OpenGU: A Comprehensive Benchmark for Graph Unlearning

Bowen Fan
Beijing Institute of Technology
fan1085165825@gmail.com

Yuming Ai
Beijing Institute of Technology
3120251027@bit.edu.cn

Xunkai Li
Beijing Institute of Technology
cs.xunkai.li@gmail.com

Zhilin Guo
Shandong University
frank04180@outlook.com

Rong-Hua Li
Beijing Institute of Technology
lironghuabit@126.com

Guoren Wang
Beijing Institute of Technology
wanggrbit@gmail.com

Abstract

Graph Machine Learning is essential for understanding and analyzing relational data. However, privacy-sensitive applications demand the ability to efficiently remove sensitive information from trained graph neural networks (GNNs), avoiding the unnecessary time and space overhead caused by retraining models from scratch. To address this issue, Graph Unlearning (GU) has emerged as a critical solution to support dynamic graph updates while ensuring privacy compliance. Unlike machine unlearning in computer vision or other fields, GU faces unique difficulties due to the non-Euclidean nature of graph data and the recursive message-passing mechanism of GNNs. Additionally, the diversity of downstream tasks and the complexity of unlearning requests further amplify these challenges. Despite the proliferation of diverse GU strategies, the absence of a benchmark providing fair comparisons for GU, and the limited flexibility in combining downstream tasks and unlearning requests, have yielded inconsistencies in evaluations, hindering the development of this domain. To fill this gap, we present OpenGU, the first GU benchmark, where 16 SOTA GU algorithms and 37 multi-domain datasets are integrated, enabling various downstream tasks with 13 GNN backbones when responding to flexible unlearning requests. Through extensive experimentation, we have drawn 10 crucial conclusions about existing GU methods, while also gaining valuable insights into their limitations, shedding light on potential avenues for future research. Our code is available at https://github.com/bwfan-bit/OpenGU.

1 Introduction

Graphs are versatile mathematical structures that represent complex interactions and relationships between entities, offering an abstract yet intuitive framework for modeling real-world systems. To effectively capture the rich and interconnected information inherent in graph data, GNNs [21, 46, 60] have emerged as transformative tools, achieving remarkable success in data management [1, 2], social networks [39, 69], recommendation systems [6, 29, 59], and biological networks [24, 93, 55].

However, the majority of machine learning paradigms including GNNs are primarily driven by data for the acquisition of knowledge, fundamentally transforming decision-making processes across various fields [3, 76, 91]. Therefore, the indiscriminate use of data has intensified the conflict between data rights and user privacy [44, 63, 71]. In response to these pressing issues, a range of pioneering regulatory frameworks has been proposed, including the European Union's General Data Protection Regulation (GDPR) [53], and the California Consumer Privacy Act (CCPA) [50] in California. Among these regulations, one of the most critical and contentious provisions is *the right*

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks.

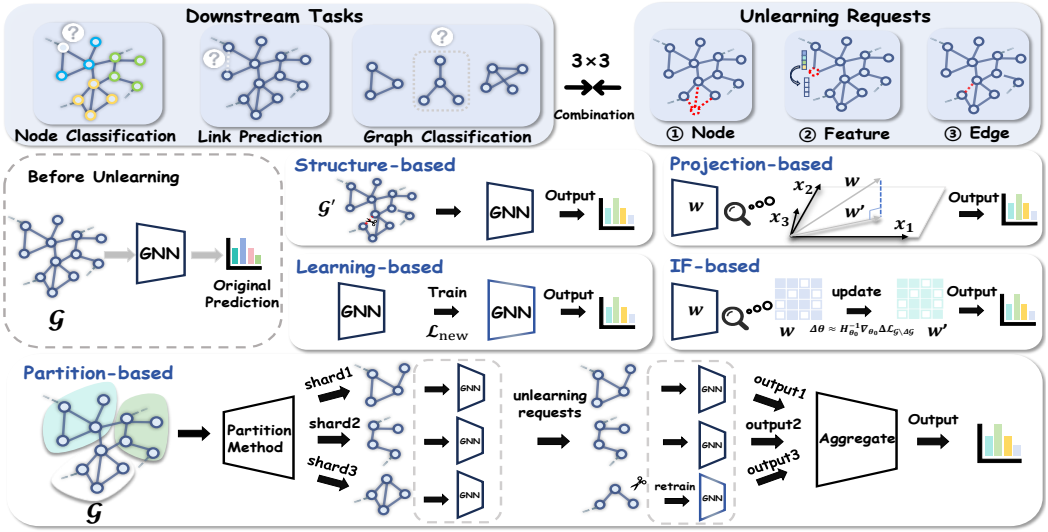

Figure 1: An overview of OpenGU framework, illustrating the key components and methodologies.

to be forgotten [35]. To technically align the effectiveness of machine learning with data rights and privacy regulations, a revolutionary frontier—machine unlearning [7, 77] has emerged.

Despite advances in traditional machine unlearning for independently distributed data, the growing reliance on graph-structured data in various fields highlights the significance of GU. As graphs underpin relational schemas and mined patterns, GU enables selective removal of sensitive information while preserving the integrity of analytical results. Unlike other domains, removing specific information from a graph requires not only modifying individual nodes or edges but also considering how these changes ripple through the entire graph. In addition, GU is jointly determined by downstream tasks (See Appendix A.2) and unlearning requests (See Appendix A.3). The interplay of these two aspects makes GU even more complex. These challenges position GU as a particularly demanding area of research, requiring tailored solutions for its structural complexities and task-specific needs.

Recently, diverse GU methods have surfaced, paving the way for more responsible and privacy-compliant graph learning via data-level techniques, model modifications, or parameter updates. Though these methods have made significant strides, there are still deficiencies in achieving unity:
(1) **Datasets**: The utilization of different datasets varies significantly in terms of domain characteristics and scale. This diversity, combined with the rigidity of processing data regarding splitting and inference settings, fundamentally makes it difficult to analyze the results from a unified perspective.
(2) **Backbones**: The lack of harmonization among GNN backbones between different methods creates barriers for fair comparisons, as experiments for many approaches are confined to specific GNNs, limiting their applicability to broader architectural variations and hindering cross-method evaluations.
(3) **Experiments**: Typically, experiments are conducted on a single downstream task under one type of unlearning request, neglecting the exploration of diverse task-request combinations. Besides, varying configurations and differing evaluation metrics lead to inconsistent interpretations of results.

To address the aforementioned challenges, we propose OpenGU, to our knowledge, the first comprehensive benchmark specifically designed for graph unlearning. OpenGU integrates 16 SOTA algorithms and 37 multi-domain datasets, enabling the flexible 3×3 combinations of unlearning requests and downstream tasks. OpenGU implements these components with unified APIs to standardize method invocation. Based on this design, we conduct an in-depth investigation of GU algorithms, providing valuable insights across three dimensions: effectiveness, efficiency, and robustness.
For **effectiveness**, OpenGU offers a comprehensive evaluation in two aspects: model updates and inference safety. We assess the forgetting ability for unlearning entities and the reasoning capability for retained data to determine if methods achieve effective trade-off between forgetting and reasoning.
For **efficiency**, OpenGU provides insights into scalability by evaluating how GU methods handle increasing amounts of data and more complex graph structures, showing their practical applicability. Furthermore, we analyze the computational resources required by various graph unlearning algorithms, focusing on time and space complexity from both theoretical and empirical perspectives.
For **robustness**, we focus on noise and sparsity scenarios in real-world applications, and delve into

the algorithms' capacity to maintain performance stability and integrity amid different unlearning intensities, particularly under varying levels of data perturbation.

Our contributions. We propose OpenGU in this paper, the inaugural benchmark specifically tailored for GU domain. Our contributions are outlined as follows:
(1) *Comprehensive Benchmark.* OpenGU integrates 16 SOTA algorithms and 37 popular multi-domain datasets, establishing a thorough evaluation framework for a fair comparison. Moreover, OpenGU expands and standardizes experimental task requirements, encompassing unlearning requests and downstream tasks, thereby facilitating a code-level implementation of 3×3 cross-experiments.
(2) *Analytical Insights.* Through extensive empirical experiments centered around algorithms' forgetting capability and reasoning capability, we conduct a thorough analysis from three perspectives and summarize 10 key conclusions, envisioning our conclusions as the catalyst for graph unlearning field.
(3) *Open-sourced Benchmark Library.* OpenGU is designed as an open-source benchmark library, providing researchers and practitioners with accessible tools and resources for expanding the exploration of graph unlearning. Additionally, comprehensive documentation and user-friendly interfaces foster collaboration and innovation within the GU community.

2 Definitions and Background

In this section, we will briefly review several key concepts and definitions to better explain the fundamentals of GU. Further details on GNNs and GU mechanisms can be found in Appendix A.1.

Graph Notations. Generally, we define a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V}$ represents a set of nodes with $|\mathcal{V}| = n$, and $\mathcal{E}$ denotes the edge set containing $|\mathcal{E}| = m$ edges. The feature matrix, $\mathcal{X} \in \mathbb{R}^{n \times d}$, represents the feature vectors of all nodes, and d represents the dimension of node features. We also define $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ as the label set, where each y_i corresponds to the node $v_i \in \mathcal{V}$ in a one-to-one manner. Apart from the aforementioned attributes, the graph is also characterized by its adjacency matrix $A \in \mathbb{R}^{n \times n}$, where each element A_{ij} indicates the edge between nodes i and j. Formally, we define $\mathcal{M}$ as the original model trained on a complete graph $\mathcal{G}$ or a graph dataset G.

Unlearning Requests. When receiving unlearning request $\Delta\mathcal{G} = \{\Delta\mathcal{V}, \Delta\mathcal{E}, \Delta\mathcal{X}\}$, the retrained model $\hat{\mathcal{M}}$ is trained from scratch on the pruned graph after the deletion and the unlearning model $\mathcal{M}'$ updates its parameters from original W to W' based on the unlearning algorithm. The goal of GU is to minimize the discrepancy between $\mathcal{M}'$ and $\hat{\mathcal{M}}$. Common unlearning requests in graph unlearning typically fall into three categories: node-level $\Delta\mathcal{G} = \{\Delta\mathcal{V}, \varnothing, \varnothing\}$, edge-level $\Delta\mathcal{G} = \{\varnothing, \Delta\mathcal{E}, \varnothing\}$, and feature-level $\Delta\mathcal{G} = \{\varnothing, \varnothing, \Delta\mathcal{X}\}$. Noting that $\Delta\mathcal{V}$ represents only the unlearning request received by the model, the edges connected to the nodes must also be considered during graph modification.

GU Taxonomy. To provide a comprehensive understanding, we categorize existing methodologies into five types based on their operational frameworks: (1) Partition-based algorithms, (2) Influence Function-based (IF-based) unlearning algorithms, (3) Learning-based algorithms, (4) Projection-based algorithms, and (5) Structure-based algorithms. This categorization provides a structured landscape of GU, as illustrated in Figure 1, with classification details provided in Appendix A.4.

3 Benchmark Design

In this section, we present a comprehensive overview of OpenGU, emphasizing the key aspects of the benchmark design, as outlined in Table 1. First, we detail the datasets and the associated preprocessing techniques (Sec. 3.1), followed by an examination of the various GU methods (Sec. 3.2). Lastly, we outline the evaluation metrics and experimental configurations (Sec. 3.3) that structure the benchmarking process, offering the core principles and the holistic view of OpenGU [1].

3.1 Dataset Overview for OpenGU

OpenGU rigorously selects domain-diverse datasets to evaluate graph understanding tasks: 19 datasets for node or edge-related work [33] (e.g., citation networks: Cora, PubMed [82]; co-author networks: CS and Physics [57]; e-commerce: ogbn-products [32]; webpage networks: Squirrel [51]) and 17 datasets for graph classification (e.g., compound networks: MUTAG [20], ogbg-molhiv [32]; protein networks: ENZYMES [8]; social networks: IMDB-BINARY [80]; 3D networks: ShapeNet [83]).

[1] Due to their specialized nature, the Projector and UtU methods are not categorized within mainstream GU algorithms in the table.

Table 1: An overview of OpenGU.

Datasets	
Type	Homophily, Heterophily, Node, Edge, Graph
Domain	Citation, Co-author, Social, Wiki, Image, Protein, Movie, ...
Preprocess	Transductive/Inductive, Label-Balanced/Label-Random, Noise, Sparsity
GNN Algorithms	
Decoupled	SGC, SSGC, SIGN, APPNP
Sampling	GraphSAGE, GraphSAINT, Cluster-gcn
Traditional	GCN, GCNII, LightGCN, GAT, GATv2, GIN
Mainstream GU Algorithms	
Partition-based	GraphEraser, GUIDE, GraphRevoker
IF-based	GIF, CGU, CEU, GST, IDEA, ScaleGUN
Learning-based	GNNDelete, MEGU, SGU, D2DGN, GUKD
Evaluations	
Attack	Membership Inference Attack, Poisoning Attack
Effectiveness	Accuracy, Precision, F1-score, AUC-ROC
Efficiency	Time, Memory, Theoretical Algorithm Complexity
Robustness	Deletion Intensity, GU Scenario Noise and Sparsity
Unlearning Request	Node-level Request, Feature-level Request, Edge-level Request
Downstream Task	Node Classification, Link Prediction, Graph Classification

To provide a clearer understanding of the datasets, a detailed overview is shown in Tables 4 and 5. Further details and preprocessing about datasets can be found in Appendix B.2 and B.3.

3.2 Algorithm Framework for OpenGU

GNN Backbones. To evaluate the generalizability of GU algorithms, we incorporate three predominant paradigms of GNNs within OpenGU: traditional GNNs [33, 65, 78], sampling GNNs [14, 27, 86], and decoupled GNNs [25, 72, 92]. This multi-faceted coverage enables rigorous benchmarking of GU algorithms across varying computational constraints, scalability demands, and representational capacities. More detailed descriptions of these GNN backbones are provided in Appendix C.

GU Algorithms. For GU algorithms, our framework encompasses 16 methods, each meticulously reproduced based on source code or detailed descriptions in the relevant publications. The detailed descriptions of GU can be found in Appendix D. Moreover, we deliver a unified interface for GU methods, merging them under a cohesive API to facilitate easier access, experimentation, and future expansion. By standardizing these methods within OpenGU, we provide a streamlined and highly efficient platform for researchers and practitioners to conduct robust, reliable, and reproducible benchmarking studies. More detailed information about OpenGU API can be found in Appendix E.1.

3.3 Evaluation Strategy for OpenGU

To assess GU algorithms in diverse real-world scenarios, OpenGU evaluation spans three dimensions tailored to GU contexts: effectiveness, efficiency, and robustness. Each dimension includes tailored evaluation methods reflecting OpenGU's mission to serve as a flexible, high-standard benchmark.

Cross-over Design. In previous GU studies, node and feature unlearning typically align with node classification tasks, while edge unlearning is often evaluated in the context of link prediction. However, real-world applications frequently demand the removal of data in scenarios where unlearning requests and downstream tasks intersect. To address this gap, we systematically modified and extended existing GU methods by decoupling their task-specific components and enabling cross-task compatibility through unified code interfaces. This approach provides a comprehensive and practical evaluation framework to assess the flexibility of GU algorithms in real-world applications. Further detailed information regarding the codebase implementation are provided in Appendix E.2.

Effectiveness. For the effectiveness of algorithms within OpenGU, we evaluate model performance on key downstream tasks while explicitly measuring unlearning impacts. We leverage F1-score, AUC-ROC, and Accuracy to evaluate the model's predictive performance at the node, edge, and graph levels, respectively. To rigorously validate unlearning effects, we employ membership inference attacks (MIA) [19, 48, 89] and poisoning attacks (PA) [94, 95], which systematically verify the erasure of target data. This multi-faceted approach provides a thorough assessment of GU effectiveness,

ensuring comprehensive evaluation of both retained and unlearned information. Rationales for metric selection and detailed methodological descriptions are provided in Appendix F.

Efficiency. In evaluating GU efficiency, we assess scalability, time complexity, and space complexity. Scalability examines adaptability to varying dataset sizes, reflecting performance stability across graph scales. Time complexity combines theoretical and empirical analysis to gauge computational demands. Space complexity focuses on memory efficiency, measuring peak usage and storage needs during unlearning to identify viability in resource-constrained settings. Together, these metrics provide a comprehensive view of each method's suitability for real-time and scalable deployment.

Robustness. To evaluate the robustness of algorithms in OpenGU, we systematically examine model performance under varying levels of deletion intensity, noise and sparsity. This involves assessing how different proportions of data perturbation affect the model's predictive and unlearning capabilities. Robust GU algorithms should ideally demonstrate minimal performance degradation as deletion intensity increases, reflecting strong resilience in maintaining optimum performance.

4 Experiments and Analyses

In this section, we delve into a series of experiments designed to rigorously evaluate the effectiveness, efficiency, and robustness of algorithms within OpenGU. By posing key questions, we aim to uncover insights into how these algorithms respond to diverse unlearning scenarios, data complexities, and practical deployment challenges, ultimately providing a comprehensive understanding of their efficacy. More detailed information about the experimental environment is presented in Appendix G.1.

For **effectiveness**, **Q1:** How effective are GU algorithms in predicting retained data under different unlearning requests? **Q2:** Do existing GU strategies achieve forgetting in response to unlearning requests? **Q3:** Do current GU algorithms effectively balance the trade-off between forgetting and reasoning? For **efficiency**,**Q4:** How do GU algorithms perform in terms of space and time complexity theoretically? **Q5:** How do the GU algorithms perform regarding space and time consumption in practical scenarios? For **robustness**, **Q6:** As the intensity of forgetting requests increases, can the GU algorithms still maintain their original performance levels? **Q7:** How do GU algorithms perform under sparse and noisy settings?

4.1 Reasoning Performance Comparison

To address **Q1**, we conducted a comprehensive comparison and analysis of existing methods across three representative combinations of downstream tasks and unlearning requests. For clarity, we denote these combinations as downstream task-unlearning request (e.g. node-node). For dataset configuration, we adopt an 80%/20% training-test split across all tasks. Unlearning requests are systematically defined: (1) node-level: removing 10% of the total nodes from the training set, (2) edge-level: deleting 10% of edges, and (3) graph-feature-level: selecting 50% of training graphs and setting 10% of their node features to zero vectors. More information about standardized experimental protocol and experimental settings can be found in Appendix G.2 and H.

Node-Node Experiment. From Table 2, we can reveal several key observations: (1) The best and second-best results are predominantly achieved by Learning-based methods, with SGU and D2DGN standing out. This indicates that the tailored strategies and loss function designs in Learning-based approaches effectively maintain SOTA performance for predicting retained data. (2) On smaller, commonly used datasets such as Cora, CiteSeer, and PubMed, most methods deliver relatively strong performance. However, on larger datasets like ogbn-arxiv, several methods, including CGU, GNNDelete, GUKD, and Projector, encounter challenges such as out-of-memory or out-of-time.

Edge-Edge Experiment. Based on the results presented in Table 9, several key observations can be drawn: (1) IF-based methods achieve superior performance in edge-edge experiments, retaining high link prediction accuracy after unlearning, with GIF and IDEA standing out. This demonstrates that leveraging influence functions enables strong generalizability. (2) Partition-based methods face challenges in effectively addressing link prediction tasks. This is primarily due to the inherent sparsity of edges in most datasets, where partitioning further reduces the number of meaningful edges. (3) While Learning-based methods excel in node-node scenarios, most of them fall significantly behind SOTA in edge-edge settings. However, GNNDelete consistently maintains high accuracy across all tested datasets. This gap likely arises because Learning-based approaches prioritize node-level optimization, neglecting explicit modeling of structural dependencies critical to edge removal.

Table 2: F1 ± STD comparison(%) for transductive node classification with node unlearning. The highest results are highlighted in bold, while the second-highest results are marked with underline.

Node-Level	Cora	CiteSeer	PubMed	ogbn-arxiv	CS	Flickr	Chameleon	Minesweeper	Tolokers
Retrain	90.41±0.13	77.18±0.24	86.89±0.19	70.39±0.04	93.51±0.06	49.03±0.32	61.75±0.18	81.29±0.05	79.26±0.05
GraphEraser	81.14±1.00	73.57±1.25	84.68±0.54	62.72±0.18	91.24±0.08	46.93±0.14	45.18±1.46	80.45±0.08	78.36±0.30
GUIDE	73.89±2.18	63.50±0.71	84.02±0.15	OOM	86.96±0.15	OOM	43.37±0.04	44.71±0.00	44.11±0.00
GraphRevoker	81.09±1.63	73.45±0.61	84.94±0.14	62.72±0.18	91.26±0.10	46.91±0.14	44.87±1.72	80.44±0.12	78.36±0.30
GIF	81.75±1.09	62.58±0.67	78.60±0.22	65.52±0.17	91.87±0.22	47.56±0.12	55.09±1.23	77.83±0.09	78.44±0.10
CGU	86.37±0.78	75.62±0.41	76.07±0.15	OOT	OOT	OOT	35.83±0.74	80.85±0.00	78.91±1.49
ScaleGUN	80.26±0.00	72.87±0.06	80.55±0.01	58.76 ±0.01	90.13±0.01	43.58±0.01	49.61±0.58	69.80±0.04	72.59±0.04
IDEA	87.71±0.25	63.66±0.49	80.44±0.13	64.22±0.17	89.47±0.22	42.01±0.04	52.89±0.80	77.93±0.04	78.72±0.15
CEU	87.12±0.07	71.56±0.15	**86.91±0.06**	57.80±0.83	92.23±0.07	**49.47±0.19**	**61.89±0.54**	81.05±0.04	78.72±0.03
GNNDelete	74.78±5.49	64.26±3.82	84.82±1.36	OOM	76.26±2.73	42.03±0.00	54.43±1.87	81.04±0.19	79.12±0.16
MEGU	82.68±1.56	63.60±1.11	79.68±0.60	66.15±0.29	91.69±0.08	47.97±0.13	53.82±1.68	77.64±0.02	**79.29±0.10**
SGU	**89.26±0.49**	72.04±1.70	86.61±0.02	**67.20±0.08**	**93.20±0.07**	48.70±0.21	60.18±0.26	81.11±0.02	79.18±0.04
D2DGN	88.41±0.20	73.81±0.28	86.06±0.05	66.08±0.18	92.99±0.03	48.01±0.03	53.25±0.70	**81.19±0.04**	79.18±0.05
GUKD	79.65±1.98	70.63±0.93	83.37±0.60	OOM	75.04±0.82	42.03±0.04	36.62±2.12	80.89±0.04	78.91±0.00
Projector	86.79±2.37	**77.00±0.65**	83.97±0.30	OOT	88.40±0.63	OOT	43.29±1.17	80.85±0.00	78.91±0.00

Graph-Feature Experiment. From the results presented in Table 10, we observe that: (1) IF-based methods remain effective for graph classification task, achieving commendable performance in most cases. Though the Partition-based approach is relatively straightforward, GraphEraser shows competitive performance across these datasets. (2) Although OpenGU provides comprehensive graph-level dataset interfaces, existing algorithms struggle to handle large-scale datasets due to inherent computational bottlenecks particularly in terms of memory scalability.

Through our exploration of **Q1**, we derive three key conclusions. **C1**: *Partition and IF-based methods demonstrate broad applicability to most tasks, while Learning-based methods, Projector, and UtU are more specialized, limiting their scalability and generalization* [17, 88]. **C2**: *Learning-based methods excel in targeted tasks, often achieving SOTA performance, while IF-based methods offer flexibility and maintain competitiveness in more tasks* [43]. **C3**: *The scalability gaps of GU algorithms manifest in two dimensions: memory bottlenecks during graph task, and the inherent incompatibility of current unlearning paradigms with feature-level unlearning requirements* [28, 70].

4.2 Forgetting Performance Comparison

To address **Q2**, we evaluate whether GU algorithms effectively forget the unlearning entity by employing commonly used attack strategies in the GU domain, including Membership Inference Attack and Poisoning Attack. These assessments are conducted from both node and edge perspectives to determine whether existing GU methods can genuinely prevent information leakage and protect privacy. The experimental setup in this section is identical to Section 4.1. More details about experimental setup and results can be found in Appendix I.

In node-node experiments, MIA assesses GU algorithms' privacy protection, with an AUC near 0.5 indicating predictions akin to random guesses, suggesting minimal information leakage and effective sensitive data removal. We assess various GU methods across multiple datasets, selecting CiteSeer as a representative case (Figure 2). The results reveal that: For Partition-based methods, GraphEraser underperforms with the BEKM partitioning strategy, while GUIDE with SR and Fast strategies excels in privacy protection. Learning-based methods like SGU, D2DGN, and GNNDelete achieve AUC values near 0.5, demonstrating robust unlearning capabilities. Although IF-based methods theoretically limit differences between original and unlearned models, CGU fails to meet the complete unlearning benchmark, as confirmed by MIA.

In edge-edge experiments, PA assesses GU methods' performance by introducing 10% heterophilic edges as poisoned data. The effectiveness of unlearning is measured by the improvement in link prediction AUC, with a larger increase indicating successful removal of poisoned edge information [2]. We evaluated various GU methods across datasets, selecting Cora as a representative case (Figure 3). The findings show that all Partition-based and IF-based methods achieve an increase in AUC

[2]The histogram column on the left represents the result before unlearning, and the corresponding one on the right represents the result after unlearning. And for convenience, UtU is presented in IF-based.

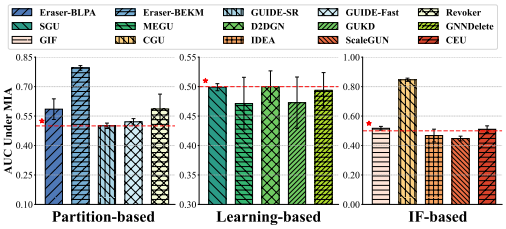

Figure 2: AUC-ROC ± STD comparison under MIA for node-node task with SGC backbone.

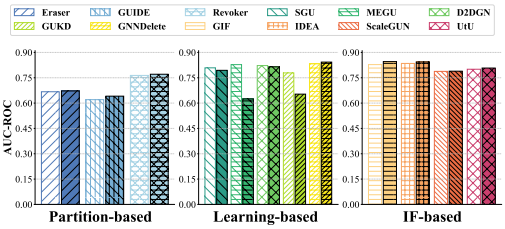

Figure 3: AUC-ROC comparison under PA for edge-edge task with GraphSAGE backbone.

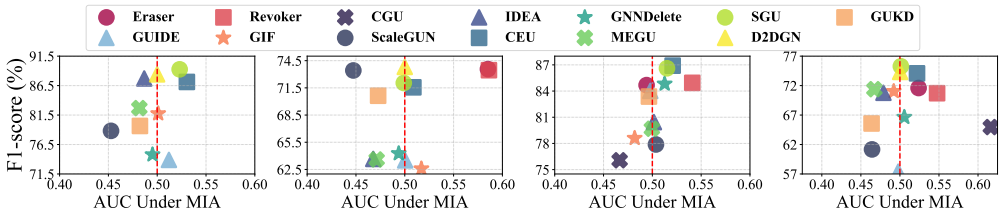

Figure 4: Trade-off between forgetting and reasoning on Cora, CiteSeer, PubMed and in average.

after unlearning, demonstrating their capability to address edge-level unlearning requests in link prediction tasks. Among these, GIF and IDEA stand out by not only effectively removing the harmful edges but also achieving high AUC scores. On the other hand, among the Learning-based methods, only GNNDelete successfully handles the removal of poisoned edges during unlearning, achieving a notably high AUC. Other Learning-based approaches exhibit varying levels of performance decline, suggesting that the unlearning process might inadvertently affect critical information.

Based on our analysis of **Q2**, we draw the conclusion **C4**: *For learning-based methods, only GNNDelete shows superiority in edge-level tasks, highlighting the need for further development in edge-level learning-based approaches* [38]. **C5**: *IF-based methods provide theoretical unlearning guarantees, but MIA reveals that some fail to fully eliminate privacy leaks, necessitating the integration of theoretical analysis and empirical evaluation* [10].

4.3 Trade-off between Forgetting and Reasoning

To address **Q3**, we synthesize insights from the preceding questions and adopt a unified perspective to analyze their interplay, providing a clear overall performance of various methods in balancing forgetting and reasoning (Figure 4). In this part, we focus on node classification on Cora, CiteSeer and PubMed, using SGC as the backbone for consistent comparisons, with the experimental setup adhering to configurations in Appendix H.1. For additional experimental details, refer to Appendix J.

Given that an AUC of 0.5 under the MIA serves as the theoretical baseline for indistinguishable inference, GU methods positioned closer to the red centerline and higher on the graph exhibit superior overall performance in balancing unlearning and retention. From a vertical perspective, which reflects reasoning capabilities, SGU, D2DGN, and CEU emerge as top performers, demonstrating exceptional strength in maintaining predictive accuracy on retained data. In contrast, GUIDE displays comparatively weaker reasoning performance, struggling to uphold effectiveness in this aspect. Horizontally, the distribution of AUC scores under MIA reveals distinct trends: apart from CGU, which attains an AUC exceeding 0.6, markedly deviating from the theoretical baseline, most methods cluster within a narrow range of 0.45 to 0.55. This concentration underscores a complex trade-off between unlearning efficacy and retention fidelity, evidencing phenomena such as over-forgetting, where excessive unlearning impairs utility, and under-forgetting, where incomplete unlearning falls short of privacy objectives. Among the evaluated methods, SGU, D2DGN, GNNDelete, and GUIDE distinguish themselves with robust unlearning capabilities, effectively reducing the influence of unlearned data, though their success in navigating these competing demands varies.

Upon exploring **Q3**, we conclude that **C6**: *Future algorithms should prioritize improving the balance between forgetting and reasoning trade-offs. The issues of under-forgetting and over-forgetting constitute critical challenges, requiring further research to enhance effectiveness* [16, 45].

Table 3: Algorithm complexity analysis for existing prevalent GU studies.

Method	Preprocessing	Training	Unlearning	Inference	Memory
GUIDE	$O(ktn^2+kctn)$	$O(Lfn/k+Lf^2n)$	$O(Lk'fn^2/k^2+Lkf^2n/k)$	$O(Lfm+Lf^2n+L_kfn)$	$O(n^2/k^2+Lfn+kn)$
GraphEraser	$O(kdtn+ktn\log(kn))$	$O(Lfn/k+Lf^2n)$	$O(Lk'fn^2/k^2+Lkf^2n/k)$	$O(Lfm+Lf^2n+Ln_sf/k+Ln_sf^2)$	$O(n^2/k^2+Lfn+kn)$
GraphRevoker	$O(k(d+c)n)$	$O(Lfn/k+Lf^2n)$	$O(Lk'fn^2/k^2+Lkf^2n/k)$	$O(Lfm+Lf^2n+kf^2n_s)$	$O(n^2/k^2+Lfn+kn)$
GIF	-	$O(Lfm+Lf^2n)$	$O(n\|\theta\|)$	$O(Lfm+Lf^2n)$	$O(\|\theta\|)$
CGU	-	$O(Lfm+f^2n)$	$O((Lmf+f^2n)u)$	$O(Lfm+f^2n)$	$O(m+f^2+fn)$
CEU	-	$O(Lfm+Lf^2n)$	$O(t\|\theta\|+u\|\theta\|)$	$O(Lfm+Lf^2n)$	$O(\|\theta\|)$
GST	-	$O(\sum_{i=0}^{N} pg_i^2)$	$O(((p+\|\theta\|)\sum_{i=0}^{N} g_i^2+\|\theta\|^3)u)$	$O(\sum_{i=0}^{N} pg_i^2)$	$O(pfn)$
IDEA	-	$O(Lfm+Lf^2n)$	$O(n\|\theta\|)$	$O(Lfm+Lf^2n)$	$O(\|\theta\|)$
ScaleGUN	-	$O(Lfm+f^2n)$	$O(L^2d^2u)$	$O(Lfm+f^2n)$	$O(m+f^2+fn)$
SGU	$O(Bf^2+Bn_s+f^2u)$	$O(Lfm+f^2n)$	$O(Lf^2+Bn_sf+(c+f)u)$	$O(Lfm+f^2n)$	$O(Bfn_s+f^2)$
MEGU	$O(Lfm+Lf^2n+d^2u)$	$O(Lfm+Lf^2n+d^2u)$	$O(d^2cu)$	$O(Lfm+Lf^2n)$	$O(Lfm+f^2n)$
GUKD	$O(Lfm+Lf^2n)$	$O(L_tfm+Lf^2n)$	$O(c(n-u))$	$O(Lfm+Lf^2n)$	$O(f^2+fn)$
D2DGN	$O(Lfm+Lf^2n)$	$O(Lfm+Lf^2n)$	$O(c(n-u))$	$O(Lfm+Lf^2n)$	$O(f^2+fn)$
GNNDelete	$O(Lfm+Lf^2n+d^2u)$	$O(Lfm+Lf^2n)$	$O(d^2fu)$	$O(Lfm+Lf^2n)$	$O(fu+d^2fu)$
Projector	-	$O(Lfm+Lf^2n)$	$O(f^2n+\max\{u^3,f^2u\})$	$O(Lfm+Lf^2n)$	$O(f^2+fn)$
UtU	-	$O(Lfm+Lf^2n)$	$O(u)$	$O(Lfm+Lf^2n)$	$O(m+Lfn)$

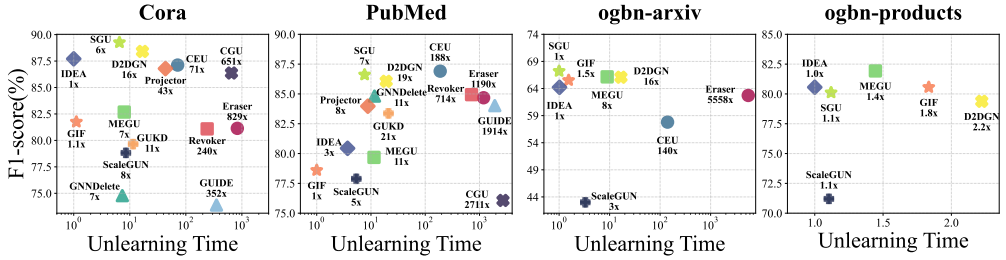

Figure 5: Unlearning Time Performance on Cora, PubMed, ogbn-arxiv and ogbn-products.

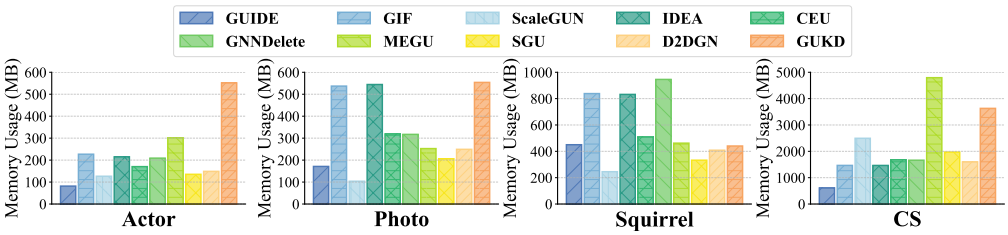

Figure 6: Memory Usage Performance on Various Datasets.

4.4 Algorithm Complexity and Practical Efficiency Analyses

Since **Q4** and **Q5** are closely interconnected, we integrate their analyses in this section. More details regarding notation definitions and experimental specifics are provided in Appendix K.

To address **Q4**, we analyze the time complexity from the perspectives of preprocessing, training, unlearning, and inference. As detailed in Table 3, the training and inference phases reveal that GST deviates from the complexity of standard GNN training, while other methods align with conventional expectations. In the unlearning phase, both time and space complexity vary based on method-specific characteristics: Partition-based methods exhibit complexities comparable to those of training, IF-based methods are predominantly governed by the parameter size $|\theta|$ and constrained by the extent of optimization applied to Hessian matrix computations, and Learning-based methods scale with data attributes, such as the feature dimension f. For **Q5**, Figure 5 illustrates that Partition-based methods incur substantial time overheads due to the combined demands of partitioning, aggregation, and shard training. In contrast, most IF-based and Learning-based methods demonstrate greater temporal efficiency. When evaluated on large-scale datasets like ogbn-products, only six GU methods completed execution successfully with competitive performance, as others encountered timeouts or memory overflows. As depicted in Figure 6, GUIDE consistently exhibits low memory overhead across various datasets, while GraphEraser (without visualization) incurs higher costs, exceeding 4000MB even on smallest dataset Actor. Notably, scalable methods such as ScaleGUN and SGU exhibit consistent memory usage across all datasets, underscoring their robustness and scalability.

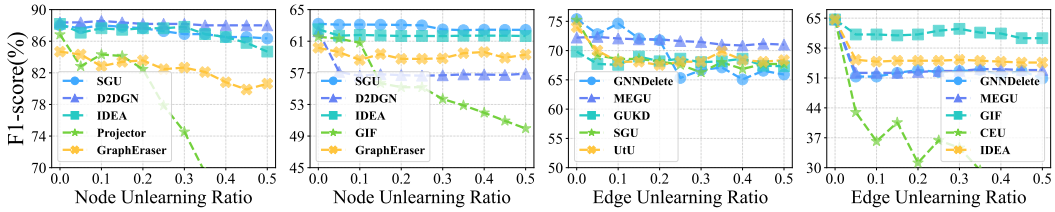

Figure 7: Performance under Different Unlearning Intensities.

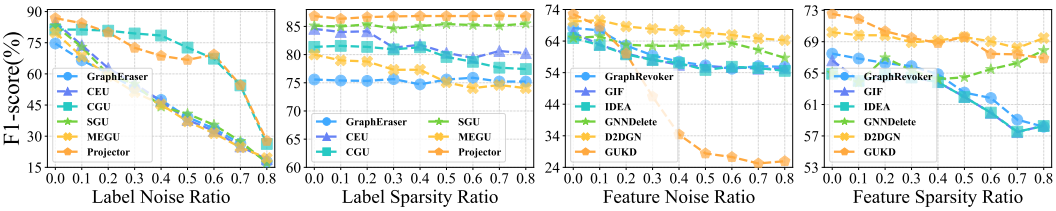

Figure 8: Performance under Different Noise and Sparsity Ratios at Label and Feature Levels.

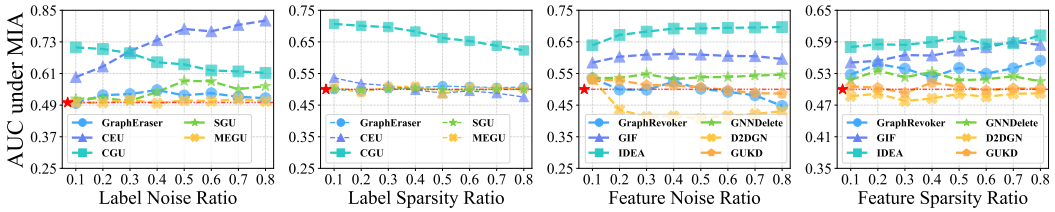

Figure 9: MIA under Different Noise and Sparsity Ratios at Label and Feature Levels.

Based on the analysis, we conclude **C7**: *To reduce time costs, partition-based methods require optimization in partitioning, IF-based methods need to minimize the computational overhead of Hessian matrix calculations in GU scenarios, and Learning-based approaches demand enhanced efficiency in preprocessing* [9, 79]. **C8**: *Existing GU methods require further optimization to effectively reduce time and space overhead, emphasizing the need for more efficient implementations* [26, 64].

4.5 Robustness Analyses

To investigate the robustness of GU algorithms, we designed comprehensive experiments targeting unlearning intensity and noise perturbations. These experiments systematically assess the algorithms' performance in challenging scenarios, with further details provided in Appendix L.

To address **Q6**, we perform two types of unlearning experiments. The first investigates the influence of varying unlearning ratios in a single batch, while the second explores the behavior of models under incremental unlearning requests. For the first experiment, we conducted node unlearning experiments on Cora and ogbn-arxiv and edge unlearning experiments on CiteSeer and Chameleon, employing various backbones for the node classification task. The unlearning ratio was incrementally increased from 0 to 0.5, and the results are presented in Figure 7. The illustration reveals a consistent downward trend in performance for all GU methods as the unlearning ratio increases, indicating that higher deletion intensities negatively impact prediction capabilities. Notably, methods such as Projector, GIF, and CEU exhibit greater sensitivity to changes in certain datasets, while Learning-based methods demonstrate a more gradual decline, highlighting their robustness under higher unlearning intensities. However, we also observe that even with minimal deletion ratios, many methods experience significant performance degradation during unlearning, particularly on the Chameleon dataset. This suggests that current GU algorithms need to take the deletion ratio into account to better reduce the gap in predictive performance between the unlearned and original model. To examine the second scenario, we perform incremental unlearning experiments, where 5% of the target nodes are sequentially removed in multiple rounds. As shown in Figure 10, all methods experience a sharp initial drop followed by a gradual decline, reflecting partial adaptation to continuous unlearning and distinct robustness differences among algorithms. In **Q7**, we simulate more realistic noise and sparsity scenarios by introducing perturbations at both the label and feature levels to comprehensively evaluate

the robustness of existing GU methods. For label noise, a certain proportion of training samples are randomly assigned incorrect labels, while for feature noise, Gaussian noise is injected based on the dimensionality of node features. Sparsity is introduced by varying the proportion of training nodes and simulating partial feature absence. Given the large number of GU methods, we select representative approaches for analysis based on their categories, using the same settings as the node-node experiments in Q1 and adopting SSGC as the backbone. Experiments on the Cora and CiteSeer datasets show that simulated noise and sparsity cause a performance drop across all methods, with label noise exhibiting a notably stronger effect (as illustrated in Figure 8). For label noise, CGU and Projector degrade slowly then sharply, unlike the steady and steep drop in other methods. Label sparsity has a milder effect, with some methods retaining performance. Feature noise markedly impairs GUKD, revealing its vulnerability, while feature sparsity reduces performance in GIF, IDEA, and GraphRevoker, yet GNNDelete improves unexpectedly. As indicated by Figure 9, representative methods are comparatively less affected by Label Sparsity Ratio. In contrast, most GU methods fail to exhibit a distinct or unified trend when faced with other forms of perturbations.

Based on analyses, we derive conclusion **C9**: *While most GU algorithms perform reasonably well across varying unlearning intensities, there remains a critical need to enhance their robustness across diverse datasets* [75]. **C10**: *Current GU methods exhibit insufficient robustness to noise and sparsity, particularly in the presence of label noise, posing a challenge to enhance the overall robustness* [54].

5 Conclusion and Future Directions

In this paper, we first review advancements in GU, detailing its applications and classifying algorithms by technical traits. Then we introduce OpenGU, the first unified and comprehensive benchmark for GU, which integrates 16 GU algorithms and datasets across multiple domains, supporting flexible combinations for downstream tasks and unlearning requests. Through OpenGU's standardized evaluation, we extensively assess GU methods' effectiveness, efficiency, and robustness, summarizing 10 insightful conclusions. To inspire further research, we outline the major challenges currently faced by GU and propose promising directions for future exploration.

Designing Generalized GU Frameworks for Diverse Tasks (*C1*, *C2* and *C3*). While some existing methods demonstrate strong predictive performance for specific tasks, their underlying principles often make it difficult to adapt to varying downstream tasks and unlearning requests. Furthermore, the current design of GU algorithms remains relatively simplistic, whereas real-world applications often require handling mixed unlearning requests or diverse graph types, such as dynamic graphs, spatialtemporal graphs, or knowledge graphs. Achieving consistently superior predictive capabilities across such complex and varied scenarios remains a considerable challenge.

Unified Metrics for Evaluating Forgetting (*C4*, *C5* and *C6*). The current methods for assessing the forgetting capability remain insufficient, as they are tightly coupled with specific unlearning requests and downstream tasks, making them less effective in handling diverse combinations of scenarios. Future research should also move beyond the current paradigm of independently assessing these two aspects (forgetting and reasoning), striving instead for a unified metric that evaluates models from an integrated perspective, ensuring a comprehensive understanding of their capabilities.

Enhancing Algorithm Efficiency (*C7* and *C8*). While theoretical analysis provides valuable insights, the practical performance of current GU methods often falls short, especially when scaling to large datasets with millions of nodes. Current methods commonly encounter OOT or OOM issues. To enable GU's effective deployment in large-scale scenarios, algorithms must be optimized for both efficiency and scalability to avoid these performance bottlenecks.

Addressing Realistic Scenarios (*C9* and *C10*). In practical applications, the presence of noise and incomplete datasets is an unavoidable challenge. However, current GU algorithms lack sufficient exploration and adaptation to such scenarios. Experimental results highlight significant weaknesses when dealing with noise and sparsity, particularly in terms of label and feature robustness. Future research should aim to broaden the scope of investigation, extending robustness analysis to encompass a wider range of real-world challenges and data imperfections.

6 Acknowledgement

This work was supported by the NSFC Grants U2241211, U24A20255, and 62427808. Rong-Hua Li is the corresponding author of this paper.