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, shengbo.gong, juntong.ni, wei.jin}@emory.edu, wenqi.fan@polyu.edu.hk,
badityap@cc.gatech.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, or graph summarization, has gained prominence for simplifying large graphs while preserving essential properties. In this survey, we 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 associated challenges. Our survey then systematically reviews the technical details of these methods and emphasizes their practical applications across diverse scenarios. Further, we outline critical research directions to ensure the continued effectiveness of graph reduction techniques.

1 Introduction

Graph-structured data has become ubiquitous in various domains, ranging from social networks and biological systems [Wu *et al.*, 2020]. The inherent relational structure of graph data makes it powerful for modeling complex interactions and dependencies. Moreover, with the rise of graph machine learning techniques, especially graph neural networks (GNNs) [Wu *et al.*, 2020], the utilization of graph datasets has seen unprecedented growth, leading to advancements in tasks such as node classification and graph classification.

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], 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**. 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.*, 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].

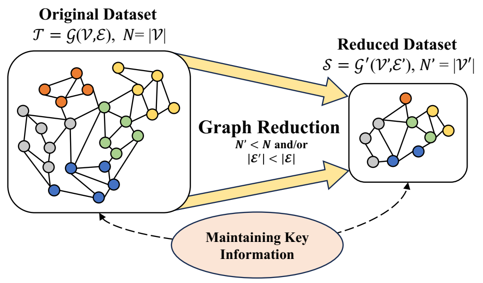
Original Dataset
$\mathcal{T} = \mathcal{G}(\mathcal{V},\mathcal{E})$, $N = |\mathcal{V}|$
Reduced Dataset
$\mathcal{S} = \mathcal{G}'(\mathcal{V}',\mathcal{E}')$, $N' = |\mathcal{V}'|$
Graph Reduction
$N' < N$ and/or $|\mathcal{E}'| < |\mathcal{E}|$
Maintaining Key Information

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.

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 *et. al*, 2012], and the more recent **graph condensation** [Jin *et al.*, 2022b; Jin *et al.*, 2022a; 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 aggregates 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. Our contributions are summarized as:

(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 that focus on one of the graph reduction paradigms [Hu and Lau, 2013; Gao *et al.*, 2024; Xu *et al.*, 2024], our study offers a comprehensive view 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. Comparing with existing surveys on acceleration of graph algorithms [Liu *et al.*, 2022; Zhang *et al.*, 2023], we emphasize that a reduction strategy should produce a static graph to accelerate a wide spectrum of downstream algorithms, rather than accelerating the algorithm itself. 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.

2 Notations and Taxonomy

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.

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 G and G' in terms of certain information:

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

On top of that, we can categorize existing graph reduction techniques into the following three groups:

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 and E' super-edges, where $N' < N$. It generally requires finding a surjective mapping from the original graph to a coarse graph denoted as $\mathbf{C} \in \{0,1\}^{N \times N'}$. We further define the reverse assignment matrix $\mathbf{P} = \text{rowNormalize}(\mathbf{C}^\top)$. Then the coarse graph is usually constructed by $\mathbf{A}' = \mathbf{C}^\top \mathbf{A}\mathbf{C}, \mathbf{X}' = \mathbf{P}\mathbf{X}$.

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

Distinctions. 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. **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 these methods, and we provide a qualitative comparison of the three graph reduction strategies mentioned earlier in Table 1.

3 Methodology

3.1 Graph Sparsification

Graph sparsification involves the selection of crucial edges or nodes based on specific criteria. 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

Traditional sparsification target at preserving essential graph properties including pairwise distances, cuts, and spectrum. These methods usually sample the subgraphs that achieve the minimal loss $\mathcal{L}(G', G)$ iteratively, 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 $D(G', G) =$

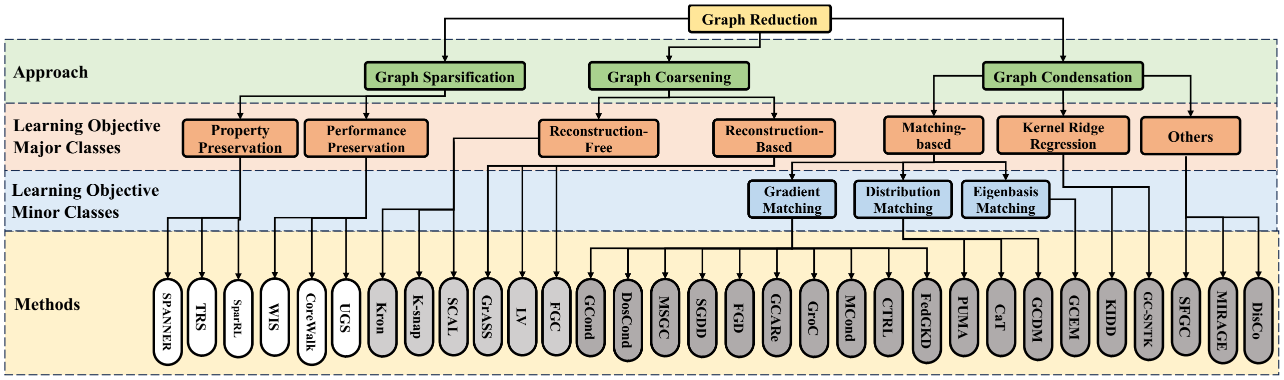

Figure 2: Taxonomy of existing graph reduction methods.

Strategy	Interpretability	Label Utilization	Objective	Output	Remark
Sparsification	✓	×	Property/Performance Preservation	Subgraph	Selects subgraphs with maximal information
Coarsening	✓	×	Property/Performance Preservation	Supergraph	Merges close nodes
Condensation	×	✓	Performance Preservation	Synthetic graph	Generates small graphs with high utility

Table 1: General qualitative comparison of graph reduction methods. "Scalability": the ability to scale up to large graphs, "Interpretability": the existence of correspondence between nodes in the original and reduced graphs, "Label Utilization": the reliance on label information.

$\frac{\mathrm{SP}(G')}{\mathrm{SP}(G)}$ for spanner and $\frac{\mathbf{x}^\top \mathbf{L}'\mathbf{x}}{\mathbf{x}^\top \mathbf{L}\mathbf{x}}$ for sparsifier. $\mathrm{SP}(G)$ 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.

Spanner. [Althöfer *et al.*, 1993] first develop an algorithm called SPANNER 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 between their connected nodes in the reduced graph. [Baswana and Sen, 2003] tighten this upper bound with a linear time algorithm that merely explores the edges in the neighborhood of a node. Also, SparRL [Wickman *et al.*, 2022] designs a reinforcement learning process and adjusts reward functions to preserve pairwise distance.

Sparsifier. One representative sparsifier is called *Twice Ramanujan Sparsifier* (TRS) [Batson *et al.*, 2009], which proves that for 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 derives 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. Other methods do not merge the nodes themselves; instead, they merge the paths between each pair of target nodes, resulting in a reweighted edge. Kron reduction [Dorfler *et. al*, 2012] is initially developed to address challenges in electrical networks, specifically to simplify resistance networks while maintaining the pairwise ER. This method selects nodes and calculates the coarse graph Laplacian $\mathbf{L}'$ by computing *Schur complement* of unselected nodes. Recently, [Sugiyama and Sato, 2023] extend it to a directed graph with self-loop. In contrast to selecting nodes arbitrarily, [Fang *et al.*, 2010] calculate *Schur complement* after finding the largest node set consisting of nodes not adjacent to each other.

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] 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. Furthermore, recent work WIS [Razin *et al.*, 2023] highlights that the ability to model node interactions is primarily determined by the metrics *walk index* (WI), i.e., the number of walks originating from the boundary of the partition. Then, important edges are selected 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 witnessed many model-based scoring methods, which utilize a parameterized model to calculate the score. For instance, [Jin *et al.*, 2022b] adopt coreset methods [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 score the nodes and edges. IGS [Li *et al.*, 2023] 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 [Jin *et al.*, 2023] which iteratively optimizes the sparsification process by interaction with GNNs. For example, UGS [Chen *et al.*, 2021] simultaneously prunes the graph adjacency matrix and the GNN weights. Recognizing that UGS fails to preserve topology, GST [Zhang *et al.*, 2024a] is proposed to combine semantic and topological information during sparsification. However, these structure learning methods are closely tied to specific selection models or sparsified GNN models. The transferability of their sparsified graphs to other graph algorithms remains inadequately explored.

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 usually 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. These coarsening techniques can be broadly categorized into spatial or spectral coarsening methods, depending on whether they try to reconstruct the adjacency or Laplacian matrix.

Spatial coarsening. Spatial coarsening adopts the Reconstruction Error (RE) [LeFevre and Terzi, 2010] as the objective function $\mathcal{L} := RE_p(\mathbf{A}_l|\mathbf{A}) = ||\mathbf{A}_l - \mathbf{A}||_{\mathrm{F}}^p$, where the *lifted* adjacency matrix $\mathbf{A}_l$ 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. Similarly, [Beg *et al.*, 2018] propose a weighted sampling scheme to sample vertices for merging that will result in the least RE.

Spectral coarsening. Spectral coarsening methods [Purohit *et al.*, 2014] usually 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 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 pairwise contraction* approaches, where different node pair scoring methods can be used including Heavy Edge [Dhillon *et al.*, 2007], Algebraic Distance [Chen and Safro, 2011] and Local Variation (LV) [Loukas and Vandergheynst, 2018]. Aside from these heuristics, [Zhao *et al.*, 2018] scale the edge weights by stochastic gradient descent to further align the eigenvalues after coarsening. FGC [Kumar *et al.*, 2023] takes both the graph structure and the node attributes as the input and alternatively optimizes $\mathbf{C}$ and $\mathbf{X}'$.

Reconstruction-Free Methods

Despite the proliferation of reconstruction-based methods, other approaches do not rely on the reconstruction while still keeping the key information. Some methods attempt to find the most informative summaries of a network. To analyze social networks with diverse attributes and relations, k-SNAP [Tian *et al.*, 2008] produces a summary graph where every node inside a super node has the same values for selected categorical attributes, and is adjacent to nodes with the same selected relations. To extend the above methods focusing on only one task, Netgist [Amiri *et al.*, 2018] defines a task-based graph summarization problem and uses RL to learn node merging policies.

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 coarse graph, with super node label $\mathbf{Y}' = \mathrm{argmax}(\mathbf{PY})$ and directly uses this model to inference. [Buffelli *et al.*, 2022] match the node embeddings output by GNNs among graphs in different coarsening ratios.

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 [Geng *et al.*, 2023], 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:

$$\mathcal{S} = \arg\min_{\mathcal{S}} \mathcal{L}(\mathrm{GNN}_{\boldsymbol{\theta}_\mathcal{S}}(\mathcal{T}), \mathrm{GNN}_{\boldsymbol{\theta}_\mathcal{T}}(\mathcal{T})), \tag{2}$$

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. (2) can be rewritten as the following bi-level problem that generalizes to the distribution of random initialization $P_{\boldsymbol{\theta}_0}$:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[\mathcal{L}\left(\mathrm{GNN}_{\boldsymbol{\theta}_{\mathcal{S}}}(\mathbf{A}, \mathbf{X}), \mathbf{Y}\right)\right], \tag{3a}$$

$$\text{s.t.} \quad \boldsymbol{\theta}_{\mathcal{S}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\mathrm{GNN}_{\boldsymbol{\theta}(\boldsymbol{\theta}_0)}\left(\mathbf{A}', \mathbf{X}'\right), \mathbf{Y}'\right), \tag{3b}$$

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. (3a) and (3b), [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:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[\sum_{t=0}^{T-1} D\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{S}}, \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}}\right)\right], \tag{4a}$$

where $D(\cdot, \cdot)$ represents a distance function, T stands for the total number of steps in the entire training trajectory, $\boldsymbol{\theta}_t$ refers to the model parameters at t-th training epoch, and $\mathcal{L}_{\mathcal{S}}$ and $\mathcal{L}_{\mathcal{T}}$ are cross-entropy loss functions over synthetic and real datasets, respectively. 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 $\mathbf{A}'$:

$$\mathbf{A}'_{ij} = \sigma\left(\left[\mathrm{MLP}_{\Phi}\left(\left[\mathbf{x}'_i; \mathbf{x}'_j\right]\right) + \mathrm{MLP}_{\Phi}\left(\left[\mathbf{x}'_j; \mathbf{x}'_i\right]\right)\right]/2\right), \tag{5}$$

where MLP_{Φ} is a multi-layer perceptron parameterized with Φ and $[.;.]$ indicates concatenation. However, the optimization process in GCond involves a nested loop as shown in Eq. (4a), 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. (4a), the objective function of DosCond becomes: $\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[D\left(\nabla_{\theta} \mathcal{L}_{\mathcal{S}}, \nabla_{\theta} \mathcal{L}_{\mathcal{T}}\right)\right]$. 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 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. Unlike GCond, which uses a single fully connected graph to generate the condensed graph dataset, 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 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 $\mathbf{A}'$. [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 is 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 [Zhang *et al.*, 2024b] argues that using cosine similarity for gradient matching leads 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 [Mehrabi *et al.*, 2021], it demonstrates 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 is introduced, as a fair graph condensation method. This is achieved by incorporating the coherence metric into the GM loss function outlined in Eq. (4a). 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. Particularly, DM-based algorithms directly optimize the distance between the two distributions using metrics such as Maximum Mean Discrepancy (MMD). For example, CaT [Liu *et al.*, 2023b] updates the condensed graph $\mathcal{S}$ using the DM objective function to find the optimal synthetic graph as follows:

$$\ell_{\text{MMD}} = \sum\nolimits_{c \in \mathcal{C}} r_c \cdot \left\| \text{Mean}\left(\boldsymbol{E}_c\right) - \text{Mean}\left(\tilde{\boldsymbol{E}}_c\right) \right\|^2, \quad (6)$$

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. Other works, such as PUMA [Liu *et al.*, 2023c], employ a similar approach for various applications like continual learning. Another work DisCo [Xiao *et al.*, 2024] is proposed to address scalability issues in current matching-based condensation methods by condensing nodes and edges separately. It is found to be significantly faster than existing methods because it conducts separate condensation processes for edges and nodes by preserving their distributions.

Kernel Ridge Regression Methods

To mitigate heavy computation in the optimization problem in Eq. (3a), 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. (3a) is instantiated as the KRR and the squared loss is applied, Eq. (3a) and Eq. (3b) can be instantiated as a single objective function which is as follows:

$$\min_{\mathcal{S}} \mathcal{L}_{\text{KRR}} = \frac{1}{2} \left\| \mathbf{y}_{\mathcal{T}} - \mathbf{K}_{\mathcal{TS}} \left(\mathbf{K}_{\mathcal{SS}} + \epsilon \mathrm{I}\right)^{-1} \mathbf{y}_{\mathcal{S}} \right\|^2, \quad (7a)$$

where $\epsilon > 0$ is a KRR hyper-parameter, $\mathbf{K}_{\mathcal{TS}}$ is the kernel matrix between original and synthetic graphs and $\mathbf{K}_{\mathcal{SS}}$ is the kernel matrix between synthetic graphs[1]; $\mathbf{y}_{\mathcal{S}}$ and $\mathbf{y}_{\mathcal{T}}$ are the concatenated graph labels from real dataset and synthetic dataset, respectively.

Other Methods

In this subsection, we delve into alternative methods for graph condensation, separate from techniques aligned with specific approaches. For example, [Liu *et al.*, 2023a] found that typical graph condensation methods use GNNs or graph filters, leading to spectrum bias. They propose GCEM to avoid this bias by matching eigenbases of real and synthetic graphs during condensation. Since direct alignment isn't feasible due to differing subspace sizes, GCEM matches node attributes instead, ensuring similar distributions: $\mathcal{L}_e = \sum_{c=1}^{C} \sum_{k=1}^{K} \left\| \overline{\mathbf{h}}_{c,k} - \overline{\mathbf{h}'}_{c,k} \right\|^2$, 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.

[Zheng *et al.*, 2023] propose SFGC, a structure-free graph condensation method using a matching-based approach that only outputs the condensed node features $\mathbf{X}'$, as the structure information of the real graphs is embedded in $\mathbf{X}'$. Concretely, unlike gradient matching-based methods, SFGC aligns their long-term GNN training trajectories using an offline expert parameter distribution as guidance.

In addition, 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.

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.

Neural Architecture Search. Neural architecture search (NAS) 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]. The architectures trained on condensed graphs significantly speed up the search process, showing a reliable performance correlation between training on condensed and full datasets. [Ding *et al.*, 2022] present a graph condensation method for NAS, indicating that traditional objectives don't generalize well across GNNs. Their method focuses on preserving hyperparameter optimization outcomes, surpassing other condensation techniques in identifying the optimal architecture.

Continual Graph Learning. The common strategy in continual learning (CL) 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, CL can be benefited by informative reduced graphs. As introduced in Section 3.3, CaT [Liu *et al.*, 2023b] 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.*, 2023c] introduce PUMA which utilizes pseudo-labeling to incorporate data from unlabeled nodes, boosting the informativeness of the buffer and addressing the problem of neglected unlabeled nodes. In addition, the sparsification method [Zhang *et al.*, 2023] reduces the number of

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

nodes and edges according to the Ricci curvature evaluation and stores them into a replay buffer for CL.

Visualization & Explanation. [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. GDM [Nian *et al.*, 2023] employs graph condensation to explain GNN behavior during the training process.

Privacy. It has been empirically investigated that reduced datasets offer good privacy preservation [Dong *et al.*, 2022]. In the context of a federated learning framework that prioritizes local data privacy, the method introduced by FedGKD [Pan *et al.*, 2023] stands out. This approach employs a feature extractor to condense a small graph during each round of the learning process. Following this condensation, the local models are trained using these condensed graphs. Once this training is complete, the models proceed to the federated aggregation phase. This process ensures that sensitive data remains secure while still allowing for effective training across multiple decentralized devices.

Data Augmentation. 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. 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 embedding sets, increasing diversity and thereby augmenting the data.

5 Conclusion and Future Work

In conclusion, this paper presents a detailed survey of graph reduction, covering definitions, taxonomy, and various techniques. Now we further highlight the current research landscape in this field and suggest potential future research paths.

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. The prevailing focus in existing graph reduction methods has primarily resolved around the 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 and accommodating various downstream tasks. On the other hand, there is an urgent need to broaden the scope of evaluation criteria. This expansion should simultaneously encompass the preservation of multiple graph properties and cater to various downstream tasks. By doing so, valuable insights into the practical utility of reduced graph datasets across different applications and domains can be obtained.

Scalability. Despite recent research efforts to accelerate graph condensation, the scalability issue persists. This increased computational overhead presents two primary challenges: (1) Generating informative condensed graphs of larger sizes demands significantly more computation. (2) Condensing large-scale graphs presents computational and resource challenges that need careful resolution.

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. One potential approach to addressing these 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].

Distribution Shift. Training GNNs on reduced graphs and evaluating them on original distribution graphs can cause a distribution shift due to the elimination of many graph elements. However, there is no consensus on defining graph data distribution or selecting properties to accurately represent it. 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. Future graph reduction should consider the potential distribution shift issues and preserve more information related to distribution to enhance the generalization of models trained on reduced graphs.

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]. 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. Future studies must investigate these aspects, focusing on the development of methodologies to assess and enhance the robustness of reduced graphs.

Acknowledgments. This paper is based on work supported in part by the funds from Emory University and Georgia Tech, NSF (Expeditions CCF-1918770, CAREER IIS-2028586, Medium IIS-1955883, Medium IIS-2106961, PIPP CCF-2200269), CDC MInD program, and Meta faculty gift.

References

[Althöfer *et al.*, 1993] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 1993.

[Amiri *et al.*, 2018] Sorour E Amiri, Bijaya Adhikari, Aditya Bharadwaj, and B Aditya Prakash. Netgist: Learning to generate task-based network summaries. In *ICDM*, 2018.

[Baswana and Sen, 2003] 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 *ICALP*, 2003.

[Batson *et al.*, 2009] Joshua D Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. In *STOC*, 2009.

[Beg *et al.*, 2018] Maham Anwar Beg, Muhammad Ahmad, Arif Zaman, and Imdadullah Khan. Scalable approximation algorithm for graph summarization. In *PAKDD*, 2018.

[Chen and Safro, 2011] Jie Chen and Ilya Safro. Algebraic distance on graphs. *SIAM Journal on Scientific Computing*, 2011.

[Chen *et al.*, 2018] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *AAAI*, 2018.

[Chen *et al.*, 2021] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *ICML*, 2021.

[Dhillon *et al.*, 2007] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *TPAMI*, 2007.

[Dong *et al.*, 2022] Tian Dong, Bo Zhao, and Lingjuan Lyu. Privacy for free: How does dataset condensation help privacy? In *ICML*, 2022.

[Fang *et al.*, 2010] Haw-ren Fang, Sophia Sakellaridi, and Yousef Saad. Multilevel manifold learning with application to spectral clustering. In *CIKM*, 2010.

[Fang *et al.*, 2024] 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*, 2024.

[Feng *et al.*, 2023] Qizhang Feng, Zhimeng Jiang, Ruiquan Li, Yicheng Wang, Na Zou, Jiang Bian, and Xia Hu. Fair graph distillation. In *NeurIPS*, 2023.

[Feng, 2016] Zhuo Feng. Spectral graph sparsification in nearly-linear time leveraging efficient spectral perturbation analysis. In *DAC*, 2016.

[Gao and Wu, 2023] Jian Gao and Jianshe Wu. Multiple sparse graphs condensation. *Knowledge-Based Systems*, 2023.

[Gupta *et al.*, 2023] Mridul Gupta, Sahil Manchanda, Sayan Ranu, and Hariprasad Kodamana. Mirage: Model-agnostic graph distillation for graph classification. *arXiv*, 2023.

[Hu and Lau, 2013] Pili Hu and Wing Cheong Lau. A survey and taxonomy of graph sampling. *arXiv*, 2013.

[Hu *et al.*, 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.

[Huang *et al.*, 2021] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *KDD*, 2021.

[Jia *et al.*, 2020] Zhihao Jia, Sina Lin, Rex Ying, Jiaxuan You, Jure Leskovec, and Alex Aiken. Redundancy-free computation for graph neural networks. In *KDD*, 2020.

[Jin *et al.*, 2022a] Wei Jin, Xianfeng Tang, Haoming Jiang, Zheng Li, Danqing Zhang, Jiliang Tang, and Bing Yin. Condensing graphs via one-step gradient matching. In *KDD*, 2022.

[Jin *et al.*, 2022b] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *ICLR*, 2022.

[Jin *et al.*, 2023] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. Empowering graph representation learning with test-time graph transformation. In *ICLR*, 2023.

[Kemker and Kanan, 2018] Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *ICLR*, 2018.

[LeFevre and Terzi, 2010] Kristen LeFevre and Evimaria Terzi. Grass: Graph structure summarization. In *SDM*, 2010.

[Liu *et al.*, 2022] Mengyang Liu, Shanchuan Li, Xinshi Chen, and Le Song. Graph condensation via receptive field distribution matching. *arXiv*, 2022.

[Liu *et al.*, 2023a] Yang Liu, Deyu Bo, and Chuan Shi. Graph condensation via eigenbasis matching. *arXiv*, 2023.

[Liu *et al.*, 2023b] Yilun Liu, Ruihong Qiu, and Zi Huang. Cat: Balanced continual graph learning with graph condensation. *arXiv*, 2023.

[Liu *et al.*, 2023c] Yilun Liu, Ruihong Qiu, Yanran Tang, Hongzhi Yin, and Zi Huang. Puma: Efficient continual graph learning with graph condensation. *arXiv*, 2023.

[Loukas and Vandergheynst, 2018] Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *ICML*, 2018.

[Mehrabi *et al.*, 2021] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys*, 2021.

[Nian *et al.*, 2023] Yi Nian, Wei Jin, and Lu Lin. In-process global interpretation for graph learning via distribution matching. *arXiv*, 2023.

[Pan *et al.*, 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*, 2023.

[Razin *et al.*, 2023] Noam Razin, Tom Verbin, and Nadav Cohen. On the ability of graph neural networks to model interactions between vertices. In *NeurIPS*, 2023.

[Sachdeva and McAuley, 2023] Noveen Sachdeva and Julian McAuley. Data distillation: A survey. *arXiv*, 2023.

[Sener and Savarese, 2018] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks:a core-set approach. In *ICLR*, 2018.

[Shuman *et al.*, 2015] David I Shuman, Mohammad Javad Faraji, and Pierre Vandergheynst. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 2015.

[Spielman and Srivastava, 2008] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *STOC*, 2008.

[Sugiyama and Sato, 2023] Tomohiro Sugiyama and Kazuhiro Sato. Kron reduction and effective resistance of directed graphs. *SIAM Journal on Matrix Analysis and Applications*, 2023.

[Batson *et al.*, 2013] Batson *et al.* Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 2013.

[Buffelli *et al.*, 2022] Buffelli *et al.* Sizeshiftreg: a regularization method for improving size-generalization in graph neural networks. *NeurIPS*, 2022.

[Ding *et al.*, 2022] Ding *et al.* Faster hyperparameter search for gnns via calibrated dataset condensation. *arXiv*, 2022.

[Dorfler *et. al*, 2012] Dorfler *et. al.* Kron reduction of graphs with applications to electrical networks. *IEEE TCS-I*, 2012.

[Du *et al.*, 2019] Du *et al.* Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *NeurIPS*, 2019.

[Gao *et al.*, 2023] Gao *et al.* Graph condensation for inductive node representation learning. *arXiv*, 2023.

[Gao *et al.*, 2024] Gao *et al.* Graph condensation: A survey. *arXiv*, 2024.

[Geng *et al.*, 2023] Geng *et al.* A survey on dataset distillation: Approaches, applications and future directions. *arXiv*, 2023.

[Imre *et al.*, 2020] Imre *et al.* Spectrum-preserving sparsification for visualization of big graphs. *Computers & Graphics*, 2020.

[Kumar *et al.*, 2023] Kumar *et al.* Featured graph coarsening with similarity guarantees. In *ICML*, 2023.

[Li *et al.*, 2023] Li *et al.* Interpretable sparsification of brain graphs: Better practices and effective designs for graph neural networks. In *KDD*, 2023.

[Liu *et al.*, 2022] Liu *et al.* Survey on graph neural network acceleration: An algorithmic perspective. In *IJCAI*, 2022.

[Mao *et al.*, 2023] Mao *et al.* Gcare: Mitigating subgroup unfairness in graph condensation through adversarial regularization. *Applied Sciences*, 2023.

[Purohit *et al.*, 2014] Purohit *et al.* Fast influence-based coarsening for large networks. In *KDD*, 2014.

[Riondato *et al.*, 2017] Riondato *et al.* Graph summarization with quality guarantees. *Data mining and knowledge discovery*, 2017.

[Salha *et al.*, 2019] Salha *et al.* A degeneracy framework for scalable graph autoencoders. In *IJCAI*, 2019.

[Xu *et al.*, 2023] Xu *et al.* Kernel ridge regression-based graph dataset distillation. In *KDD*, 2023.

[Yang *et al.*, 2023] Yang *et al.* Does graph distillation see like vision dataset counterpart? *arXiv*, 2023.

[Ying *et al.*, 2019] Ying *et al.* Gnnexplainer: Generating explanations for graph neural networks. *NeurIPS*, 2019.

[Zhang *et al.*, 2023] Zhang *et al.* A survey on graph neural network acceleration: Algorithms, systems, and customized hardware. *arXiv*, 2023.

[Zhang *et al.*, 2024a] Zhang *et al.* Two heads are better than one: Boosting graph sparse training via semantic and topological awareness. *arXiv*, 2024.

[Zhang *et al.*, 2024b] Zhang *et al.* Two trades is not baffled: Condense graph via crafting rational gradient matching. *arXiv*, 2024.

[Zheng *et al.*, 2023] Zheng *et al.* Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *NeurIPS*, 2023.

[Tian *et al.*, 2008] Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. Efficient aggregation for graph summarization. In *SIGMOD*, 2008.

[Wang *et al.*, 2018] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv*, 2018.

[Wickman *et al.*, 2022] Ryan Wickman, Xiaofei Zhang, and Weizi Li. A generic graph sparsification framework using deep reinforcement learning. In *ICDM*, 2022.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *TNNLS*, 2020.

[Xiao *et al.*, 2024] Zhenbang Xiao, Shunyu Liu, Yu Wang, Tongya Zheng, and Mingli Song. Disentangled condensation for large-scale graphs. *arXiv*, 2024.

[Xu *et al.*, 2024] Hongjia Xu, Liangliang Zhang, Yao Ma, Sheng Zhou, Zhuonan Zheng, and Bu Jiajun. A survey on graph condensation. *arXiv*, 2024.

[Yuan *et al.*, 2020] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *KDD*, 2020.

[Zhang *et al.*, 2023] Xikun Zhang, Dongjin Song, and Dacheng Tao. Ricci curvature-based graph sparsification for continual graph representation learning. *TNNLS*, 2023.

[Zhao *et al.*, 2018] Zhiqiang Zhao, Yongyu Wang, and Zhuo Feng. Nearly-linear time spectral graph reduction for scalable graph partitioning and data visualization. *arXiv*, 2018.