Graph Unlearning with Efficient Partial Retraining

B21060202 YuxuanXu
JiahaoZhang
The Hong Kong Polytechnic University
Kowloon, Hong Kong SAR
tony-jiahao.zhang@connect.polyu.hk

Abstract

Graph Neural Networks (GNNs) have achieved remarkable success in various real-world applications. However, GNNs may be trained on undesirable graph data, which can degrade their performance and reliability. To enable trained GNNs to efficiently unlearn unwanted data, a desirable solution is retraining-based graph unlearning, which partitions the training graph into subgraphs and trains sub-models on them, allowing fast unlearning through partial retraining. However, the graph partition process causes information loss in the training graph, resulting in the low model utility of sub-GNN models. In this paper, we propose GraphRevoker, a novel graph unlearning framework that better maintains the model utility of unlearnable GNNs. Specifically, we preserve the graph property with graph property-aware sharding and effectively aggregate the sub-GNN models for prediction with graph contrastive sub-model aggregation. We conduct extensive experiments to demonstrate the superiority of our proposed approach.

1 Introduction

As one of the crucial data representations, graphs are used to describe data with complex relations between objects. Recently, Graph Neural Networks (**GNNs**) have achieved remarkable success in learning from graph data in various real-world applications.

Despite the aforementioned success of GNNs, the concerns about undesirable data posing detrimental effects on GNNs are rising. For instance, *malicious data* [3,7], *low-quality data* [6,9], and *sensitive data* [2,4] are threatening GNNs' safety, prediction performance, and compliance to "the right to be forgotten" (as shown in 1), which necessities the development of **graph unlearning**.

To efffciently remove the effect of undesirable data from trained GNN models, recent years have witnessed two separate lines of research: 1) *approximate unlearning* and 2) *retraining-based unlearning*. Approximate unlearning relies on directly manipulating the model parameters to erase the effects of unwanted data points [5,10], which sacriffces the reliability of the removal, as discussed in previous literatures [1,8]. On the other hand, retraining approaches partition the training graph into small subgraphs and train disjoint sub-models on them, allowing the model owner only to retrain a small sub-model to remove the effect of some bad data. Despite their desirable security properties, these unlearning approaches may destroy the graph structure and label semantic during the partitioning process, degrading the model utility of sub-model GNNs.

In this work, our research objective revolves around 1) preserving the desirable property of retraining-based unlearning and 2) signiffcantly improving the model utility of sub-GNN models in this paradigm. To achieve this goal, we systematically review the limitations of prior works in graph unlearning (as discussed in Section 3), and then introduce a novel graph unlearning framework, namely **GraphRevoker**, equipped with *graph property-aware sharding and graph contrastive sub-model aggregation*. Our contributions can be summarized as follows: **1)** We propose the *graph property-aware sharding* module to preserve the sub-GNN models' prediction performance by keeping the structural and semantic properties in the training graph. **2)** To effectively leverage the disjoint sub-models for prediction, we propose the *graph contrastive sub-model aggregation* module, a lightweight GNN ensemble network, empowered by local-local structural reconstruction and local-global contrastive learning. **3)** Extensive experiments are conducted to illustrate the model utility and unlearning efffciency of the proposed method.

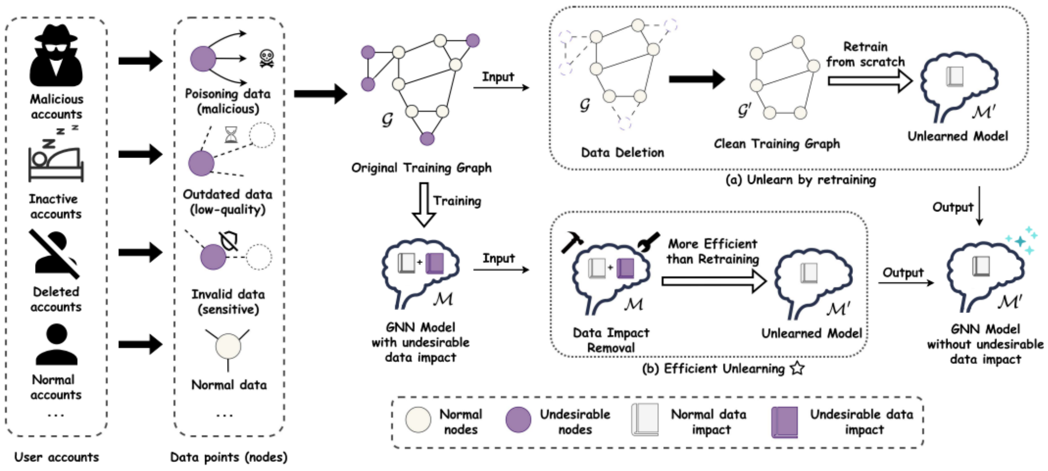

Figure 1: An example of graph unlearning in a social network

2 Problem

Notations. In general, a graph can be denoted as $G = (V, E)$, where $V = \{u_1, \cdots, u_N\}$ denotes the node set, and the edge set E is represented by adjacency matrix A. We define the diagonal degree matrix D as $D_{i,i} = \deg(u_i) = \sum_{j=1}^{N} A_{i,j}$. For node classification, the labels of nodes can be represented as $y = [y_1, \cdots, y_N]$, in which $y_i \in \{\ell_1, \cdots, \ell_C\}$ denotes the C different categories.
Problem Definition. With the concerns on malicious and low-quality graph data, given a trained GNN model F_θ, the goal of graph unlearning is to eliminate the impact of an undesirable subset of training data D^- from F_θ while preserving its model utility. Specifically, in the context of node classification, the undesirable subset with size t can be represented as a set of undesirable nodes $D^- = \{v_{j_1}, \cdots, v_{j_t}\} \subset V$.

Since undesirable knowledge from D^- has been encoded into the parameters of F_θ, retraining - based graph unlearning aims to obtain an unlearned GNN model $F_{\theta'_u}$, which is equivalent to the GNN model $F_{\theta'_r}$ that has never been trained on D^-. This definition of graph unlearning can be formalized as follows:

$$\text{Retraining} : \mathcal{G} \xrightarrow{\text{data removal}} \mathcal{G}/\mathcal{D}^- \xrightarrow{\text{retrain}} \mathcal{F}_{\theta'_r};$$
$$\text{Unlearning} : \mathcal{G} \xrightarrow{\text{train}} \mathcal{F}_\theta \xrightarrow{\text{unlearn}(\mathcal{D}^-)} \mathcal{F}_{\theta'_u},$$

where $F_{\theta'_u}$ and $F_{\theta'_r}$ are expected to be equivalent.

In graph unlearning, we also aim to achieve two goals: 1) Model Utility: After unlearning data points D^-, the latest model $F_{\theta'_u}$ should have comparable performance in comparison with retrained model $F_{\theta'_r}$; 2) Unlearning Efficiency: Compared to retraining from scratch, the unlearning process is expected to be more efficient.

3 Related Work

Approximate Unlearning To avoid the costly retraining process when unlearning undesirable data, approximate approaches directly manipulate the parameters of GNNs, and alleviate the impact of the unwanted data. For instance, GIF [10] develops a novel graph influence function tailored to GNNs and updates the parameters with the gradients from the influence function, while GNNDelete [5] inserts a trainable delete operator between each GNN layer and optimizing the parameters of the delete operator to achieve unlearning. However, recent literature [1,8] finds these methods do not guarantee the accurate removal of the deleted data, which necessitates the existence of retraining in unlearning frameworks.

Table 1: Characteristics of different unlearning frameworks

Aspect / Method	Approximate [5, 10]	SISA [1]	GraphEraser [4]	Ours
Accurate Removal	×	✓	✓	✓
Structural Preservation	✓	×	✓	✓
Semantic Preservation	✓	✓	×	✓
Effective Ensemble	N/A	×	×	✓

Efficient Retraining Regarding both accurate removal and unlearning efficiency, SISA [1] has introduced the retraining-based paradigm, randomly partitioning the training graph into small subgraphs, thereby training sub-models on them. Thus, exact data impact removal can be achieved by only retraining a sub-model. Later on, GraphEraser [4] adapts the SISA [1] paradigm to the context of graphs by introducing a clustering-based partitioning strategy and learnable sub-model aggregation, enhancing the model prediction performance while still allowing efficient and exact unlearning.

Despite the desirable removal guarantee, retraining-based graph unlearning still faces challenges in sub-optimal model utility, due to the graph structural loss in the partition stage. It is true that clustering-based partitions [4] provide a remedy for this issue, but cluster-based partitions tend to put nodes with the same label into one subgraph, sacrificing the semantic balance between different submodels. In addition, the lack of expressive sub-GNN aggregators also deteriorates the model utility problem in retraining approaches.

To this end, we propose GraphRevoker, which achieves efficient and accurate data removal, while preserving both the structural and semantic information in the graph partition phase. We also introduce a powerful contrastive learning empowered aggregation module to ensemble the subgraph GNNs. The advantages of our proposed model can be seen in Table 1.

4 Proposed Approach

To endow GNNs with efficient and exact unlearning capability, GraphRevoker follows the widely adopted SISA framework, which mainly contains three stages: 1) subgraph partition, 2) isolated training, and 3) sub-model aggregation, as shown in Fig. 2. In both partitioning and aggregation stages, we introduce innovative designs to preserve the model utility of the framework. After the aforementioned steps, we can apply GraphRevoker to make predictions by aggregating all the predictions from subgraph GNN models, or efficiently unlearn data points by partially retraining the affected sub-models.

4.1 Graph Property-Aware Sharding

To preserve model utility and unlearning efficiency in the partition phase, we first formulate the unlearning goals in Section 2 into three reachable optimization objectives, and then solve them with an effective neural framework to give desirable graph partitions. In comparison to previous random [1] and clustering-based [4] partition methods, our framework preserves both graph structure and label semantics, resulting in stronger sub-GNN models.

Unlearning Time. The efficiency of retraining-based unlearning mainly depends on the time cost of retraining corresponding sub-models, which relies on two key factors: 1) the probability of retraining a specific sub-model and 2) the time cost of retraining that sub-model. Clearly, the first factor relates to the number of nodes in a subgraph, and the second factor lies behind the cost of message-passing, which is proportional to the number of edges [11]. Let S denote the number of partitioned subgraphs, and $P = \{V_1, \cdots, V_S\}$ denotes node partition results, satisfying $\forall V_i \in V$ and $\forall i \neq j, V_i \cap V_j = \varnothing$. We propose the unlearning time objective as the expectation of retraining time as follows:

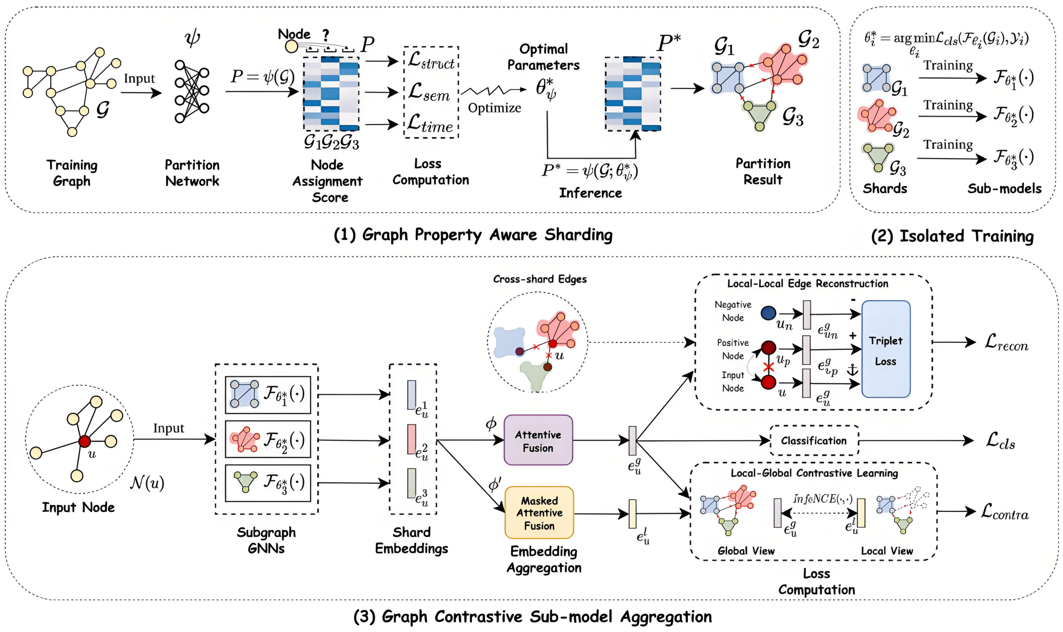

Figure 2: Illustration of the proposed framework

$$\mathcal{L}_{time} = \sum_{i=1}^{S} \Pr(\mathcal{V}_i) \cdot \mathrm{Cost}(\mathcal{V}_i) \approx \sum_{i=1}^{S} \frac{|\mathcal{V}_i|}{|\mathcal{V}|}|\mathcal{E}_i|. \tag{1}$$

Graph Structure Preservation. Edges that connect nodes in different subgraphs are inevitably removed during the partition phase. However, to maintain the model utility of sub-GNNs, preserving the original graph's structure is crucial. Prior works in graph unlearning [2,4] achieve structural preservation by using balanced K-Means [4] or balanced Label Propagation [2] in an unsupervised setting, which cannot directly protect the graph structure. To address this issue, we propose a principled and supervised objective to quantize the structural loss in the graph partitions, namely the normalized edge-cut ($Ncut$) objective, counting the number of dropped edges between different subgraphs as follows:

$$\mathcal{L}_{struct} = Ncut(\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_S) = \sum_{k=1}^{S} \frac{\mathrm{cut}(\mathcal{G}_k, \overline{\mathcal{G}}_k)}{\sum_{u_i \in \mathcal{G}_k} \deg(u_i)}, \tag{2}$$

where $\mathrm{cut}(G_k, \bar{G}_k)$ denotes the edges between subgraph G_k and the remaining part of the training graph.

Label Semantic Preservation. Besides the structural destruction in the partition phase, the label distribution of the training graph is also perturbed, which may result in biased and less generalizable subgraph models. This phenomenon is more severe in clustering-based partitions [2, 4], which tend to assign nodes with similar labels into one subgraph. To overcome this problem, we propose an objective to capture the richness of label semantics in each subgraph with the entropy of its label distribution.

Let $c_{\ell_j}(V_i) = |\{u_k | u_k \in V_i, y_k = \ell_j\}|$ denote the number of nodes annotated with label ℓ_j in the i - th shard. We have the discrete label distribution $d(V_i)$ for the i - th subgraph, where $\Pr(\ell_i) = \frac{c_{\ell_i}(V_i)}{|V_i|}$. Therefore, we propose the entropy-based semantic preservation objective as follows:

$$\mathcal{L}_{sem} = \frac{1}{S}\sum_{i=1}^{S} \mathrm{Entropy}[d(\mathcal{V}_i)] = \frac{1}{S}\sum_{i=1}^{S}\sum_{j=1}^{C} -\log\left[\frac{c_{\ell_j}(\mathcal{V}_i)}{|\mathcal{V}_i|}\right]\frac{c_{\ell_j}(\mathcal{V}_i)}{|\mathcal{V}_i|} \tag{3}$$

Differentiable Graph Partition Framework. To partition the training graph G into disjoint subgraphs while minimizing objectives Eqs. (1) - (3), the main challenge lies behind optimization. As graph partition is a combinatorial optimization problem and cannot be solved in polynomial time, we relax the graph partition into a continuous form and solve it with a neural network.

To make graph partitioning continuous, we represent the partition result with a soft node assignment matrix $P \in \mathbb{R}^{N \times S}$, where $P_{i,j} \in [0,1]$ and $\sum P_{i,:} = 1$. The node assignment matrix can be computed with a graph partition network ψ parameterized with θ_ψ, which contains GNN layers and a softmax output layer, transforming the node representations into the assignment matrix. Therefore, we can learn how to give effective graph partitions by training network ψ with the following loss function:

$$\mathcal{L}_{part} = \mathcal{L}_{time} + \mathcal{L}_{struct} + \mathcal{L}_{sem} + \frac{1}{2}\gamma\|\theta_\psi\|_2^2, \tag{4}$$

$$\text{s.t.} \quad \mathcal{L}_{time} = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{S}(1^T P_{i,i}) \sum_{\text{reduce}} [(P_{:,i}P_{:,i}^T) \odot A], \tag{5}$$

$$\mathcal{L}_{struct} = \sum_{\text{reduce}} [P \cdot \text{diag}^{-1}(1^T DP)](1 - P^T) \odot A, \tag{6}$$

$$\mathcal{L}_{sem} = \frac{1}{S}\sum_{i=1}^{S}\sum_{j=1}^{C} -\log\left(\frac{P_{:,i}^T Y_{:,j}}{P_{i,i}^T 1}\right)\frac{P_{:,i}^T Y_{i,j}}{P_{i,i}^T 1}, \tag{7}$$

where 1 denotes 1-valued column vector, and $Y \in \{0,1\}^{|V| \times C}$ denotes the one-hot label matrix. After training ψ on the training graph with L_{part}, we can use this network to infer desirable partitions. Please refer to Section A.3 in our supplementary material for further explanations of the partition loss function.

4.2 Graph Contrastive Sub-model Aggregation

After acquiring the subgraph partition $P = \{V_1, \cdots, V_S\}$, we can train S isolated submodels $F_{\theta_1}, \cdots, F_{\theta_S}$ on these disjoint subgraphs. Thus, unlearning can be efficiently achieved by only retraining a single sub-model affected by unwanted data. However, unlearning efficiency is not the only goal of retraining - based unlearning, and another challenge lies behind the difficulty of leveraging the weak sub-models to obtain an accurate prediction. Though previous works have explored some straightforward solutions, including mean averaging [1], weighted averaging [4], and attention mechanism [2], it is still an open question on how to utilize the graph structures in the sub-model aggregation phase.

In this work, we develop a contrastive learning framework to learn an effective aggregator to ensemble the weak sub-GNN models. The aggregator only contains a few parameters and can be trained efficiently with only a small subset of training nodes, namely $U = \{u_{j_1}, \cdots, u_{j_M}\} \subset V$.
Attentive Fusion When making predictions with the trained submodels, every sub - model $F_{\theta_i}(\cdot)$ generates a node embedding matrix $E^i \in \mathbb{R}^{|U| \times d}$, where row vector e_u^i represents the embedding of node $u \in U$. Towards a better global prediction, we first align embeddings from different feature spaces with a learnable linear projection, and then fuse all the embeddings with an attention mechanism. Let $\bar{E} \in \mathbb{R}^{|V| \times d}$ denote the fused embedding matrix, and $\bar{e}_u$ denote a row vector in $\overline{E}$, we define the attentive fusion as follows:

$$\alpha_u^i = \frac{\exp(w^T \text{ReLU}(W_i e_u^i + b_i))}{\sum_{j=1}^{S}\exp(w^T \text{ReLU}(W_j e_u^j + b_j))}; \quad \bar{e}_u = \frac{1}{S}\sum_{i=1}^{S}\alpha_u^i e_u^i, \tag{8}$$

where $\{W_1, \cdots, W_S\}$, $\{b_1, \cdots, b_S\}$ and w are trainable parameters, and α_u^i is the attention score for sub-model i and node u.

Local-global Contrastive Loss. In unlearning frameworks based on partial retraining, the output from sub - models contains local knowledge of subgraphs in $\mathcal{G}$, while the aggregation result is expected to incorporate a full knowledge of $\mathcal{G}$. Inspired by previous Graph Contrastive Learning (GCL) approaches [12], we find it natural to leverage the local views of a node to enhance its global aggregation result $\tilde{\boldsymbol{e}}^u$.

Specifically, we regard the fully aggregated embedding $\tilde{\boldsymbol{e}}_u$ as a global view of node u, while we generate a local view $\hat{\boldsymbol{e}}_u$ by randomly deactivating some sub - model attention scores in Eq. (8) (i.e., $\hat{\boldsymbol{e}}_u = \frac{S}{\|\boldsymbol{m}\|_1} \sum_{i=1}^{S} \boldsymbol{m}_i \alpha_u^i \boldsymbol{e}_u^i$, where $\boldsymbol{m}$ is a random 0 - 1 mask). Thus, two different views of the same node $(\tilde{\boldsymbol{e}}_u, \hat{\boldsymbol{e}}_u)$ are expected to be close, and the representation from another random node $v \neq u$ should be pulled apart. The local - global contrastive loss is formulated with the InfoNCE objective as follows:

$$\mathcal{L}_{contra} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathcal{L}_{InfoNCE}(\tilde{\boldsymbol{e}}_u, \hat{\boldsymbol{e}}_u) \tag{9}$$

$$\mathcal{L}_{InfoNCE}(\tilde{\boldsymbol{e}}_u, \hat{\boldsymbol{e}}_u) = -\log \frac{e^{\phi(\tilde{\boldsymbol{e}}_u, \hat{\boldsymbol{e}}_u)/\tau} + e^{\phi(\tilde{\boldsymbol{e}}_u, \tilde{\boldsymbol{e}}_v)/\tau} + e^{\phi(\hat{\boldsymbol{e}}_u, \tilde{\boldsymbol{e}}_v)/\tau}}{e^{\phi(\tilde{\boldsymbol{e}}_u, \hat{\boldsymbol{e}}_u)/\tau}} \tag{10}$$

where $\phi(\cdot, \cdot)$ denotes the cosine similarity function, and τ denotes the softmax temperature. This formulation not only considers both inter-view negative pairs $(\tilde{\boldsymbol{e}}_u, \tilde{\boldsymbol{e}}_v)$ and intra - view negative pairs $(\hat{\boldsymbol{e}}_u, \tilde{\boldsymbol{e}}_v)$, but also requires zero external data augmentation, resulting in expressive representations and efficient computations.

Local-local Reconstruction Loss. In graph partitioning and isolated training, the links between subgraphs are dropped to ensure each node's impact only exists in one sub-model, which allows unlearning by only retraining one specific sub-model. Nevertheless, the dropped edges could include useful structural information of training graph $\mathcal{G}$. To address this problem, we propose the local-local reconstruction loss to restore the knowledge of the previously ignored edges as follows:

$$\mathcal{L}_{recon} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \max\{\phi(\tilde{\boldsymbol{e}}_u, \tilde{\boldsymbol{e}}_{v^+}) - \phi(\tilde{\boldsymbol{e}}_u, \tilde{\boldsymbol{e}}_{v^-}) + 1, 0\} \tag{11}$$

where the positive sample $\tilde{\boldsymbol{e}}_{v^+}$ is sampled from the neighbors of u in other subgraphs, and the negative sample $\tilde{\boldsymbol{e}}_{v^-}$ does not have any connection with u.

Optimization. Our aggregation module learns to fuse sub - GNN models by minimizing the following training objective:

$$\mathcal{L}_{aggr} = \mathcal{L}_{cls} + \mathcal{L}_{contra} + \mathcal{L}_{recon} + \frac{1}{2}\gamma' \|\boldsymbol{\Theta}\|_2^2 \tag{12}$$

where $\boldsymbol{\Theta}$ denotes the trainable parameters of the linear projection and attention mechanism. After unlearning each data point, the aggregation module must be retrained to ensure an accurate removal. Fortunately, we find this module needs 10 to 20 epochs to be trained, only introducing a small computational overhead.

5 Methodology and Results

Datasets. We evaluate GraphRevoker on four real-world graph datasets, including Cora, Citeseer, LastFM-Asia, and Flickr. For the ffrst three datasets, we randomly divide nodes into train/val/test sets with a 0.7/0.2/0.1 ratio. For Flickr, we use the pre-deffned dataset splits in PyG. More implementation details and additional experiments are presented in our supplementary material.

Model Performance. We have evaluated all the unlearning frameworks on the inductive node classiffcation task. The model utility and unlearning efffciency results can be found in Tables. 2 - 3, and we draw the following conclusions: 1) GraphRevoker witnessed the state-of-the-art model performance among all the three efffcient unlearning frameworks in both model

Table 2: The comparison results of model utility in F1-score

Datasets	**Methods**	**GAT**	**GCN**	**SAGE**	**APPNP**	**JKNet**
Cora	Retrain	$\underline{83.95 \pm 0.67}$	$\underline{84.46 \pm 0.51}$	$\underline{82.84 \pm 0.73}$	$\underline{83.66 \pm 0.71}$	$\underline{84.52 \pm 0.54}$
	SISA	67.47 ± 4.43	55.02 ± 1.06	58.82 ± 4.87	66.64 ± 4.61	68.21 ± 4.30
	GraphEraser	75.70 ± 0.33	71.27 ± 2.99	72.10 ± 0.58	68.84 ± 2.56	69.94 ± 3.12
	GraphRevoker	$\mathbf{76.94 \pm 1.17}$	$\mathbf{78.75 \pm 0.47}$	$\mathbf{75.94 \pm 1.39}$	$\mathbf{72.95 \pm 0.61}$	$\mathbf{70.90 \pm 1.79}$
Citeseer	Retrain	$\underline{73.37 \pm 0.83}$	$\underline{72.94 \pm 0.63}$	$\underline{73.17 \pm 0.75}$	$\underline{71.04 \pm 0.83}$	$\underline{73.24 \pm 1.01}$
	SISA	55.90 ± 5.28	69.01 ± 3.33	63.32 ± 4.72	62.99 ± 5.31	65.32 ± 4.58
	GraphEraser	60.53 ± 5.10	69.59 ± 1.39	67.16 ± 1.85	68.36 ± 3.21	68.71 ± 2.20
	GraphRevoker	$\mathbf{70.27 \pm 0.80}$	$\mathbf{72.79 \pm 1.04}$	$\mathbf{69.35 \pm 1.42}$	$\mathbf{70.09 \pm 1.67}$	$\mathbf{70.62 \pm 1.45}$
LastFM Asia	Retrain	$\underline{85.29 \pm 0.53}$	$\underline{84.40 \pm 0.73}$	$\underline{82.18 \pm 0.34}$	$\underline{83.66 \pm 0.71}$	$\underline{85.36 \pm 0.46}$
	SISA	75.77 ± 0.44	75.46 ± 0.52	74.42 ± 0.61	76.06 ± 0.35	75.67 ± 0.38
	GraphEraser	68.12 ± 2.10	64.77 ± 0.96	64.52 ± 1.91	71.55 ± 0.90	69.87 ± 0.77
	GraphRevoker	$\mathbf{76.43 \pm 0.93}$	$\mathbf{76.60 \pm 0.63}$	$\mathbf{75.14 \pm 0.62}$	$\mathbf{77.56 \pm 0.71}$	$\mathbf{76.20 \pm 0.77}$
Flickr	Retrain	$\underline{49.36 \pm 0.64}$	$\underline{49.93 \pm 0.51}$	$\underline{49.41 \pm 1.39}$	$\underline{48.50 \pm 0.80}$	$\underline{49.61 \pm 1.67}$
	SISA	42.81 ± 0.86	46.01 ± 1.24	46.92 ± 1.24	46.72 ± 1.02	44.38 ± 0.95
	GraphEraser	43.82 ± 1.68	46.42 ± 1.10	47.52 ± 1.62	45.96 ± 1.66	44.32 ± 1.65
	GraphRevoker	$\mathbf{45.19 \pm 1.42}$	$\mathbf{48.35 \pm 0.63}$	$\mathbf{48.09 \pm 0.11}$	$\mathbf{47.36 \pm 0.82}$	$\mathbf{46.16 \pm 1.33}$

Table 3: The comparison results of the time-cost of unlearning the undesirable data points ($\mathcal{D}^-$)

Datasets	Methods	GAT	GCN	SAGE	APPNP	JKNet
Cora	Retrain	30.71	24.93	17.55	26.26	24.42
	SISA	7.79	5.04	5.43	5.74	6.41
	GraphEraser	10.37	7.77	6.40	8.52	8.55
	GraphRevoker	**4.72**	**4.65**	**3.10**	**4.23**	**4.88**
Citeseer	Retrain	36.56	30.73	18.80	29.20	32.10
	SISA	9.52	7.30	4.80	7.57	6.16
	GraphEraser	16.39	10.51	9.89	12.90	12.70
	GraphRevoker	**6.25**	**6.05**	**3.85**	**5.32**	**5.87**
LastFM Asia	Retrain	106.86	88.57	73.96	96.63	100.31
	SISA	31.59	46.33	30.58	55.04	51.81
	GraphEraser	35.51	46.95	36.50	50.18	52.84
	GraphRevoker	**12.64**	**29.49**	**16.46**	**28.57**	**27.99**
Flickr	Retrain	177.11	138.67	108.06	155.04	151.35
	SISA	139.68	110.88	72.29	123.16	115.53
	GraphEraser	152.04	120.76	87.86	130.22	127.81
	GraphRevoker	**57.37**	**53.69**	**22.75**	**41.74**	**55.11**

utility and unlearning efffciency; 2) Despite the strong model utility of Retrain, its efffciency is far worse than efffcient unlearning approaches, which is unacceptable in large-scale settings; 3) The propose GraphRevoker even outperforms SISA in efffciency evaluations, which can be attributed to our retraining-time-aware design in the partition module and our parallelized sub-model retraining implementations.

From the results, we can see that our proposed method outperforms the state-of-the-art methods in both model utility and unlearning efficiency. The higher accuracy on the test set indicates better model utility, and the lower unlearning time indicates higher unlearning efficiency.

6 Conclusion and Future Work

Graph unlearning, which involves removing the impact of undesirable data points from trained GNNs, is signiffcant in various realworld scenarios. In this work, we propose a novel graph unlearning framework, which fully unleashes the potential of retraining-based unlearning in graphs and GNNs to achieve accurate, model utility preserving, and efffcient unlearning. To alleviate the model degradation attributed to the partition and isolated training process, we introduce two main contributions: a neural graph partition network and a graph contrastive sub-model ensemble module. In our future works, we plan to explore the effectiveness of GraphRevoker on more settings and downstream tasks in graph mining.

References

[1] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette - Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In IEEE S&P.

[2] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. 2022. Recommendation unlearning. In WWW.

[3] Jingfan Chen, Wenqi Fan, Guanghui Zhu, Xiangyu Zhao, Chunfeng Yuan, Qing Li, and Yihua Huang. 2022. Knowledge - enhanced black - box attacks for recommendations. In KDD.

[4] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2022. Graph unlearning. In ACM CCS.

[5] Jiali Cheng, George Dasoulas, Huan He, Chirag Agarwal, and Marina Zitnik. 2023. GNNDelete: A General Unlearning Strategy for Graph Neural Networks. In ICLR.

[6] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2022. Graph trend filtering networks for recommendation. In SIGIR.

[7] Wenqi Fan, Han Xu, Weijie Jin, Xiaofan Tang, Suhang Wang, Qing Li, Jiliang Tang, Jianping Wang, and Charu Aggarwal. 2023. Jointly attacking graph neural network and its explanations. In ICDE.

[8] Anvith Thudi, Hengrui Jia, Iulia Shumailov, and Nicolas Papernot. 2022. On the necessity of auditable algorithmic definitions for machine unlearning. In USENIX Security.

[9] Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. 2024. Fast graph condensation with structure - based neural tangent kernel. In WWW.

[10] Jiancan Wu, Yi Yang, Yuchen Qian, Yongduo Sui, Xiang Wang, and Xiangnan He. 2023. GIF: A General Graph Unlearning Strategy via Influence Function. In WWW.

[11] Jiahao Zhang, Rui Xue, Wenqi Fan, Xin Xu, Qing Li, Jiaen Pei, and Xiaorui Liu. 2024. Linear - Time Graph Neural Networks for Scalable Recommendations. In WWW.

[12] Yangqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In WWW.