

# Learning to Compress Graphs via Dual Agents for Consistent Topological Robustness Evaluation

Qisen Chai<sup>1</sup>, Yansong Wang<sup>1</sup>, Junjie Huang<sup>1</sup>, Tao Jia<sup>1,2,\*</sup>

<sup>1</sup>College of Computer and Information Science, Southwest University

<sup>2</sup>College of Computer and Information Science, Chongqing Normal University

cqsllyt@email.swu.edu.cn, yansong0682@email.swu.edu.cn, junjiehuang.cs@outlook.com, tja@swu.edu.cn

## Abstract

As graph-structured data grow increasingly large, evaluating their robustness under adversarial attacks becomes computationally expensive and difficult to scale. To address this challenge, we propose to compress graphs into compact representations that preserve both topological structure and robustness profile, enabling efficient and reliable evaluation. We propose Cutter, a dual-agent reinforcement learning framework composed of a Vital Detection Agent (VDA) and a Redundancy Detection Agent (RDA), which collaboratively identify structurally vital and redundant nodes for guided compression. Cutter incorporates three key strategies to enhance learning efficiency and compression quality: trajectory-level reward shaping to transform sparse trajectory returns into dense, policy-equivalent learning signals; prototype-based shaping to guide decisions using behavioral patterns from both high- and low-return trajectories; and cross-agent imitation to enable safer and more transferable exploration. Experiments on multiple real-world graphs demonstrate that Cutter generates compressed graphs that retain essential static topological properties and exhibit robustness degradation trends highly consistent with the original graphs under various attack scenarios, thereby significantly improving evaluation efficiency without compromising assessment fidelity.

Code — <https://github.com/Qisenne/Cutter>

## Introduction

Graphs provide a powerful abstraction for modeling complex systems in the real world, including urban infrastructure, biomolecular interactions, and social networks(Battiston et al. 2021). These systems often exhibit intricate structures and interdependencies that can be effectively captured using graph-based representations. Among the key properties of such systems, robustness, which refers to the ability to maintain topological integrity under structural perturbations, plays a critical role in ensuring reliable performance(Artime et al. 2024). However, as real-world systems grow in size and complexity, their graph representations become increasingly large and computationally expensive to analyze. Most existing robustness evaluation methods rely on topology structural metrics, which are often costly to

compute and do not scale well as graph size increases(Besta et al. 2018; Zhang et al. 2023).

To address this challenge, various graphs reduction methods have been proposed to enable scalable analysis of large-scale graph. These include graph compression (Liu et al. 2018a), sparsification (Wickman, Zhang, and Li 2022; Wu and Chen 2020), and sampling (Zhao et al. 2020; Chen, Zeng, and Cui 2022), which aim to preserve critical structural features such as dense subgraphs, semantic backbones, and path-based relevance (Fan et al. 2021; Kang, Lee, and Shin 2022). By producing more compact graph representations, these methods help alleviate computational overhead and facilitate downstream tasks. However, they largely ignore whether the compressed graph preserves the robustness profile of the original, that is, how its connectivity degrades under targeted attacks, thereby limiting their utility in robustness analysis scenarios.

To address this gap, we recognize that certain nodes are structurally critical for maintaining connectivity, and their removal may lead to abrupt and irreversible collapse. A node's impact on overall robustness depends not only on its own role but also on the structural support of its neighbors, some of which are essential while others are redundant. Building on this observation, we distinguish vital and redundant nodes, and define graph compression as reducing graph size through node removal rather than general sparsification or coarsening.

In this paper, we propose **Cutter**, a dual-agent reinforcement learning framework for structural-robustness-preserving graph compression. Cutter comprises two specialized agents: a **Vital Detection Agent (VDA)** that identifies crucial nodes to retain, and a **Redundancy Detection Agent (RDA)** that selects non-essential nodes to remove. The two agents share a graph convolutional encoder (Kipf and Welling 2016) for capturing low-level structural features, followed by task-specific sub-encoders (Li and Hoiem 2017; Zhang et al. 2022b) tailored to their respective objectives. This architecture enables coordinated yet specialized decision-making throughout the compression process.

To enhance the coordination and learning efficiency of both agents during the compression process, we design a reinforcement learning optimization mechanism from three complementary perspectives: **1) Trajectory-Level Return Reward Shaping**. We propose a global shaping strategy that

\*Corresponding Author.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

estimates trajectory-level quality using a neural network and allocates fine-grained, step-wise rewards to each state-action pair. To ensure consistency with the trajectory return, an affine transformation is applied to align the cumulative predicted rewards with the actual return.

**2) State-Action-Level Prototype-Constrained Reward Shaping.** To improve the reward network’s ability to capture behavior-level distinctions, we introduce a prototype-based contrastive module that extracts state-action subsequences from trajectories with significant changes in connectivity, encodes them via a recurrent encoder (GRU)(Dey and Salem 2017), and uses them as supervision signals to guide reward estimation.

**3) Cross-Agent Active-Follow Exploration.** Given the task asymmetry between VDA and RDA, we design an active-follow strategy in which agents alternate between leader and follower roles during training. The follower replicates the leader’s action sequence and evaluates the trajectory under its own objective, enabling safe exploration, and effective knowledge transfer. In summary, the main contributions of this paper are as follows:

- We propose a dual-agent reinforcement learning framework for graph compression that preserves the robustness profile of the original graph by jointly identifying vital and redundant nodes. To the best of our knowledge, this is the first method specifically designed to maintain the robustness profile of the original graph during compression.
- We design a reinforcement learning optimization mechanism that integrates dense reward shaping, prototype-guided contrastive discrimination, and active-follow cooperative exploration to enhance policy effectiveness and inter-agent collaboration.
- We conduct extensive experiments on real-world graph datasets, demonstrating that our method preserves the robustness degradation patterns of the original graphs with high fidelity, even under substantial compression, while maintaining key topological structures.

## Methodology

We begin this section by outlining the overall design of our approach. We first define the graph compression task under robustness constraints, then present the architecture of our dual-agent reinforcement learning framework. Finally, we introduce three key modules that drive effective learning and structural-robustness-preserving compression. An overview of the Cutter framework is illustrated in Figure 1.

### Problem Definition

Formally, let  $G = (V, E)$  be an undirected graph with  $|V| = n$  nodes and  $E \subseteq V \times V$  edges. The robustness of  $G$  is quantified by its pairwise connectivity:

$$F(G) = \sum_{C \in \mathcal{C}(G)} \frac{|C|(|C| - 1)}{2}, \quad (1)$$

where  $\mathcal{C}(G)$  denotes the set of connected components in  $G$ , and  $|C|$  is the number of nodes in component  $C$ . We use pairwise connectivity to measure robustness, as it reflects the

number of node pairs that remain connected. This metric is well-suited for real-world systems where overall functionality depends on sustained node-to-node connectivity.(Li et al. 2021).

Let  $\mathcal{D} \subseteq V$  be a subset of nodes to be removed, yielding a compressed graph  $G' = G \setminus \mathcal{D}$  with compression ratio  $\rho = 1 - \frac{|\mathcal{D}|}{|V|}$ . To evaluate the robustness consistency between  $G$  and  $G'$ , we apply a set of adversarial node selection strategies  $X$  to both graphs as perturbation methods that simulate targeted attacks on the graph structure. Let  $\tilde{G}_x$  and  $\tilde{G}'_x$  denote the graphs resulting from applying the attack strategy  $x \in X$  to  $G$  and  $G'$ , respectively. We aim to minimize the discrepancy in robustness under all attack strategies:

$$\min_{\mathcal{D} \subseteq V} \sum_{x \in X} \left| F(\tilde{G}_x) - F(\tilde{G}'_x) \right| \quad \text{s.t.} \quad |\mathcal{D}| = (1 - \rho) |V|. \quad (2)$$

To solve the above optimization problem, we formulate the node removal process as a Markov Decision Process (MDP), defined by the tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ . Each state  $s_t \in \mathcal{S}$  represents the current residual graph at step  $t$ , and the action  $a_t \in \mathcal{A}(s_t)$  denotes the node selected for removal. The transition function is defined as:  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$  which updates the state by removing the selected node. The reward function  $\mathcal{R}(s_t, a_t)$  assigns a scalar reward to each state-action pair, which is estimated by a neural network in our framework. The discount factor  $\gamma \in [0, 1]$  controls the weighting of future rewards.

To evaluate the long-term impact of each action, we define the Q-function as the expected cumulative reward:

$$Q(s_t, a_t) = \mathbb{E}_{\pi} \left[ \sum_{i=0}^{\infty} \gamma^i \mathcal{R}(s_{t+i}, a_{t+i}) \right]. \quad (3)$$

Following the Bellman equation, the optimal Q-function satisfies:

$$Q^*(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma \cdot \max_{a'} Q^*(s_{t+1}, a'). \quad (4)$$

We adopt standard Deep Q-Networks (DQN)(Mnih et al. 2015) to approximate  $Q^*$ , using experience replay to store and reuse trajectories. This aligns naturally with our graph-based setting, where both agents benefit from sample-efficient reuse of episodic transitions. Each agent maintains its own Q-network and selects actions using an  $\varepsilon$ -greedy policy. For example:

$$\begin{aligned} \pi_{\theta}^{\text{vda}}(a_t | s_t) &= \begin{cases} \text{random action,} & \text{with probability } \varepsilon \\ \arg \max_a Q^{\text{vda}}(s_t, a), & \text{otherwise,} \end{cases} \\ \pi_{\theta}^{\text{rda}}(a_t | s_t) &= \begin{cases} \text{random action,} & \text{with probability } \varepsilon \\ \arg \max_a Q^{\text{rda}}(s_t, a), & \text{otherwise.} \end{cases} \end{aligned} \quad (5)$$

In addition to separate Q-networks, the agents also employ their own task-specific reward networks, denoted as  $\mathcal{R}_{\phi}^{\text{vda}}$  and  $\mathcal{R}_{\phi}^{\text{rda}}$ . Each reward network is a learnable module trained to estimate step-wise rewards aligned with the agent’s objective.

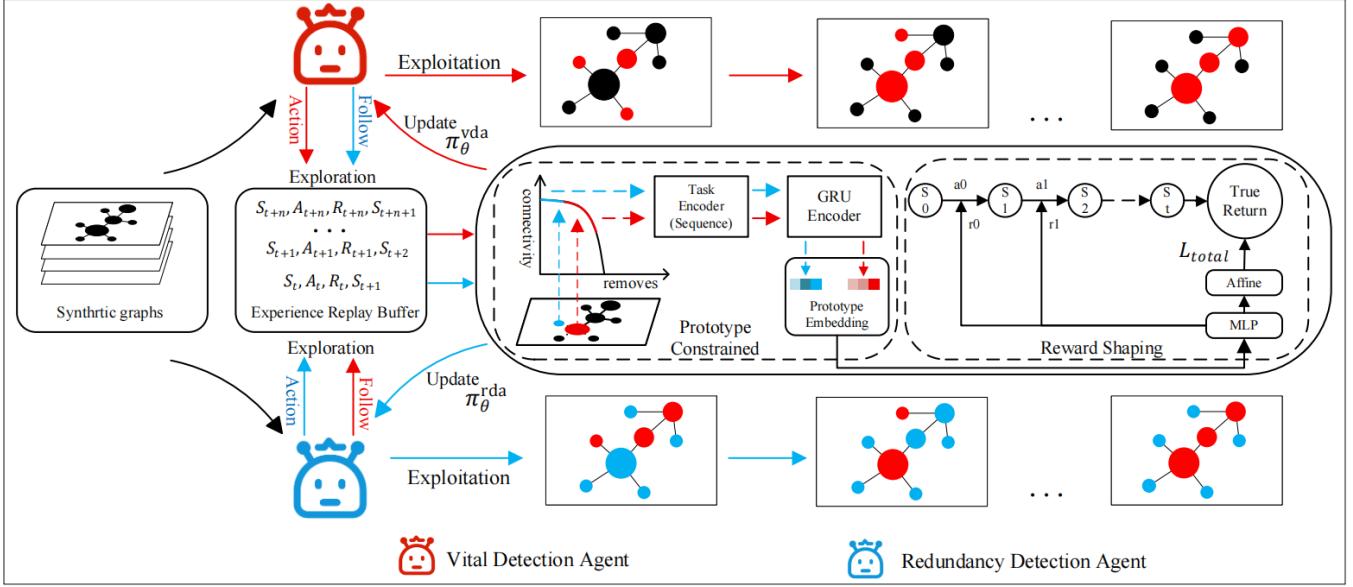


Figure 1: Overview of the Cutter Framework

### Agent-Specific Graph Encoder–Decoder

To enable collaborative learning and task-specific decision-making for dual agents in the graph compression task, we design a reinforcement learning-aware neural architecture tailored for graph-structured data. The architecture comprises three main components: a shared graph encoder that extracts general structural representations, two task-specific encoders for the VDA and RDA, and agent-specific Q-value decoders for predicting node-wise returns.

Given an undirected graph  $G = (V, E)$ , the model takes as input a sparse adjacency matrix  $A \in \mathbb{R}^{N \times N}$  and a node feature matrix  $X \in \mathbb{R}^{N \times d_{\text{in}}}$ , where  $N = |V|$  is the number of nodes. Since our focus is on topology-driven compression, we operate on graphs without explicit node features. Accordingly, we use an all-one matrix  $X = \mathbf{1}_{N \times d_{\text{in}}}$  as uniform input.

The shared encoder  $f_{\text{shared}}$  produces initial node embeddings by applying a graph convolutional layer:

$$H^{(0)} = \sigma(AXW_1), \quad H^{(0)} \in \mathbb{R}^{N \times d},$$

where  $W_1 \in \mathbb{R}^{d_{\text{in}} \times d}$  is a learnable weight matrix and  $\sigma(\cdot)$  denotes the ReLU activation. A global graph embedding is computed via mean pooling:

$$z^{(0)} = \text{GraphPool}(H^{(0)}), \quad (H^{(0)}, z^{(0)}) = f_{\text{shared}}(A, X).$$

Based on the shared representation, each agent applies a task-specific encoder to iteratively refine node and graph embeddings. Let  $f_{\text{vda}}$  and  $f_{\text{rda}}$  denote the encoders for VDA and RDA, respectively. These encoders follow the same architecture but are parameterized independently to support task-specific adaptation.

At each layer  $l$ , the encoders update node and graph embeddings as follows:

$$H^{(l+1)} = \sigma \left( \text{Linear} \left( [AH^{(l)}W_3 \parallel z^{(l)}W_2] \right) \right), \quad (6)$$

$$z^{(l+1)} = \sigma \left( \text{Linear} \left( [\text{GraphPool}(H^{(l+1)})W_3 \parallel z^{(l)}W_2] \right) \right). \quad (7)$$

This process is repeated for  $\ell$  layers, yielding the final task-specific representations:

$$\begin{aligned} H_{\text{vda}}^{(\ell)} &= f_{\text{vda}}(H^{(0)}, z^{(0)}), & z^{\text{vda}} &= \text{GraphPool}(H_{\text{vda}}^{(\ell)}), \\ H_{\text{rda}}^{(\ell)} &= f_{\text{rda}}(H^{(0)}, z^{(0)}), & z^{\text{rda}} &= \text{GraphPool}(H_{\text{rda}}^{(\ell)}). \end{aligned}$$

Finally, each agent uses an independent Q-value decoder to evaluate the expected return of removing a node. Given a node embedding  $h_i$  and a graph embedding  $z$ , the Q-value is computed by a multilayer perceptron:

$$Q_i = \text{MLP}([h_i \parallel z]), \quad (8)$$

where the MLP consists of fully connected layers with ReLU activations and layer normalization. This encoder–decoder architecture allows each agent to make context-aware decisions based on both local node properties and global structural cues.

### Trajectory-Level Return Reward Shaping

In reinforcement learning, sparse rewards pose a significant challenge, as agents often receive meaningful feedback only after completing an entire trajectory (Eschmann 2021). This issue is particularly salient in graph-based node removal tasks, where the importance of each action is only revealed in hindsight through its cumulative effect on graph robustness. To address this, we design a reward shaping mechanism that assigns dense, step-wise rewards to individual state-action pairs, guided by trajectory-level returns.

Specifically, in each exploration episode, the agent interacts with the environment to produce a trajectory  $\tau = \{(s_0, a_0), (s_1, a_1), \dots, (s_{T-1}, a_{T-1})\}$  consisting of state-action pairs. The overall quality of this trajectory is measured by a ground-truth return  $\mathcal{R}_{\text{true}}(\tau)$ , which differs across agents due to their task-specific goals.

**VDA True Return.** Defined as the relative drop in pairwise connectivity between the current original graph  $G_0$  and the final graph  $G_T$ :

$$\mathcal{R}_{\text{true}}^{\text{vda}}(\tau) = P_{\text{conn}} = \frac{F(G_0) - F(G_T)}{F(G_0)}, \quad (9)$$

this reward encourages the VDA to identify structurally critical nodes whose removal results in maximal connectivity degradation.

**RDA True Return.** Defined as a penalty-based score that subtracts from an initial value of 1.0:

$$\mathcal{R}_{\text{true}}^{\text{rda}}(\tau) = 1 - (\omega_1 \cdot P_{\text{conn}} + \omega_2 \cdot P_{\text{delete}} + \omega_3 \cdot P_{\text{embed}}), \quad (10)$$

where the weights satisfy  $\omega_1 + \omega_2 + \omega_3 = 1$ . The remaining two terms are defined as:

$$\begin{aligned} P_{\text{delete}} &= \frac{|V_{\text{vit}} \cap V_{\text{rem}}|}{|V_{\text{vit}}|}, \\ P_{\text{embed}} &= 1 - \cos(\mathbf{h}_{V_{\text{vit}}}, \mathbf{h}'_{V_{\text{vit}}}). \end{aligned} \quad (11)$$

Here,  $V_{\text{vit}}$  denotes the set of vital nodes identified by VDA,  $V_{\text{rem}}$  is the set of nodes removed by RDA, and  $\mathbf{h}_{V_{\text{vit}}}$ ,  $\mathbf{h}'_{V_{\text{vit}}}$  are the mean embeddings of  $V_{\text{vit}}$  before and after node removal. By incorporating feedback from VDA-identified vital nodes, structural connectivity, and semantic embeddings, this reward encourages RDA to preserve essential components while identifying redundancy from multiple perspectives.

**Reward Network Design.** To propagate trajectory-level supervision to each decision step, we design agent-specific reward networks  $\mathcal{R}_\phi^{\text{vda}}(s_t, a_t)$  and  $\mathcal{R}_\phi^{\text{rda}}(s_t, a_t)$  that estimate the reward for each state-action pair. These networks are trained independently and conditioned on task-specific objectives. The reward for step  $t$ , denoted as  $r_t$ , is computed as:

$$r_t = \tanh(W_2 \cdot \text{ReLU}(W_1[\mathbf{h}_G \| \mathbf{h}_a])), \quad (12)$$

where  $\mathbf{h}_G$  and  $\mathbf{h}_a$  denote the graph-level and action-specific embeddings of the current state-action pair, respectively, and  $W_1$ ,  $W_2$  are learnable weight matrices. The use of  $\tanh(\cdot)$  enables the reward signal to take both positive and negative values, allowing the network to penalize undesirable actions and encourage behavior-sensitive learning.

**Reward Network Optimization.** The predicted return of a trajectory is obtained by summing the step-wise rewards produced by the reward network:

$$R_{\text{pred}}(\tau) = \sum_{t=0}^{T-1} r_t \quad (13)$$

During training, completed trajectories and their true returns are stored in a replay buffer. The reward network is optimized by minimizing the mean squared error between predicted and true returns:

$$L_{\text{reward}}(\phi) = \mathbb{E}_{\tau \sim \pi} \left[ (\mathcal{R}_{\text{true}}(\tau) - R_{\text{pred}}(\tau))^2 \right]. \quad (14)$$

**Affine Return Alignment.** In our setting, the true trajectory returns, whether based solely on pairwise connectivity

or on multi-faceted penalties including connectivity are always bounded within  $[0, 1]$ . In contrast, the predicted cumulative rewards from the reward network may fall in a broader and potentially unbounded range, such as  $[-1, 1]$ . This mismatch in scale and distribution can lead to inconsistent gradients and unstable training, especially when minimizing the gap between predicted and true returns.

To address this issue, we introduce an affine transformation that aligns the predicted returns to the scale of the true returns:

$$\mathcal{R}_{\text{affine}}(x) = \alpha x + \beta. \quad (15)$$

The affine parameters  $\alpha$  and  $\beta$  are learned via least-squares regression on  $(R_{\text{pred}}, \mathcal{R}_{\text{true}})$  pairs from the replay buffer and frozen after convergence. This ensures that predicted returns align with the true scale, improving training stability.

The final training objective for the reward network becomes:

$$L_{\text{reward}}(\phi) = \mathbb{E}_{\tau \sim \pi} \left[ (\mathcal{R}_{\text{true}}(\tau) - \mathcal{R}_{\text{affine}}(R_{\text{pred}}(\tau)))^2 \right]. \quad (16)$$

This transformation not only improves numerical stability but also guarantees policy consistency. A formal justification is provided in the appendix, showing that such affine shaping does not affect the optimal policy under the original return formulation.

## State–Action–Level Prototype-Constrained Reward Shaping

While trajectory-level rewards provide global supervision, they often overlook local decision patterns that drive key structural changes. As nodes are progressively removed, certain critical deletions can trigger abrupt collapses in connectivity—akin to first-order phase transitions (Cao et al. 2021). To capture such fine-grained dynamics, we introduce a prototype-constrained reward shaping mechanism that provides step-wise feedback based on similarity to representative behavior fragments from past trajectories.

During training, each agent continuously collects trajectories into an experience buffer. As the buffer grows, we periodically extract the top- $K$  and bottom- $K$  trajectories based on ground-truth returns  $\mathcal{R}_{\text{true}}(\tau)$  to guide prototype construction. For each selected trajectory, we compute the agent-specific step-wise true return and identify the most critical decision point. Specifically, we select the step  $k^*$  with the highest return for top- $K$  trajectories and the lowest for bottom- $K$ :

$$k^* = \begin{cases} \arg \max_t \mathcal{R}_{\text{true}}(s_t, a_t), & \text{if top-}K \\ \arg \min_t \mathcal{R}_{\text{true}}(s_t, a_t), & \text{if bottom-}K \end{cases} \quad (17)$$

This ensures that the extracted prototypes capture the most representative local decisions characterizing high- and low-quality behaviors, that they are periodically updated based on the agent’s latest experience.

We then extract a fixed-length context window of  $n$  state–action pairs preceding that point:

$$\hat{\tau} = \{(s_i, a_i)\}_{i=k^*-n}^{k^*-1}. \quad (18)$$

Each state-action pair  $(s_t, a_t)$  is encoded into an embedding  $h_t$  using an agent-specific encoder  $f_{\text{agent}}$ , resulting in a temporal sequence  $\{h_1, h_2, \dots, h_n\}$ . This sequence is then processed by a GRU to capture its dynamic structure:

$$h_{\text{proto}} = \text{GRU} \left( \{h_i\}_{i=k^*-n}^{k^*-1} \right), \quad (19)$$

where the final hidden state summarizes the behavior pattern that causes critical structural changes.

Aggregating across  $K$  trajectories, we compute the mean embeddings for positive and negative prototypes:

$$h_{\text{pos}} = \frac{1}{K} \sum_{i=1}^K h_{\text{proto}}^{(i)}, \quad h_{\text{neg}} = \frac{1}{K} \sum_{i=1}^K \tilde{h}_{\text{proto}}^{(i)}. \quad (20)$$

These prototypes serve as reference patterns for evaluating decision quality.

When shaping the reward for a new state-action pair  $(s_t, a_t)$ , we extract its  $n$ -step historical context, encode it into an embedding:

$$h_{(s_t, a_t)} = \text{GRU} \left( \{h_i\}_{i=t-n}^{t-1} \right), \quad (21)$$

and compare it to the prototypes via cosine similarity. The reward target is defined as:

$$\begin{aligned} r_{\text{target}}(s_t, a_t) = & \text{clip} \left( \text{sim}_{\cos}(h_{(s_t, a_t)}, h_{\text{pos}}) \right. \\ & \left. - \text{sim}_{\cos}(h_{(s_t, a_t)}, h_{\text{neg}}), -1, 1 \right), \end{aligned} \quad (22)$$

assigning higher rewards to actions aligned with high-quality prototypes, and lower rewards to those resembling negative prototypes.

To propagate this auxiliary signal, the reward network  $\mathcal{R}_\phi(s, a)$  is trained to regress toward both the trajectory-level return and the prototype-derived target:

$$L_{\text{proto}}(\phi) = \mathbb{E}_{(s, a) \sim \pi} \left[ (\mathcal{R}_\phi(s, a) - r_{\text{target}}(s, a))^2 \right]. \quad (23)$$

The overall reward objective combines both learning signals:

$$L_{\text{total}}(\phi) = L_{\text{reward}}(\phi) + \lambda_{\text{proto}} \cdot L_{\text{proto}}(\phi), \quad (24)$$

where  $\lambda_{\text{proto}}$  is a hyperparameter controlling the influence of local prototype alignment. This hybrid shaping framework allows the agent to learn not only from final outcomes, but also from recurring patterns that characterize positive or negative local decisions.

### Cross-Agent Active-Follow Exploration

To encourage cooperation and improve policy generalization, we design a cross-agent exploration strategy in which the VDA and RDA alternate between *active* and *follower* roles. In each episode, both agents operate on the same graph, enabling bidirectional interaction under distinct task objectives.

**Phase I: VDA Leads, RDA Follows.** VDA explores the graph by executing its policy  $\pi_\theta^{\text{vda}}$ , generating a trajectory:

$$\tau_{\text{lead}}^{\text{vda}} = \{(s_t, a_t)\}_{t=0}^T, \quad a_t \sim \pi_\theta^{\text{vda}}(a | s_t), \quad (25)$$

which is evaluated under its ground-truth return  $\mathcal{R}_{\text{true}}^{\text{vda}}(\tau)$ .

To construct a structural prior for its counterpart, VDA selects a small subset of nodes from its trajectory that are deemed most informative for maintaining structural robustness. Following (Dorogovtsev, Goltsev, and Mendes 2008), we retain the top 15% to form an importance set:

$$\mathcal{I}^{\text{vda}} = \text{Top15\%} \left( Q^{\text{vda}}(s_t, a_t) \mid (s_t, a_t) \in \tau_{\text{lead}}^{\text{vda}} \right). \quad (26)$$

RDA then replays the same action sequence on the original graph:

$$\tau_{\text{follow}}^{\text{rda}} = \{(s'_t, a_t)\}_{t=0}^T, \quad (27)$$

where states  $s'_t$  is encoded through RDA's own graph encoder. This trajectory is evaluated as  $\mathcal{R}_{\text{true}}^{\text{rda}}(\tau)$  and stored in RDA's buffer  $\mathcal{D}_{\text{rda}}$ .

**Phase II: RDA Leads, VDA Follows.** RDA explores the graph using its policy conditioned on  $\mathcal{I}^{\text{vda}}$ :

$$\tau_{\text{lead}}^{\text{rda}} = \{(s_t, a_t)\}_{t=0}^T, \quad a_t \sim \pi_\theta^{\text{rda}}(a | s_t, \mathcal{I}^{\text{vda}}), \quad (28)$$

where the importance set provides a global prior highlighting structurally salient regions. VDA then follows this sequence to form:

$$\tau_{\text{follow}}^{\text{vda}} = \{(s''_t, a_t)\}_{t=0}^T, \quad (29)$$

where each state  $s''_t$  is encoded by VDA's own graph encoder. The resulting trajectory  $\tau_{\text{follow}}^{\text{vda}}$  is then evaluated using  $\mathcal{R}_{\text{true}}^{\text{vda}}(\tau)$  and stored in VDA's buffer  $\mathcal{D}_{\text{vda}}$ .

**Mutual Evaluation and Training.** Each agent maintains its own experience buffer  $\mathcal{D}_{\text{vda}}$  or  $\mathcal{D}_{\text{rda}}$ , into which both active and follower trajectories are stored. These samples update the policy  $\pi^{\text{agent}}\theta$ , reward model  $\mathcal{R}^{\text{agent}}\phi$ , and prototype extractor. Reinterpreting peer actions through self-objectives enables implicit transfer across asymmetric tasks, while contrasting decisions helps penalize unsafe patterns for safer, more generalizable policies.

## Experiments

### Experimental Setup

**Datasets.** We evaluate on five real-world graphs: Cora, Cite-seer, PubMed, Coauthor-Physics, and AirTraffic(Bojchevski and Günnemann 2017; Rossi and Ahmed 2015). All are treated as undirected and used without features or labels to isolate structural compression and robustness.

**Tasks and Evaluation Metrics.** To evaluate whether the compressed graph  $G'$  preserves the robustness of the original graph  $G$ , we simulate node removal attacks based on various centrality and influence measures, and record the degradation in pairwise connectivity over time. We use the *Robustness Preservation Similarity* (RPS) to measure the alignment between the degradation curves of  $G$  and  $G'$  under the same attack strategy. Let  $\mathbf{Rori} = [F_0, F_1, \dots, F_T]$  and  $\mathbf{Rcmp} = [F'_0, F'_1, \dots, F'_T]$  denote the pairwise connectivity sequences over  $T$  attack steps. RPS is computed as:

$$\text{RPS} = 1 - \frac{1}{T+1} \sum_{t=0}^T |F_t - F'_t|. \quad (30)$$

A higher RPS indicates that the compressed graph exhibits a connectivity degradation trend closer to that of the original graph, reflecting stronger robustness preservation.

We consider the following eight node attack strategies(Freitas et al. 2022): Degree Centrality, Collective Influence, Eigenvector Centrality, Betweenness Centrality, Closeness Centrality, Percolation Centrality, FINDER(Fan et al. 2020): a reinforcement learning method that removes nodes with the highest expected reward based on state-action embeddings.NIRM (Zhang and Wang 2022): ranks nodes by encoding local and global structure, trained on small synthetic graphs via score propagation.

The final robustness score of a method is reported by averaging the RPS scores across all  $X$  attack strategies:

$$\text{RPS}_{\text{mean}} = \frac{1}{|X|} \sum_{x \in X} \text{RPS}(x). \quad (31)$$

**Baselines.** We compare against five representative methods: SparRL (Wickman, Zhang, and Li 2022), DPGS (Zhou et al. 2021), GEC (Meng et al. 2024), MCGS (Chen, Zeng, and Cui 2022), and HyperSampling (Zhao et al. 2020).All methods are evaluated under uniform compression ratios, with node features set to constant one-vectors to ensure fair comparison in a structure-only setting.

**Topological Consistency under Compression** We evaluate how well each method preserves the static topological structure of the original graph using the Cora dataset as a representative benchmark. Table 1 reports four discrepancy metrics at  $\rho=0.5$ : average degree difference, clustering coefficient difference, path length within the largest component, and SP-Kernel similarity—collectively reflecting both local and global structural fidelity.

Cutter consistently aligns closest to the original graph across all metrics, achieving the best results in clustering, path length, and SP-Kernel similarity, with only a slight deviation in degree. This suggests that Cutter induces the least structural distortion during compression.

To complement these findings, we examine pairwise connectivity degradation under eight node attack strategies (Figure 2). The shaded regions indicate the variation range across different attack strategies, with lines showing mean trends. At  $\rho = 0.5$ , Cutter’s curve remains nearly identical to the original graph in early stages and shows only minor deviation thereafter. This affirms Cutter’s ability to preserve not only topological features but also robustness patterns throughout the entire attack process.

Method	Degree $\downarrow$	ClustCoeff $\downarrow$	PathLen $\downarrow$	SP-Kernel $\uparrow$
SparRL	1.309	0.097	1.856	0.871
DPGS	0.565	0.431	1.094	0.934
GEC	0.161	0.163	1.045	0.960
MCGS	0.175	0.384	0.793	0.917
HySamp	<b>0.145</b>	0.233	1.391	0.853
<b>Cutter</b>	0.167	<b>0.062</b>	<b>0.317</b>	<b>0.972</b>

Table 1: Topological differences at  $\rho=0.5$ .  $\downarrow$  means lower is better;  $\uparrow$  means higher is better.

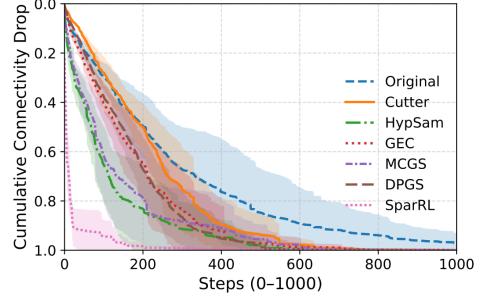


Figure 2: Connectivity drop under various node removal

**Robustness Preservation under Adversarial Attacks** We further assess whether different compression methods can preserve the robustness characteristics of the original graph under adversarial perturbations. The evaluation is conducted on five real-world datasets across three compression ratios: 0.5, 0.3, and 0.1, covering a spectrum from moderate to extreme compression. We adopt eight predefined node attack strategies, including centrality-based heuristics and learning-based methods. In each case, nodes are ranked by descending importance and progressively removed with a step size of 1% of the total node count, up to 40%. Both the original and compressed graphs are attacked using their respectively computed node sequences. At each step, we compute pairwise connectivity to construct degradation curves, and measure the RPS to quantify alignment between the compressed and original graphs.

Table 2 summarizes the RPS results across all datasets and compression settings. Cutter consistently achieves the highest RPS scores, significantly outperforming all baselines. Notably, even under aggressive compression where only  $\rho=0.1$  of the original nodes are retained, Cutter maintains degradation trends that closely mirror those of the original graph, demonstrating its superior ability to preserve structural robustness. Interestingly, on Citeseer, baseline methods such as GEC and MCGS exhibit robustness similar to Cutter when the compression ratio is moderate. However, their performance deteriorates significantly as the compression becomes more aggressive, particularly at  $\rho=0.1$ . This observation suggests that robustness preserved under light compression does not necessarily extend to more extreme scenarios. One possible explanation is that these methods rely heavily on specific structural features, such as dense local neighborhoods or community-like clusters, which tend to become unstable or disappear entirely under high compression. In contrast, Cutter maintains stable and consistent robustness across all compression levels, indicating stronger generalizability and adaptability in extreme settings.

### Ablation Study

We evaluate the effect of reward shaping by comparing RDA training with and without the reward network. As shown in Figure 3, the shaped variant achieves higher average returns (0.668 vs 0.513) and stabilizes over time, while the unshaped agent shows low variance but quickly plateaus at a suboptimal value.

$(\rho)$	Method	Cora	Cite	PubM	Phys	AirT
0.5	SparRL	0.822	0.927	0.938	0.826	0.837
	DPGS	0.916	0.961	0.931	0.805	0.920
	GEC	0.912	0.958	0.893	0.755	0.855
	MCGS	0.880	<b>0.966</b>	0.948	0.846	0.838
	HySamp	0.855	0.951	0.913	0.823	0.847
	<b>Cutter</b>	<b>0.953</b>	0.963	<b>0.965</b>	<b>0.869</b>	<b>0.938</b>
0.3	SparRL	0.814	0.886	0.886	0.693	0.768
	DPGS	0.873	0.925	0.893	0.759	0.835
	GEC	0.851	<b>0.934</b>	0.859	0.622	0.781
	MCGS	0.833	0.929	0.932	0.772	0.760
	HySamp	0.805	0.912	0.881	0.728	0.800
	<b>Cutter</b>	<b>0.885</b>	0.920	<b>0.939</b>	<b>0.820</b>	<b>0.875</b>
0.1	SparRL	0.781	0.853	0.829	0.512	0.724
	DPGS	0.793	0.843	0.802	0.737	0.758
	GEC	0.804	0.873	0.828	0.497	0.742
	MCGS	0.774	0.881	0.861	0.690	0.746
	HySamp	0.777	0.878	0.854	0.624	0.717
	<b>Cutter</b>	<b>0.832</b>	<b>0.895</b>	<b>0.872</b>	<b>0.758</b>	<b>0.780</b>

Table 2: RPS scores under different compression ratios. Higher is better.

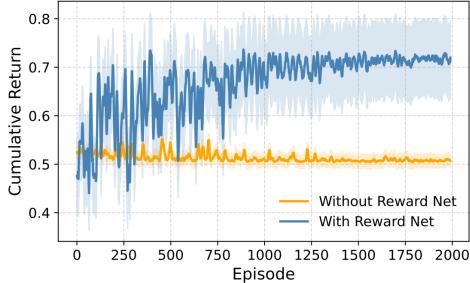


Figure 3: Ablation study on the effect of the reward network

mal level. Since each episode removes only 30% of nodes, the unshaped agent operates under strict constraints and struggles to discover effective strategies. The reward network, defined in Equation 11, combines penalties from connectivity loss, critical-node deletion, and embedding distortion, providing soft guidance that prevents over-compression and preserves structural robustness.

## Related Work

**Graph compression** has long been a fundamental technique for reducing the computational burden of graph algorithms. Due to differences in downstream tasks and algorithmic motivations, the literature has developed several closely related yet distinct research directions within this domain. Graph coarsening simplifies the graph structure by merging nodes or edges to improve the efficiency and stability of GNN training. For instance, (Chen et al. 2021) presents a unified coarsening framework, while (Meng et al. 2024) extends the concept of elementary collapse from algebraic topology to guide graph reduction. Graph sampling selects representative nodes or edges to construct a subgraph that preserves essential structural properties. (Chen, Zeng, and Cui

2022) proposes a hybrid sampling approach combining random node sampling and breadth-first search, whereas (Zhao et al. 2020) adopts a hybrid strategy that integrates triangle-based and cut-point-based heuristics to retain critical minority structures such as super pivots and ties, while balancing majority retention via greedy optimization. Graph summarization focuses on producing compact representations that capture global structural semantics. (Liu et al. 2018b) introduces Condense, an MDL-based framework that identifies diverse patterns (e.g., cliques, stars) through ensemble clustering and assembles supergraphs using k-step selection. (Zhou et al. 2021) proposes DPGS, which preserves node degrees and spectral properties through a configuration-based reconstruction scheme and MDL-driven supernode merging. Graph sparsification aims to reduce redundancy by removing or reweighting edges to obtain a sparse approximation of the original graph. Methods in (Wickman, Zhang, and Li 2022) and (Wu and Chen 2020) exemplify this direction by preserving structural fidelity while enhancing computational efficiency.

**Reinforcement learning (RL)** has proven effective for graph-related tasks due to its ability to model sequential decisions and handle delayed feedback. Prior work includes estimating node importance for robustness evaluation (Fan et al. 2020), enabling long-range reasoning through dual-agent coordination (Zhang et al. 2022a), and formulating graph sparsification as an RL problem (Wickman, Zhang, and Li 2022). A central challenge in these applications is the sparse reward issue, where agents receive feedback only at the end of long trajectories, limiting learning efficiency. To alleviate this, reward shaping introduces intermediate signals, with potential-based shaping preserving policy optimality (Ng, Harada, and Russell 1999). Recent approaches refine shaping via embedding similarity, action confidence, or trajectory-level guidance, including self-supervised ranking preservation (Memarian et al. 2021) and shaping agents modeled as Markov games (Mguni et al. 2023). Our work is also related to experience transfer (Brys et al. 2015) and safe exploration under structural constraints (Karimpanal et al. 2020).

## Conclusion

We propose Cutter, a dual-agent reinforcement learning framework for graph compression that preserves structural robustness under adversarial attacks. By integrating reward shaping, prototype-guided learning, and cross-agent exploration, Cutter effectively addresses the sparse reward challenge in graph decision-making. Experiments on real-world graphs show that it achieves high compression while maintaining robustness patterns under targeted node removal. Cutter runs independently at inference time, requiring no external modules or handcrafted rules. Though currently designed for topology-only compression, it can be extended to incorporate attributes and support tasks like GCN training or link prediction, demonstrating strong potential for scalable, robustness-aware graph compression.

## Acknowledgments

This work is supported by the Natural Science Foundation of China (No. 62402398, No. 72374173), the Natural Science Foundation of Chongqing (No. CSTB2025NSCQ-GPX1082), the University Innovation Research Group of Chongqing (No. CXQT21005), the Chongqing Graduate Research and Innovation Project (No. CYB23124), the Fundamental Research Funds for the Central Universities (No. SWU-KR24025, No. SWU-XDJH202303) and the High Performance Computing clusters at Southwest University.

## References

- Artime, O.; Grassia, M.; De Domenico, M.; Gleeson, J. P.; Makse, H. A.; Mangioni, G.; Perc, M.; and Radicchi, F. 2024. Robustness and resilience of complex networks. *Nature Reviews Physics*, 6(2): 114–131.
- Battiston, F.; Amico, E.; Barrat, A.; Bianconi, G.; Ferraz de Arruda, G.; Franceschiello, B.; Iacopini, I.; Kéfi, S.; Latora, V.; Moreno, Y.; et al. 2021. The physics of higher-order interactions in complex systems. *Nature physics*, 17(10): 1093–1098.
- Besta, M.; Hoefler, T.; Besta, M.; and Hoefler, T. 2018. Survey and taxonomy of lossless graph compression and space-efficient graph representations. arXiv 2018. *arXiv preprint arXiv:1806.01799*.
- Bojchevski, A.; and Günnemann, S. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*.
- Brys, T.; Harutyunyan, A.; Suay, H. B.; Chernova, S.; Taylor, M. E.; and Nowé, A. 2015. Reinforcement Learning from Demonstration through Shaping. In *IJCAI*, 3352–3358.
- Cao, Y.-Y.; Liu, R.-R.; Jia, C.-X.; and Wang, B.-H. 2021. Percolation in multilayer complex networks with connectivity and interdependency topological structures. *Communications in Nonlinear Science and Numerical Simulation*, 92: 105492.
- Chen, T.; Sui, Y.; Chen, X.; Zhang, A.; and Wang, Z. 2021. A unified lottery ticket hypothesis for graph neural networks. In *International conference on machine learning*, 1695–1706. PMLR.
- Chen, W.-t.; Zeng, A.; and Cui, X.-h. 2022. Preserving the topological properties of complex networks in network sampling. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(3).
- Dey, R.; and Salem, F. M. 2017. Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, 1597–1600. IEEE.
- Dorogovtsev, S. N.; Goltsev, A. V.; and Mendes, J. F. 2008. Critical phenomena in complex networks. *Reviews of Modern Physics*, 80(4): 1275–1335.
- Eschmann, J. 2021. Reward function design in reinforcement learning. *Reinforcement learning algorithms: Analysis and Applications*, 25–33.
- Fan, C.; Zeng, L.; Sun, Y.; and Liu, Y.-Y. 2020. Finding key players in complex networks through deep reinforcement learning. *Nature machine intelligence*, 2(6): 317–324.
- Fan, W.; Li, Y.; Liu, M.; and Lu, C. 2021. Making graphs compact by lossless contraction. In *Proceedings of the 2021 International Conference on Management of Data*, 472–484.
- Freitas, S.; Yang, D.; Kumar, S.; Tong, H.; and Chau, D. H. 2022. Graph vulnerability and robustness: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6): 5915–5934.
- Kang, S.; Lee, K.; and Shin, K. 2022. Personalized graph summarization: formulation, scalable algorithms, and applications. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2319–2332. IEEE.
- Karimpanal, T. G.; Rana, S.; Gupta, S.; Tran, T.; and Venkatesh, S. 2020. Learning transferable domain priors for safe exploration in reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–10. IEEE.
- Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
- Li, M.; Liu, R.-R.; Lü, L.; Hu, M.-B.; Xu, S.; and Zhang, Y.-C. 2021. Percolation on complex networks: Theory and application. *Physics reports*, 907: 1–68.
- Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12): 2935–2947.
- Liu, Y.; Safavi, T.; Dighe, A.; and Koutra, D. 2018a. Graph summarization methods and applications: A survey. *ACM computing surveys (CSUR)*, 51(3): 1–34.
- Liu, Y.; Safavi, T.; Shah, N.; and Koutra, D. 2018b. Reducing large graphs to small supergraphs: a unified approach. *Social Network Analysis and Mining*, 8(1): 17.
- Memarian, F.; Goo, W.; Lioutikov, R.; Niekum, S.; and Topcu, U. 2021. Self-supervised online reward shaping in sparse-reward environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2369–2375. IEEE.
- Meng, Y.; Li, R.-H.; Lin, L.; Li, X.; and Wang, G. 2024. Topology-preserving Graph Coarsening: An Elementary Collapse-based Approach. *Proceedings of the VLDB Endowment*, 17(13): 4760–4772.
- Mguni, D.; Jafferjee, T.; Wang, J.; Perez-Nieves, N.; Song, W.; Tong, F.; Taylor, M.; Yang, T.; Dai, Z.; Chen, H.; et al. 2023. Learning to shape rewards using a game of two partners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11604–11612.
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidje land, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.
- Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, 278–287. Citeseer.

- Rossi, R.; and Ahmed, N. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.
- Wickman, R.; Zhang, X.; and Li, W. 2022. A generic graph sparsification framework using deep reinforcement learning. In *2022 IEEE International Conference on Data Mining (ICDM)*, 1221–1226. IEEE.
- Wu, H.-Y.; and Chen, Y.-L. 2020. Graph sparsification with generative adversarial network. In *2020 IEEE international conference on data mining (ICDM)*, 1328–1333. IEEE.
- Zhang, D.; Yuan, Z.; Liu, H.; Xiong, H.; et al. 2022a. Learning to walk with dual agents for knowledge graph reasoning. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 36, 5932–5941.
- Zhang, F.; Linghu, Q.; Xie, J.; Wang, K.; Lin, X.; and Zhang, W. 2023. Quantifying node importance over network structural stability. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3217–3228.
- Zhang, J.; and Wang, B. 2022. Dismantling complex networks by a neural model trained from tiny networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2559–2568.
- Zhang, T.; Wang, X.; Liang, B.; and Yuan, B. 2022b. Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12): 9925–9939.
- Zhao, Y.; Jiang, H.; Chen, Q.; Qin, Y.; Xie, H.; Wu, Y.; Liu, S.; Zhou, Z.; Xia, J.; and Zhou, F. 2020. Preserving minority structures in graph sampling. *IEEE Transactions on Visualization and Computer Graphics*, 27(2): 1698–1708.
- Zhou, H.; Liu, S.; Lee, K.; Shin, K.; Shen, H.; and Cheng, X. 2021. Dpgs: Degree-preserving graph summarization. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, 280–288. SIAM.