Learning to Reduce the Scale of Large Graphs: A Comprehensive Survey

HONGJIA XU, Zhejiang University, China
LIANGLIANG ZHANG, Rensselaer Polytechnic Institute, USA
YAO MA, Rensselaer Polytechnic Institute, USA
SHENG ZHOU*, Zhejiang University, China
ZHUONAN ZHENG, Zhejiang University, China
JIAJUN BU, Zhejiang University, China

Graph data, prevalent across domains like social networks, biological systems, and recommendation systems, presents significant challenges due to its large scale and complex structure. The advent of Graph Neural Networks (GNNs) has revolutionized graph data mining by effectively capturing node dependencies and neighborhood information. However, the computational complexity of processing large-scale graphs remains a major hurdle, as real-world graphs often consist of millions or even billions of nodes and edges. Efficient techniques like message passing and sampling have helped mitigate this issue, but memory and processing constraints persist. A promising approach to addressing these challenges is learning to reduce the size of large-scale graphs while retaining essential information, thus facilitating faster and more efficient graph data mining tasks, such as graph condensation, reduction, coarsening, and summarization, etc. Despite the differences in terminology, approaches under these topics share the same motivation: to generate smaller yet informative graphs that can replace the original large-scale datasets. In this paper, we unify these approaches under the concept of **Graph Scaling (GS),** highlighting the shared motivation across multiple topics. Alongside this definition, to clarify the question of what principles should be followed when scaling a graph and how a scaled graph was formulated, we propose a taxonomy to methodically categorize and understand existing methods. Moreover, by organizing the dataset and evaluation metrics, we aim to provide a more comprehensive understanding of the GS methods from a practical perspective. Moving forward, We delve into the limitations and challenges of GS methods, identifying the shortcomings and potential in the literature. Finally, we conclude this paper by outlining future directions and offering concise guidelines to inspire future research in this field. A full paper list and online resources about GS are available at https://github.com/Frostland12138/Awesome-Graph-Scaling.

CCS Concepts: • **Information systems → Data mining**.

Additional Key Words and Phrases: Graph Data Mining, Data-centric Machine Learning, Graph Scaling

*Corresponding author

Authors' addresses: Hongjia Xu, The State Key Laboratory of Blockchain and Data Security, Zhejiang University, Hangzhou, China, xu_hj@zju.edu.cn; Liangliang Zhang, Rensselaer Polytechnic Institute, Troy, NY, USA, zhangl41@rpi.edu; Yao Ma, Rensselaer Polytechnic Institute, Troy, NY, USA, may13@rpi.edu; Sheng Zhou, Zhejiang Key Laboratory of Accessible Perception and Intelligent Systems, Zhejiang University, Hangzhou, China, zhousheng_zju@zju.edu.cn; Zhuonan Zheng, Zhejiang University, Hangzhou, China, zhengzn@zju.edu.cn; Jiajun Bu, Zhejiang Key Laboratory of Accessible Perception and Intelligent Systems, Zhejiang University, Hangzhou, China, bjj@zju.edu.cn.

ACM 1556-472X/2025/4-ART
https://doi.org/10.1145/3729427

1 INTRODUCTION

Graph data, representing relationships and interactions between entities, is ubiquitous in various domains including social networks [1], biological systems [2], and recommendation systems [3]. Social networks leverage graph structure to analyze user connections and interactions, helping to identify key patterns and community structures. In biological systems, graphs are used to model complex molecular structures such as proteins, aiding in the discovery of new drugs. Recommendation systems utilize graph structure to model personalized suggestions by analyzing user preferences and item relationships. Information and patterns in those scenarios have been modeled as nodes and edges, and there has been significant progress in the development of techniques for large-scale graph data mining and pattern recognition, such as community detection [4], link prediction [5], and node classification, and so on. These advancements aim to apply data mining methodologies to extract valuable insights from extensive graph datasets. Graph Neural Networks (GNNs) [6–8], which generalize neural networks to work directly with graph-structured data, have revolutionized the graph data mining tasks by effectively capturing the dependencies between nodes and neighbors and efficiently aggregating information from neighbors through message-passing mechanisms [9]. The ability of GNNs to uncover hidden patterns and learn representations that encapsulate both local neighborhood information and global graph structure has made them a powerful tool in the field of graph data mining.

The continual evolution of graph data mining techniques, particularly with the integration of GNNs, is enabling more sophisticated analysis and applications across diverse fields. However, real-world graphs typically consist of millions or even billions of nodes and edges, making their analysis computationally challenging yet incredibly valuable. For instance, the Twitter (X) social graph contains hundreds of millions of users and billions of tweets and follower relationships; the Amazon product co-purchasing network comprises millions of products and billions of co-purchase relationships, etc. The complexity and scale of real-world graphs, specifically the complex structure of large-scale graphs, characterized by massive node connectivity and significant data volume, pose significant challenges to computational efficiency and resource requirements when analyzing and processing large-scale graphs [10].

One of the primary strategies to address these challenges is efficient sampling and aggregation techniques, such as Message Passing (MP) strategies used in GraphSAGE [11], etc. These methods reduce the computational load by sampling a subset of nodes and their neighbors instead of processing the entire graph, thereby significantly lowering memory usage and computational requirements and enabling more scalable GNN training and deployment. However, as MP requires nodes to aggregate information from their neighbors, the local graph structures must be maintained during both forward and backward propagation. This still results in significant consumption of memory and processing time.

Despite the recent advances in efficient GNNs from a model-centric perspective, there has been an increasing interest in data-centric solutions. A prime and straightforward strategy within this domain is to reduce the volume of the training dataset. Particularly, the concept of Dataset Distillation [12] has attracted noteworthy attention and yielded significant success, predominantly within vision datasets. Conventional dataset distillation relies on the idea that within categories defined by class labels, instances of the same class share similar key features, e.g., shape patterns in vision datasets. This implies the existence of 'prototypes' or 'clustering centers', and thus a significant amount of redundant information exists among instances belonging to the same category. This principle of redundancy and prototype-based representation finds a parallel in graph datasets, especially in tasks like node classification. The features and topology of nodes categorized under the same class tend to exhibit similarity, and there may be numerous repetitive and similar subgraph structures in the graph. Consequently, a natural question arises: *How can we effectively formulate small-scale graphs from large-scale graphs to facilitate various graph data mining tasks?*

Recently, **Graph Dataset Distillation** [12] has emerged for distilling large-scale graphs into smaller yet informative ones. By eliminating redundant information and retaining key information from the original dataset, these methods outputs small graphs that are more manageable within the constraints of limited computation resources, thereby providing better support for graph data mining tasks and applications such as Continual learning [13] and Network Architecture Search (NAS) [14], etc. While the concept aligns with vision dataset distillation, the specific challenges posed by the uniqueness of graph data motivate us to: (1) Address the lack of universal definitions and (2) Explore and synthesize the existing knowledge in this domain into a comprehensive survey.

1.1 Related topics and surveys.

When discussing the idea of reducing graph dataset complexity by volume, there are several relevant topics: Specifically, **Graph Sampling [15]** were designed to **select** subgraphs from original graphs, including Core-set [16], sparsification [17] and subgraph mining [18] methods, etc. **Graph Summarize [19]** focuses on graph representations such as scaled graphs and embeddings, while **Graph Pooling [20]** solely focuses on graph embeddings. **Graph Condensation** [21] focuses on synthesizing a smaller, representative graph that preserves the knowledge of the original graph or the training trajectories of the model, through optimization or generative methods. According to [22], **Graph Reduction** is a broader term that encompasses various strategies to decrease the size of a graph, including node/edge removal or aggregation, and condensation approaches. **Graph Distillation** [12] follows the definition of vision dataset distillation, typically by retaining critical information or patterns that are most relevant for downstream tasks. **Graph Coarsening** [23] specifically refers to grouping nodes or edges to create a coarser representation of the graph, retaining key information of the graph via eliminating redundancy.

Table 1. Related Topic and Survey

Topic	Survey	What is wanted	Retain info from		How to get it	
			Graph	Model	Modify	Synthetic
Scaling	**ours**	Small Graphs	✓	✓	✓	✓
Condensation	[21]		✓	✓		✓
Reduction	[22]		✓	✓	✓	✓
Distillation	[12]			✓		✓
Coarsening	[23]		✓		✓	
Sampling	[15]	Sub-graphs	✓	✓	✓	
Summarize	[19]	Graphs or Embeddings	✓		✓	✓
Pooling	[20]	Embeddings	✓	✓		✓

As shown in table 1, we analyze the similarities and differences among these topics from three aspects: what is wanted (output), what information to retain (objective), and how to formulate the outputs (formulations). Specifically, in the 'Retain info from' column, 'Graph' and 'Model' refer to the original graph and the model trained on it, respectively. In the last two columns, 'Modify' refers to approaches modifying the original graph, e.g., node sampling or aggregation, etc.; while 'Synthetic' involves generating outputs via neural networks or optimizing from an initialization. Despite differences in phrasing, the following topics (Condensation, Reduction, Distillation, and Coarsening) share (1) a similar problem definition: optimizing an objective to get small and informative graph datasets; (2) the same input-output domain, i.e., Graphs → Graphs. Therefore, it is crucial to define what constitutes valuable information within the domain of graph data and explore how to generate scaled graphs, for summarizing existing methods in the field and guiding future developments. The latest and related survey (1) on Graph Condensation [21] adopts a pragmatic perspective, categorizing the design of methods based on five common concerns in machine learning ('effectiveness', 'generalization', 'efficiency', 'fairness', and 'robustness'). Their work emphasizes the relationship between method design and downstream tasks, but it merely restricts the motivation to retain downstream task performance. Our focus extends beyond model performance, where we also prioritize preserving the intrinsic properties of graphs in both the spatial and spectral domains; (2) Graph Reduction [22] categorizes learning objectives independently in the areas of

sparsification, coarsening, and condensation, but does not further explore the commonalities among the learning objectives within these three domains. In this paper, we recognize a common motivation underlying various topics, i.e., learning to generate small and informative graphs. Based on that, we propose a unified analytical framework to understand and categorize the design of these methods.

1.2 Scope and Organization of this paper.

In this paper, we name the problem of 'how to get small yet informative graphs' as **Graph Scaling (GS)**. Firstly, we formulate the problem definition of GS by two aspects, i.e., defining what information to maintain from the original graphs and how to formulate the output small graphs (namely, Scaled Graphs). Under this definition, we categorize and analyze a wide range of methods from the aforementioned subfields of similar motivations within a unified taxonomy framework. Moreover, we collect and analyze the commonly used datasets and evaluation metrics in these fields, offering a robust framework for evaluating the effectiveness and applicability of GS methods across diverse scenarios. Last but not least, the discussion of challenges and future directions extends to the consideration of various potential issues within our definition. In summary, our contributions are as follows:

- **Formal Definition and Categorization:** We present a formal problem definition of learning to reduce the scale of large graphs and systematically categorize existing methods in a unified taxonomy, based on the aspect of information to be retained (graph-guided, model-guided, and hybrid) and the approaches to formulate the output small graphs (modification and synthetic approaches).
- **Comprehensive Analysis of Datasets and Evaluation Mechanisms:** We encompass an exhaustive cataloging and summarization of benchmark datasets to the advancement of this field, as well as a rigorous analysis of evaluation metrics and their applications. By doing so, our paper furnishes a robust framework for assessing the efficacy and applicability of existing methods across diverse scenarios.
- **Analysis of Limitations and Future Research Directions:** We delve into the limitations and challenges of existing methods from a broader perspective, presenting future directions, and thus we expect to inspire more innovations that address the challenges and push the boundaries of what is currently achievable.

In the following sections, we commence with a formal problem definition and the foundational workflow of GS within Section 2. Preliminary. To clarify the question of what principles should be followed when condensing a graph and how a scaled graph was formulated, Section 3. Optimization Objective delves into the optimization objectives that were used in the literature, and we propose to use the taxonomy of objectives as the primary taxonomy for GS methods. Subsequently, Section 4. Formulation of the Scaled Graphs provides a comprehensive summary of the various approaches to formulate scaled graphs, thereby offering insights for implementation and border applications. Section 5. Dataset and Evaluation presents an exhaustive overview of the commonly used datasets in GS, as well as the evaluation metrics deployed to ascertain the effectiveness and efficiency of GS methodologies. Section 6. Limitations and Challenges presents our discussion concerning the limitations and challenges that currently beleaguer established GS methods, and Section 7. Conclusion and Future Directions concludes this paper with an outlook on the future directions.

2 PRELIMINARY

2.1 Notations and Definition

For any matrix, the symbol $\top, -, +$ represents the operations of transpose, inverse, and pseudoinverse, respectively. On top of anything, a hat symbol $\hat{}$ indicates its optimal version. $\mathcal{S} = \{\mathcal{G}_1, \cdots, \mathcal{G}_\eta\}$ denote a dataset of η graph(s), where $\eta \in N^+$, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X}\}$, $\mathcal{V}$ and $\mathcal{E}$ denotes the set of vertices (nodes) and edges, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, and $\mathbf{X} \in \mathbb{R}^{N \times d}$ represents the feature matrix. $\mathbf{L}_\mathcal{G}$ is the corresponding Laplacian matrix of $\mathcal{G}$. $\mathcal{Y} \in \{1, \cdots, C\}^M$ denotes the set of labels of the nodes or edges if $\eta = 1$, or the label of graphs if $\eta > 1$, $\mathbf{Y} = [\mathcal{Y}] \in \mathbb{N}^{\mathbf{M} \times \mathbf{1}}$ be its vector form. The downstream tasks are specifically defined by $M = N$ for node

classification, $M = N^2$ for link prediction, and $M = \eta$ for graph classification. $f(;\theta) : I \rightarrow O$ indicates a function parameterized by θ, I and O represents the corresponding input and output spaces, e.g., $\text{GNN}(;\theta_{\mathcal{S}}) : \mathcal{S} \rightarrow \mathcal{Y}$ denotes the GNN parameterized with $\theta_{\mathcal{S}}$ was trained on dataset $\{\mathcal{S}, \mathcal{Y}\}$.

2.2 Problem Definition

Denote $\mathcal{S}_s = \{\mathcal{G}'_1, \cdots, \mathcal{G}'_\mu\}$ a smaller dataset of μ graph(s), $\mu \leq \eta$, and $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}', \mathbf{A}', \mathbf{X}'\}$ where the first dimension of $\mathcal{V}', \mathbf{A}'$, and $\mathbf{X}'$ are N' ($N' \ll N$), $\mathcal{Y}' \in \{1, \cdots, C\}^{M'}$ be the labels of $\mathcal{G}'$. Existing definitions of graph condensation, reduction, and distillation follow the objective optimization formulations below:

$$\hat{\mathcal{S}}_s = \arg\min_{\mathcal{S}_s} \mathcal{L}(\mathcal{S}_s, \mathcal{S}) \tag{1}$$

where $\mathcal{L}$ measures the gap of key information between the original and the scaled graph. In the meantime, this optimization formulation also fits the problem of coarsening:

$$\hat{\mathcal{S}}_s = f(\mathcal{S}; \hat{\mathbf{P}}), \text{ s.t. } \hat{\mathbf{P}} = \arg\min_{\mathbf{P}} \mathcal{L}(\mathcal{S}_s, \mathcal{S}) \tag{2}$$

where $\mathbf{P}$ is typically a projection matrix, encoding the rules of node aggregation. The distinction lies in whether the scaled graph is directly optimized as parameters or indirectly derived through the optimization of other parameters. Based on this observation, and to systematically address the two key questions, i.e., what information to retain and how to generate the scaled graph, we propose a general formulation as follows:

$$\hat{\Omega} = \arg\min_{\Omega} \mathcal{O}\left(\phi\left(\mathcal{S}_s\right), \phi(\mathcal{S})\right), \text{ s.t. } \mathcal{S}_s = f(\mathcal{S}; \Omega) \tag{3}$$

- Objective $\mathcal{O}$ describes the loss of graph information, which is quantified by a function ϕ;
- The intermediate representation Ω is optimized by minimizing the objective $\mathcal{O}$.
- The formulation function $f(;\Omega)$ describes how to generate the scaled graphs, which is parameterized by Ω;

Under this definition, we divide the graph scaling process into two steps: the optimization of the intermediate variable (i.e., Ω), and the formulation of the scaled graph based on that. This intermediate variable can also be any part of the complete scaled graph, thus allowing for compatibility with direct optimization formulations. Moreover, one key characteristic that distinguishes our scope, i.e., learning to generate small and informative graphs, from sampling methods is that part of or the entire scaled graph is **newly generated**. Specifically, we call:

$$\begin{aligned} &\mathcal{S}_s \text{ as the Scaled graphs only if } \mathcal{S}_s \notin \mathcal{S}. \\ &\mathcal{G}', \text{ or } \mathcal{G}_s, \text{ as the Scaled graph only if } \mathcal{V}' \notin \mathcal{V}. \end{aligned} \tag{4}$$

In this paper, we will focus on methods that were designed to reduce the scale of a single graph, i.e., $\mathcal{S} = \mathcal{G} = \{\mathbf{A}, \mathbf{X}, \mathcal{Y}\}; \mathcal{S}_s = \mathcal{G}_s = \mathcal{G}' = \{\mathbf{A}', \mathbf{X}', \mathcal{Y}'\}$, while methods for multiple will be briefly introduced. The three steps for formulating scaled graphs correspond to the three steps in the GS workflow, as illustrated in fig. 1. Under our definition, specifying what information to preserve, denoted as ϕ, is crucial as the primary motivation is to reduce the scale of graph data while preserving sufficient information. The details of the optimization objectives be seen in section 3, and the formulations will be presented in section 4. We categorize current methods by the taxonomy of objectives and present the formulations of each in table 2.

3 OPTIMIZATION OBJECTIVE

The motivation of GS is that an objective quantizing the loss of information between the original and the scaled graph should be minimized during the process. Naturally, the specific perspective of information GS aims to preserve can serve as a taxonomy for categorizing GS methods at the motivational level. We categorize the

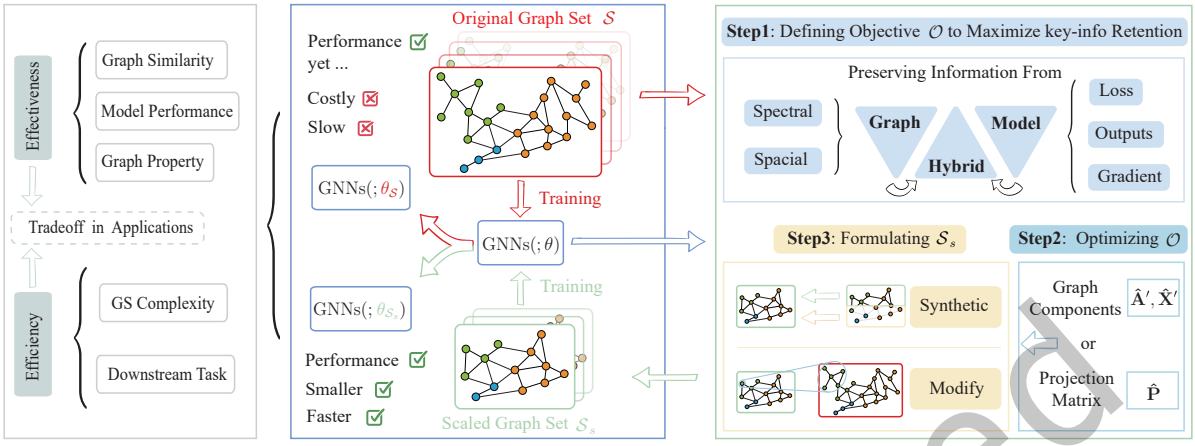


Fig. 1. Overview of GS. GNNs stand for any graph machine learning model with different architectures like GCN, GAT, ..., etc.

optimization objectives into three types: preserving certain properties of the graph (**graph guided**), retaining the GNNs' capabilities for downstream tasks (**model guided**), and simultaneously accomplishing both (**hybrid**).

3.1 Graph Property Guided Methods

These objectives can be formulated as: Get similar & smaller graph $\mathcal{G}_s$ from original graph $\mathcal{G}$, and the key is defining what are **graph properties** and how to **evaluate the similarity** between two graphs based on these properties:

$$\min_{\mathcal{G}_s} Sim(\phi(\mathcal{G}_s), \phi(\mathcal{G})), \tag{5}$$

where $Sim()$ is a similarity metric, ϕ is a function that quantifies graph properties. Together, they formulate a similarity metric over two graphs. We categorize graph property guided objectives into two categories: **Spectral** and **Spacial**, by the domain of extracted graph information. According to the taxonomy of GNNs [6, 24], the spectral domain of a graph refers to the information transformed using the graph Laplacian as a filter (or the Laplacian itself); the spatial domain of a graph can be understood more straightforwardly as the adjacency relationships between nodes; Spectral and Spatial GNNs are neural networks defined and operated on these two domains. Similarly, the graph information we aim to extract can also be categorized according to the spatial and spectral domains. Specifically:

3.1.1 Spectral Property Guided Methods. The Spectral GNNs [25] are defined by operators in the Spectral domain. Similarly, we define the *Spectral Properties* of a graph by requiring the graph Laplacian for calculations, i.e., $\phi(\mathcal{G}) = \phi(\mathbf{L}_{\mathcal{G}})$. In this case, the objective becomes minimizing the **D**istance of two graphs in the **Spe**ctral domain (**DSpe**), i.e., $O = DSpe(\phi(\mathbf{L}_{\mathcal{G}_s}), \phi(\mathbf{L}_{\mathcal{G}}))$.

The direct use of Laplacian eigenvalue and eigenvectors can be seen in this domain, as well as the Laplacian Energy Distribution (LED) used in SGDD [26] and the pseudoinverse of Laplacian matrix in ReduceG [27]. However, the difference in graph scale before and after GS may need cross-dimension metrics, such as having a distinct number of eigenvalues and eigenvectors. Specifically, GR [28] and SCAL [29] calculate the differences of smallest k eigenvalues and corresponding eigenvectors between two graphs' Laplacian; GDEM [30] take k

eigenvectors with the smallest eigenvalues to map node features, and minimize the distances of class centers in the spectral space. This is because the smaller the eigenvalue is, the more informative it and its corresponding eigenvector are on graph laplacian [31]. We will further discuss the information within eigenvectors later in section 3.1.3. Similar action that integrates k eigenvectors with the smallest eigenvalue to extract graph properties in the spectral domain is also known as 'Eigenbasis Matching', as in BiMSGC [32] and CTGC [33]. SGC [34] Further proposes graph lifting to rescale the small graph to a large one, and thus comparing all eigenvalues for a comprehensive spectral distance metric.

Despite direct optimization of spectral properties, SGC [34] and GraphZoom [35] identify similar and connected nodes to *Aggregate* in spectral embedding space, which can be seen as indirectly maximizing the spectral similarity of two graphs. Notably, maximizing *Graph Similarity* metrics based on spectral-GNNs, e.g. in CaT [13], is also considered as spectral property guided objectives with $\phi(\mathcal{G}) = GNN(\mathcal{G})$, and similarly, the use of spatial-GNNs for similarity metrics applies to the subsequent definition of Spacial ones. The GNNs we discussed here were not trained on downstream tasks and only served as information extractors.

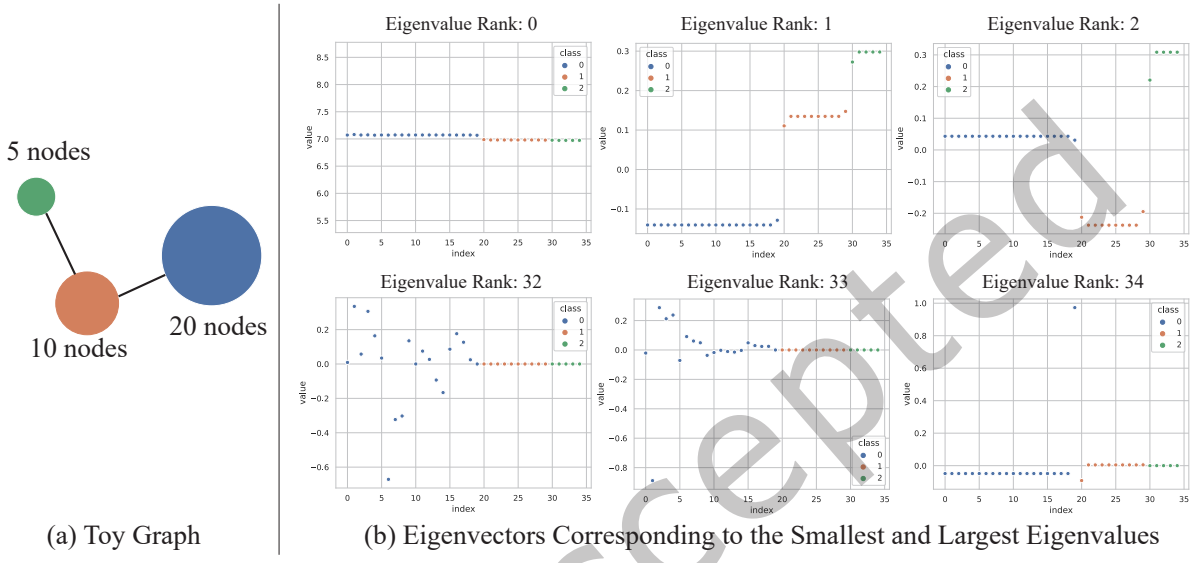
3.1.2 Spacial Property Guided Methods. The spatial domain of a graph is essentially the original topology and node features, i.e., $\phi(\mathcal{G}) = \phi(\{\mathbf{A}, \mathbf{X}\})$. Objective becomes minimizing the **D**istance of two graphs in the **Spa**cial domain (**DSpa**), i.e., $O = DSpa(\phi(\{\mathbf{A}', \mathbf{X}'\}), \phi(\{\mathbf{A}, \mathbf{X}\}))$. Specifically:

Graph Statistic properties ($\phi(\mathcal{G}) = Statistics(\mathbf{A})$) like graph density, average degree and degree variance employed in [36], and feature homophily ($\phi(\mathcal{G}) = Statistics(\{\mathbf{A}, \mathbf{X}\})$) in FGC [37]. Moreover, Structural Equivalence Matching (SEM) and Normalized Heavy Edge Matching (NHEM) used in MILE [38] can identify *Topologically Redundant* nodes. Selecting nodes by *Ranking* them through well-designed score functions can be effective in many scenarios such as FreeHGC [39] for heterogeneous graph condensation and Clnode [40] for selective curriculum training. A special class of *Reconstruction* objectives is also considered in this category because a successful reconstruction can be seen as a successful preservation of graph information. In this case, $O = \mathcal{L}_{reconstruct}(\mathcal{G}_s, \mathcal{G})$. For example, reconstructing the node features is used in OTC [41], and FGC [37], and metrics on reconstructing the whole graph in MCond [42].

Furthermore, the spatial *Relationships* between instances in the original graph can also provide valuable guidance. For example, clustering characteristics exhibited by instances in the original feature space (dense within clusters, sparse between clusters) should also be preserved, e.g., FGD [43] leverages the coherence between instances with the same sensitive attribute before and after graph condensation as a guiding objective, and CTRL [44] and HGCond [45] use the cluster centers on the original features as initialization of $\mathbf{X}'$. GCSR [46] uses the class-wise correlation matrix to regularize the optimization of $\mathbf{A}'$. Utilizing the relationships in the original dataset, identifying k nearest neighbors and aggregating them is also a popular strategy as in [34, 35, 47]. The distinction between different methods lies in the definition of 'neighbors', such as the topological neighborhood or the nearest nodes in feature spaces, or other predefined 'equivalence relationships' under certain metrics in G-Skeleton [47]. Thus, we conclude these aggregating strategies as (*Equivalence Aggregate*).

3.1.3 Analysis of Graph guided objectives. Unlike spatial methods, which are more intuitive, spectral methods rely on the information encoded in the eigenvalues and eigenvectors of the graph Laplacian, which can be difficult to understand. To address this, we use the theory of graph cut [48] and a toy example to illustrate the information encoded in the graph Laplacian. Assume there is a graph containing 3 fully connected communities (class of nodes) of size 20, 10, and 5, while only two cross-community edges exist, as in fig. 2. (a). We visualized the Laplacian eigenvectors corresponding to the smallest and largest eigenvalues in fig. 2. (b). In the theory of graph cut, the Laplacian eigenvalue (other than the smallest one) can be interpreted as the loss value to the relaxed normalized graph cut objective, while the signs of the values at corresponding indices in the Laplacian eigenvectors indicate the results of the graph cut (i.e., bi-partition labels). As observed, the eigenvectors corresponding to

smaller eigenvalues represent the most optimal partitions of the graph, encoding the macro topology between communities. In the meantime, a bad graph cut could happen within dense communities, thus the eigenvectors corresponding to larger eigenvalues may encode the microstructure within communities. This interpretation helps in better understanding how Laplacian eigenvalues and eigenvectors encode the structural information of a graph in the spectral domain.



(a) Toy Graph (b) Eigenvectors Corresponding to the Smallest and Largest Eigenvalues

Fig. 2. Visualization of a toy graph and corresponding Laplacian eigenvectors.

3.2 Model Capability Guided Methods

Since one of the ultimate motivations of SC is to achieve comparable performance via training models (including but not limited to GNNs) on smaller graphs $\mathcal{G}_s$, the bi-level optimization problem of SC is formulated as:

$$\begin{aligned} &\min_{\mathcal{G}_s} \mathcal{L}\left(\text{Model}(\mathcal{G};\boldsymbol{\theta}_{\mathcal{G}_s}),\mathcal{Y}\right) \\ \text{s.t} \quad &\boldsymbol{\theta}_{\mathcal{G}_s} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\text{Model}\left(\mathcal{G}_s;\boldsymbol{\theta}\right),\mathcal{Y}'\right), \end{aligned} \tag{6}$$

where $\mathcal{L}$ is the task-specific loss (performance) function, e.g., the Cross-Entropy loss in classification tasks. By expecting models to achieve **comparable performances** as those trained on the original dataset $\mathcal{G}$ through training on the scaled ones $\mathcal{G}_s$, the **models trained on the original graphs** $\mathcal{G}$ can be useful. The optimization objective of model-guided methods is written as follows:

$$\begin{aligned} &\min_{\mathcal{G}_s} D(\phi(\text{Model}(\mathcal{G}_s;\boldsymbol{\theta}_{\mathcal{G}_s})),\phi(\text{Model}(\mathcal{G};\boldsymbol{\theta}_{\mathcal{G}}))) \\ \text{s.t.} \quad &\boldsymbol{\theta}_{\mathcal{G}_s} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\text{Model}\left(\mathcal{G}_s;\boldsymbol{\theta}\right),\mathcal{Y}'\right), \boldsymbol{\theta}_{\mathcal{G}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\text{Model}\left(\mathcal{G};\boldsymbol{\theta}\right),\mathcal{Y}\right) \end{aligned} \tag{7}$$

In this case, ϕ quantifies the capability of the model, and D is a distance function. We categorize all objectives that utilize such trained model as input as model-guided objectives. Specifically:

3.2.1 Model Parameters. Starting from GCond [49], utilizing the training *Gradients* and *Trajectories* of model parameters as key information to be preserved in the scaled graph has been widely adopted in practice.

$$\min_{\mathcal{G}_s} \begin{cases} D(\nabla_{\theta_{\mathcal{G}_s}} \mathcal{L}, \nabla_{\theta_{\mathcal{G}}} \mathcal{L}) & \text{for Gradient Matching [44, 45, 49–59]} \\ D([\theta_{\mathcal{G}_s}], [\hat{\theta}_{\mathcal{G}}]) & \text{for Trajectory Matching [60, 61]} \end{cases} \tag{8}$$

$[\theta_{\mathcal{G}_s}]$ indicate the list of parameters ordered by training steps. Specifically, in models trained on the original graph, the gradients of parameters $\nabla_{\theta_{\mathcal{G}}}$ between epochs (i.e., training gradient) or the parameters themselves $\theta_{\mathcal{G}}$ at each epoch (i.e., training trajectory) essentially represent the knowledge the model has learned from the original graph for the given task $\mathcal{L}$. The core idea of such methods is to align the parameters of the model trained on the original graph and the scaled graph as the optimization objective, using backpropagation to directly update the scaled graph, as formulated in eq. (8). By doing so, the scaled graph is optimized to reproduce the capability (parameters) of the model trained on the original graph, maximizing the preservation of the models' performances on specific tasks.

Following GCond, many methods have been devoted to improving the effectiveness and efficiency of the gradient matching objective. Specifically, HGCond [45] proposes Orthogonal Parameter Sequence, GroC [53] proposes adversarial training, GCARe [55] includes additional regularization of the condensation GNN, and CTRL [44] enhance the distance metric D, in order to enhance the optimization of the gradient matching objective; DosCond [51] propose to match the gradient of the first training step only, EXGC [54] inject the Mean-Field approximation to GCond paradigm and selectively update scaled node features via scoring methods, TinyGraph [56] adopt curriculum gradient matching strategy, to improve the efficiency of the optimization of gradient matching objectives. Instead of matching gradients at each step, trajectory matching methods like GEOM [60] and SFGC [61] record the parameter trajectories across all steps (GCSR [46] for multi-steps) on the original graph and align them with those on the scaled graph. Since gradients at each step and trajectories across steps essentially encode the same information, i.e., the training dynamics of model parameters on the original graph, we categorize them together as *Parameter Matching* objectives.

Analysis on parameter matching methods. Originally proposed in DC [62] for image dataset distillation, gradient matching has been proven effective for graphs as well. However, unlike vision models whose capacity increases with the depth of stacked network layers, the most popular GNN models in graph data mining do not share this characteristic. In other words, the parameter capacity of vision models is theoretically unlimited, allowing them to adapt to datasets of any size. In contrast, GNNs typically have a fixed parameter capacity. *Does this imply that methods that take GNNs as the base model have an upper limit on the size of graph datasets they can handle?* According to recent benchmarks, the largest graph dataset applied in practice is Reddit (232,965 nodes) in GC-Bench [63] and GC4NC [64], and ogbn-products (2,449,029 nodes) in GCondenser [65], while the parameter space of GNN is much smaller than the scale of these datasets. In such cases, spectral alignment methods that rely on eigen decomposition may become impractical due to their exploding time complexity. In contrast, for the representative parameter matching method, GCond, the performance on the scaled graph divided by the original for the node classification task is 94% on Reddit and 93% on ogbn-products. While the retention of model performance is not perfect, these results can still be considered effective. This may be attributed to the unique characteristics of graph data, where local patterns tend to repeat throughout the global structure. We will discuss the imperfect performance retention later in section 6.

3.2.2 Loss Value. As the loss function $\mathcal{L}$ essentially quantifies the quality of model parameters, KiDD [66] writes the optimization objective as loss-matching objectives:

$$\min_{\mathcal{G}_s} \left| \mathcal{L}\left(\text{Model}\left(\mathcal{G}; \hat{\boldsymbol{\theta}}_{\mathcal{G}_s}\right), \mathcal{Y}\right) - \mathcal{L}\left(\text{Model}\left(\mathcal{G}; \hat{\boldsymbol{\theta}}_{\mathcal{G}}\right), \mathcal{Y}\right) \right| \tag{9}$$

Since the second term $\mathcal{L}\left(\text{Model}\left(\mathcal{G};\hat{\boldsymbol{\theta}}_{\mathcal{G}}\right),\mathcal{Y}\right)$ represent the training result of the original graph on the task, its value is a fixed constant (and minimized), so the optimization objective is essentially equivalent to eq. (6) previously defined:

$$\min_{\mathcal{G}_s}\mathcal{L}\left(\text{Model}\left(\mathcal{G};\hat{\boldsymbol{\theta}}_{\mathcal{G}_s}\right),\mathcal{Y}\right) \tag{10}$$

Specifically, KiDD [66] solves this problem by introducing Kernel Ridge Regression for a closed-form solution, while GC-SNTK [67] and SGDC [68] further introduces Structure-based and Attention-based Neural Tangent Kernel into this paradigm respectively, and OpenGC [69] substitute the regression loss with cross-entropy loss. FedGKD [70], as well as DisCo [71] and TCGU [72], further modifies the optimization of the scaled graph from evaluating the original graph to the scaled graph itself, using the parameter optimized in the original graph, i.e. $\min_{\mathcal{G}_s}\mathcal{L}\left(\text{Model}\left(\mathcal{G}_s;\hat{\boldsymbol{\theta}}_{\mathcal{G}}\right),\mathcal{Y}'\right)$.

3.2.3 Embedding and Logits. The outputs of a trained network typically integrate crucial information for downstream tasks, and are thus considered informative. Matching models' output *Embeddings* of training instances is used in [13, 73, 74], and *Predicted Logits* based uncertainty metric was used in [30, 36], and can be formulated as:

$$\min_{\mathcal{G}_s} D\left(\text{Model}\left(\mathcal{G};\hat{\boldsymbol{\theta}}_{\mathcal{G}_s}\right),\text{Model}\left(\mathcal{G};\hat{\boldsymbol{\theta}}_{\mathcal{G}}\right)\right) \tag{11}$$

Specifically, DisCo [71] and GCDM [73] align class centroids, i.e., mean of node embeddings for each class; SimGC [75] and TCGU [72] align the distribution parameters of embeddings (mean and standard deviation in SimGC, mean and covariance in TCGU) with the class distribution of predicted logits between the scaled and original graphs. ConvMatch [74] learns a mapping matrix to expand the number of scaled node embeddings from N' to N and directly aligns the embeddings between the scaled and original graphs. CGC [76] proposes that the scaled node feature is obtained by partitioning the embeddings (propagated original features) by enhanced labels, i.e., the class centers of the original node features become the scaled graph node features. In a self-supervised setting, CTGC [33] first generates the scaled graph structure from the clustering centroids of the original node structural embeddings, then obtains the scaled graph node features by matching embeddings to the original node semantic centroids. As well as embeddings, the class-wise prediction logits (essentially c dimensional normalized embeddings, c is the class number) can also be informative and have been adopted as an alignment constraint in GDEM [30].

3.2.4 Analysis on Model Guided Objectives. As illustrated in fig. 3, the ultimate goal of these objectives is for the model trained on the scaled graph to replicate the model's performance trained on the original graph. We raise the following questions: (1) Does this imply that the performance of the model trained on the scaled graph has an upper bound, specifically by the model trained on the original graph? (2) Does these methods experience significant challenges in generalization ability when applied to different downstream tasks or models? (3) If training on the original graph is inevitable, will the efficiency gains from GS be overshadowed by the complexity of the GS process itself?

For the first question, according to the extensive experiment results reported in recent benchmarks, i.e., GC-Bench [63], GCondenser [65] and GC4NC [64], it can be observed (especially highlighted in Table.2 in GC-Bench) that many models guided SC methods produce scaled graphs that outperform the baseline models trained on the original graphs on various datasets. This phenomenon can be interpreted in two directions: (1) The scaled graph retains key information while removing noise, improving the dataset's quality; (2) The improved performance is simply due to overfitting. The second interpretation raises the same concern as the second question, namely that the generalization of the scaled graph may be poor. According to the benchmark results (especially highlighted by GC4NC), while different methods perform differently across datasets and architectures, models trained on

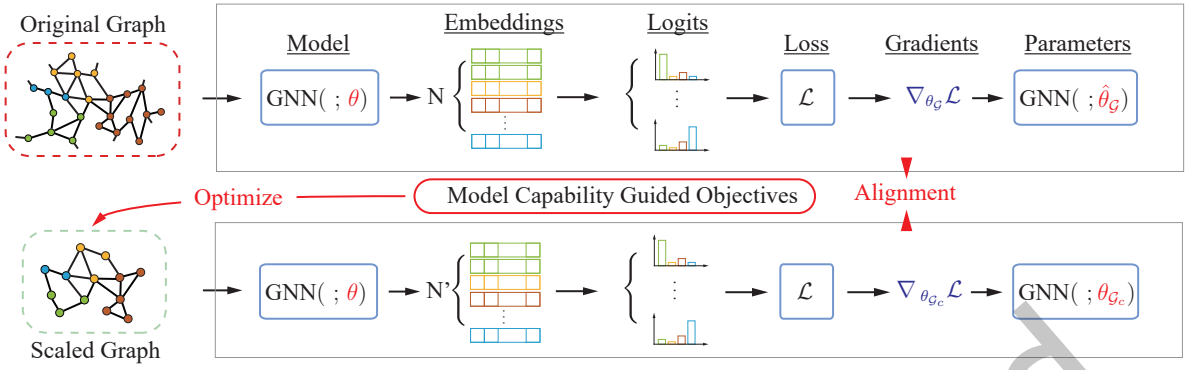


Fig. 3. Overview of Model Guided Objectives.

scaled graphs still exhibit significant performance variance when applied to different downstream architectures and tasks. This indicates that there is still considerable room for improvement in the generalization ability of scaled graphs.

For the third question, as highlighted in GCondenser, the best efficiency with effectiveness is achieved by SFGC [61], a trajectory-matching objective method with the optimization of the scaled graph structure is completely abandoned, while gradient matching methods like GCond [49] exhibit the worst efficiency. However, the efficiency of certain methods in practical application scenarios needs to be analyzed in conjunction with downstream applications. We will further discuss the efficiency concerns later in section 6.2.

3.3 Hybrid Methods

It is worth mentioning that the aforementioned two types of objectives, namely graph-guided and model-guided, are not mutually conflicting. Therefore, the third category named hybrid methods combines both the graph properties and model capabilities as guidance for SC simultaneously. There are methods such as [26, 30, 36, 42, 43] that optimize the scaled graph from both graph-guided and model-guided objectives. Specifically, SGDD [26] simultaneously match the train trajectory between two models and the Laplacian Energy Distribution between two graphs; Mcond [42] optimize the gradient matching loss and reconstruction loss together; FGD [43] use the relation coherence as an additional constraint to the gradient matching loss; GCSR [46] aligns the training trajectory while using the relation information from the original to regularize the scaled graph. Both GDEM [30], CTGC [33], and BiMSGC [32] align the spectral properties (eigenbasis in GDEM and BiMSGC, structural embeddings in CTGC), while BiMSGC further incorporates gradient matching and GDEM introduces logits (i.e., the category-level representations) alignment, CTGC introduces semantic embedding alignment, between the original and the scaled graph. Furthermore, the combination of model and graph properties can be comprehensive metrics as [36] takes the model predicted uncertainty and empirically verified useful graph properties to rank graph training instances for node selection.

3.4 Comparison of Objectives

Three types of objectives, namely graph-guided, model-guided, and hybrid, each with its advantages and drawbacks: To produce 'similar' scaled graphs, **graph-guided** objectives focus on preserving the properties of the original graph. This is suitable for applications that require retaining the patterns from original graphs. On the other hand, the **model-guided** objectives aim to maintain the performance of the model by optimizing the scaled graph. These methods are driven by motivation-oriented optimization and thus perform exceptionally well in

predefined scenarios. However, it may result in overfitting, reducing the adaptability of scaled graphs to other models or tasks. **Hybrid** methods combine the advantages of both graph-guided and model-guided approaches, intending to retain model performance while preserving graph properties for scenarios that value both graph property and model performance. However, balancing between the two objectives as well as optimizing them can be challenging.

For model-guided methods, the performance on downstream tasks is naturally ensured. However, for graph-guided methods, since the scaled graph is not tailored to the downstream task, might it fail to achieve optimal performance in those applications? We select FreeHGC [39] and HGCond [45] for comparison, where the former does not rely on knowledge from a pre-trained model on the original graph but instead uses graph-level information to guide graph scaling, while the latter depends on a gradient matching objective. According to the results reported in FreeHGC and HGCond, both methods produce scaled graphs that achieve solid performance when training on downstream tasks. Meanwhile, FreeHGC demonstrates significantly higher efficiency compared to HGCond, as it does not require training on the original graph or performing gradient matching. This indicates that even without incorporating model-specific knowledge, the performance of scaled graphs on downstream tasks may not necessarily be inferior. However, designing appropriate graph properties to preserve requires considerable domain expertise. In contrast, model-guided objectives are more versatile, as they can be applied as long as the models can be trained.

In conclusion, the choice of the appropriate objective depends on the specific requirements of the application. Graph-guided is more suitable for tasks emphasizing graph structure, model-guided applies to scenarios emphasizing model performance, and the hybrid method seeks a balance between the two. Considering the goals of the task and the characteristics of the graph, selecting the most suitable method requires careful consideration in practical applications.

4 FORMULATION OF THE SCALED GRAPHS

Here comes the question: how to formulate each component of the scaled graph $\mathcal{G}_s$? Since the scaled graphs $\{\mathcal{G}_s, \mathcal{Y}'\} = \{\{\mathbf{A}', \mathbf{X}'\}, \mathcal{Y}'\}$, therefore, $f(\mathcal{G})$ pertains to formulating these three components. We write: $\mathbf{A}' = f_{\mathbf{A}}(\mathcal{G}; \mathbf{W})$, $\mathbf{X}' = f_{\mathbf{X}}(\mathcal{G}; \mathbf{W})$, and $\mathcal{Y}' = f_{\mathcal{Y}}(\mathcal{G}; \mathbf{W})$ to formulate each. As we conclude, there are two main classes: the **Modification** and the **Synthetic** formulation. In this section, we will delve into the taxonomy and various approaches for the formulation of a single scaled graph, and based on the foundation established by the single graph formulation, we will briefly introduce the strategies to handle multi-graph datasets.

4.1 Modification formulation

Modification approaches encompass actions such as node aggregation and deletion, etc., where the scaled graph is the product of modifying the original graph. This category of formulations can be uniformly formalized as aggregating nodes from $\mathcal{G}$ to $\mathcal{G}_s$. Assuming each node $v'_i \in V'$ is aggregated from k nodes in $\mathcal{G}$, $k \in \mathbb{N}$, then the most common scheme, e.g., [28, 29, 34, 35, 37, 41, 42, 74] did, was:

$$\begin{aligned} f_{\mathbf{A}}(\mathcal{G}; \mathbf{P}) &= \mathbf{P}^{\mathbf{T}}\mathbf{A}\mathbf{P}\,, \;\; f_{\mathbf{X}}(\mathcal{G}; \mathbf{P}) = \mathbf{P}^{+}\mathbf{X}, \\ f_{\mathcal{Y}}(\mathcal{G}; \mathbf{P}) &= \arg\max \mathbf{P}^{+}[\mathcal{Y}] \end{aligned} \tag{12}$$

$\mathbf{P} \in \mathbb{R}^{\mathbf{N} \times \mathbf{N}'}$ is defined as a projection matrix, indicating that nodes $\mathcal{V}_{(i)}$ in $\mathcal{G}$ were aggregated to a new node v'_i in $\mathcal{G}_s$:

$$\mathbf{P}_{i,j} = \begin{cases} 1 & \text{if } v'_j \in \mathcal{V}_{(i)} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

In a general definition, each row of the projection matrix $\mathbf{P}$ may contain an uncertain number of nonzero entries, ranging from none (the node is considered dropped) to one (the node is being selected or aggregated

once) and even multiple (some node will be aggregated for multiple times). For example, in FreeHGC, based on selecting a subset of nodes, synthesized hyper-nodes using the mean of the original nodes are then reconnected to the selected nodes following the original connection pattern. Meanwhile, the formulation of a general multi-entry projection matrix can accommodate this scenario. The rules for modifying the original graph are either a reflection of the aforementioned optimization objective or the result of optimizing those objectives, where the specific design of node selection and dropping (e.g., FreeHGC [39], OTC [41]), and aggregation (e.g., GR [28], SGC [34], ReduceG [27], GraphZoom [35], G-Skeleton [47], MILE [38], FGC [37], OTC [41], ConvMatch [74]) is covered in the section 3.1.

The advantages of the Modify strategies are: (1) the scaled graphs are the modification of the real graph, and are thus highly interpretable; (2) The computational cost is relatively low, involving only a few matrix multiplications. Once the construction rule or optimization model for the projection matrix is established, the graph scaling process can become highly efficient. However, simple sum aggregation of nodes' topology might lead to a sharp increase in edge density in the scaled graph, thus some methods further gave sparsity constraint to matrix $\mathbf{P}$ to sparsify the scaled graphs, e.g., [41]. Moreover, since the construction of rules of graph modification relies on domain expertise, when the information contained in the graph follows entirely different patterns, the original rules are likely to become ineffective.

4.2 Synthetic formulation

On the other hand, a key characteristic of synthetic approaches is that the nodes and edges in the scaled graph may be newly generated. In terms of results, the scaled graphs can be viewed as outcomes of optimizing specific objective functions. However, due to the unique nature of graphs—comprising node features, topology, and labels—the optimization processes and strategies vary across different methods. We categorize the synthetic formulations based on the optimization strategies applied to different components of the scaled graph, distinguishing them into three processes: Predefined, Joint Optimization, and Sequential Optimization. Specifically:

4.2.1 Predefined. This kind of strategy is undoubtedly the most straightforward yet most tricky one. Two popular strategies are used in the literature: predefine $\mathbf{A}' = \mathbf{I}$ ($\mathbf{I}$ is the identity matrix) as in SFGC [61], OpenGC [69], and CaT [13]; and predefine $\mathcal{Y}' = Sample(\mathcal{Y})$. The former can be interpreted as the goal of GS being solely to learn the prototype embeddings for each class, at which point the topology information has already been integrated and is no longer necessary. The latter can be explained as achieving the same label distribution between the graph before and after GS by employing a uniform sampling of labels. The tricky initialization of parameters (including the initialization of $\mathcal{G}_s$) to optimize is also included as Predefined strategies, which will not be discussed further here.

4.2.2 Joint Optimization. Methods in this category, e.g., GC-SNTK [67], KiDD [66], SGDC [68], DosCond [51], HCDC [52], GEOM [60], BiMSGC [32] and GDEM [30], are the most simple yet the most challenging ones, where the scaled graph (topology $\mathbf{A}'$ and node features $\mathbf{X}'$) is considered as parameters for the objective, and the node labels $\mathcal{Y}'$ were often predefined by sampling the original labels $\mathcal{Y}$ (while BGC [50] optimize the node labels as well as topology and features). Meanwhile, methods represented by SFGC [61] completely discard the topology and focus solely on optimizing node features, and we also classify these approaches as joint optimization. In conclusion, the most popular formulation of joint optimization of the scaled graph is given by:

$$f_{\mathbf{A}}(;\mathbf{A}') = \mathbf{A}' \ , \ f_{\mathbf{X}}(;\mathbf{X}') = \mathbf{X}' \ , \ f_{\mathcal{Y}}(\mathcal{Y};) \subseteq \mathcal{Y} \tag{14}$$

In this scenario, $\{\mathbf{A}', \mathbf{X}'\} = \arg\min_{\{\mathbf{A}', \mathbf{X}'\}} O$ are treated as parameters to be optimized. The predefined strategy of node labels, e.g., uniform sampling, can keep label distribution unchanged. Therefore, this strategy can be perceived as generating dual features for each class: node features and their topological connection.

4.2.3 Sequential Optimization. The existence of this strategy is typically regarded as a compromise in the challenge of joint optimization: if the complete scaled graph, encompassing both matrix $\mathbf{A}'$ and vector $\mathbf{X}'$, is regarded as optimization parameters, the dimensionality of the parameter space escalates significantly, introducing challenges to the convergence of optimization objective. Therefore, optimizing part of the scaled graph first, and constructing the rest parts to complete the GS process can be an efficient solution. Specifically, represented by GCond [49], a series of works (e.g., GCDM [73], DisCo [71], TCGU [72], SimGC[75], GroC [53], EXGC [54], GCARe [55], CTRL [44], HGCond [45], TinyGraph [56], FGC [57], RobGC [58], FGD [43]) directly optimize the scaled node features first and then use an MLP model to predict the connection relationships between nodes based on this intermediate result, while CGC [76], Mcond [42], GCSR [46], SGDD [26], and CTGC [33] involves careful designs of models or functions to generate graph topologies based on the intermediate optimization results. The formulation is given by:

$$f_{\mathbf{A}}(\mathbf{X}';\omega) = g(\mathbf{X}';\omega)\ ,\ f_{\mathbf{X}}(;\mathbf{X}') = \mathbf{X}'\ ,\ f_{\mathcal{Y}}(\mathcal{Y};) \subseteq \mathcal{Y} \tag{15}$$

Where $g()$ can be MLPs, etc., and in this case, $\{\mathbf{X}', \omega\} = \arg\min_{\{\mathbf{X}',\omega\}} O$. Among the existing literature, as the optimization of $\mathbf{X}'$ also needs predefined labels, this formulation can be interpreted as generating prototypes for each class first, and subsequently predicting their relationships (i.e., topology); or aggregate hypernodes first, and determining their labels; or construct topology first, optimizing the node feature and labels as FedGKD [70] did, hence possessing greater optimization efficiency, design flexibility, and interpretability of the scaled graphs (e.g., predicting connections based on node similarity).

4.3 Comparison of formulations

Each of the formulations mentioned has its distinct methods for generating $\mathbf{A}'$, $\mathbf{X}'$, $\mathcal{Y}'$, despite that the Sequential Optimization Formulation must rely on the intermediate results of the other formulations. As we conclude, the **Modification** formulations exhibit the strongest computational efficiency and interpretability, but their applicability is limited, as designing graph modifications requires substantial domain expertise. The **Synthetic by joint optimization** formulation is the simplest to conduct for its end-to-end design (defining the objective and optimizing directly), yet it is also the most challenging since the parameter search space may be so big that convergence is difficult. To address this issue, the **Synthetic by Sequential Optimization** is a 2-step scheme, incorporating the advantages of both easy-to-implement and objective-oriented. The **Synthetic by Predefined** formulation yields intuitive results and can be effective in carefully designed scenarios.

However, **Synthetic formulations** require directly optimizing the scaled graph, which is either randomly or carefully initialized, according to predefined optimization objectives. This process often lacks transparency, making it difficult to understand how the scaled graph maintains the properties of the original graph and why it performs well on certain tasks. This limitation affects the interpretability of the generated small graph and the GS process itself. In contrast, **Modification formulations** have clear and well-defined rules for editing and aggregating the original graph. For example, in chemical molecular graphs, fixed and recurrent functional groups are condensed into single nodes; in social networks, tightly connected communities with similar traits are aggregated into single nodes, etc. These predefined rules enhance the interpretability of both the small graph and the generation process, making it easier to understand how the scaled graph retains the characteristics of the original graphs.

Depending on the application scenario, the choice between synthesis and modifying methods should be based on downstream tasks and specific requirements. The table 2 presents the objective-based category of methods included in this survey, and strategies on how to formulate the scaled graphs.

Table 2. Taxonomy of surveyed methods

/	Method	Objective O	Formulation f	Implementation Detail			Evaluation	
				f_A	f_X	f_y	Graph	Model
Graph Guided	GR [28]	Spectral Properties	Modification	P^TAP	P^+X	-	✓	-
	SGC [34]	Spectral Properties	Modification	P^TAP	P^+X	-	✓	-
	ReduceG [27]	Spectral Properties	Modification	P^TAP	-	-	✓	-
	SCAL [29]	Spectral Properties	Modification	P^TAP	P^+X	$\arg\max(P^+Y)$	-	✓
	GraphZoom [35]	Equivalence Aggregate	Modification	P^TAP	$GNN(A'X)$	-	-	✓
	G-Skeleton [47]	Equivalence Aggregate	Modification	P^TAP	$f(P^+)X$	-	-	✓
	MILE [38]	Equivalence Aggregate	Modification	P^TAP	$f(P^+)X$	-	-	✓
	FGC [37]	Graph Statistics	Modification	P^TAP	P^+X	-	✓	✓
	CaT [13]	Graph Similarity	Synthetic	I	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	FreeHGC [39]	Ranking	M + S	P^TAP	P^+X	$Sample(Y)$	-	✓
	OTC [41]	Ranking + Reconstruct	M + S	P^TAP	$GNN(A'P^+X)$	-	-	✓
Model Guided	ConvMatch [74]	Embedding Similarity	Modification	P^TAP	P^+X	$\arg\max(P^+Y)$	-	✓
	GCDM [73]	Embedding Similarity	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	CGC [76]	Embedding Similarity	Synthetic	$f(X'X'^\top)$	$\arg\min_{X'} O$	$f(A, Y)$	-	✓
	FedGKD [70]	Loss Matching	Synthetic	$f(X')$	$\arg\min_{X',Y'} O$		✓	✓
	GC-SNTK [67]	Loss Matching	Synthetic	$\arg\min_{A',X',y'} O$			-	✓
	KiDD [66]	Loss Matching	Synthetic	$\arg\min_{A',X'} O$		$Sample(Y)$	-	✓
	SGDC [68]	Loss Matching	Synthetic	$\arg\min_{A',X'} O$		$Sample(Y)$	-	✓
	OpenGC [69]	Loss + Embedding	Synthetic	I	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	DisCo [71]	Loss + Embedding	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	TCGU [72]	Loss + Embedding	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	SimGC[75]	Embedding + Logits	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	BGC [50]	Gradient Matching	Synthetic	$\arg\min_{A',X',Y'} O$			-	✓
	DosCond [51]	Gradient Matching	Synthetic	$\arg\min_{A',X'} O$		$Sample(Y)$	-	✓
	HCDC [52]	Gradient Matching	Synthetic	$\arg\min_{A',X'} O$		$Sample(Y)$	-	✓
	GCond [49]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	GroC [53]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	EXGC [54]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	GCARe [55]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	CTRL [44]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	HGCond [45]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	TinyGraph [56]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	FGC [57]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	RobGC [58]	Gradient Matching	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	MSGC [59]	Gradient Matching	Synthetic	$MLP(A, X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	GEOM [60]	Ranking + Trajectory	Synthetic	$\arg\min_{A',X'} O$		$GNN(\mathcal{G}_s)$	-	✓
	SFGC [61]	Trajectory Matching	Synthetic	I	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
Hybrid	Mcond [42]	Reconstruct + Gradient	M + S	P^TAP	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	FGD [43]	Relation + Gradient	Synthetic	$MLP(X')$	$\arg\min_{X'} O$	$Sample(Y)$	-	✓
	GCSR [46]	Relation + Trajectory	Synthetic	$f(X'X'^\top, A, Y, Y')$	$\arg\min_{X'} O$	$Sample(Y)$	✓	✓
	SGDD [26]	Spectral + Gradient	Synthetic	$GEN(A, X', Y')$	$\arg\min_{X'} O$	$Sample(Y)$	✓	✓
	BiMSGC [32]	Spectral + Gradient	Synthetic	$\arg\min_{A',X'} O$		$Sample(Y)$	-	✓
	GDEM [30]	Spectral + Logits	Synthetic	$\arg\min_{A',X'} O$		$Sample(Y)$	✓	✓
	CTGC [33]	Spectral + Embedding	Synthetic	$\mathbf{I} - \mathbf{U}'\mathbf{\Lambda}'\mathbf{U}'^\top$	$\arg\min_{X'} O$	-	-	✓

Ps. 'M', 'S' stands for Modification and Synthetic respectively. The 'Objective' and the 'Formulations' columns respectively illustrate the specific designs of certain methods in our taxonomy. Implementation detail provides specific details of the formulation of scaled graphs. 'Evaluation' columns correspond to the taxonomy of objectives, and '✓' indicates that evaluation experiments have been conducted in a certain aspect.

4.4 Strategies: $\mathcal{S} \rightarrow \mathcal{S}_s$

After introducing the methods for scaling a single graph, we will now explore the strategies for multiple graphs. Methods in this category enable more efficient and effective graph matching learning in graph-level scenarios such as continual graph learning [77]. As this paper primarily focuses on single graph scaling, we will only provide a concise overview:

One-by-One strategy: If the number of graphs remains unchanged, i.e., every single graph is scaled independently, e.g., SGC [34], we categorize this scheme as the one-by-one strategy. Any method capable of scaling a single graph can be adapted to scale multiple graphs by employing this strategy. **Joint Optimization strategy:** Similar to the single-graph joint optimization, this strategy combines multiple scaled graphs as parameters of the optimization objective, e.g., DosCond [51]. By naming the number of graphs in a graph dataset to be 1, this category of methods would essentially degenerate into a single-graph optimization strategy. **Selecting strategy:** The core idea of this strategy is to rank every single graph by score functions and select the top-ranked ones, e.g., [36, 78], or select the most frequent components (specifically the computation trees in [79]) among each graph.

5 DATASET AND EVALUATION

5.1 Dataset Statistics

We systematically organize and summarize the datasets employed in the discussed methods, categorizing them into two primary types: datasets featuring a single large graph and those comprising multiple graphs. The former is typically utilized for tasks such as node classification and edge prediction, while the latter is employed in graph classification. We present key attributes of the datasets, encompassing details such as the number of nodes, number of edges, features, and classes, and the graph type (e.g., social network or molecular network) for datasets with a single large graph. Additionally, for datasets containing multiple subgraphs, we provide organization based on the number of subgraphs, average number of nodes, average number of edges, number of labels, and the type of graph. The detailed statistics of the datasets can be found in our online resources [1].

5.2 Evaluation Metrics

GS aims to create a significantly smaller graph dataset while preserving sufficient information, thus it is crucial to evaluate how much this information is retained. The evaluation of GS methods is challenging compared to the straightforward performance evaluation of traditional GNNs, mainly due to their involvement in multiple aspects. From a holistic perspective, we summarize the evaluation of the entire GS process into two aspects: effectiveness and efficiency. Effectiveness evaluates how well the scaled graphs retain the original information, while efficiency includes both the scaling process and downstream task efficiency. Details can be found below:

5.2.1 Metrics Over Effectiveness. From the perspective of input and output, GS methods take the original graphs as input and the scaled graphs as output. To verify that the scaled graphs are informative, the effectiveness of GS is evaluated through the following different aspects: (1) The **similarity** between the original and scaled graphs is assessed in domains such as spectral and spatial characteristics. (2) The **performance** of scaled graphs in downstream tasks, while closely mirrors evaluations in traditional GNNs, a comparable performance can be considered as successful preservation of valuable information for downstream tasks.

(3) **Properties** of the scaled graphs alone, e.g., applicability which involves integrating GS as a component within an existing system, with evaluation metrics aligning with target systems like Graph Embedding and Graph Continual Learning; The capabilities such as fairness, generalizability, etc. Specifically:

[1] https://github.com/Frostland12138/Awesome-Graph-Scaling

Similarity with Original Graphs. Evaluation of how well the scaled graphs replicate the spectral and spatial characteristics of the original graphs involves metrics such as the Proportion of Low-Frequency Nodes, the Mean of High-Frequency Areas, and various spectral domain metrics. The dissimilarity between recovered graph partitions and ground-truth structures is quantified using Normalized Mutual Information (NMI). Error metrics, such as Relative Eigenerror, Reconstruction Error, Dirichlet Energy, and Hyperbolic Error, also provide insights into the similarity evaluation.

Performance of Mining on Scaled Graphs. The evaluation of GS methods entails a diverse set of metrics tailored to specific downstream tasks. In node and graph classification, widely adopted measures include Accuracy (ACC), with the Mean Classification ACC and Standard Deviation providing a more robust assessment. Furthermore, widely adopted performance evaluation metrics, such as the area under the ROC curve (AUROC), the area under the precision-recall curve (AUPRC), and the F1-score, find extensive application across diverse downstream tasks. For the link prediction task, common metrics include Hits@50 and Mean Reciprocal Rank (MRR).

Independent Graphs Property Evaluation. The aspect of evaluating independent graph properties does not have a fixed form. It can be as simple as visualization or presenting statistical features of the scaled graphs, or as complex as evaluating transferability and robustness for them being used for training across models and tasks. The utilization of GS methods in various systems entails the consideration of diverse evaluation metrics. For bias measurement in GS [43, 55], fairness metrics, including Demographic Parity (also known as Statistical Parity) and Equal Opportunity, are employed.

Model-guided GS methods raise concerns about the generalizability of scaled graphs. These methods would be more inclined to use the specific graph neural network model used in the GS process, thus further attenuating the performance of the scaled graph on other tasks with different types of models. In other words, scaled graphs may struggle to generalize to various downstream GNNs and tasks. To address these concerns, it is often necessary to introduce additional constraints to ensure that the scaled graphs maintain the necessary transferability. However, detailing these constraints is beyond the scope of this discussion.

Analysis of effectiveness metrics. In summary, the effectiveness of GS methods can be evaluated through the above three aspects. The first two aspects (namely, graph similarity and model performance before and after GS) correspond precisely to our taxonomy of GS objectives: graph-level and model-level. It is worth noting that the objective design and evaluation aspects in some methods may not necessarily correspond. For example, some methods perform GS through graph-level objectives and evaluate through the performance of the trained models, and vice versa. We present the execution of each method in these two corresponding aspects of the objective taxonomy in table 2, the last two columns.

5.2.2 Metrics Over Efficiency. The fundamental motivation of GS is to facilitate graph mining tasks on large-scale original graphs with efficiency. Consequently, it is imperative to evaluate the computational resources saved by GS in the mining of scaled graphs. Meanwhile, although GS is a one-time effort, the process of GS itself should not take too many resources.

Within the GS methods, e.g. [49, 51], analyze the **computational complexity**, primarily emphasizing time complexity and, to a lesser extent, space complexity. With respect to various datasets and scale ratios (often expressed as a percentage), this approach entails directly measuring the time required for generating the scaled dataset, commonly referred to as **scaling time**. For a more in-depth analysis of this term, certain methods further subdivide the time, e.g., GDEM[30] proposes additional metrics: time spent on the pre-processing stage; KiDD[66] measures the time dedicated to forward and reverse gradient propagation based on varying batch sizes, etc. In addition, considering that the large size of original graph data can cause the Out of Memory (OOM) problems, some methods also measure the **memory usage** in this process.

The efficiency of training a new GNN in downstream tasks using scaled graphs is undoubtedly superior to that of original graphs, where the time and space saved by GS are empirically proportional to the scale ratios. However, there are still some researches [42, 67, 74] specifically focused on evaluating downstream task training time and memory usage. Notably, not all methods explicitly address the efficiency of GS, but the absence of it does not imply its impossibility to conduct such [29, 34]. Considering both efficiency and effectiveness, we will introduce a tradeoff consideration in the following chapters.

6 LIMITATIONS AND CHALLENGES

6.1 Performance Gaps

Typically, performance saturation is observed as the scaling ratio (the number of nodes in scaled graphs compared to the original graph) increases. However, researchers also note a significant gap between saturated performance and that of the original graph [60], especially in large datasets. As shown in table 3, We collected the reported SOTA results from several recent benchmark works (specifically by Sun et al. in GC-Bench [63], Liu et al. in GCondenser [65] and Gong et al. in GC4NC [64]) and compared the reported best performance across all scale ratios with the performance on the full dataset, calculating a ratio of performance maintenance in the bottom row of table 3. If this ratio reaches 100%, it indicates that at least for the current task and model, the scaled graph retains all the key information of the original dataset, i.e., the task-specific information is lossless. However, sorted by the scale of the original dataset, it can be observed that even the SOTA methods cannot achieve 100% performance on larger datasets (Arxiv, Reddit, and Products).

Table 3. The best performance of reported SOTA at all scale rates against the performance on the full dataset.

Dataset	Cora	Citeseer	ACM	PubMed	DBLP	Flickr	Arxiv	Reddit	Products
Num. Nodes	2,708	3,327	10,942	19,717	37,791	89,250	169,343	232,965	2,449,029
SOTA Method	SGDD	GDEM	SFGC	GEOM	SFGC	GCDM	SFGC	SFGC	GEOM
Baseline	80.0	71.4	91.7	79.3	80.1	46.8	71.4	94.4	73.1
NC ACC	81.4	73.4	92.2	80.1	82.1	49.3	67.8	91.3	71.1
% Maintenance	101%	102%	101%	101%	102%	105%	94%	96%	97%

'Baseline' indicates the node classification accuracy (NC ACC) of the model trained on the original graph; 'NC ACC' reports the best performance of the same model trained on the scaled graphs.

As we analyze, maintenance of performance exceeding 100% can be attributed to the dual role of graph scaling as a denoising process, which effectively enhances the quality and information density of the training data compared to the original dataset. For cases where performance falls below 100%, since this issue only arises in extremely large-scale graphs, we hypothesize that the extreme complexity of large-scale graph structures may render current methods insufficient. Specifically, the widely used approach of employing MLPs to predict connectivity based solely on node attributes may struggle to model the intricate structural dependencies inherent in these large graphs. Structure-free methods such as SFGC offer a simple yet effective solution by completely discarding the graph structure of the scaled graph and encoding everything into an embedding as the node feature. However, this approach results in the scaled graph losing real-world semantics of node connectivity, making it understandable only to machine learning models. The performance gap observed on large-scale graphs, specifically the inability to surpass the performance of models trained on the original large-scale graphs, remains an unresolved challenge in the field.

6.2 Efficiency Concern for Applications

If the ultimate goal of GS is to train GNNs effectively on scaled datasets, it may be required to ensure that the time and resources invested in GS do not surpass the time saved by training on smaller graphs. Currently, evaluations in

this area mainly focus on estimating algorithm complexity and statistical analysis of actual runtime (with respect to optimization epochs). According to the experiment results reported in GCondenser [65], structure-free methods such as SFGC exhibit the shortest running time, while model-guided methods with sequential optimization formulations, such as GCond, have the longest runtime. However, since the GS process is a one-time effort, as long as the downstream task continues for a sufficient duration or is repeated frequently, the efficiency requirement can still be fulfilled. Thus, we believe that the downstream task scenarios should also be included as part of the efficiency evaluation scheme.

6.3 Comprehensive Effectiveness Metrics

Existing methods predominantly evaluate GS effectiveness based on the performance of scaled graphs in downstream tasks. Conventional performance metrics, like classification accuracy, might fall short in addressing critical issues such as fairness [55], robustness against adversarial attacks [53] etc. A robust synthetic dataset is one that is accurate, reliable, and useful for its intended purpose; A high-quality scaled graph dataset should be versatile, applicable across multiple settings, regularly updated, and reusable. Therefore, we believe that a comprehensive evaluation protocol should cover the ability of a trained model to maintain consistent performance when confronted with perturbations, noise, and multiple downstream purposes in real-world scenarios. The identified problems are currently limited and incomplete, despite existing research, warranting a more extensive investigation. While few efforts, e.g. [43], have proposed constraints focusing on group fairness, future exploration could broaden the scope to include other universal constraints and apply evaluation metrics (e.g., regarding fairness and robustness) for GS.

6.4 Unexplored Methodological Capabilities

The scopes of investigations on generalization were exploring performances with (1) Various GNN architectures; (2) Model convergence; (3) Optimal scaling ratios and (4) multiple downstream tasks. The main idea is to use the model performance on the scaled graph as a metric to assess the quality of the graph scaling. In this context, the performance metric is considered independent of the semantics of the scaled graph, and thus we consider GS methods using this strategy of evaluation to have lower interpretability. By incorporating the leading explanation techniques (e.g., GNNExplainer [80] and GSAT [81] to select the important nodes in the training process), EXGC [54] injects the explainability into their models.

7 CONCLUSION AND FUTURE DIRECTIONS

In conclusion, this paper investigated the research domain of Graph Scaling, commencing with a formal problem definition that distinguishes it from general dataset distillation, and progressing through taxonomy and analysis of optimization objectives, formulation methodologies, datasets, and evaluation metrics. While substantial progress has been achieved in GS, numerous avenues for future exploration and development remain. As the field continues to evolve, addressing the following areas will be crucial for advancing the capabilities and applications of GS:

7.1 Interpretability of the Scaled Graphs.

Graphs have emerged as a tool to simultaneously model relationships (edges) between **real world** entities (nodes), and GNNs were developed to learn knowledge from Graphs. Unlike the natural interpretability of dataset distillation in the field of computer vision [62], the output of scaled graphs requires further exploration to enhance their real-world interpretability. In the scaled graph, there may exist aggregated or synthetic nodes and edges that do not exist in the real world. The challenge lies in interpreting the semantics of these elements. Existing methods often lack semantic analysis; typically, after obtaining the scaled graph, it is directly fed into GNNs

as training input, neglecting the interpretation of the semantic meaning of the scaled graph itself. Thus, future research should emphasize interpreting the scaled graphs.

Moreover, by generating smaller and high-quality scaled graphs, the efficiency of graph data mining methods can be fundamentally enhanced. However, validating the usefulness of a method in a specific scenario still relies on the 'retrain-evaluate' paradigm. In other words, the assessment of graph data quality is indirectly performed by evaluating the performance of the model trained on that data. Moving one step further, quantifying the interpretability of graphs can be a solution to graph data quality assessment. We believe that, in the future, a key aspect of data-centric graph machine learning will be the development of graph data quality assessment methods. On one hand, graph data quality assessment can directly evaluate the effectiveness of graph scaling methods without the need for frequent model retraining. On the other hand, given the difficulty of handling large graphs, scaling down graphs using graph scaling methods can also empower graph quality assessment methods to run more efficiently.

7.2 More Type of Graphs to Scale.

Although GS has been successfully developed in various graphs, most of the existing methods have primarily focused on undirected, homogeneous, static graphs. However, graphs in real-world scenarios are usually more complicated [82], such as dynamic graphs (e.g., traffic flow graphs that vary over time), heterogeneous graphs (e.g., user-item graphs), etc. In a particular study [47], the Scaling target comprised heterogeneous graphs, yet the focus remained singular, concentrating solely on one category. Specifically, the research aimed to identify background nodes linked with target nodes, including affiliation nodes and bridging nodes. By isolating these nodes, the study then derived a skeleton graph, achieving an exceptionally small ratio. Scaling on these complex graphs requires preserving richer information from the original graph, which in turn poses greater challenges. More recently, there are works focusing on scaling heterogeneous graphs [39, 45], dynamic graphs [69] and multi-label scenarios [83]. Due to the complexity and diversity of real-world graph data, more graph types should also be considered.

7.3 Exploring the Correlation between GS methods.

Under our taxonomy, optimization objectives can be categorized into two groups: graph-guided and model-guided, by specific information to preserve. These two types of objectives are not inherently conflicting, yet their mutual relation has not been conclusively investigated. For instance, it remains uncertain whether there exists theoretical assurance that the preservation of certain graph properties is sufficient for the retention of GNN performance, or the other way around. Recently, FreeHGC [39] and CGC [76] have demonstrated that scaled graphs generated through training-free, heuristic-driven designs can also deliver impressive performance. We believe that future research on GS methods, by demonstrating that maintaining specific graph properties is crucial for preserving performance, could fill the gaps in the theoretical relationship between retained graph properties and downstream outcomes.

7.4 More Graph Formulation.

While numerous methods have been developed to simplify graph representations, we believe that many potential and feasible approaches remain unexplored. For example, non-uniform sampling of labels during GS might be a viable solution to address label imbalance issues [84]; node features can be predefined as mutually orthogonal one-hot vectors, similar to what was done in [85], just to generate the topology and thus facilitate relationship learning, etc. Promising directions include leveraging transfer learning (e.g., recently in [57]), integrating reinforcement learning, and employing self-supervised learning (e.g., recently in CTGC [33]). Additionally, generative models like GANs and VAEs, and hybrid models combining multiple GS techniques, could yield superior results (e.g.,

in [53] and [55]). Differences in methodological perspectives can drive experimental approaches to generate synthetic graphs according to a particular tendency, and many core concepts from the field of computer vision can be well adapted to graph data research. This raises the question: are there methods more suited to the unique properties of graphs, specifically designed for graph data in non-Euclidean spaces or for various types of GNN models? We look forward to exploring these and other directions in future research.

7.5 Tradeoff Framework.

Within the exploration of applications, we inevitably confront a crucial yet delicate question: How do we determine the scale of the scaled graph to meet the predefined purpose of GS? Although existing methods (e.g., in [65]) have recognized the tradeoff between effectiveness and efficiency (as evidenced by their ratio-performance figure), we argue that both effectiveness and efficiency should be comprehensively included in a tradeoff framework. This is crucial to specify the utility of GS and expand its application scope to more practical scenarios. By considering both aspects, we can better understand the benefits and limitations of GS techniques and make informed decisions about their applicability in real-world settings.

8 ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 62476245); and the Zhejiang Provincial Natural Science Foundation of China (Grant No. LTGG23F030005).

REFERENCES

[1] Shazia Tabassum, Fabiola SF Pereira, Sofia Fernandes, and João Gama. Social network analysis: An overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(5):e1256, 2018.
[2] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.
[3] Charu C Aggarwal and Charu C Aggarwal. An introduction to recommender systems. *Recommender systems: the textbook*, pages 1–28, 2016.
[4] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
[5] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.
[6] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.
[7] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.
[8] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5782–5799, 2022.
[9] Zhiqiang Zhong, Cheng-Te Li, and Jun Pang. Hierarchical message-passing graph neural networks. *Data Mining and Knowledge Discovery*, 37(1):381–408, 2023.
[10] Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. A comprehensive study on large-scale graph training: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 35:5376–5389, 2022.
[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
[12] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
[13] Yilun Liu, Ruihong Qiu, and Zi Huang. Cat: Balanced continual graph learning with graph condensation. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 1157–1162. IEEE, 2023.
[14] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. International Joint Conference on Artificial Intelligence, 2021.
[15] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, and Dongrui Fan. Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, 9(2):205–234, 2021.

[16] Daniel Baker, Vladimir Braverman, Lingxiao Huang, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in graphs of bounded treewidth. In *International Conference on Machine Learning*, pages 569–579. PMLR, 2020.
[17] Wai Shing Fung, Ramesh Hariharan, Nicholas JA Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 71–80, 2011.
[18] Lam BQ Nguyen, Ivan Zelinka, Vaclav Snasel, Loan TT Nguyen, and Bay Vo. Subgraph mining in a large graph: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(4):e1454, 2022.
[19] Nasrin Shabani, Jia Wu, Amin Beheshti, Quan Z Sheng, Jin Foo, Venus Haghighi, Ambreen Hanif, and Maryam Shahabikargar. A comprehensive survey on graph summarization with graph neural networks. *IEEE Transactions on Artificial Intelligence*, 2024.
[20] Chuang Liu, Yibing Zhan, Jia Wu, Chang Li, Bo Du, Wenbin Hu, Tongliang Liu, and Dacheng Tao. Graph pooling for graph neural networks: progress, challenges, and opportunities. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 6712–6722, 2023.
[21] Xinyi Gao, Junliang Yu, Tong Chen, Guanhua Ye, Wentao Zhang, and Hongzhi Yin. Graph condensation: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2025.
[22] Mohammad Hashemi, Shengbo Gong, Juntong Ni, Wenqi Fan, B. Aditya Prakash, and Wei Jin. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 8058–8066. ijcai.org, 2024.
[23] Jie Chen, Yousef Saad, and Zechen Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, pages 1–37, 2022.
[24] Zhiqian Chen, Fanglan Chen, Lei Zhang, Taoran Ji, Kaiqun Fu, Liang Zhao, Feng Chen, Lingfei Wu, Charu Aggarwal, and Chang-Tien Lu. Bridging the gap between spatial and spectral domains: A unified framework for graph neural networks. *ACM Computing Surveys*, 56(5):1–42, 2023.
[25] Zhiqian Chen, Fanglan Chen, Lei Zhang, Taoran Ji, Kaiqun Fu, Liang Zhao, Feng Chen, Lingfei Wu, Charu Aggarwal, and Chang-Tien Lu. Bridging the gap between spatial and spectral domains: A unified framework for graph neural networks. *ACM Computing Surveys*, 56(5):1–42, 2023.
[26] Beining Yang, Kai Wang, Qingyun Sun, Cheng Ji, Xingcheng Fu, Hao Tang, Yang You, and Jianxin Li. Does graph distillation see like vision dataset counterpart? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
[27] Gecia Bravo Hermsdorff and Lee Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. *Advances in Neural Information Processing Systems*, 32, 2019.
[28] Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.
[29] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 675–684, 2021.
[30] Yang Liu, Deyu Bo, and Chuan Shi. Graph distillation with eigenbasis matching. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.
[31] K Ch Das. The laplacian spectrum of a graph. *Computers & Mathematics with Applications*, 48(5-6):715–724, 2004.
[32] Xingcheng Fu, Yisen Gao, Beining Yang, Yuxuan Wu, Haodong Qian, Qingyun Sun, and Xianxian Li. Bi-directional multi-scale graph dataset condensation via information bottleneck. *arXiv preprint arXiv:2412.17355*, 2024.
[33] Xinyi Gao, Yayong Li, Tong Chen, Guanhua Ye, Wentao Zhang, and Hongzhi Yin. Contrastive graph condensation: Advancing data versatility through self-supervised learning. *arXiv preprint arXiv:2411.17063*, 2024.
[34] Yu Jin, Andreas Loukas, and Joseph JaJa. Graph coarsening with preserved spectral properties. In *International Conference on Artificial Intelligence and Statistics*, pages 4452–4462. PMLR, 2020.
[35] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, 2020.
[36] Jiarong Xu, Renhong Huang, Xin Jiang, Yuxuan Cao, Carl Yang, Chunping Wang, and Yang Yang. Better with less: A data-active perspective on pre-training graph neural networks. *Advances in Neural Information Processing Systems*, 36:56946–56978, 2023.
[37] Manoj Kumar, Anurag Sharma, Shashwat Saxena, and Sandeep Kumar. Featured graph coarsening with similarity guarantees. In *International Conference on Machine Learning*, pages 17953–17975. PMLR, 2023.
[38] Jiongqian Liang, Saket Gurukar, and Srinivasan Parthasarathy. Mile: A multi-level framework for scalable graph embedding. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 15, pages 361–372, 2021.
[39] Yuxuan Liang, Wentao Zhang, Xinyi Gao, Ling Yang, Chong Chen, Hongzhi Yin, Yunhai Tong, and Bin Cui. Training-free heterogeneous graph condensation via data selection. *arXiv preprint arXiv:2412.16250*, 2024.
[40] Xiaowen Wei, Xiuwen Gong, Yibing Zhan, Bo Du, Yong Luo, and Wenbin Hu. Clnode: Curriculum learning for node classification. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 670–678, 2023.
[41] Tengfei Ma and Jie Chen. Unsupervised learning of graph hierarchical abstractions with differentiable coarsening and optimal transport. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8856–8864, 2021.

[42] Xinyi Gao, Tong Chen, Yilong Zang, Wentao Zhang, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. Graph condensation for inductive node representation learning. In *40th IEEE International Conference on Data Engineering, ICDE 2024*, pages 3056–3069. IEEE, 2024.
[43] Qizhang Feng, Zhimeng Jiang, Ruiquan Li, Yicheng Wang, Na Zou, Jiang Bian, and Xia Hu. Fair graph distillation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
[44] Tianle Zhang, Yuchen Zhang, Kun Wang, Kai Wang, Beining Yang, Kaipeng Zhang, Wenqi Shao, Ping Liu, Joey Tianyi Zhou, and Yang You. Two trades is not baffled: Condense graph via crafting rational gradient matching. *arXiv preprint arXiv:2402.04924*, 2024.
[45] Jian Gao, Jianshe Wu, and Jingyi Ding. Heterogeneous graph condensation. *IEEE Transactions on Knowledge and Data Engineering*, 2024.
[46] Zhanyu Liu, Chaolv Zeng, and Guanjie Zheng. Graph data condensation via self-expressive graph structure reconstruction. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1992–2002, 2024.
[47] Linfeng Cao, Haoran Deng, Yang Yang, Chunping Wang, and Lei Chen. Graph-skeleton:~ 1% nodes are sufficient to represent billion-scale graph. In *Proceedings of the ACM on Web Conference 2024*, pages 570–581, 2024.
[48] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.
[49] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2021.
[50] Jiahao Wu, Ning Lu, Zeiyu Dai, Wenqi Fan, Shengcai Liu, Qing Li, and Ke Tang. Backdoor graph condensation. *arXiv preprint arXiv:2407.11025*, 2024.
[51] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 720–730, 2022.
[52] Mucong Ding, Xiaoyu Liu, Tahseen Rabbani, and Furong Huang. Faster hyperparameter search on graphs via calibrated dataset condensation. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.
[53] Xinglin Li, Kun Wang, Hanhui Deng, Yuxuan Liang, and Di Wu. Attend who is weak: Enhancing graph condensation via cross-free adversarial training. *arXiv preprint arXiv:2311.15772*, 2023.
[54] Junfeng Fang, Xinglin Li, Yongduo Sui, Yuan Gao, Guibin Zhang, Kun Wang, Xiang Wang, and Xiangnan He. EXGC: bridging efficiency and explainability in graph condensation. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 721–732. ACM, 2024.
[55] Runze Mao, Wenqi Fan, and Qing Li. Gcare: Mitigating subgroup unfairness in graph condensation through adversarial regularization. *Applied Sciences*, 13(16):9166, 2023.
[56] Yezi Liu and Yanning Shen. Tinygraph: joint feature and node condensation for graph neural networks. *arXiv preprint arXiv:2407.08064*, 2024.
[57] Bo Yan. Federated graph condensation with information bottleneck principles. *arXiv preprint arXiv:2405.03911*, 2024.
[58] Xinyi Gao, Hongzhi Yin, Tong Chen, Guanhua Ye, Wentao Zhang, and Bin Cui. Robgc: Towards robust graph condensation. *arXiv preprint arXiv:2406.13200*, 2024.
[59] Jian Gao and Jianshe Wu. Multiple sparse graphs condensation. *Knowledge-Based Systems*, 278:110904, 2023.
[60] Yuchen Zhang, Tianle Zhang, Kai Wang, Ziyao Guo, Yuxuan Liang, Xavier Bresson, Wei Jin, and Yang You. Navigating complexity: Toward lossless graph condensation via expanding window matching. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.
[61] Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems*, 36, 2024.
[62] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.
[63] Qingyun Sun, Ziying Chen, Beining Yang, Cheng Ji, Xingcheng Fu, Sheng Zhou, Hao Peng, Jianxin Li, and S Yu Philip. Gc-bench: An open and unified benchmark for graph condensation. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
[64] Shengbo Gong, Juntong Ni, Noveen Sachdeva, Carl Yang, and Wei Jin. Gc4nc: A benchmark framework for graph condensation on node classification with new insights. *arXiv preprint arXiv:2406.16715*, 2024.
[65] Yilun Liu, Ruihong Qiu, and Zi Huang. Gcondenser: Benchmarking graph condensation. *arXiv preprint arXiv:2405.14246*, 2024.
[66] Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. Kernel ridge regression-based graph dataset distillation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2850–2861, 2023.
[67] Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. Fast graph condensation with structure-based neural tangent kernel. In *Proceedings of the ACM on Web Conference 2024*, pages 4439–4448, 2024.
[68] Yuxiang Wang, Xiao Yan, Shiyu Jin, Hao Huang, Quanqing Xu, Qingchen Zhang, Bo Du, and Jiawei Jiang. Self-supervised learning for graph dataset condensation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages

3289–3298, 2024.
[69] Xinyi Gao, Tong Chen, Wentao Zhang, Yayong Li, Xiangguo Sun, and Hongzhi Yin. Graph condensation for open-world graph learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 851–862, 2024.
[70] Qiying Pan, Ruofan Wu, LIU Tengfei, Tianyi Zhang, Yifei Zhu, and Weiqiang Wang. Fedgkd: Unleashing the power of collaboration in federated graph neural networks. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.
[71] Zhenbang Xiao, Shunyu Liu, Yu Wang, Tongya Zheng, and Mingli Song. Disentangled condensation for large-scale graphs. *arXiv preprint arXiv:2401.12231*, 2024.
[72] Fan Li, Xiaoyang Wang, Dawei Cheng, Wenjie Zhang, Ying Zhang, and Xuemin Lin. Tcgu: Data-centric graph unlearning based on transferable condensation. *arXiv preprint arXiv:2410.06480*, 2024.
[73] Mengyang Liu, Shanchuan Li, X Chen, and Le S. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697*, 2022.
[74] Charles Dickens, Edward Huang, Aishwarya Reganti, Jiong Zhu, Karthik Subbian, and Danai Koutra. Graph coarsening via convolution matching for scalable graph neural network training. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1502–1510, 2024.
[75] Zhenbang Xiao, Yu Wang, Shunyu Liu, Huiqiong Wang, Mingli Song, and Tongya Zheng. Simple graph condensation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 53–71. Springer, 2024.
[76] Xinyi Gao, Tong Chen, Wentao Zhang, Junliang Yu, Guanhua Ye, Quoc Viet Hung Nguyen, and Hongzhi Yin. Rethinking and accelerating graph condensation: A training-free approach with class partition. *arXiv preprint arXiv:2405.13707*, 2024.
[77] Yilun Liu, Ruihong Qiu, Yanran Tang, Hongzhi Yin, and Zi Huang. Puma: Efficient continual graph learning for node classification with graph condensation. *IEEE Transactions on Knowledge and Data Engineering*, 2024.
[78] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Curgraph: Curriculum learning for graph classification. In *Proceedings of the Web Conference 2021*, pages 1238–1248, 2021.
[79] Mridul Gupta, Sahil Manchanda, Hariprasad Kodamana, and Sayan Ranu. Mirage: Model-agnostic graph distillation for graph classification. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
[80] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.
[81] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pages 15524–15543. PMLR, 2022.
[82] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research*, 21(1):2648–2720, 2020.
[83] Liangliang Zhang, Haoran Bao, and Yao Ma. Extending graph condensation to multi-label datasets: A benchmark study. *arXiv preprint arXiv:2412.17961*, 2024.
[84] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 833–841, 2021.
[85] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2021.

Received 27 June 2024; revised 17 February 2025; accepted 2 April 2025