Inductive Graph Unlearning

Cheng-Long Wang, *King Abdullah University of Science and Technology and SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence;* Mengdi Huai, *Iowa State University;* Di Wang, *King Abdullah University of Science and Technology, Computational Bioscience Research Center, and SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence*

https://www.usenix.org/conference/usenixsecurity23/presentation/wang-cheng-long

This paper is included in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

Inductive Graph Unlearning

Cheng-Long Wang[1,4], Mengdi Huai[2], and Di Wang[1,3,4]

[1]King Abdullah University of Science and Technology
[2]Iowa State University
[3]Computational Bioscience Research Center
[4]SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence

Abstract

As a way to implement the "right to be forgotten" in machine learning, *machine unlearning* aims to completely remove the contributions and information of the samples to be deleted from a trained model without affecting the contributions of other samples. Recently, many frameworks for machine unlearning have been proposed, and most of them focus on image and text data. To extend machine unlearning to graph data, *GraphEraser* has been proposed. However, a critical issue is that *GraphEraser* is specifically designed for the transductive graph setting, where the graph is static and attributes and edges of test nodes are visible during training. It is unsuitable for the inductive setting, where the graph could be dynamic and the test graph information is invisible in advance. Such inductive capability is essential for production machine learning systems with evolving graphs like social media and transaction networks. To fill this gap, we propose the **GU**ided **I**n**D**uctiv**E** Graph Unlearning framework (GUIDE). GUIDE consists of three components: guided graph partitioning with fairness and balance, efficient subgraph repair, and similarity-based aggregation. Empirically, we evaluate our method on several inductive benchmarks and evolving transaction graphs. Generally speaking, GUIDE can be efficiently implemented on the inductive graph learning tasks for its low graph partition cost, no matter on computation or structure information. The code is available here: https://github.com/Happy2Git/GUIDE.

1 Introduction

In various complex real-world applications, we often encounter cases where the data is represented as graphs, such as medical diagnosis [29], social media [45], advertising industry [67], and financial industry [54]. The interactions between neighboring nodes make it promising to learn rich information from graph data. After showing great promise in effectively solving graph-based machine learning tasks such as node classification, link prediction, and graph classification, Graph Neural Networks (GNNs) with their large number of variants [26, 32, 63, 65] have received much attention from the machine learning community. Despite their success, recent deployments of GNNs simultaneously raise privacy concerns when the input graphs contain sensitive information of personal data, such as social networks and biomedical data. Recently, the "right to be forgotten" has been proposed in many regulations to protect users' personal information, such as the European Union's General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) [43, 44, 46, 47]. Broadly speaking, the "right to be forgotten" provides individuals the right to request the deletion of their personal information and the right to opt out of the sale of their personal information.

As a de facto way to implement the "right to be forgotten" in machine learning, machine unlearning allows the model owner to completely remove the trace of the samples to be deleted from a trained model without affecting the contributions of other samples, while its unlearning process requires significantly lower computational cost than retraining from scratch. In recent years, a long list of work on machine unlearning has been proposed, and these methods can be categorized into two classes: model-agnostic unlearning [6, 9, 24, 64] and model-intrinsic unlearning [8, 18, 19, 28, 50, 56]. As one of the most well-known model-agnostic methods, SISA [6] uses data partitioning mechanisms to achieve efficient unlearning without full retraining. Specifically, it first divides the dataset into multiple isolated shards and trains a submodel for each shard. Then it aggregates the predictions of all submodels to obtain the final prediction. Such submodels can limit the influence of each data sample throughout the training process. When there is a data removal request, the model owner only needs to partially retrain the submodel corresponding to the data to be removed. Compared to SISA, many unlearning methods, instead of retraining submodels, aim to obtain a shifted model that satisfies some unlearning criteria by modifying the weights of the existing trained model [56]. While these unlearning methods have computational advantages, they are not as transparent as SISA.

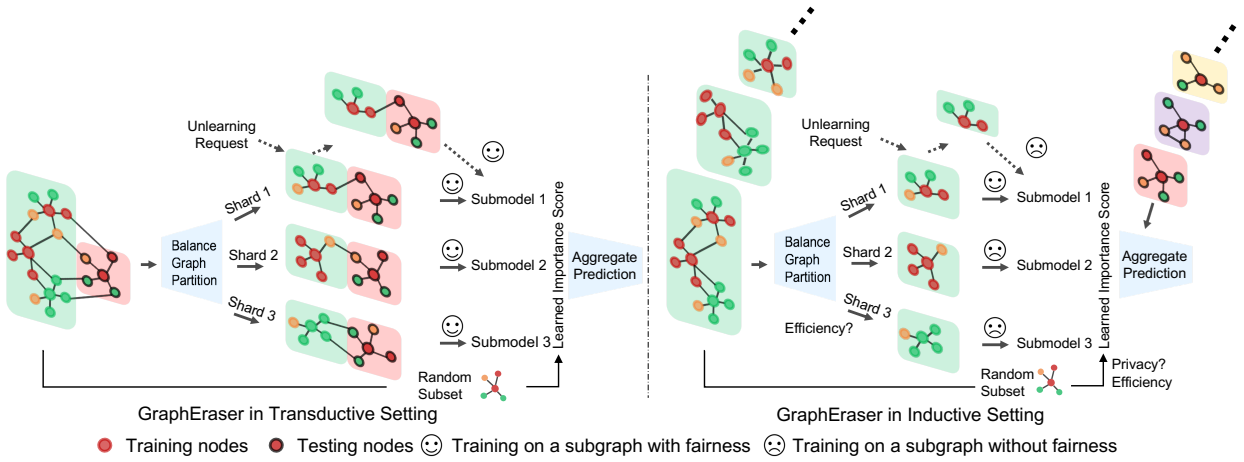

Figure 1: Behavior of GraphEraser in various settings. The colors represent the ground truth labels of the corresponding nodes. In the transductive setting, the features of the test nodes and their connections to other nodes are visible during the training process (in the same static graph with training nodes). The red-shaded subgraphs indicate the test nodes (surrounded by black circles) whose labels are unknown to the model owner in advance. In the inductive setting, training graphs can evolve over time or change incrementally. The test graphs are also completely invisible, resulting in limited information available for training each shard (with a small subgraph) without access to the test nodes and their edges.

Although there are numerous studies on machine unlearning, most are only tailored for image or text data, and unlearning methods for graph data, i.e., graph unlearning, are still lacking. Due to the additional node dependency in graph data, existing unlearning methods cannot be directly applied, which indicates that graph unlearning is more challenging. Based on the SISA framework, [13] proposes the first graph unlearning method, GraphEraser, for graph neural network (GNN) models. Compared with the random partitioning in SISA, GraphEraser provides two balanced partition methods to preserve the additional structural information in graph data. Then, it applies a learning-based aggregation method to obtain the importance scores of submodels. Later, [16] proposes a Certified Graph Unlearning (CGU) method based on the Simplifying Graph Convolutional Network (SGC) [63], which is a linear GCN model. Unfortunately, such a model-specific method is inapplicable to general GNN models.

However, as we will show later, GraphEraser and CGU are inherently designed for the *transductive* graph setting, where the attributes (but not labels) and edges of test nodes are visible during training. They are not designed for the *inductive* graph setting (where test nodes and edges are invisible during training), which is ubiquitous in high-throughput production machine learning systems, as pointed out by [26]. For example, the evolving graphs in a transaction system constantly encounter unseen data samples every day. Thus, the associated fraud detection models should be able to generalize to the newly generated graphs efficiently. Besides transaction systems, such inductive capabilities are also crucial for GNN models in social media, advertising, etc.

For GraphEraser, the time cost of graph partitioning is exceedingly high, so it is not suitable to implement this framework for the evolving graph or multi-graph cases in the inductive setting. Graph unlearning requires that each shard retains a small piece of the training graph to train a submodel. However, the loss of visibility of test nodes and their connections makes submodel training more difficult in the inductive setting. For example, it is easy to learn a weak submodel due to the unfair label composition in each shard, as shown in Figure 1. Note that the *fairness* here refers to group fairness, which ensures some form of statistical parity for members of different protected groups (e.g., gender or race) [3], i.e., the label distribution in each shard remains the same statistic as in the entire training graph. And we use *balance* in the following discussions to represent that the subgraph of each shard has the same size (number of nodes). In addition, GraphEraser aggregates the predictions of submodels on the test nodes by learning important scores for all shards. Once one shard is updated, all other shards need to retrain their important scores, which brings more computational cost and privacy risk.

Thus, we can conclude that the main challenge in model-agnostic inductive graph unlearning is to preserve as much structural information of the original graph as possible while satisfying both fairness and balance constraints in graph partitioning efficiently. This is based on the insights that more structural information leads to higher model performance, a balanced partition ensures that the expected unlearning time cost is small when facing small batch unlearning, and a fair partition would lead to a more robust learning process.

Our contributions: Motivated by our above findings, in this

paper we propose the first inductive graph unlearning framework called GUided InDuctivE Graph Unlearning (GUIDE). Briefly, GUIDE consists of three components: guided graph partitioning with fairness and balance, efficient subgraph repairing, and similarity-based aggregation. Specifically, in guided graph partitioning, we propose two novel graph partitioning methods: GPFB-Fast and GPFB-SR, to obtain a graph partition that efficiently satisfies both fairness and balance constraints. According to our experimental results, the proposed methods are superior to GraphEraser with $\sim 3\times$ balance and fairness scores. GPFB-Fast achieves $\sim 10\times$ speedup on graph partitioning. To the best of our knowledge, this is also the first study on graph partitioning with fairness and balance constraints.

Due to graph partitioning, a lot of edges would be lost, destroying the structure of the original graph. Therefore, to restore this missing information as much as possible, we propose subgraph repair methods as the second component of GUIDE. Through our methods, missing neighbors and their connections with the corresponding nodes could be efficiently generated and added to these subgraphs to repair their structure. Notably, for each shard, our repairing procedures do not involve the information of other shards. After receiving node removal requests, the corresponding repaired subgraphs can be efficiently updated by deleting the corresponding nodes and edges.

As mentioned above, the learning-based aggregation method LBAggr proposed by [13] requires access to the entire training graph when updating the importance scores of the corresponding shards. To speed up the training process, LBAggr is trained on a constructed public subset of the training graph. However, once a shard is updated, all importance scores of other shards need to be updated as well, which introduces additional computational cost. We develop a novel similarity-based aggregation method as our third component to address these issues. Unlike previous methods, our method can compute the importance score of each shard independently, and the normalized similarity score between the partitioned subgraph and the test graph can be directly used as the corresponding importance score. Such independent updating will be more efficient than GrapnEraser when the unlearning batch size is small.

We perform extensive experiments to demonstrate the performance of GUIDE in the inductive setting. GUIDE achieves superior performance ($\sim 3\times$) than the existing state-of-the-art methods on popular node classification benchmarks and the fraud detection task on a real bitcoin dataset. We also introduce two metrics to evaluate the graph partitioning results: balance score and fairness score. Specifically, experimental results show that GUIDE has lower time cost than *GraphEraser* while achieving higher fairness and balance scores in graph partitioning. In addition, we perform extensive ablation studies to demonstrate the utility of other components of GUIDE. Ablation studies show that our proposed subgraph repair methods can significantly improve the performance of GNN models trained on subgraphs. Furthermore, similarity-based aggregation can achieve comparable results to learning-based aggregation.

2 Preliminaries

2.1 Graph Neural Networks

Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges, a basic graph neural network (GNN) model attempts to learn a node representation for downstream tasks from the graph structure and any feature information we have. To train a GNN model, we always use the message passing framework. During each iteration, the GNN model updates the node embedding for each node $u \in \mathcal{V}$ by aggregating the information from u's neighbors $\mathcal{N}(u)$. The k-th update process can be formulated as follows [25]:

$$\begin{aligned} h_u^{(k+1)} &= \text{UPDATE}^{(k)}\left(h_u^{(k)}, \text{AGGR}^{(k)}(h_v^{(k)}, \forall v \in \mathcal{N}(u))\right) \\ &= \text{UPDATE}^{(k)}\left(h_u^{(k)}, m_{\mathcal{N}(u)}^{(k)}\right), \end{aligned}$$

where UPDATE and AGGR are some differentiable functions and $m_{\mathcal{N}(u)}^{(k)}$ is the aggregated 'message' from the neighbors of u. After K iterations of message passing, we can obtain the final embedding for each node. These node embeddings can be used for node classification, graph classification, and relation prediction tasks.

Transductive and Inductive Graph Learning. There are two settings for node classification tasks: the transductive setting and the inductive setting. In the transductive setting, the training nodes and test nodes are in the same static graph. Test nodes and their associated edges are involved in GNN's message passing updates, even though they are unlabeled and not used in the loss computation. In contrast, all test nodes and their edges are completely unobservable during training in the inductive setting. Besides, the training graph can also evolve over time. Compared to the transductive setting, the inductive setting is more common in production machine learning systems that operate on evolving graphs and constantly encounter unseen nodes, such as the daily user-video graphs generated on Youtube [26].

2.2 Transductive Graph Unlearning

Machine Unlearning. Machine unlearning aims to fully eliminate any influence of the data to be deleted from a trained machine learning (ML) model. To implement machine unlearning, the most natural approach is to directly delete all the revoked samples and retrain the ML model from scratch by using the original training data without deleted samples. While retraining from scratch is easy to implement, its computation cost will be prohibitively large to make it efficient when

both the model and the training data are large-scale. Later on, several methods have been proposed to reduce the computation overhead. See the related work section A in Appendix for details.

Graph Unlearning. Graph unlearning refers to machine unlearning for graph data, and in this paper we will focus on GNN learning models. Compared to the standard machine unlearning, there are additional challenges in graph unlearning, e.g. the node dependency in graph data makes most of the existing unlearning methods hard to be applied. To solve this problem, [13] proposes the first graph unlearning framework, GraphEraser.

GraphEraser. Given an undirected graph $\mathcal{G}_F = (\mathcal{V}_F, \mathcal{E}_F)$ whose node set $\mathcal{V}_F$ consists of a training set $\mathcal{V}$ and a test set $\mathcal{V}_T$ (without labels). GraphEraser consists of three phases: (1) balanced graph partition; (2) shard model training; (3) shard model aggregation. Specifically, in step (1), GraphEraser designs two balanced graph partition algorithms (BLPA and BEKM) to get a partition of the training set $\mathcal{V}$. Different from the vanilla methods such as community detection which are easy to output imbalanced partition, BLPA heuristically assigns the nodes with connections to the same group in a manner similar to Lloyd's algorithm for K-Means clustering until the size of the corresponding group arrives at some threshold. BEKM applies a similar method to the embeddings of graph data to achieve better performance. The balanced partition methods could avoid the case that the imbalanced partition contains large shards whose unlearning process is highly inefficient. Suppose the subgraph held by the i-th shard is $\{\mathcal{V}_i \cup \mathcal{V}_T, \mathcal{E}_{i\cup T}\}$, where $\mathcal{E}_{i\cup T}$ is the edge set corresponding to $\mathcal{V}_i \cup \mathcal{V}_T$. Then in step (2) GraphEraser trains a GNN model for each shard in a transductive manner where the unlabeled test nodes and their incident edges are visible to GNN during training. Then these GNN models are tested on the same graph to predict the labels of transductive test nodes. Considering these different shard models do not uniformly contribute to the final prediction, in step (3), GraphEraser applies a learning-based aggregation method (LBAggr) to optimize the importance scores of the shard models to improve the global model utility.

3 Inductive Graph Unlearning

3.1 Problem Definition

Notably, inductive training graphs are different from transductive training graphs. Given an undirected graph $\mathcal{G}_F = (\mathcal{V}_F, \mathcal{E}_F)$ whose node set $\mathcal{V}_F$ consists of a training set $\mathcal{V}$ and a test set $\mathcal{V}_T$, the transductive training graph is $\mathcal{G}_F = (\mathcal{V}_F, \mathcal{E}_F)$ except the labels of $\mathcal{V}_T$, while the inductive training graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ is the edge set corresponding to $\mathcal{V}$. Thus, inductive graph unlearning refers to graph unlearning for inductive training graphs.

Similar to the transductive setting, we have three types of unlearning requests in the inductive setting: node unlearning, feature unlearning, and edge unlearning.

- For a node unlearning request on node $u \in \mathcal{V}$, the service provider needs to retrain the GNN model on the new training graph $\mathcal{G}_u = \mathcal{G} \backslash \{X_u, e_{u,v} | \forall v \in \mathcal{N}_u \cap \mathcal{V}\}$, where X_u represents the attribute of u.
- For a feature unlearning request on node $u \in \mathcal{V}$, the service provider needs to retrain the GNN model on the new training graph $\mathcal{G}_u = \mathcal{G} \backslash \{X_u\}$.
- For an edge unlearning request on edge $e_{u,v} \in \mathcal{E}$, the service provider needs to retrain the GNN model on the new training graph $\mathcal{G}_u = \mathcal{G} \backslash \{e_{u,v}\}$.

According to the SISA framework, the above three types of unlearning lead to the same unlearning update procedure: update the corresponding training subgraph, retrain the GNN model, and compute the importance scores. This paper mainly focuses on the node unlearning task because it is the most difficult one, where feature unlearning and edge unlearning belong to its subsets.

3.2 Challenges

The main challenges in inductive graph unlearning are caused by the more limited available information in each shard compared with transductive graph unlearning. As we pointed out in Figure 1, GraphEraser is unsuitable for the inductive setting. An inductive graph unlearning method should satisfy the following objectives simultaneously to achieve satisfactory performance.

C1: Balanced and Fair Graph Partition. Since there is no help from the test graph during training, we need to make the partition both balanced and fair. A balanced partition makes the retraining time for each shard similar. A fair partition can improve the utility since if several shards are unfair for some classes, their corresponding GNN models would fail to train effective classifiers. Both constraints should be efficiently satisfied for inductive graph learning tasks, where the training graphs may evolve over time or change incrementally.

C2: Comparable Submodel Utility. Unlike the transductive setting, in our problem we have no information on test nodes, which implies that we lost a lot of information from the original graph data for each subgraph after partitioning the entire graph. Thus, to boost the performance of submodels, we need to restore as much information as possible for each subgraph without using other subgraphs' information (due to the unlearning requirement).

C3: Efficient Aggregation Procedure. Existing learning-based aggregation methods need access to a auxiliary dataset (such as the training graph). Once the the nodes in the auxiliary data need to be unlearned, the aggregation model must be retrained. Moreover, the importance score of each shard cannot be calculated independently by those aggregation methods,

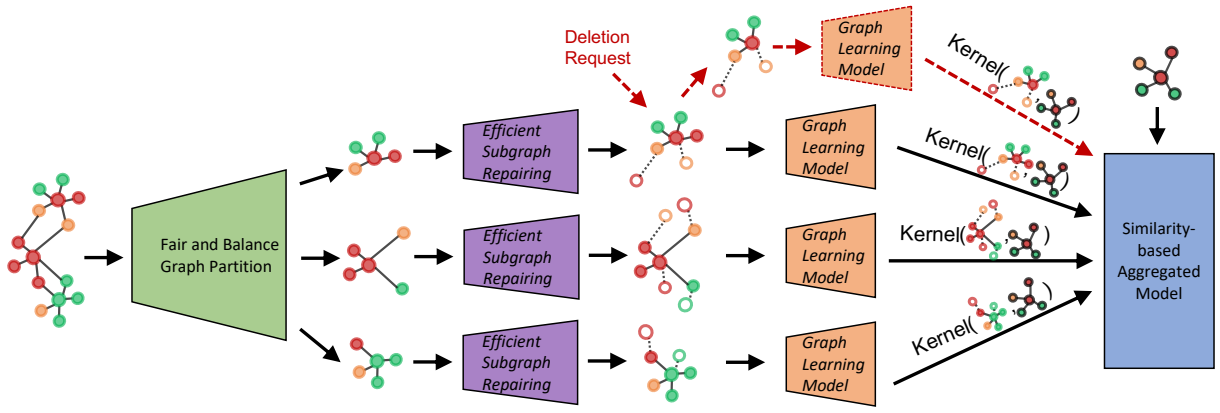

Figure 2: Guided Inductive Graph Unlearning (GUIDE) Framework

which implies that all importance scores should be updated for optimal values if one shard model is updated, i.e., they are quite inefficient. Therefore, we must design new aggregation methods that assign an importance score for each shard independently and do not rely on additional data.

4 GUIDE Framework

4.1 Overview of GUIDE Framework

We propose the Guided Inductive Graph Unlearning (GUIDE) framework to achieve the previous objectives. Generally speaking, GUIDE consists of three components: guided graph partition with fairness and balance, efficient subgraph repairing, and similarity-based aggregation. Figure 2 illustrates the framework of GUIDE.

Guided Graph Partition. To satisfy (C1), we first formulate the problem of finding balanced and fair graph partitions as spectral clustering with linear constraints, which is a quadratic programming problem with binary variables. To solve it efficiently, we relax the constraints and propose the method of GPFB-Fast to solve the relaxed problem. We then present an improved programming problem via spectral rotation and propose GPFB-SR to solve it. To the best of our knowledge, this is the first study on graph partition under fairness and balance constraints.

Efficient Subgraph Repairing. Such a method aims to satisfy (C2). During the partition process, we retain the original degree information of each node independently (note that this step is independent of future changes of other shards). When the partition is completed, we generate missing neighbors for each node independently according to its features and its original degree information. Specifically, we design three strategies: Zero-Feature Neighbor, Mirror-Feature Neighbor, and MixUp Augmented Neighbor, to reduce the side effects of our graph partition.

After repairing all subgraphs, the model owner trains GNN models (in parallel) for all shards isolatedly. The repaired nodes will involve in the GNN message-passing updates. However, the final layer embedding for those repaired nodes will not be used in loss computation.

Similarity-based Aggregation. We develop a similarity-based aggregation method to assign an importance score for each shard independently. The importance score for a shard is calculated by the similarity between its associated subgraph and the test graph. Once a shard is updated, its importance score can be updated efficiently without affecting other shards.

In the following subsections, we will provide details of our three components.

4.2 Guided Graph Partition with Fairness and Balance

In this part, we aim to get a partition satisfying the balance and fairness constraints simultaneously. It is notable that such a task is challenging. On the one hand, while some previous work [10, 13] has proposed some heuristic K-Means clustering variants to achieve balanced graph partitions. Those algorithms are difficult to be extended to graph partitions satisfying two constraints. On the other hand, existing work on fair clustering also does not satisfy the population balance constraint [1, 17, 33]. For further introductions to graph-related clustering, see Appendix B.

Before showing our method, we first show how to incorporate these two constraints into the graph partition problem. Given a graph dataset $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with all node labels, we suppose $|\mathcal{V}| = n$ and $\mathcal{V} = \dot{\cup}_{s\in[h]} \mathcal{C}_s$ where $\mathcal{C}_s$ denotes the node set with label s (and there are h classes). It is obvious to see that the ratio of label s in the whole dataset is $|\mathcal{C}_s|/n$. Motivated by [33], we can first construct a **label-membership indicator matrix** $\mathbf{F} \in \mathbb{R}^{n\times h}$, where $F_{i,s} = 1$ if the label of node i is s and $F_{i,s} = 0$ otherwise. Thus, the sum of entries in the s-th column of $\mathbf{F}$ is $|\mathcal{C}_s|$, the number of nodes with label s. For a given partition $\mathcal{V} = \dot{\cup}_{i\in[v]} \mathcal{V}_i$, we can easily see that the number of nodes with label s in the i-th shard is $|\mathcal{C}_s \cap \mathcal{V}_i|$ and its ratio for the i-th shard is $|\mathcal{C}_s \cap \mathcal{V}_i|/|\mathcal{V}_i|$.

In the most balanced case, the sizes of all shards are the same and the size of the i-th shard would be $|\mathcal{V}_i|^* = n/v$. In the fairest case, the ratio of label s in each shard should be the same as its ratio in the entire dataset, i.e., $(|\mathcal{C}_s \cap \mathcal{V}_i|/|\mathcal{V}_i|)^* = |\mathcal{C}_s|/n$. Then when the graph partition satisfies the fairness and balance constraints at the same time, the number of label s in the i-th shard should be $|\mathcal{C}_s \cap \mathcal{V}_i|^* = \frac{|\mathcal{C}_s|*|\mathcal{V}_i|^*}{n} = \frac{|\mathcal{C}_s|}{v}$.[1]

The following Theorem illustrates how we transform the fairness constraint and balance constraints to a linear constraint on the group-membership indicator matrix $\mathbf{Y} \in \mathbb{R}^{n\times v}$ (see Section B in Appendix for the definitions of group-membership indicator matrix $\mathbf{Y}$ and its normalized version $\mathbf{H}$ for a partition).

Theorem 1 (Transformation of Fairness and Balance Constraints on Indicator Matrix $\mathbf{Y}$)**.** *Based on the previous notations, denote the fairness and balance guided matrix by $\mathbf{M} \in \mathbb{R}^{h\times v}$, i.e., $M_{s,j} = \frac{|\mathcal{C}_s|}{v}$ denotes the optimal size of label s in the j-th shard. For a partition $\mathcal{V} = \cup_{i\in[v]}\mathcal{V}_i$, it is fair and balanced if and only if $\mathbf{F}^\top\mathbf{Y} = \mathbf{M}$, where $\mathbf{Y} \in \{0,1\}^{n\times v}$ is the group-membership indicator matrix of the partition that has the form in (8).*

Based on Theorem 1, it is sufficient for us to find a group-membership indicator matrix $\mathbf{Y}$ such that $\mathbf{F}^\top\mathbf{Y} = \mathbf{M}$. To incorporate into the spectral clustering problem (9), we can leverage the spectral rotation theory by supposing there is an orthogonal matrix $\mathbf{R} \in \mathbb{R}^{v\times v}$ such that $\mathbf{HR} = \mathbf{Y}$ (as illustrated in problem (10)). In total, Theorem 1 suggests that to solve the spectral clustering problem with fairness and balance constraints, it is equivalent to solve

$$\begin{aligned} \min_{\mathbf{Y}\in\mathcal{Y},\mathbf{H},\mathbf{R}} \quad & Tr(\mathbf{H}^\top\mathbf{L}\mathbf{H}) \\ s.t. \quad & \mathbf{H}^\top\mathbf{H} = \mathbf{I}, \mathbf{F}^\top\mathbf{Y} = \mathbf{M}, \mathbf{HR} = \mathbf{Y}, \mathbf{R}^\mathbf{T}\mathbf{R} = \mathbf{I}. \end{aligned} \tag{1}$$

However, problem (1) is a binary quadratic integer programming, which is hard to solve with low computation cost. By introducing a new balanced and fair guided matrix, we design a new linear constraint on the embedding matrix $\mathbf{H}$ in (8) rather than $\mathbf{Y}$.

Theorem 2 (Transformation of Fairness and Balance Constraints on Embedding Matrix $\mathbf{H}$)**.** *Denote the normalized balanced and fair guided matrix by $\widetilde{\mathbf{M}} \in \mathbb{R}^{h\times v}$, i.e., $\widetilde{\mathbf{M}}_{s,j} = \frac{|\mathcal{C}_s|}{\sqrt{nv}}$.*

For a partition $\mathcal{V} = \cup_{i\in[v]}\mathcal{V}_i$, it is fair and balanced if and only if $\mathbf{F}^\top\mathbf{H} = \widetilde{\mathbf{M}}$, where $\mathbf{H}$ is the normalized group-membership indicator matrix of the partition which has the form in (8).[2]

Therefore, the optimization problem of finding a graph partition that satisfies the fairness and balance constraints based on RatioCut is

$$\min_{\mathbf{H}} \quad Tr(\mathbf{H}^\top\mathbf{L}\mathbf{H}) \quad s.t. \quad \mathbf{H} \in \mathcal{H}, \mathbf{F}^\top\mathbf{H} = \widetilde{\mathbf{M}}, \tag{2}$$

where $\mathcal{H}$ is the set of all normalized group-membership indicator matrices. Similar to the standard spectral clustering, we can relax it to

$$\min_{\mathbf{H}} \quad Tr(\mathbf{H}^\top\mathbf{L}\mathbf{H}), \quad s.t. \quad \mathbf{H}^\top\mathbf{H} = \mathbf{I}, \mathbf{F}^\top\mathbf{H} = \widetilde{\mathbf{M}}. \tag{3}$$

Problem (3) is equivalent to the following problem for a large enough α:

$$\min_{\mathbf{H}} \quad Tr(\mathbf{H}^\top\mathbf{L}\mathbf{H}) + \alpha\|\mathbf{F}^\top\mathbf{H} - \widetilde{\mathbf{M}}\|_2^2 \quad s.t. \quad \mathbf{H}^\top\mathbf{H} = \mathbf{I}. \tag{4}$$

Problem (4) can be further written as a quadratic problem over the Stiefel manifold, which can be solved efficiently by the generalized power iteration method [41], i.e.,

$$\begin{aligned} \max_{H} \quad & Tr(\mathbf{H}^\top(\mathbf{W} - \mathbf{D} - \alpha\mathbf{F}\mathbf{F}^\top)\mathbf{H} + 2\alpha\mathbf{H}^\top\mathbf{F}\widetilde{\mathbf{M}}) \\ s.t. \quad & \mathbf{H}^\top\mathbf{H} = \mathbf{I} \end{aligned}. \tag{5}$$

After we solve problem (5) and get the optimal solution $\mathbf{H}^*$, we can apply any K-Means clustering algorithm to its rows to get the final partition of the graph. The optimization method for problem (5), Graph Partition with Fairness and Balance (Fast), is summarized into Algorithm 1 in Appendix C.

As pointed out in [27], the obtained relaxed continuous spectral solution could severely deviate from the optimal discrete solution. Motivated by [15, 60], we add a spectral rotation regularization term to learn better embedding and indicator matrices jointly. In total, we have the following problem.

$$\begin{aligned} \min_{\mathbf{H},\mathbf{Y}} \quad & Tr(\mathbf{H}^\top\mathbf{L}\mathbf{H}) + \alpha\|\mathbf{F}^\top\mathbf{H} - \widetilde{\mathbf{M}}\|_2^2 + \\ & \beta\|\mathbf{HR} - \mathbf{D}^{-\frac{1}{2}}\mathbf{Y}(\mathbf{Y}^\top\mathbf{D}\mathbf{Y})^{-\frac{1}{2}}\|_2^2 \\ s.t. \quad & \mathbf{H}^\top\mathbf{H} = \mathbf{I}, \mathbf{R}^\top\mathbf{R} = \mathbf{I} \end{aligned}. \tag{6}$$

It is notable that as compared with the above problem (5), we can get an indicator matrix directly without using K-Means clustering algorithms by solving problem (6). In Appendix D.1, we show how to solve problem (6) efficiently, and Algorithm 2 in Appendix C is our final method. When the objective function converges or satisfies certain convergence criteria, we can stop the iteration and get the final indicator matrix $\mathbf{Y}$ satisfying fairness and balance constraints.

4.3 Efficient Subgraph Repairing

Subgraph repair has been shown to be helpful in improving the performance of subgraph federated learning [71]. The missing neighbors to be repaired here refer specifically to the 1-hop neighbors of nodes. This is due to the fact that during each training iteration, each node aggregates information from its local (1-hop) neighbors, and as the iterations progress, each

[1] For simplicity, here we assume $\frac{|\mathcal{C}_s|}{v}$ is an integer. It is easy to extend to general cases.

[2] The omitted proof of Theorem 1 and 2 are provided in Appendix D.

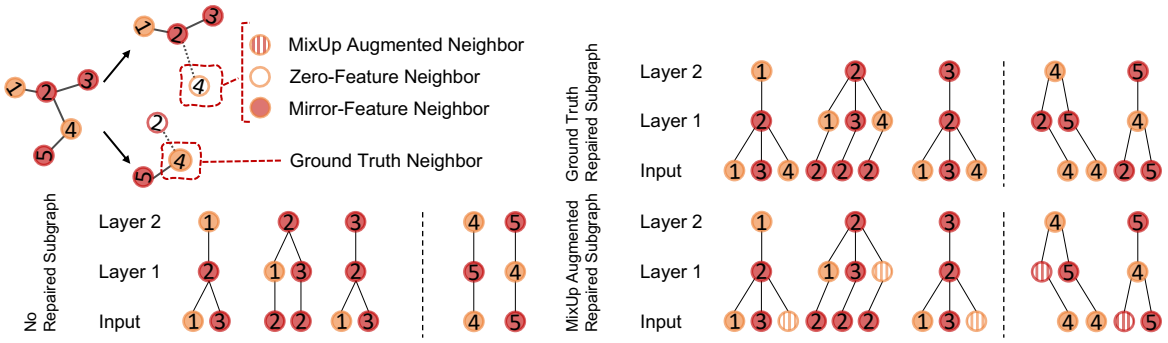

Figure 3: Efficient subgraph repairing. The side effects of partition on the computation graph (message passing operation) of a 2-layer GNN are reduced by the proposed subgraph repairing strategies. The computation graph on a full dataset without partition is provided in Appendix E.

node's embedding contains more and more information from further reaches of the graph [25]. But can these methods really be applied here?

Federated Learning for Missing Neighbors? In a subgraph federated learning system, nodes in each subgraph can potentially have connections with those in other subgraphs. To recover these cross-subgraph missing links, [71] proposes the FedSage+ method to generate the number of missing neighbors and the feature for each missing node. FedSage+ trains a local missing neighbor generation model NeighGen for each local client. The locally computed model gradients of the generative loss are transmitted among the system via the server. Unfortunately, such a federated subgraph repair method involves the training parameters of other clients, which is inapplicable in the setting of graph unlearning.

Local Generator for Missing Neighbors? What if we train a neighborhood-generated model on each subgraph? As mentioned by [71], the federated learning setting is very crucial for the training of NeighGen, which does not hold in graph unlearning. Besides, the additional time cost introduced by NeighGen is very high compared to the training time cost of GNN models. Such complex generative models for subgraph repair cannot be applied in graph unlearning, considering that the primary purpose of graph unlearning is to reduce the retraining time cost.

From the above discussions, we know that an appropriate subgraph repairing method for graph unlearning should satisfy the following properties: (1) It aims to restore the 1-hop neighbors of each node; (2) The repairing procedure should be simple since a complex local generative model will make the unlearning algorithm have high training cost; (3) Due to the unlearning requirement, its repairing procedure for each subgraph cannot rely on other subgraphs' information.

Motivated by the fact that the insight behind many successful node classification approaches is to explicitly exploit *homophily*, we propose to repair the missing nodes based on their preserved neighbors before partitioning. Generally, homophily refers to the tendency of nodes to share attributes with their neighbors [25, 40]. For example, people tend to form friendships with others who share the same interests. For a preserved node i, when homophily exists between its neighbors, we know that its missing neighbors should also have characteristics similar to $\mathbf{x}_i$. When heterogeneity exists, homophily cannot be applied to its neighbors. However, we can still use a simple and effective strategy to repair its local structure. Specifically, we design the following three efficient subgraph repair strategies.

Zero-Feature Neighbor. In this approach, each missing neighbor's attribute of node i will be constructed by

$$\tilde{\mathbf{x}} = \mathbf{0}_{d\times 1},$$

where $\mathbf{0}_{d\times 1}$ is a d-dimentional vector of 0. As an extreme case where homophily does not exist, we construct $\tilde{\mathbf{x}}$ without using any information from node i's feature vector, $\mathbf{x}_i$. As we show in Appendix E, this strategy is sufficient to recover a basic structure of the computation graph.

Mirror-Feature Neighbor. Here each missing neighbor's attribute of node i is constructed by

$$\tilde{\mathbf{x}} = \mathbf{x}_i.$$

As another extreme case of homophily, we directly copy the feature vector of node i as its missing neighbor's feature. Its repaired computation graph is also shown in Appendix E.

MixUp Augmented Neighbor. For a node, the MixUp Augmented Neighbor approach assigns a randomly masked version of the node to its neighbors. In detail, for note i, the attribute of each its missing neighbor is constructed as follows.

$$\tilde{\mathbf{x}} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{0}_{d\times 1},$$

where λ is randomly sampled from the uniform distribution of $[0,1]$ each time for creating diverse neighbors. MixUp Augmented Neighbor strategy could be considered a trade-off between homophily and heterogeneity. Our strategy seems similar to MixUp [70], which has been used as an efficient

data augmentation routine. In short, MixUp extends the training distribution based on the observation that linear interpolations of feature vectors lead to linear interpolations of the associated labels. However, our idea differs from MixUp in that we fix the zero vector in the linear combination and consider only the feature vector, while MixUp requires both features and labels. Directly applying MixUp to repair the missing neighbors of node i requires that node i can provide enough information about the features of its existing neighbors, which is unrealistic after graph partitioning. It is notable that the labels of these newly constructed nodes will not be used during the training process of GNN models. Thus, here we do not need to care what their labels will be.

The effects of MixUp Augmented Neighbor on the computation graph are shown in Figure 3. Such simple methods can recover the structure of the computation graph of the GNN model to some extent and with low computation cost.

4.4 Similarity-based Aggregation

Our aggregation method is motivated by recent developments in the interpretability of GNNs. In particular, several GNN explanation studies have been proposed [38,48,68] and claim that the behavior of GNN models is strongly related to the structure of the training graph. [26] points out that for an inductive GNN model, its generalization to unseen nodes requires "aligning" newly observed subgraphs to the node embeddings on which the algorithm has already been optimized. A new graph with more similar substructures to the training graph is expected to yield better inference results. Thus, we should assign subgraphs which are more similar to the test graph higher importance scores during the inference stage. Here we can directly use graph kernels to measure such similarity. In this paper, we will use the pyramid match graph kernel [42] to compute the similarity score between the test graph and each subgraph, which is a state-of-the-art algorithm for measuring the similarity between unlabeled graphs. [3] Motivated by our ideas above, we propose our similarity-based aggregation method.

Specifically, in our method we first represent each graph as a set of vectors corresponding to the embeddings of its vertices in the eigenspace. To find an approximate correspondence between two sets of vectors, we then map these vectors onto multi-resolution histograms and compare these two histograms through a weighted histogram intersection measure [42]. Given the test graph G_t and the subgraph G_i of shard i (with depth L), denote $H^l_{G_t}$ and $H^l_{G_i}$ as the histogram of G_t and G_i at level l, respectively. We then calculate the pyramid match kernel over these two histograms:

$$k(G_t,G_i) = I(H^L_{G_t},H^L_{G_i}) + \sum_{l=0}^{L-1} \frac{1}{2^{L-l}} (I(H^l_{G_t},H^l_{G_i}) - I(H^{l+1}_{G_t},H^{l+1}_{G_i})), \tag{7}$$

[3] Note that any similarity measuring algorithm can be used here, depending on the settings of different tasks.

where $I(H^l_{G_t},H^l_{G_i})$ is the number of nodes that match at level l in the two sets. We refer the readers to [42] for more details on this kernel. In practice, we can use the *grakel* library [53][4] to implement the pyramid match graph kernel.

4.5 Discussions

Choices of Different Components. We recommend service providers to choose the appropriate partition and subgraph repair methods according to their needs. The choice between GPFB-Fast and GPFB-SR depends on the service provider's preference for graph partitioning: GPFB-SR could lead to a considerably fair and balanced partition, while GPFB-Fast is much faster than GPFB-SR. The choice of subgraph repair strategy depends on the GNN structure we plan to use, as shown in Table 3. Zero-Feature Neighbor is more appropriate for the GraphSAGE model, while Mirror-Feature Neighbor is more appropriate when using the GIN model. The MixUp Augmented method is a general method for all GNNs.

Guarantee of Unlearning. Each component of GUIDE follows the principle of minimizing the use of training graph information. The two proposed graph partitioning algorithms, GPFB-Fast and GPFB-SR, both require only the edge information of nodes with their IDs and labels. The feature information of the nodes is not involved in the graph partitioning step. The subgraph repair procedure uses only the degree information of the entire training graph and the corresponding feature information of each node. The similarity-based aggregation computes the importance score for each shard independently based on the similarity between its corresponding subgraph and the test graph during inference. After receiving an unlearning request, except for the graph partition, both its corresponding shard models and importance scores can be unlearned deterministically. Therefore, similar to SISA [6] and GraphEraser [13], GUIDE is an approximate unlearning approach. To prove the unlearning ability of GUIDE, we perform the membership inference attack on GUIDE in section 5.7 and show our results are close to random guessing. These results are consistent with the conclusion of existing work [6, 13, 14].

Computation Complexity Analysis. For GPFB-Fast, the time cost on initializing $\mathbf{B}$ is $O(nvh)$. In each iteration, the time complexity of updating $\mathbf{P}$ is $O(n^2v+nvh)$, while the time complexity for computing $\mathbf{WH}$ and $\mathbf{FF}^\intercal\mathbf{H}$ is $O(n^2v)$ and $O(nvh)$ respectively. The complexity of calculating reduced SVD on $\mathbf{P}$ is $O(n^2v)$. The computation cost of K-Means is $O(nv^2)$. Suppose the iteration number of updating $\mathbf{H}$ is t_1, then the total computation cost of GPFB-Fast is $O(t_1(n^2v+nvh)+nv^2+nvh)$.

For GPFB-SR, the time cost of solving $\mathbf{R}$ is $O(v^3)$, and the computational complexity for obtaining $\mathbf{Y}$ is $O(nv)$. Therefore, suppose the iteration number of updating $\mathbf{R},\mathbf{H},\mathbf{Y}$ is t_2 and the iteration number for obtaining $\mathbf{Y}$ is t_3, the total time

[4] https://ysig.github.io/GraKeL/

complexity of GPFB-Rotation is $O(t_2(v^3+t_1(n^2v+nvh)+t_3(nv))+nvh)$.

Although the orders of the time complexity of GPFB-Fast and GPFB-SR both are quadratic in n in theory, the main bottleneck is the matrix computation which can be implemented efficiently in parallel. In Section 5.2, we will illustrate that in practice the computation costs of GPFB-Fast and GPFB-SR are less than the computation costs of BLPA and BEKM in [13], which must be performed sequentially by nodes.

5 Experimental Results

We evaluate the performance of GUIDE on the real-world Bitcoin illicit transactions detection task [61] and four popular inductive node classification benchmarks [4, 51, 66].

The evaluation aims to answer the following questions: (1) Unlearning and Implementation Efficiency: How fast can GUIDE handle batch unlearning requests? How efficient are GPFB-Fast and GPFB-SR in practice? (2) Model Utility: Can GUIDE provide state-of-the-art performance for inductive graph learning tasks? (3) Partition Efficacy: Can GPFB-Fast and GPFB-SR output fair and balanced partitions? (4) Efficacy of Subgraph Repairing: Will our subgraph repair strategies help to improve model performance? (5) Efficacy of Similarity-based Aggregation: Can our similarity-based aggregation method reach a level of performance comparable to previous learning-based aggregation methods? (6) Unlearning Ability: Can GUIDE really unlearn the requested nodes?

5.1 Experimental Setup

Datasets and Experimental Setup. The Elliptic Bitcoin Dataset [61] consists of a time series graph (49 distinct time steps, evenly spaced with an interval of about two weeks) of over 200K bitcoin transactions (nodes) and 234K payment flows (edges) with a total value of $6 billion. Twenty-one percent of entities (42,019) are labeled licit (exchanges, wallet providers, miners, licit services, etc.). Two percent (4,545) are labeled illicit (scams, malware, terrorist organizations, ransomware, Ponzi schemes, etc.). The remaining transactions are not labeled with regard to licit versus illicit but have other features. A GNN detection model would learn from past transaction graphs and make a prediction for each entity of the new transaction graph. Similar to the temporal split in [61], which reflects the nature of the task, the first 30 time steps are used to train a GNN model for detecting illicit entities, the next 4 are used for validation, and the last 15 are used for testing. As such, the GNN model is trained in an inductive setting. We set the number of shards for Elliptic to 20, which means that the graph of each time step would be partitioned into 20 subgraphs.

The four popular node classification benchmarks consist of static citation networks and coauthor networks: Cora [66], CiteSeer [66], DBLP [4], and CS [51]. The details of four benchmarks are provided in Appendix F.1. We follow a generally accepted inductive setting in [12, 63]: we construct one graph containing only training nodes and another graph containing all nodes. Graph partitioning and GNN training are applied to the former one. That means the testing nodes are invisible during the training process. Similar to the setting of [13], we set the number of shards for Cora, CiteSeer, DBLP, and CS to 20, 20, 100, and 100, respectively, which makes the number of nodes in each shard similar. For all static graph datasets, we randomly split nodes into 80% and 20% for training and testing and report the average performance of all models over 10 random splits. In fairness to the evaluation, we also report the performance of graph unlearning methods on the transductive setting with the same data splitting and model architecture in Appendix G.5.

Metrics. For the illicit entity detection task, we opt for two commonly used metrics - AUC and Macro F1 score [55]. AUC measures the area under the ROC Curve. Macro F1 score, the mean of the F1-score of both classes without weighting, provides an objective measure of model performance in the face of extreme class imbalance. For inductive node classification benchmarks, we consider classification accuracy as in [12, 63].

To measure the quality of a graph partition, we design two partition metrics: balance score and fairness score. In the following, we provide the definitions of balance score and fairness score for a partition $\{\mathcal{V}_i\}_{i=1}^{v}$ of the graph $(\mathcal{V},\mathcal{E})$ with number of nodes n.

Balance Score: Denote the optimal size of the i-th shard as $|\mathcal{V}_i|^* = \frac{n}{v}$. To quantify the degree of balance, we formally define its population balance score as follows:

$$\mathcal{B}_b = -\frac{1}{2}\Sigma_{i=1}^{v}\frac{||\mathcal{V}_i| - |\mathcal{V}_i|^*|}{n},$$

where $-1 \le \mathcal{B}_b \le 0$. In the optimal case, $||\mathcal{V}_i| - |\mathcal{V}_i|^*| = 0$ for all $i \in [v]$, which implies that $\mathcal{B}_b = 0$. When the partition of the i-th shard is unbalanced, we have $||\mathcal{V}_i| - |\mathcal{V}_i|^*| > 0$, i.e., $\mathcal{B}_b < 0$. We can also easily see that larger $\mathcal{B}_b$ indicates that the partition is more balanced.

Fairness Score: Denote the node set with label s as $\mathcal{C}_s$ for $s \in [h]$, we have $\mathcal{V} = \dot{\cup}_{s\in[h]}\mathcal{C}_s$. It is easy to know that the ratio of nodes with label s in the full dataset is $\frac{|\mathcal{C}_s|}{|\mathcal{V}|}$. Similarly, the ratio of nodes with label s in the i-th shard is $\frac{|\mathcal{C}_s \cap \mathcal{V}_i|}{|\mathcal{V}_i|}$. The fairness score can be computed by

$$\mathcal{B}_f = -\frac{1}{2v}\sum_{i=1}^{v}\sum_{s=1}^{h}\left|\frac{|C_s \cap V_i|}{|V_i|} - \frac{|C_s|}{n}\right|,$$

where $-1 \le \mathcal{B}_f \le 0$. In the fairest case, the ratios for every class over all shards are equal, i.e., $\frac{|\mathcal{C}_s \cap \mathcal{V}_i|}{|\mathcal{V}_i|} = \frac{|\mathcal{C}_s|}{|\mathcal{V}|}$ for all $i \in [v]$, which implies that $\mathcal{B}_f = 0$. When the class s in the i-th shard is unfair, we have $|\frac{|\mathcal{C}_s \cap \mathcal{V}_i|}{|\mathcal{V}_i|} - \frac{|\mathcal{C}_s|}{|\mathcal{V}|}| > 0$ so that $\mathcal{B}_f < 0$. Moreover, we can see a larger B_f indicates the partition is fairer.

Baselines. We compare GUIDE with two standard baselines (Scratch, Random) and two graph unlearning methods (Eraser-BLPA, Eraser-BEKM). For the fraud detection task, we apply graph unlearning methods on a designed illicit entity detection GNN model. For inductive node classification task, we apply graph unlearning methods on 6 popular inductive GNN models to compare their efficiency and model utility, including GraphSAGE [26], GIN [65], GAT [58], GATv2 [7], SuperGAT [31], APPNP [34]. The detailed settings of those baselines and GNN models are reported in Appendix F.2.

For the implementation of GUIDE, we first apply GPFB-Fast or GPFB-SR on the training graph. The partitioned subgraphs are then repaired by our proposed graph repair strategies. After training the GNN model for each shard independently, we compute an importance score for each shard using the similarity-based aggregation. We name the two implementations of GUIDE (with different partition methods) as GUIDE-Fast (GUIDE with GPFB-Fast) and GUIDE-SR (GUIDE with GPFB-SR) for convenience, respectively.

For both GPFB-Fast and GPFB-SR, the regularization parameter α is determined via grid search from $\{0.0001, 0.001, 0.01\}$. For GPFB-SR, the regularization parameter β is determined by grid search from $\{1,2,3,4,5\}$. Unless otherwise indicated, we take the MixUp Augmented Neighbor as our default repairing strategy. The performances of Zero-Feature Neighbor and Mirror-Feature Neighbor are also reported. We use the pyramid match graph kernel to compute the similarity score between each repaired subgraph and the test graph. But we argue that any method of measuring the similarity between graphs can be applied here.

Implementation. All experiments are conducted on a server with 128G memory, two NVIDIA RTX 3090 GPUs with 24GB RAM, and Ubuntu 20.04 LTS OS.

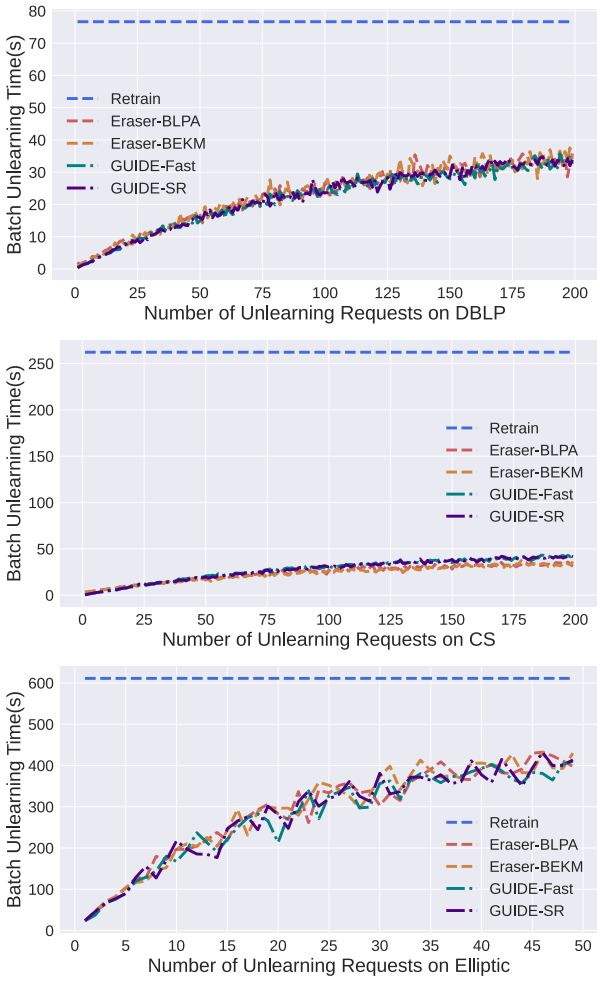

Figure 4: Batch unlearning time on large-scale graph datasets.

5.2 Unlearning and Implementation Efficiency

Batch Unlearning Time. We compare the batch unlearning time of GUIDE and GraphEraser on three graph datasets. The time of Scratch is also reported as the baseline. Our results are shown in Figure 4. We can see that as the number of unlearning nodes increases, more and more shards are involved, so it will take a longer time to unlearn. When all shards need to be updated, the unlearning time tends to be stable. However, since the size of each subgraph is small, it is still faster than retraining from scratch on a large graph. The interesting point is that GUIDE is expected to have a lower unlearning time than GraphEraser due to a more balanced partition and independent importance score updates, but as shown in Appendix G.4, we can only observe such a trend when the batch size of unlearning is small. The reason is that subgraph repair makes each subgraph's size larger than its original size. The actual training time of each submodel may be higher than its training time on a smaller subgraph without repairing. The submodel training time will dominate the unlearning time when the unlearning batch size is large. However, we claim that such a trade-off between unlearning efficiency and model utility is reasonable because the unlearning efficiency degrades slightly in the comparison, while the model utility gets a significant improvement (as we will show later).

Implementation Time. In the inductive setting, the GNN model should learn continuously or keep life-long learning based on those incremental samples. Therefore, implementation efficiency is especially important when facing evolving graphs or multi-graphs. In the following, we report the graph partition time cost for four methods in Table 1.

It is notable that the results in [13] follow a different setting compared to our experiments. [13] sets the number of shards on CS to $k = 30$ and uses a pre-trained GNN model to generate node embeddings for BEKM in the transductive setting. In our setting, the number of shards on CS is $k = 100$. Following the requirements of the inductive setting, we generate node embeddings with the default setting of BEKM, which is time-consuming when the dataset size is large. For the Elliptic dataset, we partition its temporal transaction graphs separately

Table 1: Graph partition time of 4 methods(s).

Dataset	BLPA	BEKM	GPFB-Fast	GPFB-SR
Cora	5.41	10.10	**0.24**	2.85
CiteSeer	6.36	14.56	**0.31**	3.54
CS	38.77	5454.36	**15.71**	40.02
DBLP	37.30	5182.10	**14.44**	33.52
Elliptic	303.02	1089.72	**26.19**	201.99

* The huge increase in BEKM's computation cost comes from its linear relationship with the number of shards and node embedding generation process.

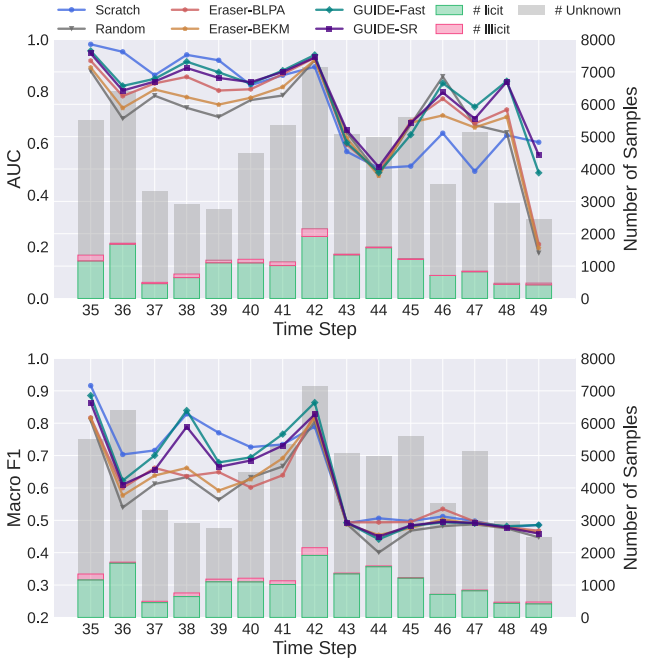

Figure 5: Illicit entity detection results over test time span.

according to their timestamps. As observed in Table 1, GPFB-Fast takes the shortest time for partition. As explained in section 4.2, GPFB-Fast is simple to implement and can be solved efficiently by using standard linear algebra software. For GPFB-SR, we can see it is always faster than BEKM and is comparable with BLPA in some cases. Moreover, it is slower than GPFB-Fast, which is reasonable as it needs more iterations to find a better solution.

5.3 Model Utility

5.3.1 Fraud Detection

We construct a GNN model with three GINconv layers to conduct the illicit entity detection task. Each GINconv layer consists of a 3-layer MLP. After tuning the hyperparameters based on the validation data, we set the size of node embedding to 1024. The model is trained with 200 epochs. The performance of different graph unlearning methods based on our GNN model is shown in Figure 5. It is easy to see that GUIDE performs very close to Scratch for two metrics during the first 8 time steps, and there is a clear performance gap between GUIDE and others. Especially in the 38th time step, GUIDE outperforms other methods with more than 10% Macro F1 score. During the last 7 time steps, all methods provide similar Macro F1 scores due to the very limited illicit samples in the test graph, while GUIDE still produces higher AUC scores than other methods.

5.3.2 Inductive Node Classification

We evaluate the model utility of different graph unlearning methods on 6 commonly used inductive GNN models. Table 2 presents the average results for these methods on four graph datasets. Comparing the results of Scratch and Random, we can find that there is a large gap between them. Most of the time, the node classification accuracy of the Random method is less than half of the Scratch method. Taking this gap as 100%, we can calculate normalized scores of the results given by other graph unlearning methods to quantify the improvement of those methods to the Random method. We can see the improvement of GraphEraser methods to the Random method is only 20%. It is not surprising because the available information is very limited in the inductive setting. Thus, we can see GraphEraser is unsuitable for the inductive setting. However, we also find that the GIN model can achieve the highest node classification accuracy with the help of GraphEraser sometimes (e.g., over the CS data). This is mainly due to the unbalanced partition since the learning-based aggregation assigns a small score to the shards with a small size. But we argue that it is not advisable to sacrifice too much balance for better performance since it will increase the unlearning time cost. Compared to the results of GraphEraser, GUIDE achieves the best performance for almost all models on four datasets. The normalized scores of GUIDE-Fast and GUIDE-SR are both $\sim 2\times$ higher than the results of GraphEraser.

In comparison to the Certified Graph Unlearning method [16], we apply the SGC model to the Scratch method and five graph unlearning methods. The results are provided in Appendix G.1, showing that there is a large gap between the performance of SGC and the performance of state-of-the-art GNNs for inductive graph learning tasks. We also report the results of a 2-layer MLP (Multi-Layer Perceptron) model without considering the graph structure in Appendix G.1.

5.4 Partition Efficacy

We can quantify the partition efficacy of graph unlearning methods by calculating the balance score and fairness score of each partition. The average results on different parts (20%, 40%, 60%, 80%) of four datasets are presented in Figure 6, where the partition score is the summation of the balance

Table 2: Node classification accuracy of 5 graph unlearning methods with 6 inductive GNN models (%).

Dataset	Model	Scratch	Random	Eraser-BLPA	Eraser-BEKM	GUIDE-Fast	GUIDE-SR
Cora	SuperGAT	89.17±0.00	31.57±0.04	41.74±0.16	44.92±0.57	65.69±0.04	**66.49±0.12**
	GATv2	88.94±0.00	31.22±0.04	43.83±0.68	36.62±0.55	66.80±0.08	**68.10±0.16**
	SAGE	92.73±0.00	53.68±0.18	44.20±0.37	53.57±0.60	71.33±0.10	**72.26±0.04**
	GIN	87.07±0.13	56.49±0.26	67.84±0.14	65.55±0.29	76.40±0.05	**77.06±0.06**
	GAT	88.97±0.00	31.90±0.07	38.91±0.36	34.10±0.34	66.25±0.09	**66.40±0.09**
	APPNP	85.96±0.03	51.28±0.13	38.02±0.26	46.38±0.12	64.14±0.07	**64.56±0.05**
CiteSeer	SuperGAT	79.33±0.00	25.44±1.34	53.31±1.15	45.98±0.48	70.66±0.02	**71.17±0.02**
	GATv2	79.53±0.00	25.88±1.45	58.50±0.36	41.04±1.58	70.78±0.02	**71.26±0.02**
	SAGE	83.08±0.00	69.10±0.05	66.90±0.06	69.25±0.05	**72.71±0.02**	72.38±0.01
	GIN	81.20±0.06	58.02±0.41	66.29±0.11	64.21±0.13	69.64±0.07	**69.67±0.04**
	GAT	79.61±0.00	26.32±1.46	58.57±0.64	43.46±1.17	70.66±0.02	**71.02±0.02**
	APPNP	77.49±0.00	72.98±0.02	66.33±0.40	71.29±0.04	73.09±0.03	73.43±0.02
DBLP	SuperGAT	84.21±0.00	44.67±0.00	70.27±0.01	69.84±0.01	**71.67±0.01**	69.29±0.01
	GATv2	83.93±0.00	44.67±0.00	70.23±0.01	69.06±0.05	**71.69±0.01**	69.10±0.00
	SAGE	86.72±0.00	60.38±0.02	70.13±0.00	69.70±0.00	71.92±0.01	**72.16±0.01**
	GIN	87.35±0.01	67.76±0.02	**79.09±0.02**	75.78±0.09	77.11±0.03	77.51±0.00
	GAT	84.05±0.00	44.67±0.00	70.41±0.01	68.51±0.08	**71.39±0.01**	68.70±0.01
	APPNP	83.80±0.00	67.53±0.00	71.56±0.01	70.96±0.01	**73.62±0.01**	72.84±0.01
CS	SuperGAT	87.57±0.00	22.79±0.01	53.01±0.02	41.98±0.25	**69.63±0.00**	69.53±0.01
	GATv2	86.98±0.00	22.79±0.01	53.58±0.04	40.08±0.29	**73.28±0.01**	73.15±0.01
	SAGE	91.79±0.00	71.96±0.02	57.37±0.04	74.38±0.01	**80.68±0.00**	80.67±0.00
	GIN	83.69±0.18	36.70±0.01	75.42±0.15	**83.65±0.01**	79.24±0.01	79.73±0.02
	GAT	87.37±0.00	22.79±0.01	53.24±0.01	43.17±1.04	**69.55±0.01**	69.45±0.01
	APPNP	78.70±0.01	58.03±0.01	48.24±0.10	47.81±0.09	74.38±0.01	**74.44±0.01**
Normalized Score		100.00	0.00	20.42	23.71	59.52	59.40

score and the fairness score. A smaller absolute value of this negative score indicates that the corresponding partition is fairer and more balanced. As we can see from Figure 6, the performance of GPFB-SR is always comparable with the performance of Random. The partition scores of GPFB-SR are $\sim 3\times$ better than the scores of BLPA and BEKM. We also present the distribution of shard sizes in Appendix G.3, which supports this claim. Although the partition scores of GPFB-Fast are worse than those of GPFB-SR, they are almost always better than the results of BLPA and BEKM. The results demonstrate that GUIDE could bring about partitions with balance and fairness, achieving satisfactory performance.

5.5 Efficacy of Subgraph Repairing

To test the efficacy of our proposed subgraph repair strategies, we compare the performance of our three strategies with the ground truth subgraphs and the subgraphs without repairing on the Cora dataset as an ablation study. As shown in Table 3, all three subgraph repairing strategies are helpful in improving model performance. The simplest Zero-Feature Neighbor could achieve a 62.32% improvement. It is not surprising that Mirror-Feature Neighbor behaves worse than Zero-Feature Neighbor since the contributions of the aggregated information from the Mirror-Feature Neighbor are zero. Here we randomly select $\lambda \in [0,1]$ for each missing node to generate the mix-up between the zero and mirror feature. Considering the real application where heterogeneous neighbors may not share the same feature, we can also control this mix-up process by randomly selecting $\lambda \in [0,\tau]$, where $\tau \in [0,1)$ can be decided by testing on a small subset.

5.6 Efficacy of Similarity-based Aggregation

To illustrate the performance of the similarity-based aggregation, we compare it with the average aggregation and the learning-based aggregation methods. We aggregate the predictions of GNN models trained based on the partition of GPFB-Fast and GPFB-SR. For convenience, we denote the two partition methods as 'Fast' and 'SR' respectively in Table 4. Even though we train the LBAggr on the full training graph, its performance is not as stunning as SimiAgg. The reason may be caused by the inductive setting, where the behaviors of those submodels on the training subgraphs may differ from those on the test graph. But still the differences between the three aggregation methods are quite small. It is because the fair and balance graph partition and subgrpah repair have improved each subgraph, leading to an improved submodel in each shard. Thus SimAgg isn't significantly better than the average weighting. But it is still useful to make

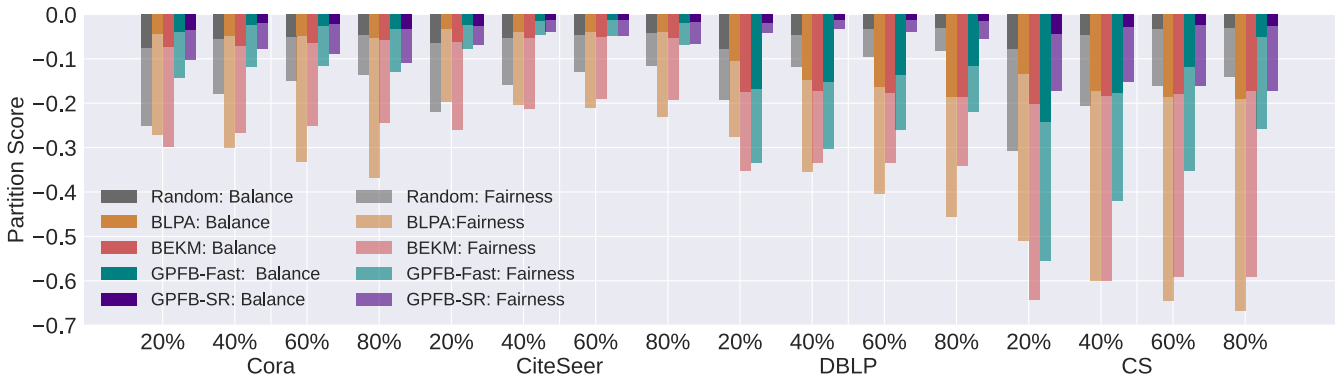

Figure 6: Partition scores of 5 methods on different datasets.

Table 3: Results of different subgraph repairing strategies on Cora(%)

Partition Method	Model	Ground Truth	No Repairing	Mirror Feature	Zero Feature	MixUp
GPFB-Fast	SAGE	77.26±0.04	59.98±0.19	63.22±0.08	73.55±0.05	71.33±0.10
	GIN	79.26±0.04	70.09±0.05	77.13±0.07	72.07±0.06	76.40±0.05
	GAT	70.52±0.10	49.63±0.08	62.90±0.09	66.52±0.07	66.25±0.09
GPFB-SR	SAGE	77.78±0.02	59.98±0.09	65.67±0.08	74.38±0.04	72.26±0.04
	GIN	78.96±0.02	69.28±0.14	75.16±0.09	72.20±0.06	77.06±0.06
	GAT	70.85±0.06	50.00±0.20	65.18±0.09	67.08±0.08	66.40±0.09
Normalized Score		100.00	0.00	54.09	62.32	73.65

Table 4: Results of 3 aggregation methods on Cora(%).

Method		Average	LBAggr	SimiAgg
Fast	SAGE	71.11±0.10	69.02±0.17	**71.33±0.10**
	GIN	**76.40±0.06**	75.25±0.08	**76.40±0.05**
	GAT	66.10±0.09	65.97±0.18	**66.25±0.09**
SR	SAGE	72.07±0.05	70.55±0.16	**72.26±0.04**
	GIN	76.69±0.06	76.32±0.03	**77.06±0.06**
	GAT	66.01±0.09	**67.15±0.20**	66.40±0.09

Table 5: AUC of membership inference attack on GUIDE(%).

Dataset	SAGE	GAT	GIN
Cora	51.34±0.08	49.78±0.02	53.57±0.19
CiteSeer	53.36±0.10	50.97±0.12	50.70±0.08
DBLP	53.34±0.07	51.22±0.19	55.83±0.7
CS	50.34±0.14	51.27±0.14	48.09±0.14

the framework more robust and more explainable.

5.7 Unlearning Ability

Following the same setting as in [14], we evaluate the unlearning ability of GUIDE using the state-of-the-art privacy attack against machine unlearning. We take the aggregated model of GUIDE as the unlearned model after processing 100 random unlearning requests. Using an enhanced membership inference attack [14], the attacker with access to the original model and the unlearned model could determine whether a specific node is indeed removed from the unlearned model. The ratio of member and non-member is set to 1:1. As shown in Table 5, the AUC of membership inference attack on GUIDE is close to 50% (random guess), showing that GUIDE is enough to conduct machine unlearning with low privacy risk.

The study on the sensitivity of GUIDE to the number of shards is provided in Appendix G.2.

6 Conclusions

In this work, we proposed the first general framework, GUIDE, for solving the inductive graph unlearning problem. Generally speaking, GUIDE consists of three components: guided graph partition with fairness and balance, efficient subgraph repairing, and similarity-based aggregation. Due to its exceptional performance compared with the existing methods, we believe this work could serve as a cornerstone for future work on inductive graph unlearning tasks in production machine learning systems.

Although GUIDE offers advantageous performance, it comes with additional memory cost due to its subgraph repair, making each subgraph larger than the original size. Furthermore, a generalization of "partition fairness" to unsupervised graph learning is needed for further applications.

Acknowledgments

Di Wang and Cheng-Long Wang were supported by BAS/1/1689-01-01, URF/1/4663-01-01, FCC/1/1976-49-01, RGC/3/4816-01-01, and REI/1/4811-10-01 of King Abdullah University of Science and Technology (KAUST) and KAUST-SDAIA Center of Excellence in Data Science and Artificial Intelligence.

References

[1] Mohsen Abbasi, Aditya Bhaskara, and Suresh Venkatasubramanian. Fair clustering via equitable group representations. In Madeleine Clare Elish, William Isaac, and Richard S. Zemel, editors, *FAccT '21: 2021 ACM Conference on Fairness, Accountability, and Transparency, Virtual Event / Toronto, Canada, March 3-10, 2021*, pages 504–514. ACM, 2021.

[2] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. Scalable fair clustering. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 405–413. PMLR, 2019.

[3] Reuben Binns. On the apparent conflict between individual and group fairness. In Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna, editors, *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pages 514–524. ACM, 2020.

[4] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[5] Avishek Joey Bose and William L. Hamilton. Compositional fairness constraints for graph embeddings. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 715–724. PMLR, 2019.

[6] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 141–159. IEEE, 2021.

[7] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[8] Jonathan Brophy and Daniel Lowd. Machine unlearning for random forests. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1092–1104. PMLR, 2021.

[9] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 463–480. IEEE Computer Society, 2015.

[10] Chong Chen, Fei Sun, Min Zhang, and Bolin Ding. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2768–2777, New York, NY, USA, 2022. Association for Computing Machinery.

[11] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. Fedgraph: Federated graph learning with intelligent sampling. *IEEE Trans. Parallel Distributed Syst.*, 33(8):1775–1786, 2022.

[12] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[13] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. *CoRR*, abs/2103.14991, 2021.

[14] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 896–911. ACM, 2021.

[15] Xiaojun Chen, Feiping Nie, Joshua Zhexue Huang, and Min Yang. Scalable normalized cut with improved spectral rotation. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1518–1524. ijcai.org, 2017.

[16] Eli Chien, Chao Pan, and Olgica Milenkovic. Certified graph unlearning. *CoRR*, abs/2206.09140, 2022.

[17] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5029–5037, 2017.

[18] Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making AI forget you: Data deletion in machine learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3513–3526, 2019.

[19] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 792–801. Computer Vision Foundation / IEEE, 2021.

[20] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9301–9309. Computer Vision Foundation / IEEE, 2020.

[21] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX*, volume 12374 of *Lecture Notes in Computer Science*, pages 383–398. Springer, 2020.

[22] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 11516–11524. AAAI Press, 2021.

[23] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3832–3842. PMLR, 2020.

[24] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 16319–16330, 2021.

[25] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020.

[26] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[27] Jin Huang, Feiping Nie, and Heng Huang. Spectral rotation versus k-means in spectral clustering. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*. AAAI Press, 2013.

[28] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In Arindam Banerjee and Kenji Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 2008–2016. PMLR, 2021.

[29] Wengong Jin, Jonathan M. Stokes, Richard T. Eastman, Zina Itkin, Alexey V. Zakharov, James J. Collins, Tommi S. Jaakkola, and Regina Barzilay. Deep learning identifies synergistic drug combinations for treating COVID-19. *Proc. Natl. Acad. Sci. USA*, 118(39):e2105070118, 2021.

[30] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. Inform: Individual fairness on graph mining.

In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 379–389. ACM, 2020.

[31] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[32] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[33] Matthäus Kleindessner, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern. Guarantees for spectral clustering with fairness constraints. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3458–3467. PMLR, 2019.

[34] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[35] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 2017.

[36] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. Peer-to-peer federated learning on graphs. *CoRR*, abs/1901.11173, 2019.

[37] Suyun Liu and Luís Nunes Vicente. A stochastic alternating balance k-means algorithm for fair clustering. *CoRR*, abs/2105.14172, 2021.

[38] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[39] Sepideh Mahabadi and Ali Vakilian. Individual fairness for k-clustering. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6586–6596. PMLR, 2020.

[40] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[41] Feiping Nie, Rui Zhang, and Xuelong Li. A generalized power iteration method for solving quadratic problem on the stiefel manifold. *CoRR*, abs/1701.00381, 2017.

[42] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In Satinder Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 2429–2435. AAAI Press, 2017.

[43] Stuart L Pardau. The california consumer privacy act: towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y*, 23:68, 2018.

[44] Eugenia Politou, Alexandra Michota, Efthimios Alepis, Matthias Pocs, and Constantinos Patsakis. Backups and the right to be forgotten in the gdpr: An uneasy relationship. *Computer Law & Security Review*, 34(6):1247–1257, 2018.

[45] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2110–2119. ACM, 2018.

[46] Jeffrey Rosen. The right to be forgotten. *Stan. L. Rev. Online*, 64:88, 2011.

[47] Wenqiang Ruan, Mingxin Xu, Haoyang Jia, Zhenhuan Wu, Lushan Song, and Weili Han. Privacy compliance: Can technology come to the rescue? *IEEE Secur. Priv.*, 19(4):37–43, 2021.

[48] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for NLP with differentiable edge masking. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[49] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[50] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 18075–18086, 2021.

[51] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018.

[52] Takashi Shibata, Go Irie, Daiki Ikami, and Yu Mitsuzumi. Learning with selective forgetting. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 989–996. ijcai.org, 2021.

[53] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

[54] Ivan Sukharev, Valentina Shumovskaia, Kirill Fedyanin, Maxim Panov, and Dmitry Berestnev. EWS-GCN: edge weight-shared graph convolutional network for transactional banking data. In Claudia Plant, Haixun Wang, Alfredo Cuzzocrea, Carlo Zaniolo, and Xindong Wu, editors, *20th IEEE International Conference on Data Mining, ICDM 2020, Sorrento, Italy, November 17-20, 2020*, pages 1268–1273. IEEE, 2020.

[55] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. Rethinking graph neural networks for anomaly detection. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 21076–21089. PMLR, 2022.

[56] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling SGD: understanding factors influencing machine unlearning. In *7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022*, pages 303–319. IEEE, 2022.

[57] Enayat Ullah, Tung Mai, Anup Rao, Ryan A. Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In Mikhail Belkin and Samory Kpotufe, editors, *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pages 4126–4142. PMLR, 2021.

[58] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[59] Ulrike von Luxburg. A tutorial on spectral clustering. *Stat. Comput.*, 17(4):395–416, 2007.

[60] Zhen Wang, Zhaoqing Li, Rong Wang, Feiping Nie, and Xuelong Li. Large graph clustering with simultaneous spectral embedding and discretization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(12):4426–4440, 2021.

[61] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *CoRR*, abs/1908.02591, 2019.

[62] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 13(1):1–10, 2022.

[63] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 2019.

[64] Yinjun Wu, Edgar Dobriban, and Susan B. Davidson. Deltagrad: Rapid retraining of machine learning models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 10355–10366. PMLR, 2020.

[65] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[66] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 40–48. JMLR.org, 2016.

[67] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 974–983. ACM, 2018.

[68] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9240–9251, 2019.

[69] Ye Yuan, Delong Ma, Zhenyu Wen, Zhiwei Zhang, and Guoren Wang. Subgraph matching over graph federation. *Proc. VLDB Endow.*, 15(3):437–450, 2021.

[70] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[71] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu-Ming Yiu. Subgraph federated learning with missing neighbor generation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6671–6682, 2021.

[72] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. Graph-less neural networks: Teaching old mlps new tricks via distillation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.