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

Mohammad Hashemi†1, Shengbo Gong†1, Juntong Ni†1, Weni Fan2, B. Aditya Prakash3, Wei Jin1

1Emory University, 2The Hong Kong Polytechnic University, 3Georgia Institute of Technology

mohammad.hashemi@emory.edu, shengbo.gong@emory.edu, juntong.ni@emory.edu, weni.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/Emory-Melody/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; Ma and Tang, 2021], the utilization of graph datasets has seen unprecedented growth, leading to advancements in fields such as chemistry [Reiser et al., 2022], biology [Gligorijević et al., 2021], and epidemiology [Liu et al., 2024].

Fig 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† [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., 2021].

†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.


**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 $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathbf{X} = [x_1, x_2, ..., x_N]$ 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 $D_{ii} = \sum_j A_{ij}$. 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.

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

### 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:

$$G' = \arg \min_{G'} \mathcal{L}(G, G').$$ (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 include discussions on graph reduction but we offer a more detailed and thorough examination of reduction techniques. Furthermore, our work shares connections with recent surveys on dataset distillation [Geng et al., 2023; Sachdeva and McAuley, 2023], while they majorly focus on condensation methods applied to image data.

---

[2222] 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., 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 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; Chen et al., 2022], 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. Compared with the latest survey on graph learning acceleration [Zhang et al., 2023a] which also categorizes graph reduction algorithms into those as mentioned above three, our work uses more general definitions and discusses more related methods on each category in a wider scope. Additionally, our survey explores synergies between graph reduction and GNNs, an aspect rarely covered in existing surveys. Also, some data-centric graph learning surveys [Zha et al., 2023; Zheng et al., 2023a] include discussions on graph reduction but we offer a more detailed and thorough examination of reduction techniques. Furthermore, our work shares connections with recent surveys on dataset distillation [Geng et al., 2023; Sachdeva and McAuley, 2023], while they majorly focus on condensation methods applied to image data.
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:

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

**Definition of Graph Coarsening** Given a graph $G = (A, X)$, graph coarsening outputs $G' = (A', X')$ containing $N'$ super-nodes 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 $C \in \{0, 1\}^{N \times N'}$ that assigns nodes to super-nodes. We further define the reverse assignment matrix $P = \text{rowNormalize}(C^\top)$.

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

**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 3, 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

#### 3.1 Graph Sparsification

Graph sparsification, as an intuitive method for graph reduction, involves selecting essential edges or nodes based on specific criteria and then constructing a smaller graph from these selected elements. 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 $\epsilon$-spanner/sparsifier. Specifically, $\mathcal{L}(G', G)$ is expressed as $|D(G', G) - 1|$ with $D(\cdot, \cdot)$ defined as follows:

$$
D(G', G) = \begin{cases} 
\frac{\text{SP}(G', G)}{\text{SP}(G)} & \text{for spanner,} \\
\frac{1}{\text{LP}(G')} & \text{for sparsifier,}
\end{cases}
$$

(2)

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

**Spanners:** Althöfer et al. [1993] first develop an algorithm named 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. 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] presents 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. GSGAN [Wu and Chen, 2020] designs a reward function to guide the graph generator in a generative adversarial network creating random walks. These random walks are finally combined to form a smaller edge set that is effective for community detection.

**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, CoreWalk [Brandes et al., 2020] utilizes this framework for reducing the graph but generate the node embeddings with different methods. Furthermore, recent work WIS [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. For example, 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 methods fall under graph structure learning [Zhu et al., 2021], which can be viewed as renders the scoring of edges learnable. For example, Zheng et al. [2020] learn a sparsified graph structure by neighbor sampling according to the and reparameterization trick [Jang et al., 2016] during GNN training; UGS [Chen et al., 2021] and CGP [Liu et al., 2023a] simultaneously prune the elements in graph adjacency matrix (edges) and the GNN weights to reduce the graph and tighten the model. Differing from the iterative approaches described above, TEDDY [Seo et al., 2024] introduces a one-shot reduction strategy. This strategy begins by sparsifying the graph using only degree information, followed by training a lightweight model through knowledge distillation and Projected Gradient Descent (PGD) on the $l_0$ ball. Although these graphs are merely byproducts of graph structure learning, they have the potential to serve as reduced inputs for other algorithms.

However, there are a few works that do not fall into the scoring-and-selection strategy and instead adopt a holistic approach to selection. For example, OTC [Garg and Jaakkola, 2019] achieves selection by minimizing the optimal transport distance between an initial node distribution supported on $V$ and a target distribution supported on a subset $V'$ selected via the associated transport problem [Peyré et al., 2019]. The reduced graph is then obtained by restricting the original graph.

Table 2: 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 the label information.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Interpretability</th>
<th>Label Utilization</th>
<th>Objective</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparsification</td>
<td>✓</td>
<td>×</td>
<td>Property Preservation</td>
<td>Sampled graph</td>
</tr>
<tr>
<td>Coarsening</td>
<td>✓</td>
<td>×</td>
<td>Property/Performance Preservation</td>
<td>Supergraph</td>
</tr>
<tr>
<td>Condensation</td>
<td>×</td>
<td>✓</td>
<td>Performance Preservation</td>
<td>Synthetic graph</td>
</tr>
</tbody>
</table>
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 (C or 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 $C$:

$$ RE_{p}(A_l | A) = ||A_l - A||_P^p $$

where the lifted adjacency matrix $A_l$ [LeFevre and Terzi, 2010] is usually defined as:

$$ A_l(u, v) = \begin{cases} 0 & \text{if } u = v \\ E_i / \sqrt{N_i} & \text{if } u, v \in V_i \\ E_{ij} / \sqrt{N_i N_j} & \text{if } u \in V_i, v \in V_j \end{cases} $$

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 $A_l$ can be expressed as a function of $P$ and $A$ [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 $L_i$ and $L$ by comparing their eigenvalues or eigenvectors. The lifted Laplacian matrix is defined as $L = P^\top L P$ [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 $REE = \sum_{i=1}^{k} |\lambda_i - \lambda'_i|/\lambda_i$, where $\lambda_i$ and $\lambda'_i$ are the top-$k$ eigenvalues of the matrices $L$ and $L'$, respectively. Note that they use $L'$ instead of $L$ 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], Affinity [Livne and Brandt, 2012] and Local Variation (LV) [Loukas, 2019]. Some works hold that the edge weights can be further optimized after pairwise 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 pairwise 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 $C$ and $X'$. SGC [Bravo Herrmsdorf and Gunderson, 2019] considers the edge sparsification and contraction as edge weights of 0 and $\infty$. Then they develop a probabilistic framework to preserve the pseudo-inverse of graph Laplacian $L^+$ by $L_i^+$. GOREN [Cai et al., 2021] learns the edge weights in the coarse graph by a GNN with the loss to preserve the graph Laplacian $L$ by $L_i$. 

Figure 3: Taxonomy of existing graph reduction methods.
Reconstruction-Free Methods

Despite the proliferation of reconstruction-based methods, other approaches do not rely on the reconstruction while still keeping the key information. [Itzkovitz et al., 2005] coarsen complex engineered and biological networks into smaller and more comprehensible versions, where nodes represent probabilistically generalized network motifs in the original 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] relaxes this condition to continual ones with adaptive cutoffs and proposes a comprehensive metric named interestingness. To extend the above methods focusing only on 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 both influence maximization and propagation patterns detection tasks. To flexibly achieve higher performance among different tasks, 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 \( Y' = \operatorname{argmax}(PY) \) 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 [Dicksens et al., 2023] merges nodes that are equivalent or similar w.r.t. the GNN convolution operation. Similarly, Buffelli et al. [2022] match the node embeddings output by GNNs among graphs in different coarsening ratios. Many works employ coarsening to improve the scalability when training GNNs in large-scale graphs. VNG [Si et al., 2023] highlights the ongoing challenge of efficiently deploying a GNN model in online applications when edges exist between testing nodes and training nodes. To tackle this issue, they match the forward propagation to obtain the mapping matrix \( C \) by applying weighted k-means. Recognizing that only a subset of nodes (target nodes) requires analysis in large-scale web graphs, Graph Skeleton [Cao et al., 2024] focuses reducing the background nodes while preserving the classification performance of target nodes. It merges background nodes sharing the same distance to target nodes, as well as target nodes and their affiliated nodes—that is, background nodes linked exclusively to a single target node.

Other methods do not merge the nodes but the paths between each pair of target nodes. For example, 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 \( L' \) by Schur complement

\[
L' = L_{\psi'} - L_{\psi'} L_{\psi'}^{-1} (L_{\psi'} L_{\psi'})^{-1}
\]

(5)

where \( \psi' \) denotes the selected index from \( \psi \), \( \psi = \psi' \cup \psi'' \), and \( L_{A,B} \) is the submatrix of \( L \) whose row index is \( A \) and column index is \( B \). Recently, Sugiya and Sat [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.

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 pairwise 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 [Titsulin 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 \( T \) and the synthetic graph \( S \). Thus, the objective function in Eq. (1) can reformulated as follows:

\[
S = \arg \min_S \mathcal{L}(\text{GNN}_{\theta_{S}}(T), \text{GNN}_{\theta_{T}}(T)),
\]

(6)

where \( \text{GNN}_{\theta_{S}} \) and \( \text{GNN}_{\theta_{T}} \) denote the GNN models trained on \( S \) and \( T \), respectively; \( \mathcal{L} \) represents the loss function used
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. A denotes that the method can only applied to a symmetric adjacency matrix. \( \tilde{A} \) indicates the adjacency matrix with multiple relations.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Method</th>
<th>Learning Objective</th>
<th>Evaluation</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Sparsification</td>
<td>SPANNER [Althöfer et al., 1993]</td>
<td>Property Preservation</td>
<td>Value of ( \epsilon )</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>TRS [Batson et al., 2009]</td>
<td></td>
<td>Value of ( \epsilon )</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>SparRL [Wickman et al., 2022]</td>
<td></td>
<td>Value of ( \epsilon )</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>WIS [Razin et al., 2023]</td>
<td>Performance Preservation</td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>CoreWalk [Brandeis et al., 2020]</td>
<td></td>
<td>LP</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>UGS [Chen et al., 2021]</td>
<td></td>
<td>NC, LP</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>k-snap [Tian et al., 2008]</td>
<td></td>
<td>Clustering Metrics</td>
<td>X, A</td>
</tr>
<tr>
<td></td>
<td>NetGist [Amiri et al., 2018]</td>
<td></td>
<td>Clustering Metrics</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>SCAL [Huang et al., 2021]</td>
<td>Reconstruction-Free</td>
<td>NC</td>
<td>X, A</td>
</tr>
<tr>
<td></td>
<td>VNG [Si et al., 2023]</td>
<td>Reconstruction-Based</td>
<td>REE</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>LV [Loukas and Vanderheynst, 2018]</td>
<td></td>
<td>REE, RE, NC</td>
<td>X, A</td>
</tr>
<tr>
<td></td>
<td>FGC [Kumar et al., 2023]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>GCond [Jin et al., 2022b]</td>
<td>Gradient Matching</td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>DosCond [Jin et al., 2022a]</td>
<td></td>
<td>NC, GC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>MSGC [Gao and Wu, 2023]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>SGGD [Yang et al., 2023]</td>
<td></td>
<td>NC, LP, AD</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>FGDD [Feng et al., 2023]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>GCARes [Mao et al., 2023]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>CTRL [Zhang et al., 2024a]</td>
<td></td>
<td>NC, GC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>GroC [Li et al., 2023b]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>EXGC [Fang et al., 2024]</td>
<td></td>
<td>NC, GC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>MCond [Gao et al., 2023]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>GCDM [Liu et al., 2022]</td>
<td>Distribution Matching</td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>PUMA [Liu et al., 2023d]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>CaT [Liu et al., 2023c]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>FedGKD [Pan et al., 2023]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>KiDD [Xu et al., 2023]</td>
<td></td>
<td>GC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>GC-SNTEK [Wang et al., 2024]</td>
<td></td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>GCEM [Liu et al., 2023b]</td>
<td>Eigenbasis Matching</td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>MIRAGE [Gupta et al., 2023]</td>
<td>Computation Tree Compression</td>
<td>GC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>DISCo [Xiao et al., 2024]</td>
<td>Disentangled Condensation</td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
<tr>
<td></td>
<td>GEOM [Zhang et al., 2024b]</td>
<td>Trajectory Matching</td>
<td>NC</td>
<td>X, A, Y</td>
</tr>
</tbody>
</table>

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 \( S \) and \( T \) like gradients w.r.t. the model parameters and distribution of node classes. 

**Gradient Matching.** For computing the optimum synthetic graph dataset \( S \), Eq. (6) can be rewritten as the following bi-level problem that generalizes to the distribution of random initialization \( P_{\theta_0} \):

\[
\min_{S} \mathbb{E}_{\theta_0 \sim P_{\theta_0}} \left[ \mathcal{L} \left( GNN_{\theta_S} (A, X), Y \right) \right],
\]

s.t. \( \theta_S = \arg \min_{\theta} \mathcal{L} \left( GNN_{\theta_0^0} (A', X'), Y' \right), \quad (7b) \)

where \( \theta (\theta_0) \) denotes that \( \theta \) is a function acting on \( \theta_0 \). 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:

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

where \( D(\cdot, \cdot) \) represents a distance function, \( T \) stands for the total number of steps in the entire training trajectory, and \( \theta_t \) refers to the model parameters at \( t \)-th training epoch. By optimizing the above objective, the training process on
on the larger real dataset and the smaller synthetic graph dataset $S$ mimics the path taken on the larger real dataset $T$, 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 $A'$:

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

(9)

where $MLP_\Phi$ is a multi-layer perceptron (MLP) parameterized with $\Phi$ 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 $\theta_0$ while discarding the training trajectory of $\theta_t$. By dropping the summation in Eq. (8a), the objective function of DosCond becomes:

$$\min_{S} E_{\theta_0 \sim P_{\theta_0}} \left[ D \left( \nabla_\theta \mathcal{L}_3, \nabla_\theta \mathcal{L}_2 \right) \right] .$$

(10a)

Note that, DosCond treats the graph structure $A'$ as a probabilistic model, which allows for 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 [Gao and Wu, 2023; Yang et al., 2023; Feng et al., 2023; Mao et al., 2023; Li et al., 2023b; Gao et al., 2023; Zhang et al., 2024a]. Unlike GCond, which uses a single fully connected graph to generate the condensed graph dataset $S$, 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 $A$ by broadcasting it into the construction of synthetic graph structure $A'$. In this way, it is shown that SGDD reduces the Laplacian Energy Distribution (LED) [Das et al., 2016; Gutman and Zhou, 2006] shift across 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 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., 2024a] 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 $\Delta_{DP}$) 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. (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

\(^2\)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]
and Bilen, 2023]. For example, CaT [Liu et al., 2023c] updates the condensed graph \( S \) using the DM objective function to find the optimal synthetic graph as follows:

\[
\ell_{\text{MMD}} = \sum_{c \in C} r_c \left\| \text{Mean} \left( E_c \right) - \text{Mean} \left( \tilde{E}_c \right) \right\|^2 ,
\]

where \( C \) is the set of node classes, \( E_c \) and \( \tilde{E}_c \) 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 \( \text{GNN}_S \) 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:

\[
\min_S \mathcal{L}_{\text{KRR}} = \frac{1}{2} \left\| y_T - K_{TS} (K_{SS} + \epsilon I)^{-1} y_S \right\|^2 ,
\]

where \( \epsilon > 0 \) is a KRR hyper-parameter, \( K_{TS} \) is the kernel matrix between original and synthetic graphs while \( K_{SS} \) is the kernel matrix between synthetic graphs\(^3\). Also, \( y_T \) and \( y_S \) are the concatenated graph labels from real dataset and synthetic dataset, respectively. Also, Zheng et al. [2023b] propose SFGC, a structure-free graph condensation method using KRR that only outputs the condensed node features \( X' \), as the structure information of the real graphs is embedded in \( X' \).

**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:

\[
\mathcal{L}_c = \sum_{c=1}^C \sum_{k=1}^K \left\| \tilde{h}_{c,k} - h_{c,k} \right\|^2 ,
\]

where \( h_{c,k} \) and \( \tilde{h}_{c,k} \) 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 hyper-parameters. 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 \( T \) 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.

Beyond the above efforts, GEOM [Zhang et al., 2024b] makes the first attempt toward lossless graph condensation, i.e., significantly reducing the graph size without any loss in performance. It points out that the previous state-of-art method SFGC [Zheng et al., 2023b] provides biased and limited supervisory signals to difficulty nodes while overlooking easy nodes. To address this, GEOM introduces an expanding window technique to adjust the matching range for difficult and easy nodes, during the process of matching the training trajectories between \( T \) and \( S \). Remarkably, it achieves lossless graph condensation across standard benchmark datasets.

**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) [Elskens et al., 2019; Ren et al., 2021] focuses on identifying the most effective architec-
ture 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., 2023b] 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 more accessible for algorithms and computers to parse 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 tailored for customized discovery, which means the user can set the granularity of the graph structure and then find perspectives they are interested in. Graph visualization is also used for visualization and explanation. For instance, GCond [Jin et al., 2022h] 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 [Dong et al., 2022]. For example, Bonchi et al. [2014] show random sparsification can achieve meaningful levels of anonymity while preserving features of the original graph. Arora and Upadhyay [2019] consider edge privacy and provide a mechanism for sparsification of a graph that ensures differential privacy and approximates the spectrum of the original graph. However, the trade-off between the degree of reduction and the utility, i.e., preservation of key information, always exists. Dataset condensation can be a promising technique to solve the dilemma [Dong et al., 2022]. In a federated learning framework, where client devices collaboratively contribute to model development by aggregating their updates on a central server, there exists a risk that a malicious server could infer sensitive local information from the model updates sent by these clients. FedGKD [Pan et al., 2023] trains models on condensed local graphs within each client to mitigate the potential leakage of membership of training set [Shokri et al., 2017].

### 4.5 Data Augmentation

Graph data augmentation [Zhao et al., 2022, 2021; Zhou et al., 2022] 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 $X$ into continuous synthetic node attributes $X'$, 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 employ a specific type 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 distribution-related properties 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.

## 5.6 Diverse Types of Graphs

Existing graph reduction techniques have primarily concentrated on simple non-attributed or attributed graphs. Nonetheless, graph data in real-world scenarios is increasingly intricate. This complexity is evidenced by the presence of heterophilous graphs [Zheng et al., 2022], heterogeneous graphs [Bing et al., 2023], directed graphs [Aho et al., 1972], knowledge graphs [Schlichtkrull et al., 2018], hypergraphs [Antelmi et al., 2023] and dynamic graphs [Skarding et al., 2021]. Each graph type introduces distinct characteristics, demanding a thorough comprehension of their structures and the...
adoption of varied graph reduction methods. While we have discussed works [Generale et al., 2022; Sugiyama and Sato, 2023] related to some of the aforementioned graph types, the remainder are yet to be fully explored.

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


Linfeng Cao, Haoran Deng, Chunping Wang, Lei Chen, and Yang Yang. Graph-skeleton: 1% nodes are sufficient to represent billion-scale graph. In WWW, 2024.


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


Junfeng Fang, Xinglin Li, Yongdoo Sui, Yuan Gao, Guibin Zhang, Kun Wang, Xiang Wang, and Xiangnan He. Exgc: Bridging efficiency and explainability in graph condensation. In *WWW*, 2024.


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


Xikun Zhang, Dongjin Song, and Dacheng Tao. Ricci curvature-based graph sparsification for continual graph


