PGB: Benchmarking Differentially Private Synthetic Graph Generation Algorithms

Shang Liu[1,3], Hao Du[2], Yang Cao[3], Bo Yan[4,3], Jinfei Liu[5], Masatoshi Yoshikawa[6]

[1]Kyoto University, [2]Hokkaido University, [3]Tokyo Institute of Technology,
[4]Beijing University of Posts and Telecommunications, [5]Zhejiang University, [6]Osaka Seikei University

liu.shang.33s@st.kyoto-u.ac.jp, hao.du.y4@elms.hokudai.ac.jp, cao@c.titech.ac.jp,
boyan@bupt.edu.cn, jinfeiliu@zju.edu.cn, yoshikawa-mas@osaka-seikei.ac.jp

***Abstract*—Differentially private graph analysis is a powerful tool for deriving insights from diverse graph data while protecting individual information. Designing private analytic algorithms for different graph queries often requires starting from scratch. In contrast, differentially private synthetic graph generation offers a general paradigm that supports one-time generation for multiple queries. Although a rich set of differentially private graph generation algorithms has been proposed, comparing them effectively remains challenging due to various factors, including differing privacy definitions, diverse graph datasets, varied privacy requirements, and multiple utility metrics.**

To this end, we propose PGB (Private Graph Benchmark), a comprehensive benchmark designed to enable researchers to compare differentially private graph generation algorithms fairly. We begin by identifying four essential elements of existing works as a 4-tuple: mechanisms, graph datasets, privacy requirements, and utility metrics. We discuss principles regarding these elements to ensure the comprehensiveness of a benchmark. Next, we present a benchmark instantiation that adheres to all principles, establishing a new method to evaluate existing and newly proposed graph generation algorithms. Through extensive theoretical and empirical analysis, we gain valuable insights into the strengths and weaknesses of prior algorithms. Our results indicate that there is no universal solution for all possible cases. Finally, we provide guidelines to help researchers select appropriate mechanisms for various scenarios.

***Index Terms*—differential privacy, benchmark, synthetic graph generation.**

I. Introduction

Graph analysis serves as an effective method for deriving insights from diverse graph datasets, including social networks, traffic networks, and epidemiological networks. For instance, the degree distribution [1]–[3], which counts the connections per node, illuminates the connectivity within social graphs. Additionally, subgraph counting [4]–[6], such as triangles or stars, aids in assessing central properties like the clustering coefficient [7], reflecting the probability that two connections of an individual are mutually linked. However, publicly sharing these graph statistics risks disclosing personal details [8], as graph analytics are often conducted over sensitive information.

Differential privacy (DP) [9], [10] has become the defacto standard for privacy preservation, providing individual privacy against adversaries with arbitrary background knowledge. Unlike previous privacy definitions (e.g., k-anonymity, l-diversity, t-closeness), DP ensures that modifications of a single node or edge have a minimal impact on the output results. Many differentially private graph analytic algorithms have been designed for various graph queries, such as degree distribution [1]–[3], subgraph counts [4]–[6], and community detection [11]–[13]. Unfortunately, these solutions are usually tailored to specific graph queries. For different queries, differentially private graph algorithms must be designed from scratch. One solution is to generate a synthetic graph that maintains semantic similarity to the original graph while satisfying DP. This paradigm is superior to tailored algorithms as it enables one-time generation for multiple queries.

Despite a rich set of differentially private synthetic graph generation algorithms [14]–[29] having been proposed, there is no generally acknowledged and unified procedure to perform empirical studies on them. Concretely, it is challenging to compare them effectively due to the following factors:

- Algorithms use different privacy definitions to protect individual information in a graph, such as edge differential privacy [14]–[21] and node differential privacy [22], [23]. It is unfair to compare algorithms based on different privacy definitions.
- Few algorithms in our literature survey offer open-source support. Correctly re-implementing differentially private graph generation algorithms can be challenging due to their intrinsic complexity.
- Many algorithms in publications exhibit data-dependent errors. Their utility depends on the choice of input graph characteristics, such as graph size, average clustering coefficient, and graph type.
- Algorithms are often associated with the privacy parameter ε, achieving optimal utility under different privacy requirements. For example, DP-2K [14] exhibits lower error than DK-1K [14] on one graph when $\varepsilon > 20$; however, the results reverse when $\varepsilon \leq 20$.
- All algorithms in our literature review cover only a subset of graph queries. Additionally, even when evaluating the same query, different algorithms employ different utility metrics. For instance, PrivHRG [18] uses normalized mutual information [30] to measure the utility of community detection, whereas LF-GDPR [26] uses the adjusted random index [31] and adjusted mutual information [32].

In this paper, we aim to address the aforementioned challenges with a comprehensive benchmark, PGB (Private Graph Benchmark). Our contributions are summarized as follows:

Benchmark Design Principles. Based on a comprehensive literature review, we identify four essential elements of existing studies as a 4-tuple (**M**, **G**, **P**, **U**): mechanisms, graph datasets, privacy requirements, and utility metrics. For each element, we discuss the limitations of existing works and propose requirements to ensure comparable results (see more details in Section IV).

Benchmark Instantiation. We introduce the benchmark PGB to evaluate the utility of differentially private graph generation algorithms while adhering to all design principles. Our benchmark is implemented, and the source code is publicly available (details in Section V).

Empirical Study and Findings. We have conducted the largest empirical evaluation of private graph generation algorithms so far. Based on our benchmark, it has at least 43,200 single experiments comprising 6 selected algorithms, 8 graph datasets, 6 privacy budgets, and 15 queries. Our findings suggest that while some generation algorithms are generally strong performers, there is no one-size-fits-all solution (details in Section VI).

II. Related Works

A. Private Graph Generation

Research on differentially private graph generation can be broadly categorized into two main approaches: *statistics-based algorithms* and *deep learning (DL)-based algorithms*. Statistics-based algorithms [14]–[29] investigate the choices of graph representation, perturbation, and generation, as illustrated in Figure 1. DL-based algorithms focus on designing privacy-preserving deep generative models [33]–[38], such as graph neural network (GNN). In this work, we concentrate on benchmarking statistics-based graph generation algorithms. In our evaluation, we consider five state-of-the-art private graph generation algorithms: DP-dK [14], TmF [15], PrivSKG [17], PrivHRG [18], and PrivGraph [19], as well as one baseline approach DGG [24].

DP-dK. DP-dK first condenses the graph into the degree distribution of K-connected components (dk-series). It then adds Laplace noise to the learned parameters and generates synthetic graphs with the perturbed parameters using the dK-series model [39]. For the DP-2K model, noise is calibrated based on smooth sensitivity rather than global sensitivity, resulting in noise of a smaller magnitude. Despite these improvements, the privacy budget required remains unreasonably large (i.e., $\varepsilon \geq 100$).

TmF. It first represents a graph as an adjacency matrix, then adds Laplace noise to each cell. Finally, TmF selects the top-m noisy cells as the edges in the randomized adjacency matrix, where m is the noisy number of edges. However, most of the true edges cannot be retained from the top-m noisy cells, especially when ε is small.

PrivSKG. It uses the stochastic Kronecker graph model to represent a graph and then constructs a private estimator of the true parameters. This private estimator defines a probability distribution over the graph. Finally, PrivSKG generates a synthetic graph by sampling from this distribution. Nevertheless, PrivSKG cannot accurately capture the structural properties of the true graph, as the generation process is determined by a single parameter.

PrivHRG. PrivHRG first leverages a statistical hierarchical random graph (HRG) model [40] to represent a graph, recording connection probabilities between any pair of nodes. It then privately samples a dendrogram via Markov-Chain Monte Carlo (MCMC) [41]. Finally, the synthetic graph is generated based on the noisy connection probabilities. However, partial information of the true graph can be lost during the construction of the HRG model.

PrivGraph. It first generates a coarse node partition using a community detection algorithm and applies the Exponential mechanism to obtain the community partitions privately. Then, PrivGraph computes the degree sequences within communities and the number of edges between communities. Finally, it generates a synthetic graph based on the noisy degree sequences using the CL model [42]. Compared with prior works, PrivGraph preserves more structural information of a graph by exploiting community information.

DGG. Node degree is fundamental information in a graph and has been used for private graph generation [24], [26]. We revise DGG [24] to satisfy Edge CDP as our benchmark baseline. Specifically, DGG first calculates the node degrees and then perturbs these degrees using the Laplace mechanism. Finally, it generates a synthetic graph using the BTER model [43]. However, DGG fails to capture the graph structure beyond node degrees, thereby losing detailed information about the true graph.

B. DP Benchmarks

DP benchmarks on data analysis have recently received much attention from researchers, encompassing both *graph data* and *tabular data*. Ning *et al.* [44] implement and benchmark various graph queries (i.e., degree distribution and subgraph counting) by examining the trade-offs between privacy, accuracy, and performance. These implementations of private graph algorithms have been integrated into DPGraph [45]. DPGraph is a benchmark platform for differentially private graph analysis. This platform helps researchers understand the trade-offs between privacy, accuracy, and performance of existing private graph analysis algorithms, primarily focusing on degree distribution and subgraph counting. These benchmarks motivate us to design a comprehensive benchmark for differentially private synthetic graph generation algorithms.

In addition, there are many benchmarks on differentially private tabular data analysis. DPBench [46] is a principled framework for evaluating differential privacy algorithms, such as 1- and 2-dimensional range queries. DPComp [47] is a publicly accessible web-based system to support the principled evaluation of private data analysis and to encourage the dissemination of related code and data. Tao *et al.* [48] propose a systematic benchmark on differentially private synthetic tabular data generation algorithms, including GAN-based, marginal-based, and workload-based methods. Basu *et al.* [49] design a benchmark on the utility of central and federated training of BERT-based models using depression and sexual harassment-related Tweets. Schäler *et al.* [50] introduce a comparable benchmark that meets all design requirements. They conduct the largest empirical study on w-event differential privacy mechanisms. Rosenblatt *et al.* [51] propose an evaluation methodology for DP synthesizers based on reproducibility. However, these benchmarks cannot be directly used to evaluate graph data due to the unique characteristics of graphs, such as privacy definitions, representations, utility metrics, and so forth.

III. PRELIMINARY

A. Differential Privacy

Differential privacy (DP) [9], [10] has become a de-facto standard for preserving individual privacy, which can be formalized in Definition 1. Based on different trusted assumptions, differential privacy can be categorized into two types: *central differential privacy* (CDP) and *local differential privacy* (LDP).

Definition 1 (Differential Privacy [9])**.** *Let $\varepsilon > 0$ be the privacy budget. A randomized algorithm $\mathcal{M}$ with domain $\mathcal{X}$ satisfies ε-DP, if for any neighboring databases $D, D' \in \mathcal{X}$ that differ in a single datum and any subset $S \subseteq Range(\mathcal{M})$,*

$$Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(D') \in S]$$

In the context of graphs, which are composed of nodes and edges, differential privacy can be defined in two ways: *edge differential privacy* (Edge DP) and *node differential privacy* (Node DP) [3]. Edge DP ensures that the output of a randomized mechanism does not reveal whether any specific friendship information (i.e., edge) exists in a graph. In contrast, Node DP conceals the existence of a particular user (i.e., node) along with all her adjacent edges. Node DP provides a stronger privacy guarantee because it protects both node and edge information. However, this stronger privacy comes at the cost of introducing more errors compared to Edge DP. Definition 2 and Definition 3 provide the formal definitions of Edge DP and Node DP, respectively.

Definition 2 (Edge DP)**.** *Let $\varepsilon > 0$ be the privacy budget. A randomized algorithm $\mathcal{M}$ with domain $\mathcal{G}$ satisfies ε-Edge DP, if for any two neighboring graphs $G, G' \in \mathcal{G}$ that differ in one edge and any subset $S \subseteq Range(\mathcal{M})$,*

$$Pr[\mathcal{M}(G) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(G') \in S]$$

Definition 3 (Node DP)**.** *Let $\varepsilon > 0$ be the privacy budget. A randomized algorithm $\mathcal{M}$ with domain $\mathcal{G}$ satisfies ε-Node DP, if for any two neighboring graphs $G, G' \in \mathcal{G}$ that differ in one node with all edges incident to it, and any subset $S \subseteq Range(\mathcal{M})$,*

$$Pr[\mathcal{M}(G) \in S] \leq e^{\epsilon} Pr[\mathcal{M}(G') \in S]$$

The Laplace Mechanism [52] satisfies the requirements of differential privacy (DP) by adding random Laplace noise to the aggregated results. The magnitude of the noise is determined by the sensitivity Δf, i.e., global sensitivity. It is defined as the maximum change in the output of the aggregation function f when the input data D is modified. When f is a numeric query, the formal definition is as follows:

Definition 4 (Laplace Mechanism)**.** *Given any function $f : D \rightarrow R^k$, let Δf be the sensitivity of function f. $M(x) = f(x) + (Y_1, ..., Y_k)$ satisfies ε-differential privacy, where Y_i are $i.i.d$ random variables drawn from Lap($\Delta f/\varepsilon$) and ε is the privacy budget.*

While the Laplace mechanism is effective for handling numeric queries, it is not suitable for queries with non-numeric values. Exponential mechanism [53] is applied whether a function's output is numerical or categorical. The formal definition is described as follows:

Definition 5 (Exponential Mechanism)**.** *Given any quality function $q : (D \times O) \rightarrow R$, and a privacy budget ε, the exponential mechanism $M(D)$ outputs $o \in O$ with probability proportional to $\exp(\frac{\varepsilon q(D,o)}{2\Delta q})$, where $\Delta q = \max_{\forall o, D \simeq D'} |q(D, o) - q(D', o)|$ is the sensitivity of the quality function. $M(D)$ satisfies ε-differential privacy under the following equation.*

$$\Pr[M(D) = o] = \frac{\exp(\frac{\varepsilon q(D,o)}{2\Delta q})}{\sum_{o' \in O} \exp(\frac{\varepsilon q(D,o')}{2\Delta q})} \tag{1}$$

Sensitivity [9] captures the amount of necessary noise to ensure differential privacy (DP). Two common sensitivity definitions are global sensitivity [9] and smooth sensitivity [54].

Definition 6 (Global Sensitivity [9])**.** *For a query function f: $D \rightarrow \mathbb{R}$, the global sensitivity is defined by*

$$\triangle_{GS} = \max_{D \sim D'} |f(D) - f(D')|,$$

where D and D' are neighboring databases that differ in a single user's data.

Definition 7 (Smooth Sensitivity [54])**.** *For a query function $f : D \rightarrow \mathbb{R}$, the β-smooth sensitivity at a database D is defined by*

$$S_f^{\beta}(D) = \max_{D' \sim D} \left(\triangle_f(D') \cdot e^{-\beta \cdot d(D,D')} \right),$$

where $\triangle_f(D')$ is the local sensitivity at D' given by $\triangle_f(D') = \max_{D'' \sim D'} |f(D') - f(D'')|$, D and D' are neighboring databases differing in a single user's data, and $d(D, D')$ is the distance between D and D'.

B. Graph Synthesis with DP

We now introduce a common framework for differentially private graph generation, designed to encompass all mechanisms of our literature survey. This framework enables us to compare mechanisms both theoretically and empirically. As shown in Figure 1, the common framework of differentially

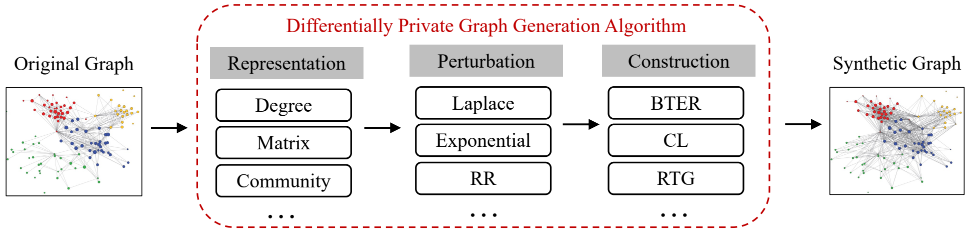

Fig. 1. The common steps for differentially private graph generation algorithms: Representation, Perturbation, and Construction.

private graph synthesis models consists of three main stages: representation, perturbation, and construction.

Representation. The first stage involves modeling the original graph and identifying a compact representation. Various representations, such as degree information [14], [24], [26], adjacency matrix [15]–[17], or community structure [19], [20], [25], [29], are used to capture the essential properties of the graph. It is worth noting that the compact representation effectively addresses the challenge of high-dimensional graph data by reducing the added noise required to guarantee DP.

Perturbation. In this stage, suitable noise is added to the compact representation to satisfy the differential privacy. Common randomized mechanisms include the Laplace mechanism [52], Exponential mechanism [53], Randomized Response (RR) [55], and so forth. According to the post-processing property [9], subsequent processes of graph synthesis do not compromise individual privacy.

Construction. The final stage involves constructing a synthetic graph from the perturbed representations. Some graph constructors such as Block Two-level Erdős-Rényi (BTER) [43] and Chung-Lu (CL) [42] are employed to construct synthetic graphs while preserving the desired structural properties. In fact, graph constructor models have been widely discussed in research communities [56]. A vast majority of graph constructors are created for different requirements. Publications in our literature survey use different constructors to generate graphs. For instance, LDPGen [24] leverages the BTER model and PrivGraph [19] uses the CL model.

Remark 1. *We treat differentially private graph generation algorithms as black boxes, aiming to provide motivation for selecting them in various scenarios. Thus, the choices made at each step (i.e., representation, perturbation, and construction) within the algorithms are beyond the scope of our benchmark.*

IV. Benchmark Design Principles

In this section, we outline the fundamental design principles that underpin our benchmarking framework PGB (see Section V). These principles are critical for ensuring comprehensive, fair, and meaningful comparisons of differentially private graph synthesis algorithms. Prior works often neglect these principles, leading to incomplete or biased evaluations. To develop a robust benchmarking framework, we conducted a thorough literature review as listed in Table I, encompassing key publications from notable conferences or journals such as CCS, VLDB, SIGMOD, and TKDE. We identified essential elements of empirical studies as a 4-tuple (**M**, **G**, **P**, **U**):

- **M**: A set of mechanisms being compared.
- **G**: A set of graph datasets.
- **P**: A set of privacy requirements.
- **U**: A set of graph queries to quantify the utility.

Next, we delve into these elements in detail and discuss the requirements necessary to ensure the comprehensiveness of designing a benchmark.

A. Mechanisms $\boldsymbol{M}$

We consider 4 principles ($\mathbf{M}_1 \sim \mathbf{M}_4$) that the mechanism **M** should satisfy to ensure fair comparisons. Additionally, we discuss the extent to which existing works adhere to these principles, as summarized in Table I.

1) Privacy Definition ($\boldsymbol{M}_1$): A fair comparison of algorithms needs identical privacy definitions. When DP is applied to graph analysis, we have two kinds of privacy definitions since a graph consists of nodes and edges:: edge differential privacy and node differential privacy [3]. The former guarantees that a randomized mechanism does not disclose the addition or deletion of a specific edge belonging to an individual [57], while the latter obscures the addition or deletion of a node and all its connected edges [58]. Besides, there are two another privacy definitions based on different trust assumptions (users trust or untrust the server): central differential privacy (CDP) and local differential privacy (LDP). In CDP [59], a trusted server collects all original data from each user to compute and perturb the query results. In LDP, [60], each user randomizes the data to ensure local DP directly. Therefore, we have four privacy definitions to protect individual information in graph analysis: edge CDP, edge LDP, node CDP, and node LDP. Our literature study (refer to Table I) reveals that half of 16 publications satisfy edge CDP; 6 out of 16 publications satisfy edge LDP; only 2 studies satisfy CDP and no publication satisfies node LDP for private graph synthesis. It is worth noting that algorithms cannot be comparable since they use different privacy definitions . For example, node CDP provides a stronger privacy guarantee than edge CDP but at the cost of utility. Similarly, edge CDP provides a higher utility than edge LDP but relies on a trust server.

2) Sensitivity ($\boldsymbol{M}_2$): Our literature review reveals that the majority of publications use *global sensitivity* [52] to determine the magnitude of the added noise. In contrast, only three publications (DP-dK [14], PrivSKG [17], TriCycLe [21])

TABLE I

COMPARISONS OF PREVIOUS WORKS.

Algorithm	Mechanism (**M**)				Graph (**G**)				Privacy (**P**)	Utility (**U**)	
	P.D.	Δ	Attr.	Code	$\lvert V\rvert$ (10^x)	$\lvert E\rvert$ (10^x)	ACC	Type	ε	Query	Metric
DP-dK [14]	E.C.	S	✗	✗	$2\sim3$	$2\sim4$	$0.25\sim0.63$	$T_{1,3,6}$	[0.2,2000]	$Q_{1\sim4,7,8,11,13,15}$	E_1
TmF [15]	E.C.	G	✗	✗	$2\sim6$	$2\sim6$	$0.25\sim0.63$	$T_{1\sim3,5}$	(0, 50)	$Q_{4\sim10}$	E_1
DER [16]	E.C.	G	✗	✗	3	$3\sim5$	$0.14\sim0.61$	$T_{1,3,4}$	(0.6,1)	$Q_{1,6,8}$	$E_{2,3}$
PrivSKG [17]	E.C.	S	✗	✗	$3\sim4$	$4\sim5$	$0.25\sim0.61$	$T_{2,3,7}$	0.2	$Q_{6,11,15}$	-
PrivHRG [18]	E.C.	G	✗	✓	$3\sim5$	$4\sim5$	$0.14\sim0.63$	$T_{1\sim3}$	1	$Q_{6,9,15}$	E_7
PrivGraph [19]	E.C.	G	✗	✓	$3\sim5$	$4\sim5$	$0.13\sim0.61$	$T_{1\sim3}$	[0.5,3.5]	$Q_{6,7,10,12,15}$	$E_{1,3,7,11}$
C-AGM [20]	E.C.	G	✓	✗	$3\sim4$	$4\sim5$	$0.13\sim0.54$	$T_{1\sim3}$	[2, 9]	$Q_{2,3,6,10}$	$E_{1,4,6}$
TriCycLe [21]	E.C.	S	✓	✗	$3\sim5$	$4\sim6$	$0.10\sim0.18$	$T_{1\sim3}$	[0.01,ln3]	$Q_{2,3,6,10,11}$	$E_{2,5}$
PrivCom [22]	N.C.	G	✗	✗	3	4	0.52	$T_{1\sim3}$	[0.1, 20]	Q_{12}	E_6
π_v, π_e [23]	N.C.	G	✗	✗	$3\sim6$	$4\sim7$	$0.11\sim0.61$	$T_{1,3,7}$	[0.1, 20]	$Q_{1\sim3,6,10,11}$	$E_{2,4}$
LDPGen [24]	E.L.	G	✗	✗	$3\sim5$	$4\sim5$	$0.49\sim0.61$	T_1	(0, 7]	$Q_{10,12\sim14}$	$E_{1,9,10}$
CGGen [25]	E.L.	G	✗	✗	$3\sim4$	$4\sim5$	$0.49\sim0.61$	T_1	(0, 7]	$Q_{10,13,14}$	$E_{1,9,10}$
LF-GDPR [26]	E.L.	G	✗	✓	$3\sim5$	$4\sim7$	$0.49\sim0.63$	$T_{1,3}$	[1, 8]	$Q_{10,12,13}$	$E_{1,8,9,10}$
AsgLDP [27]	E.L.	G	✓	✗	$3\sim5$	$4\sim7$	$0.49\sim0.61$	T_1	[0.1,9]	$Q_{6,10,13}$	$E_{1,4}$
Block-HRG [28]	E.L.	G	✗	✗	$3\sim4$	$4\sim5$	$0.49\sim0.63$	$T_{1,3,6}$	[1, 8]	$Q_{4,6,10,11,13}$	$E_{1,9,10}$
DP-LUSN [29]	E.L.	G	✗	✗	$2\sim3$	3	-	$T_{2,3}$	[0.1, 1]	$Q_{2,10}$	-

P.D.: Privacy Definition E.C.: Edge CDP N.C.: Node CDP E.L.: Edge LDP Δ: Sensitivity G: Global S: Smooth
$|V|$: Number of Nodes $|E|$: Number of Edges ACC: Average Clustering Coefficient ε: Privacy Budget ✓: yes ✗: no
Table II, Table III, and Table IV provide details for Type, Query, and Metric, respectively.

utilize *smooth sensitivity* [54]. Global sensitivity considers *any* two neighboring graphs, which can be pessimistic since it covers the largest difference of all cases. Alternatively, *local sensitivity* [54] fixes one graph and considers all of its neighbors. However, local sensitivity can potentially leak sensitive information about the fixed graph. To address this, smooth sensitivity employs a "smooth approximation" of local sensitivity to calibrate the noise, thereby satisfying differential privacy (DP). It is acceptable for different algorithms to use various sensitivity definitions to measure the added noise. However, the premise is that they should provide identical privacy definitions to ensure the compatibility of the benchmark.

3) Consideration of Attributed Graph ($\boldsymbol{M}_3$): There are rich but sensitive node attributes and edge attributes in real-world graphs. For example, in a disease transmission analysis, we need to collect reports on each person's health condition (i.e., age, gender, and trajectory) and details of the disease transmission (i.e., transmission time, transmission method, and infection probability). Our literature review indicates that most studies have focused on purely structured graphs, only a few algorithms [20], [21], [27] consider graphs with node attributes, and no studies focus on graphs with edge attributes. Directly comparing algorithms for attributed and non-attributed graphs may be unfair, as a portion of the privacy budget must be allocated to protect attributes. One solution is to transform an attributed graph synthesis algorithm into a non-attributed one, allowing the entire privacy budget to be used for protecting structural information.

4) Availability of Source Code ($\boldsymbol{M}_4$): Our survey reveals that only 3 out of 16 publications provide access to their source code. Most algorithms in literature study are intrinsically complex. For example, among the algorithms [19], [20], [25], [29] rely on *community detection* [11]–[13], we find that minor differences in the implementation or parameters (e.g., allocating the privacy budget in each iteration) can have a significant impact on the overall utility. Additionally, some open-sourced algorithms are implemented using different programming languages, such as *Java* [26], *Python* [19], or *C++* [18], which makes it challenging to compare them fairly (i.e., efficiency issue). Therefore, we encourage the public availability of implementations to provide additional insights and facilitate comparisons.

Remark 2. *Most publications do not provide open-source codes, posing a significant challenge for the research community. Although a few algorithms [18], [19] have available source codes, the lack of accessible codes for their competitors complicates the replication of experiments.*

B. Graph Datasets $\boldsymbol{G}$

Ideally, graph datasets used in the empirical analysis should consider the following key attributes ($\mathbf{G}_1\sim\mathbf{G}_4$): graph size (i.e., number of nodes or edges), average clustering coefficient (ACC), and graph types.

1) Graph Size ($\boldsymbol{G}_1$-$\boldsymbol{G}_2$): Our literature survey reveals that graph datasets used in different algorithms vary significantly in size, such as the number of nodes ($|V|$) and the number of edges ($|E|$). The size of graphs plays a crucial role in their utility and efficiency. On the one hand, graph size determines the *density*, which is an important metric for measuring the sparsity of graphs, represented as $\frac{2|E|}{|V|^2}$. Real-world graphs are usually sparse (low density), meaning that $|E|$ is much smaller than the maximum possible number of edges, i.e., $|E| \ll \frac{|V|(|V|-1)}{2}$. However, some perturbation mechanisms, such as randomized response, add significant noise to a graph, resulting in a much denser synthetic graph and undermining the utility [24], [26]. Theoretically, the sparser the graph, the more significant the density problem becomes. On the

TABLE II
DETAILS OF GRAPH TYPES IN DIFFERENT ALGORITHMS.

Alg. \ Type	Social (T_1)	Web (T_2)	Academic (T_3)	Traffic (T_4)	Financial (T_5)	Technology (T_6)	Synthetic (T_7)
DP-dK [14]	✓		✓			✓	
TmF [15]	✓	✓	✓		✓		
DER [16]	✓		✓	✓			
PrivSKG [17]		✓	✓				✓
PrivHRG [18]	✓	✓	✓				
PrivGraph [19]	✓	✓	✓				
C-AGM [20]	✓	✓	✓				
TriCycLe [21]	✓	✓	✓				
PrivCom [22]	✓	✓	✓				
π_v, π_e [23]	✓		✓				✓
LDPGen [24]	✓						
CGGen [25]	✓						
LF-GDPR [26]	✓		✓				
AsgLDP [27]	✓						
Block-HRG [28]	✓		✓			✓	
DP-LUSN [29]		✓	✓				

Social (V: people, E: relationships) Web (V: webpages, E: hyperlinks) Academic (V: researchers, E: collaborations)
Traffic (V: intersections, E: roads) Financial (V: products, E: links) Technology (V: apps, E: relationships)

other hand, processing time also increases with graph size. Therefore, to ensure the comprehensiveness of a benchmark, various graph datasets with different sizes should be evaluated.

2) Average Clustering Coefficient ($\boldsymbol{G}_3$): The clustering coefficient [61] is a fundamental metric in graph theory, quantifying the extent to which nodes in a graph cluster together. This metric offers valuable insights into the local connectivity of the graph by indicating the probability that two neighbors of a given node are also neighbors of each other. The clustering coefficient can be calculated: $C_i = e_i / \binom{d_i}{2}$, where e_i is the number of edges in the subgraph of G induced by a node v_i's neighbors, and d_i is node degree of v_i.

The average clustering coefficient (ACC) [62] measures the overall clustering within a network by averaging the clustering coefficients of all nodes. A network with a high ACC and a small average path length is often referred to as a "small-world" network. The formal definition can be represented by:

$$\overline{C} = \frac{1}{n}\sum_{i=1}^{n} C_i = \frac{2}{n}\sum_{i=1}^{n}\frac{e_i}{d_i(d_i-1)}, \tag{2}$$

where n is the number of nodes in a graph.

Our literature survey indicates that graph datasets used in algorithms exhibit significant variation in ACC. Intuitively, some synthetic graph algorithms [19], [20], [24], [25], [29] that leverage community or clustering information perform exceptionally well on graphs with high ACC. Therefore, a fair and comparable benchmark should include graphs with a range of ACCs.

3) Graph Type ($\boldsymbol{G}_4$): As presented in Table II, multiple graphs from various domains are used to verify the performance of algorithms. Our literature review reveals that three graph types (social, web, and academic) are commonly used in most algorithms, while another three types (traffic, financial, and technology) are evaluated less frequently. Graphs of different types possess distinct characteristics (e.g., node size, edge size, graph density, average clustering coefficient, number of triangles, etc.) that can influence the performance of proposed synthetic methods. For example, the social graphs often exhibit strong community structures, which is suitable for some community-based graph synthetic algorithms [19], [20], [25], [29]. Therefore, it's important to consider a variety of graphs in experimental evaluations to have a fair assessment to algorithms' performance.

Additionally, the synthetic graph (T_7) can simulate special characteristics that real-world graphs may not possess, such as binomial or uniform distributions. Although only two algorithms [17], [23] evaluate synthetic graphs, as shown in Table II, we advocate for the inclusion of synthetic graphs in experiments to ensure the comprehensiveness of a benchmark.

C. Privacy Requirements $\boldsymbol{P}$

In differentially private graph synthetic algorithms, data owners express their privacy requirements by controlling the privacy budget ε. In Table I, the range of privacy budgets in various publications differs significantly, ranging from 0.01 to 2000. In fact, using an excessively large ε (e.g., 2000) could be meaningless for protecting information. To facilitate the comparability of a benchmark, the privacy budget should be set reasonably and identically. Additionally, some generation algorithms, such as DP-dK [14], PrivSKG [17], PrivCom [22], provide (ε, δ)-DP that is a relaxation of ε-DP. It introduces an additional parameter δ to account for the allowable probability that the privacy guarantee may be violated. In general, a randomized algorithm is considered safe when δ is preferably smaller than $1/n$ [63], [64], where n is the number of users.

D. Utility $\boldsymbol{U}$

We consider two principles in the evaluation of generation algorithms: graph query ($\mathbf{U}_1$) and error metric ($\mathbf{U}_2$).

TABLE III
GRAPH QUERIES.

Alg. \ Query	Counting			Degree			Path			Topology					Centrality
	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}
	|V|	|E|	$\triangle$	d	d_σ	$\boldsymbol{d}$	l_{max}	$\bar{l}$	$\boldsymbol{l}$	GCC	ACC	CD	Mod	Ass	EVC
DP-dK [14]	✓	✓	✓	✓			✓	✓			✓		✓		✓
TmF [15]				✓	✓	✓	✓	✓	✓	✓					
DER [16]	✓					✓		✓							
PrivSKG [17]						✓					✓				✓
PrivHRG [18]						✓			✓						✓
PrivGraph [19]						✓	✓			✓		✓			✓
C-AGM [20]		✓	✓			✓				✓					
TriCycLe [21]		✓	✓			✓				✓	✓				
PrivCom [22]												✓			
π_v, π_e [23]	✓	✓	✓			✓				✓	✓				
LDPGen [24]										✓		✓	✓	✓	
CGGen [25]										✓			✓	✓	
LF-GDPR [26]										✓		✓	✓		
AsgLDP [27]						✓				✓			✓		
Block-HRG [28]				✓		✓				✓	✓		✓		
DP-LUSN [29]		✓								✓					

Table IV provides details for each query symbol.

TABLE IV
DETAILS OF GRAPH QUERIES AND METRICS.

Query	Description	Metrics
|V|	Number of nodes	RE, MRE
|E|	Number of edges	RE, MRE
$\triangle$	Triangle counts	RE, MRE
$\bar{d}$	Average degree	RE
d_σ	Degree variance	RE
$\boldsymbol{d}$	Degree distribution	KL, HD, KS
l_{max}	Diameter	RE
$\bar{l}$	Average of all shortest paths	RE
$\boldsymbol{l}$	Distance distribution	RE
GCC	Global clustering coefficient	RE, MRE
ACC	Average clustering coefficient	RE, MRE, MSE
CD	Community detection	NMI, Avg-F_1, ARI, AMI
Mod	Modularity	RE
Ass	Assortativity coefficient	RE
EVC	Eigenvector centrality	MAE

RE (E_1): relative error MRE (E_2): mean relative error;
KL (E_3): KL-divergence HD (E_4): Hellinger distance
KS (E_5): Kolmogorov-Smirnov statistic
Avg-F_1 (E_6): average F_1 score
MAE (E_7): mean absolute error MSE (E_8): mean square error
ARI (E_9): adjusted random index
AMI (E_{10}): adjusted mutual information
NMI (E_{11}): normalized mutual information

1) Graph Query ($\boldsymbol{U}_1$): Multiple graph queries are employed to evaluate the performance of the proposed synthetic algorithms. As shown in Table III, we classify 15 graph queries into five categories: general counting, degree information, path condition, topology structure, and centrality. Table IV provides the detailed content for each query. Our literature survey indicates that all existing publications only cover a subset of these queries. In fact, some works evaluate only one of the five query types. For instance, LDPGen [24], LF-GDPR [26], and CGGen [25] focus solely on topology structure. It is important to use a comprehensive set of graph queries to ensure a fair comparison of all algorithms.

2) Error Metric ($\boldsymbol{U}_2$): For each graph query, researchers compare the error metric between the true and the noisy graph. As illustrated in Table IV, relative error (RE) is used to evaluate 12 out of 15 graph queries. Given a query result of the true graph $Q(G)$ and a query result of the noisy graph $Q(G')$, RE can be computed as $\frac{|Q(G)-Q(G')|}{Q(G)}$. Five graph queries (i.e., $|V|$, $|E|$, $\triangle$, GCC, and ACC) use the mean relative error (MRE) to calculate the utility loss, which can be represented as $\frac{1}{n}\sum_{i=1}^{n}|Q(G_i)-Q(G_i')|$, where $Q(G_i)$ (or $Q(G_i')$) is the result on the node v_i. Additionally, some queries use special metrics to measure output results. For instance, degree distribution is evaluated with Kullback-Leibler divergence (KL) [65], Hellinger distance (HD) [66], or Kolmogorov-Smirnov statistic (KS) [67]. In community detection, the similarity of communities between the true and the synthetic graph can measured by normalized mutual information (NMI) [30], average F_1 score [22], [68], adjusted random index (ARI) [31], and adjusted mutual information (AMI) [32]. Consistent use of error metrics in the benchmark is crucial for ensuring a fair comparison of all algorithms.

V. BENCHMARK INSTANTIATION

In this section, we describe PGB, a benchmark designed to evaluate the utility of differentially private synthetic graph algorithms. The goal of PGB is to establish a set of elements for empirical evaluation that satisfies the design principles outlined in Section IV. Table V provides an overview of the PGB benchmark. Next, We discuss how each element meets the required criteria and how to maintain validity and comprehensiveness.

TABLE V
PGB BENCHMARK WITH 4-TUPLE (**M**, **G**, **P**, **U**)

Element	Instantiation
M	(1) Model: Edge CDP (2) Unattributed graph (3) Algorithms: DP-dK [14], TmF [15], PrivSKG [17], PrivHRG [18], PrivGraph [19], DGG [24]
G	6 real-world graphs and 1 synthetic graph (Table VI)
P	$\varepsilon \in [0.1,10]$
U	15 graph queries listed in Table IV

TABLE VI
DETAILS OF GRAPH DATASETS.

Graph	|V|	|E|	ACC	Type
Minnesota[3]	2,600	3,300	0.0160	Traffic
Facebook[4]	4,039	88,234	0.6055	Social
Wiki-Vote[5]	7,115	103,689	0.1409	Web
ca-HepPh[6]	12,008	118,521	0.6115	Academic
poli-large[7]	15,600	17,500	0.3967	Financial
Gnutella[8]	22,687	54,705	0.0053	Technology
ER graph	10,000	250,278	0.0050	Synthetic
BA graph	10,000	49,975	0.0074	Synthetic

A. Mechanisms *M*

In this subsection, we discuss how to select algorithms in our benchmark to satisfy all design principles mentioned in Section IV.

1) Mechanisms (M_1, M_2, and M_3): As we discussed in SectionIV, algorithms with different elements (i.e., privacy definition, sensitivity, (un)attributed) cannot be compared in a benchmark. Instead, the graph synthesis algorithms included in the benchmark must adhere to the same privacy definition. Consistency in whether attributed information is protected should also be maintained. Besides, according to Table I, the edge CDP definition is employed in 8 out of 16 publications. Among them, 75% of the algorithms target unattributed graph synthesis. Therefore, following the majority of publications, we evaluate unattributed graph generation algorithms under edge CDP in PGB. This can apply to DP-dK [14], TmF [15], DER [16], PrivSKG [17], PrivHRG [18], and PrivGraph [19].

Remark 3. *Our benchmark is not limited to edge CDP and unattributed graphs. When the criteria are unified, any graph synthesis algorithms, such as those using edge LDP and attributed graphs, can be compared using this benchmark.*

2) Algorithm Implementation (M_4): The correct implementation of algorithms is crucial to ensure the fairness and validity of empirical analysis. We implement algorithms in the benchmark based on the following principles: (a) *Original source code*. Unfortunately, this only holds for PrivHRG [18] and PrivGraph [19] (cf. Table I). What's more, these two algorithms are implemented in different programming languages, namely, PrivHRG[1] in C++ and PrivGraph[2] in Python. (b) *Reuse of components in SOTA algorithm*. Multiple algorithms utilize the same components, such as graph queries, which can be applied across different algorithms. For instance, PrivGraph evaluates its performance using various graph queries (e.g., community detection, degree distribution, path condition), which are available as open-source tools. In such cases, we consistently apply these components across all algorithms. (c) *Correctness guarantee*. We check the results of re-implemented algorithms to ensure that they align with the results reported in publications. (d) *Same programming language and running environment*. To guarantee the fairness and validity of comparisons, we implement all algorithms in Python and evaluate them in the same running environment.

As a result, we select six algorithms in our benchmark: DP-dK [14], TmF [15], PrivSKG [17], PrivHRG [18], PrivGraph [19], and DGG [24]. Among them, we use implementations from the authors for PrivGraph and re-implement other algorithms in Python. It should be noted that we include one naive baseline DGG [24] mainly because it generates graphs based on node degrees, which are fundamental but significant pieces of features in differentially private graph algorithms [24], [26]. Since DGG is developed with LDP, we re-implement DGG with the central setting as our benchmark baselines. All experiments are conducted on Linux machines running Ubuntu 20.04.5 LTS with 16 AMD EPYC 7313P@3.7Ghz with 512GB of RAM.

B. Graph Datasets *G*

To meet the design principles outlined in Section IV, we conducted a series of benchmark experiments on a comprehensive set of graphs. Table VI provides an overview of the graph datasets, summarizing four key properties: the number of nodes (|V|), the number of edges (|E|), the average clustering coefficient (ACC), and the graph types. The node sizes range from 2,600 to 22,687, and the edge sizes range from 3,300 to 250,278. These graphs are sourced from seven different domains, with each graph type utilized at least once to evaluate a generation algorithm (cf. Table II). Among them, 6 out of 8 graphs are derived from public datasets (i.e., SNAP [69], NR [70]), while two are synthesized using generative models, specifically the Erdos-Renyi (ER) model [71] and the Barabasi-Albert (BA) model [72]. Node degrees in ER graphs follow a binomial distribution [73], whereas node degrees in BA graphs follow a power-law distribution [74]. In our experiments, both the ER and BA graphs were generated with |V| = 10,000.

C. Privacy Requirements *P*

Following the example of most experimental analyses in publications, we also conduct experiments with varying ε

[1]https://github.com/kaseyxiao/privHRG
[2]https://github.com/Privacy-Graph/PrivGraph

[3]https://networkrepository.com/road-minnesota.php
[4]http://snap.stanford.edu/data/ego-Facebook.html
[5]http://snap.stanford.edu/data/wiki-Vote.html
[6]http://snap.stanford.edu/data/ca-HepPh.html
[7]https://networkrepository.com/econ-poli-large.php
[8]http://snap.stanford.edu/data/p2p-Gnutella25.html

values. Determining an appropriate ε is an ongoing area of research [75]–[78]. In our experiments, we vary the privacy budget ε from 0.1 to 10, similar to the ranges used in most studies listed in Table I. For algorithms we implement in $\mathbf{M}_4$, DP-dK [14] and PrivSKG [17] maintain (ε, δ)-DP, while the others provide ε-DP. To ensure a fair comparison, we set δ = 0.01 for DP-dK and PrivSKG, following the parameters used in this work [14], [17].

D. Utility *U*

To ensure the comparability of our benchmark, we apply all the queries listed in Table III to evaluate the performance of the algorithms. These queries represent the union of those used in 16 different publications. Due to the inherent randomness of the algorithms, the utility can differ significantly under the same combination of privacy budget and graph dataset. Similar to various studies in related work, we run each experiment 10 times and calculate the average of the utility metrics. We use different metrics for various graph queries. First, we use Relative Error (RE) for most queries, including |V|, |E|, $\triangle$, $\overline{d}$, d_σ, l_{max}, $\overline{l}$, GCC, ACC, Mod, and Ass. Second, we use Kullback-Leibler divergence (KL), Normalized Mutual Information (NMI), and Mean Absolute Error (MAE) to evaluate the utility error of $\boldsymbol{d}$, CD, and EVC, respectively. Third, we use KL for $\boldsymbol{l}$ instead of RE, as KL can better measure how one probability distribution differs from another compared to RE. The details of graph queries and metrics are explained in Table IV.

VI. Experimental Results

We formulate the following research questions:

- **Q1**: How do algorithms compare in terms of the overall utility across various graphs and privacy budgets?
- **Q2**: How do graph datasets, privacy budgets, and utility metrics affect the utility of different algorithms?
- **Q3**: What are the time and space costs of the algorithms?

A. Overall Uility Analysis

We first present the comprehensive results of our benchmark study on differentially private graph generation algorithms. Table VII summarizes the performance of six state-of-the-art algorithms across various graph datasets under different privacy budgets (ε). Each entry in the table indicates the number of times an algorithm achieved the best performance out of 15 queries for a given dataset and privacy budget (Definition 8). The highest frequency in each case is highlighted in gray. We can conclude some key findings from the overall results.

Definition 8. *Let $\mathcal{A}$ be target algorithm. Let G and ε be the graph dataset and privacy budget, respectively. Let $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_p\}$ be a set of p queries. Let B_i be the best performance indicator:*

$$B_i = \begin{cases} 1 & \textit{if } A \textit{ performs best on } Q_i \textit{ for } G \textit{ and } \varepsilon \\ 0 & \textit{otherwise} \end{cases}$$

Finally, we have:

$$C_A(G, \varepsilon) = \sum_{i=1}^{p} B_i,$$

where $C_A(G, \varepsilon)$ is the count of how often algorithm A performs best across the p queries for G and ε.

Algorithm Performance Consistency: TmF emerges as the most consistent and dominant performer across multiple datasets and privacy budgets. It achieves the highest number of top performances in 40 instances, showcasing its robustness and adaptability. Particularly, TmF excels in the ER and BA datasets, consistently outperforming other algorithms across all tested privacy budgets ($\varepsilon = 0.1$, 0.5, 1, 2, 5, and 10). This indicates TmF's strong capability in generating high-quality differentially private graphs. DGG also shows significant performance, especially in the Facebook and Wiki datasets. It frequently achieves the highest scores across different privacy budgets, indicating its effectiveness in these specific types of graphs. For instance, DGG performs best in the Facebook dataset at lower privacy budgets, maintaining strong performance even as the privacy budget increases.

Graph Dataset Sensitivity: The Minnesota dataset reveals distinct performance patterns where DP-dK performs exceptionally well at lower privacy budgets (ε = 0.1 and 0.5). However, its performance diminishes at higher privacy budgets. On the other hand, TmF and PrivSKG gain prominence as the privacy budget increases, suggesting their scalability and efficiency in handling more relaxed privacy constraints. In the Facebook dataset, DGG and DP-dK demonstrate strong performance. DGG consistently outperforms others at lower budgets and maintains a significant presence across all budgets, highlighting its robustness. DP-dK also performs well, particularly at higher privacy budgets.

Impact of Privacy Budget: As the privacy budget ε increases, TmF and DGG generally improve in performance, indicating their robustness and scalability. For instance, TmF achieves top performance in 8 instances at ε = 10, the highest count in the entire table. This suggests that these algorithms are capable of generating high-quality graphs even with more relaxed privacy constraints. At lower privacy budgets (ε = 0.1), there is a more diverse performance spread among algorithms. No single algorithm consistently dominates across all datasets, indicating the complexity and challenge of achieving high performance with stringent privacy guarantees.

Algorithm Specialization: PrivSKG exhibits a mixed performance. While it excels in certain instances (e.g., Facebook at $\varepsilon = 10$), it often underperforms compared to other algorithms. This suggests that PrivSKG may be more effective in specific contexts or datasets, particularly at higher privacy budgets. PrivHRG has the least number of top performances, indicating its limitations in achieving competitive results across a wide range of queries and datasets. This highlights the need for further improvements or specialized applications for PrivHRG to enhance its utility.

Overall Best Performers: TmF stands out as the most reli-

TABLE VII
OVERALL RESULTS.

ε	Algorithms	Graph Datasets							
		Minnesota	Facebook	Wiki	HepPh	Poli	Gnutella	ER	BA
0.1	DP-dK	5	4	3	3	4	2	0	0
	TmF	6	4	3	3	5	4	14	6
	PrivSKG	1	1	3	2	2	3	2	2
	PrivHRG	2	0	1	0	2	4	2	3
	PrivGraph	1	1	1	2	2	1	1	3
	DGG	2	7	6	7	2	3	1	3
0.5	DP-dK	5	5	1	4	2	2	0	1
	TmF	5	4	3	3	4	5	13	4
	PrivSKG	2	0	3	2	3	4	2	6
	PrivHRG	2	0	2	0	3	3	3	3
	PrivGraph	2	2	1	1	1	1	1	1
	DGG	1	7	7	7	4	2	1	2
1	DP-dK	5	5	2	3	2	2	0	1
	TmF	4	4	3	3	4	5	12	4
	PrivSKG	3	0	4	2	2	5	2	4
	PrivHRG	1	0	2	0	3	1	4	1
	PrivGraph	3	2	2	4	2	2	1	5
	DGG	1	6	4	5	4	2	1	2
2	DP-dK	4	6	2	5	2	3	0	1
	TmF	3	4	3	3	4	4	13	4
	PrivSKG	4	0	2	2	3	4	2	8
	PrivHRG	0	0	2	0	2	1	3	1
	PrivGraph	4	2	4	3	2	2	1	1
	DGG	2	5	4	4	4	3	1	2
5	DP-dK	4	5	2	6	2	3	1	1
	TmF	4	5	4	4	4	4	11	3
	PrivSKG	5	0	1	1	3	4	2	7
	PrivHRG	0	0	1	0	3	1	4	2
	PrivGraph	2	2	6	2	2	2	1	2
	DGG	2	6	3	4	3	3	1	2
10	DP-dK	4	5	1	4	2	3	1	1
	TmF	8	5	11	9	4	8	13	8
	PrivSKG	0	0	0	1	3	3	2	3
	PrivHRG	0	0	2	0	3	0	2	2
	PrivGraph	2	3	1	1	2	1	1	1
	DGG	3	4	2	2	3	2	1	2

[1] Each number shows how often the algorithm performs best across 15 queries, given a privacy budget ε and a graph dataset. For example, the first number '5' means that DP-dK outperforms others in 5 queries (i.e., Q_5, Q_6, Q_9, Q_{12}, Q_{13}) for the Minnesota graph with $\varepsilon = 0.1$.
[2] The highest frequency in each case is highlighted in gray.

able and versatile algorithm across different privacy budgets and datasets. Its consistent top performance underscores its adaptability and effectiveness in differentially private graph generation. TmF's ability to handle various data characteristics and privacy requirements makes it a suitable choice for diverse applications. DGG emerges as a strong contender, particularly excelling in specific datasets like Facebook and Wiki. Its frequent top performance in these datasets suggests it may be more tailored to certain types of graphs, making it a valuable tool for targeted graph generation.

B. Utility in Specific Cases

To further illustrate the utility of algorithms, we examine specific cases from the benchmark results shown in Fig. 2. Due to the limited space, we list the results of five queries on four graphs. This analysis highlights the strengths and limitations of the algorithms in generating utility-preserving graphs under varying privacy requirements.

Triangle Counting. For Facebook and CA-HepPh, DP-dK exhibits significant fluctuations and higher relative error at lower privacy budgets, stabilizing only at $\varepsilon = 10$. In contrast, the others maintains consistently relative error across all privacy budgets. For the ER Graph, TmF owns very low relative error across all privacy budgets, while DP-dK and DGG have higher errors, suggesting limitations in this specific context.

Degree Distribution. DP-dK consistently outperforms other methods across most of graphs, achieving the lowest KL divergence at higher ε values. Other methods like PrivGraph and Tmf show varied performance, generally improving as ε increases but not to the extent of DP-dK.

Diameter. In general, DGG maintains a low and consistent RE across most of graphs and privacy budgets. For all graphs, DP-dK have the highest relative errors than others in diameter. For the ER Graph, TmF, PrivSKG, and PrivHRG own the lowest relative error, which is equal to 0 approximately.

Community Detection. In most of graphs, DP-dK and

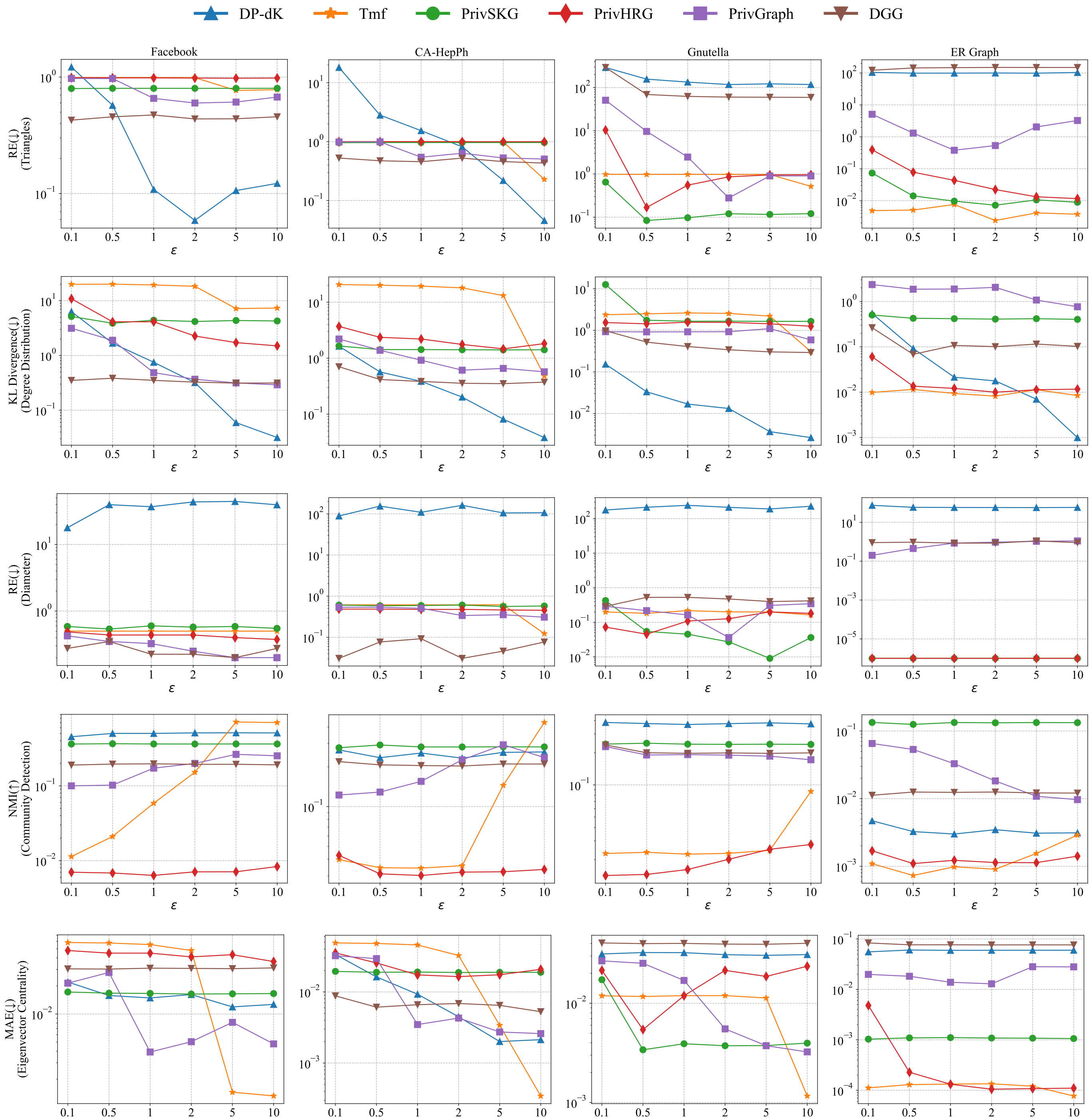

Fig. 2. End-to-end comparison of algorithms under different graph datasets, privacy budget ε and qureies.

PrivSKG achieve highest NMI values than others, which means that they can preserve the community structure very well. In contrast, PrivHRG performs the worst for all graphs and privacy budgets. The performance of TmF can be improved as the privacy budget increases, i.e., when $\varepsilon = 10$. Tmf and PrivSKG maintain moderate MAE values but show improvement with higher ε levels.

Eigenvector Centrality. DP-dK demonstrates a steep decline in MAE with increasing ε, achieving the lowest errors across all datasets when $\varepsilon = 10$. Both PrivGraph and PrivSKG maintain moderate MAE values, but show improvement with higher ε values. For the ER graph, the influence of privacy budgets on most algorithms, excluding PrivHRG, is minimal.

In summary, TmF consistently achieves high utility across various datasets and privacy budgets, making it a reliable choice for differentially private graph generation. DGG demonstrates particular strength in preserving utility in specific datasets such as Facebook and Wiki, suggesting its suitability

TABLE VIII
COMPARISON OF TIME AND SPACE COMPLEXITY.

Algorithms	Time Complexity	Space Complexity
DP-dK	$O(n^2)$	$O(n^2)$
TmF	$O(n^2)$	$O(n^2)$
PrivSKG	$O(n^2m)$	$O(n^2)$
PrivHRG	$O(n^2 \log n)$	$O(m+n)$
PrivGraph	$O(n^2)$	$O(m+n)$
DGG	$O(n^2)$	$O(n^2)$

n: number of nodes m: number of edges

for certain graph structures. Its performance is comparable to TmF in several cases. PrivGraph excels in multiple metrics, particularly in preserving community structures and eigenvector centrality. It shows competitive performance across various datasets and privacy budgets, making it a versatile choice for graphs with evident community structures. DP-dK exhibits higher error rates and lower NMI scores in several cases, especially at lower privacy budgets, indicating potential limitations in utility preservation under strict privacy constraints. PrivHRG and PrivSKG show mixed performance, with higher error rates in several metrics, highlighting areas where further optimization and research could enhance their utility preservation capabilities.

C. Time and Space Analysis

In this part, we compare the performance of algorithms theoretically and empirically, including time and space cost.

Theoretical Analysis. Table VIII summarizes the theoretical results of time complexity and space complexity.

Time Complexity. DP-dK, TmF, PrivGraph, and DGG all have a time complexity of $O(n^2)$, where n is the number of nodes in the graph. This quadratic complexity suggests that these algorithms should handle moderate-sized graphs efficiently but may struggle with extremely large graphs. PrivSKG has a higher time complexity of $O(n^2m)$, indicating potential inefficiency for very large graphs with many edges. PrivHRG has a slightly higher time complexity of $O(n^2 \log n)$, indicating that it may be less efficient for very large graphs due to the additional logarithmic factor.

Space Complexity. DP-dK, TmF, PrivSKG, and DGG have a space complexity of $O(n^2)$, indicating substantial memory requirements for large graphs. PrivGraph and PrivHRG are more space-efficient with a complexity of $O(m+n)$, where m is the number of edges, making them more suitable for sparse graphs.

Remark 4. *We represent graphs as an adjacency matrix in re-implementing algorithms for efficient queries. Thus, the time complexity and space complexity are $O(n^2)$ for most algorithms (e.g., DP-dK, TmF, PrivSKG, and DGG).*

Empirical Analysis. Table IX presents the empirical time cost (in seconds) for running each algorithm on various graph datasets. DP-dK consistently shows the lowest time cost across most datasets, indicating its efficiency in practice. TmF and DGG also demonstrate reasonable time costs, making them practical for larger datasets. PrivSKG has significantly higher time costs, particularly on larger datasets like ca-HepPh and ER graph, suggesting scalability issues. The main reason is that PrivSKG has to spend additional time to compute the smooth sensitivity. PrivGraph shows moderate time costs, balancing efficiency and performance.

TABLE IX
COMPARISON OF TIME COST (SECONDS).

Graphs	Algorithms				
	DP-dK	TmF	PrivSKG	PrivGraph	DGG
Minnesota	0.12	9.28	252.72	0.88	0.11
Facebook	1.36	27.83	9230.63	3.37	0.65
Wiki-Vote	1.97	77.56	21833.8	7.05	1.21
ca-HepPh	9.58	207.97	43452.83	16.97	2.00
poli-large	8.75	317.35	6721.03	21.33	2.22
Gnutella	4.65	688.26	22630.92	46.29	4.24
ER graph	4.27	164.86	46995.37	16.38	1.58
BA graph	8.01	137.83	9230.20	10.54	0.95

TABLE X
COMPARISON OF MEMORY CONSUMPTION (MEGATYPES).

Graphs	Algorithms				
	DP-dK	TmF	PrivSKG	PrivGraph	DGG
Minnesota	108.26	53.28	75.15	22.93	111.00
Facebook	129.27	124.50	117.46	79.85	303.28
Wiki-Vote	156.93	386.29	327.08	184.49	846.83
ca-HepPh	6649.7	1100.20	1200.01	461.97	2291.97
poli-large	8861.51	1850.87	1167.29	711.27	3730.51
Gnutella	7821.59	3927.03	4640.66	1508.71	7913.61
ER graph	1783.50	763.02	1245.09	379.87	1624.07
BA graph	5600.40	763.02	1174.19	308.90	1562.60

Table X provides a comparison of empirical memory consumption (in megabytes) for the algorithms. PrivGraph is the most memory-efficient, particularly on smaller datasets like Minnesota and Facebook. TmF, PrivSKG, and DGG generally require moderate memory, making them suitable for memory-constrained environments. DP-dK consumes more memory than others, especially on larger datasets, indicating potential challenges in memory-limited scenarios.

VII. CONCLUSIONS

We addressed the challenge of comparable empirical studies on differentially private synthetic graph generation algorithms. Through a comprehensive literature study, we identified key elements of existing studies, including mechanisms, graph datasets, privacy requirements, and utility metrics, and formulated design principles to ensure comparability. Based on these principles, we introduced PGB, a benchmark that meets all principles for fair comparison. We conducted the largest empirical study on differentially private synthetic graph algorithms to date, revealing valuable insights into the strengths and weaknesses of existing mechanisms. Our study highlights that while some algorithms perform well under certain conditions, no single solution is universally optimal.

REFERENCES

[1] W.-Y. Day, N. Li, and M. Lyu, "Publishing graph degree distribution with node differential privacy," in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 123–138.

[2] S. Liu, Y. Cao, T. Murakami, and M. Yoshikawa, "A crypto-assisted approach for publishing graph statistics with node local differential privacy," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 5765–5774.

[3] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate estimation of the degree distribution of private networks," in *2009 Ninth IEEE International Conference on Data Mining*. IEEE, 2009, pp. 169–178.

[4] S. Liu, Y. Cao, T. Murakami, J. Liu, and M. Yoshikawa, "Cargo: Crypto-assisted differentially private triangle counting without trusted servers," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024.

[5] J. Imola, T. Murakami, and K. Chaudhuri, "Locally differentially private analysis of graph statistics." in *USENIX Security Symposium*, 2021, pp. 983–1000.

[6] Y. Liu, S. Zhao, Y. Liu, D. Zhao, H. Chen, and C. Li, "Collecting triangle counts with edge relationship local differential privacy," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 2008–2020.

[7] M. E. Newman, "Random graphs with clustering," *Physical review letters*, vol. 103, no. 5, p. 058701, 2009.

[8] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, "Private analysis of graph structure," *Proc. VLDB Endow.*, vol. 4, no. 11, p. 1146–1157, aug 2011. [Online]. Available: https://doi.org/10.14778/3402707.3402749

[9] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[10] N. Li, M. Lyu, D. Su, and W. Yang, "Differential privacy: From theory to practice," *Synthesis Lectures on Information Security, Privacy, & Trust*, vol. 8, no. 4, pp. 1–138, 2016.

[11] H. H. Nguyen, A. Imine, and M. Rusinowitch, "Detecting communities under differential privacy," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, 2016, pp. 83–93.

[12] M. S. Mohamed, D. Nguyen, A. Vullikanti, and R. Tandon, "Differentially private community detection for stochastic block models," in *International Conference on Machine Learning*. PMLR, 2022, pp. 15 858–15 894.

[13] N. Fu, W. Ni, L. Hou, D. Zhang, and R. Zhang, "Community detection in decentralized social networks with local differential privacy," *Information Sciences*, vol. 661, p. 120164, 2024.

[14] Y. Wang and X. Wu, "Preserving differential privacy in degree-correlation based graph generation," *Transactions on data privacy*, vol. 6, no. 2, p. 127, 2013.

[15] H. H. Nguyen, A. Imine, and M. Rusinowitch, "Differentially private publication of social graphs at linear cost," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 2015, pp. 596–599.

[16] R. Chen, B. C. Fung, P. S. Yu, and B. C. Desai, "Correlated network data publication via differential privacy," *The VLDB Journal*, vol. 23, pp. 653–676, 2014.

[17] D. Mir and R. N. Wright, "A differentially private estimator for the stochastic kronecker graph model," in *Proceedings of the 2012 Joint EDBT/ICDT workshops*, 2012, pp. 167–176.

[18] Q. Xiao, R. Chen, and K.-L. Tan, "Differentially private network data release via structural inference," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 911–920.

[19] Q. Yuan, Z. Zhang, L. Du, M. Chen, P. Cheng, and M. Sun, "Privgraph: Differentially private graph data publication by exploiting community information," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 3241–3258.

[20] X. Chen, S. Mauw, and Y. Ramírez-Cruz, "Publishing community-preserving attributed social graphs with a differential privacy guarantee," *Proceedings on Privacy Enhancing Technologies*, 2020.

[21] Z. Jorgensen, T. Yu, and G. Cormode, "Publishing attributed social graphs with formal privacy guarantees," in *Proceedings of the 2016 international conference on management of data*, 2016, pp. 107–122.

[22] S. Zhang, W. Ni, and N. Fu, "Community preserved social graph publishing with node differential privacy," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1400–1405.

[23] X. Jian, Y. Wang, and L. Chen, "Publishing graphs under node differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 4164–4177, 2021.

[24] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 425–438.

[25] X. Ju, X. Zhang, and W. K. Cheung, "Generating synthetic graphs for large sensitive and correlated social networks," in *2019 IEEE 35th international conference on data engineering workshops (ICDEW)*. IEEE, 2019, pp. 286–293.

[26] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Lf-gdpr: A framework for estimating graph metrics with local differential privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 10, pp. 4905–4920, 2020.

[27] C. Wei, S. Ji, C. Liu, W. Chen, and T. Wang, "Asgldp: Collecting and generating decentralized attributed graphs with local differential privacy," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3239–3254, 2020.

[28] L. Hou, W. Ni, S. Zhang, N. Fu, and D. Zhang, "Block-hrg: Block-based differentially private iot networks release," *Ad Hoc Networks*, vol. 140, p. 103059, 2023.

[29] P. Liu, Y. Xu, Q. Jiang, Y. Tang, Y. Guo, L.-e. Wang, and X. Li, "Local differential privacy for social network publishing," *Neurocomputing*, vol. 391, pp. 273–279, 2020.

[30] T. O. Kvalseth, "Entropy and correlation: Some comments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 17, no. 3, pp. 517–519, 1987.

[31] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[32] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1073–1080.

[33] I. E. Olatunji, T. Funke, and M. Khosla, "Releasing graph neural networks with differential privacy guarantees," *arXiv preprint arXiv:2109.08907*, 2021.

[34] X. Miao, W. Zhang, Y. Jiang, F. Fu, Y. Shao, L. Chen, Y. Tao, G. Cao, and B. Cui, "P 2 cg: A privacy preserving collaborative graph neural network training framework," *The VLDB Journal*, vol. 32, no. 4, pp. 717–736, 2023.

[35] Y. Miao, X. Wang, and H. Lin, "A graph generation network with privacy preserving capabilities," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2023, pp. 67–79.

[36] L. Hou, W. Ni, S. Zhang, N. Fu, and D. Zhang, "Wdp-gan: Weighted graph generation with gan under differential privacy," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 5155–5165, 2023.

[37] C. Yang, H. Wang, K. Zhang, L. Chen, and L. Sun, "Secure deep graph generation with link differential privacy," *arXiv preprint arXiv:2005.00455*, 2020.

[38] K. Zhang, L. Sun, B. Ding, S. M. Yiu, and C. Yang, "Deep efficient private neighbor generation for subgraph federated learning," in *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 2024, pp. 806–814.

[39] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "Systematic topology analysis and generation using degree correlations," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 135–146, 2006.

[40] A. Clauset, C. Moore, and M. E. Newman, "Hierarchical structure and the prediction of missing links in networks," *Nature*, vol. 453, no. 7191, pp. 98–101, 2008.

[41] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[42] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000, pp. 171–180.

[43] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of erdős-rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, 2012.

[44] H. Ning, S. Udayashankar, and S. Q. K. K. X. He, “Benchmarking differentially private graph algorithms,” in *Workshop Theory and Practice of Differential Privacy, ICML*. JPC, 2021.
[45] S. Xia, B. Chang, K. Knopf, Y. He, Y. Tao, and X. He, “Dpgraph: A benchmark platform for differentially private graph analysis,” in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2808–2812.
[46] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang, “Principled evaluation of differentially private algorithms using dpbench,” in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 139–154.
[47] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, D. Zhang, and G. Bissias, “Exploring privacy-accuracy tradeoffs using dpcomp,” in *Proceedings of the 2016 International Conference on Management of Data*, 2016, pp. 2101–2104.
[48] Y. Tao, R. McKenna, M. Hay, A. Machanavajjhala, and G. Miklau, “Benchmarking differentially private synthetic data generation algorithms,” *arXiv preprint arXiv:2112.09238*, 2021.
[49] P. Basu, T. S. Roy, R. Naidu, Z. Muftuoglu, S. Singh, and F. Mireshghallah, “Benchmarking differential privacy and federated learning for bert models,” *arXiv preprint arXiv:2106.13973*, 2021.
[50] C. Schäler, T. Hütter, and M. Schäler, “Benchmarking the utility of w-event differential privacy mechanisms-when baselines become mighty competitors,” *Proceedings of the VLDB Endowment*, vol. 16, no. 8, pp. 1830–1842, 2023.
[51] L. Rosenblatt, B. Herman, A. Holovenko, W. Lee, J. Loftus, E. McKinnie, T. Rumezhak, A. Stadnik, B. Howe, and J. Stoyanovich, “Epistemic parity: Reproducibility as an evaluation metric for differential privacy,” *ACM SIGMOD Record*, vol. 53, no. 1, pp. 65–74, 2024.
[52] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 2006, pp. 265–284.
[53] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 2007, pp. 94–103.
[54] K. Nissim, S. Raskhodnikova, and A. Smith, “Smooth sensitivity and sampling in private data analysis,” in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 2007, pp. 75–84.
[55] S. L. Warner, “Randomized response: A survey technique for eliminating evasive answer bias,” *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.
[56] A. Bonifati, I. Holubová, A. Prat-Pérez, and S. Sakr, “Graph generators: State of the art and open challenges,” *ACM computing surveys (CSUR)*, vol. 53, no. 2, pp. 1–30, 2020.
[57] J. Blocki, A. Blum, A. Datta, and O. Sheffet, “The johnson-lindenstrauss transform itself preserves differential privacy,” in *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 2012, pp. 410–419.
[58] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, “Analyzing graphs with node differential privacy,” in *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*. Springer, 2013, pp. 457–476.
[59] C. Dwork, “Differential privacy: A survey of results,” in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.
[60] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, “What can we learn privately?” *SIAM Journal on Computing*, vol. 40, no. 3, pp. 793–826, 2011.
[61] P. W. Holland and S. Leinhardt, “Transitivity in structural models of small groups,” *Comparative group studies*, vol. 2, no. 2, pp. 107–124, 1971.
[62] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’networks,” *nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[63] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*. Springer, 2006, pp. 486–503.
[64] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.
[65] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.
[66] M. S. Nikulin *et al.*, “Hellinger distance,” *Encyclopedia of mathematics*, vol. 78, 2001.
[67] W. W. Daniel, “Kolmogorov–smirnov one-sample test,” *Applied nonparametric statistics*, vol. 2, 1990.
[68] G. Rossetti, L. Pappalardo, D. Pedreschi, and F. Giannotti, “Tiles: an online algorithm for community discovery in dynamic social networks,” *Machine Learning*, vol. 106, pp. 1213–1241, 2017.
[69] J. Leskovec and A. Krevl, “SNAP Datasets: Stanford large network dataset collection,” http://snap.stanford.edu/data, Jun. 2014.
[70] R. A. Rossi and N. K. Ahmed, “The network data repository with interactive graph analytics and visualization,” in *AAAI*, 2015. [Online]. Available: https://networkrepository.com
[71] P. Erd6s and A. Rényi, “On the evolution of random graphs,” *Publ. Math. Inst. Hungar. Acad. Sci*, vol. 5, pp. 17–61, 1960.
[72] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *science*, vol. 286, no. 5439, pp. 509–512, 1999.
[73] M. E. Newman, S. H. Strogatz, and D. J. Watts, “Random graphs with arbitrary degree distributions and their applications,” *Physical review E*, vol. 64, no. 2, p. 026118, 2001.
[74] B. e. Bollobás, O. Riordan, J. Spencer, and G. Tusnády, “The degree sequence of a scale-free random graph process,” *Random Structures & Algorithms*, vol. 18, no. 3, pp. 279–290, 2001.
[75] C. Dwork, N. Kohli, and D. Mulligan, “Differential privacy in practice: Expose your epsilons!” *Journal of Privacy and Confidentiality*, vol. 9, no. 2, 2019.
[76] S. P. Kasiviswanathan and A. Smith, “On the’semantics’ of differential privacy: A bayesian formulation,” *Journal of Privacy and Confidentiality*, vol. 6, no. 1, 2014.
[77] J. Lee and C. Clifton, “How much is enough? choosing ε for differential privacy,” in *Information Security: 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings 14*. Springer, 2011, pp. 325–340.
[78] A. Pankova and P. Laud, “Interpreting epsilon of differential privacy in terms of advantage in guessing or approximating sensitive attributes,” in *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*. IEEE, 2022, pp. 96–111.