Privacy-Preserving Graph Machine Learning from Data to Computation: A Survey

Dongqi Fu* †, Wenxuan Bao†, Ross Maciejewski§, Hanghang Tong†, Jingrui He†
†University of Illinois Urbana-Champaign
§Arizona State University
dongqif2@illinois.edu, wbao4@illinois.edu, rmacieje@asu.edu, htong@illinois.edu, jingrui@illinois.edu

ABSTRACT

In graph machine learning, data collection, sharing, and analysis often involve multiple parties, each of which may require varying levels of data security and privacy. To this end, preserving privacy is of great importance in protecting sensitive information. In the era of big data, the relationships among data entities have become unprecedentedly complex, and more applications utilize advanced data structures (i.e., graphs) that can support network structures and relevant attribute information. To date, many graph-based AI models have been proposed (e.g., graph neural networks) for various domain tasks, like computer vision and natural language processing. In this paper, we focus on reviewing privacy-preserving techniques of graph machine learning. We systematically review related works from the data to the computational aspects. We first review methods for generating privacy-preserving graph data. Then we describe methods for transmitting privacy-preserved information (e.g., graph model parameters) to realize the optimization-based computation when data sharing among multiple parties is risky or impossible. In addition to discussing relevant theoretical methodology and software tools, we also discuss current challenges and highlight several possible future research opportunities for privacy-preserving graph machine learning. Finally, we envision a unified and comprehensive secure graph machine learning system.

1. INTRODUCTION

According to the recent report from the United Nations [1], strengthening multilateralism is indispensable to solve the unprecedented challenges in critical areas, such as hunger crisis, misinformation, personal identity disclosure, hate speech, targeted violence, human trafficking, etc. Addressing these problems requires collaborative efforts from governments, industry, academia, and individuals. In particular, effective and efficient data collection, sharing, and analysis are at the core of many decision-making processes, during which preserving privacy is an important topic. Due to the distributed, sensitive, and private nature of the large volume of involved data (e.g., personally identifiable information, images, and video from surveillance cameras or body cameras), it is thus of great importance to make use of the data while avoiding the sharing and use of sensitive information. On the other side, in the era of big data, the relationships among entities have become remarkably complicated. Graph, as a relational data structure, attracts much industrial and research interest for its carrying complex structural and attributed information. For example, with the development of graph neural networks, many application domains have obtained non-trivial improvements, such as computer vision [7], natural language processing [93], recommender systems [85], drug discovery [25], fraud detection [55], etc. Within the trend of applying graph machine learning methods to systematically address problems in various application domains, protecting privacy in the meanwhile is non-neglectable [20]. To this end, we consider two complementary strategies in this survey, namely, (1) to share faithfully generated graph data instead of the actual sensitive graph data, and (2) to enable multi-party computation without graph data sharing. Inspired by the above discussion, we focus on introducing two fundamental aspects of privacy-preserving techniques on graphs, i.e., **privacy-preserving graph data** and **graph data privacy-preserving computation**.

For the data aspect, **privacy-preserving graph data** as shown in Figure 1, we focus on the scenario that when publishing or sharing the graph data is inevitable, how could we protect (e.g., mask, hide, or perturb) sensitive information in the original data to make sure that the published or shared data could survive from the external attackers (e.g., node identify disclosure and link re-identification). Hence, in Section 2, we systematically introduce various attackers [2] first (Subsection 2.1) and what backgroud knowledge they need to execute attacks (Subsection 2.2). Then, we introduce the corresponding protection mechanisms and explain why they can address the challenges placed by attackers (Subsection 2.3). Also, we share some graph statistical properties (other than graph data itself) privacy protection mechanisms (Subsection 2.4). After that, we list several possible challenges for privacy-preserving graph data generation when facing complex structures and attributes, e.g., time-evolving graphs and heterogeneous information graphs (Subsection 2.5).

For the computation aspect, **graph data privacy-preserving computation**, we focus on the multi-party computation

*First two authors contribute equally to this research.
[1] https://press.un.org/en/2022/sc15140.doc.htm

[2] Throughout the paper, we use "attackers" to denote the attacks on graphs. There are also attackers that are designed not for graphs but for Euclidean data, for example. Those are not in the scope of this paper.

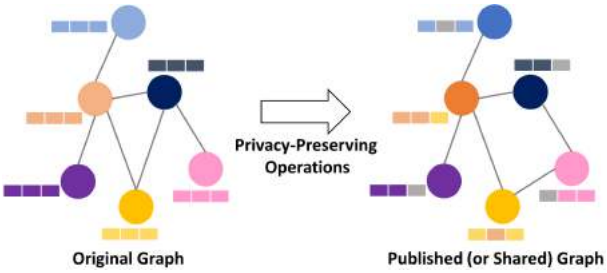


Figure 1: Privacy-Preserving Graph Data. After the privacy-preserving generation, the original graph data is perturbed with certain connections and features permuted.

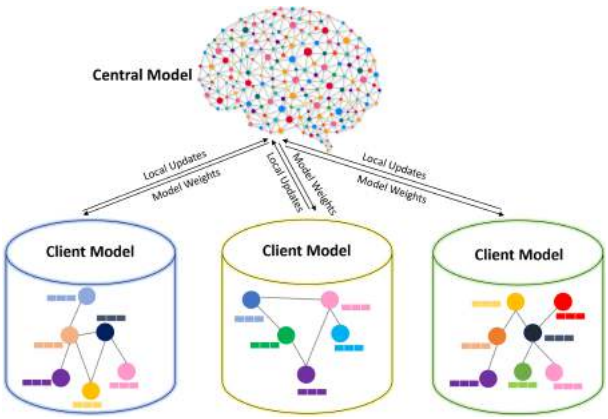


Figure 2: Graph Data Privacy-Preserving Computation. In the federated learning framework, each client model has its own graph data, and the data itself is not transmitted (but the model parameters) to the central model.

scenario where the input data is structured, distributed over clients, and exclusively stored (i.e., not shareable among others). Here, federated learning can be a quick-win solution. However, relational data structures (i.e., graphs) bring a significant challenge (i.e., *non-IIDness*) to the traditional federated learning setting. This means that the data from intra-clients and/or inter-clients can violate the independent and identically distributed assumption (i.e., the i.i.d. assumption) due to the presence of the complex graph features, whose data complexity hinders many existing federated learning frameworks from getting the optimal performance. Motivated by this observation, in Section 3, we first discuss the adaption of federated learning on graphs and the corresponding challenge from non-IIDness brought by graphs (Subsection 3.1), then we introduce how nascent graph federated learning research works to address the non-IIDness issues from three levels, i.e., graph-level federated learning (Subsection 3.2), subgraph-level (Subsection 3.3), and node-level (Subsection 3.4). Then, we list several challenges and promising research directions, including model heterogeneity and avoiding cross-client transmission (Subsection 3.5).

After we introduce **privacy-preserving graph data** and **graph data privacy-preserving computation** with their own methodologies, advances, software tools, limitations, and future directions. In Section 4, we envision the necessity of combing these two directions into **privacy-preserving graph data privacy-preserving computation** to meet any possibility of leaking sensitive information, to further achieve a comprehensive, well-defined, and end-to-end graph machine learning system. Finally, the paper is concluded in Section 5.

Relation with Previous Studies. For the **privacy-preserving graph data**, we systematically review the privacy attackers and the corresponding privacy protection techniques, which takes a balance of classic methods [113; 94] and emerging solutions [36], such as topology perturbation methods, deep generation methods, etc. Beyond that, we extend the privacy-preserving techniques review from the data level to the computation level, i.e., the **graph data privacy-preserving computation** within the federated learning framework. Most of the existing federated learning reviews do not primarily concentrate on graph federated learning [37; 84; 46; 77]. Recently, two survey papers [53; 24] introduce two problem settings in graph federated learning and their corresponding techniques. They exclusively focus on graph federated learning solutions and ignore the connections to traditional federated learning. Thus, we start from various application scenarios and provide a comprehensive classification and exposition of graph federated learning. While our focus primarily revolves around graph federated learning, we also highlight its connections and distinctions to traditional federated learning, aiming to present the big picture of this field. In addition to reviewing the two aspects (i.e., **privacy-preserving graph data** and **graph data privacy-preserving computation**), we also discuss the necessity and possibility of combining these two directions and propose several promising future research directions.

2. PRIVACY-PRESERVING GRAPH DATA

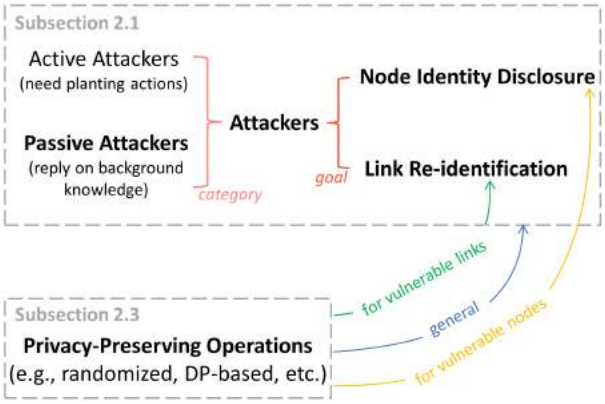


Figure 3: Taxonomy Structure of Section 2.

As for making privacy-preserving graph data to publish or share, the ultimate goal is to successfully protect the published graph data from various attacks from adversaries or attackers. To this end, we first introduce the different kinds of attackers, such as node identity disclosure or sensitive link re-identification in Subsection 2.1 and necessary background knowledge in Subsection 2.2. Then, we introduce how the corresponding privacy-preserving mechanisms are proposed, such as several of them being deliberately designed to defend against certain attackers and some of them being general protections and not aiming at specific attacks, in Subsection 2.3. The taxonomy is shown in Figure 3.

2.1 Privacy Attackers on Graphs

According to [5], what the attackers aim to attack is that they (1) want to learn whether edges exist or not between specific target pairs of nodes and also (2) want to reveal the true identities of targeted users, even from just a single anonymized copy of the graph, with a surprisingly small investment of effort.

2.1.1 Category of Attackers

Attackers can be classified into the **active attackers** and **passive attackers** [5].

The first category is **active attackers**, where the core idea is that the attackers actively plant certain structures into the graph before it is being published. Then, the attackers can identify victims in the published graph by locating the planted structures. For example [113], the attackers create a subgraph H containing k nodes and then use H to connect b target nodes in the original graph G (subgraph H is better to be unique and has the property to be recovered in the published graph). After the original graph G is privacy-preserved (e.g., mask and disturb connections) and published as G', the attackers try to find H in G' and then determine those b nodes.

Active attackers usually need to access the original graph beforehand and then make corresponding active actions like creating new nodes, linking new edges, and planting subgraphs. The planting and recovery operations are usually computationally costly [5]. Therefore, another direction points to passive attacks and defense.

Passive attackers are based on the fact or the assumption that most entities (e.g., nodes and edges) in graphs usually belong to a unique, small identifiable graph. Then, different from active attackers, passive ones do not need to create new nodes and edges in the original but mostly rely on the observation of the published graph to identify victims. In the initial proposal of passive attacks [5], a passive attacker (e.g., a node in a social network) needs to collude with other $(k-1)$ nodes on the original graph, and the coalition needs to know the external information (e.g., their 1-hop neighbors' name in the social network), such that they can reconnect on the published graph to identify the victims. *Here, we expand the scope of passive attacks to include the attackers whose core is observation plus little external information.* For example, in [29], an attacker knows the external background information like "Greg is connected to at least two nodes, each with degree 2" and tries to observe the candidate of plausible Greg in the published social network.

2.1.2 Goal of Attackers

The ultimate goals of most graph privacy attackers can be roughly divided into disclosing the node identity (e.g., name, DOB, and SSN in the social network) and the link existence (e.g., sensitive connections in the social network) [51; 94; 101; 113]. Next, we formally introduce the general definition of these two goals.

Node Identity Disclosure. The node identity disclosure problem often arises from the scenario that the attackers aim to identify a target node identity in the published graph (usually, which has been anonymized already). For example, in a published social network with usernames masked already, the node identity disclosure aims to identify which node is Greg [29]. To be more specific, the identity disclosure can be detailedly divided into node existence disclosure (i.e., whether a target node existed or not in a published graph), node property disclosure (i.e., partial features of a target node are disclosed like its degree, distance to the center, or even sensitive labels, etc) [113].

Link Re-Identification. In a given graph, edges may be of different types and can be classified as either sensitive or not. Some links (i.e., edges) are safe to release to the public, such as classmates or friendships. And some links are sensitive and should maintain private but not published, like the personal disease records with hospitals. The problem of link re-identified is defined as inferring or predicting sensitive relationships from anonymized graphs [111]. Briefly speaking, the adversary (or attacker) achieves the goal when it is able to correctly predict a sensitive link between two nodes. For example, if the attacker can figure out which there is a transaction between two users, given the properties of the released financial graph. Also, there are some detailed categorizations of the line re-identification other than the link existence, such as the link weight and link type or labels [113].

Compared with active attackers, passive attackers are typically efficient in executing for adversaries and do not need to interact with the original graph beforehand very much. **Thus, within the scope of passive attackers, achieving those attacking goals (node identity disclosure or link re-identification) relies on the observation of the published graph and certain external background knowledge to further identify victims.**[3] Next, we focus on introducing what requirements passive attackers need to execute attacks passively.

2.2 Background Knowledge for Passive Attacks

Here, we first discuss some background knowledge that could contribute to the goal of node identity disclosure. Then, we list some background knowledge that could contribute to sensitive link re-identification attacks.

2.2.1 Background Knowledge for Node Identity Disclosure

In general, the background knowledge for achieving node identity disclosure is to help them to detect the uniqueness of victims (i.e., nodes in the published graph) and thus narrow down the scope of candidate sets to increase the successful attack probability. For example, assume that the attackers know some background knowledge $\mathcal{H}$ about a target node, after that, the attackers observe the published graph and find 2 candidates satisfying the condition (i.e., $\mathcal{H}$), then the attackers have 50% confidence to reveal the identity of that target node in the published graph. Next, we introduce some methods to acquire background knowledge.

Vertex Refinement Queries [29]. These are interactive queries, which describe the local structure of the graph around a target node x. The initial query in vertex refinement queries is denoted as $\mathcal{H}_0(x)$ that simply returns the label of node x in the labeled graph (or a constant ϵ in the unlabeled graph). And $\mathcal{H}_1(x)$ returns the degree of node x. Then, iteratively, $\mathcal{H}_i(x)$ is defined as the multiset of $\mathcal{H}_{i-1}(\cdot)$ queries on 1-hop neighbors of node x, which can be expressed

[3]Node identity disclosure and link re-identification can also be achieved in active ways [5], but in the paper, we focus on introducing the passive manners that achieve those goals.

as follows.

$$\mathcal{H}_i(x) = \{\mathcal{H}_{i-1}(z_1), \mathcal{H}_{i-1}(z_2), \ldots, \mathcal{H}_{i-1}(z_{d_x})\} \quad (1)$$

where d_x is the degree of node x. For example, in a social network, $\mathcal{H}_2(Bob) = \{1, 1, 4, 4\}$ means that Bob has four neighbors their degrees are 1, 1, 4, and 4, respectively.

Subgraph Queries [29]. These queries assert the existence of a subgraph around a target node. Compared with the above vertex refinement queries, subgraph queries are more general (i.e., the information is not exclusively occupied to a certain graph structure) and flexible (i.e., informativeness is not limited by the degree of a target node). In brief, the adversary is assumed capable of gathering some fixed number of edges around a target node x and figuring out what subgraph structure those collected edges can form. For example, still targeting Bob in a social network, when collecting 3 edges, attackers can find 3 distinct neighbors. And collecting 4 edges can find a tree rooted by Bob. Those existences of structures form H such that attackers can use them to reveal the identity of Bob. Also, different searching strategies can result in different subgraph structures. For example, based on collecting 3 edges from Bob, breadth-first exploration may result in a star subgraph, and depth-first exploration may end up with a three-node-line. We refer to [29], where a range of searching strategies are tested to empirically illustrate the descriptive power of background knowledge.

Hub Fingerprint Queries [29]. First of all, a hub stands for a node that has a high degree and a high betweenness centrality (i.e., the proportion of shortest paths in the graph that include that node) in the graph. Then, a hub fingerprint is the description of a node's connections to hubs. To be more specific, for a target node x, the corresponding hub fingerprint query $\mathcal{H}_i(x)$ records the shortest distance towards each hub in a graph. In $\mathcal{H}_i(x)$, i is the limit of measurable distance. For example, $\mathcal{H}_1(Bob) = (1, 0)$ means Bob is 1 distance away from the first hop and not connected to (or 1 distance non-reachable from) the second hub. And, $\mathcal{H}_2(Bob) = (1, 2)$ means that Bob is 1 distance away from the first hop and 2 distance away from the second hub.

Neighborhood Relationships Queries [112]. Targeting a node, if an adversary has background knowledge about its neighbors and the relationship among the neighbors, then the victim can be identified in the anonymized graph. To be specific, the neighborhood relationship query rely more on the isomorphism of the ego-graph (i.e., 1-hop neighbors) of a target node to reveal its identity, compared with iterative vertex refinement query [29] and general subgraph query [29]. For example, in a social network, if Bob has two close friends who know each other (i.e., are connected) and two close friends who do not know each other (i.e., are not connected), then this unique information obtained by the adversary can be used to find Bob in the published anonymized graph.

2.2.2 Background Knowledge for Link Re-Identification

Link Prediction Probabilistic Model [111]. This probabilistic model is proposed to determine whether a relationship between two target nodes. And different kinds of background information (i.e., observation) can be leveraged to formalize the probabilistic model, such as (1) *node attributes*, e.g., two social network users who share the same interest are more likely to be friends; (2) *existing relationships*, e.g., two social network users in the same community are more likely to be friends; (3) *structural properties*, e.g., the high degree nodes are more likely to connect in a graph; and (4) inferred relationships (i.e., a complex observation that is more likely based on the inference of the invisible relationship), e.g., two social network users are more likely to be friends if they both are close friends of a third user.

Mathematically, those above observations can be expressed for predicting the existence of a sensitive relation between node i and node j as $P(e^s_{ij}|O)$, where e^s_{ij} stands for the sensitive relationship and O consists of several observations $\{o_1, \ldots, o_n\}$. For example, if we use the second kind of information (i.e., existing relationships), then $\{o_1, \ldots, o_n\}$ is a set of edges between node i and node j with the edge type other than s, denoted as e^l_{ij} and $l \in \{1, \ldots, n\}$ is the index of other edge relationships. To solve out $P(e^s_{ij}|O)$, the noisy-or model [61] can be used as suggested by [111], where each observation $o_l \in \{o_1, \ldots, o_n\}$ is considered as independent with each other and parameterised as $\lambda_l \in \{\lambda_1, \ldots, \lambda_n\}$. Moreover, there is a leak parameter λ_0 to capture the probability that the sensitive edge is there due to other unmodeled reasons. Hence, the probability of a sensitive edge is expressed as follows.

$$P(e^s_{ij} = 1|o_1, \ldots, o_n) = 1 - \prod_{l=0}^{n}(1 - \lambda_l) \quad (2)$$

where s in e^s_{ij} is the indicator of sensitive relationship, and the details of fitting the values of λ_l can be found in [111].

Randomization-based Posterior Probability [100]. To identify a link, this observation is based on randomizing the published graph G' and counting the possible connections over a target pair of nodes i and j. And those countings are utilized for the posterior probability to determine whether there is a link between nodes i and j in the original graph G. Formally, the posterior probability for identifying the link e_{ij} in the original graph G is expressed as follows.

$$P(e_{ij} = 1|G'_s) = \frac{1}{N}\sum_{s=1}^{N} \mathbb{1}(G'_s(i, j) == 1) \quad (3)$$

where the attacker applies a certain randomization mechanism on the published graph G' N times to get a sequence of G'_s, and $s \in \{1, \ldots, N\}$. In each G'_s, if there is an edge connects the target nodes i and j, then the indicator function $\mathbb{1}(G'_s(i, j) == 1)$ will count one.

2.3 Privacy-Preserving Mechanisms

Here, we discuss some privacy-preserving techniques that are deliberately designed for specific attackers and also some general protection techniques that are not targeting attackers but can be widely applied.

2.3.1 Protection Mechanism Designed for Node Identity Dislosure

In general, the protection mechanisms are proposed to enlarge the scope of candidates of victims, i.e., reduce the uniqueness of victims in the anonymized graphs.

k-degree Anonymization [52]. The motivation for k-degree anonymization is that degree distribution is highly skewed in real-world graphs, such that it is usually effective to collect the degree information (as the background knowledge) to identify a target node. Therefore, this protection mechanism aims to ensure that there at least exist $k - 1$

nodes in the published graph G', in which $k-1$ nodes share the same degree with any possible target node x. In this way, it can largely prevent the node identity disclosure even if the adversary has some background knowledge about degree distribution. To obtain such anonymized graph G', the method is two-step. First, for the original graph G with n nodes, the degree distribution is encoded into a n-dimensional vector $\mathbf{d}$, where each entry records the degree of an individual node; And then, based on $\mathbf{d}$, the authors proposed to create a new degree distribution $\mathbf{d}'$, which is k-anonymous with a tolerated utility loss (e.g., isomorphism cost) instanced by the L_1 distance between two vectors $\mathbf{d}$ and $\mathbf{d}'$. Second, based on the k-anonymous degree vector $\mathbf{d}'$, the authors proposed to construct a graph G' whose degree distribution is identical to $\mathbf{d}'$.

k-degree Anonymization in Temporal Graphs [69]. For temporal graphs (i.e., graph structures and attributes are dependent on time [19]), this method aims to ensure that the temporal degree sequence of each node is indistinguishable from that of at least $k-1$ other nodes. On the other side, this method also tries to preserve the utility of the published graph as much as possible. To achieve the k-anonymity, the proposed method first partition n nodes in the original temporal graph G into m groups using k-means based on the distance of temporal degree vectors $\mathbf{d}$ of each node, which is a T-dimensional vector records the degree of a node at different timestamp t.To realize the utility, constrained by the cluster assignment, the method refines $\mathbf{d}$ of each node into $\mathbf{d}'$ while minimizing the L_1 distance between matrices $\mathbf{D}$ and $\mathbf{D}'$ (which are stacks of $\mathbf{d}$ and $\mathbf{d}'$). After that, the anonymized temporal graph G' is constructed by $\mathbf{D}'$ to release for each timestamp individually.

k-degree Anonymization in Knowledge Graphs [33]. Different from the ordinary graph, the knowledge graph has rich attributes on nodes and edges [35]. Therefore, the k-degree is upgraded with the k-attributed degree that aims to ensure a target node in the anonymized knowledge graph has $k-1$ other nodes who share the same attributes (i.e., node level) and degree (i.e., edge level) [32]. Then the k-degree anonymization solution gets upgraded in [33], which aims to solve the challenge when the data provider wants to continually publish a sequence of anonymized knowledge graphs (e.g., the original graph needs to update and so the anonymized does). Then, in [33], the k-ad (short for k-attributed degree) is extended to k^ω-ad, which targets to defend the node identity disclosure in the ω continuous anonymized versions of a knowledge graph. The basic idea is to partition nodes into clusters based on the similarity of node features and degree; Then, for the knowledge graph updates (like newly inserted nodes or deleted nodes), manual intervention is applied (e.g., adding fake nodes) to ensure the k^ω anonymity; Finally, the anonymized knowledge graph gets recovered from the clusters. This initial idea [33] gets further formalized and materialized in [34].

k-neighborhood Anonymization [112]. This protection is proposed to defend the node identity disclosure when the adversary comprehends the background knowledge about neighborhood relationships of a target node (i.e., Neighborhood Relationship Queries discussed in Subsection 2.2.1). The core idea is to insert nodes and edges in the original graph G to get an anonymized graph G', such that a target node x can have multiple nodes whose neighborhood structure is isomorphic in G'. Given a pair node v and u in graph G (suppose node v is the target), the authors first propose the *neighborhood component* and use DFS search to encode the ego-net $Neighbor_G(v)$ and $Neighbor_G(u)$ into vectors. Then, by comparing the difference between $Neighbor_G(v)$ and $Neighbor_G(u)$, the authors then greedy insert missing (labeled) nodes and edges (into $Neighbor_G(v)$ or $Neighbor_G(u)$) to make $Neighbor_G(v)$ and $Neighbor_G(u)$ isomorphic. Those inserted nodes and edges make G into G'.

k-automorphism Anonymization [117]. This method is proposed for the structural queries by attackers, especially for the subgraph queries (as discussed in Subsection 2.2.1). Basically, given an original graph G, this method produces an anonymization graph G' to publish, where G is the subgraph of G' and G' is k-automorphic. To do this, the authors propose the KM algorithm, which partitions the original graph G and adds the crossing edge copies into G, to further convert G into G'. Hence, the G' can satisfy the k-different match principle to defend the subgraph query attacks, which means that there are at least k-different matches in G' for a subgraph query, but those matches do not share any nodes.

2.3.2 Protection Mechanism Designed for Link Re-Identification

The general idea of solutions here is proposed to reduce the confidence of attackers (which usually can be realized by a probabilistic model) for inferring or predicting links based on observing the published anonymized graphs.

Intact Edges [111]. This solution is straightforward and trivial. Given the link re-identification attacker aims to predict a target link between two nodes, and the corresponding link type (i.e., edge type) is denoted as s, then the intact edges strategy is to remove all s type edges in the original graph G and publish the rest as the anonymized graph G'. Those remaining edges are so-called intact.

Partial-edge Removal [111]. This approach is also based on removing edges in the original graph G to publish the anonymized graph G'. Partial-edge removal does not exhaustively remove all sensitive (indexed by s type) edges in G, but it removes part of existing edges. Those removed existing edges are selected based on the criteria of whether their existence contributes to the exposure of sensitive links, e.g., they are sensitive edges, they connect high-degree nodes, etc. Even those removals can be selected randomly.

Cluster-edge Anonymization [111]. This method requires that the original graph G can be partitioned into clusters (or so-called equivalence classes) to publish the anomymized graph G'. The intra-cluster edges are removed to aggregate a cluster into a supernode (i.e., the number of clusters in G is now the number of nodes in G'), but the inter-cluster edges are reserved in G'. To be more specific, for each edge whose edge type is not sensitive (i.e., not s type), if it connects any two clusters, it will be reserved in G'; otherwise, it will be removed. It can be observed that this method needs the clustering pre-processing, which also means that it can cooperate with the node anonymization method. For example, the k-anonymization [71; 44; 56] can be applied on the original graph G first to identify the equivalence classes, i.e., which nodes are equivalent in terms of k-anonymization (for example, nodes who have the same degree).

Cluster-edge Anonymization with Constraints [111]. This method is the upgraded version of the previous cluster-edge anonymization, and it is proposed to strengthen the

utility of the anonymized graph G' by adjusting the edges between clusters (i.e., equivalence classes). The core idea is to require the equivalence class nodes (i.e., cluster nodes or supernodes in G') to have the same constraints as any two nodes in the original graph G. For example, if there can be at most two edges of a certain type between nodes in G, there can be at most two edges of a certain type between the cluster nodes in G'.

2.3.3 General Privacy Protection Mechanisms

Besides the protections that are designed deliberately for the node identity disclosure and link re-identification risks, there are also other protection mechanisms that are not designed for a specific kind of attacker but for the general and comprehensive scenario, such as randomized mechanisms with constraints and differential privacy schema. Next, we will discuss these research works.

Graph Summarization [29]. This method aims to publish a set of anonymized graphs G' given an original graph G, through the graph summarization manner. To be specific, this method relies on a pre-defined partitioning method to partition the original graph G into several clusters, then each cluster will just serve as a node in the anonymized graph G'. The selection of connecting nodes in G' results in the variety of G', which means that a sequence of G' will appear with a different edge connecting strategy. The detailed connection strategy can be referred to [29].

Switching-based Graph Generation [100]. Here, the authors aim to publish the anonymized graph G' that should also preserve the utility of the original graph G. Therefore, they propose the graph generation method based on the switching operations that can preserve the graph features. Moreover, the switching is realized in an iterative Monte Carlo manner, each time two edges (a, b) and (c, d) are selected. Then they will switch into (a, d) and (b, c) or (a, c) and (b, d). The authors constrain that two selected edges are switchable if and only if the switching generates no more edges or self-edges, such that the overall degree distribution will not change. After sufficient Monte Carlo switching operations, the authors show that the original graph features (e.g., eigenvalues of adjacency matrix, eigenvectors of Laplacian matrix, harmonic mean of geodesic path, and graph transitivity) can be largely preserved in the anonymized graph G'.

Spectral Add/Del and Spectral Switch [99]. The idea of this method starts from Rand Add/Del and Rand Switch. Rand Add/Del means that the protection mechanism randomly adds an edge after deleting another edge and repeats multiple times, such that the total number of edges in the anonymized graph will not change. Rand Switch is the method that randomly switches a pair of existing edges (t, w) and (u, v) into (t, v) and (u, w) (if (t, v) and (u, w) do not exist in the original graph), such that the overall degree distribution will not change. In [99], the authors develop the spectrum-preserving randomization methods Spectral Add/Del and Spectral Switch, which preserve the largest eigenvalue λ_1 of the adjacency matrix $\mathbf{A}$ and the second smallest eigenvalue μ_2 of the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$. To be specific, the authors first investigate which edges will cause the λ_1 and μ_2 increase or decrease in the anonymized graph and then select the edges from different categories to do Rand Add/Del and Rand Switch to control the values of λ_1 and μ_2 not change too much in the anonymized graph.

RandWalk-Mod [60]. This method aims to inject the connection uncertainty by iteratively copying each existing edge from the original graph G to an initial null graph G' with a certain probability, guaranteeing the degree distribution of G' is unchanged compared with G. Starting from each node u in the original graph G, this method first gets the neighbor of node u in G denoted as $\mathcal{N}_u$. Then for each node in $\mathcal{N}_u$, this method runs multiple random walks and denotes the terminated node in each walk as z. Finally, RandWalk-Mod adds the edge (u, z) to G' with certain probabilities under different conditions (e.g., 0.5, a predefined probability α, or $\frac{0.5d_u-\alpha}{d_u-1}$, where d_u is the degree of node u in G).

Next, we introduce an important component in the graph privacy-preserving techniques, i.e., differential privacy [36]. The general idea of differential privacy is that two adjacent graphs (e.g., one node/edge difference between two graphs) are indistinguishable through the permutation algorithm $\mathcal{M}$. Then, this permutation algorithm $\mathcal{M}$ satisfies the differential privacy. The behind intuition is that the randomness of $\mathcal{M}$ will not make the small divergence produce a considerably different distribution, i.e., the randomness of $\mathcal{M}$ is not the cause of the privacy leak. If the indistinguishable property is measured by ϵ, then the algorithm is usually called ϵ-differential privacy algorithm. The basic idea can be expressed as follows.

$$\frac{Pr[\mathcal{M}(G) \in S]}{Pr[\mathcal{M}(\tilde{G}) \in S]} \leq e^{\epsilon} \tag{4}$$

where G and G' are adjacenct graphs, $\mathcal{M}$ is the differential privacy algorithm, and ϵ is the privacy budget. The above equation illustrates that the probability of the same output range is almost equivalent.

Within the context of graph privacy, the differential privacy algorithm can be roughly categorized as edge-level differential privacy and node-level differential privacy. Given the input original graph G, the output graph of the differential algorithm $\mathcal{M}(G)$ can be used as the anonymized graph G' to publish.

Edge-level Differential Privacy Graph Generation. We first introduce the edge-level differential privacy algorithms, which means that the privacy algorithm can permute adjacent graphs (e.g., one edge difference) indiscriminately.

- **DP-1K and DP-2K Graph Model** [87]. This edge-level differential privacy algorithm is proposed with the utility preserving concern of complex degree distribution. Here, 1k-distribution denoted by $P_1(G)$ is the ordinary node degree distribution in graph G, e.g., the number of nodes having 1 degree is 10 then $P_1(1) = 10$, the number of nodes having 2 degrees is 5 then $P_1(2) = 5$, etc. 2K-distribution denoted by $P_2(G)$ is the joint graph distribution in graph G, e.g., the number of edges connecting an i-degree node and a j-degree node, with iterating i and j. And $P_2(2, 3) = 6$ means that the number of edges in G connecting a 2-degree node and a 3-degree node is 6. Hence, DP-1K (or DP-2K) Graph Model first computes the 1K-(or 2K-) degree distribution $P_1(G)$ (or $P_2(G)$) and then permutes the degree distribution under the edge-level DP to obtain the $P_1(G)'$ (or $P_2(G)'$). Finally, an off-the-shelf graph generator (e.g., [70]) is called to build the anonymized graph G' based on $P_1(G)'$ (or $P_2(G)'$).
- **Local Differential Privacy Graph Generation** (LDP-

Table 1: Graph Privacy-Preserving Mechanisms

Scenario	Name	Description	Link
Node Identity Disclosure	k-degree Anonymization [52]	Generate $k-1$ plausible candidate for protecting the victim	Github (Unofficial)
	k-degree Anonymization in Temporal Graphs [69]	Adaption of k-degree Anonymization on temporal graphs	——
	k-degree Anonymization in Knowledge Graphs [34]	Adaption of k-degree Anonymization on knowledge graphs	Github
Link Re-identification	Partial-edge Removal, Cluster-edge Anonymization [111]	Edge removing methods that are deliberately designed for link re-identification tasks	——
General	Graph Summarization [29]	Partitioning (or clustering) + Publishing supernodes and superedges	Github (Unofficial)
	Local Differential Privacy Graph Generation [64]	Proportionally flipping existing and non-existing edges with graph utility preserved	——
	Edge-level DP Algorithm [97]	A deep learning graph generative model under the edge-level differential privacy constraints	Github
	Node-level DP Algorithms [39]	Some node-level differential privacy algorithms that compute low-sensitivity approximations to several classes of graph statistics	Github (Unofficial)

GEN) [64] is motivated by permuting the connection distribution, i.e., proportionally flipping the existing edge to non-existing and vice versa. To make the generated graph preserve the original utility, LDP-GEN [64] first partitions the original graph G into the disjoint clusters and adds Laplacian noise on the node's degree vector in each cluster, which guarantees the local edge-level differential privacy. After that, the estimator is used to estimate the connection probability of intra-cluster edges and inter-cluster edges based on the noisy degree vectors, such that the anonymized graph G' is generated.

- **Differentially Private Graph Sparsification** [3]. On the one hand, this method constrains the number of edges in the anonymized graph G' is less than the original graph G to a certain extent. On the other hand, the method requires that the Laplacian of the anonymized graph G' is approximated to the original graph G (i.e., see Eq.1 in [3]). The two above objectives are unified into an edge-level differential privacy framework. The new graph G' is then obtained by solving an SDP (i.e., semi-definite program) problem.

- **Temporal Edge-level Differential Privacy**. In [82], two temporal graphs are adjacent if they only differ in one update (i.e., the existence and non-existence of a temporal edge, different weights of an existing temporal edge). Based on the Priv-Graph algorithm (i.e., adding noise to graph Laplacian matrix), Sliding-Priv-Graph [82] is proposed to (1) take recent updates and ensure the temporal edge-level differential privacy and (2) meet the smooth Laplacian property (i.e., the positive semi-definite of consecutive Laplacian matrices). Moreover, in [14], the authors distinguish the edge-adjacency and node-adjacency in the temporal graphs. Two temporal graphs are node-adjacent (or edge-adjacent) if they only differ in one node (or edge) insertion or deletion.

- **Deep Graph Models with Differential Privacy**. Following the synergy of deep learning and differential privacy [1], another way to preserve privacy is targeting the gradient of deep graph learning models. In [97], a deep graph generative model called $DPGG_{AN}$ is proposed under the edge-level differential privacy constraints, where the privacy protection mechanism is executed during the gradient descent phase of the generation learning process, by adding Gaussian noise to the gradient of deep learning models.

Node-level Differential Privacy Graph Generation. Compared with edge-level differential privacy, node-level differential privacy is relatively difficult to be formalized and solve. In [39], authors contribute several theoretical node-level differential privacy solutions such as Flow-based Lipschitz extension and LP-based Lipschitz extensions. But they all focus on realizing part of the graph properties instead of the graph data itself, such as anonymized degree distribution, subgraph counting, etc. The same kind of research flavor also appeared in relevant node-level differential privacy works like [6; 39; 66]. Again, differential privacy mechanisms on graphs is a large and comprehensive topic, a more detailed introduction and extensive literature review can be found in [36].

2.4 Other Aspects of Graph Anonymization

Here, we would also like to review several graph anonymization techniques, but the difference from the majority mentioned above is that: they are not publishing the anonymized graph G' but anonymize some non-trivial and graph statistics of the original graph G and release them to the public [88; 105; 74; 81]. The central requirement for protecting the graph statistics is that some scalar graph parameters are essential to describe the graph topology (e.g., degree distributions) or even reconstruct the graph topology (e.g., the number of nodes and edge connection probability in the Erdos-Renyi graph). To this end, some methods focus on protecting the important graph parameters and their statistics before releasing them. For example, the spectrum of a graph (i.e., eigen-decomposition of the graph Laplacian matrix) can preserve many important graph properties such as topological connections, low-pass or high-pass graph single filters, etc. Therefore, in [88], the authors proposed to per-

mute the eigen-decomposition under the differential privacy and then release the permuted parameters. To be specific, given the original eigenvalues and eigenvectors, certain calibrated random noises are sampled and added to them under the differential privacy constraint. Under the same protection mechanism, i.e., differential privacy, the protection goal is set to be the number of occurrences of subgraphs in [105], the sequence of degree distribution in directed graphs and undirected graphs in [74], and the edge connection probability of random graphs in [81].

2.5 Challenges and Future Opportunities

After introducing different graph anonymization techniques, we would like to share some open questions and corresponding challenges.

2.5.1 Preserving Privacy for Temporal Graphs

As discussed above, most privacy-preserving graph anonymization methods still consider the input graphs as static. However, in complex real-world scenarios, the graphs are usually evolving over time [22; 114; 17; 18], which brings critical challenges to the current privacy-preserving static graph generation process. In other words, the time domain enriches the node attribute dimension and may also dictate the attribute distribution, which leads to increased exposure risk. For example, some graphs contain multiple dynamics and accurately representing them could contribute to graph tasks like classification [115; 16]. But, the existence of various dynamics increases the probability of being unique and enlarges the leaking risk.

2.5.2 Preserving Privacy for Heterogeneous Graphs

During the node identity disclosure and link re-identification, it can be observed that the majority of background knowledge is solely from structural queries, which is already forceful enough. In heterogeneous graphs [76; 21], the abundant node and edge features increase the risk of leaking sensitive information and bring challenges to protection mechanisms, especially the heterogeneous graphs start to evolve [23; 48].

To the best of our knowledge, how to generate privacy-preserving heterogeneous or temporal graphs remains open.

- What kind of feature information is sensitive in heterogeneous or time-evolving graphs and should be hidden in the generated graph?
- If the corresponding sensitive information is determined, what techniques are effective for protecting structures and features in the heterogeneous or time-evolving environment?
- Last but not least, if the corresponding protection mechanism is designed, how to maintain the generation utility simultaneously with privacy constraints?

3. GRAPH DATA PRIVACY-PRESERVING COMPUTATION

In recent years, graph machine learning has become increasingly popular due to the abundance of graph-structured data in various domains, such as social networks, recommendation systems, and bioinformatics. However, graph data is usually distributed in multiple data sources, and each data owner does not have enough data to train satisfactory machine learning models, which require a massive amount of graph data. For example, biochemical industries may wish to collaboratively train a graph neural network model to predict the property of molecules. While we introduce one solution with privacy-preserving graph data generation in the last section, another solution is to enable multi-party computation without exchanging raw data. In this section, we introduce federated learning (FL) [58], a machine learning system where multiple clients (i.e., data owners) collaboratively train machine learning models without exchanging their raw data. In particular, we first introduce the framework of federated learning and its applications with graph data in Subsection 3.1. Then we introduce important FL algorithms under three representative graph federated learning scenarios: graph-level FL (Subsection 3.2), subgraph-level FL (Subsection 3.3), and node-level FL (Subsection 3.4). Finally, we summarize the challenges of future opportunities of graph FL in Section 3.5.

3.1 Framework and Applications of Federated Learning

Federated learning (FL) [58] is a distributed learning system where multiple clients (i.e., data sources) collaborate to train a machine learning model under the orchestration of the central server (i.e., the service provider), while keeping their data decentralized and private [37]. This subsection provides an exposition on the FL framework, followed by an overview of the application of federated learning on graph data.

3.1.1 Federated Learning Framework

A typical FL framework has one central server and N clients, each with its own dataset $\mathcal{D}_i$. The main steps can be summarized as follows:

1. *Parameter broadcasting.* The server broadcasts the current global model to (selected) clients.
2. *Local update.* Each client locally trains its local model.
3. *Parameter uploading.* Each client sends upload the model update back to the server.
4. *Model aggregation.* The server aggregates the model updates collected from clients and updates the global model.
5. *Repeat*: Steps 1-4 are repeated for multiple communication rounds until the global model converges to satisfactory performance.

One of the most popular FL algorithms is FedAvg [58]. In each communication rounds, the server randomly selects a subset of clients, and broadcasts the global model to them. Each client locally updates the model with multiple iterations of stochastic gradient descent, and uploads its local model back to the server. Finally, the server computes a weighted average of local model parameters, and updates the global model parameters. Algorithm 1 gives the pseudo-code of FedAvg. Notice that in FedAvg, local data never leaves the client side. Besides FedAvg, most of the FL algorithms strictly follow the aforementioned training protocol [47; 13], or roughly follow it with a few modifications [27; 102].

FL protects client privacy in two main ways. Firstly, instead of transmitting the raw data, FL transmits only the model

Algorithm 1 FedAvg. The K clients are indexed by k; C is the participation rate, B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize model parameter w_0
for each round $t = 1, 2, \dots, T$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for

ClientUpdate(k, w): // *Run on client* k
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
  end for
end for
return w to server

parameters, which are updated based on the local data of each client. By doing so, FL ensures that sensitive data remains on the client's device and is not transmitted to the central server and other clients. Secondly, the model parameters uploaded to the server only reveal the distribution of local data, rather than individual data points. This approach helps to maintain privacy by obscuring the specific data points used to train the model.

FL can be equipped with differential privacy mechanisms [59; 91] to enhance privacy protection. As described in the last section, differential privacy is a technique that involves adding noise to data in order to obscure individual contributions while still maintaining overall data patterns. However, different from graph generation, where the noise is added to the data (e.g., node feature, edges, etc), in the context of FL, the noise is added to the uploaded and downloaded model parameters. This ensures that even if an attacker were to obtain the model parameters, they would not be able to accurately infer the raw data from the model parameter. By adding moderate noise to the parameters, the model's accuracy may be slightly reduced, but the overall performance remains comparable to non-private models. In summary, by using differential privacy mechanisms, FL can achieve even better privacy protection by making it harder for attackers to identify the sensitive data contributed by individual clients.

3.1.2 Application of Graph Federated Learning

In this part, we introduce important applications of federated learning on graph data. Roughly, we survey three representative application scenarios: *graph-level FL*, *subgraph-level FL*, and *node-level FL*.

1. *Graph-level FL*: Each client has one or several graphs, while different graphs are isolated and independent. One typical application of graph-level FL is for drug discovery [31], where biochemical industries collaborate to train a graph neural network model predicting the property of molecules. Each molecule is a graph with basic atoms as nodes and chemical bonds as edges.

2. *Subgraph-level FL*: Each client has one graph, while each graph is a subgraph of an underlying global graph. One representative application of subgraph-level FL is for financial transaction data [43]. Each FL client is a bank that keeps a graph encoding the information of its customers, where nodes are individual customers and edges are financial transaction records. While each bank holds its own graph, customers in one bank may have connections to customers in another bank, introducing cross-client edges. Thus, each bank's own graph is a subgraph of an underlying global graph.

3. *Node-level FL*: Each client is a node of a graph, and edges are the pairwise relationships between clients, e.g., their distribution similarity or data dependency. One example is the smart city, where clients are traffic sensors deployed on the road and linked to geographically adjacent sensors. While clients form a graph, each client can make an intelligent decision based on the collected road conditions and nearby devices.

Figure 4 illustrates the three application scenarios above. Next, we investigate each application scenario in the following three subsections individually.

3.2 Graph-level FL

In this subsection, we investigate graph-level FL. Graph-level FL is a natural extension of traditional FL: while each client has one or several graphs, different graphs are isolated and independent. The goal of each client is to train a graph neural network (GNN) model for a variety of local tasks, e.g., node-level (e.g., node classification), link-level (e.g., edge prediction), or graph-level (e.g., graph classification).

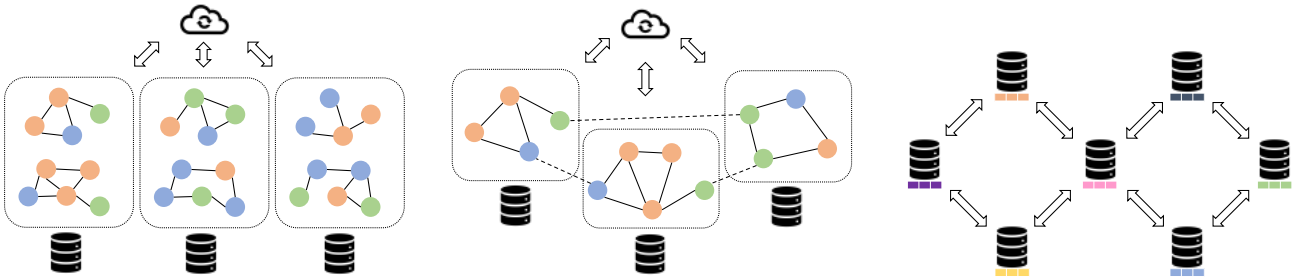
One of the most representative applications of graph-level FL is drug discovery, where graphs are molecules with atoms as nodes and chemical bonds as edges. Each FL client can be a pharmaceutical corporation that owns molecule data. Multiple corporations collaborate to train better model for molecular property prediction.

The biggest challenge of graph-level FL is the non-identical distribution among different clients' data. Since each client in FL collects their local data individually, their local datasets usually have a different distribution. For example, different pharmaceutical corporations may focus on different types of molecules. Such heterogeneity among clients' data distributions introduces optimization challenges to FL. Moreover, when clients' distribution is largely different, it might be harmful or even impossible to train one universal global model across all clients. More sophisticated techniques are required to achieve beneficial collaboration.

Next, we will introduce algorithms for graph-level FL in two parts: global federated learning and personalized federated learning. Since graph-level FL is a natural extension of traditional FL, we will cover both general FL algorithms and graph FL algorithms.

3.2.1 Global Federated Learning

Global federated learning (GFL) aims to train a *shared* global model for all clients. FedAvg [58] provides an initial solution for training GNNs with isolated graphs from multiple clients.



(a) Graph-level (Subsection 3.2) (b) Subgraph-level (Subsection 3.3) (c) Node-level (Subsection 3.4)

Figure 4: Three application scenarios of graph federated learning.

However, when clients have significantly different underlying distributions, FedAvg needs much more communication rounds for convergence to a satisfactory model, and may converge to a sub-optimal solution [58]. This phenomenon of worse convergence is usually explained by *weight divergence* [110], i.e, even with the same parameter initialization, the model parameters for different clients are substantially different after the first local stochastic gradient descent (SGD) step. With different model parameters, the mean of client gradients can be different from the gradient in centralized SGD, and introduce error to the model loss [84].

Data-sharing. To tackle the non-IID challenge to FL optimization, a simple but effective method is to share a small amount of data among clients. [110] first explore an association between the weight divergence and the non-IIDness of the data, and propose a method to share a small amount of data among the server and all clients. As a result, the accuracy can be increased by 30% for the CIFAR-10 dataset [40] with only 5% globally shared data. [102] further improves the privacy of this approach by sharing the average of local data points, instead of raw data. Specifically, each client uploads averaged data, receives averaged data from other clients, and performs Mixup [104] data augmentation locally to alleviate weight divergence. However, both methods require modification of the standard FL protocol and transmission of data. Another way to improve privacy is to share synthetic data generated by generative adversarial networks (GANs) [28], instead of the raw data. The synthetic data can be a collection of each client's synthetic data generated with local GANs or generated with one global GAN trained in FL [67; 12]. However, it is unclear whether GAN can provide enough privacy, since it may memorize the training data [4].

Modifying local update. Another line of research works modifies the local update procedure to alleviate weight divergence without changing the communication protocol of FL. FedProx [47] adds a proximal term to the local objective to stabilize the training procedure. The proximal term is the squared L2 distance between the current global model and the local model, which prevents the local model from drifting too far from the global model. SCAFFOLD [38] estimates how local updates deviate from the global update, and it then corrects the local updates via variance reduction. Based on the intuition that the global model can learn better representation than local models, MOON [45] conducts contrastive learning at the model level, encouraging the agreement of representation learned by the local and global models.

3.2.2 *Personalized Federated Learning*

While the aforementioned algorithms can accelerate the model optimization for GFL, one model may not always be ideal for all participating clients [72]. Recently, personalized federated learning (PFL) has been proposed to tackle this challenge. PFL allows FL clients to collaboratively train machine learning models while each client can have different model parameters.

Clustered FL. In clustered FL, clients are partitioned into non-overlapping groups. Clients in the same group will share the same model, while clients from different groups can have different model parameters. In IFCA [27], k models are initialized and transmitted to all clients in each communication round, and each client picks the model with the smallest loss value to optimize. FedCluster [72] iteratively bipartition the clients based on their cosine similarity of gradients. GCFL [95] generalizes this idea to graph data, enabling collaborative training with graphs from different domains. Observing that the gradients of GNNs can be fluctuating, GCFL+ [95] uses a gradient sequence-based clustering mechanism to form more robust clusters.

Personalized Modules. Another prevalent way for PFL is personalized modules. In these works, the machine learning model is divided into two parts: the shared part and the personalized part. The key is to design a model structure suitable for personalization. For example, when a model is split into a feature extractor and classifier, FedPer [2] shares the feature extractor and personalizes the classifier, while LG-FedAvg [49] personalizes the feature extractor and shares the classifier. Similar techniques in used in FMTGL [54] and NGL-FedRep [80]. Moreover, PartialFed [75] can automatically select which layers to personalize and which layers to share. On graph data, [78] observe that while the feature information can be very different, some structural properties are shared by various domains, revealing the great potential for sharing structural information in FL. Inspired by this, they propose FedStar that trains a feature-structure decoupled GNN. The structural encoder is globally shared across all clients, while the feature-based knowledge is personalized.

Local Finetuning and Meta-Learning. Finetuning is widely used for PFL. In these works, a global model is first trained with all clients. The global model encodes the information of the population but may not adapt to each client's own distribution. Therefore, each client locally finetunes the global model with a few steps of gradient descent. Besides vanilla finetuning, Per-FedAvg [13] combines FL with

MAML [15], an algorithm for meta-learning, to improve the performance of finetuning. Similarly, pFedMe [10] utilize Moreau Envelopes for personalization. It adds a proximal term to the local finetuning objective, and aims to find a local model near the global model, with just a few steps of gradient descent. GraphFL [83] applies a similar meta-learning framework on graph data, addressing the heterogeneity among graph data and handling new label domains with a few new labeled nodes.

Multi-task Learning. PFL is also studied within the framework of multi-task learning. MOCHA [73] uses a matrix to model the similarity among each pair of clients. Clients with similar distribution will be encouraged to have similar model parameters. FedGMTL [31] generalizes this idea to graph data. Similarly, SemiGraphFL [79] computes pairwise cosine similarity among clients' hidden representations. As a result, clients with more similar data will have greater mutual influence. However, it requires the transmission of hidden representation. FedEM [57] assumes that each client's distribution is a mixture of unknown underlying distributions and proposes FedEM, an EM-like algorithm for multi-task FL. Finally, FedFOMO [107] allows each client to have a different mixture weight of local models during the aggregation steps. It provides a flexible way for model aggregation.

Graph Structure Augmentation. In the previous works, graph structures are considered as ground truth. However, graphs can be noisy or incomplete, which can hurt the performance of GNNs. To tackle incomplete graph structures, FedGSL [109] optimizes the local client's graph and GNN parameters simultaneously.

3.3 Subgraph-level FL

Similar to graph-level FL, each client in subgraph-level FL holds one graph. However, clients' graphs are a subgraph of a latent large entire graph. In other words, there are cross-client edges in the entire graph, where the two nodes of these edges belong to different clients. The task is usually node-level, while the cross-client edges can contribute to the task.

One application of subgraph-level FL is financial fraud detection. Each FL client is a bank aiming to detect potential fraud with transaction data. Each bank keeps a graph of the information of its customers, where nodes are individual customers and edges are transaction records. While each bank holds its own graph, customers in one bank may have connections to customers in another bank, introducing edges across clients. These cross-client edges help to train better ML models.

The biggest challenge for subgraph-level FL is to handle cross-client edges. In GNNs, each node iteratively aggregates information from its neighboring nodes, which may be from other clients. However, during local updates in traditional FL, clients cannot get access to the data from other clients. Directly exchanging raw data among clients is prohibited due to privacy concerns. It is challenging to enable cross-client information exchange while preserving privacy. Moreover, when nodes' identities are not shared across clients, the cross-client edges can be missing and stored in none of the clients. Even if we collect clients' local subgraphs, we cannot reconstruct the global graph.

In this subsection, we will mainly focus on two scenarios. In the first part, we introduce algorithms when the hidden entire graph is given but stored separately in different clients. In the second part, we consider a more challenging setting: the cross-client edges are missing, and we cannot simply concatenate local graphs to reconstruct the entire graph losslessly. We focus on how to generate these missing edges or missing neighbors for each node.

3.3.1 Cross-client Propagation

When the cross-client edges are available, the major challenge is to enable cross-client information propagation without leaking raw data. FedGraph [8] designs a novel cross-client convolution operation to avoid sharing raw data across clients. It avoids exchanging representations in the first GCN layer. Similarly, FedPNS [11] control the number of neighbor sampling to reduce communication costs. FedCog [43] proposes graph decoupling operation, splitting local graph to internal graph and border graph. The graph convolution is accordingly divided into two sequential steps: internal propagation and border propagation. In this process, each client sends the intermediate representation of internal nodes to other clients. Considering that directly exchanging feature representations between clients can leak private information. In user-item graphs, FedPerGNN [92] design a privacy-preserving user-item graph expansion protocol. Clients upload encrypted item IDs to the trusted server, and the server matches the ciphertexts of item IDs to find clients with overlapping item IDs. DP-FedRec [65] uses private set intersection to exchange the edges information between clients and applies differential privacy techniques to further protect privacy. Different from the above methods, FedGCN [98] does not rely on communication between clients. Instead, it transmits all the information needed to train a GCN between the server and each client, only once before the training. Moreover, each node at a given client only needs to know the accumulated information about the node's neighbors, which reduces possible privacy leakage.

3.3.2 Missing Neighbors

For some applications, the cross-client edges can be missing or not stored in any clients. Notice that although each client also holds a disjoint graph in graph-level FL, graph-level FL and subgraph-level FL with missing neighbors are substantially different. For graph-level FL, there are essentially no cross-client edges. For example, there are no chemical bonds between two molecules from different corporations' datasets. However, for subgraph-level FL, the cross-client edges exist, but are missing in certain applications. We may get suboptimal GNN models if ignoring the existence of cross-client edges. Therefore, the major challenge is to reconstruct these missing edges, or reconstruct missing neighbors for each node.

FedSAGE [106] first defines the missing neighbors' challenge, and proposes a method the generate pseudo neighbors for each node. It uses existing subgraphs to train a neighbors generator and generate one-hop neighbors for each client to mend the graph. Since missing neighbors are generated locally, no feature exchange is required between clients after the local subgraphs are mended. However, the training of neighbor generators requires cross-client hidden representation exchanges. Similarly, FedNI [63] uses a graph GAN model to generate missing nodes and edges.

3.4 Node-level FL

The final application scenario of graph federated learning is

Table 2: Repositories for Graph Federated Learning

Scenario	Name	Description	Link
Graph-level	FedProx [47]	A general GFL algorithm with modified local update	Github
	IFCA [27]	A general clustered FL algorithm	Github
	GCFL [95]	A graph-specific clustered FL algorithm	Github
	LG-FedAvg [49]	A general PFL algorithm with personalized modules	Github
	FedStar [78]	A graph-specific PFL algorithm with personalized modules	Github
	pFedMe [10]	A general PFL algorithm based on meta-learning	Github
	GraphFL [83]	A graph-specific PFL algorithm based on meta-learning	Github
	FedFOMO [107]	A general PFL algorithm based on multi-task learning	Github
Subgraph-level	FedGCN [98]	An FL algorithm with one-shot cross-client propagation	Github
	FedSAGE [106]	An FL algorithm with missing neighbors generation	Github
Node-level	SpreadGNN [31]	A serverless PFL algorithm	Github
	FedGS [89]	An FL algorithm with graph as distribution similarities	Github
	SFL [9]	An GL with pre-defined graph for server aggregation	Github
Others	TensorFlow Federated	A framework for implementing federated learning	Github
	FedLab [103]	A Flexible Federated Learning Framework	Github
	PFL-Non-IID	Reproduction of popular PFL algorithms	Github
	FedGraphNN [30]	FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks	Github
	FederatedScope-GNN [90]	A unified, comprehensive and efficient package for Federated Graph Learning	Github

node-level. Different from the aforementioned two scenarios, each client in node-level FL can hold any type of data, not restricted to graphs. Instead, the clients themselves are nodes in a graph, while the edges are their pairwise relationship of communication or distribution similarity.

One typical application of node-level FL is the Internet of Things (IOT) devices in a smart building [68]. Due to bandwidth constraints, it can be costly for each IoT device to communicate with the central server. However, IoT devices in the same local area network can communicate very efficiently. As a result, IoT devices form a graph with pairwise communication availability as edges. Another application is for the smart city [89], where clients are traffic sensors deployed on the road and linked to geographically adjacent sensors. Each device can collect data and make the real-time decision without waiting for the response of cloud servers. Each device needs to make an intelligent decision based on the collected road conditions and nearby devices.

In this subsection, we will first introduce algorithms where the graph models communication constraints among clients. In these works, there is no central server, and clients can only exchange information along edges. Then, we will introduce algorithms where the graph models the relationship between clients' distributions. In these works, although a central server is available, the graph among clients models distributional similarity or dependency among clients, potentially contributes to the model performance.

3.4.1 Graph as Communication Network

Traditional FL relies on a central server to enable communication among clients. Each client trusts the central server and uploads their model update to the server. However, in many scenarios, a trusted central server may not exist. Even when a central server exists, it may be expensive for clients to communicate with the server. Therefore, serverless FL (a.k.a. peer-to-peer FL) has been studied to relieve communication constraints.

The standard solution for serverless FL is fully decentralized FL [42; 41], where each client only averages its model parameter with its neighbors. D-FedGNN [62] uses these techniques to train GNN models. SpreadGNN [31] generalizes this framework to personalized FL, where each client has non-IID data and a different label space.

3.4.2 Graph as Distribution Similarities

When the central server is available, a graph of clients may still be beneficial when it models distributional relationships among clients. When edges link clients with highly similar distributions, parameter sharing along edges can potentially improve the model performance for both clients. When edges link clients with data dependency, information exchange along edges can even provide additional features for inference.

FedGS [89] models the data correlations of clients with a data-distribution-dependency graph, and improves the unbiasedness of the client sampling process. Meanwhile, SFL [9] assumes a pre-defined client relation graph stored on the server, and the client-centric model aggregation is conducted along the relation graph's structure. GraphFL [108] considers client-side information to encourage similar clients to have similar models. BiG-Fed [96] applies graph convolution on the client graph, so each client's prediction can benefit from its neighbors with highly correlated data. Finally, [86] designs a client sampling technique considers both communication cost and distribution similarity.

Finally, we summarize the official implementation of FL algorithms and useful repositories in Table 2.

3.5 Challenges and Future Opportunities

In this part, we present several limitations in current works and provide open problems for future research.

3.5.1 Model Heterogeneity for Graph-Level FL

In previous works of graph-level FL, although each FL client usually has different data distribution it is usually assumed that the model architecture is shared across all clients. However, the optimal architecture for different clients can be

different. For example, a well-known issue in GNNs is the over-smoothing problem. When the number of graph convolutional layers is higher than the diameter of the graph, GNN models may learn similar representations for all nodes in the graph, which harms the model performance. When each FL clients hold a substantially different size of graphs, it is highly likely that the optimal depth of the GNN model is different for them.

3.5.2 *Avoiding Cross-Client Transmission for Subgraph-Level FL*

Most of the previous subgraph-level FL algorithms highly rely on direct information exchange along cross-client edges. While such operations are natural variants of graph convolution, such operations also raise privacy concerns. Moreover, different from traditional FL where each client downloads aggregated model parameters that reveal the population, feature exchange along the edges can expose information about individuals. It would be beneficial if the cross-client transmission can be avoided without greatly degrading the model.

4. ENVISIONING

In this section, we analyze the current developments and limitations of privacy-preserving graph machine learning, and explain the necessity of combining them. In addition, we identify a number of unsolved research directions that could be addressed to improve the privacy of graph machine learning systems.

4.1 Limitation of Current Techniques

In the previous two sections, we introduced privacy-preserving graph data generation and computation, respectively. However, both techniques have their own limitations.

- For privacy-preserving graph generation, while it can provide good privacy protection for graph data, it also has a significant drawback on model utility. The privacy-preserving techniques applied during data generation are not designed for specific machine learning tasks and may influence the utility of the resulting model. For example, consider a graph with four nodes a, b, c, and d. The nodes a and b have a positive label, while c and d have a negative label. Switching the edges from $(a, b), (c, d)$ to $(a, c), (b, d)$ does not change the degree distribution of the graph, but it changes the graph from a homophilous graph to a heterophilous graph, i.e., edges are more likely to link two nodes with different labels. This change can harm the performance of many GNN models, which are designed to work well with homogeneous graphs [50]. It is important to consider the downstream machine learning tasks when designing privacy-preserving techniques for graph data.

- For privacy-preserving graph computation, while FL can avoid the transmission of raw data, it has been shown that transmitting raw model parameters or gradients may not provide enough privacy, as attackers can use the gradient or model update to reconstruct private data [116; 26]. Moreover, many subgraph-level and node-level federated learning algorithms require the transmission of hidden representations, which can also leak private information. Therefore, protecting the raw data from being reconstructed is essential to federated learning systems.

4.2 Combination of Privacy-Preserving Graph Data Generation and Computation

To address the limitations of current privacy-preserving techniques, it is essential to combine privacy graph data generation with the graph federated learning frameworks, as shown in Figure 5. This approach can provide an effective solution to the privacy preservation issues of graph machine learning models.

Specifically, the generated synthetic data is used instead of the real data during the training process. This means that even if the transmitted information is decrypted, it is just from the generated synthetic data and not the real data. The synthetic data can be generated in such a way that it preserves the statistical properties of the original data while ensuring privacy preservation. This can be achieved using various techniques, including differential privacy, homomorphic encryption, and secure multi-party computation.

The combination of privacy graph data generation and graph federated learning frameworks has several benefits. First, it ensures privacy preservation during the training process by using synthetic data. Second, it enables the transfer of graph machine learning model parameters rather than embedding vectors or other information. This can improve the accuracy and efficiency of the model. Finally, it provides a robust defense against privacy attacks and reverse-engineering, as the transmitted information is just from the generated synthetic data and not the real data.

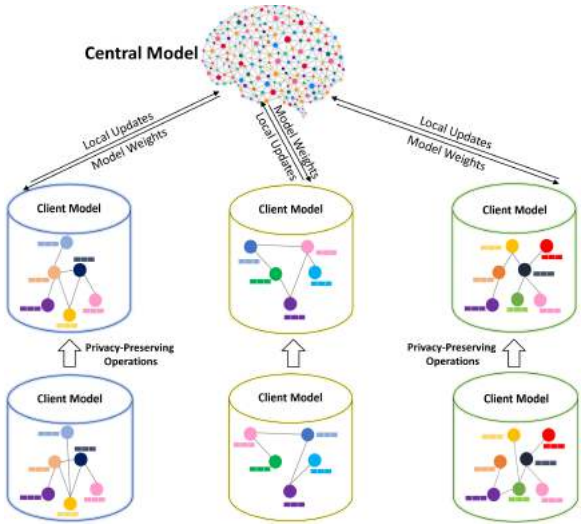


Figure 5: Privacy-preserving Graph Data with Privacy-preserving Computation.

4.3 Future Directions

Combining privacy-preserving data generation and computation is a promising approach to protect individual privacy while maintaining model utility in machine learning. However, it also poses several challenges and possible future directions.

4.3.1 Distribution of Privacy Budget

When combining privacy-preserving data generation with computation, noises are added to both raw data and model parameters. However, it is still unclear how to distribute the privacy budget between data generation and computation in a way that optimizes the privacy-utility trade-off. In this approach, noises are added to the graph data during data generation and to the model parameters during data computation (i.e., federated learning), which results in an overall reduction in accuracy. However, while the privacy analysis for data generation is directly defined on the data space, the privacy analysis for federated learning requires transforming the change on parameter space back to data space. Such transformation requires estimating the sensitivity of a machine learning algorithm (i.e., how the change of a data point affects the learned parameters), which is only loosely bounded in current works [59; 91]. A more precise analysis of privacy is required to better understand the impact of privacy budget allocation on the overall privacy-utility trade-off.

4.3.2 Parameter Information Disentanglement

Another future challenge when combining privacy-preserving data generation and computation is the disentanglement of task-relevant and task-irrelevant information. Currently, the noise added to the model parameters is isotropic, meaning that task-relevant and task-irrelevant information are equally protected. However, not all information is equally important for model utility. If we can identify which information has a significant influence on model performance, we can distribute more privacy budget to this information while allocating less privacy budget to task-irrelevant information. This can result in a better privacy-utility trade-off. Disentangling task-relevant and task-irrelevant information would require a more sophisticated analysis of model architecture and data characteristics to determine which features contribute most to model performance.

5. CONCLUSION

In this paper, we review the research for privacy-preserving techniques for graph machine learning from the data to the computation, considering the situation where the data need to be shared or are banned from being transmitted. To be specific, for privacy-preserving graph data generation techniques, we analyze the forceful attackers first and then introduce how corresponding protection methods are proposed to defend attackers. For the privacy graph data computation, we circle around the federated learning setting and discuss how the general federated learning framework applied to graph data and what the potential challenges originated from non-IIDness, and how the nascent research works address them. In the end, we analyze the current limitation and propose several promising research directions.

6. ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation (1947203, 2117902, 2137468, 1947135, 2134079, and 1939725), the U.S. Department of Homeland Security (2017-ST-061-QA0001, 17STQAC00001-06-00, and 17STQAC00001-03-03), DARPA (HR001121C0165), NIFA (2020-67021-32799), and ARO (W911NF2110088). The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

7. REFERENCES

[1] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *SIGSAC 2016*, 2016.

[2] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019.

[3] R. Arora and J. Upadhyay. On differentially private graph sparsification and applications. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13378–13389, 2019.

[4] S. Arora, A. Risteski, and Y. Zhang. Do gans learn the distribution? some theory and empirics. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[5] L. Backstrom, C. Dwork, and J. M. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 181–190. ACM, 2007.

[6] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In R. D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 87–96. ACM, 2013.

[7] C. Chen, Y. Wu, Q. Dai, H. Zhou, M. Xu, S. Yang, X. Han, and Y. Yu. A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective. *CoRR*, abs/2209.13232, 2022.

[8] F. Chen, P. Li, T. Miyazaki, and C. Wu. Fedgraph: Federated graph learning with intelligent sampling. *IEEE Trans. Parallel Distributed Syst.*, 33(8):1775–1786, 2022.

[9] F. Chen, G. Long, Z. Wu, T. Zhou, and J. Jiang. Personalized federated learning with a graph. In L. D. Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2575–2582. ijcai.org, 2022.

[10] C. T. Dinh, N. H. Tran, and T. D. Nguyen. Personalized federated learning with moreau envelopes. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information*

Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.

[11] B. Du and C. Wu. Federated graph learning with periodic neighbour sampling. In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2022.

[12] E. Ekblom, E. L. Zec, and O. Mogren. EFFGAN: ensembles of fine-tuned federated gans. In S. Tsumoto, Y. Ohsawa, L. Chen, D. V. den Poel, X. Hu, Y. Motomura, T. Takagi, L. Wu, Y. Xie, A. Abe, and V. Raghavan, editors, *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*, pages 884–892. IEEE, 2022.

[13] A. Fallah, A. Mokhtari, and A. E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[14] H. Fichtenberger, M. Henzinger, and W. Ost. Differentially private algorithms for graphs under continual observation. In P. Mutzel, R. Pagh, and G. Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 42:1–42:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[15] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017.

[16] D. Fu, L. Fang, R. Maciejewski, V. I. Torvik, and J. He. Meta-learned metrics over multi-evolution temporal graphs. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, 2022.

[17] D. Fu and J. He. SDG: A simplified and dynamic graph neural network. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 2021.

[18] D. Fu and J. He. DPPIN: A biological repository of dynamic protein-protein interaction network data. In *IEEE International Conference on Big Data, Big Data 2022, Osaka, Japan, December 17-20, 2022*, 2022.

[19] D. Fu and J. He. Natural and artificial dynamics in graphs: Concept, progress, and future. *Frontiers in Big Data*, 5, 2022.

[20] D. Fu, J. He, H. Tong, and R. Maciejewski. Privacy-preserving graph analytics: secure generation and federated learning. *arXiv preprint arXiv:2207.00048*, 2022.

[21] D. Fu, Z. Xu, B. Li, H. Tong, and J. He. A view-adversarial framework for multi-view network embedding. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, 2020.

[22] D. Fu, D. Zhou, and J. He. Local motif clustering on time-evolving graphs. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, 2020.

[23] D. Fu, D. Zhou, R. Maciejewski, A. Croitoru, M. Boyd, and J. He. Fairness-aware clique-preserving spectral clustering of temporal graphs. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, 2023.

[24] X. Fu, B. Zhang, Y. Dong, C. Chen, and J. Li. Federated graph machine learning: A survey of concepts, techniques, and applications. *SIGKDD Explor.*, 24(2):32–47, 2022.

[25] T. Gaudelet, B. Day, A. R. Jamasb, J. Soman, C. Regep, G. Liu, J. B. R. Hayter, R. Vickers, C. Roberts, J. Tang, D. Roblin, T. L. Blundell, M. M. Bronstein, and J. P. Taylor-King. Utilizing graph machine learning within drug discovery and development. *Briefings Bioinform.*, 22(6), 2021.

[26] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[27] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran. An efficient framework for clustered federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[28] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[29] M. Hay, G. Miklau, D. D. Jensen, D. F. Towsley, and P. Weis. Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114, 2008.

[30] C. He, K. Balasubramanian, E. Ceyani, Y. Rong, P. Zhao, J. Huang, M. Annavaram, and S. Avestimehr. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *CoRR*, abs/2104.07145, 2021.

[31] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr. Spreadgnn: Decentralized multi-task federated learning for graph neural networks on molecular data. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 6865–6873. AAAI Press, 2022.

[32] A. Hoang, B. Carminati, and E. Ferrari. Cluster-based anonymization of knowledge graphs. In M. Conti, J. Zhou, E. Casalicchio, and A. Spognardi, editors, *Applied Cryptography and Network Security - 18th International Conference, ACNS 2020, Rome, Italy, October 19-22, 2020, Proceedings, Part II*, volume 12147 of *Lecture Notes in Computer Science*, pages 104–123. Springer, 2020.

[33] A. Hoang, B. Carminati, and E. Ferrari. Privacy-preserving sequential publishing of knowledge graphs. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 2021–2026. IEEE, 2021.

[34] A.-T. Hoang, B. Carminati, and E. Ferrari. Time-aware anonymization of knowledge graphs. *ACM Transactions on Privacy and Security*, 2022.

[35] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Networks Learn. Syst.*, 33(2):494–514, 2022.

[36] H. Jiang, J. Pei, D. Yu, J. Yu, B. Gong, and X. Cheng. Applications of differential privacy in social network analysis: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(1):108–127, 2023.

[37] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. A. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1-2):1–210, 2021.

[38] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. SCAFFOLD: stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 2020.

[39] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. D. Smith. Analyzing graphs with node differential privacy. In A. Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 457–476. Springer, 2013.

[40] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[41] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar. Peer-to-peer federated learning on graphs. *CoRR*, abs/1901.11173, 2019.

[42] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar. Fully decentralized federated learning. In *Third workshop on bayesian deep learning (NeurIPS)*, volume 2, 2018.

[43] R. Lei, P. Wang, J. Zhao, L. Lan, J. Tao, C. Deng, J. Feng, X. Wang, and X. Guan. Federated learning over coupled graphs. *IEEE Trans. Parallel Distributed Syst.*, 34(4):1159–1172, 2023.

[44] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In R. Chirkova, A. Dogac, M. T. Özsu, and T. K. Sellis, editors, *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 106–115. IEEE Computer Society, 2007.

[45] Q. Li, B. He, and D. Song. Model-contrastive federated learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 10713–10722. Computer Vision Foundation / IEEE, 2021.

[46] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.*, 37(3):50–60, 2020.

[47] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In I. S. Dhillon, D. S. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.

[48] Z. Li, D. Fu, and J. He. Everything evolves in personalized pagerank. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, 2023.

[49] P. P. Liang, T. Liu, Z. Liu, R. Salakhutdinov, and L. Morency. Think locally, act globally: Federated learning with local and global representations. *CoRR*, abs/2001.01523, 2020.

[50] D. Lim, X. Li, F. Hohne, and S. Lim. New benchmarks for learning on non-homophilous graphs. *CoRR*, abs/2104.01404, 2021.

[51] K. Liu, K. Das, T. Grandison, and H. Kargupta. Privacy-preserving data analysis on graphs and social

networks. In H. Kargupta, J. Han, P. S. Yu, R. Motwani, and V. Kumar, editors, *Next Generation of Data Mining*, Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press / Chapman and Hall / Taylor & Francis, 2008.

[52] K. Liu and E. Terzi. Towards identity anonymization on graphs. In J. T. Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 93–106. ACM, 2008.

[53] R. Liu and H. Yu. Federated graph neural networks: Overview, techniques and challenges. *CoRR*, abs/2202.07256, 2022.

[54] Y. Liu, D. Han, J. Zhang, H. Zhu, M. Xu, and W. Chen. Federated multi-task graph learning. *ACM Trans. Intell. Syst. Technol.*, 13(5), jun 2022.

[55] X. Ma, J. Wu, S. Xue, J. Yang, Q. Z. Sheng, and H. Xiong. A comprehensive survey on graph anomaly detection with deep learning. *CoRR*, abs/2106.07178, 2021.

[56] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. *L*-diversity: Privacy beyond *k*-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1):3, 2007.

[57] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal. Federated multi-task learning under a mixture of distributions. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15434–15447, 2021.

[58] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 2017.

[59] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[60] H. H. Nguyen, A. Imine, and M. Rusinowitch. Anonymizing social graphs via uncertainty semantics. In *CCS 2015*, 2015.

[61] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[62] Y. Pei, R. Mao, Y. Liu, C. Chen, S. Xu, F. Qiang, and B. E. Tech. Decentralized federated graph neural networks. In *International Workshop on Federated and Transfer Learning for Data Sparsity and Confidentiality in Conjunction with IJCAI*, 2021.

[63] L. Peng, N. Wang, N. Dvornek, X. Zhu, and X. Li. Fedni: Federated graph learning with network inpainting for population-based disease prediction. *IEEE Transactions on Medical Imaging*, pages 1–1, 2022.

[64] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren. Generating synthetic decentralized social graphs with local differential privacy. In *CCS 2017*, 2017.

[65] Y. Qiu, C. Huang, J. Wang, Z. Huang, and J. Xiao. A privacy-preserving subgraph-level federated graph neural network via differential privacy. In G. Memmi, B. Yang, L. Kong, T. Zhang, and M. Qiu, editors, *Knowledge Science, Engineering and Management*, pages 165–177, Cham, 2022. Springer International Publishing.

[66] S. Raskhodnikova and A. D. Smith. Lipschitz extensions for node-private graph statistics and the generalized exponential mechanism. In I. Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 495–504. IEEE Computer Society, 2016.

[67] M. Rasouli, T. Sun, and R. Rajagopal. Fedgan: Federated generative adversarial networks for distributed data. *CoRR*, abs/2006.07228, 2020.

[68] A. Rasti-Meymandi, S. M. Sheikholeslami, J. Abouei, and K. N. Plataniotis. Graph federated learning for ciot devices in smart home applications. *IEEE Internet of Things Journal*, 10(8):7062–7079, 2023.

[69] L. Rossi, M. Musolesi, and A. Torsello. On the k-anonymization of time-varying and multi-layer social graphs. In M. Cha, C. Mascolo, and C. Sandvig, editors, *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*, pages 377–386. AAAI Press, 2015.

[70] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao. Sharing graphs using differentially private graph models. In P. Thiran and W. Willinger, editors, *Proceedings of the 11th ACM SIGCOMM Internet Measurement Conference, IMC '11, Berlin, Germany, November 2-, 2011*, pages 81–98. ACM, 2011.

[71] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.

[72] F. Sattler, K. Müller, and W. Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Trans. Neural Networks Learn. Syst.*, 32(8):3710–3722, 2021.

[73] V. Smith, C. Chiang, M. Sanjabi, and A. Talwalkar. Federated multi-task learning. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4424–4434, 2017.

[74] S. Song, S. Little, S. Mehta, S. A. Vinterbo, and K. Chaudhuri. Differentially private continual release of graph statistics. *CoRR*, abs/1809.02575, 2018.

[75] B. Sun, H. Huo, Y. Yang, and B. Bai. Partialfed: Cross-domain personalized federated learning via partial initialization. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23309–23320, 2021.

[76] Y. Sun and J. Han. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explor.*, 14(2):20–28, 2012.

[77] A. Z. Tan, H. Yu, L. Cui, and Q. Yang. Towards personalized federated learning. *CoRR*, abs/2103.00710, 2021.

[78] Y. Tan, Y. Liu, G. Long, J. Jiang, Q. Lu, and C. Zhang. Federated learning on non-iid graphs via structural knowledge sharing. *CoRR*, abs/2211.13009, 2022.

[79] Y. Tao, Y. Li, and Z. Wu. Semigraphfl: Semi-supervised graph federated learning for graph classification. In G. Rudolph, A. V. Kononova, H. Aguirre, P. Kerschke, G. Ochoa, and T. Tušar, editors, *Parallel Problem Solving from Nature – PPSN XVII*, pages 474–487, Cham, 2022. Springer International Publishing.

[80] A. Thakur, P. Sharma, and D. A. Clifton. Dynamic neural graphs based federated reptile for semi-supervised multi-tasking in healthcare applications. *IEEE Journal of Biomedical and Health Informatics*, 26(4):1761–1772, 2022.

[81] J. R. Ullman and A. Sealfon. Efficiently estimating erdos-renyi graphs with node differential privacy. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3765–3775, 2019.

[82] J. Upadhyay, S. Upadhyay, and R. Arora. Differentially private analysis on graph streams. In A. Banerjee and K. Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 1171–1179. PMLR, 2021.

[83] B. Wang, A. Li, M. Pang, H. Li, and Y. Chen. Graphfl: A federated learning framework for semi-supervised node classification on graphs. In X. Zhu, S. Ranka, M. T. Thai, T. Washio, and X. Wu, editors, *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, pages 498–507. IEEE, 2022.

[84] J. Wang, Z. Charles, Z. Xu, G. Joshi, H. B. McMahan, B. A. y Arcas, M. Al-Shedivat, G. Andrew, S. Avestimehr, K. Daly, D. Data, S. N. Diggavi, H. Eichner, A. Gadhikar, Z. Garrett, A. M. Girgis, F. Hanzely, A. Hard, C. He, S. Horváth, Z. Huo, A. Ingerman, M. Jaggi, T. Javidi, P. Kairouz, S. Kale, S. P. Karimireddy, J. Konečný, S. Koyejo, T. Li, L. Liu, M. Mohri, H. Qi, S. J. Reddi, P. Richtárik, K. Singhal, V. Smith, M. Soltanolkotabi, W. Song, A. T. Suresh, S. U. Stich, A. Talwalkar, H. Wang, B. E. Woodworth, S. Wu, F. X. Yu, H. Yuan, M. Zaheer, M. Zhang, T. Zhang, C. Zheng, C. Zhu, and W. Zhu. A field guide to federated optimization. *CoRR*, abs/2107.06917, 2021.

[85] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, L. Cao, F. Ricci, and P. S. Yu. Graph learning based recommender systems: A review. In Z. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4644–4652. ijcai.org, 2021.

[86] S. Wang, M. Lee, S. Hosseinalipour, R. Morabito, M. Chiang, and C. G. Brinton. Device sampling for heterogeneous federated learning: Theory, algorithms, and implementation. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, pages 1–10, 2021.

[87] Y. Wang and X. Wu. Preserving differential privacy in degree-correlation based graph generation. *Trans. Data Priv.*, 6(2):127–145, 2013.

[88] Y. Wang, X. Wu, and L. Wu. Differential privacy preserving spectral graph analysis. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, editors, *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, volume 7819 of *Lecture Notes in Computer Science*, pages 329–340. Springer, 2013.

[89] Z. Wang, X. Fan, J. Qi, H. Jin, P. Yang, S. Shen, and C. Wang. Fedgs: Federated graph-based sampling with arbitrary client availability. *CoRR*, abs/2211.13975, 2022.

[90] Z. Wang, W. Kuang, Y. Xie, L. Yao, Y. Li, B. Ding, and J. Zhou. Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning. In A. Zhang and H. Rangwala, editors, *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 4110–4120. ACM, 2022.

[91] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.*, 15:3454–3469, 2020.

[92] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 13(1):3091, Jun 2022.

[93] L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, J. Pei, and B. Long. Graph neural networks for natural language processing: A survey. *Found. Trends Mach. Learn.*, 16(2):119–328, 2023.

[94] X. Wu, X. Ying, K. Liu, and L. Chen. *A Survey of Privacy-Preservation of Graphs and Social Networks*, pages 421–453. Springer US, Boston, MA, 2010.

[95] H. Xie, J. Ma, L. Xiong, and C. Yang. Federated graph classification over non-iid graphs. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 18839–18852, 2021.

[96] P. Xing, S. Lu, L. Wu, and H. Yu. Big-fed: Bilevel optimization enhanced graph-aided federated learning. *IEEE Transactions on Big Data*, pages 1–12, 2022.

[97] C. Yang, H. Wang, K. Zhang, L. Chen, and L. Sun. Secure deep graph generation with link differential privacy. In *IJCAI 2021*, 2021.

[98] Y. Yao and C. Joe-Wong. Fedgcn: Convergence and communication tradeoffs in federated training of graph convolutional networks. *CoRR*, abs/2201.12433, 2022.

[99] X. Ying and X. Wu. Randomizing social networks: a spectrum preserving approach. In *SDM 2008*, 2008.

[100] X. Ying and X. Wu. Graph generation with prescribed feature constraints. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 966–977. SIAM, 2009.

[101] X. Ying and X. Wu. On link privacy in randomizing social networks. *Knowl. Inf. Syst.*, 28(3):645–663, 2011.

[102] T. Yoon, S. Shin, S. J. Hwang, and E. Yang. Fedmix: Approximation of mixup under mean augmented federated learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[103] D. Zeng, S. Liang, X. Hu, H. Wang, and Z. Xu. Fedlab: A flexible federated learning framework. *Journal of Machine Learning Research*, 24(100):1–7, 2023.

[104] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[105] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao. Private release of graph statistics using ladder functions. In T. K. Sellis, S. B. Davidson, and Z. G. Ives, editors, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 731–745. ACM, 2015.

[106] K. Zhang, C. Yang, X. Li, L. Sun, and S. Yiu. Subgraph federated learning with missing neighbor generation. In M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6671–6682, 2021.

[107] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez. Personalized federated learning with first order model optimization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[108] Y. Zhang, S. Wei, S. Liu, Y. Wang, Y. Xu, Y. Li, and X. Shang. Graph-regularized federated learning with shareable side information. *Knowledge-Based Systems*, 257:109960, 2022.

[109] G. Zhao, Y. Huang, and C. H. Tsai. Fedgsl: Federated graph structure learning for local subgraph augmentation. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 818–824, 2022.

[110] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018.

[111] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. In F. Bonchi, E. Ferrari, B. A. Malin, and Y. Saygin, editors, *Privacy, Security, and Trust in KDD, First ACM SIGKDD International Workshop, PinKDD 2007, San Jose, CA, USA, August 12, 2007, Revised Selected Papers*, volume 4890 of *Lecture Notes in Computer Science*, pages 153–171. Springer, 2007.

[112] B. Zhou and J. Pei. Preserving privacy in social networks against neighborhood attacks. In G. Alonso, J. A. Blakeley, and A. L. P. Chen, editors, *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, pages 506–515. IEEE Computer Society, 2008.

[113] B. Zhou, J. Pei, and W. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor.*, 10(2):12–22, 2008.

[114] D. Zhou, L. Zheng, J. Han, and J. He. A data-driven graph generative model for temporal interaction networks. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020.

[115] D. Zhou, L. Zheng, J. Xu, and J. He. Misc-gan: A multi-scale generative model for graphs. *Frontiers Big Data*, 2:3, 2019.

[116] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Annual Conference on Neural Information Processing Systems*, 2019.

[117] L. Zou, L. Chen, and M. T. Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proc. VLDB Endow.*, 2(1):946–957, 2009.