Community-Centric Graph Unlearning

Yi Li[1,2], **Shichao Zhang**[1,2], **Guixian Zhang**[3], **Debo Cheng**[4*]
[1]Key Lab of Education Blockchain and Intelligent Technology, Ministry of Education,
Guangxi Normal University, Guilin, 541004, China
[2]Guangxi Key Lab of Multi-Source Information Mining and Security, Guangxi Normal University, Guilin, 541004, China
[3]School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu, 221116, China
[4]UniSA STEM, University of South Australia, Mawson Lakes, Adelaide, Australia
liiyi.xsjl@gmail.com, zhangsc@mailbox.gxnu.edu.cn, guixian@cumt.edu.cn, chedy055@mymail.unisa.edu.au

Abstract

Graph unlearning technology has become increasingly important since the advent of the 'right to be forgotten' and the growing concerns about the privacy and security of artificial intelligence. Graph unlearning aims to quickly eliminate the effects of specific data on graph neural networks (GNNs). However, most existing deterministic graph unlearning frameworks follow a balanced partition-submodel training-aggregation paradigm, resulting in a lack of structural information between subgraph neighborhoods and redundant unlearning parameter calculations. To address this issue, we propose a novel Graph Structure Mapping Unlearning paradigm (**GSMU**) and a novel method based on it named Community-centric Graph Eraser (**CGE**). CGE maps community subgraphs to nodes, thereby enabling the reconstruction of a node-level unlearning operation within a reduced mapped graph. CGE makes the exponential reduction of both the amount of training data and the number of unlearning parameters. Extensive experiments conducted on five real-world datasets and three widely used GNN backbones have verified the high performance and efficiency of our CGE method, highlighting its potential in the field of graph unlearning.

Code — https://github.com/liiiyi/CCGU
Extended version — https://arxiv.org/pdf/2408.09705

Introduction

As Artificial Intelligence (AI) permeates various sectors, protecting individual privacy and data security has become an increasingly pressing concern (Wei et al. 2024). Legal frameworks, such as the "right to be forgotten" (Pardau 2018; Cofone 2020), alongside the rise of techniques like membership inference attacks (He et al. 2021) have underscored the need for model providers to swiftly remove specific data and mitigate its impacts. In parallel, Graph Neural Networks (GNNs) (Li et al. 2022; Xu et al. 2018; Li et al. 2024a,b) have gained prominence as a powerful tool, renowned for their ability to model complex data relationships. However, their widespread application in areas involving highly sensitive data (Zhang et al. 2025; Deng et al. 2024), such as social networks(Qiu et al. 2018; Deng et al. 2021) and recommendation systems (Fan et al. 2019; Deng et al. 2022), has intensified the demand for effective privacy-preserving techniques. Among these, ***Graph Unlearning*** (Chen et al. 2022b) has gained particular prominence.

Graph unlearning seeks to precisely quantify and remove the influence of specific data within graph-structured networks while maintaining the integrity of the overall model, thereby addressing critical privacy concerns in an era of expanding AI applications. A straightforward approach to unlearning involves retraining the entire model from scratch. However, this method is highly impractical due to the substantial time costs and associated model downtime. Consequently, current research in unlearning has focused on developing techniques that reduce this time expenditure while maintaining the model's effectiveness (Said et al. 2023).

To achieve efficient and effective unlearning, researchers have developed various strategies that circumvent the costly process of full retraining (Guo et al. 2019; Izzo et al. 2021; Bourtoule et al. 2021; Chen et al. 2022b). These unlearning frameworks are typically divided into approximate and deterministic methods (Nguyen et al. 2022). Approximate methods (Guo et al. 2019; Izzo et al. 2021) involve fine-tuning models to implement unlearning; however, they only provide abstract statistical guarantees of privacy. In contrast, deterministic methods, the focus of this paper, involve retrospective retraining to remove data effects entirely (Yan et al. 2022). For example, the SISA (Bourtoule et al. 2021) approach partitions data for selective retraining. GraphEraser (Chen et al. 2022b), building on SISA, uses balanced partitioning to ensure equal node distribution and aggregates submodels using a learning-based method. Recently introduced, GUIDE (Wang, Huai, and Wang 2023) enhances predictive performance by repairing edges before aggregating partitioned subgraphs.

In summary, the prevailing deterministic graph unlearning frameworks (Chen et al. 2022b; Wang, Huai, and Wang 2023; Chen et al. 2022a) are primarily based on the Balance Partition- Submodel - Training and Aggregation paradigm (**BP-SM-TA**). The principal limitation of this paradigm lies in its assumption that the unlearning operation can be decomposed into a set of discrete sub-problems, each with its own global representativeness. This decomposition, however, sacrifices the connectivity between sub-problems and

*Corresponding author

increases the complexity of managing them. Graph unlearning with BP-SM-TA, two specific challenges arise:

Challenge 1 **(Low Structural Utilization)**: In the BP-SM-TA paradigm, submodels are treated as independent, leading to the disassembly of the original graph during partitioning and resulting in the loss of critical structural information between shards that were originally connected. This, in turn, diminishes the overall comprehension of the graph's structure. Additionally, the balanced partitioning method, which aims to achieve uniform training times for submodels by selecting nodes through clustering, further compromises the graph's structural integrity.

Challenge 2 **(High Model Complexity)**: In the BP-SM-TA paradigm, submodels must independently represent the original graph, meaning each submodel needs to capture the essential features of the original structure. This requirement introduces significant challenges in terms of partitioning and structuring the submodels. During the unlearning process, this paradigm necessitates retraining all affected submodels, which leads to extensive parameter computations. For large graphs, simultaneously loading all submodels for aggregation incurs substantial time costs, which contradicts the primary objective of efficient graph unlearning.

To tackle the aforementioned challenges, we propose a novel Graph Structure Mapping Unlearning paradigm (**GSMU**). Specifically, GSMU generates a mapped graph comprising the non-redundant features and structural information of the original graph. The nodes and edges of the mapped graph are derived from the subgraphs and inter-subgraph node relationships in the original graph. The construction of edges between mapped nodes addresses the limitation of submodel independence identified in *Challenge 1*. Meanwhile, unlearning operations only require updating the affected mapped nodes and edges, thereby avoiding extensive parameter retraining and addressing the limitation of independent submodel representativeness identified in *Challenge 2*. Moreover, we propose a novel community-centric graph unlearning method, Community-centric Graph Eraser (**CGE**), which serves as a practical implementation of GSMU. CGE comprises two components: parameter-free community-centric graph mapping and a node-level unlearning strategy. Specifically, CGE divides the original graph into multiple communities to accomplish the subgraph-node mapping required by the GSMU, and subsequently the unlearning strategy only needs to be performed on the nodes after the mapping, thus controlling the part affected by the unlearning requirement at the node level. Our contributions are summarized as follows:

- We propose a novel graph structure mapping unlearning paradigm, **GSMU**, to overcome the space and time constraints of the traditional BP-SM-TA paradigm in addressing unlearning concerns. To the best of our knowledge, this is the first application of a mapped graph approach in graph unlearning.
- Based on GSMU, we introduce a new efficient graph unlearning framework, **CGE**, which supports deterministic data unlearning while maximizing the retention of the original graph's structural information and semantic feature. By mapping communities to nodes, unlearning operations can be executed at the node level.
- Extensive experiments on five real-world datasets demonstrate that our proposed CGE achieves superior performance and remarkable unlearning efficiency across datasets of varying scales and characteristics.

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Original Graph
$\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$	Mapped Graph
$\mathcal{G} \mapsto \widetilde{\mathcal{G}}$	The Mapping from $\mathcal{G}$ to $\widetilde{\mathcal{G}}$
$\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$	Node Subgraphs Set
$\mathcal{V}_u = \{v_{u1}, v_{u2}, \ldots v_{un}\}$	Unlearning Requests Set
$\mathcal{I}_{\mathcal{V}_u}$	Unlearning Influence Set
$\mathrm{Update}_{\widetilde{\mathcal{G}}}(\widetilde{\mathcal{V}}', \widetilde{\mathcal{E}}', \widetilde{\mathcal{X}}', \widetilde{\mathcal{Y}}')$	The process of updating $\widetilde{\mathcal{G}}$

Table 1: Summary of Notations.

Preliminaries

Notations. In this paper, we use specific notations to represent various components of GNNs and related operations. The tilde notation $\widetilde{\cdot}$, is used to indicate entities produced by mapping operations. Calligraphic fonts are employed to denote sets. We summarize the notations in Table 1.

Deterministic Graph Unlearning. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ represents the set of edges, and a GNN backbone model denoted as $f_\theta(\mathcal{G})$, with θ being the model parameters. Suppose the unlearning request is a sequence of nodes $\mathcal{V}_u \subseteq \mathcal{V}$. The goal of deterministic graph unlearning is to eliminate the influence of these nodes from the model, formalized as:

$$f_{\theta'}(\mathcal{G}') \approx f_\theta(\mathcal{G}) \quad \text{subject to} \quad \mathcal{V}_u \cup \mathcal{G}' = \emptyset, \tag{1}$$

where $\mathcal{G}' = (\mathcal{V} \setminus \mathcal{V}_u, \mathcal{E} \setminus \mathcal{E}_u)$ and θ' denotes the updated model parameters after unlearning. The advanced objectives of graph unlearning should ensure portability, comparable model utility to training from *Scratch* (Chen et al. 2022b), the maintenance of model performance before and after unlearning, high graph structure utilization, and minimal unlearning window periods. Graph unlearning can be broadly divided into node-level and edge-level approaches. In our GSMU paradigm, edge unlearning is considered a subproblem of node unlearning. Consequently, our work focuses on the node classification task, treating node unlearning as the fundamental unit.

Community Detection. Community detection aims to partition a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ into a community set $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, optimizing intra-community connectivity while minimizing inter-community connectivity (Jin et al. 2021; Khan and Niazi 2017). Hierarchical community detection (Lancichinetti et al. 2011) further refines this process by creating multi-level community structures, which clarify relationships at each level. In this paper, we first perform coarse-grained segmentation followed by fine-grained segmentation, as described in the next section.

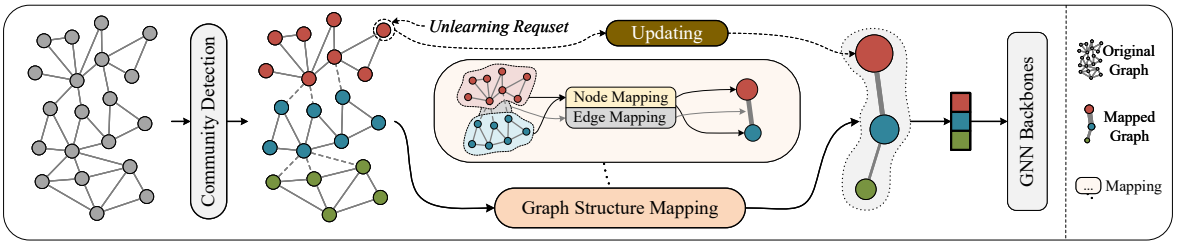


Figure 1: Community-centric Graph Eraser (CGE) framework. CGE maps the communities in original graph $\mathcal{G}$ to obtain graph $\widetilde{\mathcal{G}}$ by graph structure mapping. Only the related mapped nodes need to be updated when nodes require unlearning.

Community-Centric Unlearning

In this section, we detail the GSMU paradigm and the specific implementation of CGE.

GSMU. In contrast to the BP-SM-TA paradigm, GSMU employs a subgraph-based graph structure mapping approach to derive a representative mapped graph, $\widetilde{\mathcal{G}}$, from the original graph, $\mathcal{G}$. Each mapped node in $\widetilde{\mathcal{G}}$ corresponds to a subgraph in $\mathcal{G}$. After partitioning, $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ are isolated to ensure privacy. When it is necessary to unlearn a node, the unlearning operation only affects its corresponding mapped nodes, i.e., the subgraphs in which the node resides.

Definition 1. ***Graph Structure Mapping*** *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of predefined subgraphs $\mathcal{C}$, GSMU maps it to a graph $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$. Each subgraph $C \in \mathcal{C}$ constructs a mapped node, i.e., $\mathcal{V} \mapsto \widetilde{\mathcal{V}} = \{\widetilde{v}_i \mid C_i \in \mathcal{C}\}$. An edge $\widetilde{e}_{ij}$ is created for each edge $e_{ij} \in \mathcal{E}$ that connects nodes in different subgraphs C_i and C_j, i.e., $\mathcal{E} \mapsto \widetilde{\mathcal{E}} = \{\widetilde{e}_{ij} \mid e_{ij} \in \mathcal{E},\ e_{ij}$ connects C_i and $C_j\}$. Node features fusion is performed on each subgraph, i.e., $\mathcal{X} \mapsto \widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}}_i \mid \widetilde{\mathbf{x}}_i = f(\{\mathbf{x}_v \mid v \in \mathcal{V}_{C_i}\})\}$, where f is a fusion function. Node labels are determined by voting within each subgraph, i.e., $\mathcal{Y} \mapsto \widetilde{\mathcal{Y}} = \{\widetilde{y}_i \mid \widetilde{y}_i = g(\{y_v \mid v \in \mathcal{V}_{C_i}\})\}$, where g is a labeling function.* ***The mapping relationship between $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ is defined as:*** $\mathcal{G} \mapsto \widetilde{\mathcal{G}} = \left\{\mathcal{V} \mapsto \widetilde{\mathcal{V}},\ \mathcal{E} \mapsto \widetilde{\mathcal{E}}, \mathcal{X} \mapsto \widetilde{\mathcal{X}},\ \mathcal{Y} \mapsto \widetilde{\mathcal{Y}}\right\}$.

Under the developed GSMU framework, we propose a community-centric graph unlearning method, i.e., Community-Centric Graph Eraser (CGE). The subsequent parts will provide an overview of the operational process of CGE and explain how CGE constructs mapped graph and utilises it for node prediction and unlearning.

Overview of CGE

The CGE framework, illustrated in Figure 1, consists of two stages: the generation of a community-centered mapped graph and a node-level unlearning/training stage. In particular, CGE employs hierarchical community detection to construct a subgraph set $\mathcal{C}$, establish the graph structure mapping in Definition 1, and generate a mapped graph $\widetilde{\mathcal{G}}$ for subsequent operations.

- **Unlearning:** Upon receiving a sequence of unlearning requests $\mathcal{V}_u$, CGE employs the node-level unlearning strategy to determine the specific influence $\mathcal{I}_{\mathcal{V}_u}$ of $\mathcal{V}_u$ on the mapped graph $\widetilde{\mathcal{G}}$ identifying the specific affected mapped edges and nodes. Subsequently, CGE recalculates the nodes and edges within $\mathcal{I}_{\mathcal{V}_u}$ according to the mapping, thereby updating the graph $\widetilde{\mathcal{G}}$.
- **Prediction:** When the node v_p requires prediction, CGE determines destination community C_i based on the community of the adjacent nodes of v_p. The node features of v_p are then fused with the mapped nodes $\widetilde{v}_i$ corresponding to C_i, which in turn updates the graph $\widetilde{\mathcal{G}}$ acting on the GNN. The node $\widetilde{v}_i$ is reanalyzed to map the attributes of the node v_p.

Community-Centric Mapping

CGE generates a community-centered mapped graph $\widetilde{\mathcal{G}}$ and establishes a graph structure mapping $\mathcal{G} \mapsto \widetilde{\mathcal{G}}$, as described in Definition 1. In particular, $\widetilde{\mathcal{G}}$ is utilized solely to retain the representative information of $\mathcal{G}$, thereby enabling its replacement for representation learning. Each community within $\mathcal{G}$ is mapped into a node, and the robustness score between the mapped node pairs is employed to construct the edge relationships.

Community Detection CGE introduces the Louvain (Blondel et al. 2008) to initialize communities and uses the OSLOM method (Lancichinetti et al. 2011) to optimize community structure. Louvain is a community detection algorithm based on modularity optimization. It attempts to move each node into the community of its neighboring nodes to maximize the modularity increment. The modularity is calculated as follows:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \tag{2}$$

where A_{ij} represents the edge weight between node v_i and node v_j. Additionally, k_i and k_j denote the degrees of v_i and v_j, respectively. The term m denotes the total weight of all edges in $\mathcal{G}$. $\delta(c_i, c_j)$ is an indicator variable that equals 1 when nodes i and j belong to the same community, and 0 otherwise.

OSLOM calculates the statistical significance of the initialized communities using the following formula:

$$p(C) = \sum_{t \geq t_C} \binom{T_C}{t} p^t (1-p)^{T_C - t}, \tag{3}$$

where T_C represents the total number of possible connections in community C, t_C represents the actual number of connections in C, and p represents the probability that nodes in C are connected to other nodes. OSLOM then returns an optimized community set $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

Mapping As delineated in Definition 1, the mapping of $\mathcal{G} \mapsto \widetilde{\mathcal{G}}$ comprises four distinct mappings. Among these, we categorize the node, feature, and label mapping as **node mapping**, which align with **edge mapping**.

Node Mapping. The initial step is to construct a node mapping based on the community set $\mathcal{C}$:

$$\mathcal{V} \longmapsto \widetilde{\mathcal{V}} = \{\widetilde{v}_i \mid C_i \in \mathcal{C}\}. \tag{4}$$

To ensure that supervised learning goes well, it is essential to assign the correct labels and features to the mapped nodes. Despite the strict division of communities, node features within a community may vary and have inconsistent labels, making it impractical to average the node labels and features simply (Zhang et al. 2017; Deng et al. 2023). To address this, we calculated the core feature components of the community and designed a feature-based label mapping method. We construct a feature matrix $\mathbf{X}_C \in \mathbb{R}^{n \times d}$, where d is the feature space dimensionality and rows correspond to node feature vectors in the community C. Principal Component Analysis (PCA) is then applied to $\mathbf{X}_C$ to reduce its dimensionality. Let $\mathbf{X}'_C \in \mathbb{R}^{n \times d'}$ be the transformed feature matrix by PCA, where d' is the number of principal components based on the ratio n components. The community feature $\widetilde{\mathbf{f}}_C$ is calculated as the mean of the transformed feature vectors:

$$\widetilde{\mathbf{f}}_C = \frac{1}{|\mathcal{V}_C|} \sum_{v \in \mathcal{V}_C} \mathbf{x}'_v, \tag{5}$$

where $\mathbf{x}'_v$ is the transformed feature vector of node v in community C. The feature mapping is then constructed using the fusion feature:

$$\mathcal{X} \longmapsto \widetilde{\mathcal{X}} = \{\widetilde{\mathbf{x}}_i \mid \widetilde{\mathbf{x}}_i = \widetilde{\mathbf{f}}_{C_i} \text{ for } C_i \in \mathcal{C}\}, \tag{6}$$

where $\widetilde{\mathbf{f}}_{C_i}$ represents the fusion feature of community C_i.

Next, we identify a subset of nodes within the current community C that have the highest feature similarity to the mapped node $\widetilde{v}$, and then perform majority voting on these nodes. Specially, we consider a community C with node labels $\{l_1, l_2, \ldots, l_n\}$ and their corresponding feature vectors $\mathbf{X}_C$. We calculate the Euclidean distance between each node feature vector $x_i \in \mathbf{X}_C$ in the community C and $\widetilde{\mathbf{f}}_C$.

$$d_i = \|\widetilde{\mathbf{f}}_C - \mathbf{x}_i\|_2. \tag{7}$$

We sort the distances in ascending order and calculate the gradient $g_i = d_{i+1} - d_i$ between consecutive distances. The distance corresponding to the maximum gradient is given by:

$$\tau = d_{\arg\max(g)}. \tag{8}$$

Finally, we select nodes with distances $d_i \leq \tau$ and aggregate their labels using majority voting:

$$\widetilde{l}_C = \text{mode}(\{l_i \mid d_i \leq \tau\}). \tag{9}$$

The label mapping is then constructed as:

$$\mathcal{Y} \longmapsto \widetilde{\mathcal{Y}} = \{\widetilde{y}_i \mid \widetilde{y}_i = \widetilde{l}_{C_i} \text{ for } C_i \in \mathcal{C}\}. \tag{10}$$

Edge Mapping. Using the adjacency matrix $\mathbf{A}$ to represent the graph $\mathcal{G}$, for each edge $e_{uv} \in \mathcal{E}$ where node u belongs to community C_i and node v belongs to community C_j, we define the edge count between the community pair (C_i, C_j) as $s_{ij} = \sum_{u \in \mathcal{V}_{C_i}} \sum_{v \in \mathcal{V}_{C_j}} \mathbf{A}_{uv}$. The robustness score R_{ij} of the connection between communities C_i and C_j is evaluated using the following formula:

$$R_{ij} = \left(\frac{s_{ij}}{\sqrt{D_i^{\text{out}}}}\right) \times \left(\frac{s_{ij}}{\sqrt{D_j^{\text{in}}}}\right) + \frac{s_{ij}}{|C_i \cup C_j|}, \tag{11}$$

where D_i^{out} and D_j^{in} denote the out-degree of C_i and the in-degree of C_j, respectively.

A smoothing function, $w_{ij} = \lambda \times \exp(-R_{ij}) + \eta$ is then employed to process the robustness score, representing the edge weight of the original community relationship. To store these weights, we introduce a set $\mathcal{W}$, where each element is a key-value pair: the key is the community pair (C_i, C_j), and the value is w_{ij}. Subsequently, a threshold σ is utilized to filter and construct the edge mapping $\mathcal{E} \to \widetilde{\mathcal{E}}$ as follows:

$$\mathcal{E} \longmapsto \widetilde{\mathcal{E}} = \{\widetilde{e}_{ij} \mid (C_i, C_j) \in \mathcal{W}, w_{ij} \geq \sigma\} \tag{12}$$

Finally, the total mapping relationship is established as follows:

$$\mathcal{G} \longmapsto \widetilde{\mathcal{G}} = \left\{\mathcal{V} \mapsto \widetilde{\mathcal{V}},\ \mathcal{E} \mapsto \widetilde{\mathcal{E}}, \mathcal{X} \mapsto \widetilde{\mathcal{X}},\ \mathcal{Y} \mapsto \widetilde{\mathcal{Y}}\right\}. \tag{13}$$

Node-Level Unlearning Strategy

Upon receipt of an unlearning request as a sequence, denoted as $\mathcal{V}_u = \{v_{u1}, v_{u2}, \ldots v_{un}\}$, CGE initially identifies the communities of the nodes in $\mathcal{V}_u \subset \mathcal{V}$ for this batch and subsequently returns a set of affected communities, denoted as $\mathcal{C}_u$. Based on the graph structure mapping $\mathcal{G} \mapsto \widetilde{\mathcal{G}}$, the set of all affected mapped nodes and edges, denoted as $\mathcal{I}_{\mathcal{V}_u}$, is obtained.

It is important to note that the application of node-level PCA computation acts as a *filter* to exclude non-principal components and unlearning requests that are not affected. If a node v_{ui} is not utilized in the calculation of the features and labels associated with the mapped node, it can be inferred that the information contained within the node is not retained within the mapped graph. Consequently, the influence of node-level unlearning of v_{ui} is effectively negated, thereby avoiding unnecessary expenditure of time and resources.

Based on $\mathcal{I}_{\mathcal{V}_u}$, CGE updates the relevant feature and structure information of the graph and generates an updated graph $\widetilde{\mathcal{G}}'$:

$$\widetilde{\mathcal{G}}' = \text{Update}_{\widetilde{\mathcal{G}}}(\widetilde{\mathcal{V}}', \widetilde{\mathcal{E}}', \widetilde{\mathcal{X}}', \widetilde{\mathcal{Y}}'). \tag{14}$$

The updated graph $\widetilde{\mathcal{G}}'$ and the mapped graph $\widetilde{\mathcal{G}}$ must satisfy the following constraints to completely eliminate specific data and its derived information within the model.

$$f_{\theta'}(\widetilde{\mathcal{G}}') \approx f_\theta(\widetilde{\mathcal{G}}) \quad \text{subject to} \quad \mathcal{V}_u \cup \mathcal{G}' = \emptyset. \tag{15}$$

Dataset	#Nodes	#Edges	#Classes	#Features	Density
Cora	2,708	5,429	7	14,33	0.148%
Citeseer	3,327	4,732	6	3,703	0.085%
CS	18,333	163,788	15	6,805	0.097%
Reddit	232,965	1.1×10^8	41	602	0.040%

Table 2: Statistics of Datasets.

Experiments

To evaluate the efficacy of CGE, we conducted a series of comprehensive experiments using four real-world datasets and three prevalent GNN backbones. The evaluation addresses the following questions: (1) Can CGE provide excellent and comparable model utility? (2) How efficient is CGE in practical applications? (3) Can CGE achieve deterministic graph unlearning? Additionally, a series of ablation studies were performed to examine CGE's superiority at each stage of the graph unlearning process. We also assessed the applicability of different community detection methods to CGE, as detailed in Appendix D [1].

Experimental Setup

Datasets. We evaluated the CGE on four real-world datasets of various sizes, including Cora (Yang, Cohen, and Salakhudinov 2016), Citeseer (Yang, Cohen, and Salakhudinov 2016), CS (Shchur et al. 2018) and Reddit [2], all of which are commonly used in GNN evaluations. The large-scale Reddit dataset was specifically included to assess the unlearning framework's performance in a real-world context. The statistics for these datasets are summarized in Table 2.

GNN Backbones. We equipped our CGE with three GNN backbones, GCN (Kipf and Welling 2016), GAT (Veličković et al. 2017), and GraphSAGE (Hamilton, Ying, and Leskovec 2017), to evaluate its versatility. Detailed descriptions of the three backbones can be found in Appendix A.

Baselines. To demonstrate the efficacy of CGE, we compare it with the state-of-the-art unlearning algorithms, including SISA (Bourtoule et al. 2021), GraphEraser (Chen et al. 2022b), and GUIDE (Wang, Huai, and Wang 2023), all of which are deterministic unlearning methods based on the BP-SM-TA paradigm. We also introduced *Scratch*, a scheme involving retraining from beginning to end, to evaluate comparable model utility. For each unlearning batch, 0.5% of the original dataset's nodes were randomly selected. More detailed experimental settings are provided in Appendix B.

Metrics. We consider two key aspects to measure the performance of CGE: model utility and unlearning efficiency. Model utility is assessed using the *Macro F1* score, while unlearning efficiency is evaluated based on the *time cost* required for model deployment and unlearning.

[1]All appendices are available in our arXiv version.

[2]https://docs.dgl.ai/generated/dgl.data.RedditDataset.html

Implementation. CGE is implemented using Python 3.8.19 and DGL[3]. All experiments were conducted on an NVIDIA Tesla A800 GPU server running the Ubuntu 23.04 LTS operating system.

Model Utility of CGE

To answer question (1), we conducted a comparative analysis of CGE against the best results from three baselines across three GNN backbones and four datasets. This analysis aims to verify CGE's model utility in comparison to *Scratch*, as referenced in the Preliminaries.

Results. The experimental results presented in Table 3 reveal the following findings: (1) CGE outperforms all established baselines in terms of model utility. For instance, on Reddit, CGE shows an average increase of 11.56% compared to GUIDE and an even more significant average increase of 33.24% compared to GraphEraser. This superiority is attributed to two factors: (a) CGE employs a graph-optimized community detection method, resulting in a more accurate community distribution than the balanced partitioning used in the BP-SM-TA paradigm. (b) CGE accounts for the inherent relationships between subgraphs, reducing information loss, a consideration overlooked by the BP-SM-TA paradigm. (2) CGE demonstrates comparable model utility to *Scratch*, which serves as the foundational baseline due to its broad and outstanding performance achieved through the entire retraining of the dataset. Notably, CGE has even outperformed *Scratch* in certain cases, primarily due to its pre-integration of similar data within the same community in the original graph, effectively reducing noise. This results in a newly mapped graph with more robust and accurate information. CGE achieves average performance closer to *Scratch*, fulfilling the objectives of unlearning.

Efficiency of CGE

To answer the question (2), we evaluate CGE's efficiency in two tasks: model deployment and unlearning.

In the ***model deployment stage***, the unlearning framework generates a complete prediction model, including graph partitioning, data pre-processing, and initial model training. In the ***unlearning stage***, the model performs deterministic unlearning on a demand sequence and produces an updated model based on the deployed one. Since the training time across three backbones on the same dataset is linear, we show the average efficiency of different unlearning frameworks across all backbones, as presented in Table 4.

During ***model deployment stage***, CGE's efficiency surpasses most baselines by retaining only the most representative information in the mapped graph. Compared to unlearning methods trained on the original graph, the mapped graph offers smaller-scale training data with higher representativeness. For example, as shown in Table 5, the CS dataset requires fewer training data and parameters for the mapped graph compared to the representative method GraphEraser under the BP-SM-TA paradigm.

[3]https://www.dgl.ai

Dataset	Model	Method				
		Scratch	SISA	GraphEraser	GUIDE	CGE
Cora	SAGE	0.7638 ± 0.0018	0.6024 ± 0.0073	0.6910 ± 0.0032	0.7226 ± 0.0031	$\mathbf{0.8745 \pm 0.0043}$
	GAT	0.7435 ± 0.0039	0.5793 ± 0.0027	0.7324 ± 0.0051	0.6640 ± 0.0049	$\mathbf{0.7463 \pm 0.0083}$
	GCN	0.7675 ± 0.0012	0.5190 ± 0.0057	0.6992 ± 0.0009	0.7403 ± 0.0093	$\mathbf{0.7586 \pm 0.0092}$
Citeseer	SAGE	0.7232 ± 0.0003	0.5189 ± 0.0021	0.7112 ± 0.0022	$\mathbf{0.7271 \pm 0.0022}$	0.7170 ± 0.0103
	GAT	0.7194 ± 0.0006	0.5020 ± 0.0007	0.7004 ± 0.0041	0.7102 ± 0.0019	$\mathbf{0.7473 \pm 0.0059}$
	GCN	0.7197 ± 0.0013	0.4771 ± 0.0011	0.6983 ± 0.0062	0.7033 ± 0.0011	$\mathbf{0.7082 \pm 0.0003}$
CS	SAGE	0.8913 ± 0.0007	0.5920 ± 0.0004	0.7033 ± 0.0007	0.8016 ± 0.0013	$\mathbf{0.8460 \pm 0.0004}$
	GAT	0.8811 ± 0.0014	0.6611 ± 0.0011	0.7227 ± 0.0014	0.6939 ± 0.0009	$\mathbf{0.7664 \pm 0.0039}$
	GCN	0.8907 ± 0.0023	0.7004 ± 0.0007	0.6313 ± 0.0022	0.7053 ± 0.0007	$\mathbf{0.7806 \pm 0.0032}$
Reddit	SAGE	0.9443 ± 0.0024	0.5901 ± 0.0013	0.5793 ± 0.0085	0.8201 ± 0.0012	$\mathbf{0.9451 \pm 0.0015}$
	GAT	0.9477 ± 0.0018	0.6611 ± 0.0029	0.7865 ± 0.0029	0.8817 ± 0.0091	$\mathbf{0.9138 \pm 0.0006}$
	GCN	0.9312 ± 0.0061	0.6882 ± 0.0044	0.7727 ± 0.0008	0.8033 ± 0.0009	$\mathbf{0.9302 \pm 0.0011}$

Table 3: The comparison of *Macro F1* scores across different graph unlearning frameworks on four datasets and three GNN backbones is shown. *Scratch* is a retraining scheme from beginning to end, which is used to evaluate the utility of comparable models. The best-performing unlearning framework's performance is in **bold**, and the runner-up's performance is underlined.

Method	Model Deployment				Unlearning			
	Cora	Citeseer	CS	Reddit	Cora	Citeseer	CS	Reddit
Scratch	131.52	109.62	702.11	13048.67	130.23	110.05	700.12	13056.33
GraphEraser	179.53	167.16	1042.07	61220.05	10.78	12.13	62.56	3128.22
GUIDE	124.58	127.76	**745.10**	39064.71	10.11	11.07	71.60	3411.89
CGE	**93.12**	**108.35**	747.82	**18143.95**	**8.12**	**9.58**	**9.33**	**49.52**

Table 4: The comparison of two-stage time consumption (in seconds) of different graph unlearning frameworks on four datasets.

During ***unlearning stage***, CGE's efficiency advantage becomes even more pronounced, significantly surpassing Scratch and all baselines. The graph structure mapping minimizes the impact of data size differences, allowing CGE to reduce unlearning time to seconds without sacrificing model utility. This efficiency results from both the exponential reduction in data and parameters and the node-level unlearning strategy. Unlike traditional methods that require retraining and aggregation of submodels, CGE only needs to recalculate the relevant mapped node-level data, significantly reducing time complexity.

Index	GraphEraser	CGE
#Nodes (T)	18,333	1,756
#Parameters (T)	43,498,280	217,875
#Parameters (U)	4,890,365	217,875

Table 5: Comparison of the quantity of training nodes and parameters across different unlearning frameworks on CS Dataset. Where 'T' and 'U' denote the training and unlearning processes, respectively.

Unlearning Ability

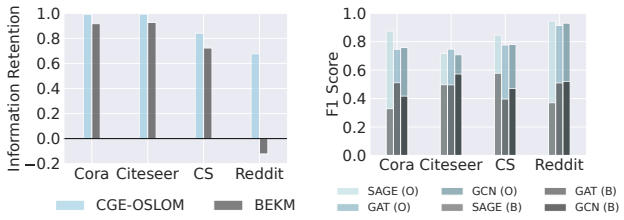
To determine the effectiveness of CGE in unlearning, we utilize a node-level GNN member inference attack methodology (MIA) (He et al. 2021). This approach involves calculating the discrepancy between the original model and the model after CGE unlearning, to determine whether a node has been successfully removed from the model. After randomly removing 0.5% of all nodes according to the experimental parameters, the MIA attack was executed. We consider two scenarios: $\mathcal{A}$ and $\mathcal{A}_{\mathcal{C}}$, representing the MIA attack results on the original GNN backbones and CGE, respectively. As shown in Table 6, the AUC of MIA on CGE is significantly lower than that of the original GNN backbones, approaching 0.5, which is analogous to random guessing. This indicates that CGE does not produce additional information leakage and that its unlearning of nodes and associated effects is effective.

Dataset	GCN		GAT		SAGE	
	$\mathcal{A}$	$\mathcal{A}_{\mathcal{C}}$	$\mathcal{A}$	$\mathcal{A}_{\mathcal{C}}$	$\mathcal{A}$	$\mathcal{A}_{\mathcal{C}}$
Cora	73.3	50.1	75.2	49.8	74.1	52.1
Citeseer	78.3	52.1	79.1	50.2	79.7	50.7
CS	73.1	50.1	72.3	49.3	70.8	48.8
Reddit	63.1	50.0	65.5	49.6	65.1	49.8

Table 6: Attack AUC of membership inference against GNN backbones and CGE (%).

Ablation Study

We aim to evaluate the contribution of different modules in CGE to the system by addressing the following questions:



(a) (b)

Figure 2: Quality and utility evaluation of different graph partitioning schemes.

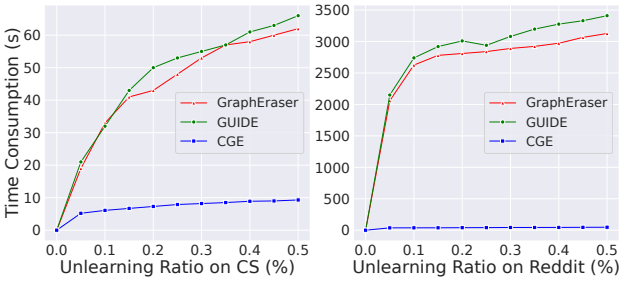


Figure 3: Time consumption of each method under different unlearning node ratios.

Compared to previous schemes, is overlapping community detection more advantageous? Most previous schemes use similar node feature clustering methods for subgraph-balanced partitioning, as BP-SM-TA requires each subgraph to represent the original graph. In contrast, CGE calculates a highly representative mapped graph without cutting the original graph, thereby limiting information loss. To evaluate the schemes for CGE, we use a custom metric, *Information Retention*, which measures cosine similarity between the low-dimensional embeddings of the original graph and the subgraph—higher values indicate less information loss. Additional details and validation experiments are provided in Appendix C. Figure 2(a) shows the evaluation results of different graph partitioning schemes Balanced Kmeans (BEKM) and Louvain-OSLOM (OSLOM) across four datasets. The balanced partitioning scheme is suboptimal, especially in the Reddit dataset, whose original graph representativeness is below zero, explaining why GraphEraser performs poorly on Reddit. In contrast, overlapping community detection better suits CGE. Figure 2(b) shows that BEKM (B) leads to a significant decrease in the performance of CGE when using OSLOM (O), further supporting the above explanation.

Why can CGE overcome the limitations of balanced partitioning? Previous experiments have shown that balanced partitioning is a suboptimal graph partitioning scheme, as it causes significant information loss due to graph cutting. This scheme is adopted by the BP-SM-TA paradigm to prevent imbalanced submodels from causing unmanageable learning time consumption, but it sacrifices model utility in the process. In contrast, CGE, based on the GSMU paradigm, offers a node-level unlearning solution with low parameter count and complexity. Figure 3 shows the unlearning efficiency of CGE on two large-scale datasets CS and Reddit under different unlearning node ratios. Notably, CGE avoids exponential increases in unlearning time as the number of forgotten nodes grows, thus eliminating the need for balanced partitioning as a protective measure.

Related Work

Machine unlearning aims to erase specific data and its derived effects. Following Cao's work (Cao and Yang 2015), this concept has evolved toward the development of both approximate and deterministic unlearning methods.

Approximate unlearning (Wu et al. 2023; Guo et al. 2019) achieves statistical unlearning by fine-tuning model parameters, but it cannot guarantee precise data erasure, posing challenges for non-convex models like deep learning. Conversely, deterministic unlearning ensures the complete removal of data points and their influence, often through retraining. SISA (Bourtoule et al. 2021) proposed a method involving random balanced partitioning into multiple submodels, where only the affected sub-models are retrained based on unlearning requests, and the final prediction is aggregated from these sub-models. This approach is referred to as the BP-SM-TA paradigm in this paper. Building on this, Chen et al. extended the concept to graph-structured data with GraphEraser (Chen et al. 2022b), and the recently proposed GUIDE (Wang, Huai, and Wang 2023) further optimized it for inductive graphs.

In contrast to previous work, the proposed CGE framework employs a novel graph structure mapping paradigm, which maps the original graph to a highly representative low-order graph. It introduces a low-parameter node-level unlearning strategy to enhance the utilization and efficiency of the graph structure within the unlearning framework.

Conclusions

In this work, we introduce GSMU, a novel graph structure mapping unlearning paradigm that offers an intuitive and efficient approach to graph unlearning. We propose the Community-Centric Graph Eraser (CGE) as a practical implementation of GSMU. CGE first detects communities and maps them to nodes, creating a mapped graph. This graph is then used to reconstruct node-level unlearning operations and representation learning tasks, resulting in high graph structure utilization and significantly reduced unlearning computational costs. Comprehensive evaluations demonstrate that CGE effectively leverages graph structures, offering excellent model utility and significantly improved unlearning efficiency. This work establishes a groundbreaking paradigm for graph unlearning with its streamlined architecture and superior performance.

CGE shows strong potential but faces challenges with traditional community detection techniques on large-scale graphs. Future work will leverage deep learning-based methods to enhance data insights while addressing privacy concerns, aiming for a balanced and effective approach.

Acknowledgments

This work was partially supported by the Project of Guangxi Science and Technology (GuiKeAB23026040), Research Fund of Guangxi Key Lab of Multi-source Information Mining & Security (MIMS24-13), Research Fund of Guangxi Key Lab of Multi-source Information Mining & Security (24-A-01-02), and Innovation Project of Guangxi Graduate Education (XYCSR2024102).

References

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10): P10008.

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 141–159. IEEE.

Cao, Y.; and Yang, J. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, 463–480. IEEE.

Chen, C.; Sun, F.; Zhang, M.; and Ding, B. 2022a. Recommendation unlearning. In *Proceedings of the ACM Web Conference 2022*, 2768–2777.

Chen, M.; Zhang, Z.; Wang, T.; Backes, M.; Humbert, M.; and Zhang, Y. 2022b. Graph unlearning. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, 499–513.

Cofone, I. 2020. *The right to be forgotten: A Canadian and comparative perspective*. Routledge.

Deng, J.; Chen, X.; Fan, Z.; Jiang, R.; Song, X.; and Tsang, I. W. 2021. The pulse of urban transport: Exploring the co-evolving pattern for spatio-temporal forecasting. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(6): 1–25.

Deng, J.; Chen, X.; Jiang, R.; Song, X.; and Tsang, I. W. 2022. A multi-view multi-task learning framework for multi-variate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 35(8): 7665–7680.

Deng, J.; Chen, X.; Jiang, R.; Yin, D.; Yang, Y.; Song, X.; and Tsang, I. W. 2024. Disentangling Structured Components: Towards Adaptive, Interpretable and Scalable Time Series Forecasting. *IEEE Transactions on Knowledge and Data Engineering*.

Deng, J.; Deng, J.; Yin, D.; Jiang, R.; and Song, X. 2023. TTS-norm: Forecasting tensor time series via multi-way normalization. *ACM Transactions on Knowledge Discovery from Data*, 18(1): 1–25.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph neural networks for social recommendation. In *The world wide web conference*, 417–426.

Guo, C.; Goldstein, T.; Hannun, A.; and Van Der Maaten, L. 2019. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

He, X.; Wen, R.; Wu, Y.; Backes, M.; Shen, Y.; and Zhang, Y. 2021. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429*.

Izzo, Z.; Smart, M. A.; Chaudhuri, K.; and Zou, J. 2021. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, 2008–2016. PMLR.

Jin, D.; Yu, Z.; Jiao, P.; Pan, S.; He, D.; Wu, J.; Philip, S. Y.; and Zhang, W. 2021. A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(2): 1149–1170.

Khan, B. S.; and Niazi, M. A. 2017. Network community detection: A review and visual survey. *arXiv preprint arXiv:1708.00977*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lancichinetti, A.; Radicchi, F.; Ramasco, J. J.; and Fortunato, S. 2011. Finding statistically significant communities in networks. *PloS one*, 6(4): e18961.

Li, C.; Cheng, D.; Zhang, G.; and Zhang, S. 2024a. Contrastive learning for fair graph representations via counterfactual graph augmentation. *Knowledge-Based Systems*, 305: 112635.

Li, Z.; Chen, B.; Xi, P.; Huang, Z.; Zhang, W.; and Zhang, S. 2024b. Spatio-Temporal Dynamically Fused Graph Convolutional Network. In *2024 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Li, Z.; Jian, X.; Wang, Y.; and Chen, L. 2022. Cc-gnn: A community and contraction-based graph neural network. In *2022 IEEE International Conference on Data Mining (ICDM)*, 231–240. IEEE.

Nguyen, T. T.; Huynh, T. T.; Nguyen, P. L.; Liew, A. W.-C.; Yin, H.; and Nguyen, Q. V. H. 2022. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*.

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

Qiu, J.; Tang, J.; Ma, H.; Dong, Y.; Wang, K.; and Tang, J. 2018. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2110–2119.

Said, A.; Derr, T.; Shabbir, M.; Abbas, W.; and Koutsoukos, X. 2023. A survey of graph unlearning. *arXiv preprint arXiv:2310.02164*.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, C.-L.; Huai, M.; and Wang, D. 2023. Inductive graph unlearning. In *32nd USENIX Security Symposium (USENIX Security 23)*, 3205–3222.

Wei, Y.; Yuan, H.; Fu, X.; Sun, Q.; Peng, H.; Li, X.; and Hu, C. 2024. Poincaré Differential Privacy for Hierarchy-aware Graph Embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 9160–9168.

Wu, J.; Yang, Y.; Qian, Y.; Sui, Y.; Wang, X.; and He, X. 2023. Gif: A general graph unlearning strategy via influence function. In *Proceedings of the ACM Web Conference 2023*, 651–661.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Yan, H.; Li, X.; Guo, Z.; Li, H.; Li, F.; and Lin, X. 2022. ARCANE: An Efficient Architecture for Exact Machine Unlearning. In *IJCAI*, volume 6, 19.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 40–48. PMLR.

Zhang, G.; Yuan, G.; Cheng, D.; Liu, L.; Li, J.; and Zhang, S. 2025. Disentangled contrastive learning for fair graph representations. *Neural Networks*, 181: 106781.

Zhang, S.; Li, X.; Zong, M.; Zhu, X.; and Cheng, D. 2017. Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(3): 1–19.