LENS: NEIGHBORHOOD-PRESERVING PARTITIONING FOR EXACT GRAPH UNLEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine unlearning has emerged as a critical research frontier, driven by increasing privacy concerns and regulatory requirements. To circumvent the prohibitive cost of retraining from scratch, several efficient unlearning methods have been proposed. The Sharded, Isolated, Sliced, and Aggregated (SISA) framework is the most prominent for exact unlearning. However, applying partition-based approaches to graph-structured data remains a significant challenge. Naive partitioning disrupts the essential structural dependencies and node neighborhoods upon which Graph Neural Networks (GNNS) rely. In this work, we propose LENS (Local Edge-preserving Node Sharding), a novel partition-based framework for graph unlearning. LENS introduces a partitioning objective that explicitly prioritizes preserving node neighborhoods and maintaining the structural integrity required for effective message passing in GNNS. We provide a theoretically grounded optimization strategy for this objective. We further present a training-free aggregation scheme that combines entropy filtering with adaptive temperature scaling, outperforming more expensive learnable aggregators. Extensive experiments demonstrate that LENS consistently achieves superior performance on downstream tasks across multiple standard GNN backbones and datasets. Finally, we expose a critical limitation in existing partition-based methods that rely on learnable aggregators, challenging the assumption that these aggregators do not require frequent retraining. We show that unlearning high-degree nodes induces shifts in submodels' prediction that degrade learnable aggregators, whereas LENS's training-free aggregation scheme maintains utility and avoids hidden retraining costs.

1 INTRODUCTION

Machine unlearning has emerged as an important requirement for modern machine learning systems, driven by increasing privacy concerns and regulatory mandates such as the *right to be forgotten* (Cao & Yang, 2015). In deployed settings, user data may need to be removed due to legal obligations, outdated information, or malicious activity. A straightforward solution is to retrain the model from scratch after removing the revoked data. However, for large-scale models and continuously evolving systems, full retraining is often computationally prohibitive. This motivates the development of efficient unlearning methods that selectively remove the influence of specific training instances from the model (Bourtoule et al., 2021; Ginart et al., 2019; Golatkar et al., 2020; Guo et al., 2019; Neel et al., 2021).

Graphs present a challenging setting for machine unlearning. Many real-world systems, such as social networks, recommender systems, and biological interaction graphs, are naturally represented as graphs. Graph Neural Networks (GNNS) are widely used to model and solve machine learning tasks on graphs (Kipf, 2017; Hamilton et al., 2017). Unlike i.i.d. data, graphs encode rich relational dependencies, and the prediction of a single node may depend on the features and structure of its multi-hop neighborhood (Xu et al., 2019). Consequently, removing a node from the training graph requires eliminating not only its direct contribution but also its indirect influence on surrounding nodes (Chen et al., 2022). This is highly relevant when the predictive tasks are related to nodes.

One of the most widely adopted paradigms for machine unlearning is the Sharded, Isolated, Sliced, and Aggregated (SISA) framework (Bourtoule et al., 2021). Partition-based graph unlearning methods build on this shard-and-aggregate strategy by dividing the training graph into disjoint subgraphs

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.

and handling unlearning requests through retraining only the models of the affected shards, rather than the entire model. Representative approaches such as GraphEraser (Chen et al., 2022) extend this paradigm to graph-structured data and enable exact unlearning by isolating the influence of revoked nodes within specific partitions. Subsequent works, including GUIDE (Wang et al., 2023), further adapt partition-based unlearning to inductive settings by synthesizing missing neighborhood information across shards. Unlike the algorithmic exact unlearning, approximate unlearning methods edit the model parameters directly by estimating how the model's parameters would change after a deletion request. The estimation often relies on the concept of Influence Function (Wu et al., 2023; Dong et al., 2024; Yang et al., 2025) or heuristic finetuning (Cheng et al., 2023; Sinha et al., 2025).

Belonging to the family of exact unlearning, SISA-inspired methods offer stronger privacy and security assurance than approximate techniques. However, their effectiveness critically depends on the quality of the underlying graph partitioning. Real-world graphs are difficult to shard without disrupting the local neighborhoods required for GNN message passing, and severing cross-partition edges often leads to substantial degradation in model utility (Chen et al., 2022; Wang et al., 2023). A fundamental limitation of existing methods is their sole reliance on global partitioning objectives, such as minimizing the total number of cut edges, which fail to reflect the heterogeneous sensitivity of nodes to neighborhood disruption. In particular, the removal of a small number of edges may be inconsequential for high-degree nodes but catastrophic for low-degree nodes whose representations depend on only a few neighbors. This indicates that, for GNNS, preserving local neighborhood structure is more critical than minimizing global edge cuts. Consequently, effective unlearning calls for partitioning strategies that explicitly account for neighborhood-level structural integrity rather than relying solely on global measures. Furthermore, existing approaches typically rely on heuristic procedures (e.g., label propagation) or lightweight neural models for graph partitioning. The convergence of these techniques is questionable, highlighting the absence of strong theoretical foundations and motivating the need for more principled, theoretically grounded partitioning strategies.

To address these limitations, we propose a neighborhood preserving partition-based framework LENS (Local Edge-preserving Node Sharding) for exact graph unlearning. LENS partitions the training graph by minimizing a relative neighborhood loss that quantifies the fraction of the neighborhood of each node disrupted by partitioning. However, in highly homophilous graphs, minimizing cross-partition edges risks grouping similar nodes into the same shard, which degrades model discriminability. To overcome this while preserving structural context, independent GNN submodels are trained on their shards using a degree difference-weighted contrastive loss. Finally, to mitigate the loss of global information, we present a non-training aggregation scheme that filters high-entropy predictions and applies adaptive temperature scaling before taking their average. This design avoids parameter updates entirely, preventing the utility degradation caused by distribution shifts of submodels' output.

Our main contributions are summarized as follows:

- *Neighborhood-aware Partitioning.* We propose LENS, a framework that explicitly preserves node neighborhoods. Unlike prior graph partitioning objectives, LENS optimizes a relative neighborhood loss that accounts for the heterogeneous impact of edge removal across nodes to better align with the message-passing mechanism of GNNS.
- *Theoretical Insights.* We show that the proposed neighborhood-aware partitioning objective is submodular. Leveraging this, we derive a convex relaxation via the Lovász extension and develop an efficient optimization procedure with balance constraints for unlearning.
- *Degree Difference-Weighted Contrastive Training.* We identify that minimizing cross-partition edges in homophilous graphs forces similar nodes into the same subgraph, reducing submodel discriminability. Additionally, because submodels are trained on fragmented subgraphs but evaluated on the complete graph during inference, representations are prone to over-smoothing. To address these challenges, we introduce a degree difference-weighted contrastive loss for submodel training.
- *Empirical Validation*. We demonstrate that LENS consistently outperforms or matches state-of-the-art partition-based methods across various GNN backbones and datasets. Our results show that LENS achieves the highest average ranking and utility stability compared to the baselines, which exhibit volatile performance across different settings.
- *Risks of Learnable Aggregation*. We challenge the common misconception that learnable aggregators only require retraining when a deleted node belongs to their tuning sample. We

expose a critical vulnerability where unlearning high-degree nodes induces significant output distribution shifts in retrained submodels, causing severe utility degradation in learnable aggregators. Our empirical analysis demonstrates that LENS's training-free, non-parametric aggregation scheme dynamically adapts to these shifts and avoids retraining overhead.

Reproducibility. Our code is at: `https://anonymous.4open.science/r/0B12/`.

2 PROBLEM DEFINITION

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$ denote an undirected training graph, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges, with node features $X \in \mathbb{R}^{|\mathcal{V}| \times d}$. We consider a GNN model $\mathcal{M}$ trained on G for a downstream prediction task. Each node $u \in \mathcal{V}$ corresponds to an entity in the system (e.g., a user in a social network), and its representation is computed via message passing over its neighborhood.

Node Classification. As the downstream learning task, we focus on node classification. Let $\mathcal{V}_{\text{train}} \subseteq \mathcal{V}$ denote the set of labeled training nodes with corresponding labels $Y_{\text{train}} = \{y_v \mid v \in \mathcal{V}_{\text{train}}\}$. The objective is to learn a predictor f_Θ that assigns each node $v \in \mathcal{V}$ a probability distribution over C classes, and whose performance is evaluated on a test set $\mathcal{V}_{\text{test}} \subseteq \mathcal{V} \setminus \mathcal{V}_{\text{train}}$. A GNN training algorithm $\mathcal{M}$ takes as input the dataset $D = (\mathcal{V}_{\text{train}}, \mathcal{E}, X, Y_{\text{train}})$ and outputs the model parameters Θ defining the predictor f_Θ.

Graph Unlearning. Unlearning aims to remove the influence of specific training elements from the learned model (Chen et al., 2022). Following prior work (Wang et al., 2023), unlearning requests on graphs can be categorized into three types: *(i) Feature unlearning:* removing the feature vector X_u of a node u; *(ii) Edge unlearning:* removing an edge $e(u, v) \in \mathcal{E}$; *(iii) Node unlearning:* removing a node u, its features together with all of its incident edges. In this work, we focus on *node unlearning*, as it subsumes both feature and edge unlearning and represents the most general and challenging setting.

Definition 2.1 (**Node Unlearning**). Given a trained model f_Θ and a deletion request specified by a subset of training nodes $\Delta\mathcal{V}_{\text{train}} \subseteq \mathcal{V}_{\text{train}}$, let the updated dataset be

$$D' = \big(\mathcal{V}_{\text{train}} \setminus \Delta\mathcal{V}_{\text{train}},\, \mathcal{E}',\, X',\, Y'_{\text{train}}\big),$$

where X' and Y'_{train} denote the set of features and labels restricted to $\mathcal{V}_{\text{train}} \setminus \Delta\mathcal{V}_{\text{train}}$ respectively, and $\mathcal{E}' = \{(u, v) | u, v \notin \Delta\mathcal{V}_{\text{train}}\}$. Let Θ^* denote the parameters obtained by retraining from scratch on D' (the *gold* model $\mathcal{M}^*$). The goal is to design a node unlearning algorithm $\mathcal{M}'$ that takes the original parameters Θ and the deletion set $\Delta\mathcal{V}_{\text{train}}$ as input, and outputs the updated parameters Θ'.

We restrict our study to the transductive setting, where training and test nodes belong to a single graph. An effective unlearning method should satisfy the following criteria:

- **Model utility:** $\mathcal{M}'$ should achieve performance comparable to $\mathcal{M}^*$ on a held-out test set.
- **Efficiency:** The cost of obtaining $\mathcal{M}'$ should be significantly lower than full retraining and obtaining $\mathcal{M}^*$.
- **Security:** $\mathcal{M}'$ should not retain meaningful information about the revoked nodes, nor allow their membership and information to be inferred.

3 OUR METHOD: LENS

LENS supports exact unlearning while minimizing retraining cost and preserving predictive performance. It consists of three stages: (i) neighborhood-aware partitioning, which produces shards that preserve local message-passing structure; (ii) isolated training, which ensures retraining locality while using a degree difference-weighted contrastive loss to counter homophilic over-clustering and over-smoothing; and (iii) adaptive calibration and aggregation, which restores global information during inference through entropy filtering and temperature scaling without additional training.

Unlike traditional sharding (partition-based) methods that focus on global edge-cut counts, LENS prioritizes the structural integrity for GNNS by minimizing the total relative neighborhood loss across

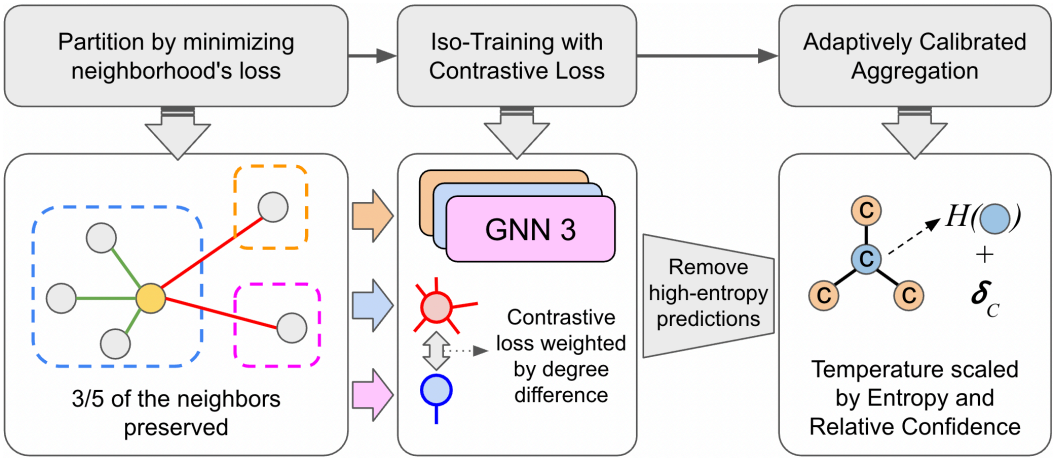


Figure 1: Pipeline of LENS: (1) Partition the graph $\mathcal{G}$ such that the total relative neighborhood loss of all nodes is minimized. (b) Train isolated submodels on each subgraph with degree-difference-weighted contrastive loss (c) Calibrate predictions with entropy and relative confidence-scaled temperature and discard predictions whose entropy is above the threshold before taking the average.

all nodes. Specifically, we define the loss for each node as the fraction of its neighbors assigned to other shards or partitions. We prove that this formulation is an instance of the Submodular k-way Multiway Partition problem (Chekuri & Ene, 2011). With this, we can derive a convex relaxation of this objective via the Lovász extension and solve it effectively using Projected Subgradient Descent. Following partitioning, independent GNN submodels are trained on disjoint subgraphs using a degree difference-weighted contrastive loss. This isolation guarantees that unlearning requires retraining only the submodel containing the revoked node. For inference, we use a training-free aggregation scheme that filters out high-entropy predictions and applies adaptive temperature scaling before averaging predictions. Our pipeline is illustrated in Figure 1.

3.1 NEIGHBORHOOD-AWARE PARTITIONING

The efficacy of GNNS is fundamentally rooted in the message-passing paradigm, where a node's representation is iteratively updated based on its local neighborhood. Conventional graph partitioning objectives, such as Min-Cut, prioritize minimizing the absolute number of edges lost across shards. However, such objectives fail to account for the heterogeneous impact of structural loss across nodes. For low-degree nodes, even a small number of lost edges can eliminate most of their message-passing context, while the same loss is negligible for high-degree nodes. To preserve the structural integrity required for high-utility unlearning, we propose the Relative Neighborhood Loss (RNL). Rather than counting total edge cuts, RNL measures the proportion of a node's neighborhood that is lost due to partitioning, a metric that is poorly aligned with GNN behavior.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the input graph and k be the number of disjoint shards. We denote the set of nodes in the i-th shard as $\mathcal{V}_i$, where $\bigcup_{i=1}^{k} \mathcal{V}_i = \mathcal{V}$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for $i \neq j$. For a node $u \in \mathcal{V}_i$, we define its local *Relative Neighborhood Loss (RNL)* as the ratio of its neighbors located outside its assigned shard $\mathcal{V}_i$ to its total degree:

$$\mathcal{L}_{rel}(u, \mathcal{V}_i) = \frac{\left|\mathcal{N}(u) \setminus \mathcal{V}_i\right|}{\left|\mathcal{N}(u)\right| + \epsilon}$$

where $\mathcal{N}(u)$ denotes the set of first-order neighbors of node u, and ϵ is an arbitrarily small, positive number to avoid division by zero caused by isolated nodes. This normalization ensures that structural disruption is measured in relation to the available context of a node, rather than in absolute terms. While RNL provides a fine-grained measure of local structural preservation, we observe that global edge-cut minimization is inherently correlated with local neighborhood retention. To leverage this relationship and accelerate the convergence of our optimization process, we augment our primary objective with a traditional Min k-cut component. Let $E(\mathcal{V}_i, \bar{\mathcal{V}}_i)$ denote the set of edges with one

endpoint in $\mathcal{V}_i$ and the other in its complement. Our final partitioning objective is defined as:

$$\min_{\mathcal{V}_1,\ldots,\mathcal{V}_k} \sum_{i=1}^{k} \left(\sum_{u \in \mathcal{V}_i} \mathcal{L}_{rel}(u, \mathcal{V}_i) + \lambda_g |E(\mathcal{V}_i, \bar{\mathcal{V}}_i)| \right) \tag{1}$$

where $\lambda_g \geq 0$ is a hyperparameter that controls the trade-off between local neighborhood preservation and global connectivity, allowing the objective to benefit from faster convergence without sacrificing per-node structural integrity.

Theorem 3.1. *Finding a k-way partition that minimizes the augmented objective defined in Eq. 1 is a Submodular k-way partition problem.*

We provide the detailed proof in Appendix A.2. We now establish that the proposed objective admits a well-behaved combinatorial structure, which is critical for efficient optimization. Even though the problem is NP-hard (Zhao et al., 2005), the submodular structure enables us to build a scalable approximation algorithm with theoretical guarantees even for large graphs. In the next section, we apply a convex relaxation to optimize our objective (Eq. 1).

3.2 Convex Relaxation & Balance Constraints

To solve the discrete multiway partition problem, we follow the relaxation framework by Chekuri & Ene (2011). Given the submodular set function $g : 2^{\mathcal{V}} \to \mathbb{R}$, we utilize its Lovász extension $\hat{g} : [0,1]^n \to \mathbb{R}$ to project the discrete objective into a continuous, convex domain. The Lovász extension is defined as:

$$\hat{g}(\mathbf{x}) = \mathbb{E}_{\theta \in [0,1]}[g(\mathbf{x}^\theta)] = \int_0^1 g(\mathbf{x}^\theta) d\theta$$

where $\mathbf{x}^\theta$ is a thresholded binary vector. It has been proven that $\hat{g}$ is convex if and only if g is submodular (Lovász, 1983).

For each node $u \in \mathcal{V}$, we define variables $x(u,i) \in [0,1]$ representing the degree of membership of node u to shard i. Let $\mathbf{x}_i$ be the vector $[x(u_1,i), \ldots, x(u_n,i)]$. We formulate the resulting convex relaxation as follows:

$$\min \quad \sum_{i=1}^{k} \hat{g}(\mathbf{x}_i) \;\; \text{s.t.} \; \sum_{i=1}^{k} x(u,i) = 1, \forall v \in \mathcal{V}; x(t_i,i) = 1, \forall i; x(u,i) \geq 0, \quad \forall u, i \tag{2}$$

where t_i are the predetermined terminal nodes which must stay in their corresponding shard $\mathcal{V}_i$. Because $\hat{g}$ is convex if and only if g is submodular, this formulation guarantees a global minimum that can be achieved via first-order optimization methods. From the optimal solution to Eq. 2, Chekuri & Ene (2011) provide a rounding scheme to guarantee *2-approximation*. However, this rounding scheme results in highly unbalanced shards. This is not suitable for the unlearning task as revoked nodes are more likely to fall in the largest shard, which requires a significant amount of retraining time. To overcome this challenge, we first introduce the balance constraint to the relaxation in Eq. 2:

$$\min \quad \sum_{i=1}^{k} \hat{g}(\mathbf{x}_i) \;\; \text{s.t. constraints in Eq. 2 hold } \textit{and} \sum_{u \in \mathcal{V}} x(u,i) = \frac{|\mathcal{V}|}{k}, \quad \forall i \tag{3}$$

Crucially, because the balance constraint is also linear, the optimization problem remains a convex program in the continuous space, preserving our ability to efficiently find a global optimum.

Practical Optimization. To solve the balanced relaxation in Eq. 3, we employ a specialized initialization and iterative optimization procedure. We begin by identifying terminal nodes $\{t_i\}_{i=1}^k$ using Bisecting k-means. To align initialization with the neighborhood-preserving objective, we construct node representations using localized graph convolutions before clustering. More specifically, the input features for k-means are generated via a localized graph convolution, where each node's features are aggregated with features from their h-hop neighbors. However, feature-based clustering may lead to grouping nodes too homogeneously. Injecting controlled noise prevents these overly homogeneous shards, which can otherwise reduce the expressive capacity of downstream GNN

submodels. To finalize the initialization, we apply the Sinkhorn-Knopp algorithm (Sinkhorn & Knopp, 1967) to strictly satisfy the row-sum and balance constraints.

We use Projected Subgradient Descent (Boyd et al., 2003) for the core optimization. The subgradient of the Lovász extension $\hat{g}$ at $\mathbf{x}_i$ is derived from the values of the discrete set function g. Specifically, let π be a permutation that sorts the components of $\mathbf{x}_i$ in descending order such that $1 = x_{\pi(0)} \geq x_{\pi(1)} \geq x_{\pi(2)} \geq \cdots \geq x_{\pi(n)} = x_{\pi(n+1)} = 0$. Let $S_j = \{\pi(1), \ldots, \pi(j)\}$ be the set of the j largest elements, with $S_0 = \emptyset$. The $\pi(j)$-th component of the subgradient vector $\partial \hat{g}(\mathbf{x}_i)$ is given by:

$$\partial \hat{g}(\mathbf{x}_i)_{\pi(j)} = g(S_j) - g(S_{j-1}) \tag{4}$$

The increment shown in Eq. 4 can be precisely calculated (see Appendix A.3), thereby enabling a dynamic programming approach to efficiently solve the convex relaxation.

Rounding. To guarantee strict balance constraints, the rounding process can be formulated as a Linear Sum Assignment Problem (LSAP) (Malinen & Fränti, 2014). Let $X \in [0,1]^{|\mathcal{V}| \times k}$ denote the soft assignment matrix from the continuous optimization. We define a cost matrix $C \in \mathbb{R}^{|\mathcal{V}| \times M}$ by replicating each shard $i \in \{1, \ldots, k\}$ into $m = \lceil |\mathcal{V}|/k \rceil$ discrete capacity slots, yielding $M = k \cdot m$ total available slots. For node $u \in \mathcal{V}$ and slot $j = (i-1)m + r$ (where $1 \leq r \leq m$), the assignment cost is defined as $C_{u,j} = -x(u,i)$. This discrete matching problem on C can be solved exactly in polynomial time via the Hungarian algorithm, or efficiently scaled via entropic optimal transport using Sinkhorn's algorithm (Cuturi, 2013) followed by an argmax operation for discrete assignments.

3.3 Isolated Training with Degree Difference-Weighted Contrastive Loss

In highly homophilous graphs, minimizing cross edges during sharding can group nodes with similar features and labels into the same partition, reducing the submodel's discriminative power. To counteract this without sacrificing edge retention, we introduce a contrastive loss term that encourages distinct node representations. Furthermore, while submodels are trained on isolated shards, they are evaluated on the complete graph at inference time. In this setting, low-degree nodes likely recover more missing neighbors during full-graph evaluation than high-degree nodes. This relative shift risks pulling low-degree and high-degree representations together, degrading the decision boundary. To provide a buffer against this over-smoothing effect, we weight the contrastive penalty by pairwise degree differences. Formally, let $S_{uv} = \frac{\mathbf{h}_u^\top \mathbf{h}_v}{\tau_s}$ be the τ_s-scaled cosine similarity between nodes u and v, and z_u be the Z-score standardized log-degree of node u. The degree difference-weighted contrastive loss is defined as:

$$\mathcal{L}_{\text{contras}} = \frac{1}{|\mathcal{V}_i|} \sum_{u \in \mathcal{V}_i} \log \sum_{v \in \mathcal{V}_i \setminus \{u\}} \exp\left(S_{uv} \frac{|z_u - z_v|}{\tau_d} \right)$$

where τ_d scales the degree difference penalty. The entire training objective is $\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_c \mathcal{L}_{\text{contras}}$, where λ_c controls the contrastive regularization.

3.4 Adaptive Calibration and Aggregation

Training GNN submodels on isolated subgraphs exposes two key challenges: (i) submodels fail on nodes with out-of-distribution local structures, and (ii) contrary to the common view that GNNs are underconfident, training on smaller, fragmented subgraphs may induce overfitting and overconfidence. To address both issues without learnable parameter risks during unlearning, we introduce a non-parametric aggregation strategy. For each target node u, we filter out submodels whose normalized prediction entropy $\hat{H}_i(u)$ exceeds threshold τ_h. For the remaining submodels, we lower prediction confidence by scaling logits $\mathbf{z}_i(u)$ with an adaptive temperature $T_i(u) = T_{\text{ent}}(u) + T_\delta(u)$ where:

$$T_{\text{ent}}(u) = \min\left(1 + \gamma_e(1 - \hat{H}_i(u)), T_{\max}\right), \quad T_\delta(u) = \gamma_n \max\left(0, \bar{c}_i(\mathcal{N}(u)) - c_i(u)\right).$$

Here, $T_{\text{ent}}(u)$ forces lower-entropy models to use higher temperatures, inspired by Balanya et al. (2024). Additionally, following Hsu et al. (2022), we inflate temperature by $T_\delta(u)$ when a submodel's confidence $c_i(u) = \max_y p_i(y|u)$ falls below the average confidence of node u's neighbors $\bar{c}_i(\mathcal{N}(u))$. Finally, $p_i^{\text{calib}}(y|u) = \text{softmax}(\mathbf{z}_i(u)/T_i(u))$ are averaged strictly across non-filtered submodels.

Table 1: Macro F1 score comparison. We run experiments with 7 seeds and report the mean and standard deviation. For our method LENS, Green indicates first rank, blue indicates second rank among unlearning methods. Notably, LENS outperforms other baselines in 8 out of 12 experimental settings across various datasets and GNN backbones.

Backbone	Method	Cora	CiteSeer	PubMed	BlogCatalog
GCN	Full Train	0.865 ± 0.0024	0.6744 ± 0.0051	0.8692 ± 0.002	0.7184 ± 0.0132
	GraphEraser	0.7065 ± 0.0304	0.6239 ± 0.0266	0.8507 ± 0.0028	0.6662 ± 0.0126
	GraphRevoker	0.7929 ± 0.0145	0.6538 ± 0.0269	0.8476 ± 0.004	0.7327 ± 0.0029
	GUIDE	0.8362 ± 0.0126	0.642 ± 0.0371	0.8455 ± 0.0049	0.6191 ± 0.0455
	LENS	0.8193 ± 0.0145	0.6886 ± 0.0083	0.8511 ± 0.001	0.6971 ± 0.0213
GAT	Full Train	0.8455 ± 0.0107	0.6732 ± 0.0117	0.8618 ± 0.0039	0.7268 ± 0.0233
	GraphEraser	0.6506 ± 0.0603	0.637 ± 0.0386	0.8418 ± 0.0039	0.5932 ± 0.023
	GraphRevoker	0.7855 ± 0.0238	0.6237 ± 0.0324	0.8401 ± 0.0072	0.659 ± 0.0243
	GUIDE	0.7475 ± 0.0395	0.6189 ± 0.0379	0.8057 ± 0.0128	0.6252 ± 0.0081
	LENS	0.8253 ± 0.0055	0.6878 ± 0.01	0.8442 ± 0.0021	0.5504 ± 0.0159
SAGE	Full Train	0.851 ± 0.0055	0.6945 ± 0.0052	0.8781 ± 0.0021	0.9602 ± 0.0024
	GraphEraser	0.6785 ± 0.0492	0.6461 ± 0.0177	0.8496 ± 0.0034	0.8852 ± 0.026
	GraphRevoker	0.7724 ± 0.0212	0.6472 ± 0.0174	0.8422 ± 0.0058	0.9105 ± 0.0091
	GUIDE	0.8294 ± 0.0114	0.5892 ± 0.03	0.8326 ± 0.0045	0.8188 ± 0.0237
	LENS	0.8234 ± 0.0101	0.6779 ± 0.0109	0.8501 ± 0.0022	0.9254 ± 0.0173

Empirically, our adaptive aggregation outperforms complex learnable alternatives while mitigating severe utility degradation risks. Learnable aggregation modules can be vulnerable to distribution shifts in submodel predictions caused by unlearned high-degree nodes, even when those nodes lie outside the aggregator's tuning set. As detailed in Section 4.2, this causes substantial performance drop unless the aggregator is retrained upon every deletion request, thus incurring computational overhead that our training-free scheme completely avoids.

4 EXPERIMENTAL RESULTS

4.1 MODEL UTILITY EVALUATION

We evaluate the effectiveness of LENS by measuring the predictive utility of the backbone model. To establish a comprehensive benchmark, we compare LENS against three state-of-the-art partition-based exact unlearning frameworks: GraphEraser (Chen et al., 2022), GraphRevoker (Zhang, 2024), and GUIDE (Wang et al., 2023). We also include the Full Train baseline, which represents the model trained on the full unpartitioned dataset, serving as the upper bound for model utility.

Table 2: Average rankings of unlearning methods. While individual baselines sometimes outperform our approach in specific settings, LENS demonstrates great consistency by securing the best average rank across all backbones.

Method	GCN	GAT	SAGE
GraphEraser	3.25	2.75	3
GraphRevoker	2.25	2.25	2.5
GUIDE	3	3.25	3.25
LENS	1.5	1.75	1.25

Experimental Setup. We conduct experiments across four standard graph datasets: Cora, CiteSeer, PubMed, and BlogCatalog. To ensure the generalizability of our findings, we employ three widely used GNN architectures as backbones: GCN (Kipf, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017). We use Macro F1 as our primary evaluation metric to account for class imbalances and ensure fairness in performance assessment. All unlearning methods use the same training hyperparameters [1], train-test splits, and the number of shards $k = 20$.

[1] except for GUIDE and LENS in GAT for BlogCatalog, which use one fewer GNN layer to avoid overfitting

Performance Analysis. Table 1 summarizes the Macro F1 scores for all methods. LENS consistently outperforms or matches the performance of existing state-of-the-art baselines. Specifically, LENS achieves the highest utility in 8 out of the 12 evaluated settings (combinations of backbone and dataset) and ranks second in three others. While some baselines like GUIDE or GraphRevoker perform well in specific scenarios (e.g., GUIDE on Cora with GCN), they lack the consistent performance across different architectures and datasets demonstrated by LENS.

Ranking and Consistency. To further quantify the robustness of our approach, we report the average rankings of all methods across GNN backbones in Table 2. LENS achieves an overall average rank of 1.5, significantly outperforming the runner-up, GraphRevoker, which holds an average rank of 2.33. Notably, LENS maintains the highest average rank for every individual GNN backbone tested: 1.5 for GCN, 1.75 for GAT, and 1.25 for GraphSAGE. This stability suggests that our neighborhood-aware partitioning and adaptive calibration are less sensitive to the underlying message-passing mechanism than other partitioning-based methods.

4.2 RISKS OF LEARNABLE AGGREGATION

Among the baselines, GraphRevoker achieves the second-highest utility. A key component of its design is a learnable aggregator: an MLP trained on a small sample of training nodes to combine the output predictions of individual submodels. Prior work (Chen et al., 2022) claims that a learnable aggregator only requires retraining when a deletion request revokes one of the nodes in its tuning set. We challenge this claim and expose a vulnerability in learnable aggregation modules.

Motivation. When any node is unlearned, the corresponding submodel containing that node must be retrained to guarantee exact unlearning. However, removing a node, particularly a high-degree node, alters the local topology and message-passing dynamics of its shard, causing the retrained submodel's output probability distribution to shift. If the influential node lies outside the tuning set, its removal fails to trigger a retraining of the MLP aggregator, which continues to assume a stale output distribution across submodels. This mismatch may severely degrade overall utility.

Setup. To test this hypothesis, we isolate the aggregation mechanism within a shared SISA framework using identical partitioning and training parameters. We compare two aggregation schemes: (i) the parametric MLP aggregator of GraphRevoker, and (ii) the training-free, non-parametric aggregation scheme of LENS. Using a GAT backbone, we measure the drop in Macro F1 score before and after deletion across three datasets: CiteSeer, PubMed, and BlogCatalog, which have 50, 100, and 50 nodes removed respectively. To maximize disruption without triggering an update of the aggregator, we selectively remove the nodes with the highest degrees that are not in the aggregator's tuning set.

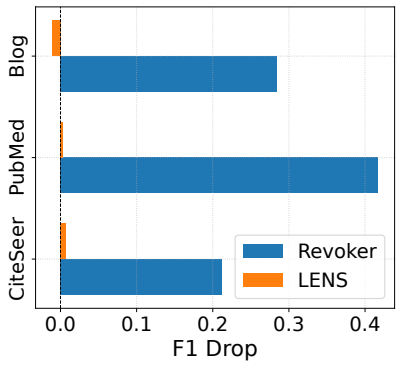
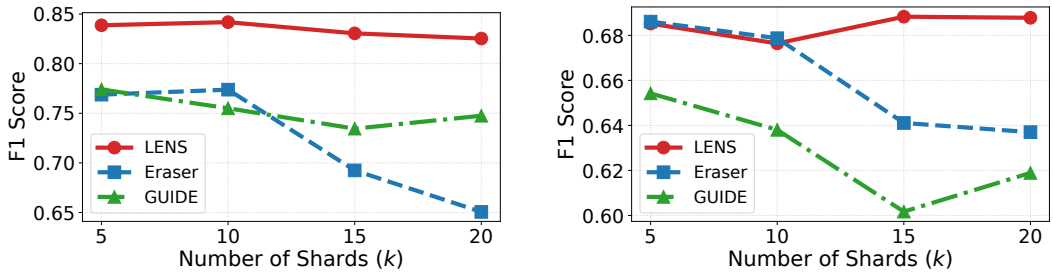


Figure 2: Macro F1 score drop before and after targeted node removal for GraphRevoker vs. LENS across datasets with GAT backbone. LENS is robust against this perturbation.

Empirical Results. As shown in Figure 2, targeted node removal causes GraphRevoker's performance to drop drastically, losing approximately half of its Macro F1 utility on average. Because the deleted nodes lie outside the aggregator's sampled set, the MLP is not triggered for retraining, leaving it incapable of handling the retrained submodels' shifted output distributions. Conversely, the predictive utility of LENS remains virtually unchanged across all three datasets. By leveraging entropy filtering and adaptive temperature scaling rather than static parametric weights, LENS dynamically accommodates distribution shifts without parameter updates. These results demonstrate that maintaining utility in learnable aggregators requires frequent retraining even for non-sampled nodes, incurring overhead that LENS effectively avoids.

4.3 ABLATION STUDY

In this section, we investigate the sensitivity of LENS to the number of shards k, a critical hyperparameter that governs the trade-off between unlearning efficiency and model utility. While increasing



(a) Dataset: Cora. Backbone: GAT.

(b) Dataset: CiteSeer. Backbone: GAT.

Figure 3: The effect of the number of shards (partitions) k on the model utility of different unlearning methods. Our Method LENS is superior and more stable across different k compared to baselines.

k reduces the size of each subgraph and thus accelerates the retraining process for unlearning, it simultaneously increases the degree of structural fragmentation within the global graph.

Experimental Setup. We evaluate the Macro F1 score of LENS and the primary baselines across a range of shard counts $k \in \{5, 10, 15, 20\}$. This analysis is performed on two datasets: Cora and CiteSeer (both using GAT). The purpose of this study is to assess not only the absolute performance of the partitioning schemes but also their robustness as the sharding becomes more aggressive.

Utility-Sharding Trade-off. As illustrated in Figure 3, all methods exhibit a general downward trend in model utility as k increases. This is expected, as a higher k results in more inter-shard edge cuts, thereby depriving the GNN submodels of critical message-passing pathways. However, the final utility at $k = 20$ (reported in our main results) is determined by two factors: the initial partitioning quality at low k and the rate of utility decay as k scales.

Robustness and Consistency. Our method LENS distinguishes itself in two key aspects. First, it consistently starts at a higher utility baseline at $k = 5$, indicating that our neighborhood-aware initialization provides a strong structural foundation. Second, and more importantly, LENS decays significantly slower than baselines such as GraphEraser and GUIDE, staying nearly flat on CiteSeer (Figure 3b). While these methods suffer from sharp drops in F1 scores as the graph is fragmented into smaller pieces, LENS maintains high consistency. This robustness suggests that by accounting for the relative neighborhood loss rather than global edge counts, our partitioning method preserves the local structures that GNN backbones rely on for generalization, even under high granularity constraints.

5 CONCLUSION

In this work, we introduced LENS, a novel framework for exact graph unlearning designed to preserve model utility through explicit neighborhood preservation. Our primary methodological contribution, Neighborhood-aware Partitioning, addresses the limitations of traditional graph partitioning objectives by accounting for the heterogeneous impact of edge removal across individual nodes. By minimizing the relative neighborhood loss, LENS ensures that the local structural information critical to the message-passing mechanism of GNNS remains intact within each subgraph. Our empirical evaluation across multiple datasets and GNN backbones demonstrates that LENS consistently achieves superior model utility. Furthermore, our analysis reveals the fragility of learnable aggregation modules under targeted node removals, highlighting that LENS's training-free aggregation scheme effectively avoids severe utility degradation and hidden retraining overhead.

AI USE STATEMENT

In this work, we used generative AI tools for: formulating mathematical claims, designing or providing feedback on research methodology or experiments, and implementing methods. We have not used generative AI tools for: helping develop theoretical models or conceptual frameworks, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, cleaning and reformating dataset, interpreting results and generating synthetic data sets, assisting with translation, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for: creating or editing software code, summarizing or analysing existing literature, brainstorming, sourcing/searching for information, editing a research paper to improve readability, identifying relevant literature, and proposing a title or keywords for a research paper. We have reviewed all AI-assisted work. Specifically, LLMs assisted with literature discovery, minor methodological refinements, and code generation; however, all literature was manually reviewed, code was thoroughly verified and tested, and the primary research vision and methodology are entirely our own. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we make our code available at: `https://anonymous.4open.science/r/0B12/`. All experiments are conducted using publicly available datasets.

REFERENCES

Sergio A Balanya, Juan Maronas, and Daniel Ramos. Adaptive temperature scaling for robust calibration of deep neural networks. *Neural Computing and Applications*, 36(14):8073–8095, 2024.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pp. 141–159. IEEE, 2021.

Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004(01), 2003.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Chandra Chekuri and Alina Ene. Approximation algorithms for submodular multiway partition. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pp. 807–816. IEEE, 2011.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pp. 499–513, 2022.

Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marinka Zitnik. GNNDelete: A general strategy for unlearning in graph neural networks. *arXiv preprint arXiv:2302.13406*, 2023.

Eli Chien, Chao Pan, and Olgica Milenkovic. Certified graph unlearning. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*.

Weilin Cong and Mehrdad Mahdavi. Efficiently forgetting what you have learned in graph representation learning via projection. In *International Conference on Artificial Intelligence and Statistics*, pp. 6674–6703. PMLR, 2023.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Yushun Dong, Binchi Zhang, Zhenyu Lei, Na Zou, and Jundong Li. Idea: A flexible framework of certified unlearning for graph neural networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 621–630, 2024.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9304–9312, 2020.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Jamie Hayes, Ilia Shumailov, Eleni Triantafillou, Amr Khalifa, and Nicolas Papernot. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 497–519. IEEE, 2025.

Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? *Advances in Neural Information Processing Systems*, 35:13775–13786, 2022.

TN Kipf. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

László Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art: Bonn 1982*, pp. 235–257. Springer, 1983.

Mikko I Malinen and Pasi Fränti. Balanced k-means for clustering. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pp. 32–41. Springer, 2014.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pp. 931–962. PMLR, 2021.

Chao Pan, Eli Chien, and Olgica Milenkovic. Unlearning graph classifiers with limited data resources. In *Proceedings of the ACM Web Conference 2023*, pp. 716–726, 2023.

Yash Sinha, Murari Mandal, and Mohan Kankanhalli. Distill to delete: Unlearning in graph networks with knowledge distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In *31st USENIX Security Symposium (USENIX Security 22)*, pp. 4007–4022, 2022.

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

Cheng-Long Wang, Mengdi Huai, and Di Wang. Inductive graph unlearning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 3205–3222, 2023.

Jiancan Wu, Yi Yang, Yuchun Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. Gif: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, pp. 651–661, 2023.

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

Zhe-Rui Yang, Jindong Han, Chang-Dong Wang, and Hao Liu. Erase then rectify: A training-free parameter editing approach for cost-effective graph unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 13044–13051, 2025.

Lu Yi and Zhewei Wei. Scalable and certifiable graph unlearning: Overcoming the approximation error barrier. In *The Thirteenth International Conference on Learning Representations*.

Jiahao Zhang. Graph unlearning with efficient partial retraining. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 1218–1221, 2024.

Liang Zhao, Hiroshi Nagamochi, and Toshihide Ibaraki. Greedy splitting algorithms for approximating multiway partition problems. *Mathematical Programming*, 102(1):167–183, 2005.

A Appendix

A.1 Additional Details on Related Work

SISA Framework-based Methods. GraphEraser (Chen et al., 2022) is a pioneering work in the domain of graph unlearning. It was adapted from the universal SISA framework (Bourtoule et al., 2021), which consists of three steps: divide the training set into disjoint shards; train isolated submodels on the shards; aggregate the submodels' results. Once a training instance is removed, only the shard containing that instance must be retrained. GraphEraser, GraphRevoker (Zhang, 2024), and GUIDE (Wang et al., 2023) follow this formulation. The first component is a partitioning algorithm that optimizes the trade-off between retraining time and preservation of structural information. More specifically, the generated shards need to be approximately "balanced" without destroying the graph's structure to ensure both retraining efficiency and model utility. "Balanced" shards should have equivalent sizes and label compositions. GraphEraser (Chen et al., 2022) solves this using a modified Label Propagation Algorithm (LPA) to detect communities and greedily assign nodes without violating the balance constraints. On the other hand, GraphRevoker (Zhang, 2024) treats the partitioning process as a learnable function that can be estimated using a multilayer perceptron (MLP). The loss for the MLP is the sum of three terms, which capture the structural loss, the time cost, and the difference between the local and global label distributions. By contrast, GUIDE (Wang et al., 2023) frames the partitioning task as a constrained spectral clustering problem. Besides finding a good graph cut, they suggest a synthesis technique for restoring the missing edges and neighbors of the boundary nodes. The second component of SISA-inspired graph unlearning methods is an aggregation module for combining the submodels' outputs. This module should only require a small sample to be fine-tuned if it needs retraining. The attention mechanism, graph kernels, and contrastive loss are some of the methods that have been tried (Chen et al., 2022; Wang et al., 2023; Zhang, 2024).

Model Parameters Update-based Methods. Another paradigm of graph unlearning seeks to directly manipulate the model's parameters based on gradient analysis, estimation of the revoked data's influence, or heuristics (Pan et al., 2023; Chien et al.; Wu et al., 2023; Cong & Mahdavi, 2023; Cheng et al., 2023; Dong et al., 2024; Yang et al., 2025; Yi & Wei). GIF (Wu et al., 2023) extends the concept of Influence Function (Koh & Liang, 2017) to account for dependencies in graphs where the influence is not confined to the removed element but its encompassing neighborhood. IDEA (Dong et al., 2024) treads the same path of estimating the change in the model's parameters after deletion. Erase then Rectify (ETR) (Yang et al., 2025) assigns importance scores for parameters using the Fisher Information Matrix (FIM) with respect to each deleted element and edits them accordingly. In the rectification phase, the parameters are further updated according to the approximated gradient on the remaining data. Within the same methodological framework, the contribution of Certified Graph Unlearning and ScaleGUN (Chien et al.; Yi & Wei) is their update rules' proven guarantees on the removal of the deleted elements, but their theoretical analysis is restricted to linear GNNs. Along a related line, PROJECTOR (Cong & Mahdavi, 2023) performs unlearning by mapping the model parameters into a subspace that excludes information about the revoked nodes' features. However, its limitation is the narrow applicability to a customized linear GNN. A different tactic of directly modifying the existing model's parameters involves finetuning for an auxiliary loss or heuristic tailored to the goal of unlearning. GNNDelete (Cheng et al., 2023) introduces an additional Deletion operator on top of the standard operators in every layer. When there is an unlearning request, the parameters of the Deletion units are tuned to meet these two properties: *(i) Deleted Edge Consistency:* the model cannot tell whether or not the removed edge actually exists; and (ii) *Neighborhood Influence:* the removal of an edge should not affect the representations of its enclosing subgraph. Along similar lines, Distill to Delete (D2DGN) (Sinha et al., 2025) instead conducts its tuning via Knowledge Distillation. Specifically, the method minimizes the KL divergence between the source model and two auxiliary modules, termed the Preserver and the Destroyer. The Preserver produces embeddings for the retained nodes, whereas the Destroyer outputs embeddings for the deleted nodes under a neutral setting, such as a randomly initialized GNN performing inference on disconnected deleted nodes. While effective at sustaining the original model's capability, the limit of these works from the paradigm of parameter modification is the theoretically and empirically weak assurance of complete removal (Bourtoule et al., 2021; Thudi et al., 2022; Hayes et al., 2025) or their confinement to linear and customized GNNs.

A.2 PROOFS

A.2.1 PROOF OF THEOREM 3.1

Proof. By definition (Chekuri & Ene, 2011), we need to show that: $g(\mathcal{V}_i) = f(\mathcal{V}_i) + \lambda_g |E(S, \bar{S})|$ is non-negative and submodular where $f(\mathcal{V}_i) = \sum_{u \in \mathcal{V}_i} \mathcal{L}_{rel}(u, \mathcal{V}_i)$. Because it is already well-established that the Min-Cut function $|E(S, \bar{S})|$ is non-negative and submodular and that the sum of two submodular functions remains submodular, the only pending part is the proof that $f(\mathcal{V}_i)$ is also non-negative and submodular. Specifically:

$$f(\mathcal{V}_i) = \sum_{u \in \mathcal{V}_i} \mathcal{L}_{rel}(u, \mathcal{V}_i) = \sum_{u \in \mathcal{V}_i} \frac{\left|\mathcal{N}(u) \setminus \mathcal{V}_i\right|}{\left|\mathcal{N}(u)\right| + \epsilon}$$

It is easy to see that:

$$\mathcal{L}_{rel}(u, \mathcal{V}_i) = \frac{\left|\mathcal{N}(u) \setminus \mathcal{V}_i\right|}{\left|\mathcal{N}(u)\right| + \epsilon} \geq 0$$

Hence, $f(\mathcal{V}_i)$ must be non-negative. To verify submodularity, we consider any two sets: $A \subseteq B \subseteq \mathcal{V}$. We must show that:

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \;\; \forall u \in \mathcal{V} \setminus B$$

Let $\mathbf{deg}(u) = |\mathcal{N}(u)|$. We have:

$$\begin{aligned} f(A \cup \{u\}) - f(A) &= \frac{\left|\mathcal{N}(u) \setminus A\right|}{\mathbf{deg}(u) + \epsilon} - \sum_{v \in \mathcal{N}(u) \cap A} \frac{1}{\mathbf{deg}(v)} \\ f(B \cup \{u\}) - f(B) &= \frac{\left|\mathcal{N}(u) \setminus B\right|}{\mathbf{deg}(u) + \epsilon} - \sum_{v \in \mathcal{N}(u) \cap B} \frac{1}{\mathbf{deg}(v)} \end{aligned} \tag{5}$$

First, we show that:

$$\begin{aligned} \left|\mathcal{N}(u) \setminus A\right| &\geq \left|\mathcal{N}(u) \setminus B\right| \\ \left|\mathcal{N}(u) \cap (\mathcal{V} \setminus A)\right| &\geq \left|\mathcal{N}(u) \cap (\mathcal{V} \setminus B)\right| \\ \mathcal{N}(u) \cap (\mathcal{V} \setminus A) &\supseteq \mathcal{N}(u) \cap (\mathcal{V} \setminus B) \\ \mathcal{V} \setminus A &\supseteq \mathcal{V} \setminus B \end{aligned} \tag{6}$$

Because $A \subseteq B \subseteq \mathcal{V}$, it follows that: $\mathcal{V} \setminus A \supseteq \mathcal{V} \setminus B$.
Since $A \subseteq B \subseteq \mathcal{V}$, we also have: $\mathcal{N}(u) \cap A \subseteq \mathcal{N}(u) \cap B$. Hence, it must be true that:

$$\sum_{v \in \mathcal{N}(u) \cap A} \frac{1}{\mathbf{deg}(v)} \leq \sum_{v \in \mathcal{N}(u) \cap B} \frac{1}{\mathbf{deg}(v)} \tag{7}$$

From (5), (6), and (7), we can conclude that:

$$f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B) \;\; \forall u \in \mathcal{V} \setminus B$$

Extension The above logic trivially applies to any set of h-hop neighborhood $\mathcal{N}_h(u)$. This is true even if $\mathcal{N}_h(u)$ is a multiset, in which case the set operations must be manipulated to handle the duplicated neighbors. Therefore, a more general objective to preserve h-hop neighborhoods can be:

$$\min_{\mathcal{V}_1, \ldots, \mathcal{V}_k} \sum_{i=1}^{k} \left(\sum_{j=1}^{h} \frac{1}{j} f^j(\mathcal{V}_i) + \lambda_g |E(\mathcal{V}_i, \bar{\mathcal{V}}_i)| \right)$$

$$f^j(\mathcal{V}_i) = \sum_{u \in \mathcal{V}_i} \frac{\left|\mathcal{N}_j(u) \setminus \mathcal{V}_i\right|}{\left|\mathcal{N}_j(u)\right| + \epsilon}$$

Because each $f^j(\mathcal{V}_i)$ is submodular, their sum is also submodular. In other words, this generalized objective remains an instance of the Submodular k-way Partition problem.

A.3 LENS IMPLEMENTATION

Below, we present the derivation for incrementally computing the subgradient of the Lovász extension of our cutting objective. This enables Algorithm 1 to solve the neighborhood-aware partitioning problem efficiently.

Increment Logic. With $f(\mathcal{V}_i) = \sum_{u \in \mathcal{V}_i} \mathcal{L}_{rel}(u, \mathcal{V}_i)$, we already see that:

$$f(S \cup \{u\}) - f(S) = \frac{\left|\mathcal{N}(u) \setminus S\right|}{\mathbf{deg}(u) + \epsilon} - \sum_{v \in \mathcal{N}(u) \cap S} \frac{1}{\mathbf{deg}(v)}$$

With the global edge-cut component, we have:

$$E(S \cup \{u\}, \bar{S} \setminus \{u\}) - E(S, \bar{S}) = |\mathcal{N}(u) \cap \bar{S}| - |\mathcal{N}(u) \cap S|$$

Combining the two components of function g gives us:

$$\begin{aligned} g(S \cup \{u\}) - g(S) = \frac{\left|\mathcal{N}(u) \setminus S\right|}{\mathbf{deg}(u) + \epsilon} - \sum_{v \in \mathcal{N}(u) \cap S} \frac{1}{\mathbf{deg}(v)} \\ + \lambda_g \left(|\mathcal{N}(u) \cap \bar{S}| - |\mathcal{N}(u) \cap S|\right) \end{aligned}$$

Algorithm 1 Balanced Neighborhood-aware Partitioning

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, shards k, hops h, iters T.
Initialize:
  $\mathbf{H} \leftarrow \text{Conv}^h(\mathcal{G})$
  $P, \{t_i\}_{i=1}^k \leftarrow \text{BisectingKMeans}(\mathbf{H})$ {P is the normalized inverse distance to the centroids}
  $\mathbf{X}^{(0)} \leftarrow \text{Sinkhorn-Knopp}(P + \text{noise})$
Optimization:
for $t = 0$ **to** $T - 1$ **do**
  for $i = 1$ **to** k **do**
    $\partial_i^{(t)} \leftarrow$ Subgradient of $\hat{g}(\mathbf{x}_i^{(t)})$
  end for
  $\mathbf{Y}^{(t+1)} \leftarrow \mathbf{X}^{(t)} - \eta_t \partial^{(t)}$
  $\mathbf{X}^{(t+1)} \leftarrow \text{Project}(\mathbf{Y}^{(t+1)})$ onto constraints in Eq. 3
end for
Output: $\text{LSAP}(X^{T-1})$