal{R}, \gamma \rangle \tag{1}$$

where $\mathcal{N}$ is the agent set, $\mathcal{S}$ is the state space, $\mathcal{O}_i$ and $\mathcal{A}_i$ are the i-th agent's observation and action spaces, $\mathcal{P} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathcal{S}$ and $\mathcal{R} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ are the transition and reward functions depending on the joint state and action, and $\gamma \in [0, 1)$ is a discount factor. Local observations $\{\boldsymbol{o}_t^i\}_{i\in\mathcal{N}}$ are given by an unknown function of the global state $\boldsymbol{s}_t$. The agents' objective is to learn a joint policy $\boldsymbol{\pi}$ composed of agent-specific policies $\{\pi^i\}_{i\in\mathcal{N}}$ that maximizes the expected discounted cumulative reward, denoted as $J(\boldsymbol{\pi})$. We assume agents to be homogeneous (Albrecht et al., 2024), and the joint policy is a collection of identical single-agent policies, i.e., $\pi^i = \pi \; \forall \; i \in \mathcal{N}$.

Continual MARL: In Continual MARL (CMARL; Pan et al., 2025; Tomilin et al., 2026), the multi-agent system undergoes a (possibly infinite) sequence of tasks $\mathcal{T} = \mathcal{M}_1, \ldots, \mathcal{M}_K$, where each $\mathcal{M}_k$ is a Dec-POMDP (Eq. 1) occurring for a fixed number of iterations. Tasks are drawn from a fixed, but unknown, task distribution (Abel et al., 2018). The subscript k denotes a Dec-POMDP with potentially different components, i.e., different $\mathcal{N}_k, \mathcal{P}_k, \mathcal{R}_k$.[1] In the remainder, we denote tasks using the index k to distinguish them from the time step t within an RL episode.

[1]The most general formulation of CRL (Khetarpal et al., 2022) allows for task-dependent state, observation, and action spaces, and the same can be adopted for Continual MARL. However, in practice, they are assumed to be fixed to facilitate learning (Pan et al., 2025).

3 CONTINUAL GRAPH MARL

In this section, we formalize the Continual Graph MARL (CGMARL) framework. First, we introduce the graph-based MARL problem, which corresponds to the single-task formulation. Then, after defining the CGMARL problem, we illustrate relevant task sequences.

3.1 GRAPH-BASED MARL TASK

In a graph-based MARL problem, the Dec-POMDP in Eq. 1 is further equipped with a graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{E}, \boldsymbol{X}, \boldsymbol{E} \rangle$, with $\mathcal{N}$ and $\mathcal{E}$ representing the set of nodes (agents) and edges, respectively, each equipped with feature matrices $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{N}| \times d_n}$ (nodes) and $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ (edges). Importantly, the graph $\mathcal{G}$ represents the task structure and determines the environment dynamics, i.e., the transition function $\mathcal{P}_\mathcal{G}$ and/or reward function $\mathcal{R}_\mathcal{G}$ depend on $\mathcal{G}$, providing local observations and rewards (Schneider et al., 1999). Thus, the Dec-POMDP tuple in Eq. 1 becomes:

$$\mathcal{M} = \langle \mathcal{G}, \mathcal{S}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, \mathcal{P}_\mathcal{G}, \mathcal{R}_\mathcal{G}, \gamma \rangle \tag{2}$$

For example, in a traffic light network, rewards and next observations depend on incoming and outgoing vehicles at each intersection, i.e., the 1-hop neighbors of each node in $\mathcal{G}$. Furthermore, agents can propagate messages through the network to coordinate effectively. We note that $\boldsymbol{X}$ and $\boldsymbol{E}$ may contain both static and dynamic features. In graph-based MARL, dynamic features are given by the agents' observations, which change during the episode. Therefore, we use $\boldsymbol{X}$ and $\boldsymbol{E}$ to denote the *static graph features* only, representing fixed node (e.g., agent identifier or type) and edge (e.g., sensor distances) features, respectively. We represent the edge set $\mathcal{E}$ with the adjacency matrix $A \in \{0,1\}^{|\mathcal{N}| \times |\mathcal{N}|}$, where $a_{ij} = 1$ if an edge connects node j to i. In the general formulation, A can vary within a task, i.e., the edge set $\mathcal{E}$ can change across steps (e.g., in navigation problems with distance-based connections). Specifically, edges are given by a state-dependent connectivity function $\varphi : \mathcal{N} \times \mathcal{S} \to \{0,1\}^{|\mathcal{N}| \times |\mathcal{N}|}$, which, at each step t, generates the adjacency matrix of the graph according to the current state $\boldsymbol{s}_t$. In this case, the graph is characterized by the tuple $\mathcal{G} = \langle \mathcal{N}, \varphi, \boldsymbol{X}, \boldsymbol{E} \rangle$, where we remind that the non-static graph features consist of the observations.

Comparison with Communication MARL: Graph-based MARL is fundamentally different from communication MARL (Zhu et al., 2024). In our formulation, the graph represents the problem structure and determines the dynamics. This is more similar to other graph-based MDPs, where, e.g., each node's rewards and observations depend on its neighbors in the graph (Hu et al., 2024a; Rashwan et al., 2026). Conversely, communication MARL focuses on developing effective communication methods across agents rather than using graphs to represent environment configurations with different dynamics. Specifically, the transition $\mathcal{P}$ and reward $\mathcal{R}$ functions do not depend on the graph structure $\mathcal{G}$, which is used only for message passing. While communication methods can exchange messages on different graph structures, they assume the underlying task to be fixed and, as a result, fail to adapt to unseen problems. We empirically analyse these failure modes in Sec. 5.1.

3.2 CGMARL SEQUENCE

We now formalize a Continual MARL problem in which the task sequence can be represented by a series of attributed graphs, each defining a specific graph-based MARL task (Eq. 2). We assume the graph-dependent environment dynamics (i.e., $\mathcal{P}_\mathcal{G}$ and $\mathcal{R}_\mathcal{G}$) to be fixed across tasks. Thus, each task $\mathcal{M}_k$ in the sequence is uniquely identified by the graph $\mathcal{G}_k$, which determines the transition function $\mathcal{P}_{\mathcal{G}_k}$ and reward function $\mathcal{R}_{\mathcal{G}_k}$. We now define the Continual Graph MARL (CGMARL) problem.

Definition 3.1 (CGMARL Problem)**.** Let $\mathcal{S}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, \gamma$ be the fixed components of the graph-based Dec-POMDP (Eq. 2) representing state, observation, and action spaces, and discount factor, respectively. Let $\mathcal{G} = \langle \mathcal{N}, \varphi, \boldsymbol{X}, \boldsymbol{E} \rangle$ be a graph, and let $\mathcal{P}_\mathcal{G}$ and $\mathcal{R}_\mathcal{G}$ be the graph-dependent transition and reward functions. Given a fixed but unknown distribution $\mathcal{Z}$ over graphs, the Continual Graph Multi-Agent Reinforcement Learning (CGMARL) problem is defined as a sequence of tasks $\mathcal{T} = \mathcal{M}_1, \ldots, \mathcal{M}_K$, where each task $\mathcal{M}_k$ is defined as in Eq. 2 and characterized by the graph $\mathcal{G}_k$ sampled from the distribution $\mathcal{Z}$.

This is the most general formulation of Continual Graph MARL. This class of problems can be motivated by considering a navigation task with distance-based connections. In this setting, each pair of

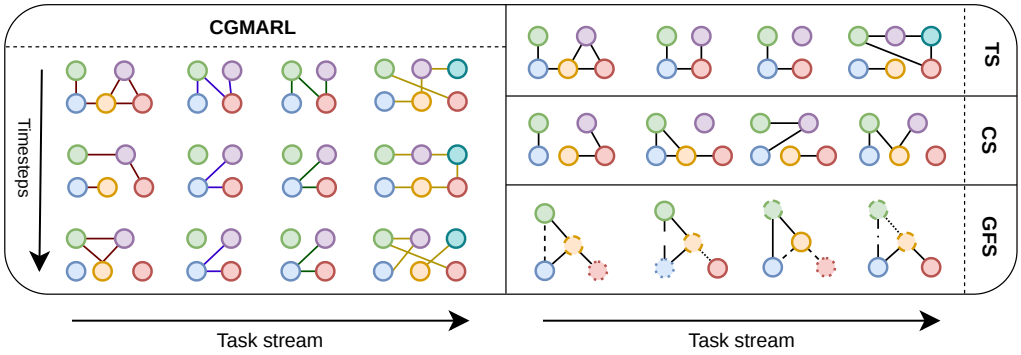


Figure 1: Scheme of the CGMARL problem. In the left panel, we illustrate the general setup, in which edges are generated dynamically across timesteps using task-specific connectivity functions (indicated by different colors for edges). In the right panel, we illustrate relevant classes of the CGMARL problem. In Topology Switch (TS), both nodes and edges can change across tasks. In Connectivity Switch (CS), edges can change while the node set is fixed. In Graph Features Switch (GFS), the topology is kept static, and graph features can change; this is highlighted by using different dashed patterns to represent nodes and/or edges.

agents is connected if their distance is within a specified threshold, such as the vision range (Nayak et al., 2023). Across tasks, the team composition $\mathcal{N}_k$ may vary depending on, e.g., the number of target positions to be covered or map size. Additionally, different scenarios may exhibit varying thresholds, thereby influencing the connectivity function φ_k that generates the communication structure (edge set) at each time step. Depending on the graph distribution $\mathcal{Z}$, we can derive different classes of CGMARL problems, each with its relevance and potential applicability. In the following, we present formal definitions alongside real-world examples fitting the CGMARL framework in decreasing order of generality. Please refer to Fig. 1 for a visual summary.

3.2.1 Topology Switch

In the Topology Switch (TS) problem, connections (edge set $\mathcal{E}_k$) remain fixed *within* a task, but may vary across *different* tasks. Formally, for each task $\mathcal{M}_k$, the graph $\mathcal{G}_k$ sampled by the distribution $\mathcal{Z}$ has a connectivity function φ_k that is state-independent, thus generating a fixed edge set $\mathcal{E}_k = \varphi_k(\mathcal{N}_k)$. In other words, the adjacency matrix A_k remains fixed within the task $\mathcal{M}_k$.

Definition 3.2 (TS Problem)**.** We call the Topology Switch (TS) problem a CGMARL problem where the task sequence $\mathcal{T}$ is generated under state-independent connectivity functions $\varphi : \mathcal{N} \to \{0,1\}^{|\mathcal{N}|\times|\mathcal{N}|}$; it follows that each task $\mathcal{M}_k$ of $\mathcal{T}$ is characterized by a graph $\mathcal{G}_k = \langle \mathcal{N}_k, \mathcal{E}_k, \boldsymbol{X}_k, \boldsymbol{E}_k \rangle$ with a static edge set $\mathcal{E}_k = \varphi_k(\mathcal{N}_k)$.

A graph-based control policy for efficient traffic light control (Nishi et al., 2018; Devailly et al., 2022) is an example of a TS problem. In this setting, the same model could be deployed and updated on completely different network topologies, e.g., different cities or districts within a city. Importantly, the connections do not change within the same task.

3.2.2 Connectivity Switch

The Connectivity Switch (CS) problem is a sub-case of the TS setting in which the node set remains fixed across tasks. In other words, in this CGMARL setting, we consider a *static* team of agents that cooperate to solve different tasks, each with its own communication structure and graph features. Therefore, the sampling process from $\mathcal{Z}$ corresponds to sampling a sequence of different adjacency matrices[2] and graph features given a fixed node set.

[2]With a state-dependent connectivity function, edges could also change *within* a single task. However, unless the environment dynamics vary, the fixed node set restricts the possible edge configurations, inducing task similarity. We leave this variant of the CS setting to future work.

Definition 3.3 (CS Problem)**.** We call the Connectivity Switch (CS) problem a CGMARL problem in which the task sequence $\mathcal{T}_{\mathcal{N}}$ is generated under a specific choice for the node set $\mathcal{N}$; it follows that each task $\mathcal{M}_k$ of $\mathcal{T}_{\mathcal{N}}$ is characterized by a graph $\mathcal{G}_k = \langle \mathcal{N}, \mathcal{E}_k, \boldsymbol{X}_k, \boldsymbol{E}_k \rangle$, with $\mathcal{E}_k = \varphi_k(\mathcal{N})$.

The CS problem can be relevant, for example, in efficient power distribution management (Yoon et al., 2021). Here, the physical network representing the power grid consists of a fixed set of nodes (e.g., substations), while the connections may vary according to the current operational regime. Specifically, the network can transition between predefined edge configurations in response to external operational conditions (e.g., fault isolation or maintenance).

3.2.3 Graph Features Switch

When the topology (i.e., node and edge sets) remains fixed throughout the task sequence, static graph features can discriminate between different graphs (tasks). In the Graph Features Switch (GFS) problem, the graph distribution $\mathcal{Z}$ is a distribution over static graph features ($\boldsymbol{X}$ and $\boldsymbol{E}$).

Definition 3.4 (GFS Problem)**.** We call the Graph Features Switch (GFS) problem a CGMARL problem in which the task sequence $\mathcal{T}_{\mathcal{N},\mathcal{E}}$ is generated under a specific choice for the topology ($\mathcal{N}$ and $\mathcal{E} = \varphi(\mathcal{N})$); it follows that each task $\mathcal{M}_k$ of $\mathcal{T}_{\mathcal{N},\mathcal{E}}$ is characterized by a graph $\mathcal{G}_k = \langle \mathcal{N}, \mathcal{E}, \boldsymbol{X}_k, \boldsymbol{E}_k \rangle$.

In this class of CGMARL problems, graph features play a critical role in tackling the task. For instance, consider a robotic agent where each joint (actuator) is represented as a node, and the limbs connecting them are the edges (Wang et al., 2018; Marzi et al., 2024; Yang et al., 2024). In scenarios where limb lengths depend on the task, it is essential to learn a policy that adapts to different configurations. For example, limbs may be lengthened or shortened in response to different obstacles or terrain. In this case, $\boldsymbol{E}_k$ can encode task-dependent distances between node pairs, allowing robots to operate effectively with longer or shorter limbs. Similarly, node features $\boldsymbol{X}_k$ can encode information on task-specific conditions, such as node-level body masses. Note that, unlike TS and CS sequences where the topology implicitly serves as the communication structure, in the GFS problem the features $\boldsymbol{X}$ and $\boldsymbol{E}$ characterizing the graph are *explicitly* used in decision-making.

4 GRAFO Benchmark

As a first testbed for CGMARL problems, we propose Graph-based Formation (GRAFO), a novel graph-based navigation benchmark that supports generating arbitrarily long CGMARL sequences. In GRAFO, the agents' objective is to navigate to a target configuration determined by the static input graph characterizing the task (refer to Fig. 2 for a visual example). Specifically, a landmark (the green, solid cross in Fig. 2) represents the target position of a specific agent (the green circle), and it is known to all agents. Instead, other agents' targets (dashed crosses) are not directly observable and are implicitly defined by the graph topology. Since only one target is explicitly provided, agents must leverage structural information to identify the correct formation. Each agent must reach its target position, and the agents' graph-based rewards are dense signals based on the distances from their assigned targets, i.e., they depend on the task-specific graph (Sec. 3.1). CGMARL sequences can be easily defined by considering a series of graphs (App. E.2). We report a more detailed description of GRAFO in App. C.

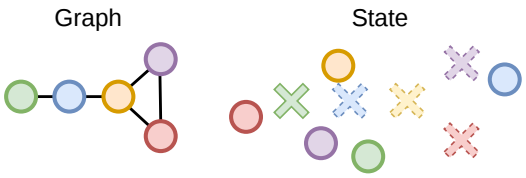


Figure 2: Example of a GRAFO task with five agents (circles). Left: agent graph identifying the task. Right: current state of the environment, where the green solid cross is the landmark known to all agents, i.e., the target position of the green agent, while the dashed crosses are the target positions of other agents. These positions are defined based on the graph, but are not explicitly provided.

5 Emergence of Forgetting in CGMARL

In this section, we first show how GNN-based policies fail to preserve structural information in CGMARL problems. Then, we present a simple method to mitigate forgetting in this setting.

5.1 GNN-BASED POLICIES IN GRAPH-BASED MARL

Graph Neural Networks (GNNs; Bacciu et al., 2020) operate on graph-structured data and generate node-level representations by aggregating information over each node's neighborhood. The updated representation $\boldsymbol{x}_i'$ of a node i with neighboring nodes $\mathcal{B}(i)$ can be written as (Gilmer et al., 2017):

$$\boldsymbol{x}_i' = \psi_U\left(\boldsymbol{x}_i, \underset{j\in\mathcal{B}(i)}{\text{AGGR}}\left\{\psi_M\left(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{e}_{ij}\right)\right\}\right), \tag{3}$$

where $\boldsymbol{e}_{ij} \in \mathbb{R}^{d_e}$ is the feature vector of edge a_{ij}, AGGR is a permutation-invariant aggregation function, and ψ_U, ψ_M are learnable update and message functions, respectively. GNNs are designed to process graphs of *any* size, i.e., with a varying number of nodes and edges. This makes GNN-based policies suitable candidates to generalize naturally across different graph-based MARL tasks. However, we argue that this cannot be easily achieved: GNNs learn local (node-level) rules that may not directly apply to unseen tasks if the underlying graph-dependent dynamics change. When considering CGMARL sequences, this requires overwriting previously learned rules when adapting to new tasks, making GNN-based policies prone to forgetting.

To empirically assess these observations, we evaluate InforMARL (App. E.1; Nayak et al., 2023) on two GRAFO tasks, from both zero-shot and continual perspectives. Specifically, we train the model on graphs with 3 and 5 nodes individually (GRAFO-3 and GRAFO-5) and sequentially (GRAFO-$(3,5)$ and GRAFO-$(5,3)$), and test it on both tasks (further details are reported in App. E.2). The results reported in Tab. 1 support our claim. In particular, the zero-shot transfer (TRAN.) results show that the model solves all tasks individually, but the learned behavior cannot be directly transferred to different target formations. When learning sequentially (CONT.), the model fails to retain previously acquired knowledge, resulting in complete forgetting of the first task learned (e.g., task GRAFO-3 in the GRAFO-$(3,5)$ sequence).

Table 1: Solved percentage at test time (median $\pm$ standard deviation of 5 seeds) for two GRAFO tasks in both zero-shot transfer (TRAN.) and continual (CONT.) settings.

	Train \ Test	GRAFO-3	GRAFO-5
TRAN.	GRAFO-3	**1** ± 0	0 ± 0
	GRAFO-5	0.016 ± 0.016	**1** ± 0
CONT.	GRAFO-$(3,5)$	0.000 ± 0.027	**1** ± 0
	GRAFO-$(5,3)$	**1** ± 0	0.000 ± 0.014

We did not apply any CL strategy (e.g., feature importance, Kirkpatrick et al., 2017) in the sequential (CONT.) setting of Tab. 1. Standard CL methods can mitigate forgetting but, as also shown by the results in Sec. 6, they do not explicitly account for changes encoded in the graph representation of the task. In the following, we propose a simple method specifically tailored to CGMARL problems.

5.2 FROG: A SIMPLE APPROACH TO MITIGATE FORGETTING IN CGMARL

Standard CL approaches do not account for forgetting in graph representation learning. We address this by decoupling graph processing from action prediction. Specifically, consider a policy $\pi_{\boldsymbol{\theta}} = \mu_{\theta_2} \circ \psi_{\theta_1}$, where ψ_{θ_1} is a generic GNN as in Eq. 3 with learnable parameters θ_1 and μ_{θ_2} is an action prediction module (e.g., MLP) with learnable parameters θ_2. Actions are taken as:

$$\boldsymbol{a}_t^i \doteq \pi_{\boldsymbol{\theta}}(\boldsymbol{o}_t^i; \mathcal{G}) = \mu_{\theta_2}(\psi_{\theta_1}(\boldsymbol{o}_t^i; \mathcal{G})), \tag{4}$$

where we assumed a deterministic policy (it can be easily generalized to a stochastic one). Under this formulation, the graph encoder ψ_{θ_1} captures the structural information, while the action module μ_{θ_2} directly maps node representations to actions. This facilitates analyzing forgetting associated with graph representation learning. Formally, given task indexes k and h, with $k > h$, forgetting on task h after training on task k is defined as (details in App. A):

$$J_h(\boldsymbol{\pi}_k) - J_h(\boldsymbol{\pi}_h) \leq \underbrace{\|\boldsymbol{\mu}_k(\boldsymbol{\psi}_k(\boldsymbol{o}; \mathcal{G}_h)) - \boldsymbol{\mu}_k(\boldsymbol{\psi}_h(\boldsymbol{o}; \mathcal{G}_h))\|_\infty}_{\text{Graph Representation Learning}} + \underbrace{\|\boldsymbol{\mu}_k(\boldsymbol{\psi}_h(\boldsymbol{o}; \mathcal{G}_h)) - \boldsymbol{\mu}_h(\boldsymbol{\psi}_h(\boldsymbol{o}; \mathcal{G}_h))\|_\infty}_{\text{Action Prediction}} \tag{5}$$

where $J_i(\boldsymbol{\pi}_j)$ is the performance (e.g., expected return) achieved on task i using joint policy $\boldsymbol{\pi}_j$ (with parameters $j := \boldsymbol{\theta}_j = \{\theta_{1,j}, \theta_{2,j}\}$ after training task j). From the first term on the right-hand side of Eq. 5, we can *provably* eliminate forgetting in the graph representation learning module caused by structural changes by imposing $\psi_k = \psi_h$ for each task pair h, k. This can be achieved by freezing

the graph encoder ψ either *before* or *after* training the first task. In our experiments, we adopt the second option and call our method Frozen Graph Encoder (FROG). Specifically, freezing the graph encoder *after* learning the first task preserves its learned parameters while allowing it to generate distinct graph representations for each task-specific graph. These are then passed to the action module μ_{θ_2}, which continuously adapts to new tasks. To mitigate forgetting in this module (second term on the right-hand side of Eq. 5), we pair it with existing CL methods. In this way, we leverage the generalization capacity of the GNNs (Sec. 5.1) while confining task-specific adaptation to the action prediction module. Note that freezing the graph encoder *before* training provides node-level representations as random graph features (Zambon et al., 2020; Navarin et al., 2023), which have been proven effective in CL problems on graphs (Donghi et al., 2026). Nonetheless, we argue that this option may hinder learning in complex graph-based MARL settings, especially if the GNN lacks sufficient expressive power. We compare FROG with this untrained variant in Sec. 6.2.

6 Empirical Evaluation

In this section, we present a series of experiments to empirically support our claims. Specifically, we have two main objectives: evaluate how standard CL methods perform in CGMARL problems (**Obj. 1**), and investigate whether FROG, i.e., a simple method designed to account for structural changes, can provide benefit in this setting (**Obj. 2**), paving the way for novel CGMARL approaches.

CGMARL Sequences: We consider multiple CGMARL sequences from our proposed GRAFO benchmark (please refer to App. C and App. E.2 for further details). Specifically, for the Topology Switch (TS) scenario (Sec. 3.2.1), we consider a sequence of 6 tasks, with 3 to 7 agents; the Connectivity Switch (CS) scenario (Sec. 3.2.2) is a sequence of 5 tasks, each with 6 agents; for the Graph Feature Switch (GFS) scenario (Sec. 3.2.3), we consider a sequence of 4 tasks, each with 4 agents. Furthermore, we analyze the impact of task order for the TS sequence in Sec. 6.2. Following related work (Tomilin et al., 2026), we consider the task-incremental learning paradigm with separate task heads and known task identities. We assess the impact of conditioning on task identities in App. D.4.

Models: We employ InforMARL (Nayak et al., 2023) as the single-task GNN-based policy, as it is a well-established model for graph-based navigation domains (more details in App. E.1). To address forgetting in the CGMARL sequence, we pair InforMARL with common CL approaches (Tomilin et al., 2026). Specifically, we consider L2, where we apply an L2 penalty to the weights, Elastic Weight Consolidation (EWC; Kirkpatrick et al., 2017), and Memory Aware Synapses (MAS; Aljundi et al., 2018). For EWC and MAS, we consider two variants depending on how parameter importance is computed: the "multi" (M) versions aggregate the importance matrix from all tasks via summation, while the "online" (O) versions employ an online exponential-decay running average of the importance matrix from past tasks. We pair FROG with all the CL methods, and consider an additional Fine-Tuning (FT) baseline without any regularization. Hyperparameters for InforMARL and the CL methods are reported in App. E.4.

Metrics: We evaluate the models using metrics from standard CL literature (Wolczyk et al., 2021; Caccia et al., 2022; Sokar et al., 2025), i.e., Average Final Performance (AFP), Average Forgetting (AF), and Average Plasticity (AP). To do so, for each task, we measure the percentage of solved episodes during evaluation, where an episode is considered solved if the agents reach the target configuration within the time limit. Please refer to App. E.3 for a more detailed description.

6.1 Main Results

We report the main results on three different CGMARL scenarios in Fig. 3. Standard CL methods exhibit mixed performance across the various sequences, with none consistently outperforming the others. On average, EWC achieves better AFP than MAS and L2, but at the cost of plasticity loss (lower AP). Combined with the high variance, this suggests that standard CL methods may require additional mechanisms tailored for CGMARL to be more effective (**Obj. 1**). Overall, compared with their counterparts that do not incorporate it, FROG consistently improves performance when combined with standard CL methods (**Obj. 2**), and is less prone to forgetting (higher AF). The only exception is EWC on the TS sequence, where the standard method has a higher AF value. However, this can be explained by plasticity loss: as the lower AP metric highlights, the standard version fails to learn new tasks. This reduced learning capacity necessarily leads to less forgetting, since the

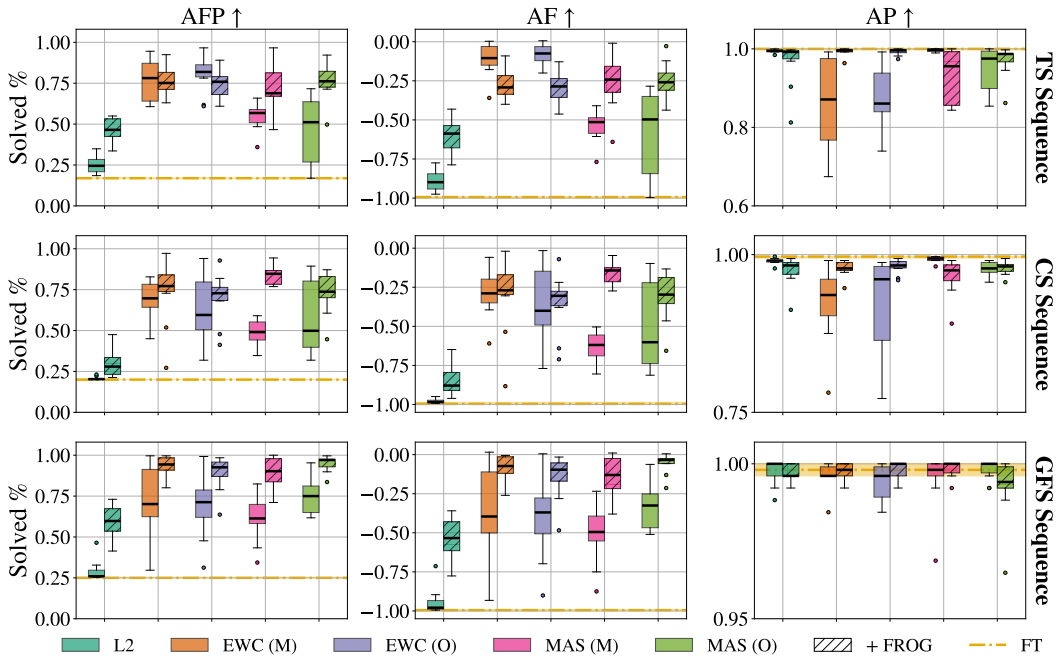


Figure 3: Boxplot showing the distribution of 10 independent seeds for the different CL methods (with and without FROG) across the three scenarios, i.e., TS (top), CS (middle), and GFS (bottom). Boxes represent the IQR, horizontal lines indicate medians, and whiskers extend to $1.5 \times$ IQR.

more difficult tasks are not successfully learned in the first place and therefore cannot subsequently be forgotten. We hypothesize that this may be caused by using a single regularization coefficient for both the graph encoder and action module, and we investigate this behavior in App. D.2. Finally, empirical evidence shows that, on average, using frozen graph representations stabilizes learning.

6.2 Additional Analyses

In this section, we conduct a set of experiments to further assess the effectiveness of FROG.

Untrained Graph Encoder: We compare FROG with its untrained variant (FROG (RND)), where the graph encoder is frozen at initialization, and the corresponding representations are random. Aggregated results in Tab. 2 (please refer to Fig. 7 for a detailed plot) achieved in the TS sequence support our claim (Sec. 5.2): using completely random graph features hinders learning in GRAFO, as the model struggles to learn the hardest tasks (low AP).

Table 2: Aggregated medians (average ± standard deviation) for the different CL methods combined with FROG and FROG (RND) on the TS sequence. Complete boxplots are reported in Fig. 7.

	AFP ↑	AF ↑	AP ↑
FROG	**0.69** ±0.13	−0.33 ±0.14	**0.99** ±0.02
FROG (RND)	0.38 ±0.037	**−0.29** ±0.04	0.64 ±0.03

Task Order: FROG relies on freezing the graph encoder after training the first task. This implies that the quality of the node representations generated throughout the CGMARL sequence depends on the first task encountered. To assess this behavior, we consider two variants of the TS sequence used in Sec. 6.1 with an incremental/decremental number of agents, and dub them TS (inc) and TS (dec), respectively. The results reported in Fig. 4 show that, on average, FROG achieves competitive performance against the standard counterpart. In particular, in TS (inc), the first task requires the lowest coordination degree (3 agents). This causes plasticity loss (lower AP) in FROG when paired with MAS and L2, while it still guarantees good performance when combined with EWC. Conversely, in TS (dec), results are similar to the standard TS sequence, favoring FROG and suggesting that the learned graph encoder is rich enough to perform well on the full sequence. This shows that the first task can have an impact on FROG, but despite that, it can still achieve competitive performance. Finally, this analysis reveals that task order in GRAFO can also impact learning in standard CL methods, which achieve, on average, lower AFP and AP in the TS (inc) sequence.

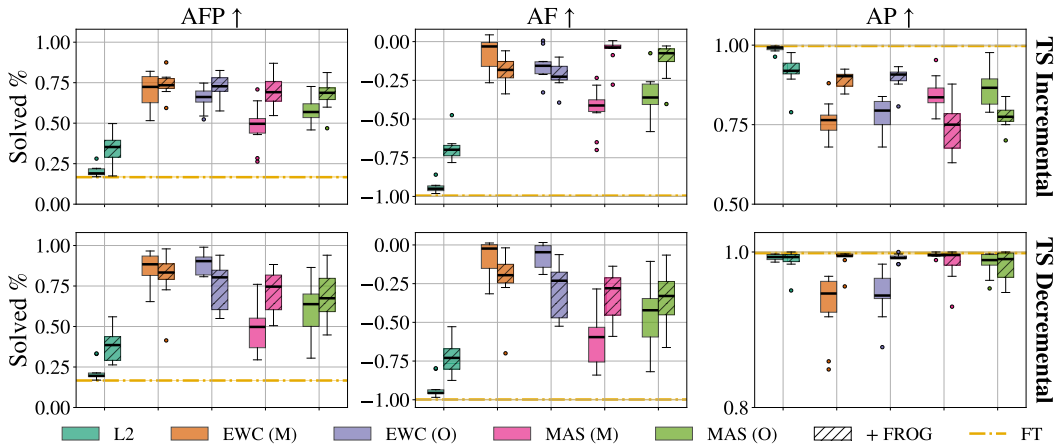


Figure 4: Boxplot showing the distribution of 10 independent seeds for the different CL methods (with and without FROG) across the TS (inc) and TS (dec) sequences, i.e., the incremental/decremental variants of the TS sequence (top and bottom panels, respectively). Boxes represent the IQR, horizontal lines indicate medians, and whiskers extend to $1.5 \times$ IQR.

7 Related Work

Graph representations have been used in MARL to characterize the task structure in several settings. For example, many approaches (Hu et al., 2024a; Rashwan et al., 2026) consider problems where local (agent-specific) dynamics are conditioned on actions of each agent's neighbors in the graph. Practical applications of graph-based MARL range from networked systems (Liu et al., 2025) to power grid control (Hassouna et al., 2026). Communication MARL (Zhu et al., 2024) emerged as an effective paradigm for addressing those tasks by improving agents' coordination through communication. This is commonly achieved using Graph Neural Networks (GNNs; Gilmer et al., 2017) in decision-making to learn message-passing policies over the graph structure (Jiang et al., 2020; Niu et al., 2021; Nayak et al., 2023). In Continual RL (CRL; Khetarpal et al., 2022; Abel et al., 2023), the agent aims to sequentially adapt to new tasks without deteriorating performance on previously learned ones. To prevent forgetting, CL methods such as replay-based mechanisms (Chaudhry et al., 2019) or parameter regularizations (Kirkpatrick et al., 2017) have been employed. The CRL problem has been extended to multi-agent systems (Pan et al., 2025; Tomilin et al., 2026), but to the best of our knowledge, this is the first work that considers continual graph-based MARL settings. Related to our method, frozen modules or representations have been effectively used to mitigate forgetting in CRL (Rusu et al., 2016; Powers et al., 2022; Nikishin et al., 2023; Hu et al., 2026). Please refer to App. B for an extended discussion.

8 Conclusion and Future Work

In this work, we introduced the Continual Graph Multi-Agent Reinforcement Learning (CGMARL) framework, which formalizes sequences of MARL tasks characterized by different structures. Through this framework, we identified several real-world applications induced by different task (graph) distributions. Motivated by the failure modes of standard approaches in CGMARL, we proposed Frozen Graph Encoder (FROG), a simple approach that leverages frozen graph representations to mitigate forgetting. Experiments on a novel CGMARL benchmark, named Graph-based Formation (GRAFO), validated FROG's effectiveness when combined with standard CL methods.

We foresee several possible directions for future theoretical and empirical research. First, a theoretical investigation of the failure modes of standard CL approaches in this setting could help design more effective CGMARL methods. Furthermore, considering off-policy MARL algorithms would pave the way for testing experience-based CL methods. Lastly, it would be interesting to develop other CGMARL evaluation benchmarks, possibly with different practical applications (e.g., power distribution management or traffic light control).

AI USE STATEMENT

In this paper, we used generative AI tools to improve readability and polish writing. Furthermore, we used generative AI tools to help implement parts of the code used to run the experiments, and we carefully verified and tested these parts. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All the material needed to reproduce the results is reported in the main text and the appendix. In particular, App. A provides further details on the theoretical part of Sec. 5.1; in Sec. 4 and App. C we provide a complete description of Graph-based Formation (GRAFO), our proposed benchmark; in App. E we describe all the components (model, task sequences, metrics, and hyperparameters) used in our experiments (Sec. 6). Additionally, we provide a link to the anonymous source code at https://anonymous.4open.science/r/cgmarl-code/.

REFERENCES

David Abel, Yuu Jinnai, Sophie Yue Guo, George Konidaris, and Michael Littman. Policy and value transfer in lifelong reinforcement learning. In *International conference on machine learning*, pp. 20–29. PMLR, 2018.

David Abel, André Barreto, Benjamin Van Roy, Doina Precup, Hado P van Hasselt, and Satinder Singh. A definition of continual reinforcement learning. *Advances in Neural Information Processing Systems*, 36:50377–50407, 2023.

Akshat Agarwal, Sumit Kumar, Katia Sycara, and Michael Lewis. Learning transferable cooperative behavior in multi-agent teams. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pp. 1741–1743, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

Stefano V Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part III*, pp. 144–161, Berlin, Heidelberg, 2018. Springer-Verlag. ISBN 978-3-030-01218-2. doi: 10.1007/978-3-030-01219-9_9. URL https://doi.org/10.1007/978-3-030-01219-9_9.

Alessandro Amato, Raffaele Galliera, K Brent Venable, and Niranjan Suri. Encoding goals as graphs: Structured objectives for scalable cooperative multi-agent reinforcement learning. In *Second Coordination and Cooperation in Multi-Agent Reinforcement Learning Workshop*, 2025.

Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 2020.

Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1485–1494, 2023.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=rJNwDjAqYX.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=N8MaByOzUfb.

Eduardo Candela, Leandro Parada, Luis Marques, Tiberiu-Andrei Georgescu, Yiannis Demiris, and Panagiotis Angeloudis. Transferring multi-agent reinforcement learning policies for autonomous driving using sim-to-real. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8814–8820. IEEE, 2022.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Boyuan Chen, Chuning Zhu, Pulkit Agrawal, Kaiqing Zhang, and Abhishek Gupta. Self-supervised reinforcement learning that transfers using random features. *Advances in Neural Information Processing Systems*, 36:56411–56436, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Yanping Chen, Qingyang Xu, Chi Zhang, and Zhengmao Li. Multi-agent deep reinforcement learning (madrl)-based end-to-end formation control for uav swarm with dynamic topology. *Applied Sciences*, 16(13):6554, 2026.

Tianshu Chu, Sandeep Chinchali, and Sachin Katti. Multi-agent reinforcement learning for networked system control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx7A3NFvH.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on machine learning*, pp. 1538–1546. PMLR, 2019.

François-Xavier Devailly, Denis Larocque, and Laurent Charlin. Ig-rl: Inductive graph reinforcement learning for massive-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7496–7507, 2022. doi: 10.1109/TITS.2021.3070835.

Giovanni Donghi, Luca Pasa, Daniele Zambon, Cesare Alippi, and Nicolò Navarin. Online continual graph learning. *arXiv preprint arXiv:2508.03283*, 2025.

Giovanni Donghi, Daniele Zambon, Luca Pasa, Cesare Alippi, and Nicolò Navarin. The unreasonable effectiveness of randomized representations in online continual graph learning. *Machine Learning*, 115, 09 2026. doi: 10.1007/s10994-026-07128-5.

Wei Duan, Jie Lu, and Junyu Xuan. Bayesian ego-graph inference for networked multi-agent reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=3qeTs05bRL.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Daniele Gammelli, Kaidi Yang, James Harrison, Filipe Rodrigues, Francisco C Pereira, and Marco Pavone. Graph neural network reinforcement learning for autonomous mobility-on-demand systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 2996–3003. IEEE, 2021.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Mohamed Hassouna, Clara Holzhüter, Pawel Lytaev, Josephine Thomas, Bernhard Sick, and Christoph Scholz. Graph reinforcement learning for power grids: A comprehensive survey. *Energy and AI*, pp. 100671, 2026.

Tyler L Hayes and Christopher Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 887–896. IEEE, 2020.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Jinwu Hu, Zihao Lian, Zhiquan Wen, Chenghao Li, Guohao Chen, Xutao Wen, Bin Xiao, and Mingkui Tan. Continual knowledge adaptation for reinforcement learning. *Advances in Neural Information Processing Systems*, 38:36283–36313, 2026.

Jueming Hu, Zhe Xu, Weichang Wang, Guannan Qu, Yutian Pang, and Yongming Liu. Decentralized graph-based multi-agent reinforcement learning using reward machines. *Neurocomputing*, 564: 126974, 2024a.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=Qox9rO0kN0.

Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.

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

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkxdQkSYDB.

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

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Máté Kolat, Bálint Kővári, Tamás Bécsi, and Szilárd Aradi. Multi-agent reinforcement learning for traffic signal control: A cooperative approach. *Sustainability*, 15(4):3479, 2023.

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1920–1929. IEEE, 2019.

Depeng Li and Zhigang Zeng. Crnet: A fast continual learning framework with random theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10731–10744, 2023.

Yiheng Lin, Guannan Qu, Longbo Huang, and Adam Wierman. Multi-agent reinforcement learning in stochastic networked systems. *Advances in neural information processing systems*, 34:7825–7837, 2021.

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

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

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. VIP: Towards universal visual reward and representation via value-implicit pre-training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=YJ7o2wetJ2.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773. IEEE, 2018.

Tommaso Marzi, Arshjot Singh Khehra, Andrea Cini, and Cesare Alippi. Feudal graph reinforcement learning. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=wFcyJTik90.

Tommaso Marzi, Cesare Alippi, and Andrea Cini. Hierarchical message-passing policies for multi-agent reinforcement learning. *arXiv preprint arXiv:2507.23604*, 2025.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton Van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36:12022–12053, 2023.

Golnaz Mesbahi, Parham Mohammad Panahi, Olya Mastikhina, Steven Tang, Martha White, and Adam White. Position: Lifetime tuning is incompatible with continual reinforcement learning. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025. URL https://openreview.net/forum?id=JMoWFkwnvv.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 892–909. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/nair23a.html.

Nicolò Navarin, Luca Pasa, Claudio Gallicchio, and Alessandro Sperduti. An untrained neural model for fast and accurate graph classification. In *International Conference on Artificial Neural Networks*, pp. 278–290. Springer, 2023.

Siddharth Nayak, Kenneth Choi, Wenqi Ding, Sydney Dolan, Karthik Gopalakrishnan, and Hamsa Balakrishnan. Scalable multi-agent reinforcement learning through intelligent information aggregation. In *International conference on machine learning*, pp. 25817–25833. PMLR, 2023.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36:37142–37159, 2023.

Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 877–883, 2018. doi: 10.1109/ITSC.2018.8569301.

Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. Multi-agent graph-attention communication and teaming. In *AAMAS*, volume 21, pp. 20th, 2021.

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

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Chaofan Pan, Xin Yang, Yanhua Li, Wei Wei, Tianrui Li, Bo An, and Jiye Liang. A survey of continual reinforcement learning. *arXiv preprint arXiv:2506.21872*, 2025.

Yoh-Han Pao, Stephen M Phillips, and Dejan J Sobajic. Neural-net computing and the intelligent control of systems. *International Journal of Control*, 56(2):263–289, 1992.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Sam Powers, Eliot Xing, and Abhinav Gupta. Self-activating neural ensembles for continual reinforcement learning. In *Conference on Lifelong Learning Agents*, pp. 683–704. PMLR, 2022.

Sam Powers, Abhinav Gupta, and Chris Paxton. Evaluating continual learning on a home robot. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup (eds.), *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pp. 493–512. PMLR, 22–25 Aug 2023. URL https://proceedings.mlr.press/v232/powers23a.html.

Ameya Prabhu, Shiven Sinha, Ponnurangam Kumaraguru, Philip H Torr, Ozan Sener, and Puneet K Dokania. Randumb: Random representations outperform online continually learned representations. *Advances in Neural Information Processing Systems*, 37:37988–38006, 2024.

Chao Qu, Shie Mannor, Huan Xu, Yuan Qi, Le Song, and Junwu Xiong. Value propagation for decentralized networked deep multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Guannan Qu, Adam Wierman, and Na Li. Scalable reinforcement learning for multiagent networked systems. *Operations Research*, 70(6):3601–3628, 2022.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Ahmed Rashwan, Keith Briggs, Chris Budd, and Lisa Maria Kreusser. Factored value functions for graph-based multi-agent reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gNwwzeqPF0.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Jeff G Schneider, Weng-Keen Wong, Andrew W Moore, and Martin A Riedmiller. Distributed value functions. In *Proceedings of the Sixteenth International Conference on Machine Learning*, pp. 371–378, 1999.

Joseph Selvaraaj. Jaxinformarl: Multi-agent target mpe rl environments with gnns in jax, 2025. URL https://github.com/jselvaraaj/JaxInforMARL.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rye7knCqK7.

Ghada Sokar, Gintare Karolina Dziugaite, Anurag Arnab, Ahmet Iscen, Pablo Samuel Castro, and Cordelia Schmid. Continual learning in vision-language models via aligned model merging. *arXiv preprint arXiv:2506.03189*, 2025.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, Constantin Ruhdorfer, Bram Grooten, Yali Du, Andreas Bulling, Mykola Pechenizkiy, and Meng Fang. MEAL: A benchmark for continual multi-agent reinforcement learning, 2026. URL https://openreview.net/forum?id=I3W8PynQU0.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

Maciej Wolczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=5qsptDcsdEj.

Tianpei Yang, Heng You, Jianye Hao, Yan Zheng, and Matthew E Taylor. A transfer approach using graph neural networks in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 16352–16360, 2024.

Yuxuan Yi, Ge Li, Yaowei Wang, and Zongqing Lu. Learning to share in networked multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:15119–15131, 2022.

Deunsol Yoon, Sunghoon Hong, Byung-Jun Lee, and Kee-Eung Kim. Winning the l2{rpn} challenge: Power grid management via semi-markov afterstate actor-critic. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=LmUJqB1Cz8.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Lei Yuan, Lihe Li, Ziqian Zhang, Fuxiang Zhang, Cong Guan, and Yang Yu. Multiagent continual coordination via progressive task contextualization. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):6326–6340, 2024.

Daniele Zambon, Cesare Alippi, and Lorenzo Livi. Graph random neural features for distance-preserving graph representations. In *International Conference on Machine Learning*, pp. 10968–10977. PMLR, 2020.

Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *International conference on machine learning*, pp. 5872–5881. PMLR, 2018.

Chengqi Zheng, Haiyan Yin, Jianda Chen, Terence Ng, Yew-Soon Ong, and Ivor Tsang. Mastering continual reinforcement learning through fine-grained sparse network allocation and dormant neuron exploration. *arXiv preprint arXiv:2503.05246*, 2025.

Guangyao Zheng, Samson Zhou, Vladimir Braverman, Michael A Jacobs, and Vishwa Sanjay Parekh. Selective experience replay compression using coresets for lifelong deep reinforcement learning in medical imaging. In *Medical Imaging with Deep Learning*, pp. 1751–1764. PMLR, 2024.

Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4714–4722, 2021.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1):4, 2024.

A Theoretical Analysis of Forgetting

In this section, we provide a theoretical analysis of how forgetting emerges in a decoupled graph-based policy as in Eq. 4. For simplicity, we remove the time index t within a task and call $\boldsymbol{a} \doteq \boldsymbol{\pi}_k(\boldsymbol{o}, \mathcal{G}_h)$ the joint action taken by the joint policy $\boldsymbol{\pi}_k$ evaluated on task h (characterized by graph $\mathcal{G}_h$) after being trained on task k. Therefore, we have:

$$\boldsymbol{a} \doteq \boldsymbol{\pi}_k(\boldsymbol{o}, \mathcal{G}_h) = \{\pi_{\theta_k}(\boldsymbol{o}^i; \mathcal{G}_h)\}_{i=1}^{\mathcal{N}} \tag{6}$$

Given task indexes k and h, with $k > h$, forgetting on task h after training on task k is defined as the performance difference:

$$J_h(\boldsymbol{\pi}_k) - J_h(\boldsymbol{\pi}_h) \leq |J_h(\boldsymbol{\pi}_k) - J_h(\boldsymbol{\pi}_h)| \leq K^{\boldsymbol{\pi}} \|\boldsymbol{\pi}_k(\boldsymbol{o}; \mathcal{G}_h) - \boldsymbol{\pi}_h(\boldsymbol{o}; \mathcal{G}_h)\|_\infty \tag{7}$$

where we assumed a Lipschitz continuous objective and denoted as $K^{\boldsymbol{\pi}}$ the Lipschitz constant. If we substitute a generic decoupled policy $\boldsymbol{\pi} = \boldsymbol{\mu} \circ \boldsymbol{\psi}$ and use the triangle inequality (sup-norm sub-additivity), we get:

$$\begin{aligned} &J_h(\boldsymbol{\pi}_k) - J_h(\boldsymbol{\pi}_h) \leq K^{\boldsymbol{\pi}} \|\boldsymbol{\mu}_k(\boldsymbol{\psi}_k(\boldsymbol{o}; \mathcal{G}_h)) - \boldsymbol{\mu}_h(\boldsymbol{\psi}_h(\boldsymbol{o}; \mathcal{G}_h))\|_\infty \\ &\leq \|\boldsymbol{\mu}_k(\boldsymbol{\psi}_k(\boldsymbol{o}; \mathcal{G}_h)) - \boldsymbol{\mu}_k(\boldsymbol{\psi}_h(\boldsymbol{o}; \mathcal{G}_h))\|_\infty + \|\boldsymbol{\mu}_k(\boldsymbol{\psi}_h(\boldsymbol{o}; \mathcal{G}_h)) - \boldsymbol{\mu}_h(\boldsymbol{\psi}_h(\boldsymbol{o}; \mathcal{G}_h))\|_\infty \end{aligned} \tag{8}$$

In the last inequality, the two terms represent the performance drop caused by representation learning and action prediction, respectively.

B Extended Related Work

Graph-based MARL and Communication Graph-based MARL formalizes settings where the environment dynamics depend on the underlying graph. This has been adopted across several settings, e.g., where the dynamics of each agent depend on its neighbors in the graph (Hu et al., 2024a; Rashwan et al., 2026). The same graph possibly serves as the agent communication structure. Such problems have been extensively studied in networked systems (Zhang et al., 2018). For example, NeurComm (Chu et al., 2020) employs an effective communication protocol that prevents information loss during message aggregation. Conversely, BayesG (Duan et al., 2025) learns a context-aware interaction graph that reduces unnecessary message propagation. Other works address networked problems by either leveraging consensus-based mechanisms (Qu et al., 2019; Yi et al., 2022) or incorporating relational information (Lin et al., 2021; Qu et al., 2022). Communication MARL (Zhu et al., 2024) focuses on developing effective methods for improving agents' coordination through communication. Pioneering methods (Foerster et al., 2016; Sukhbaatar et al., 2016) focused on learning communication protocols end-to-end. Subsequent works (Jiang & Lu, 2018; Singh et al., 2019; Das et al., 2019; Hu et al., 2024b) employed message-broadcasting schemes that adaptively select communication channels. Other approaches employ GNNs in the decision-making process to learn message-passing policies (Jiang et al., 2020; Niu et al., 2021; Bettini et al., 2023; Marzi et al., 2025), possibly focusing on the zero-shot capabilities of the learned behaviors (Agarwal et al., 2020; Nayak et al., 2023). Here, the communication topology is either given by the environment or learned end-to-end, and does not necessarily determine the environment dynamics.

Continual RL In Continual RL (CRL; Khetarpal et al., 2022; Abel et al., 2023), the agent learns from a sequence of RL problems, aiming to effectively adapt to new tasks while retaining acquired knowledge from previously observed ones. CRL has been applied to different domains, including robotics (Powers et al., 2023) and medical imaging (Zheng et al., 2024). A common approach to address CRL problems is to leverage standard CL methods (Pan et al., 2025), such as replay-based mechanisms (e.g., AGEM; Chaudhry et al., 2019), parameter regularizations such as EWC (Kirkpatrick et al., 2017) or MAS (Aljundi et al., 2018), or parameter isolation (e.g., PackNet; Mallya & Lazebnik, 2018). The Continual MARL (CMARL; Tomilin et al., 2026) framework extends the CRL problem to systems of multiple cooperative agents, requiring novel methodologies tailored for this paradigm (Yuan et al., 2024).

Frozen Representations Frozen representations have proved their effectiveness as both purely random features (Pao et al., 1992; Jaeger & Haas, 2004; Rahimi & Recht, 2007) and as pre-trained features (Houlsby et al., 2019; Kolesnikov et al., 2019; Chen et al., 2020). In Reinforcement Learning (RL), random features have been used to improve exploration (Burda et al., 2019a;b), maintain accurate uncertainty estimates (Osband et al., 2018), or generate intrinsic learning signals (Chen et al., 2023). In contrast, pre-trained representations have been successfully applied to different domains, such as visual RL (Nair et al., 2023; Ma et al., 2023) or graph-based MARL (Amato et al., 2025). Frozen representations have also been explored in Continual Learning problems (Hayes & Kanan, 2020; Li & Zeng, 2023; McDonnell et al., 2023; Prabhu et al., 2024). Notably, Donghi et al. (2026) leverage randomized representations generated by different GNN backbones (Zambon et al., 2020; Navarin et al., 2023) in CL problems on graphs (Donghi et al., 2025). In Continual RL, several works rely on frozen representations or modules (Rusu et al., 2016; Powers et al., 2022; Nikishin et al., 2023; Hu et al., 2026) to mitigate forgetting and facilitate learning by reducing the number of learnable parameters (Zheng et al., 2025). Nonetheless, to the best of our knowledge, no existing approaches have investigated frozen GNN-based encoders for graph-based CMARL settings.

C GRAPH-BASED FORMATION (GRAFO) BENCHMARK

In this section, we present our novel graph-based MARL benchmark, named Graph-based Formation (GRAFO). Building on other formation environments (Agarwal et al., 2020), GRAFO is a collection of graph-based MARL tasks in which the agents' objective is to navigate to a target configuration that resembles the input graph. Notably, such configurations can be generated arbitrarily, making GRAFO a suitable benchmark for CGMARL problems.

C.1 SINGLE-TASK DESCRIPTION

In GRAFO, the goal of the agents is to navigate to a *partially unknown* target configuration, which depends on the input graph. A single task configuration is specified by:

- the adjacency matrix A of the graph, which influences the environment dynamics and possibly serves as the communication structure (Sec. 2).
- a tuple of target distances $\boldsymbol{g} = (\boldsymbol{g}_1, \dots, \boldsymbol{g}_N)$, indicating the relative position shifts with respect to the known landmark in the 2-dimensional plane; a target distance $\boldsymbol{g}_i = (0, 0)$ corresponds to the reference node, i.e., the node assigned to the known landmark.
- possibly, a set of node indices $\mathcal{K} \subset \mathcal{N}$ that are structurally equivalent in the graph.
- optionally, static graph (node and edge) features $\boldsymbol{X}$ and $\boldsymbol{E}$ (please refer to Sec. 3.1).

We report an example for a one-step transition with a 4-agent task in Fig. 5. The corresponding configuration is the following:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad \boldsymbol{g} = \Big((0,0), (1,0), (2,1), (2,-1)\Big) \qquad \mathcal{K} = \{3,4\} \subset \mathcal{N} = \{1,2,3,4\} \tag{9}$$

Similar to the Formation environment (Agarwal et al., 2020), all agents have access to the position of a reference landmark (green cross), which serves as the target position for a specific agent (green circle). In the example, agent 1 (green) is the reference node, as its goal shift is $\boldsymbol{g}_1 = (0, 0)$, meaning that its target position coincides with the known landmark. Notably, the reference agent is *unknown* to the team. Importantly, in GRAFO, the target positions (dashed crosses) of other agents must be deduced from the underlying graph (adjacency matrix A, and optionally, static graph features), rather than being explicitly provided. In other words, while agents in Formation must arrange themselves in a symmetric configuration around the landmark, in GRAFO, the graph structure is crucial for determining the target positions $\boldsymbol{g}$ and solving the task. We assess this claim in App. D.1.

Each agent's reward is the negative distance from its landmark, plus a one-time bonus whenever an agent reaches its target position. When two agents are structurally equivalent (i.e., symmetric under a graph automorphism), their target positions are assigned at each step to candidate landmarks using

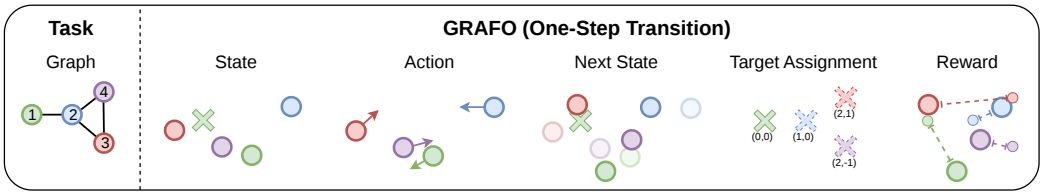


Figure 5: One-step transition in GRAFO. At the beginning of the episode, an initial state in which the agents (colored circles) and known landmark (solid, green cross) positions are randomly assigned is generated. Then, at each step, agents take actions depending on the current state. Each agent's reward is the negative of the distance to its corresponding target (dashed cross with the same color). In the case of two symmetric agents, i.e., with a non-empty set $\mathcal{K}$ of structurally-equivalent nodes, a linear sum assignment problem is solved to assign the targets.

a linear sum assignment problem that minimizes the total distance. In the example, agents 3 and 4 can be swapped without altering the graph topology, and therefore, they belong to the set $\mathcal{K}$ of structurally equivalent nodes. This implies that at each step, target positions $\boldsymbol{g}_3 = (2, 1)$ and $\boldsymbol{g}_4 = (2, -1)$ are assigned to agent 3 and 4 depending on the configuration that minimizes the total distance. Lastly, the position of the known landmark is randomly initialized at the beginning of each episode, and other target positions are assigned using $\boldsymbol{g}$ accordingly.

Graph-based MARL Formulation We note that the adjacency A and the target positions $\boldsymbol{g}$ determine the reward function $\mathcal{R}_{\mathcal{G}}$, which makes GRAFO a proper graph-based MARL problem (Eq. 2). Indeed, each agent's reward is defined based on the distance to its target position, which changes depending on A and $\boldsymbol{g}$. Consequently, standard GNN-based policies are likely to fail when applied in a zero-shot manner, and we assess this behavior in Sec. 5.1. The transition function, as well as the observation and action spaces, are identical to those of the Formation environment from Agarwal et al. (2020).

C.2 Continual GRAFO

In line with Definition 3.1, we can define a task sequence by simply considering a series of different graphs. This implies that we can procedurally generate (possibly infinite) CGMARL sequences by simply varying the task specification in Eq. 9. We now show how the different CGMARL settings (Sec. 3) can be induced in GRAFO. The complete sequences used in our experiments are reported in App. E.2.

Topology Switch In Topology Switch (Sec. 3.2.1), different tasks in the TS sequence are specified by different topologies, i.e., nodes and/or edges. For example, a 2-task TS sequence can be defined as:

$$A_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad \boldsymbol{g}_1 = \Big((0,0),(1,0),(2,0.5),(2,-0.5)\Big) \qquad \mathcal{K}_1 = \{3,4\}$$

$$A_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \boldsymbol{g}_2 = \Big((-1,0),(0,0),(1,0)\Big) \qquad \mathcal{K}_2 = \{1,3\}$$

where the first task has 4 agents, while the second one has 3 (please refer to GRAFO-4(1) and GRAFO-3 in Fig. 9 for a visual representation).

Connectivity Switch In Connectivity Switch (Sec. 3.2.2), different tasks in the CS sequence are specified by different edges, while the node set is fixed. For example, a 2-task CS sequence can be

defined as:

$$A_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \quad \boldsymbol{g}_1 = \Big((0,0),(1,0),(2,0),(3,0),(4,0.5),(4,-0.5)\Big) \quad \mathcal{K}_1 = \{5,6\}$$

$$A_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \quad \boldsymbol{g}_2 = \Big((0,0),(1,0),(2,0),(3,0.5),(4,0),(4,0.5)\Big) \quad \mathcal{K}_2 = \{4,5\}$$

where, in this case, the sequence is specified by 6 agents for both tasks (please refer to GRAFO-6(1) and GRAFO-6(2) in Fig. 9 for a visual representation).

Graph Features Switch In Graph Features Switch (Sec. 3.2.3), different tasks in the GFS sequence are specified by different graph features. For example, a 2-task GFS sequence can be defined as:

$$A_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad \boldsymbol{g}_1 = \Big((0,0),(1,0),(2,0.5),(2,-0.5)\Big) \quad \mathcal{K}_1 = \{3,4\}$$

$$A_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad \boldsymbol{g}_2 = \Big((0,0),(1,0),(2,1.5),(2,-1.5)\Big) \quad \mathcal{K}_2 = \{3,4\}$$

where, in this case, the sequence is characterized by 4 agents with fixed edges (please refer to GRAFO-4(1) and GRAFO-4(2) in Fig. 9 for a visual representation). We note that, in the GFS scenario, the topology is fixed across the sequence, and the graph features are used to distinguish different tasks. Therefore, in GRAFO, the target positions $\boldsymbol{g}$ must be encoded within the policy. This can be achieved by, e.g., providing the relative position shift as node features. In the example above, the first task would have node feature 1, while the second one would have node feature 2. Another option is to provide the target L^2-distances as edge features.

D Additional Results

D.1 Single-Task Performance in GRAFO

In Tab. 3, we report the performance achieved by InforMARL (App. E.1) on each of the single tasks of the TS sequence (App. E.2). In particular, we consider InforMARL and compare it with a variant where the policy does not rely on a GNN, i.e., without message exchange across agents. The results show that the graph-based version can correctly solve each task individually, while the variant without GNN fails to do so. This supports our claim that GNN-based policies are crucial for effectively solving individual graph-based tasks in the GRAFO benchmark.

Table 3: Solved percentage of InforMARL and the variant without GNN in the actor on the single tasks of the TS sequence (reported as median $\pm$ standard deviation of 5 seeds).

	GRAFO-3	GRAFO-4	GRAFO-5	GRAFO-6 (1)	GRAFO-6 (3)	GRAFO-7
InforMARL	**1** ± 0	**1** ± 0	**1** ± 0	**0.994** ± 0.009	**1** ± 0	**0.997** ± 0.007
InforMARL (no GNN)	0.531 ± 0.013	0.328 ± 0.091	0.297 ± 0.258	0 ± 0	0.172 ± 0.169	0 ± 0

D.2 Impact of Regularization Coefficients

To investigate the results achieved by EWC (M) in Fig. 3, we perform an additional experiment on the TS sequence to assess the impact of using two separate regularization coefficients. Specifically,

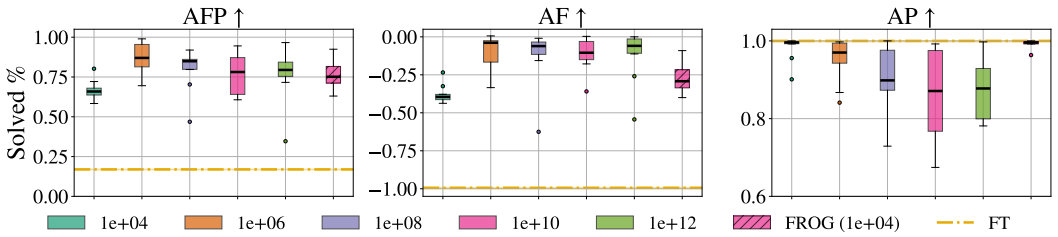


Figure 6: Boxplot showing the distribution of 10 independent seeds in the sensitivity analysis of λ_μ for EWC (M) in the TS sequence. Specifically, we consider different values for the regularization coefficient λ_μ of the action prediction module. The regularization coefficient of the graph encoder is fixed to $\lambda_\psi = 10^{10}$. When the coefficient for the action module is $\lambda_\mu = 10^{10}$ (pink box), the results coincide with those reported in Fig. 3. We also report the results achieved by FROG, where we remark that the graph encoder is kept fixed, i.e., the coefficient $\lambda_\mu = 10^4$ regularizes only the action prediction module. Boxes represent the IQR, horizontal lines indicate medians, and whiskers extend to $1.5 \times$ IQR.

we set the coefficient for the graph encoder $\lambda_\psi = 10^{10}$ (as in Fig. 3), which is the best value achieved during parameter tuning (refer to App. E.4.2). In contrast, we vary the coefficient λ_μ for the action prediction module from 10^4 to 10^{12}, and evaluate each configuration $(\lambda_\psi, \lambda_\mu)$ on the TS sequence. We note that, when $\lambda_\mu = 10^{10} = \lambda_\psi$ (pink box), the results are those reported in Fig. 3. Our hypothesis to explain the low plasticity of EWC (M) in Fig. 3 is that the graph encoder requires a much stronger regularization coefficient than the action prediction module, since EWC is not designed to preserve structural knowledge. Therefore, despite being a single value in practice, these coefficients lie on two different scales: large coefficients for the action module (large λ_μ) can deteriorate learning new tasks, whereas small coefficients for the GNN (small λ_ψ) cannot preserve past structural information.

The results in Fig. 6 support this hypothesis: the best tradeoff is achieved with $\lambda_\mu = 10^6$, which is much lower than the value associated with the graph encoder ($\lambda_\psi = 10^{10}$). Specifically, smaller regularization coefficients for the action module increase plasticity, but may suffer from forgetting for very low values (in this case, $\lambda_\mu = 10^4$). Conversely, strong regularization (large λ_μ) causes plasticity loss, i.e., the model struggles to learn new tasks. In principle, finding the right balance between the two coefficients could be done by tuning separate values, but this may be computationally prohibitive as it requires a larger grid search. Additionally, the results achieved in this sensitivity analysis were obtained by evaluating each configuration on the entire TS sequence, which is not representative of practice. Indeed, hyperparameter tuning is usually far more restrictive: the best configuration can typically be selected only through the performance on the first few tasks (Mesbahi et al., 2025), and we adopted this more realistic setting in the main experiments (further details in App. E.4.2). Therefore, this result further supports adopting FROG to address forgetting in CG-MARL sequence: since it uses a frozen graph encoder, only λ_μ needs to be tuned, also allowing for a much lower regularization coefficient ($\lambda_\mu = 10^4$). In this way, FROG can maintain plasticity (high AP) while achieving competitive performance (high AFP).

D.3 Comparison with Untrained FROG

As stated in Sec. 5.2, FROG relies on a frozen graph backbone to mitigate forgetting arising from structural changes. To achieve this, a possible design choice consists of freezing the graph encoder *before* learning the first task, i.e., after initializing the network. This is more general, as it does not adapt the GNN to a specific graph, and has proven to be effective in continual graph learning problems (Donghi et al., 2026). Nonetheless, relying on random graph features may not provide enough expressiveness to solve MARL tasks, as agents must explore possibly large state spaces using random signals only. This may hinder learning, and for this reason, the experiments reported in Sec. 6 are done by freezing the graph encoder *after* learning the first task. This enables learning an encoder that generates features that may also be beneficial for solving future tasks.

In this section, we compare FROG against the untrained variant (FROG (RND)), where the graph encoder is frozen at initialization, and the generated graph features are random. Results in Fig. 7

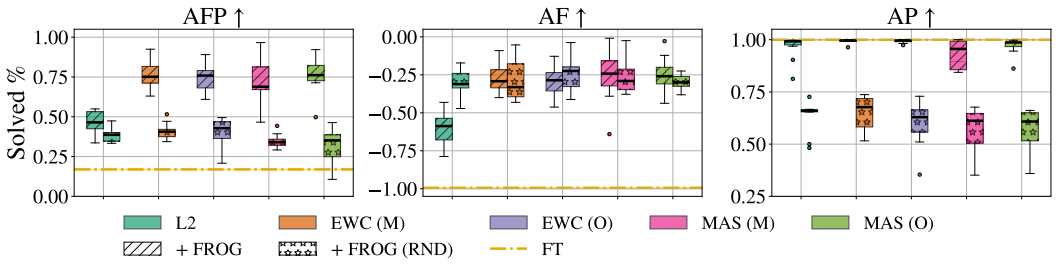


Figure 7: Boxplot showing the distribution of 10 independent seeds for FROG and FROG (RND) combined with different CL methods in the TS scenario. Boxes represent the IQR, horizontal lines indicate medians, and whiskers extend to $1.5 \times$ IQR.

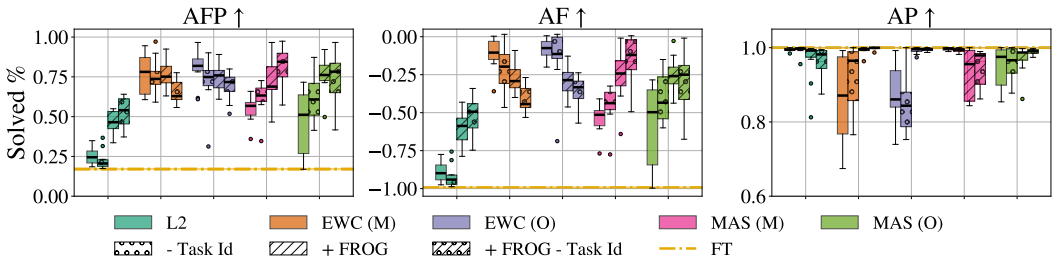


Figure 8: Boxplot showing the distribution of 10 independent seeds for different CL methods in the TS scenario with and without conditioning on task identities. Boxes represent the IQR, horizontal lines indicate medians, and whiskers extend to $1.5 \times$ IQR.

show that, for all CL models, FROG (RND) struggles to learn most tasks (low AP values), thus consistently achieving worse performance (AFP) compared to FROG. This supports our preliminary hypothesis that, in GRAFO, random graph features may not be expressive enough to solve hard tasks.

D.4 Impact of Task Identities

In our experiments (Sec. 6), we adopted the task-incremental learning paradigm, where we used separate task-specific heads and conditioned the observations on task identities via one-hot encoding. In Fig. 8, we assess the impact of task identities by comparing the results on the TS sequence with the variant that does not use the one-hot encoded identities as input. The AFP metric shows that removing task identities does not alter the overall trend when comparing standard CL methods with FROG. Within each method, consistently with Tomilin et al. (2026), results remain compatible, with performance differences falling mostly within the uncertainty range.

E Experimental Setting

E.1 InforMARL

InforMARL (Nayak et al., 2023) is a multi-agent actor-critic method based on multi-agent PPO (Yu et al., 2022) that leverages a GNN to exchange messages across connected agents. Specifically, at each step t, each agent takes an action $\boldsymbol{a}_t^i$:

$$\boldsymbol{a}_t^i = \mu(\boldsymbol{o}_t^i || \boldsymbol{x}_{agg}^i), \quad \text{where } \boldsymbol{x}_{agg}^i = \psi(\{\boldsymbol{o}^i\}_{i \in \mathcal{N}}; \mathcal{G}) \tag{10}$$

where the GNN encoder ψ follows the Unified Message Passing Model (UniMP; Shi et al., 2020) to share messages and update the node representation:

$$\boldsymbol{x}_i' = W_1 \boldsymbol{x}_i + \sum_{j \in \mathcal{B}(i)} \alpha_{i,j} W_2 \boldsymbol{x}_j, \quad \text{where } \alpha_{i,j} = \text{softmax}\left(\frac{(W_3 \boldsymbol{x}_i)^T (W_4 \boldsymbol{x}_j + W_5 \boldsymbol{e}_{ij})}{\sqrt{c}} \right) \tag{11}$$

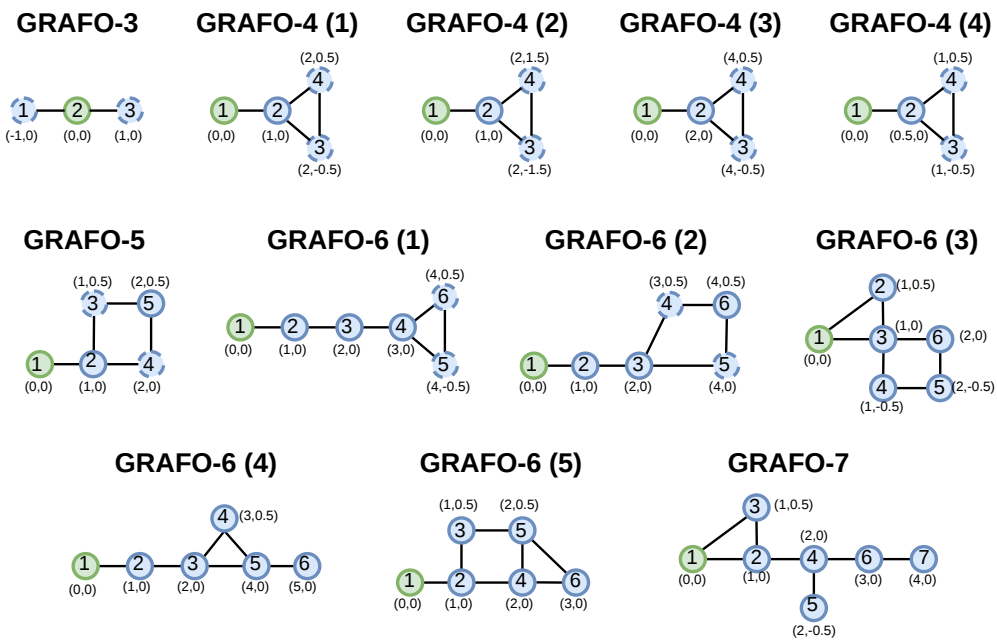


Figure 9: GRAFO tasks used to define the CGMARL sequences. The green circle is the reference node, i.e., the agent for which the target position corresponds to the reference landmark. Blue circles represent agents for which the corresponding target positions $\boldsymbol{g} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n)$ are unknown, and are reported as relative shifts (one per agent) with respect to the green node. Dashed blue nodes indicate agents in $\mathcal{K} \subset \mathcal{G}$ with a structurally equivalent role in the graph, i.e., that can be exchanged without altering the topology and for which a linear assignment with respect to the target positions is solved.

where $W_{k=1,2,3,4,5}$ are learnable parameter matrices, $\mathcal{B}(i)$ is the set of neighbors of agent i, and c is the output dimension of the layer. We concatenate features at each message-passing round to generate node representations at different scales. Conversely, the critic leverages a graph-level aggregation module to estimate the value function:

$$\boldsymbol{X}_{agg} = \frac{1}{n} \sum_{i \in \mathcal{N}} \boldsymbol{x}^i_{agg} \tag{12}$$

We remark that InforMARL operates on a relative frame of reference, where each agent sees other agents' positions and landmarks with respect to its position.

E.2 CGMARL Sequences

GRAFO Tasks In this Section, we report the GRAFO tasks and sequences used in our experiments. In particular, we report the single tasks in Fig. 9: we indicate the reference agent for which the target position coincides with the known landmark as the green circle, while blue ones are agents for which the targets are unknown. The (x, y) coordinates associated with each circle represent the relative shifts $\boldsymbol{g}$ with respect to the known landmark. Moreover, dashed blue circles represent structurally equivalent nodes, i.e., agents for which the target position can be exchanged.

Preliminary Experiments In the experiments conducted in Sec. 5.1, we used GRAFO-3 and GRAFO-5 from Fig. 9. For the continual learning (CONT.) experiment, we considered the following sequences:

$$\text{GRAFO-}(3,5) = [\text{GRAFO-3}; \text{GRAFO-5}]$$
$$\text{GRAFO-}(5,3) = [\text{GRAFO-5}; \text{GRAFO-3}]$$

Main Experiments For the main experiments (Sec. 6), we define CGMARL sequences using the graphs from Fig. 9. Specifically, the sequences used in our experiments are as follows:

$$\mathcal{T}_{TS} = [\text{GRAFO-5}; \text{GRAFO-6}(1); \text{GRAFO-7}; \text{GRAFO-4}; \text{GRAFO-3}; \text{GRAFO-6}(3)]$$
$$\mathcal{T}_{TS\,(\text{inc})} = [\text{GRAFO-3}; \text{GRAFO-4}; \text{GRAFO-5}; \text{GRAFO-6}(1); \text{GRAFO-6}(3); \text{GRAFO-7}]$$
$$\mathcal{T}_{TS\,(\text{dec})} = [\text{GRAFO-7}; \text{GRAFO-6}(3); \text{GRAFO-6}(1); \text{GRAFO-5}; \text{GRAFO-4}; \text{GRAFO-3}]$$
$$\mathcal{T}_{CS} = [\text{GRAFO-6}(1); \text{GRAFO-6}(2); \text{GRAFO-6}(3); \text{GRAFO-6}(4); \text{GRAFO-6}(5)]$$
$$\mathcal{T}_{GFS} = [\text{GRAFO-4}(1); \text{GRAFO-4}(2); \text{GRAFO-4}(3); \text{GRAFO-4}(4)]$$

We remind that $\mathcal{T}_{TS\,(\text{inc})}$ and $\mathcal{T}_{TS\,(\text{dec})}$ are defined by sorting the tasks of the TS sequence $\mathcal{T}_{TS}$ in incremental and decremental order with the number of agents. Moreover, the CS setting is characterized by a fixed number of nodes, which is fixed to 6 agents for the sequence $\mathcal{T}_{CS}$. Lastly, the GFS setting uses a fixed topology and varies only the graph features. For the $\mathcal{T}_{GFS}$ sequence, we consider a 4-agent graph with different target distances $\boldsymbol{g}$, and encode the L^2-distances between target positions of connected nodes as edge features.

E.3 Evaluation Metrics

Given a task sequence $\mathcal{T}$ with K tasks, let us denote $R_{i,j}$ as the evaluation performance on task j after training on task i. In GRAFO, this value is the percentage of solved tasks during evaluation, i.e., the percentage of episodes in which the agent team successfully managed to reach the target configuration. To evaluate the models, we use CL metrics commonly employed in the literature (Wolczyk et al., 2021; Caccia et al., 2022; Sokar et al., 2025). The metrics of interest are defined as follows.

Average Final Performance The Average Final Performance (AFP) measures the model's performance across all tasks after learning the last one:

$$AFP = \frac{1}{K}\sum_{j=1}^{K} R_{K,j} \tag{13}$$

Average Forgetting The Average Forgetting (AF) measures how performance on past tasks deteriorated after learning the final task (backward transfer). It is defined as:

$$AF = \frac{1}{K-1}\sum_{j=1}^{K-1} (R_{K,j} - R_{j,j}) \tag{14}$$

Average Plasticity The Average Plasticity (AP) measures the performance on a task immediately after learning it. It is defined as:

$$AP = \frac{1}{K}\sum_{j=1}^{K} R_{j,j} \tag{15}$$

E.4 Implementation Details

E.4.1 Software

We implemented the GRAFO benchmark on top of the *Jax* version (Selvaraaj, 2025) [3] of InforMARL (Nayak et al., 2023). To implement the continual learning methods, we adapted parts of the code from the repository [4] of the MEAL benchmark (Tomilin et al., 2026).

[3] https://github.com/jselvaraaj/JaxInforMARL
[4] https://github.com/TTomilin/MEAL

E.4.2 HYPERPARAMETERS

We report the hyperparameters used in our experiments in Tab. 4. To select them, we first tuned the parameters of the GNN-based policy (InforMARL) on a single task. These are kept fixed for all the CL methods, both with and without FROG. Tab. 3 in App. D.1 show that the chosen configuration can solve each task individually. Then, we followed k-percent tuning (Mesbahi et al., 2025) to choose the parameters for each different CL method. Specifically, for each configuration, we ran 3 seeds on the first 2 tasks of the TS sequence, and selected the hyperparameters with the best average final performance.

Table 4: Hyperparameters used in our experiments. We note that the parameters specific to the online variants of EWC and MAS (with and without FROG), i.e., online decay, importance episodes, importance steps, and normalization, are common for all methods.

Context	Parameter	Value
Architecture (InforMARL)	Activation	$ReLU$
	Num. of hidden layers (action module)	2
	Hidden feature dim (action module)	128
	Recurrent unit (action module)	False
	Num. of hidden layers (critic)	1
	Hidden feature dim (critic)	128
	Recurrent unit (critic)	True
	Num. of hidden layers (GNN)	2
	Hidden feature dim (GNN)	128
	Num. of graph attention layers (GNN)	4
	Num. of attention heads (GNN)	3
	Graph attention key dim (GNN)	16
Optimization (PPO)	Clip ϵ	0.2
	Learning rate	$5 \cdot 10^{-4}$
	Update epochs	4
	Discount factor	0.99
	GAE λ	0.95
	Entropy coeff.	0.001
	Value coeff.	0.01
	Rollout Length	128
	Minibatches per update	32
Env. (GRAFO)	Env. steps per task	$5 \cdot 10^6$
	Episode length	50
	Parallel environments	32
	Evaluation episodes	64
	Enable collisions	False
Continual Learning	Reg. coeff. (L2 / L2 + FROG)	10^3 / 10^2
	Reg. coeff. (EWC (M) / EWC (M) + FROG)	10^{10} / 10^4
	Reg. coeff. (EWC (O) / EWC (O) + FROG)	10^{11} / 10^5
	Reg. coeff. (MAS (M) / MAS (M) + FROG)	10^7 / 10^6
	Reg. coeff. (MAS (O) / MAS (O) + FROG)	10^9 / 10^4
	Regularize critic	False
	Regularize head	False
	Online decay	0.9
	Importance episodes	64
	Importance steps	50
	Normalize importance	False