GRAPH ACTOR-CRITIC: COMMUNICATION-AWARE MESSAGE PASSING IN GRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks (GNNs) have shown remarkable success on homophilic graphs, where connected nodes tend to share similar features or labels. However, their performance degrades significantly on heterophilic graphs, where connected nodes often have different characteristics. In this work, we propose a novel reinforcement learning (RL) framework for adaptive graph rewiring that learns to dynamically modify graph structure to improve GNN performance on heterophilic graphs. We formulate graph rewiring as a Markov Decision Process (MDP) where an RL agent learns to select actions that optimize a reward function that encourages diverse node representations, thereby minimizing oversmoothing, all without requiring labeled data. We evaluate our method on diverse graph datasets and show that our RL-based rewiring approach, *Graph Actor-Critic* (*GAC*), significantly outperforms standard GNNs and existing rewiring methods on heterophilic graphs, while maintaining competitive performance on homophilic graphs.

1 INTRODUCTION

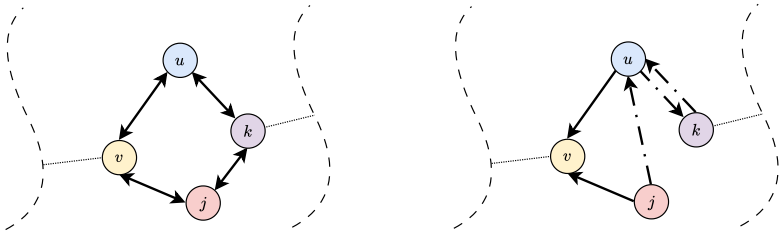
Graph neural networks (GNNs) have emerged as the de facto paradigm for learning on relational data, enabling state-of-the-art performance across molecular design, recommendation, program analysis, and physical reasoning (Polykovskiy et al., 2020; Hu et al., 2020; Robinson et al., 2024; Dwivedi et al., 2025). At their core, most GNNs implement *message passing* (Duvenaud et al., 2015; Li et al., 2016; Kipf & Welling, 2017; Wang et al., 2019), in which node representations are updated by aggregating and transforming information from neighbors through a stack of layers (Hamilton et al., 2017; Gilmer et al., 2017; Veličković et al., 2018).

While highly effective on homophilic graphs, where connected nodes tend to be similar, these methods struggle on heterophilic graphs, where naively aggregating all neighbors may dilute task-relevant signals, induce oversmoothing, and entangle conflicting local contexts (Li et al., 2018; Wu et al., 2021; Chen et al., 2020; Rusch et al., 2023; Wu et al., 2023). This tension has motivated topology learning (Gulcehre et al., 2017; Klicpera et al., 2019; Karhadkar et al., 2023), edge reweighting (Lai et al., 2020; Corso et al., 2020; Bevilacqua et al., 2022), rewiring (Alon & Yahav, 2021; Ma et al., 2019; Dai et al., 2022; Jamadandi et al., 2024; Zhang et al., 2024), and adaptive aggregation schemes (Veličković et al., 2018; Li et al., 2020; Finkelshtein et al., 2024). More recently, concurrent work has looked at heterophily-aware message passing and graph-level heterophily (Jamadandi et al., 2025; Ghazizadeh et al., 2025; Gu et al., 2025; Zheng et al., 2025; Hou et al., 2025), and has analyzed oversquashing, oversmoothing, and vanishing gradients in deeper architectures (Marisca et al., 2025; Sun et al., 2025; Arroyo et al., 2025).

In this work, we propose a novel interpretation of GNNs as a system of interacting agents that collectively determine how information propagates through the graph. We interpret each node as an adaptive communicator that can learn when and how much information to share based on its context. Intuitively, if we can learn a policy that dictates when and how to exchange information to balance the competing demands of expressivity and stability by pruning, modifying, or creating new edges on the graph (e.g. see Figure 1), then, as we will show in this paper, it can mitigate issues like oversmoothing by keeping the node similarities spread while preserving the efficiency of localized reasoning.

More formally, we introduce a novel reinforcement learning (RL) (Sutton et al., 1998) framework for adaptive GNN rewiring called *Graph Actor-Critic* or (*GAC*). In essence, this actor-critic framework parameterizes a node-level communication policy that outputs communication decisions for each node

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.



(a) Standard message passing

(b) Adaptive message passing

Figure 1: A subgraph where nodes pass messages to each other. **(a)** This sub-figure depicts standard message passing, which follows a fixed communication pattern. **(b)** This sub-figure depicts adaptive message passing, as enabled by our proposed Graph Actor-Critic (GAC) approach, which can remove or create new communication patterns. The dotted arrows indicate partial information sharing.

based on its current embedding, such that the agent jointly learns *how to communicate* and *what to represent*. Empirically, we find that our approach yields robust gains on heterophilic graphs, preserves competitive performance on homophilic graphs, and provides qualitative interpretability through the lens of learned communication. Beyond performance, our formulation advances a unifying view of message passing that bridges topology learning, adaptive aggregation, and reinforcement learning in a single, modular framework.

Our work is organized as follows: Section 2 reviews background on message-passing GNNs and RL. Section 3 presents GAC, including the MDP formulation for depth-wise communication, the various learning objectives, as well as a supporting theoretical analysis. Section 4 reports results on synthetic and real-world experiments. Finally, Section 5 summarizes our work and speculates on potential directions for future research.

2 Background

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2009; Gilmer et al., 2017) provide a general framework for learning over graph-structured data, where nodes exchange messages along edges to update their latent representations. Formally, given a graph, $\mathcal{G} \doteq (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (with node features $\mathcal{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$) and $\mathcal{E}$ is the set of edges, a message-passing neural network (MPNN) updates the node embeddings of node v, h_v, iteratively as:

$$h_v^{(\ell)} = \phi^{(\ell-1)}\Big(h_v^{(\ell-1)}, \psi^{(\ell-1)}\Big(\{\, h_u^{(\ell-1)} \mid u \in \mathcal{N}(v) \,\}\Big)\Big), \tag{1}$$

such that the model applies ℓ message-passing layers to update the node representations, where at layer ℓ the node v has d-dimensional embedding, such that $h_v^{(\ell)} \in \mathbb{R}^d$. Here, $\mathcal{N}(v)$ denotes the set of neighbors of the node v, and $\psi^{(\ell)}$, $\phi^{(\ell)}$ are learnable aggregation and update functions at layer ℓ, collectively referred to as the message passing operator, $\mathcal{M}^{(\ell)} \doteq \phi^{(\ell)} \circ \psi^{(\ell)}$. Popular realizations of GNNs include GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019), which differ in how they aggregate or weigh neighbor messages.

Despite their success, traditional GNNs often suffer from severe performance degradation when stacking multiple message-passing layers, a phenomenon widely known as *oversmoothing* (e.g. see Li et al. (2018); Oono & Suzuki (2020)). In particular, as the number of layers ℓ increases, the repeated application of the message passing operator, $\mathcal{M}^{(\ell)}$, acts as a form of "smoothing" over the graph topology. This causes the embeddings of nodes within the same connected component to become completely indistinguishable (or "collapse"), retaining only rudimentary structural information like node degrees, thereby severely limiting the model's expressive power for downstream tasks.

2.2 Reinforcement Learning

Reinforcement Learning (RL) (Sutton et al., 1998) formalizes sequential decision-making as an interaction between an agent and an environment defined by a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{R} \subset \mathbb{R}$ is a bounded set of rewards, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a probabilistic transition function that describes the dynamics of the environment, and $\gamma \in [0, 1]$ is a discount factor. At each step, n, the agent selects an action, $a^{(n)} \in \mathcal{A}$, based on its current state, $s^{(n)} \in \mathcal{S}$, transitions to the next state, $s^{(n+1)}$, according to $\mathcal{T}(s^{(n+1)}|s^{(n)}, a^{(n)})$, and receives a scalar reward, $r(s^{(n)}, a^{(n)}) \doteq r^{(n)} \in \mathcal{R}$. The agent's goal is to learn a policy, $\pi : \mathcal{S} \to \mathcal{A}$, that maximizes the expected discounted return, $\mathbb{E}[\sum_t \gamma^t r^{(t)}]$, which often takes the form of the *state-value function*, $V(s) \doteq \mathbb{E}[\sum_t \gamma^{(n)} r^{(n)} \mid S_o = s]$, or the *state-action value function*, $Q(s, a) \doteq \mathbb{E}[\sum_t \gamma^{(n)} r^{(n)} \mid S_0 = s, A_0 = a]$.

A central paradigm in reinforcement learning is the *actor–critic* framework (Konda & Tsitsiklis, 1999; Mnih et al., 2016), which decomposes learning into two coupled objectives: an actor (typically a parameterized policy) that determines what actions to take, and a critic (typically a parameterized value function) that estimates the expected discounted return. Typically, the parameterized policy is learned by minimizing some actor loss, $\mathcal{L}_{\text{actor}}$, and the parameterized value function is learned by minimizing some critic loss, $\mathcal{L}_{\text{critic}}$.

3 Graph Actor-Critic

In this section, we present our primary contribution: a novel RL framework for adaptive GNN rewiring called *Graph Actor-Critic* or (*GAC*). In essence, this framework parameterizes a node level communication policy that outputs communication decisions for each node based on its current embedding, such that the agent jointly learns *how to communicate* and *what to represent*. In Section 3.1, we formalize message passing in a GNN as an MDP. In Section 3.2, we perform a theoretical analysis of this MDP formulation. In Section 3.3, we introduce our Graph Actor-Critic architecture. In Section 3.4 we describe the architecture used to train Graph Actor-Critic.

3.1 GNN Message Passing as an MDP

To dynamically reshape message passing in a GNN, we cast the depth-wise computation of the GNN as a Markov Decision Process (MDP), such that at each layer, ℓ, the agent makes communication-based decisions for every node in the graph through its incoming and outgoing edges. We now formally define the various components of this MDP:

State-Space: The state (at a given layer, ℓ) is defined as follows:

$$s^{(\ell)} \doteq \big(\{h_v^{(\ell)}\}_{v \in \mathcal{V}}, \tilde{\mathcal{E}}\big), \tag{2}$$

where $\{h_v^{(\ell)}\}_{v \in \mathcal{V}}$ are the node embeddings (which capture the learned representations at the current layer), and $\tilde{\mathcal{E}} \doteq \{(\phi(u, v), w_{u \to v}, w_{v \to u})\}_{u,v \in \mathcal{V}}$ is the current graph structure which captures the connectivity pattern of the graph. Here, $\phi(u, v)$ denotes the edge index for edge (u, v), and $w_{u \to v}$, $w_{v \to u}$ denote the probability of message flow along edges (u, v) and (v, u), respectively. We denote the corresponding state-space as $\mathcal{S}$. Since evaluating $w_{u \to v}$ over all $O(|\mathcal{V}|^2)$ node pairs is computationally prohibitive, $\tilde{\mathcal{E}}$ is restricted at each layer to the union of the original edges $\mathcal{E}$ and a k-nearest-neighbor candidate set $\mathcal{E}_{\text{KNN}}^{(\ell)}$ (defined below), rather than the full pairwise set.

Action-Space: The action-space is defined by two fundamental communication primitives: *send* and *receive*. More formally, for each node, u, the agent selects a discrete action, $a_u^{(\ell)}$, where

$$a_u^{(\ell)} \in \mathcal{A}_u \doteq \{\text{Send-and-Receive, Send-Only, Receive-Only, Idle}\}. \tag{3}$$

We denote the joint action taken across all nodes (at a given layer, ℓ) as $\mathbf{a}^{(\ell)} \doteq \{a_u^{(\ell)}\}_{u \in \mathcal{V}}$ with corresponding action-space $\mathcal{A}$.

Transition Dynamics: As discussed in Section 2.1, the node embeddings are updated based on the message-passing operator, $\mathcal{M}$. The graph structure is updated based on the probability of taking a

particular node-specific action, $a_u^{(\ell)}$, which we can extract from the policy. More formally, for each node, u, we can extract the probability of sending, $P(\text{Send})_u^{(\ell)}$, and receiving, $P(\text{Receive})_u^{(\ell)}$, from the policy, and then map these probabilities directly to continuous, node-level gating variables:

$$\alpha_u^{(\ell)} \doteq P(\text{Send})_u^{(\ell)} \quad \text{and} \quad \beta_u^{(\ell)} \doteq P(\text{Receive})_u^{(\ell)}, \tag{4}$$

where $\alpha_u^{(\ell)} \in [0, 1]$ controls the extent to which node u sends messages to its neighbors, and $\beta_u^{(\ell)} \in [0, 1]$ controls the extent to which node u receives messages. When $\beta_v^{(\ell)} = 0$, node v does not receive any messages from u. When $\beta_v^{(\ell)} = 1$, it fully receives the incoming message from u. Values between 0 and 1 represent partial or weighted message reception.

The weight of the message passed along any directed edge, $w_{u\to v}^{(\ell+1)}$, can then be computed as the joint probability that the source node sends and the target node receives, such that:

$$w_{u\to v}^{(\ell+1)} = \alpha_u^{(\ell)} \cdot \beta_v^{(\ell)} \quad \text{and} \quad w_{v\to u}^{(\ell+1)} = \alpha_v^{(\ell)} \cdot \beta_u^{(\ell)}. \tag{5}$$

It then immediately follows that our transition dynamics follow a Markovian depth-wise evolution: the next state $s^{(\ell+1)}$ depends only on the current state $s^{(\ell)}$ and the joint action $\mathbf{a}^{(\ell)}$ through one message-passing step. This is formalized as Lemma 3.1 below:

Lemma 3.1 (Markov Property of Depth-Wise Message Passing)**.** *The state evolution induced by the message-passing operator $\mathcal{M}^{(\ell)}$ and joint action $\mathbf{a}^{(\ell)}$ satisfies the Markov property:*

$$\mathcal{T}(s^{(\ell+1)}|s^{(0)} \dots s^{(\ell)}, \mathbf{a}^{(0)} \dots \mathbf{a}^{(\ell)}) = \mathcal{T}(s^{(\ell+1)}|s^{(\ell)}, \mathbf{a}^{(\ell)}). \tag{6}$$

Reward Function: We propose a label-free reward function that evaluates the quality of the node embeddings by measuring how well they preserve discriminability between connected nodes while avoiding the collapse that characterizes oversmoothing. This signal guides the policy to learn which communication patterns lead to effective representations. We begin by defining the various components used to calculate the reward:

Definition 3.2 (Edge Difference)**.** *Let $\bar{h}_v^{(\ell)} \doteq h_v^{(\ell)}/\|h_v^{(\ell)}\|_2$ denote the L_2-normalized node embeddings at layer ℓ. For each edge (u, v), we define the edge difference, $\Delta_{uv}^{(\ell)}$, as follows:*

$$\Delta_{uv}^{(\ell)} \doteq \bar{h}_u^{(\ell)} - \bar{h}_v^{(\ell)}. \tag{7}$$

The edge difference, $\Delta_{uv}^{(\ell)}$, captures the dissimilarity between connected nodes. In heterophilic graphs, connected nodes often belong to different classes or have different properties, so we want $\Delta_{uv}^{(\ell)}$ to be large (i.e., nodes should be distinguishable despite being connected).

We next seek to quantify heterophily by encouraging high variance of edge differences (encouraging diversity and discriminability) and penalizing neighbor similarity (discouraging oversmoothing). To this end, we can first define the variance of edge differences across the graph as follows:

$$\text{Var}_\Delta^{(\ell)} \doteq \mathbb{E}_{(u,v)\sim\mathcal{E}}\big[\|\Delta_{uv}^{(\ell)}\|_2^2\big] - \big\|\mathbb{E}_{(u,v)\sim\mathcal{E}}[\Delta_{uv}^{(\ell)}]\big\|_2^2, \tag{8}$$

where the expectation is over edges in the graph. Next, to discourage oversmoothing, we define an edge similarity penalty that measures how directionally aligned connected nodes are to each other. Because we utilize the L_2-normalized embeddings, this reduces to the expected dot product:

$$\text{Sim}^{(\ell)} \doteq \mathbb{E}_{(u,v)\sim\mathcal{E}}\big[\bar{h}_u^{(\ell)} \cdot \bar{h}_v^{(\ell)}\big]. \tag{9}$$

Finally, we can combine these two components to define our reward function:

$$r^{(\ell)} = \text{Var}_\Delta^{(\ell)} - \lambda \cdot \text{Sim}^{(\ell)} \tag{10}$$

where $\lambda \geq 0$ is a hyperparameter that controls the trade-off between variance and similarity penalties. In Appendix A, we decompose the variance term in Equation equation 10 into a more tractable form.

3.2 Theoretical Analysis

We now provide theoretical justification for our approach. First, we show that our MDP formulation of depth-wise message passing in a GNN admits an optimal message-passing policy (Theorem 3.3). We then justify our choice of reward function by showing that it promotes pruning edges that have low diversity (Proposition 3.4), and how this pruning can prevent oversmoothing (Corollary 3.5).

Theorem 3.3 (Existence of an Optimal Message-Passing Policy)**.** *Consider the MDP formulation described by Equations equation 2 through equation 34 for a finite graph, $\mathcal{G}$, with bounded initial node features. Given bounded rewards and continuous, linearly bounded message-passing operators, there exists an optimal policy, $\pi^* : \mathcal{S} \to \mathcal{A}$, that maximizes the expected discounted return, $J(\pi) = \mathbb{E}_\pi \left[\sum_{\ell=0}^{L-1} \gamma^\ell r^{(\ell)}\right]$.*

Proof. In Lemma A.1, we show that the state and action spaces are bounded and compact. Similarly, the boundedness of the reward function and the continuity of both the transition dynamics and the reward function can be easily verified. As such, by *Bellman optimality* (Puterman, 1994), we can establish the existence of an optimal policy via backward induction over a finite horizon L:

At the terminal layer, $\ell = L - 1$, the optimal value function is defined as:

$$V^*(s^{(L-1)}) = \max_{\mathbf{a}^{(L-1)} \in \mathcal{A}} r^{(L-1)}(s^{(L-1)}, \mathbf{a}^{(L-1)}). \tag{11}$$

This maximum is trivially guaranteed to exist because the joint action-space, $\mathcal{A}$, is finite and discrete, and the reward function is bounded on the compact state-space, $\mathcal{S}$. For earlier layers, $\ell = L-2, \ldots, 0$, assume the optimal value function at the next layer, $V^*(s^{(\ell+1)})$, exists. Because the GNN message-passing update is deterministic, the action-value function simplifies to:

$$Q^*(s^{(\ell)}, \mathbf{a}^{(\ell)}) = r^{(\ell)}(s^{(\ell)}, \mathbf{a}^{(\ell)}) + \gamma V^*\big(\mathcal{T}(s^{(\ell)}, \mathbf{a}^{(\ell)})\big), \tag{12}$$

where $\mathcal{T}$ represents the deterministic state transition and $\gamma \in (0, 1]$ is the discount factor. Since the action-space, $\mathcal{A}$, is finite, the maximum over the action-space,

$$V^*(s^{(\ell)}) = \max_{\mathbf{a}^{(\ell)} \in \mathcal{A}} Q^*(s^{(\ell)}, \mathbf{a}^{(\ell)}) \tag{13}$$

always exists. This completes the induction. Thus, by Bellman optimality, there exists a deterministic policy, $\pi^*(s^{(\ell)}) \in \arg\max_{\mathbf{a} \in \mathcal{A}} Q^*(s^{(\ell)}, \mathbf{a})$, that maximizes $J(\pi)$. This completes the proof. □

Proposition 3.4 (Edges that promote collapse are pruned.)**.** *Assume the reward is edge-factorized and upper bounded as $\|\Delta_{uv}^{(\ell)}\|_2^2$. Suppose that the policy performs ascent on $\mathbb{E}[\sum_\ell \gamma^\ell r^{(\ell)}]$. If for some edge, (u, v), we have $\|\Delta_{uv}^{(\ell)}\|_2^2 < \lambda(\bar{h}_u^{(\ell)} \cdot \bar{h}_v^{(\ell)})$, then any ascent method with a monotone update will eventually decrease $w_{u\to v}^{(\ell+1)}$ and $w_{v\to u}^{(\ell+1)}$ to 0, thereby suppressing communication along (u, v).*

Proof. Taking the partial derivative of the edge-factorized reward with respect to $w_{u\to v}^{(\ell+1)}$ yields:

$$\frac{\partial r^{(\ell)}}{\partial w_{u\to v}^{(\ell+1)}} = \frac{\partial}{\partial w_{u\to v}^{(\ell+1)}} \sum_{(u',v')} w_{u'\to v'}^{(\ell)} \left(\|\Delta_{u'v'}^{(\ell)}\|_2^2 - \lambda(\bar{h}_{u'}^{(\ell)} \cdot \bar{h}_{v'}^{(\ell)})\right) = \|\Delta_{uv}^{(\ell)}\|_2^2 - \lambda(\bar{h}_u^{(\ell)} \cdot \bar{h}_v^{(\ell)}). \tag{14}$$

As such, if the stated condition holds, then the derivative is strictly negative, which implies that the edge weight will eventually decrease to 0. Taking the partial derivative with respect to $w_{v\to u}^{(\ell+1)}$ yields the same result. This completes the proof. □

In words, Proposition 3.4 reveals that gradient ascent on our proposed reward function will decrease edge weights in situations when we have low diversity relative to similarity (and vice-versa). This means that in oversmoothed regions where $\bar{h}_u^{(\ell)} \approx \bar{h}_v^{(\ell)}$, we have $\|\Delta_{uv}^{(\ell)}\|_2^2 \approx 0$, $\Delta_{uv}^{(\ell)} \approx \mathbf{0}$, and $\mathrm{Sim}(\bar{h}_u^{(\ell)}, \bar{h}_v^{(\ell)}) \approx 1$, which results in a negative gradient. As such, a policy that maximizes our proposed reward function will learn to prune such edges, thereby breaking the feedback loop that can cause the representations to collapse. Given this result, we arrive at Corollary 3.5:

Corollary 3.5 (Disconnected nodes prevent representation collapse)**.** *If node v disconnects at layer ℓ due to the relevant edge weights being decreased to 0 (see Proposition 3.4), then its embedding remains fixed, such that $h_v^{(\ell+1)} = h_v^{(\ell)}$. More generally, if a subset $\mathcal{V}' \subseteq \mathcal{V}$ disconnects through such actions, then $h_v^{(\ell+1)} = h_v^{(\ell)}$ for all $v \in \mathcal{V}'$.*

Proof. Let $\mathbf{h}^{(\ell)} \in \mathbb{R}^{|\mathcal{V}| \times d}$ denote the stacked matrix of node embeddings at layer ℓ. The update for node v under message passing is:

$$h_v^{(\ell+1)} = (\mathcal{M}^{(\ell)}\mathbf{h}^{(\ell)})_v = \sum_{u \in \mathcal{V}} (\mathcal{M}^{(\ell)})_{vu} h_u^{(\ell)} \tag{15}$$

$$= \sum_{u \in \mathcal{V}} \frac{w_{v \to u}^{(\ell)}}{\sum_{u' \in \mathcal{V}} w_{v \to u'}^{(\ell)}} h_u^{(\ell)}. \tag{16}$$

When v is isolated, $w_{v \to u}^{(\ell)} = 0$ for all $u \neq v$. By convention, we set $(\mathcal{M}^{(\ell)})_{vv} = 1$ in this case (indicating a self-loop), thus:

$$h_v^{(\ell+1)} = 1 \cdot h_v^{(\ell)} + \sum_{u \neq v} 0 \cdot h_u^{(\ell)} = h_v^{(\ell)} \tag{17}$$

which yields the desired result for node v. The general case follows by applying the above argument to each $v \in \mathcal{V}'$ independently. This completes the proof. □

3.3 Communication-Aware Message Passing

Having formalized and justified depth-wise message passing in a GNN as an MDP, we now introduce our Graph Actor-Critic (GAC) framework (see Figure 2). In essence, the aim of GAC is to learn a policy that maximizes the discounted return associated with the reward function in Equation equation 10. To this end, we adopt an actor-critic architecture, where the actor selects actions at each layer, while the critic evaluates the quality of the actions through temporal difference (TD) learning, such that the agent learns to discover communication patterns that preserve node discriminability throughout the graph.

Beyond gating messages along existing edges, GAC can also introduce new edges that were absent from the original graph, as illustrated in Figure 2. Since letting every node pair compete for a communication decision is computationally prohibitive, we restrict candidate new edges to a k-nearest-neighbor (kNN) graph computed in embedding space at each layer. Specifically, at layer ℓ, for each node u we identify its k nearest neighbors under cosine similarity between L_2-normalized embeddings $\bar{h}_u^{(\ell)}$ and $\bar{h}_v^{(\ell)}$:

$$\mathcal{E}_{\text{KNN}}^{(\ell)} \doteq \left\{ (u, v) : v \in \text{top-}\, k_{v \in \mathcal{V} \setminus \{u\}}\ \bar{h}_u^{(\ell)} \cdot \bar{h}_v^{(\ell)} \right\}, \tag{18}$$

Because embeddings evolve across layers, this candidate set is recomputed at every layer, allowing GAC to propose different new edges as node representations change. The graph structure used by the actor and critic at layer ℓ is then the union of the original edges and this candidate set, $\mathcal{E} \cup \mathcal{E}_{\text{KNN}}^{(\ell)}$, over which the send/receive gates from the action space (defined above) are applied: an edge $(u, v) \in \mathcal{E}_{\text{KNN}}^{(\ell)} \setminus \mathcal{E}$ is effectively *created* whenever $w_{u \to v}^{(\ell+1)} = \alpha_u^{(\ell)} \cdot \beta_v^{(\ell)} > 0$, and otherwise remains absent.

Given the large size of the state and action-spaces, we utilize a parameterized actor and critic. Moreover, we utilize a *node-specific* critic, rather than a graph-specific critic (over the joint action $\mathbf{a}^{(\ell)}$), to leverage the localized structure of the graph. More formally, the critic (parameterized by ζ) evaluates the discrete communication choice of each node individually, yielding a node-level value denoted as $Q_\zeta(s^{(\ell)}, a_u^{(\ell)})$. By evaluating actions at the node level, we ensure that the critic focuses on localized communication dynamics, rather than the dynamics governed by the (potentially computationally prohibitive) joint action space. More formally, we make the following approximation:

$$Q(s^{(\ell)}, \mathbf{a}^{(\ell)}) \approx \sum_{u \in \mathcal{V}} Q_\zeta(s^{(\ell)}, a_u^{(\ell)}). \tag{19}$$

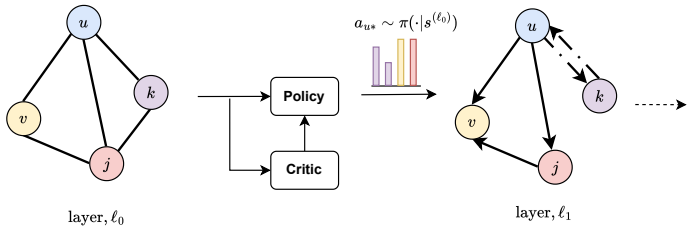


Figure 2: Communication-aware adaptive message passing induced by our Graph Actor-Critic (GAC) framework. Starting from an input graph (left), a learned policy assigns layer-wise edge actions for each node to modulate information flow between the edges, including new edges proposed via k-nearest-neighbor search in embedding space.

To train the (node-specific) critic, we estimate the quality of the node-specific actions taken by the actor by minimizing the following loss, computed as the mean squared error over all nodes and GNN layers:

$$\mathcal{L}_{\text{critic}} = \frac{1}{L}\sum_{\ell=0}^{L-1} \mathbb{E}_{u\sim\mathcal{V}}\left[\left\| y_u^{(\ell)} - Q_\zeta\big(s^{(\ell)}, a_u^{(\ell)}\big)\right\|_2^2\right]. \tag{20}$$

where $y_u^{(\ell)} = r^{(\ell)} + \gamma \max_{a'} Q_\zeta(s^{(\ell+1)}, a')$, $a' \in \mathcal{A}_u$, is the temporal difference (TD) target.

Simultaneously, at a given layer, ℓ, the actor, π, produces communication decisions for all nodes (i.e., it selects the joint action, $\mathbf{a}^{(\ell)}$). To train the actor, we utilize an advantage-based policy gradient formulation. In particular, we measure how much better (or worse) the achieved reward is compared to the critic's baseline prediction by estimating the *advantage*, $\mathrm{A}_u^{(\ell)} \doteq y_u^{(\ell)} - Q_\zeta(s^{(\ell)}, a_u^{(\ell)})$ (Mnih et al., 2016). Moreover, to encourage exploration, we incorporate an entropy regularization term, $\mathcal{H}(\pi(\cdot \mid s^{(\ell)}))$. More formally, the actor loss is defined as follows:

$$\mathcal{L}_{\text{actor}} = -\frac{1}{L}\sum_{\ell=0}^{L-1} \mathbb{E}_{u\sim\mathcal{V}}\Big[\log \pi\big(a_u^{(\ell)} \mid s^{(\ell)}\big)\, \mathrm{A}_u^{(\ell)} + \eta\mathcal{H}\big(\pi(\cdot \mid s^{(\ell)})\big)\Big]. \tag{21}$$

where η is a hyperparameter controlling the strength of the entropy penalty.

3.4 Training Architecture

To train our Graph Actor-Critic agent, we adopt a two-stage training paradigm. First, we learn communication policies using the actor-critic framework described in Section 3.3. Then, we freeze these policies for task-specific training. More specifically, we freeze the actor, π, and train only the environment GNN network parameters, θ_{env}, and task head parameters, θ_{task}, using a supervised loss on labeled data. This ensures that the learned graph structure is preserved and serves as an inductive bias for downstream tasks. Formally, given frozen policy parameters, we optimize:

$$\min_{\theta_{\text{env}},\theta_{\text{task}}} \mathcal{L}_{\text{task}}\big(\mathcal{X}, \tilde{\mathcal{G}}, y_{\text{task}}; \theta_{\text{env}}, \theta_{\text{task}}\big), \tag{22}$$

where $\mathcal{X}$ denotes the graph node features, $\tilde{\mathcal{G}}$ is the reconstructed graph (based on actions from the policy), and y_{task} is the task label. This two-stage approach prevents the policy from collapsing to homophilic patterns when exposed to labels, while allowing task-specific representation learning. Altogether, this approach allows the agent to jointly learn both *how to communicate* (by learning a message-passing policy) and *what to represent* (in the task-specific training).

4 Numerical Experiments

We evaluate our approach through two complementary experimental settings. First, we apply GAC on controlled toy graphs to build an intuitive understanding of how our approach operates. Second, we evaluate GAC on real-world datasets, and compare its performance against other methods.

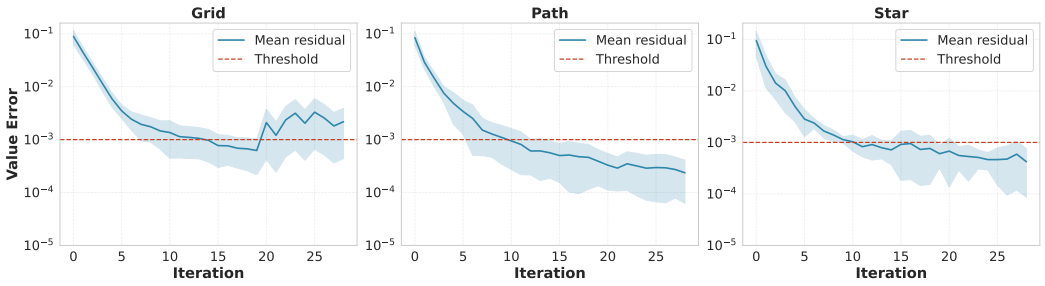


Figure 3: Convergence of value function for different synthetic heterophilic graphs with a threshold of 10^{-3}. Grid graph is of size 5x5, Path graph has 5 nodes and Star graph has 4 leaf nodes and one central node.

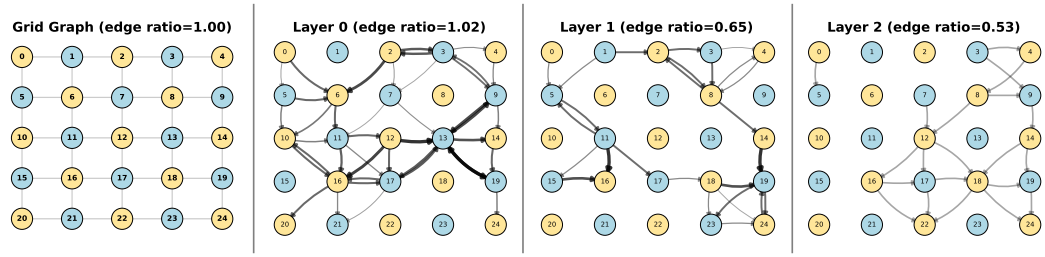


Figure 4: Adaptive edge modification in GNN layers. GAC modifies the graph structure at layers 0, 1, and 2 by creating, removing or downweighting edges based on learned policy. Edge ratio indicates the fraction of edges retained, while edge thickness and opacity reflect learned edge weights.

4.1 Toy Experiments

In this set of experiments, we consider three canonical graph families: *Path graph*, *Star graph*, and *Grid graph*. *Path graph* is a linear chain with alternating class labels (thereby creating 100% edge heterophily) that tests whether the policy can learn sequential information flow. *Star graph* consists of a central hub (class 0) connected to all peripheral nodes (class 1) (also yielding 100% heterophily) that tests selective broadcasting from a single bottleneck. *Grid graph* consists of a 2D lattice with checkerboard labeling (≈90% heterophily) that tests coordinated send-receive decisions in locally heterophilic neighborhoods. Node features are drawn from class-conditional Gaussians, such that we sample class prototypes $\mu_c \sim \mathcal{N}(0, 4I)$, then generate $x_u \sim \mathcal{N}(\mu_{y_u}, 0.25I)$ for each node u. This ensures features are informative but not trivially separable.

Figures 3 and 4 show the results of these experiments. In particular, Figure 3 shows that the Bellman residual (i.e., value function error) decreases rapidly and stabilizes below a pre-determined error threshold, $\epsilon = 10^{-3}$, thereby showing that GAC can converge to ϵ-optimal policies. The practical impact is visualized in Figure 4, which shows how GAC adaptively modifies a 5×5 *Grid graph* across layers 0–2. The policy progressively sparsifies the graph, reducing edge retention from 100% to approximately 60% by layer 2, while varying edge weights based on local embedding similarity.

4.2 Real-World Dataset Experiments

Next, we evaluate GAC on real-world datasets. First, following Platonov et al. (2023), we report *Accuracy* for the Roman Empire and Amazon Ratings datasets and *ROC-AUC* for the Minesweeper, Tolokers, and Questions datasets. All metrics are computed on the held-out test portion of each split using the model checkpoint selected by validation performance. We evaluate GAC against classical baselines: GCN, GraphSAGE, GAT, GT (Shi et al., 2021), and GT-sep. For more comprehensive evaluation, we also compare against CoGNN (Finkelshtein et al., 2024), FoSR-GCN (Karhadkar et al., 2023), ProxyGAP (Jamadandi et al., 2024), PolicyGNN (Lai et al., 2020) and BernNet (He et al., 2021). For training the models, we take the data processing and split definitions from Platonov et al. (2023). In Table 1, we summarize node classification results (mean ±

std over splits). As shown in the table, GAC is the best performing method on all datasets. Averaged across datasets, this corresponds to an overall gain of about +2.00 points over the best baseline.

Table 1: Results on node classification. Best in bold, second-best underlined.

	ROMAN EMPIRE	AMAZON RATINGS	MINESWEEPER	TOLOKERS	QUESTIONS
MLP	66.13 ± 0.49	41.67 ± 0.48	80.19 ± 0.26	78.16 ± 0.08	70.83 ± 1.03
GCN	72.46 ± 0.74	48.03 ± 0.76	89.57 ± 0.53	78.72 ± 0.87	69.21 ± 1.27
FoSR	73.60 ± 1.11	49.68 ± 0.73	90.15 ± 0.55	79.23 ± 1.21	74.51 ± 1.44
GResNet-SGC	73.85 ± 0.39	50.02 ± 0.65	70.92 ± 0.86	80.53 ± 0.94	74.12 ± 1.01
ProxyGap	77.54 ± 0.74	49.75 ± 0.46	89.59 ± 0.50	77.34 ± 0.76	70.45 ± 1.24
GAT	80.79 ± 0.39	48.46 ± 0.35	91.07 ± 0.53	79.84 ± 1.36	74.77 ± 1.37
PolicyGNN	83.60 ± 1.32	47.61 ± 0.33	88.25 ± 0.65	78.42 ± 1.23	73.21 ± 1.54
GraphSAGE	84.29 ± 0.54	53.06 ± 0.54	91.60 ± 0.55	79.04 ± 1.11	68.27 ± 1.70
GT	85.02 ± 0.47	51.35 ± 0.70	91.11 ± 0.52	79.71 ± 1.21	70.51 ± 1.33
BernNet	85.39 ± 1.14	51.44 ± 0.79	90.45 ± 0.43	78.43 ± 1.21	72.51 ± 1.34
GT-sep	85.63 ± 0.44	51.85 ± 0.47	90.58 ± 0.57	80.78 ± 1.54	77.59 ± 1.24
GAT-sep	87.53 ± 0.63	52.32 ± 0.61	90.83 ± 0.54	79.50 ± 1.29	73.13 ± 1.21
CoGNN	89.59 ± 0.69	50.49 ± 0.67	91.19 ± 0.27	81.27 ± 1.10	73.05 ± 1.37
GAC	**91.79 ± 0.90**	**53.94 ± 0.57**	**94.09 ± 0.39**	**84.54 ± 0.77**	**78.76 ± 1.13**

Next, Table 2 reports graph-classification performance (mean *Accuracy* ± std over splits) on six benchmark datasets spanning social networks (IMDB-B, IMDB-M, REDDIT-B) and bio/chemical graphs (NCI1, PROTEINS, ENZYMES). Classical graph classification baselines (DGCNN, DiffPool, ECC, GIN, and GraphSAGE) provide competitive performance but exhibit missing entries on some datasets due to model or resource limitations (e.g., `OOR`).

Table 2: Results on graph classification. Best in bold, second-best is underlined.

	IMDB-B	IMDB-M	REDDIT-B	NCI1	PROTEINS	ENZYMES
DGCNN	53.10 ± 3.62	38.08 ± 3.63	77.79 ± 1.10	64.65 ± 2.39	72.12 ± 3.49	33.27 ± 6.4
DiffPool	67.87 ± 2.61	43.65 ± 4.31	74.21 ± 4.69	70.31 ± 2.51	73.25 ± 3.72	63.00 ± 4.81
ECC	65.23 ± 2.65	41.23 ± 4.21	OOR	72.21 ± 2.45	71.35 ± 2.21	27.79 ± 3.24
GIN	69.14 ± 4.35	45.53 ± 4.10	72.36 ± 3.64	76.64 ± 1.21	74.34 ± 3.25	63.16 ± 6.34
GraphSAGE	67.41 ± 3.10	46.92 ± 3.17	74.55 ± 2.67	75.30 ± 3.21	71.19 ± 2.30	59.99 ± 4.82
PolicyGNN	68.35 ± 2.65	45.22 ± 2.07	77.24 ± 3.10	73.46 ± 2.11	69.81 ± 2.05	61.20 ± 3.62
CoGNN	72.41 ± 2.10	49.32 ± 2.78	87.89 ± 1.50	**77.20 ± 2.40**	70.40 ± 1.15	68.40 ± 2.62
GAC	**73.80 ± 2.30**	**52.65 ± 2.37**	**89.00 ± 1.23**	74.34 ± 3.21	**74.77 ± 1.77**	**71.67 ± 2.33**

Finally, Table 9 (Appendix C) reports performance on standard homophilic benchmarks (CITESEER, PUBMED, CORA), where connected nodes tend to share labels. While GAC is designed primarily for heterophilic settings, it still maintains competitive performance on homophily-dominated graphs, achieving results comparable to or exceeding most baselines. See Appendix B for additional details related to the experiments. See Appendix C for additional results and discussions.

5 CONCLUSION

In this work, we introduced a novel reinforcement learning (RL) (Sutton et al., 1998) framework for adaptive GNN rewiring called *Graph Actor-Critic* or (*GAC*). First, we formalized message passing in a GNN as a Markov decision process (MDP), and provided analytical results that provided a theoretical justification of our approach. We then derived an actor-critic framework based on this MDP formulation, and empirically, we showed how this approach yields robust gains on heterophilic graphs, preserves competitive performance on homophilic graphs, and provides qualitative interpretability through the lens of learned communication. Altogether, these results suggest that learning *when* and *with whom* to communicate is a simple, modular upgrade to static message passing that can greatly improve GNN performance. Moving forward, we believe that extending this approach to richer action spaces and to dynamic and temporal graphs is a promising direction for future research.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *ICLR*, 2021.

Álvaro Arroyo, Alessio Gravina, Benjamin Gutteridge, Federico Barbero, Claudio Gallicchio, Xiaowen Dong, Michael M Bronstein, and Pierre Vandergheynst. On vanishing gradients, over-smoothing, and over-squashing in GNNs: Bridging recurrent and graph learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=N4cyRMuLyl.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *ICLR*, 2022.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, 2020.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *NeurIPS*, 2020.

Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *WSDM*, 2022.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2015.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. In *JMLR*, 2023.

Vijay Prakash Dwivedi, Charilaos Kanatsoulis, Shenyang Huang, and Jure Leskovec. Relational deep learning: Challenges, foundations and next-generation architectures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5999–6009, 2025.

Ben Finkelshtein, Xingyue Huang, Michael Bronstein, and İsmail İlkan Ceylan. Cooperative graph neural networks, 2024. URL https://arxiv.org/abs/2310.01267.

Amir Ghazizadeh, Rickard Ewetz, and Hao Zheng. GAMMA: Gated multi-hop message passing for homophily-agnostic node representation in GNNs. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=S9YexPytpi.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry, 2017.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *IJCNN*, 2005.

Ming Gu, Zhuonan Zheng, Sheng Zhou, Meihan Liu, Jiawei Chen, Qiaoyu Tan, Liangcheng Li, and Jiajun Bu. Making classic GNNs strong baselines across varying homophily: A smoothness–generalization perspective. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=IAGbhDARZd.

Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718*, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Mingguo He, Zhewei Wei, Hongteng Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in neural information processing systems*, 34:14239–14251, 2021.

Qinhan Hou, Yilun Zheng, Xichun Zhang, Sitao Luan, and Jing Tang. Exploring heterophily in graph-level tasks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=d5cSo1bWhU.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Adarsh Jamadandi, Celia Rubio-Madrigal, and Rebekka Burkholz. Spectral graph pruning against over-squashing and over-smoothing. *Advances in Neural Information Processing Systems*, 37: 10348–10379, 2024.

Adarsh Jamadandi, Jing Xu, Adam Dziedzic, and Franziska Boenisch. Memorization in graph neural networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=jAJv3CaS4f.

Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montufar. FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In *ICLR*, 2023.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13333–13345, 2019.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. Policy-GNN: Aggregation optimization for graph neural networks. In *KDD*, 2020.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *NeurIPS*, 2020.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

Derek Lim, Felix Maria Hohne, Alexis Li, Patricia Huang, Serhan Gupta, Omkar Daigavane, Kenta Oono, and Shuiwang Ji. Characterizing graph datasets for node classification: Homophily–heterophily dichotomy and beyond. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22072–22085, 2022.

Junhong Lin, Xiaojie Guo, Shuaicheng Zhang, Yada Zhu, and Julian Shun. When heterophily meets heterogeneity: Challenges and a new large-scale graph benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 5607–5618. ACM, August 2025. doi: 10.1145/3711896.3737421. URL http://dx.doi.org/10.1145/3711896.3737421.

Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. Attacking graph convolutional networks via rewiring, 2019.

Ivan Marisca, Jacob Bamberger, Cesare Alippi, and Michael M Bronstein. Over-squashing in spatiotemporal graph neural networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=CVp0WCw4a1.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In *ICLR*, 2023.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887.

André Ribeiro, Ana Luiza Tenório, Juan Belieni, Amauri H. Souza, and Diego Mesquita. Cooperative sheaf neural networks, 2025. URL https://arxiv.org/abs/2507.00647.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E. Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. Relbench: A benchmark for deep learning on relational databases, 2024. URL https://arxiv.org/abs/2407.20060.

T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *IJCAI*, 2021.

Li Sun, Zhenhao Huang, Ming Zhang, and Philip S Yu. Deeper with riemannian geometry: Overcoming oversmoothing and oversquashing for graph foundation models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=yNej0aGtAZ.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. In *TOG*, 2019.

Xinyi Wu, Amir Ajorlou, Zihui Wu, and Ali Jadbabaie. Demystifying oversmoothing in attention-based graph neural networks. *Advances in Neural Information Processing Systems*, 36:35084–35106, 2023.

Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. In *NeurIPS*, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Guibin Zhang, Xiangguo Sun, Yanwei Yue, Chonghe Jiang, Kun Wang, Tianlong Chen, and Shirui Pan. Graph sparsification via mixture of graphs. *arXiv preprint arXiv:2405.14260*, 2024.

Zhuonan Zheng, Yuanchen Bei, Zhiyao Zhou, Sheng Zhou, Yao Ma, Ming Gu, Hongjia Xu, Jiawei Chen, and Jiajun Bu. Understanding and enhancing message passing on heterophilic graphs via compatibility matrix. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://neurips.cc/virtual/2025/poster/119967.

A Additional Theoretical Results

In this appendix, we provide additional theoretical results related to our proposed MDP formulation of message passing in a GNN.

A.1 Boundedness and Compactness of State and Action Spaces.

Lemma A.1. *Given a finite graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and bounded initial node features, if the message-passing layers are composed of continuous, linearly bounded operations, then the state-space, $\mathcal{S}$, and the action-space, $\mathcal{A}$, of the MDP formulation described by Equations equation 2 through equation 34 are bounded and compact.*

Proof. We first show that the state space, $\mathcal{S}$, is bounded and compact. Recall from Section 3.1 that the state is defined by $s^{(\ell)} \doteq \left(\{h_v^{(\ell)}\}_{v\in\mathcal{V}}, \tilde{\mathcal{E}}\right)$. Under the proposed MDP formulation, message passing is modulated by the continuous edge weights, $w_{u\to v}^{(\ell-1)} \in [0, 1]$. More formally, we have that:

$$h_v^{(\ell)} = \phi^{(\ell-1)}\Big(h_v^{(\ell-1)}, \psi^{(\ell-1)}\Big(\{\, w_{u\to v}^{(\ell-1)} h_u^{(\ell-1)} \mid u \in \mathcal{N}(v)\,\}\Big)\Big). \tag{23}$$

Because $w_{u\to v}^{(\ell-1)} \leq 1$ and the graph degree is bounded by some constant, $D > 0$ (given that we assume a finite graph), the weighted aggregated message is strictly bounded:

$$\left\|\psi^{(\ell-1)}\Big(\{w_{u\to v}^{(\ell-1)} h_u^{(\ell-1)} \mid u \in \mathcal{N}(v)\}\Big)\right\| \;\leq\; K \cdot \max_{u\in\mathcal{N}(v)} \|h_u^{(\ell-1)}\| \tag{24}$$

for some constant, $K > 0$, that depends on the aggregation rule and D.

Next, we observe that the update map $\phi^{(\ell-1)}$ is composed of linear operators, normalizations, and Lipschitz-continuous activations (e.g., ReLU), which strictly satisfy linear growth conditions. Thus, there exists a constant, $C_\phi^{(\ell-1)} > 0$, such that:

$$\|h_v^{(\ell)}\| \leq C_\phi^{(\ell-1)}\Big(\, \|h_v^{(\ell-1)}\| \;+\; \|\psi^{(\ell-1)}\Big(\{w_{u\to v}^{(\ell-1)} h_u^{(\ell-1)}\}\Big)\,\|\Big) \tag{25}$$

$$\|h_v^{(\ell)}\| \leq C_\phi^{(\ell-1)}(1+K) \max_{k\in\mathcal{N}(v)\cup\{v\}} \|h_k^{(\ell-1)}\| \tag{26}$$

Defining the layer-wise maximum embedding norm as $M_\ell \doteq \max_{v\in\mathcal{V}} \|h_v^{(\ell)}\|$, the inequality yields the recursive relation $M_\ell \leq C \cdot M_{\ell-1}$, where $C \doteq \max_\ell C_\phi^{(\ell)}(1+K) < \infty$. Unrolling this recursion over the finite GNN depth L gives:

$$\|h_v^{(\ell)}\| \leq M_\ell \leq C^L M_0 < \infty \quad \forall \ell \in \{0, 1, \ldots, L\}. \tag{27}$$

That is, that all node embeddings lie in a closed and bounded (hence compact by the Heine-Borel theorem) subset of $\mathbb{R}^d$.

Next, we observe that the graph structure component, $\tilde{\mathcal{E}}$, is parameterized by discrete edge indices and continuous edge weights, $w \in [0, 1]$. Since the graph is finite and $[0, 1]$ is closed and bounded, the space of possible structures is also compact.

As such, because the Cartesian product of compact spaces is itself compact, we can conclude that the state-space, $\mathcal{S}$, is bounded and compact.

Next, we establish the compactness of the action space. Recall from Section 3.1 that for each node, u, the action-space, $\mathcal{A}_u$, is a finite, discrete set of four communication primitives. Consequently, the joint action-space, $\mathcal{A} = \prod_{u\in\mathcal{V}} \mathcal{A}_u$, is also a finite set, and is therefore trivially bounded and compact.

This completes the proof. □

A.2 Variance Decomposition of the Reward Function

We decompose the variance term in the reward function to get an analytical form. Let the edge difference be defined as $\Delta_{uv}^{(\ell)} = \bar{h}_u^{(\ell)} - \bar{h}_v^{(\ell)}$. Given the normalization constraint $\|\bar{h}_u^{(\ell)}\|_2 = \|\bar{h}_v^{(\ell)}\|_2 = 1$, we define the cosine similarity as $\mathrm{Sim}_{uv}^{(\ell)} = \bar{h}_u^{(\ell)\top} \bar{h}_v^{(\ell)}$. The squared Euclidean distance is then:

$$\|\Delta_{uv}^{(\ell)}\|_2^2 = \|\bar{h}_u^{(\ell)}\|_2^2 + \|\bar{h}_v^{(\ell)}\|_2^2 - 2\,\bar{h}_u^{(\ell)\top} \bar{h}_v^{(\ell)} \tag{28}$$

$$= 2 - 2\mathrm{Sim}_{uv}^{(\ell)} \tag{29}$$

$$= 2(1 - \mathrm{Sim}_{uv}^{(\ell)}). \tag{30}$$

Taking the expectation over the set of edges $(u, v) \sim \mathcal{E}$, we obtain:

$$\mathbb{E}_{(u,v)\sim\mathcal{E}}\big[\|\Delta_{uv}^{(\ell)}\|_2^2\big] = 2(1 - \mathrm{Sim_{uv}}^{(\ell)}), \tag{31}$$

where $\mathrm{Sim}^{(\ell)} = \mathbb{E}_{(u,v)\sim\mathcal{E}}[\mathrm{Sim}_{uv}^{(\ell)}]$. Using the variance identity for vectors, $\mathrm{Var}[X] \doteq \mathbb{E}[\|X\|_2^2] - \|\mathbb{E}[X]\|_2^2$, the variance of the edge differences is expressed as:

$$\mathrm{Var}_{\Delta}^{(\ell)} = \mathbb{E}_{(u,v)\sim\mathcal{E}}\big[\|\Delta_{uv}^{(\ell)}\|_2^2\big] - \big\|\mathbb{E}_{(u,v)\sim\mathcal{E}}[\Delta_{uv}^{(\ell)}]\big\|_2^2 \tag{32}$$

$$= 2(1 - \mathrm{Sim_{uv}}^{(\ell)}) - \big\|\mathbb{E}_{(u,v)\sim\mathcal{E}}[\Delta_{uv}^{(\ell)}]\big\|_2^2. \tag{33}$$

We now consider the reward function defined as:

$$r^{(\ell)} = \mathrm{Var}_{\Delta}^{(\ell)} - \lambda \cdot \mathrm{Sim_{uv}}^{(\ell)}. \tag{34}$$

Substituting the decomposed variance into this expression yields the following relation:

$$r^{(\ell)} = \Big(2(1 - \mathrm{Sim_{uv}}^{(\ell)}) - \big\|\mathbb{E}_{(u,v)\sim\mathcal{E}}[\Delta_{uv}^{(\ell)}]\big\|_2^2\Big) - \lambda \cdot \mathrm{Sim_{uv}}^{(\ell)} \tag{35}$$

$$= 2 - 2\mathrm{Sim_{uv}}^{(\ell)} - \lambda\mathrm{Sim_{uv}}^{(\ell)} - \big\|\mathbb{E}_{(u,v)\sim\mathcal{E}}[\Delta_{uv}^{(\ell)}]\big\|_2^2 \tag{36}$$

$$= 2 - (2 + \lambda)\mathrm{Sim_{uv}}^{(\ell)} - \big\|\mathbb{E}_{(u,v)\sim\mathcal{E}}[\Delta_{uv}^{(\ell)}]\big\|_2^2. \tag{37}$$

Thus, the reward is maximized by minimizing both the average edge similarity and the squared norm of the expected edge difference, with the similarity penalty scaled by $(2 + \lambda)$.

A.3 Computational Complexity

Indeed, a GNN model such as GIN (Xu et al., 2019) trained with our actor–critic objective introduces additional computational overhead compared to standard supervised GNNs, due to the separate actor and critic networks, repeated policy updates, and per-layer rollout collection through the message-passing MDP. If the encoder has L layers, embedding dimension d, $|\mathcal{V}|$ nodes, and $|\mathcal{E}|$ edges, the per-rollout training complexity is approximately

$$\mathcal{O}\big(L\,(|\mathcal{E}|\,d \;+\; |\mathcal{V}|\,d^2)\big)\,,$$

Because encoder relies on sparse local message passing, the cost scales approximately linearly in $|\mathcal{E}|$ for sparse graphs, making GAC substantially more scalable than transformer-based graph architectures, whose per-layer global self-attention requires $\mathcal{O}(|\mathcal{V}|^2\,d)$ time and $\mathcal{O}(|\mathcal{V}|^2)$ memory. Therefore, while GAC is computationally more expensive than conventional supervised GNN models, its overall complexity remains considerably lower than full graph-transformer architectures on benchmark graph datasets.

B EXPERIMENT DETAILS

In this appendix, we provide details pertaining to the experiments performed in this work. Through these experiments, we evaluate our method on a diverse suite of graphs spanning heterophilic, homophilic, synthetic, and domain-specific settings to demonstrate the effectiveness and generality of RL-based graph rewiring. Across tasks, we adhere to standard evaluation metrics (e.g., Accuracy, AUC-ROC, Average Precision) and use fixed or cross-validated splits for fair comparison.

B.1 DATASET CATEGORIES

B.1.1 SYNTHETIC GRAPH AND FEATURE GENERATION.

We construct a family of controlled synthetic benchmarks to validate layer-wise communication policies under varying graph topologies and degrees of heterophily. Each instance consists of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node labels $y \in \{0, \dots, C-1\}^{|\mathcal{V}|}$ (for all graphs, we use C=2), and node features $\mathcal{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ (d=16). The different synthetic graphs considered are shown in Figure 5.

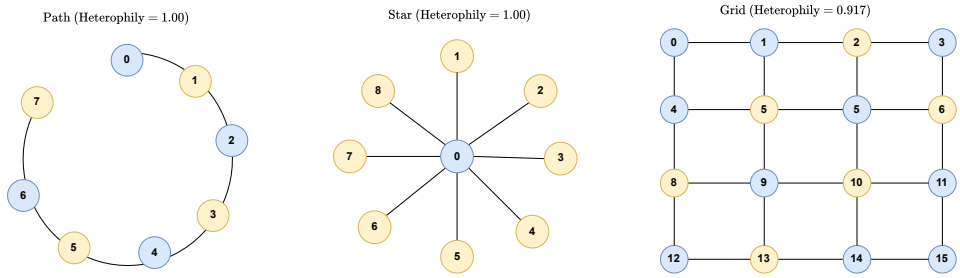


Figure 5: Synthetic graph- Path, Star and Grid like structure.

B.1.2 LABEL ASSIGNMENT AND HETEROPHILY CONTROL

We define heterophily as the fraction of edges whose endpoints carry different labels:

$$\mathcal{H}(\mathcal{G}, y) \;=\; \frac{1}{|\mathcal{E}|} \sum_{(u,v)\in E} \mathbf{1}[y_u \neq y_v]\,. \tag{38}$$

For path graphs, we assign alternating labels along the chain, yielding predominantly cross-class edges. For star graphs, we assign the hub to class 0 and all leaves to class 1, making every hub–leaf edge heterophilic. For grid graphs, we target a heterophily level $p_{\text{hetero}} \in [0, 1]$. We initialize labels using a checkerboard pattern, $y_{i,j} \;=\; (i + j) \bmod 2$, which is (near-)maximally heterophilic under 4-neighbour connectivity. If the resulting $\mathcal{H}(\mathcal{G}, y)$ exceeds p_{hetero}, we reduce heterophily by flipping the labels of a randomly sampled subset of nodes $\mathcal{F} \subseteq V$ (sampled without replacement), applying $y_u \leftarrow 1 - y_u$ for $u \in \mathcal{F}$. The number of flips is chosen heuristically as

$$|\mathcal{F}| \;=\; \left\lfloor \frac{|\mathcal{V}|\big(\mathcal{H}(\mathcal{G}, y) - p_{\text{hetero}}\big)}{2} \right\rfloor. \tag{39}$$

Because flips are applied once and interact combinatorially with the grid structure, the achieved heterophily matches p_{hetero} approximately rather than exactly. For a target heterophily of 0.80, we achieve 0.91 for a 4×4 graph.

B.2 Dataset Statistics

The statistics of the real-world long-range, node-based, and graph-based benchmarks used can be found in Tables 3, 4, and 5. For node classification datasets we report *edge homophily* and *adjusted homophily* (Lim et al., 2022); values for the heterophilic suite align with standard releases (Platonov et al., 2023).

Table 3: Statistics of the heterophilic node classification benchmarks.

	roman-empire	amazon-ratings	minesweeper	tolokers	questions
# nodes	22662	24492	10000	11758	48921
# edges	32927	93050	39402	519000	153540
avg. degree	2.91	7.60	7.88	88.28	6.28
# node features	300	300	7	10	301
# classes	18	5	2	2	2
edge homophily	0.05	0.38	0.68	0.59	0.84
adj. homophily	-0.05	0.14	0.01	0.09	0.02
metrics	ACC	ACC	AUC-ROC	AUC-ROC	AUC-ROC

Table 4: Statistics of the graph classification benchmarks.

	IMDB-B	IMDB-M	REDDIT-B	NCI1	PROTEINS	ENZYMES
# graphs	1000	1500	2000	4110	1113	600
# average nodes	19.77	13.00	429.63	29.87	39.06	32.63
# average edges	96.53	65.94	497.75	32.30	72.82	64.14
avg. degree	9.77	10.14	2.32	2.16	3.73	3.93
# classes	2	3	2	2	2	6
metrics	ACC	ACC	ACC	ACC	ACC	ACC

Table 5: Statistics of the homophilic node classification benchmarks.

	pubmed	cora	citeseer
# nodes	18717	2708	3327
# edges	44327	5278	4552
avg. degree	4.74	3.90	2.74
# node features	500	1433	3703
# classes	3	6	7
edge homophily	0.80	0.81	0.74
adj. homophily	0.69	0.77	0.67
metrics	ACC	ACC	ACC

B.3 HYPERPARAMETERS

The hyperparameters considered in the experiments are shown in Tables 6, 7, and 8.

Table 6: RL Parameters

Parameter	Search Space	Default	Description
policy_coef	[0.5, 1.0, 2.0]	1.0	Policy loss coefficient
value_coef	[0.5, 1.0, 2.0]	1.0	Value loss coefficient
entropy_coef	[0.3, 0.5, 1.0]	1.0	Entropy coefficient
gamma	[0.95, 0.99]	0.99	Discount factor for TD learning
lambda	[0.5, 0.75, 1.00]	1.00	Penalization factor for node similarity

Table 7: Model Architecture

Parameter	Search Space	Default	Description
env_dim	[64, 128, 256]	128	Hidden dimension for environment network
action_dimension	[8, 16, 32]	16	Hidden dimension for policy network
env_num_layers	[2, 4, 6, 8, 10]	2	Number of GNN layers for environment
policy_num_layers	[2, 4, 6, 8, 10]	1	Number of GNN layers in policy networks
env_gnn_type	[avergage, sum]	average	Environment network aggregation type
policy_gnn_model	[average, sum]	average	Policy network aggregation type
dropout	[0.0, 0.2, 0.5]	0.2	Dropout rate

Table 8: Training Hyperparameter

Parameter	Search Space	Default	Description
lr	[1e-3, 1e-4, 5e-4]	1e-3	Learning rate
batch_size	[32, 64, 128]	64	Batch size
env_num_layers	2 to 10	10	Environment GNN layer
policy_num_layers	2 to 10	10	Policy GNN layer
num_epochs	1000	10000	Num epochs
optimizer_type	[Adam, Adamw]	Adam	Optimizer type
weight_decay	[0.0, 1e-5, 1e-4]	0.0	Learning Weight decay

C Additional Empirical Results and Discussions

In this appendix, we provide additional results and discussions that address three potential questions about how GAC achieves the gains reported in Section 4.

C.1 Performance on Homophilic Benchmarks

Table 9 reports performance on standard homophilic benchmarks (Citeseer, Pubmed, Cora), where connected nodes tend to share labels. While GAC is designed primarily for heterophilic settings, it still maintains competitive performance on homophily-dominated graphs, achieving results comparable to or exceeding most baselines.

Table 9: Results on homophilic node classification datasets.

	Citeseer	Pubmed	Cora
GraphSAGE	76.04 ± 1.30	88.45 ± 0.50	86.90 ± 1.04
GCN	76.50 ± 1.36	88.42 ± 0.50	86.98 ± 1.27
GAT	76.55 ± 1.23	87.30 ± 1.10	86.33 ± 0.48
GGCN	77.14 ± 1.45	89.15 ± 0.37	87.95 ± 1.05
H2GCN	77.11 ± 1.57	89.49 ± 0.38	87.87 ± 1.20
GPRGNN	77.13 ± 1.67	87.54 ± 0.38	<u>87.95</u> ± <u>1.18</u>
MixHop	76.26 ± 1.33	85.31 ± 0.61	87.61 ± 0.85
CoGNN	77.65 ± 1.27	89.60 ± 0.42	86.53 ± 1.20
GCNII	77.33 ± 1.48	**90.15 ± 0.43**	**88.37 ± 1.25**
Geom-GCN	**78.02 ± 1.15**	89.95 ± 0.47	85.35 ± 1.57
PairNorm	73.59 ± 1.47	87.53 ± 0.44	85.79 ± 1.01
GAC	<u>78.21</u> ± <u>1.45</u>	<u>90.01</u> ± <u>0.55</u>	87.33 ± 1.21

C.2 How Does the Learned Policy Differ from Standard Message-Passing?

The learned policy exhibits adaptive, layer-dependent communication patterns that fundamentally differ from uniform aggregation. Figure 4 shows how GAC adaptively modifies a 5×5 grid graph structure across layers 0–2. The original uniform neighborhood connectivity (leftmost panel) is transformed layer-by-layer: edge retention ratios drop from 100% (layer 0) to 60% (layer 2), with edge weights varying continuously based on local embedding similarity. Thick edges indicate strong learned connections (high affinity between nodes), while thin/absent edges reveal suppressed communication. This heterogeneous pattern contrasts sharply with standard GNNs, which apply uniform aggregation regardless of local structure or embedding quality. The learned policy effectively implements a data dependent graph rewiring that adapts to both topology and learned representations—a capability standard message-passing lacks.

In Figure 6, we show how GAC maintains diversity-similarity reward across layers on five heterophilic datasets, exposing two distinct learned strategies. On Amazon Ratings and Tolokers, the reward exhibits a mid-depth optimum, peaking around layers 4 to 6 before declining indicating the policy learns that deeper propagation reintroduces contraction pressure and adaptively isolates nodes in late layers. In contrast, Minesweeper and Roman Empire show steadily increasing reward with late layer acceleration, revealing the policy learned that deeper receptive fields remain beneficial when node communication is done adaptively.

C.3 Does GAC Mitigate Oversmoothing?

GAC provably and empirically mitigates oversmoothing through learned layer-wise actions that prevent representation collapse. In Figure 7, we measure the average pairwise cosine similarity of node embeddings across layers. Standard GNNs (GCN, GAT, GIN) rapidly saturate toward high similarity ($>$0.8) which is a signature of oversmoothing whereas GAC maintains markedly lower similarity ($<$0.6) throughout the depth range. This divergence is most pronounced on Minesweeper and Roman Empire, precisely the datasets where GAC shows strongest depth robustness. The

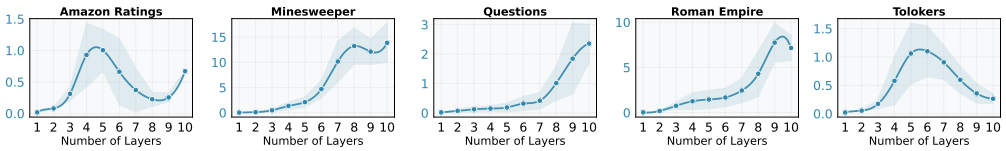


Figure 6: Layer-wise reward for GAC across multiple layers on five heterophilic graph datasets across 10 layers. Reward quantifies the diversity of node representations at each layer.

lower similarity indicates that GAC successfully prevents nodes from collapsing to indistinguishable representations, directly validating Proposition 3.4 by learning to suppress communication along edges that promote collapse, enabling GAC to perform effective deep information propagation.

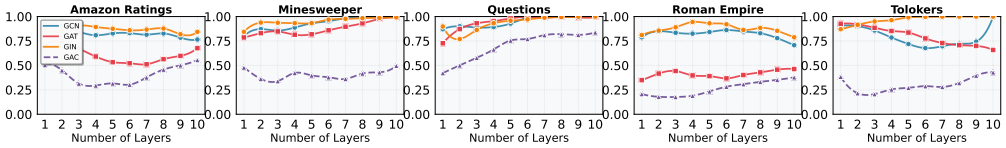


Figure 7: Node similarity across 10 layers for GCN, GAT, GIN, and GAC on five heterophilic datasets. Similarity is measured as the average pairwise cosine similarity of node embeddings per layer. GAC (dashed purple) is compared against baseline models (solid lines).

In Figure 8, we plot test accuracy as a function of network depth for GCN, GAT, and GAC across four representative datasets. On homophilic benchmarks (CORA and CITESEER, top row), all methods benefit from shallow architectures (2 to 4 layers) where neighborhood aggregation aligns with label similarity. As depth increases, GCN and GAT exhibit characteristic oversmoothing collapse: accuracy degrades sharply beyond 4 to 6 layers as node representations converge to indistinguishable values. GAC shows more graceful degradation and offers an advantage over these traditional methods. For heterophilic graphs, standard message-passing methods struggle at all depths: GCN and GAT achieve suboptimal accuracy even at shallow depths and rapidly worsen as layers accumulate. In contrast, GAC not only achieves substantially higher peak performance but maintains robust accuracy across a much wider depth range (2 to 8 layers).

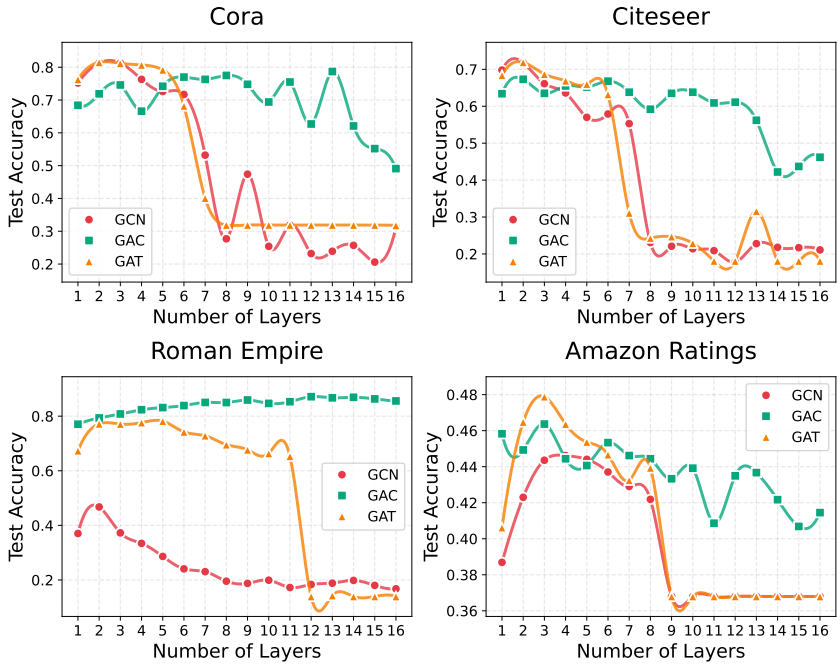


Figure 8: Test accuracy as a function of the number of GNN layers for GCN, GAT, and GAC on representative homophilic and heterophilic benchmarks

C.4 CAN LABEL-FREE REWARDS GUIDE EFFECTIVE GRAPH REWIRING?

If the heterophily reward, which is computed without any label information, successfully identifies beneficial communication patterns, then GAC should outperform models trained on the original topology, despite the learned policy never observing labels. The results shown in Table 1 confirm the hypothesis. On ROMAN EMPIRE, a model trained on the GAC-rewired graph achieves 91.79% accuracy versus 72.5% on the original graph using GCN which is a gain of 19.3 points purely from structural modification. On AMAZON RATINGS, GAC rewiring improves performance from 48.03% to 53.94% (+5.91 points). These substantial gains validate that the label-free reward acts as a general purpose, task-agnostic graph preprocessing method. The learned communication patterns transfer to supervised learning, demonstrating they capture fundamental structural properties rather than artifacts of the reward optimization. This result has important implications: it shows that heterophily-aware structural learning is possible without labels, and that the diversity-similarity objective aligns with downstream task requirements.

C.5 PERFORMANCE ON LONG-RANGE GRAPHS

To assess whether adaptive rewiring helps when long-range dependencies dominate, we evaluate on the peptides tasks from the Long Range Graph Benchmark (Dwivedi et al., 2023). The split

contains roughly 15,000 molecular graphs and two targets: PEPTIDES-FUNC, a graph-level multi-label classification task reported in *average precision* (AP), and PEPTIDES-STRUCT, a regression task reported in *mean absolute error* (MAE).

Table 10 shows that GAC achieves the best AP on PEPTIDES-FUNC and the second-lowest MAE on PEPTIDES-STRUCT, slightly behind Cooperative Sheaf Neural Networks (Ribeiro et al., 2025). Taken together with the LRGB setting, this pattern supports using learned communication, which can also be useful to mitigate over-squashing problem on graph tasks that stress long-range information flow. The results and experiment protocol are taken from (Ribeiro et al., 2025).

Table 10: Performance comparison of models on the peptides datasets.

Model	**peptides-func** ↑	**peptides-struct** ↓
GCN	68.60 ± 0.50	24.60 ± 0.07
GINE	66.21 ± 0.67	24.73 ± 0.17
GatedGCN	67.65 ± 0.47	24.77 ± 0.09
DReW	71.50 ± 0.44	25.36 ± 0.15
SAN	64.39 ± 0.75	25.45 ± 0.12
GPS	65.34 ± 0.91	25.09 ± 0.14
G-ViT	69.42 ± 0.75	24.49 ± 0.16
Exphormer	65.27 ± 0.43	24.81 ± 0.07
BuNN	$\underline{72.76} \pm \underline{0.65}$	24.63 ± 0.12
CSNN	71.58 ± 0.80	$\mathbf{24.32 \pm 0.04}$
GAC	$\mathbf{73.21 \pm 0.51}$	$\underline{24.41} \pm \underline{0.13}$

C.6 LARGE-SCALE HETEROGENEOUS GRAPH BENCHMARKS

To further test whether GAC's gains hold at scale and under heterophily arising from graph heterogeneity (rather than homogeneous graphs alone), we evaluate on the H2GB benchmark suite (Lin et al., 2025), which combines heterogeneous node types/edge types with heterophilic label distributions across nine large-scale graphs: OGBN-MAG, MAG-YEAR, OAG-CS, OAG-ENG, OAG-CHEM, H-POKEC, RCDD, IEEE-CIS-G, and PDNS. Table 11 reports test accuracy (mean ± std) against the baselines reported by Lin et al. (2025). GAC achieves the best performance on all nine datasets, indicating that learned, node-level communication gating remains effective even when heterophily is entangled with heterogeneous node and edge semantics at scale.

Table 11: Results on the H2GB large-scale heterogeneous graph benchmark suite. Best in bold.

	OGBN-MAG	MAG-YEAR	OAG-CS	OAG-ENG	OAG-CHEM	H-POKEC	RCDD	IEEE-CIS-G	PDNS
GCN	42.90 ± 0.50	32.91 ± 0.50	18.22 ± 0.60	29.09 ± 0.52	18.57 ± 1.06	70.63 ± 0.36	85.81 ± 0.87	28.79 ± 1.07	81.22 ± 0.30
GAT	48.60 ± 0.29	33.50 ± 0.62	19.12 ± 0.25	28.74 ± 0.60	14.05 ± 0.44	70.89 ± 0.20	86.71 ± 1.27	28.51 ± 0.45	93.97 ± 0.27
MixHop	46.99 ± 0.41	36.36 ± 0.28	23.04 ± 0.24	36.88 ± 0.73	25.03 ± 0.90	78.78 ± 0.27	85.43 ± 1.22	30.13 ± 0.86	92.78 ± 0.18
FAGCN	33.06 ± 0.59	27.10 ± 0.66	10.46 ± 0.44	22.75 ± 0.94	13.01 ± 0.44	67.15 ± 0.09	81.06 ± 1.24	10.09 ± 5.09	82.84 ± 1.07
HGT	50.23 ± 0.48	39.47 ± 1.66	22.51 ± 0.40	35.51 ± 0.52	25.48 ± 0.76	78.91 ± 0.43	86.05 ± 1.01	30.89 ± 0.80	92.76 ± 0.15
SHGN	43.39 ± 0.28	34.43 ± 1.23	22.03 ± 0.46	36.93 ± 0.67	24.07 ± 0.94	50.50 ± 0.89	79.67 ± 2.53	$\mathbf{31.66 \pm 0.86}$	89.33 ± 0.21
$\mathcal{H}^2$G-former	55.67 ± 0.35	52.55 ± 0.66	28.47 ± 0.93	46.63 ± 0.65	30.62 ± 0.31	82.45 ± 0.19	87.35 ± 0.80	31.55 ± 0.92	96.43 ± 0.21
GAC	$\mathbf{56.23 \pm 0.35}$	$\mathbf{53.08 \pm 0.66}$	$\mathbf{28.75 \pm 0.93}$	$\mathbf{47.10 \pm 0.65}$	$\mathbf{30.93 \pm 0.31}$	$\mathbf{83.68 \pm 0.19}$	$\mathbf{97.39 \pm 0.31}$	$\mathbf{31.66 \pm 0.86}$	$\mathbf{97.39 \pm 0.21}$

D LIMITATIONS

In GAC, we use the mean-field approximation to determine the value of the Q-function, treating each node's communication decision as independent of its neighbours. Given that the edge weights are jointly determined by sender's and receiver's action values, neighbouring actions are coupled, and using mean-field approximation may yield suboptimal policies. However, this is a deliberate choice for computational efficiency, considering the large joint state-action space.

Additionally, in GAC, the policy assigns send and receive probabilities to every node in the graph, which can result in $\mathcal{O}(|\mathcal{V}|^2)$ complexity in the worst case making the approach intractable for large graphs . While neighbor sampling strategies can mitigate this cost, scaling GAC to very large graphs remains an open direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for organizing the paper structure, find relevant more related works and cooperative agent to implement standard algorithms and visualization of the results and latex code along with citation formatting. We have not used generative AI tools for raw mathematical derivation, gathering research ideas. We want to emphasize that the idea presented in the paper is our own and not brainstormed with generative AI. We have reviewed all AI-assisted work. We have verified all citations manually at our best to ensure their correctness and cross checked in the internet. Also, code written with generative AI are unit tested to make sure the accuracy of the implementation is correct. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

We do not foresee any ethical violation of our work.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we have anonymized our codebase here `https://anonymous.4open.science/r/gac-67CF/README.md`. Full model, artifacts and training hyperparameters and end to end pipeline will be published upon acceptance of the paper.