A Comprehensive Survey on Graph Reduction: Sparsification, Coarsening, and Condensation

Mohammad Hashemi[*1], **Shengbo Gong**[*1], **Juntong Ni**[1],
Wenqi Fan[2], **B. Aditya Prakash**[3], **Wei Jin**[1]

[1]Emory University, [2]The Hong Kong Polytechnic University, [3]Georgia Institute of Technology
mohammad.hashemi@emory.edu, jshmhsb@gmail.com, juntongni02@gmail.com,
wenqi.fan@polyu.edu.hk, badityap@cc.gatech.edu, wei.jin@emory.edu

Abstract

Many real-world datasets can be naturally represented as graphs, spanning a wide range of domains. However, the increasing complexity and size of graph datasets present significant challenges for analysis and computation. In response, graph reduction techniques have gained prominence for simplifying large graphs while preserving essential properties. In this survey, we aim to provide a comprehensive understanding of graph reduction methods, including graph sparsification, graph coarsening, and graph condensation. Specifically, we establish a unified definition for these methods and introduce a hierarchical taxonomy to categorize the challenges they address. Our survey then systematically reviews the technical details of these methods and emphasizes their practical applications across diverse scenarios. Furthermore, we outline critical research directions to ensure the continued effectiveness of graph reduction techniques, as well as provide a comprehensive paper list at https://github.com/ChandlerBang/awesome-graph-reduction. We hope this survey will bridge literature gaps and propel the advancement of this promising field.

1 Introduction

Graph-structured data has become ubiquitous in various domains, ranging from social networks and biological systems to recommendation systems and knowledge graphs [Fan *et al.*, 2019; Wu *et al.*, 2022b, 2018; Shi and Weninger, 2017; Wang *et al.*, 2021]. The inherent relational structure of graph data makes it a powerful representation for modeling complex interactions and dependencies. Moreover, with the rise of graph machine learning techniques, especially graph neural networks (GNNs) [Kipf and Welling, 2016; Wu *et al.*, 2020], the utilization of graph datasets has seen unprecedented growth, leading to advancements in tasks such as node classification, link prediction, graph classification, and graph generation [Zhou *et al.*, 2020; Ma and Tang, 2021].

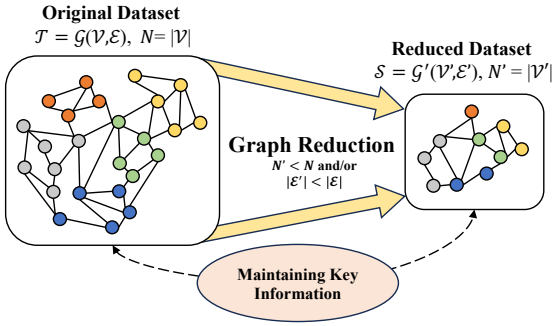

Figure 1: A general framework of graph reduction. Graph reduction aims to find a reduced (smaller) graph dataset that can preserve certain information of the original graph dataset.

Recent years have witnessed an exponential increase in the size and complexity of graph datasets. Large-scale networks, such as social graphs and citation networks [Hu *et al.*, 2021], challenge the scalability and efficiency of existing algorithms and demand innovative solutions for efficient model training. Despite recent efforts to design GNNs that can scale with large graphs [Jia *et al.*, 2020; Zeng *et al.*, 2021; Song *et al.*, 2023; Liu *et al.*, 2021], an alternative approach focuses on reducing the size of the graph dataset, including the number of graphs, nodes, and edges, which we term as **graph reduction**[1] [Jin *et al.*, 2022b; Huang *et al.*, 2021]. In this paper, we define graph reduction as *the process of finding a graph dataset of smaller size while preserving its key information*. Specifically, this definition requires an algorithm that takes the original graph dataset as input and produces a smaller one. As shown in Figure 1, graph reduction aims to extract the essential information from the massive graph dataset by maintaining its structural and semantic characteristics. In addition to accelerating graph algorithms, graph reduction offers a range of advantages. **First**, reduced graphs demonstrate compatibility with various downstream models architectures [Jin *et al.*, 2022b]. **Second**, graph reduction may contribute to privacy preservation since it alters the original structure or node attributes, making them challenging to recover [Dong *et al.*,

[*]Equal contribution.

[1]It is also known as graph summarization, simplification or degeneracy in some literature. We choose to consistently use "graph reduction" throughout this survey for clarity and uniformity.

2022]. **Third**, the reduced graph is notably smaller and more comprehensible for humans compared to its larger counterpart, which aids in graph visualization [Imre *et al.*, 2020].

Given the importance of graph reduction, numerous algorithms have been developed, falling into three distinct strategies: **graph sparsification** [Althöfer *et al.*, 1993; Batson *et al.*, 2009], **graph coarsening** [Loukas and Vandergheynst, 2018; Dorfler and Bullo, 2012], and the more recent **graph condensation** [Jin *et al.*, 2022b,a; Xu *et al.*, 2023; Liu *et al.*, 2022]. Graph sparsification revolves around the approximation of a graph by retaining only a subset of its edges and vital nodes. In contrast, graph coarsening does not eliminate any nodes but instead groups and amalgamates nodes into super nodes, with original inter-group edges being aggregated into super edges using a specified aggregation algorithm. Differing from the aforementioned two strategies, graph condensation has been recently introduced as a method to condense a graph by synthesizing a smaller graph while preserving the performance of GNNs. Despite the proliferation of these methods, they have often been studied in isolation, leaving the connections and distinctions between them somewhat obscured. Therefore, it is both necessary and timely to offer a systematic overview of these existing algorithms in order to enhance our understanding of graph reduction techniques.

Contributions. In this work, we aim to present a comprehensive and up-to-date survey focusing on graph reduction techniques and their diverse applications in tackling graph-related challenges. We aspire for this survey to serve as a valuable resource for both novice researchers and practitioners interested in exploring this field, while also catalyzing future research endeavors. Our contributions can be summarized as follows:

(a) We offer the first comprehensive review of graph reduction methods, encompassing graph sparsification, graph coarsening, and graph condensation.
(b) We develop a unified perspective for existing graph reduction methods, differentiating them based on their characteristics in Section 2, and provide a detailed review of representative algorithms in Section 3.
(c) We discuss practical applications of graph reduction methods in Section 4, shedding light on real-world scenarios where these techniques prove valuable.
(d) In Section 5, we identify key challenges and promising future research directions, guiding the continued advancement of graph reduction techniques.

Connection to Existing Surveys. In contrast to previous reviews on graph reduction [Liu *et al.*, 2018; Interdonato *et al.*, 2020; Shabani *et al.*, 2023], our study offers a comprehensive overview of the emerging field of graph condensation and presents a unified framework that bridges graph condensation with conventional graph reduction techniques. Additionally, our survey explores synergies between graph reduction and graph neural networks, an aspect rarely covered in existing surveys. Furthermore, our work shares connections with recent surveys on dataset distillation [Geng *et al.*, 2023; Sachdeva and McAuley, 2023], while they predominantly focus on condensation methods applied to image data.

Table 1: Notations used in this paper.

Notation	Description
G	A graph
G'	A reduced graph
$\mathcal{V}$	Set of graph nodes
$\mathcal{E}$	Set of graph edges
$\mathbf{X}$	Node feature matrix
$\mathbf{A}$	Adjacency matrix
$\mathbf{Y}$	One-hot label matrix
$\mathbf{L}$	Graph Laplacian matrix
$\mathbf{C}$	Node assignment/mapping matrix
$\mathcal{T}$	Original graph dataset
$\mathcal{S}$	Synthetic graph dataset
N	Number of nodes

2 Taxonomy of Graph Reduction

Before we formally introduce the definition of graph reduction, we first introduce the notations used in this paper. Given the node set $\mathcal{V}$ and edge set $\mathcal{E}$, we denote a graph as $G = (\mathcal{V}, \mathcal{E})$. In attributed graphs, nodes are associated with features, and thus can be represented as $G = (\mathbf{A}, \mathbf{X})$, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$ denotes the node attributes and $\mathbf{A}$ denotes the adjacency matrix. The graph Laplacian matrix is $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal degree matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. We use $N = |\mathcal{V}|$ and $E = |\mathcal{E}|$ to denote the number of nodes and edges, respectively. We summarize the major notations in Table 1.

A Unified Framework of Graph Reduction Given a graph $G = (\mathcal{V}, \mathcal{E})$, graph reduction outputs a graph $G' = (\mathcal{V}', \mathcal{E}')$ which contains N' nodes and E' edges, subject to $N' < N$, or $E' < E$ edges. The reduced graph G' preserves the desired information of the original graph G. This process can be understood as finding a graph G' that minimizes a loss function $\mathcal{L}$ explicitly or implicitly, which measures the difference between $\mathbf{G}$ and $\mathbf{G}'$ in terms of certain information:

$$G' = \arg\min_{G'} \mathcal{L}(G, G'). \tag{1}$$

Remark 1. Note that the desired outcome of graph reduction should be represented as a graph, leading to the exclusion of graph representation learning methods [Wu *et al.*, 2020] in the context of graph reduction. Furthermore, we restrict this definition to graphs that converge to an optimal state through the algorithmic process. This distinction sets it apart from data augmentation techniques [Rong *et al.*, 2019; Zhao *et al.*, 2022], where the augmented graphs vary with each epoch.

Remark 2. While the majority of graph reduction methods focus on decreasing the number of nodes or edges within a graph, there are also studies [Jin *et al.*, 2022a; Xu *et al.*, 2023] that reduce the number of distinct graphs, particularly in applications such as graph classification. In this survey, unless explicitly specified otherwise, our primary focus remains on the former approach, as it is more commonly employed.

On top of that, we can categorize existing graph reduction techniques into the following three groups, based on how they produce the reduced graphs:

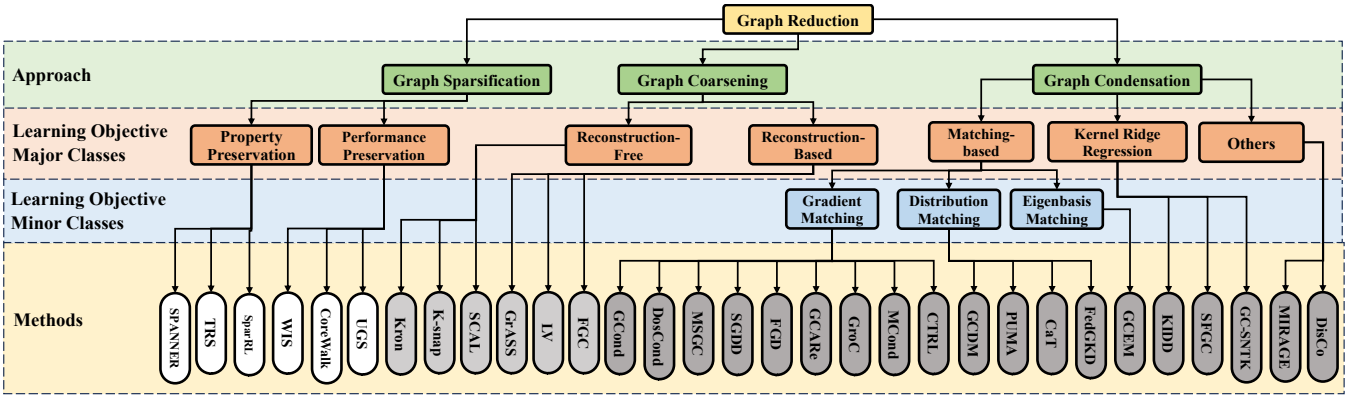

Figure 2: Taxonomy of existing graph reduction methods.

Definition of Graph Sparsification Given a graph $G = (\mathbf{A}, \mathbf{X})$, graph sparsification selects existing nodes or edges from the graph G and outputs $G' = (\mathbf{A}', \mathbf{X}')$. In other words, the elements in $\mathbf{A}'$ or $\mathbf{X}'$ are the subset of those in $\mathbf{A}$ or $\mathbf{X}$.

Definition of Graph Coarsening Given a graph $G = (\mathbf{A}, \mathbf{X})$, graph coarsening outputs $G' = (\mathbf{A}', \mathbf{X}')$ containing N' super-nodes (clusters) and E' super-edges, where $N' < N$. It requires finding a surjective mapping from the original graph to a coarse graph, which can be formulated by a one-hot matrix $\mathbf{C} \in \{0,1\}^{N \times N'}$ that assigns nodes to super-nodes. We further define the reverse assignment matrix $\mathbf{P} = \text{rowNormalize}(\mathbf{C}^\top)$. Then the coarse graph is usually constructed by $\mathbf{A}' = \mathbf{C}^\top \mathbf{A}\mathbf{C}, \mathbf{X}' = \mathbf{P}\mathbf{X}$ with the Laplacian matrix being $\mathbf{L}' = \mathbf{C}^\top \mathbf{L}\mathbf{C}$.

Definition of Graph Condensation Given a graph $\mathcal{T} = (\mathbf{A}, \mathbf{X}, \mathbf{Y})$, with $\mathbf{Y}$ being the node labels, graph condensation aims to learn a small-synthetic graph dataset $\mathcal{S} = (\mathbf{A}', \mathbf{X}', \mathbf{Y}')$, where $N' < N$, such that a GNN trained on $\mathcal{S}$ obtains a comparable performance to the one trained on $\mathcal{T}$.

Distinctions. The above three strategies share a common goal: to obtain a small graph that can preserve key information and benefit downstream processes. However, they differ in three key aspects. **First**, graph condensation synthesizes fake graph elements, while sparsification selects existing ones and coarsening aggregates them. The latter two strategies enjoy certain interpretability in the reduction process, as the reduced graph can be easily understood and related back to the original graph. **Second**, these strategies have distinct objectives. Graph condensation aims to maintain the performance of GNN models in downstream tasks, while the other two often target at preserving graph properties. **Third**, graph condensation relies on labels, whereas the other two generally do not.

In Figure 2, we present a detailed taxonomy of existing graph reduction methods within the aforementioned categories, and we will elaborate in the following section. Additionally, Table 2 provides a qualitative comparison of the three graph reduction strategies mentioned earlier.

3 Methodology

In this section, we introduce the representative algorithms for the aforementioned three strategies of graph reduction. For each strategy, we categorize methods by their learning objectives and summarize popular approaches in Table 3.

3.1 Graph Sparsification

Graph sparsification, as an intuitive method for graph reduction, involves the selection of crucial edges or nodes based on specific criteria. Conventional approaches typically focus on preserving specific graph properties, such as spectrum and centrality. With the increasing popularity of GNNs, there is a growing array of methods aimed at maintaining the quality of node representations. Consequently, we categorize existing techniques into two groups based on their preservation goals: those focused on preserving graph properties and those dedicated to maintaining model performance.

Preserving Graph Properties

In traditional graph sparsification, essential graph properties include pairwise distances, cuts, and spectrum [Batson *et al.*, 2013]. Sparsification methods iteratively sample the subgraphs that achieve the minimal loss $\mathcal{L}(G', G)$ in a greedy manner, which measures the approximation to the original graph w.r.t. one of the above graph properties. A reduced graph is called *spanner* if it maintains pairwise distances, and *sparsifier* if it preserves cut or spectrum [Batson *et al.*, 2013]. To evaluate these algorithms, one common way is to establish the loss bound for their output graph G': If G' is proved to satisfy $\mathcal{L}(G', G) \leq \epsilon, \epsilon \in (0, 1)$, it is called ϵ-spanner/sparsifier. Specifically, $\mathcal{L}(G', G)$ is expressed as $|D(G', G) - 1|$ with $D(\cdot, \cdot)$ defined as follows:

$$D(G', G) = \begin{cases} \frac{\text{SP}(G')}{\text{SP}(G)} & \text{for spanner,} \\ \frac{\mathbf{x}^\top \mathbf{L}' \mathbf{x}}{\mathbf{x}^\top \mathbf{L} \mathbf{x}} & \text{for sparsifier,} \end{cases} \tag{2}$$

where $\text{SP}(G)$ denotes the sum of the shortest path length for all node pairs in G, and $\mathbf{x} \in \mathbb{R}^N$ is an arbitrary vector.

Spanners. Althöfer *et al.* [1993] first develop an algorithm to obtain spanners in graphs. It starts with an empty graph defined on the original node set and adds edges from the original graph only if their weight is smaller than the current distance

Table 2: General qualitative comparison of graph reduction methods. "Scalability": the ability to scale up to large graphs, "Interpratability": the existence of correspondence between nodes in the original and reduced graphs, "Label Utilization": the reliance on the label information.

Strategy	Interpretability	Label Utilization	Objective	Output
Sparsification	✓	×	Property Preservation	Sampled graph
Coarsening	✓	×	Property/Performance Preservation	Supergraph
Condensation	×	✓	Performance Preservation	Synthetic graph

between their connected nodes in the reduced graph. They also prove that every weighted graph has a $2t$-spanner with $O(N^{1+1/t})$ edges. Baswana and Sen [2003] tighten this upper bound to $(2t-1)$ with a linear time algorithm that merely explores the edges in the neighborhood of a node. Furthermore, by defining a reinforcement learning process and adjusting reward functions to the preservation of pairwise distance, SparRL [Wickman *et al.*, 2022] outperforms all conventional graph sparsification approaches.

Sparsifiers. One representative sparsifier is called *Twice Ramanujan Sparsifier* (TRS) [Batson *et al.*, 2009], which prove that for every $\epsilon \in (0, 1)$ and every undirected graph G, there exists a weighted graph G' with at most $(N-1)/2$ edges such that G' is the $(1+\epsilon)$-sparsifier of G with high probability. This approach presents an algorithm for deriving G' by decomposing the graph into subgraphs with high *conductance*, calculating pairwise effective resistance (ER) [Spielman and Srivastava, 2008], and sampling edge based on normalized ER as probabilities. Then the edges in the reduced graph are reweighted as the probabilities. Furthermore, Lee and Sun [2018] present an almost-linear time algorithm for constructing such a sparsifier. Previous studies typically necessitate alterations to edge weights as part of the reweighting process. Anderson *et al.* [2014] address the sparsifier problem conditioned by keeping the original edge weights, employing a method of unweighted column selection. Since most theories are constructed upon the undirected graph, Chung [2005] first extends them into directed graphs by first symmetrizing the graph Laplacians. Different from the above methods that only cut edges, Feng [2016] first finds an extremely sparse subgraph – low-stretch spanning tree, and recovers small portions of off-tree edges to further approximate the spectrum.

Preserving Performance

With the emergence of GNNs, a new goal of graph sparsification has arisen: maintaining the prediction performance of GNNs trained on the sparsified graph. In this context, the sparsification process selects the top-k nodes or edges based on various scoring methods, such as ER [Spielman and Srivastava, 2008], PageRank [Langville and Meyer, 2004], KCenter [Sener and Savarese, 2018] and explanations from a trained GNN [Ying *et al.*, 2019].

Many methods employ model-free heuristics as the scoring strategy, which calculate the score with metrics derived from the graph structure. For example, Salha *et al.* [2019] use k-core decomposition to find interconnected subgraphs with different density index k. By treating subgraphs corresponding to high values of k as the reduced graph, they effectively circumvent the computational demands associated with calculating node embeddings for large graphs. Similarly, Brandeis *et al.* [2020] utilize this framework for reducing the graph but generate the node embeddings with different methods. Furthermore, recent work by Razin *et al.* [2023] highlights that the ability to model node interactions is primarily determined by the model-free metrics *walk index* (WI), which refers to the number of walks originating from the boundary of the partition. Consequently, unimportant edges are removed based on sorted WI values. Although these metrics offer insights from a certain perspective of the graph, they might not be compatible with the downstream models and tasks.

In contrast, recent years have also witnessed many model-based scoring methods, which utilize a parameterized model to calculate the score. For instance, Jin *et al.* [2022b] adopt coreset methods [Welling, 2009; Sener and Savarese, 2018] to select nodes based on their embeddings from a trained GNN model. Apart from these general scoring methods which can be used for any modality, recent works on the interpretability of GNN, e.g., GNNexplainer [Ying *et al.*, 2019] can also be related to graph sparsification. IGS [Li *et al.*, 2023a] sparsifies a graph based on edge importance obtained from GNNexplainer and feeds the sparsified graph into the next iteration.

Other sparsification methods belong to the graph structure learning [Zheng *et al.*, 2020; Chen *et al.*, 2021; Liu *et al.*, 2023a; Jin *et al.*, 2023] which iteratively optimize the sparsification process by interaction with GNN models. For example, Zheng *et al.* [2020] learn a sparsified graph structure by neighbor sampling and reparameterization trick [Jang *et al.*, 2016] during GNN training; UGS [Chen *et al.*, 2021] and CGP [Liu *et al.*, 2023a] simultaneously prune the graph adjacency matrix and the GNN weights.

3.2 Graph Coarsening

The selection of nodes or edges in sparsification methods can inevitably lose some information. To ensure that a sufficient amount of information is preserved, coarsening techniques have been developed, which involve grouping nodes and aggregating them. This process can be carried out iteratively, yielding hierarchical views of the original graph. Existing coarsening methods can be categorized into two groups depending on whether a reconstruction objective exists: reconstruction-based methods and reconstruction-free methods, which will be elaborated upon subsequently.

Reconstruction-Based Methods

Reconstruction-based coarsening methods involve a two-step process. First, they reconstruct the original graph from the coarse graph, where super nodes are mapped back to their

original nodes. This way, the super nodes are *lifted* to sizes comparable to those in the original graph [LeFevre and Terzi, 2010]. Subsequently, the goal is to find the coarsening mapping matrix ($\mathbf{C}$ or $\mathbf{P}$) that can minimize the differences between the reconstructed graph and the original one, which are quantified by examining their adjacency or Laplacian matrices. These coarsening techniques can be broadly categorized into spatial or spectral coarsening methods, depending on whether they utilize the adjacency or Laplacian matrix for this purpose.

Spatial coarsening. Spatial coarsening adopts the Reconstruction Error (RE) [LeFevre and Terzi, 2010] as the objective function $\mathcal{L}$:

$$RE_p(\mathbf{A}_l|\mathbf{A}) = ||\mathbf{A}_l - \mathbf{A}||_{\mathrm{F}}^p \tag{3}$$

where the *lifted* adjacency matrix $\mathbf{A}_l$ [LeFevre and Terzi, 2010] is usually defined as:

$$\mathbf{A}_l(u,v) = \begin{cases} 0 & \text{if } u = v \\ E_i/\binom{N_i}{2} & \text{if } u, v \in V_i \\ E_{ij}/(N_i N_j) & \text{if } u \in V_i, v \in V_j \end{cases} \tag{4}$$

where E_i represents the number of edges within the super node V_i, E_{ij} denotes the number of edges between V_i and V_j, and N_i is the number of nodes belonging to V_i. It is also proved that $\mathbf{A}_l$ can be expressed as a function of $\mathbf{P}$ and $\mathbf{A}$ [Riondato *et al.*, 2017]. As the first work proposing RE, GraSS [LeFevre and Terzi, 2010] randomly samples part of node pairs and merges one of them causing the smallest increase of RE. Riondato *et al.* [2017] show the connection of minimizing RE with geometric clustering problems and develops a polynomial-time approximation. Similarly, Beg *et al.* [2018] propose a weighted sampling scheme to sample vertices for merging that will result in the least RE.

Spectral coarsening. Different from spatial methods, spectral coarsening methods compare the $\mathbf{L}_l$ and $\mathbf{L}$ by comparing their eigenvalues or eigenvectors. The *lifted* Laplacian matrix is defined as $\mathbf{L}_l = \mathbf{P}^\top \mathbf{L}' \mathbf{P}$ [Kumar *et al.*, 2023]. Loukas [2019]; Loukas and Vandergheynst [2018] propose *restricted spectral approximation* and derive a relaxed evaluation called Relative Eigenvalue Error (REE) defined as $\mathrm{REE} = \sum_{i=1}^{k} |\lambda_i - \lambda_i'|/\lambda_i$, where λ_i and λ_i' are the top-k eigenvalues of the matrices $\mathbf{L}$ and $\mathbf{L}'$, respectively. Note that they use $\mathbf{L}'$ instead of $\mathbf{L}_l$ because the comparison of eigenvalues does not require the alignment of the sizes. They also give the theoretical guarantee of *greedy edge contraction* approaches, where different edge scoring methods can be used including Heavy Edge [Dhillon *et al.*, 2007], Algebraic Distance [Chen and Safro, 2011], Affinity [Livne and Brandt, 2012] and Local Variation (LV) [Loukas, 2019]. Some works hold that the edge weights can be further optimized after edge contraction. For example, Zhao *et al.* [2018] scale the edge weights by stochastic gradient descent to further align the eigenvalues after coarsening. In addition, some endeavor is made for lossless coarsening, e.g., Navlakha *et al.* [2008]; Khan *et al.* [2015] keep the correction set recording the missed edges during edge contraction.

Aside from these heuristics, there are other approaches. FGC [Kumar *et al.*, 2023] takes both the graph structure and the node attributes as the input and alternatively optimizes $\mathbf{C}$ and $\mathbf{X}'$. SGC [Bravo Hermsdorff and Gunderson, 2019] considers the edge sparsification and contraction as edge weights of 0 and ∞. Then they develop a probabilistic framework to preserve the pseudo-inverse of graph Laplacian $\mathbf{L}^+$ by $\mathbf{L}_l^+$. GOREN [Cai *et al.*, 2021] learns the edge weights in the coarse graph by a GNN with the loss to preserve the graph Laplacian $\mathbf{L}$ by $\mathbf{L}_l$.

Reconstruction-Free Methods

Despite the proliferation of reconstruction-based methods, other approaches do not rely on the reconstruction while still keeping the key information. Kron reduction [Dorfler and Bullo, 2012] is initially developed to address challenges in electrical networks, specifically to simplify resistance networks while maintaining the pairwise ER. This method calculates the coarse graph Laplacian by

$$\mathbf{L}' = \mathbf{L}_{\mathcal{V}',\mathcal{V}'} - \mathbf{L}_{\mathcal{V}',\bar{\mathcal{V}}'} \mathbf{L}_{\bar{\mathcal{V}}',\bar{\mathcal{V}}'}^{-1} \mathbf{L}_{\bar{\mathcal{V}}',\mathcal{V}} \tag{5}$$

where $\mathcal{V}'$ denotes the selected index from $\mathcal{V}$, $\bar{\mathcal{V}}' = \mathcal{V} - \mathcal{V}'$, and $\mathbf{L}_{\mathcal{V},\mathcal{V}'}$ is the submatrix of $\mathbf{L}$ whose row index is $\mathcal{V}$ and column index is $\mathcal{V}'$. Recently, Sugiyama and Sato [2023] extend it to a directed graph with self-loop. Some methods attempt to find the most informative summaries of a network. To analyze social networks with diverse attributes and relations, SNAP [Tian *et al.*, 2008] produces a summary graph where every node inside a super node has the same values for selected attributes, and is adjacent to nodes with the same selected relations. *k*-snap [Tian *et al.*, 2008] relaxes this homogeneity requirement and allows users to control the resolutions of summaries. AGSUMMARY [Wu *et al.*, 2014] utilizes the *Minimum Description Length* principle to design a cost function and compute an optimal summary by neighborhood greedy strategy. Since the former two methods only apply to discrete attributes, CANCEL [Zhang *et al.*, 2010] extends the node attributes to continual ones with adaptive cutoffs and proposes a comprehensive metric named *interestingness*. To extend the above methods focusing on only one task, CoarseNet [Purohit *et al.*, 2014] tries to find a succinct representation of the original network that preserves important diffusive characters, which can be applied to influence maximization and detecting patterns of propagation. Netgist [Amiri *et al.*, 2018] defines a task-based graph summarization problem and uses RL to create a flexible framework for learnable node merging policies.

Remarks on Graph Coarsening

Graph Coarsening in GNNs. There are growing numbers of works that combine coarsening with GNNs. For instance, SCAL [Huang *et al.*, 2021] first trains a GNN model in a graph coarsened by LV, with super node label defined as $\mathbf{Y}' = \mathrm{argmax}(\mathbf{PY})$ and then directly uses this model to inference. Following the same framework, Generale *et al.* [2022] define the relation summarization and uses R-GCN [Schlichtkrull *et al.*, 2018] for knowledge graphs. To mimic the pooling layer in the convolutional neural network, Such *et al.* [2017] make the mapping matrix learnable and produce a pooled graph reduced to fewer nodes layer by layer. CONVMATCH [Dickens *et al.*, 2023] merges nodes that are equivalent or similar

w.r.t. the GCN convolution operation. Similarly, Buffelli *et al.* [2022] match the node embeddings output by GNNs among graphs in different coarsening ratios. VNG [Si *et al.*, 2023] highlights the ongoing challenge of efficiently deploying a GNN model in online applications when connections exist between testing nodes and training nodes. To tackle this issue, they match the forward propagation by applying weighted k-means to obtain the mapping matrix $\mathbf{C}$.

Connections with Graph Clustering/Partition. Partition and clustering in graphs are long-developed areas. Graph partition aims to find a split for a graph with the least cost, e.g., cutting the fewest edges. The representative partition method, METIS [Karypis and Kumar, 1997], coarsens a graph iteratively by edge contraction, splits the nodes in the coarsest graph, and reversely maps them to the original graph. This reconstruction-based framework is widely used for partition [Safro *et al.*, 2015], which means the areas of partition and coarsening are mutually reinforcing. Graph clustering attempts to find groups of nodes with high in-group edge density, and relatively low out-group density [Tsitsulin *et al.*, 2023]. Thus, by mapping these dense groups to super nodes and aggregating them, any clustering method can apply to the coarsening strategy.

3.3 Graph Condensation

While sparsification and coarsening methods have proven effective in reducing the size of graph data, they have inherent limitations. As many of these methods prioritize preserving specific graph properties, they do not leverage the downstream task information and could lead to suboptimal model performance. Furthermore, these techniques rely on the assumption of the existence of representative nodes or edges in the original graph, which might not always hold true in the original dataset. To address these issues, graph condensation, first introduced by [Jin *et al.*, 2022b], has come into play.

Motivated by dataset distillation [Wang *et al.*, 2018] and dataset condensation [Zhao *et al.*, 2020], graph condensation revolves around condensing knowledge from a large-scale graph dataset to construct a much smaller synthetic graph from scratch. The goal is to ensure that models trained on this condensed graph dataset exhibit comparable performance to those trained on the original one. In other words, we can see graph condensation as a process of minimizing the loss defined on the models trained on the real graph $\mathcal{T}$ and the synthetic graph $\mathcal{S}$. Thus, the objective function in Eq. (1) can reformulated as follows:

$$\mathcal{S} = \arg\min_{\mathcal{S}} \mathcal{L}(\mathrm{GNN}_{\boldsymbol{\theta}_\mathcal{S}}(\mathcal{T}), \mathrm{GNN}_{\boldsymbol{\theta}_\mathcal{T}}(\mathcal{T})), \tag{6}$$

where $\mathrm{GNN}_{\boldsymbol{\theta}_\mathcal{S}}$ and $\mathrm{GNN}_{\boldsymbol{\theta}_\mathcal{T}}$ denote the GNN models trained on $\mathcal{S}$ and $\mathcal{T}$, respectively; $\mathcal{L}$ represents the loss function used to measure the difference of these two models. Based on the specific designs of $\mathcal{L}$, we classify existing graph condensation methods into three categories: matching-based methods, kernel ridge regression methods, and others.

Matching-Based Methods

To find the optimum synthetic graph dataset that minimizes the loss for a GNN trained on it, while having the lowest loss on the original graph dataset, one approach is to match some meta-data elements related to $\mathcal{S}$ and $\mathcal{T}$ like gradients w.r.t. the model parameters and distribution of node classes.

Gradient Matching. For computing the optimum synthetic graph dataset $\mathcal{S}$, Eq. (6) can be rewritten as the following bi-level problem that generalizes to the distribution of random initialization $P_{\boldsymbol{\theta}_0}$:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[\mathcal{L}\left(\mathrm{GNN}_{\boldsymbol{\theta}_\mathcal{S}}(\mathbf{A}, \mathbf{X}), \mathbf{Y}\right)\right], \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{\theta}_\mathcal{S} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\mathrm{GNN}_{\boldsymbol{\theta}(\boldsymbol{\theta}_0)}\left(\mathbf{A}', \mathbf{X}'\right), \mathbf{Y}'\right), \tag{7b}$$

where $\boldsymbol{\theta}(\boldsymbol{\theta}_0)$ denotes that $\boldsymbol{\theta}$ is a function acting on $\boldsymbol{\theta}_0$. To simplify the bi-level optimization of Eq. (7a) and (7b), Jin *et al.* [2022b] propose GCond framework, the first graph condensation method, that matches the gradients from both graph datasets match during each step of training:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\theta_0}} \left[\sum_{t=0}^{T-1} D\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_1, \nabla_{\boldsymbol{\theta}} \mathcal{L}_2\right)\right], \tag{8a}$$

$$\mathcal{L}_1 = \mathcal{L}\left(\mathrm{GNN}_{\boldsymbol{\theta}_t}\left(\mathbf{A}', \mathbf{X}'\right), \mathbf{Y}'\right), \tag{8b}$$

$$\mathcal{L}_2 = \mathcal{L}\left(\mathrm{GNN}_{\boldsymbol{\theta}_t}\left(\mathbf{A}, \mathbf{X}\right), \mathbf{Y}\right), \tag{8c}$$

where $D(\cdot, \cdot)$ represents a distance function, T stands for the total number of steps in the entire training trajectory, and $\boldsymbol{\theta}_t$ refers to the model parameters at t-th training epoch. By optimizing the above objective, the training process on the smaller synthetic graph dataset $\mathcal{S}$ mimics the path taken on the larger real dataset $\mathcal{T}$, which leads to models trained on real and synthetic datasets ending up with similar solutions. To prevent overlooking the implicit correlations between node attributes and graph structure, GCond condenses the graph structure by leveraging a function to parameterize the adjacency matrix $\mathbf{A}'$:

$$\mathbf{A}'_{ij} = \sigma\left(\left[\mathrm{MLP}_\Phi\left(\left[\mathbf{x}'_i; \mathbf{x}'_j\right]\right) + \mathrm{MLP}_\Phi\left(\left[\mathbf{x}'_j; \mathbf{x}'_i\right]\right)\right]/2\right) \tag{9}$$

where MLP_Φ is a multi-layer perceptron (MLP) parameterized with Φ and $[.;.]$ indicates concatenation. However, the optimization process in GCond involves a nested loop as shown in Eq. (8a), which hinders the scalability of the condensation method. To address this, DosCond [Jin *et al.*, 2022a] proposes a one-step GM scheme, where it exclusively matches the network gradients for the model initialization $\boldsymbol{\theta}_0$ while discarding the training trajectory of $\boldsymbol{\theta}_t$. By dropping the summation in Eq. (8a), the objective function of DosCond becomes:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[D\left(\nabla_\theta \mathcal{L}_1, \nabla_\theta \mathcal{L}_2\right)\right]. \tag{10a}$$

Note that, DosCond treats the graph structure $\mathbf{A}'$ as a probabilistic model to learn a discretized graph structure by learning a Bernoulli distribution over the edge set. Moreover, DosCond offers **a theoretical insight** into the GM scheme in graph condensation: the smallest gap between the resulting loss (achieved by training on synthetic graphs) and the optimal loss is upper bounded by the gradient matching loss. Additionally, it is worth mentioning that DosCond is the first method that does graph condensation focusing on graph classification for reducing the number of multiple graphs. In subsequent research, EXGC [Fang *et al.*, 2024] further identifies two primary causes for the inefficiency of those graph

Table 3: Summary of representative graph reduction methods. NC – Node Classification, GC – Graph Classification, LP – Link Prediction, and AD – Anomaly Detection. The "Input" column shows the type of the input graph. $\tilde{\mathbf{A}}$ indicates that the method can only applied to a symmetric adjacency matrix. $\mathbf{A}_r$ denotes the adjacency matrix with multiple relations.

Approach	Method	Learning Objective	Evaluation	Input
Graph Sparsification	SPANNER [Althöfer *et al.*, 1993]	Property Preservation	Value of ϵ	$\tilde{\mathbf{A}}$
	TRS [Batson *et al.*, 2009]		Value of ϵ	$\tilde{\mathbf{A}}$
	SparRL [Wickman *et al.*, 2022]		Value of ϵ	$\tilde{\mathbf{A}}$
	WIS [Razin *et al.*, 2023]	Performance Preservation	NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	CoreWalk [Brandeis *et al.*, 2020]		LP	$\mathbf{A}$
	UGS [Chen *et al.*, 2021]		NC, LP	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
Graph Coarsening	k-snap [Tian *et al.*, 2008]	Reconstruction-Free	Clustering Metrics	$\mathbf{X}$, $\mathbf{A}_r$
	NetGist [Amiri *et al.*, 2018]		Clustering Metrics	$\mathbf{A}$
	SCAL [Huang *et al.*, 2021]		NC	$\mathbf{X}$, $\mathbf{A}$
	VNG [Si *et al.*, 2023]		NC	$\mathbf{X}$, $\mathbf{A}$
	LV [Loukas and Vandergheynst, 2018]	Reconstruction-Based	REE	$\mathbf{A}$
	FGC [Kumar *et al.*, 2023]		REE, RE, NC	$\mathbf{X}$, $\mathbf{A}$
Graph Condensation	GCond [Jin *et al.*, 2022b]	Gradient Matching	NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	DosCond [Jin *et al.*, 2022a]		NC, GC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	MSGC [Gao and Wu, 2023]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	SGDD [Yang *et al.*, 2023]		NC, LP, AD	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	FGD [Feng *et al.*, 2023]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	GCARe [Mao *et al.*, 2023]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	CTRL [Anonymous, 2024]		NC, GC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	GroC [Li *et al.*, 2023b]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	EXGC [Fang *et al.*, 2024]		NC, GC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	MCond [Gao *et al.*, 2023]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	GCDM [Liu *et al.*, 2022]	Distribution Matching	NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	PUMA [Liu *et al.*, 2023d]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	CaT [Liu *et al.*, 2023c]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	FedGKD [Pan *et al.*, 2023]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	KiDD [Xu *et al.*, 2023]	Kernel Ridge Regression	GC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	SFGC [Zheng *et al.*, 2023]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	GC-SNTK [Wang *et al.*, 2024]		NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	GCEM [Liu *et al.*, 2023b]	Eigenbasis Matching	NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	MIRAGE [Gupta *et al.*, 2023]	Computation Tree Compression	GC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$
	DisCo [Xiao *et al.*, 2024]	Disentangled Condensation	NC	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$

condensation methods: the concurrent updating of large parameter sets and the parameter redundancy. Built on the GM scheme, it employs the Mean-Field variational approximation to expedite convergence and integrate explanation techniques [Ying *et al.*, 2019] to selectively focus on important nodes during the training process, thereby enhancing the efficiency of graph condensation.

Several subsequent studies target at improving GM for graph condensation to enhance the effectiveness of GCond [Gao and Wu, 2023; Yang *et al.*, 2023; Feng *et al.*, 2023; Mao *et al.*, 2023; Li *et al.*, 2023b; Gao *et al.*, 2023; Anonymous, 2024]. Unlike GCond, which uses a single fully connected graph to generate the condensed graph dataset $\mathcal{S}$, MSGC [Gao and Wu, 2023] is introduced to leverage multiple sparse graphs to create diverse neighborhoods for nodes that enhance the capturing of neighborhood information. This, in turn, allows GNNs to generate more informative embeddings in the condensed graphs. Regarding the generalizability of GCond across different GNN architectures, SGDD [Yang *et al.*, 2023] is proposed to explicitly prevent overlooking the original graph dataset structure $\mathbf{A}$ by broadcasting it into the construction of synthetic graph structure $\mathbf{A}'$. In this way, it is shown that SGDD reduces the Laplacian Energy Distribution (LED) [Das *et al.*, 2016; Gutman and Zhou, 2006] shift crossing various datasets significantly compared to GCond. In addition to the node classification task, to validate the effectiveness of SGDD, extensive link prediction problems have been explored. Gao *et al.* [2023] identify the potential issues in existing graph condensation methods for inductive node representation learning and emphasize the under-explored need for an explicit mapping between original and synthetic nodes. Consequently, a GM-based method named MCond was introduced, which explicitly learns a sparse mapping matrix to smoothly integrate new nodes into the synthetic graph for inductive representation learning. MCond employs an alternating optimization scheme compared to GCond, allowing the

synthetic graph and mapping matrix to take turns updating toward dedicated objectives. Furthermore, CTRL [Anonymous, 2024] argues the limited approach of using cosine similarity for gradient matching, leading to biases, and suggests adding gradient magnitudes into the objective function introduced in GCond for a more accurate match. Their empirical findings also show that this approach better aligns frequency distributions between condensed and original graphs.

Despite the effectiveness of the previously mentioned graph condensation methods, Feng *et al.* [2023] recognize that these methods tend to exhibit fairness issues. By identifying the group fairness[2], it demonstrated that as distillation performance increases, fairness (Demographic Parity Δ_{DP}) decreases [Feng *et al.*, 2023]. Particularly, it is showcased that, by measuring the fairness of GNNs trained on original graphs versus those trained on condensed graphs, an improvement in performance correlates with heightened fairness issues in the synthetic condensed graph. To address this challenge, FGD was introduced, as a fair graph condensation method. This was achieved by incorporating the coherence metric into the GM loss function outlined in Eq. (8a). Particularly, the coherence metric is a bias calculator that captures the variance of the estimated sensitive group membership. Similarly, to address the fairness issue of current graph condensation methods, Mao *et al.* [2023] propose graph condensation with Adversarial Regularization (GCARe), which is a method that directly regularizes the condensation process to distill the knowledge of different subgroups fairly into resulting graphs.

Distribution Matching. While GM-based methods offer benefits compared to traditional methods, it faces two challenges. **First**, the condensation process becomes computationally expensive when minimizing the GM loss due to the need for computing second-order derivatives with respect to GNN parameters. **Second**, the architecture-dependent nature of the GM loss may hinder the condensed graph's generalization to new GNN architectures [Liu *et al.*, 2022]. Alternatively, the Distribution Matching (DM) approach seeks to acquire synthetic graph data whose distribution closely approximates that of real data. To address the limitations of GM-based methods, such as the reduced generalizability of graph condensation across different GNN architectures and the computational overhead, DM-based algorithms directly optimize the distance between the two distributions using metrics such as Maximum Mean Discrepancy (MMD) [Zhao and Bilen, 2023]. For example, CaT [Liu *et al.*, 2023c] updates the condensed graph $\mathcal{S}$ using the DM objective function to find the optimal synthetic graph as follows:

$$\ell_{\text{MMD}} = \sum_{c \in \mathcal{C}} r_c \cdot \left\| \text{Mean}\left(\boldsymbol{E}_c\right) - \text{Mean}\left(\tilde{\boldsymbol{E}}_c\right) \right\|^2, \quad (11)$$

where $\mathcal{C}$ is the set of node classes, $\boldsymbol{E}_c$ and $\tilde{\boldsymbol{E}}_c$ are the embeddings of nodes with class c in the original and condensed graph, respectively, and r_c is the class ratio for class c. Likewise, GCDM [Liu *et al.*, 2022] regards the original graph as a distribution of receptive fields and seeks to synthesize a smaller graph whose receptive fields share a similar distribution to that of the original graph. Other works, such as PUMA [Liu *et al.*, 2023d] and FedGKD [Pan *et al.*, 2023], employ a similar approach for various applications in continual learning and federated learning, respectively.

Kernel Ridge Regression Methods

To mitigate heavy computation in the optimization problem in Eq. (7a), KIDD [Xu *et al.*, 2023], the first Kernel Ridge Regression (KRR) method for graph condensation, simplifies the optimization objective into a single-level problem by substituting the closed-form solution of the lower-level problem into the upper-level objective. To implement KRR for graph-level tasks, a graph kernel is essential [Xu *et al.*, 2023]. Thus, a Graph Neural Tangent Kernel (GNTK) [Du *et al.*, 2019] for the KRR graph classifier is chosen, as GNTK effectively characterizes the training dynamics of GNNs and yields such a closed-form solution. Concretely, if $\text{GNN}_{\boldsymbol{\theta}_{\mathcal{S}}}$ in Eq. (7a) is instantiated as the KRR and the squared loss is applied, Eq. (7a) and Eq. (7b) can be instantiated as a single objective function which is as follows:

$$\min_{\mathcal{S}} \mathcal{L}_{\text{KRR}} = \frac{1}{2} \left\| \mathbf{y}_{\mathcal{T}} - \mathbf{K}_{\mathcal{T}\mathcal{S}} \left(\mathbf{K}_{\mathcal{S}\mathcal{S}} + \epsilon \mathrm{I} \right)^{-1} \mathbf{y}_{\mathcal{S}} \right\|^2, \quad (12a)$$

where $\epsilon > 0$ is a KRR hyper-parameter, $\mathbf{K}_{\mathcal{T}\mathcal{S}}$ is the kernel matrix between original and synthetic graphs while $\mathbf{K}_{\mathcal{S}\mathcal{S}}$ is the kernel matrix between synthetic graphs[3]. Also, $\mathbf{y}_{\mathcal{S}}$ and $\mathbf{y}_{\mathcal{T}}$ are the concatenated graph labels from real dataset and synthetic dataset, respectively. Also, Zheng *et al.* [2023] propose SFGC, a structure-free graph condensation method using KRR that only outputs the condensed node features $\mathbf{X}'$, as the structure information of the real graphs is embedded in $\mathbf{X}'$.

Other Methods

Most graph condensation methods involve GNNs or graph filters when generating condensed graphs, which can be biased to a specific spectrum and potentially miss the overall distribution of the real graph [Liu *et al.*, 2023b]. To overcome this, Liu *et al.* [2023b] propose to avoid spectrum bias in the condensation process, which is caused by utilizing GNNs. To obtain representation spaces similar to the ones in the real graph, GCEM [Liu *et al.*, 2023b] matches the representative eigenbasis (the underlying graph structure) of real and synthetic graphs during condensation. Nevertheless, due to the differing sizes of the subspaces defined by the eigenvectors of the real and synthetic graphs, direct alignment is not feasible, prompting GCEM to match the node attributes in the subspaces as an alternative, which will make them share similar distributions:

$$\mathcal{L}_e = \sum_{c=1}^{C} \sum_{k=1}^{K} \left\| \overline{\mathbf{h}}_{c,k} - \overline{\mathbf{h}'}_{c,k} \right\|^2, \quad (13)$$

[2]Group fairness in algorithms ensures unbiased and fair treatment across diverse demographic groups. It seeks to prevent any form of discrimination or bias against specific groups within the algorithmic decision-making process [Mehrabi *et al.*, 2021]

[3]Each kernel indicates infinitely wide multi-layer GNNs trained by gradient descent through squared loss [Du *et al.*, 2019]

where $\mathbf{h}_{c,k}$ and $\mathbf{h'}_{c,k}$ are the representation of the c-th class center in k-th subspace for real and synthetic graphs, respectively.

Gupta *et al.* [2023] investigate that in GM-based methods, the gradients of model weights are contingent on various factors such as the specific GNN architecture and hyperparameters. This leads to a reduction in performance when alternating the GNN during testing. Furthermore, the requirement for the original graph for training in graph condensation still exists, causing computational and storage constraints. With this motivation in mind, MIRAGE [Gupta *et al.*, 2023] is introduced to condense multiple graphs to address graph classification problems. It utilizes GNNs to break down any graph into a collection of computation trees and then extracts frequently co-occurring computation trees from this set. It is shown that a concise set of top-k frequently co-occurring trees can effectively capture a significant portion of the distribution mass while preserving rich information. Consequently, a GNN trained solely on the frequent tree sets should be adequate for subsequent predictive tasks.

DisCo [Xiao *et al.*, 2024] addresses scalability issues in current Matching-based condensation methods through an iterative process that condenses nodes and edges separately. In the node condensation step, synthetic nodes are generated using an MLP pre-trained on $\mathcal{T}$ while considering label distribution. For the synthetic graph structure, edges are predicted using a pre-trained link prediction model, matching the original graph's edge distribution. Notably, DisCo is found to be significantly faster than existing methods because it conducts separate condensation processes for edges and nodes.

4 Applications

While the primary purpose of graph reduction was to enhance the efficiency of graph algorithms, its versatility has led to its advantageous utilization in a range of applications, as will be elaborated upon in this section.

4.1 Neural Architecture Search

Neural architecture search (NAS) [Elsken *et al.*, 2019; Ren *et al.*, 2021] focuses on identifying the most effective architecture from a vast pool of potential models to enhance generalization in a given dataset. This technique is characterized by its intensive computational demands, necessitating the training of numerous architectures on the full dataset and choosing the top performer based on validation results. To address the computational challenge in NAS for GNNs, graph condensation methods are utilized for searching the best GNN architecture [Jin *et al.*, 2022b; Yang *et al.*, 2023]. Specifically, the architectures are trained on the condensed graph which leads to significant speedup in the search process, and a reliable correlation in performance between training on the condensed dataset and the whole dataset is observed. Moreover, Ding *et al.* [2022] introduce a dedicated graph condensation method for NAS, highlighting that traditional graph condensation objectives fall short of achieving this objective due to their lack of generalization across GNNs. Particularly, it proposes a condensation objective for preserving the outcome of hyperparameter optimization and outperforms other graph condensation methods in terms of finding the optimal architecture.

4.2 Continual Graph Learning

Continual learning [De Lange *et al.*, 2021] aims to learn on a stream of data from a stationary data distribution, which requires tackling the issue of catastrophic forgetting, i.e., new data can interfere with current knowledge and erase it. The common strategy is *memory replay* [Kemker and Kanan, 2018], which involves storing representative samples from previous tasks in a buffer to recall knowledge when needed. In the context of graphs, continual learning can be benefited by informative reduced graphs. As introduced in Section 3.3 CaT [Liu *et al.*, 2023c] is applied to continual graph learning by condensing the incoming graph and updating the model with condensed graphs, not the whole incoming graph. To further improve CaT, Liu *et al.* [2023d] introduce PUMA which utilizes pseudo-labeling to incorporate data from unlabeled nodes, boosting the informativeness of the memory bank and addressing the problem of neglected unlabeled nodes. In addition, the sparsification method [Zhang *et al.*, 2023] reduces the number of nodes and edges according to the Ricci curvature evaluation and stores them into a replay buffer for continual learning.

4.3 Visualization & Explanation

The reduced dataset is not only easier to parse by algorithms and computers but also more friendly for people to understand, which leads to the application of visualization and explanation. For instance, Zhao *et al.* [2018] combine spectral graph coarsening method [Loukas and Vandergheynst, 2018] and sparsification methods [Feng, 2016] to develop a nearly linear time algorithm for multilevel graph visualization. *Pyramid Transform* [Shuman *et al.*, 2015] selects nodes corresponding to top-k eigenvector repeatedly and creates a multi-resolution view of large graphs. In addition, k-core decomposition has been widely used for graph visualization and finding specific structural characteristics of networks [Alvarez-Hamelin *et al.*, 2005]. As mentioned in Section 3.2, some coarsening methods like k-snap [Tian *et al.*, 2008] and CANCEL [Zhang *et al.*, 2010] are tailed for customized discovery, which means the user can set the granularity of the graph structure and then find perspectives they are interested in. Graph condensation is also used for visualization and explanation. For instance, GCond [Jin *et al.*, 2022b] observes patterns from original data by visualizing the reduced graph; GDM [Nian *et al.*, 2023] employs condensation to explain GNN behavior during the training process.

4.4 Privacy

It has been empirically investigated that reduced datasets offer good privacy preservation [Bonchi *et al.*, 2014; Dong *et al.*, 2022]. However, the degree of reduction always influences the utility, i.e., preservation of key information. Dataset condensation can be a promising technique to solve the dilemma [Dong *et al.*, 2022]. Under a federated learning framework with particular concerns for local data privacy, FedGKD [Pan *et al.*, 2023] leverages a features extractor that

condenses a small graph in each round. Then, the local models are trained on the condensed graphs before federated aggregation.

4.5 Data Augmentation

Graph data augmentation [Zhao *et al.*, 2022, 2021] is commonly used to enrich the data and improve the model performance. Methods for graph reduction can be employed to generate various perspectives of a graph by repeatedly applying reductions at different ratios, thereby augmenting the data for subsequent models. For example, HARP [Chen *et al.*, 2018] coarsens a graph in a series of levels and then embeds the hierarchy of graphs from the coarsest one to the original. In each step, HARP initializes node embeddings in the finer-level graph using the mapped embeddings from super nodes in the coarser level. By employing various graphs in each iteration, this method augments the training data. Meanwhile, MILE [Liang *et al.*, 2021] enhances this process by substituting the *random walk* employed in HARP with GNNs to improve the embedding quality and efficiency. DistMILE [He *et al.*, 2021] advances the MILE framework by adopting high-performance computing techniques. As a condensation method, MSGC [Gao and Wu, 2023] initializes multiple small graphs by various connection schemes and employs gradient matching to optimize them. This process results in a variety of node embeddings sets, increasing diversity and thereby augmenting the data.

5 Future Work

The field of graph reduction shows considerable potential, with various algorithms already being implemented across different domains. Despite achieving notable performance, current methods in graph reduction still face several challenges and limitations. In this section, we will outline these key challenges.

5.1 Comprehensive Evaluation

Despite the proliferation of graph reduction methods, a significant gap exists in the field concerning the establishment of a comprehensive evaluation methodology for these emerging approaches. As depicted in Table 3, the prevailing focus in existing graph reduction methods has primarily revolved around their ability to preserve specific graph properties or sustain the performance of GNNs on particular downstream tasks. On one hand, the development of novel reduction algorithms should embrace a more inclusive approach, extending to the preservation of a diverse range of graph properties like homophily [Zhu *et al.*, 2020; Gong *et al.*, 2023] and accommodating various downstream tasks, such as node classification, node clustering, link prediction, graph classification, and more. On the other hand, there is an urgent need to broaden the scope of evaluation criteria. This expansion should encompass not only the preservation of multiple graph properties but also cater to various downstream tasks concurrently. By doing so, we can gain valuable insights into the practical utility of reduced graph datasets across different applications and domains.

5.2 Scalability

The majority of sparsification methods have attained linear-time complexity. Similarly, most coarsening methods adopt a local approach by emphasizing smaller neighborhood-level computations to mitigate computational overhead. Consequently, scalability is presently not a prominent concern for these sparsification and coarsening strategies. Conversely, graph condensation methods often come with higher computational costs, involving substantial memory usage and execution time due to their intricate optimization processes. Despite recent research efforts to accelerate graph condensation, the scalability issue persists. This increased computational overhead presents two primary challenges: (1) It becomes increasingly difficult to generate an informative condensed graph with a larger size without significantly increasing computational demands. (2) Applying condensation to large-scale graphs poses computational and resource challenges that require careful consideration and resolution.

5.3 Interpretability of Condensation Process

While graph condensation can itself serve as an explanation or visualization of the original graph, the challenge lies in the interpretability of the condensation process. First, since most condensation methods transform the original one-hot bag-of-words node attributes $\mathbf{X}$ into continuous synthetic node attributes $\mathbf{X}'$, it remains unclear how to interpret the acquired condensed features. Second, there is the question of how to interpret the condensed graph structure. In much of the existing research on condensation, a clear correspondence between synthetic and real nodes is often lacking, giving rise to doubts about how effectively synthetic nodes encapsulate information from their real graph counterparts. One potential approach to addressing this issue is to explore the development of a general framework for enhancing interpretability during the graph condensation process and incorporating GNN interpretability techniques into this endeavor [Ying *et al.*, 2019; Yuan *et al.*, 2020]. Furthermore, it is essential to conduct further theoretical analysis to complement and expand upon the insights presented by Jin *et al.* [2022a].

5.4 Distribution Shift

It is a common observation that GNNs often exhibit poor generalization on a test set when there exists a disparity between the distributions of the training and test sets [Wu *et al.*, 2022a; Li *et al.*, 2022]. When GNNs are trained on reduced graphs and evaluated on graphs from the original distribution, a distribution shift may occur due to the reduction process, which eliminates substantial amounts of graph elements. However, consensus is lacking regarding the definition of graph data distribution or the selection of specific properties to represent the distribution accurately. While several condensation methods utilize one kind of distribution matching as we mentioned in Section 2, other measures of distribution may change after the reduction, e.g., size shift [Buffelli *et al.*, 2022]. Future graph reduction should consider the potential distribution shift issues and preserve properties related to distribution to enhance the generalization of models trained on reduced graphs.

5.5 Robustness

Node attributes after graph reduction risk losing fidelity, posing challenges in distinguishing them from the original graph structure. This makes them susceptible to data poisoning attacks, with injected triggers during the reduction process serving as potential backdoor vulnerabilities as it happens in other data modalities like image [Wang *et al.*, 2018]. To improve the distillation process against such attacks, Tsilivis *et al.* [2022] have combined the KIP (Kernel Including Point) [Nguyen *et al.*, 2020, 2021] method using adversarial training to improve the robustness of the distilled dataset. However, there is a significant gap in systematic evaluation concerning their robustness for graph modality. This oversight extends to a lack of development in both attack strategies and potential defenses tailored to reduced graph structures. It is imperative that future studies investigate these aspects, focusing on the development of methodologies to assess and enhance the robustness of reduced graphs. Exploring these directions will not only provide a deeper understanding of the vulnerabilities inherent in reduced graphs but also lead to the creation of more resilient graph reduction techniques.

6 Conclusion

In this paper, we offer a structured and forward-looking survey of graph reduction. We begin by establishing a formal definition of graph reduction and then develop a detailed hierarchical taxonomy that systematically organizes the diverse methodologies in this area. Our survey divides graph reduction techniques into three primary categories: sparsification, coarsening, and condensation. Each of these groups represents a unique approach to reducing graph complexity while preserving essential properties. Within each category, we systematically delve into the technical intricacies of prominent methods and highlight their practical applications in various real-world scenarios. Moreover, we shed light on the existing challenges within this domain and pinpoint potential directions for future research endeavors. Our aim is to inspire and guide upcoming studies, contributing to the ongoing evolution and advancement of graph reduction methodologies.

References

Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993.

J Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. *Advances in neural information processing systems*, 18, 2005.

Sorour E Amiri, Bijaya Adhikari, Aditya Bharadwaj, and B Aditya Prakash. Netgist: Learning to generate task-based network summaries. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 857–862. IEEE, 2018.

David G Anderson, Ming Gu, and Christopher Melgaard. An efficient algorithm for unweighted spectral graph sparsification. *arXiv preprint arXiv:1410.4273*, 2014.

Anonymous. CTRL: Graph condensation via crafting rational trajectory matching, 2024.

Surender Baswana and Sandeep Sen. A simple linear time algorithm for computing a (2 k—1)-spanner of o (n 1+ 1/k) size in weighted graphs. In *Automata, Languages and Programming: 30th International Colloquium, ICALP 2003 Eindhoven, The Netherlands, June 30–July 4, 2003 Proceedings 30*, pages 384–396. Springer, 2003.

Joshua D Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 255–262, 2009.

Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013.

Maham Anwar Beg, Muhammad Ahmad, Arif Zaman, and Imdadullah Khan. Scalable approximation algorithm for graph summarization. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22*, pages 502–514. Springer, 2018.

Francesco Bonchi, Aristides Gionis, and Tamir Tassa. Identity obfuscation in graphs through the information theoretic lens. *Information Sciences*, 275:232–256, 2014.

Simon Brandeis, Adrian Jarret, and Pierre Sevestre. About graph degeneracy, representation learning and scalability. *arXiv e-prints*, pages arXiv–2009, 2020.

Gecia Bravo Hermsdorff and Lee Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. *Advances in Neural Information Processing Systems*, 32, 2019.

Davide Buffelli, Pietro Liò, and Fabio Vandin. Sizeshiftreg: a regularization method for improving size-generalization in graph neural networks. *Advances in Neural Information Processing Systems*, 35:31871–31885, 2022.

Chen Cai, Dingkang Wang, and Yusu Wang. Graph coarsening with neural networks. In *9th International conference on Learning Representations*, 2021.

Jie Chen and Ilya Safro. Algebraic distance on graphs. *SIAM Journal on Scientific Computing*, 33(6):3468–3490, 2011.

Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International conference on machine learning*, pages 1695–1706. PMLR, 2021.

Fan Chung. Laplacians and the cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005.

Kinkar Ch Das, Seyed Ahmad Mojallal, and Vilmar Trevisan. Distribution of laplacian eigenvalues of graphs. *Linear Algebra and its Applications*, 508:48–61, 2016.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

Charles Dickens, Eddie Huang, Aishwarya Reganti, Jiong Zhu, Karthik Subbian, and Danai Koutra. Graph coarsening via convolution matching for scalable graph neural network training. *arXiv preprint arXiv:2312.15520*, 2023.

Mucong Ding, Xiaoyu Liu, Tahseen Rabbani, Teresa Ranadive, Tai-Ching Tuan, and Furong Huang. Faster hyperparameter search for gnns via calibrated dataset condensation. 2022.

Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *International Conference on Machine Learning*, pages 5378–5396. PMLR, 2022.

Florian Dorfler and Francesco Bullo. Kron reduction of graphs with applications to electrical networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(1):150–163, 2012.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

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 *WWW*. ACM, 2024.

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.

Zhuo Feng. Spectral graph sparsification in nearly-linear time leveraging efficient spectral perturbation analysis. In *Proceedings of the 53rd Annual Design Automation Conference*, pages 1–6, 2016.

Jian Gao and Jianshe Wu. Multiple sparse graphs condensation. *Knowledge-Based Systems*, 278:110904, 2023.

Xinyi Gao, Tong Chen, Yilong Zang, Wentao Zhang, Quoc Viet Hung Nguyen, Kai Zheng, and Hongzhi Yin. Graph condensation for inductive node representation learning. *arXiv preprint arXiv:2307.15967*, 2023.

Alessandro Generale, Till Blume, and Michael Cochez. Scaling r-gcn training with graph summarization. In *Companion Proceedings of the Web Conference 2022*, pages 1073–1082, 2022.

Jiahui Geng, Zongxiong Chen, Yuandou Wang, Herbert Woisetschlaeger, Sonja Schimmler, Ruben Mayer, Zhiming Zhao, and Chunming Rong. A survey on dataset distillation: Approaches, applications and future directions. *arXiv preprint arXiv:2305.01975*, 2023.

Shengbo Gong, Jiajun Zhou, Chenxuan Xie, and Qi Xuan. Neighborhood homophily-based graph convolutional network. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, 2023.

Mridul Gupta, Sahil Manchanda, Sayan Ranu, and Hariprasad Kodamana. Mirage: Model-agnostic graph distillation for graph classification. *arXiv preprint arXiv:2310.09486*, 2023.

Ivan Gutman and Bo Zhou. Laplacian energy of a graph. *Linear Algebra and its applications*, 414(1):29–37, 2006.

Yuntian He, Saket Gurukar, Pouya Kousha, Hari Subramoni, Dhabaleswar K Panda, and Srinivasan Parthasarathy. Distmile: a distributed multi-level framework for scalable graph embedding. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 282–291. IEEE, 2021.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *NeurIPS*, 34, 2021.

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.

Martin Imre, Jun Tao, Yongyu Wang, Zhiqiang Zhao, Zhuo Feng, and Chaoli Wang. Spectrum-preserving sparsification for visualization of big graphs. *Computers & Graphics*, 87:89–102, 2020.

Roberto Interdonato, Matteo Magnani, Diego Perna, Andrea Tagarelli, and Davide Vega. Multilayer network simplification: approaches, models and methods. *Computer Science Review*, 36:100246, 2020.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2016.

Zhihao Jia, Sina Lin, Rex Ying, Jiaxuan You, Jure Leskovec, and Alex Aiken. Redundancy-free computation for graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 997–1005, 2020.

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.

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*, 2022.

Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. Empowering graph representation learning with test-time graph transformation. In *The Eleventh International Conference on Learning Representations*, 2023.

George Karypis and Vipin Kumar. Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1997.

Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018.

Kifayat Ullah Khan, Waqas Nawaz, and Young-Koo Lee. Set-based approximate approach for lossless graph summarization. *Computing*, 97:1185–1207, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

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.

Amy N Langville and Carl D Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3), 2004.

Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. *SIAM Journal on Computing*, 47(6):2315–2336, 2018.

Kristen LeFevre and Evimaria Terzi. Grass: Graph structure summarization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 454–465. SIAM, 2010.

Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987*, 2022.

Gaotang Li, Marlena Duda, Xiang Zhang, Danai Koutra, and Yujun Yan. Interpretable sparsification of brain graphs: Better practices and effective designs for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 1223–1234, New York, NY, USA, 2023. Association for Computing Machinery.

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.

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.

Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *ACM computing surveys (CSUR)*, 51(3):1–34, 2018.

Zirui Liu, Kaixiong Zhou, Fan Yang, Li Li, Rui Chen, and Xia Hu. Exact: Scalable graph neural networks training via extreme activation compression. In *International Conference on Learning Representations*, 2021.

Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. Graph condensation via receptive field distribution matching. *arXiv preprint arXiv:2206.13697*, 2022.

Chuang Liu, Xueqi Ma, Yibing Zhan, Liang Ding, Dapeng Tao, Bo Du, Wenbin Hu, and Danilo P Mandic. Comprehensive graph gradual pruning for sparse training in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Yang Liu, Deyu Bo, and Chuan Shi. Graph condensation via eigenbasis matching. *arXiv preprint arXiv:2310.09202*, 2023.

Yilun Liu, Ruihong Qiu, and Zi Huang. Cat: Balanced continual graph learning with graph condensation. *arXiv preprint arXiv:2309.09455*, 2023.

Yilun Liu, Ruihong Qiu, Yanran Tang, Hongzhi Yin, and Zi Huang. Puma: Efficient continual graph learning with graph condensation. *arXiv preprint arXiv:2312.14439*, 2023.

Oren E Livne and Achi Brandt. Lean algebraic multigrid (lamg): Fast graph laplacian linear solver. *SIAM Journal on Scientific Computing*, 34(4):B499–B522, 2012.

Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pages 3237–3246. PMLR, 2018.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *J. Mach. Learn. Res.*, 20(116):1–42, 2019.

Yao Ma and Jiliang Tang. *Deep learning on graphs*. Cambridge University Press, 2021.

Runze Mao, Wenqi Fan, and Qing Li. Gcare: Mitigating subgroup unfairness in graph condensation through adversarial regularization. *Applied Sciences*, 13(16):9166, 2023.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 419–432, 2008.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050*, 2020.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34:5186–5198, 2021.

Yi Nian, Wei Jin, and Lu Lin. In-process global interpretation for graph learning via distribution matching. *arXiv preprint arXiv:2306.10447*, 2023.

Qiying Pan, Ruofan Wu, Tengfei Liu, Tianyi Zhang, Yifei Zhu, and Weiqiang Wang. Fedgkd: Unleashing the power of collaboration in federated graph neural networks. *arXiv preprint arXiv:2309.09517*, 2023.

Manish Purohit, B Aditya Prakash, Chanhyun Kang, Yao Zhang, and VS Subrahmanian. Fast influence-based coarsening for large networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1296–1305, 2014.

Noam Razin, Tom Verbin, and Nadav Cohen. On the ability of graph neural networks to model interactions between vertices. In *Advances in Neural Information Processing Systems*, 2023.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

Matteo Riondato, David García-Soriano, and Francesco Bonchi. Graph summarization with quality guarantees. *Data mining and knowledge discovery*, 31:314–349, 2017.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Noveen Sachdeva and Julian McAuley. Data distillation: A survey. *arXiv preprint arXiv:2301.04272*, 2023.

Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:1–24, 2015.

Guillaume Salha, Romain Hennequin, Viet Anh Tran, and Michalis Vazirgiannis. A degeneracy framework for scalable graph autoencoders. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3353–3359, 2019.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

Nasrin Shabani, Jia Wu, Amin Beheshti, Jin Foo, Ambreen Hanif, and Maryam Shahabikargar. A survey on graph neural networks for graph summarization. *arXiv preprint arXiv:2302.06114*, 2023.

Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

David I Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2015.

Si Si, Felix Yu, Ankit Singh Rawat, Cho-Jui Hsieh, and Sanjiv Kumar. Serving graph compression for graph neural networks. In *International Conference on Learning Representations*, 2023.

Xiran Song, Jianxun Lian, Hong Huang, Zihan Luo, Wei Zhou, Xue Lin, Mingqi Wu, Chaozhuo Li, Xing Xie, and Hai Jin. xgcn: An extreme graph convolutional network for large-scale social link prediction. In *Proceedings of the ACM Web Conference 2023*, pages 349–359, 2023.

Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 563–568, 2008.

Felipe Petroski Such, Shagan Sah, Miguel Alexander Dominguez, Suhas Pillai, Chao Zhang, Andrew Michael, Nathan D Cahill, and Raymond Ptucha. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):884–896, 2017.

Tomohiro Sugiyama and Kazuhiro Sato. Kron reduction and effective resistance of directed graphs. *SIAM Journal on Matrix Analysis and Applications*, 44(1):270–292, 2023.

Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580, 2008.

Nikolaos Tsilivis, Jingtong Su, and Julia Kempe. Can we achieve robustness from data alone? *arXiv preprint arXiv:2207.11727*, 2022.

Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21, 2023.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Hao Wang, Jiaxin Yang, and Jianrong Wang. Leverage large-scale biological networks to decipher the genetic basis of human diseases using machine learning. *Artificial Neural Networks*, pages 229–248, 2021.

Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. Fast graph condensation with structure-based neural tangent kernel. *Proceedings of the ACM Web Conference*, 2024.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128, 2009.

Ryan Wickman, Xiaofei Zhang, and Weizi Li. A generic graph sparsification framework using deep reinforcement learning. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1221–1226. IEEE, 2022.

Ye Wu, Zhinong Zhong, Wei Xiong, and Ning Jing. Graph summarization for attributed graphs. In *2014 International conference on information science, electronics and electrical engineering*, volume 1, pages 503–507. IEEE, 2014.

Mengmeng Wu, Wanwen Zeng, Wenqiang Liu, Hairong Lv, Ting Chen, and Rui Jiang. Leveraging multiple gene networks to prioritize gwas candidate genes via network representation learning. *Methods*, 145:41–50, 2018.

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.

Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. *arXiv preprint arXiv:2202.02466*, 2022.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

Zhenbang Xiao, Shunyu Liu, Yu Wang, Tongya Zheng, and Mingli Song. Disentangled condensation for large-scale graphs. *arXiv preprint arXiv:2401.12231*, 2024.

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.

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? *arXiv preprint arXiv:2310.09192*, 2023.

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.

Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.

Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the depth and scope of graph neural networks. *Advances in Neural Information Processing Systems*, 34:19665–19679, 2021.

Ning Zhang, Yuanyuan Tian, and Jignesh M Patel. Discovery-driven graph summarization. In *2010 IEEE 26th international conference on data engineering (ICDE 2010)*, pages 880–891. IEEE, 2010.

Xikun Zhang, Dongjin Song, and Dacheng Tao. Ricci curvature-based graph sparsification for continual graph representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6514–6523, 2023.

Zhiqiang Zhao, Yongyu Wang, and Zhuo Feng. Nearly-linear time spectral graph reduction for scalable graph partitioning and data visualization. *arXiv preprint arXiv:1812.08942*, 2018.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 11015–11023, 2021.

Tong Zhao, Wei Jin, Yozen Liu, Yingheng Wang, Gang Liu, Stephan Günneman, Neil Shah, and Meng Jiang. Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871*, 2022.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR, 2020.

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. *arXiv preprint arXiv:2306.02664*, 2023.

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.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020.