Optimizing PrivGraph: The Impact of Community Initialization and Privacy Budget Allocation Strategies on Differential Privacy

Robyn Burger, Gloria Cai, Anika Cave, Alice Sze
CS 6850

December 2023

Abstract

Information networks offer valuable insights through the analysis of their structure and data. However, these networks often contain sensitive information, presenting privacy concerns if analyzed or published directly. Algorithms designed to privatize these datasets must be careful to obscure enough information as to ensure privacy, but not so much as to lose the original properties of the graph. The balance of usefulness and privacy of information is highly dependent on the nature of the dataset. We explore optimizations to *PrivGraph*, an algorithm that offers one such definition of privacy, *differential privacy*, while preserving network metrics such as community finding and average degree. Although PrivGraph is tested over a variety of hyperparameters, provided with an unknown dataset, there are few guidelines for how to divide the privacy budget ε among stages or set community initialization hyperparameter, N. In order to address these ambiguities, we develop and test a linear regression model on the hyperparameter results.

1 Introduction

In an era increasingly defined by the advantages of data-driven analysis, it is essential to recognize the data that empowers us can also be used for harm. Lenient privacy standards can inadvertently leak intimate information, and may create channels for malicious abuse. Take, for instance, artificial intelligence software which aids in early cancer diagnosis[9] or simulations used to mitigate the spread of COVID-19 [17]. Such technologies require sensitive health information in order to achieve their successful results. Personal information such as health records[2], financial history[23], and political affiliation[8] all have both practical and malicious applications. Thus, to mitigate potential for abuse, institutions must strive for robust privacy protection of the underlying dataset before analysis can occur.

Fittingly, in recent years, we have seen an increased urgency for data protection both in the popular imagination and global institutions. U.S. President Joe Biden codified this notion in his October 2023 "Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence"[3]. He urges the use of "secure multiparty computation, homomorphic encryption, zero-knowledge proofs, federated learning, secure enclaves, differential privacy, and synthetic-data-generation tools," when working with large language models and their corresponding datasets.

1.1 Differential Privacy

Our work focuses on the privacy guarantee granted by differential privacy (DP), which reduces the ability to leak private information while still providing beneficial analysis of the dataset [4]. Proposed in 2006 by Cynthia Dwork, DP provides extremely accurate information about the database while ensuring exceptional privacy standards. This privacy guarantee is so robust that it was adopted by the White House as their standard for security in the deployment of large language models [28]. Ideally the result of analysis on a differentially private dataset is identical to that of its non-differentially private counterpart. However, in application, the lower the privacy-budget of a differentially-private dataset is, the less likely it is to accurately retain features of the original dataset. This is especially true for features which are not specifically adjusted by the perturbation function, but are rather consequently and unintentionally perturbed. With an decrease in privacy budget, and thus an increase in privacy, we experience a decrease in the utility of the resulting dataset. Therefore, the privacy budget can impact both the utility and privacy of the resulting dataset. Since many algorithms use hyperparameters to allocate the privacy budget between phases, it is essential to allocate efficiently to maximize the utility while maintaining privacy guarantees.

DP is effective by bounding the impact of a single data-point on the final output. Two datasets D, D' are considered *neighbors* if and only if they differ by

one record, i.e. either $D = D' + r$ or $D' = D + r$ for some record r [30]. We will now formally define DP.

Definition 1. *ε-Differential Privacy.* An algorithm $\mathcal{A}$ satisfies ε-differential privacy if and only if, given two neighboring datasets, D and D', the following holds $\forall T \subseteq \text{Range}(\mathcal{A})$:

$$\Pr[\mathcal{A}(D) \in T] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(D') \in T]$$

for the *privacy budget*, $\varepsilon > 0$, which determines the strength of privacy protection [30].

1.2 Graph Differential Privacy

Like datasets at large, graph representations of information networks can be extremely useful but are not immune to privacy vulnerabilities. The need for secure analysis inspired the creation of a new class of algorithms which provide differential privacy guarantees for information networks. We further specify the terms of differential privacy on graphs by subdividing DP into *ε-node differential privacy* and *ε-edge differential privacy*.

Node-differential privacy provides a bound on the privacy of all information associated with a node which could include, but is not limited to: degree, identification number, zip-code, and exiting edges. Edge-differential privacy only enforces a privacy guarantee on the edges connecting nodes in the graph.

Formally, we define *ε-node differential privacy* and *ε-edge differential privacy* as follows:

Definition 2. *ε-Node Differential Privacy.* Two graphs are *neighbors* if one can be obtained by the other by removing a node and its adjacent edges. Given a *privacy-budget* $\varepsilon > 0$, an algorithm $\mathcal{A}$ is *ε-node differentially private* if for all neighbor graphs $G = (V, E)$ and $G' = (V', E')$ and for all sets S of possible outputs produced by $\mathcal{A}$:

$$\Pr[\mathcal{A}(G) \subseteq S] \leq e^{\varepsilon} \Pr[\mathcal{A}(G') \subseteq S]$$

[22].

Definition 3. *System Difference Operator.* Given sets A, B the *system difference* $\oplus$ is the sets of elements in either set A or set B, but not in both:

$$A \oplus B = (A \cup b) \setminus (A \cap B)$$

[30].

Definition 4. *Edge neighboring graph.* Given a graph $G = (V, E)$, a graph $G' = (V', E')$ is an edge neighboring graph of G if and only if $|V \oplus V'| + |E \oplus E'| = 1$

[30].

Definition 5. *ε-Edge Differential Privacy.* An algorithm $\mathcal{A}$ satisfies ε-edge differential privacy, where $\varepsilon > 0$ if and only if for any two *edge neighboring graphs* G and G', where $\text{Range}(\mathcal{A})$ denotes the set of all possible outputs of $\mathcal{A}$:

$$\forall T \subseteq \text{Range}(\mathcal{A}) :$$
$$\Pr[\mathcal{A}(G) \in T] \leq e^{\varepsilon} \Pr[\mathcal{A}(G') \in T]$$

[30].

By definition, *ε-node differential privacy* (*ε-node DP*) implies *ε-edge differential privacy* (*ε-edge DP*). Thus, *ε-node DP* is a stronger guarantee than *ε-edge DP*. However, *ε-node differential privacy* is often much harder to achieve without compromising accuracy. This is because many properties of graphs are highly sensitive to the addition or deletion of a single node. For example, given a graph with n nodes, the addition or deletion of a single node could change the number of edges by up to n [25].

1.3 Application of Different Graph-Differential Privacy Methods

For many graph structures, *ε-node DP* is essential, whereas for others, *ε-edge DP* is sufficient. An example of a graph whose analysis offered beneficial results from a result of its publication, but necessitates *ε-node DP* rather than the weaker *ε-edge DP* is used in sociosexual network analysis to understand the transmission of HIV and syphilis cases[18]. For the publication of this network or its data, it is not sufficient to only provide a privacy guarantee on the edges that connect individuals in this network, as knowing whether or not an individual is in this network, or what their degree is reveals substantial amounts of intimate information. Without *ε-node DP*, private information such as an individual's number of sexual partners and potential contact-tracing to HIV or syphilis could be disclosed to the public.

For some networks *ε-edge DP* is too weak a guarantee. *ε-edge DP* operates under the assumption that each edge is independent of one another, however when applied to networks which have dependent edges, such as social networks or co-authorship networks, edges may be deduced for others. Thus, in order to maintain the same level of privacy, dependent graphs require a smaller privacy budget. A solution to this is to approach graph differential privacy with *k-edge DP* where G, G' are neighbors if $|V \oplus V'| + |E \oplus E'| \leq k$[30]. So, for the same level of utility as *ε-edge DP*, *k-edge DP* requires 10 times the privacy budget of *ε-edge DP*. This consequence can make synthetic representations of dependent networks difficult to achieve with both reasonable privacy and utility. Thus, utilizing the privacy budget

efficiently is crucial to maintaining both utility and privacy.

The focus of this paper, the PrivGraph algorithm, only supports ε-edge DP. This limits the applications of PrivGraph to datasets where this level of privacy is sufficient. The sensitivity of certain graph properties (i.e., degree) necessitates the appropriate level of privacy needed and is highly dependent on the nature of the dataset. For example, in a network that represents sexual partners, an individual node's degree (i.e. number of sexual partners), and the edge pairs themselves (sexual partners) may be highly sensitive data. However, in a network representation of social media interactions, the degree of any particular node (i.e., number of interactions) may not be sensitive, but the edge pairs themselves (i.e., users with social media connections) may be sensitive. Clearly, the context of the database creates unique privacy needs and dictates the level of privacy that may be necessary.

2 Related Work

An example of an existing differential privacy framework that employs weak-edge differential privacy is LDPGen [32], which proposes an algorithm for locally differentially private synthetic decentralized social graph generation. This differs from classical differential privacy, where privacy is only guaranteed at the analysis and publishing level. Under local differential privacy (LDP), information is locally perturbed using a randomized mechanism before being collected and incorporated into a large-scale graph. We define LDP's privacy guarantee as follows:

Definition 6. *Local differential privacy.* Given a randomized mechanism $\mathcal{M}$, two neighboring databases that differ in exactly one record, D and D', and some $s \in range(\mathcal{M})$, we say $\mathcal{M}$ satisfies ε-differential privacy if and only if

$$\frac{\Pr[\mathcal{M}(D)] = s}{\Pr[\mathcal{M}(D')] = s]} \leq e^{\varepsilon}.$$

The *privacy budget*, ε, determines the strength of privacy protection. Higher privacy offered by $\mathcal{M}$ is granted by a smaller ε, meaning a closer distribution of $\mathcal{M}(D)$ and $\mathcal{M}(D')$.

Given a decentralized graph, where no party has access to the entire graph, it is challenging to produce a useful synthetic large-scale graph while maintaining LDP. Successful randomized mechanisms under LDP must strike a balance between maintaining privacy and ensuring accuracy. On one extreme, collecting detailed edge-level information requires adding such a high level of noise the data may no longer be useful. On the other hand, collecting less specific graph statistics can distort the graph to the point that it no longer preserves the properties of the original data. LDPGen works by partitioning the graph, then incrementally clustering users based on hyperparameters that aim to minimize their proximity to users in other partitions of the graph at large. This clustering then becomes the input to another synthetic graph generator to produce the final synthetic graph. This occurs in three phases: an initial grouping, group refinement, and graph generation. LDPGen outperforms other synthetic graph generation methods in preserving the relative density of the original graph. This is useful in preserving the structure crucial to the small-world-property, where edges within communities are denser than those between communities.

A standard technique to ensure differential privacy is publishing a synthetic graph that maintains the original graph's structure. While the LDPGen algorithm generates a synthetic graph under local differential privacy (which can also be adapted to be used in a centralized differential privacy setting), it faces many of the same shortcomings that most existing algorithms under classical differential privacy encounter with graph synthesis. Usually, differentially private graph synthesis approaches either introduce too much noise in the synthetic graph or suffer significant information loss during the graph encoding process. This trade-off between perturbation noise and information loss can lead to a final synthetic graph that does not retain a significant portion of the original graph structure.

The PrivGraph algorithm [30] aims to target these weaknesses by exploiting the community information of graph data. The authors of PrivGraph also note that LDPGen has a high space-complexity which constrains its applications to large graphs. In addition, they found that because LDPGen defines a preset number of clusters to encode and perturb all edges uniformly, it could potentially introduce excessive noise in sparser areas. Motivated by these weaknesses, the Quan et al [30] proposed the PrivGraph algorithm, which, similar to LDPGen, also relies on the "small-world property" of large networks; the property that many large networks have short paths between most pairs of nodes. The algorithm uses this property to construct the intuition that edges within "communities" in a network are denser while the edges between communities are more sparse. By exploiting this property and considering different characteristics within and between various communities, PrivGraph is able to extract information about the structure of the graph in order to reconstruct it accurately.

At a high level, there are 3 phases to the PrivGraph algorithm: community division, information extrac-

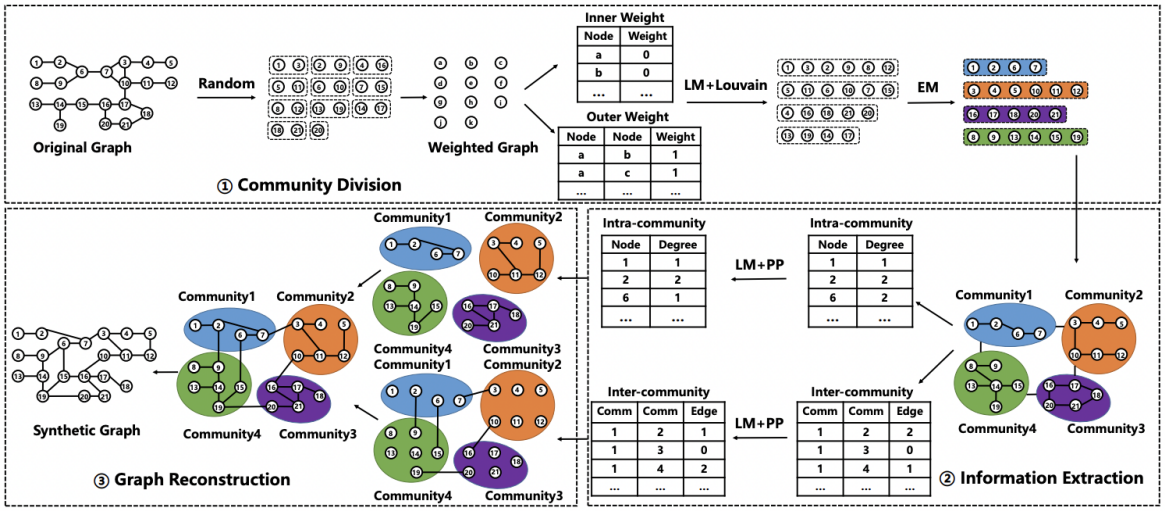

Figure 1: *An overview of the 3 phases in the PrivGraph algorithm. For the formal algorithm, see the original paper [30].*

tion, and graph reconstruction. Figure 1 provides a visual overview of what occurs in each of the phases in the algorithm [30]. In the community division phase, PrivGraph starts by randomly dividing the graph into "communities" or super-nodes, and constructs a weighted graph containing information about edges within a community (inner weights) and the edges between communities (outer weights). Laplacian noise is first added to both the inner and outer weights, and then the Louvain method (a greedy optimization algorithm that extracts communities from large networks) is used as a way to optimize the community divisions. The next phase, information extraction, records information such as each node's degree in their own community and the sum of edges between community pairs from the original graph based on the communities found from the first phase. Again, PrivGraph adds Laplace noise to this information order to satisfy edge-DP. The final phase is the reconstruction of the graph, where intra-community edges are constructed based on the noisy degree of each node, and nodes between different communities are connected randomly under the constraint of the sum of edges. Given a privacy budget of ε, the current PrivGraph implementation sets $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \frac{1}{3}\varepsilon$, where the 2 steps of the community division each take ε_1 and ε_2, respectively, and the information extraction phase consumes ε_3.

In comparison to LDPGen's space compexity of $O(n^2)$, the authors of PrivGraph found that PrivGraph had a significant improvement where its space compexity was $O(m+n)$. Additionally, through their evaluation across various performance metrics, they found that the performance of LDPGen was not as stable across datasets compared to PrivGraph. While PrivGraph already sees notable enhancement compared to LDPGen, the authors utilize set hyperparameters and privacy budget allocation across all datasets, even though this could greatly affect performance depending on the structure and features of the graph. In this paper, we aim to improve PrivGraph's differentially private synthetic graph generation from PrivGraph by predicting these hyperparameters values based on limited knowledge of features of the graph in order to maintain the privacy of the data.

3 Motivation

The PrivGraph algorithm successfully avoids introducing excessive noise, as compared to approaches that perturb individual edges by encoding the graph as a whole in each node's degree distribution. We seek to optimize this method by fine-tuning PrivGraph's hyperparameters, namely: N (number of nodes in community initialization), ε (privacy budget). The privacy budget is further partitioned into $\varepsilon_1, \varepsilon_2$, and ε_3, which are consumed, respectively, by the community initialization, community adjustment, and information extraction phases. It is worth noting that the graph reconstruction phase does not consume any of the privacy budget, as it has no access to the original data. The current PrivGraph implementation distributes the privacy budget equally across the three consumers, i.e. $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \frac{1}{3}\varepsilon$. We propose relaxing this restriction, only requiring that $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 = \varepsilon$. In this paper, we focus on finding a suitable allocation of ε across the three stages, i.e. an optimal N, ε, ε_1, and ε_2.

A larger value of N corresponds to fewer communities, which each contain more nodes. When coupled with a smaller privacy budget, larger communities help to distribute the effect of the noise on the final

results. As we decrease the value of N, the number of initial communities increases, as each community contains fewer nodes. In this case, a larger privacy budget would be better suited, as the injected noise has a comparatively smaller effect. We seek to find the balance of N size and privacy budget to mitigate both the effect of noise and of error caused by random division.

4 Data Collection

Building our model required the collection of a substantial amount of data. This data had to represent a variety of different networks, differing in size and construction, in order to study the granularity of our hyperparameters. We needed a metric for measuring the success of network retention and an easy-to-replicate experimental procedure for collecting the data we use to build our model.

4.1 Network Selection

To generate a training dataset for our model, we wanted to span a large variety of network types to avoid over-fitting. Because most networks that benefit from differential privacy are not available to us, we opted to use publicly-available networks that are less sensitive and may not require differential privacy. This is an inevitable limitation to the training set of our model.

We particularly looked to gather a set of networks which were not incredibly similar to one another, and could be applied to make meaningful change (as opposed to training our model on a random graph). We provided these as supplements to the datasets which the authors of PrivGraph[30] provided alongside their open-source code. Below, we articulate the contents of each dataset.

Chameleon [24] Sourced from The Stanford Network Analysis Project (SNAP)[26], this dataset represents Wikipedia page networks connecting chameleons, crocodiles, and squirrels. Here, a node represents a unique Wikipedia article, and an edge exists between two articles that share a mutual link.

Facebook [15] Collected from SNAP[26], this network is a representation of Facebook friend lists (edited for publication). Here, a node represents a user, and an edge exists between two nodes if either node appears on the other's friend list. Applying differential privacy to this type of network can provide the framework for publishing larger sets of friendship networks from sources like Instagram, X, and BeReal.

CA-HepPh [14] Published through SNAP [26], this dataset represents the Arxiv High Energy Physics - Phenomenology collaboration network. Where a node represents an individual, and an edge exists between two nodes if they collaborated. Differential privacy applied to co-authorship networks could pave the way for the future publication of sensitive co-authorship networks under strong data protection requirements such as FERPA [6] for educational institutions in the US.

Congress [7]. Collected from SNAP[26], this provides Twitter interactions between members of the 117th United States Congress from February to June of 2022. Here, nodes represent congresspeople, and an edge between two nodes represents a Twitter interaction between those two congresspeople. Future analysis of this dataset could yield results about the nature of the relationships between these congresspeople, their political beliefs over time, and more broadly, the study of information diffusion within social networks.

Bitcoin [13, 11] The Bitcoin Alpha Trust Network collected from SNAP[26], it provides a who-trusts-whom network of users of Bitcoin-Alpha. Because Bitcoin is anonymous, the platform attempts to maintain trust-scores associated with the anonymous keys. And, the dataset includes a network of these transactions between users. Venmo, an online transaction platform which is not anonymous, could benefit from the same application of PrivGraph differential privacy as in this Bitcoin network. Applying network differential privacy guarantees could allow a service, such as Venmo, to publish a trust network between transactions of users in order to identify fraud.

Enron [21] Our final dataset, collected through SNAP [26], is the largest of any of our networks. We included this dataset as a validation network, to see if our model, which we trained on medium-sized networks, would extrapolate to larger networks. This dataset represents *Enron* email communications and boasts over 36692 nodes and 183831 edges. The nodes are email addresses, and the edges are representative of if user i sent at least one email to user j, there exists an edge between the two addresses. Inclusion of this larger network in our data is invaluable because we trained our model under restrictions such as lack of compute time. Thus, we are interested in whether such results would scale to networks that take significantly more compute time and resources to generate its differentially private counterpart using PrivGraph. The application of PrivGraph to this network could allow the future publication of large scale telephone carrier communication networks, email networks, and large-scale social media communications such as on Messenger or WhatsApp.

For an undirected graph with n nodes and m edges, we say the *average degree* (Avg Degree) is $\frac{n}{m} \cdot 2$. A

Table 1: *Dataset Statistics*

Dataset	Nodes	Edges	Avg Degree
Chameleon [24]	2,277	31,421	0.1449
Facebook [16]	4,039	88,234	0.09155
CA-HepPh [14]	12,008	118,521	0.2026
Congress [7]	475	13,289	0.0714
Bitcoin [13]	7,604	24,186	0.6288

higher average degree reflects a more densely connected graph.

4.2 Algorithm Success Metrics

In the initial PrivGraph[30] paper, in order to measure the effectiveness of PrivGraph, metrics are provided to measure how proficiently a synthetic graph retains features of its non-private counterpart. We utilize these metrics to measure the success of a resulting graph. In particular, we chose metrics such as community discovery, node information, degree distribution, path condition, and topology structure as our interests when building the model. All metrics are formally defined in "Our Model."

4.3 Hyperparameter Selection

Motivated by providing a sufficient search space for our estimation model, across our diverse datasets, we performed a modified grid-search across the hyperparameter space. We performed a grid-search for the following values of epsilon:

$$\varepsilon \in \{0.5, 2, 3.5\}$$

For small values of ε, i.e. $\varepsilon \leq 1$, we searched over a community initialization hyperparameter of:

$$N \in 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55$$

And for larger values of ε, we searched over a smaller range of:

$$N \in 5, 10, 15, 20, 25, 30, 35$$

We decided to include larger values of N for smaller ε, instead of stopping at 35 as in the original code, because both we and the authors have observed that fewer initial communities means that the initial data collected (e.g. intra- and inter-community degrees) is more robust to noise.

We allocated $e_1, e_2, e_3 \in \{0.1, 0.2, \ldots, 0.9\}$ such that $e_1 + e_2 + e_3 = 1$. We then let $\varepsilon_i = e_i \cdot N : \forall i$. This allocates our privacy budget between the three stages Community Division (ε_1), Graph Reconstruction (ε_2), and Information Extraction (ε_3), such that $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 = \varepsilon$. Therefore, the division amongst stages still satisfies the *ε-edge DP* guarantee.

4.4 Experimental Setup

PrivGraph is open-source and accessible on GitHub [31]. We altered the initial PrivGraph code to allow for concurrently testing of different groups of hyperparameters. We prepared to run grid-search as described in 4.3 across all networks referenced in 4.1 for all hyperparameter combinations. To measure the similarity of the synthetic graph structure against the original graph, we compute performance metrics [30] as described in 5. We then output these results as .csv files which were later used to train our model.

4.5 Execution

Since PrivGraph has a time complexity of $O(n^2)$, when scaled to larger datasets (i.e. *CA-HepPh* and *Facebook*), the algorithm takes significantly longer to run each iteration. For each combination of hyperparameters, we ran 10 iterations and computed the average for each metric. Due to the high number of hyperparameter combinations performed in our exhaustive grid-search, the total run-time for the entire experiment of these larger datasets would often take days at a time. Given our limited time and resources, we opted not to run our initial experimental iterations of our largest dataset, *Enron*. Instead, we used our model to predict the optimal hyperparameters for *Enron* as an unseen dataset. We also ran with a larger step (fewer combinations) for the larger datasets such as *CA-HepPh*. A complete list of our experiment results can be found in our GitHub repository.

5 Our Model

We constructed a multiple linear regression (MLR) model that predicts the performance of a set of hyperparameter values, namely ε, ε_1, ε_2 and N, based on various features of the graph.

We decided to use multiple linear regression for several reasons. Our first reason is that this method requires virtually no training time unlike other more complex models. This was critical to our experiment, given our already limited computing resources which were in constant use, running experiments on multiple datasets with continually changing data. Our second reason was that we could directly inspect the coefficients of each covariate and get a sense of the impact they have on the predictions. This was essential in our ability to fine-tune our methods as our experiment progressed.

The general MLR model can be written as

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u$$

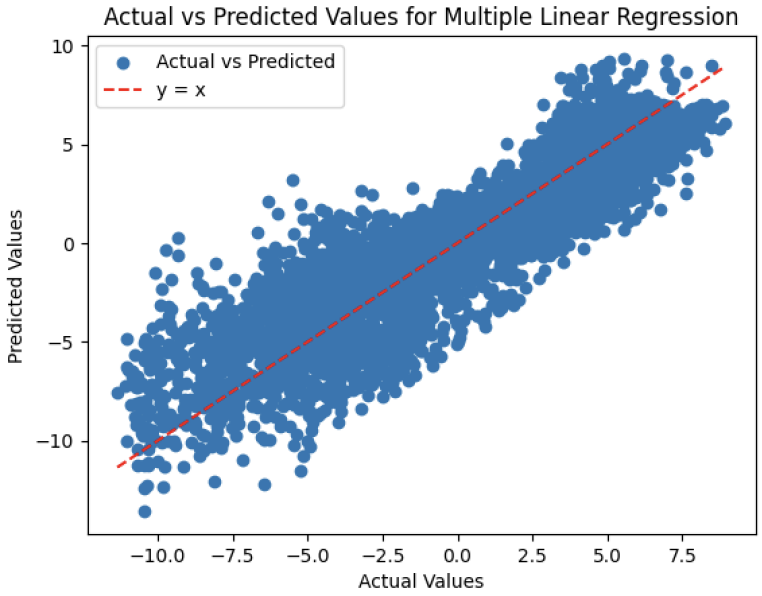

Figure 2: *A visualization of our model's performance. We compare the actual values of our performance measure function versus the predicted performance values across a wide variety of hyperparameters and datasets.*

where β_0 is the intercept, β_i is the coefficient associated with variable x_i, and variable u is the residual term that contains other omitted variables that affect y [29].

In our use, the dependent variable of interest, y, is a score we calculate based on the seven metrics used in the original PrivGraph paper.

- **Community Discovery** We apply the Louvain method to obtain partitions from the original and synthetic graphs, then measure the difference between the group partitions using Normalized Mutual Information (NMI) [30].
- **Node Information** We rank nodes by their influence, using their eigenvector centrality (EVC) score. We let N, $\hat{N}$ denote the top 1% eigenvector values in the original and synthetic graphs, respectively.

$$Overlap_{Node} = \frac{|N \bigcap \hat{N}|}{|N|}$$

[30].
- **Mean Absolute Error (MAE)** To find the MAE of such nodes' EVC scores, we let s_i and $\hat{s_i}$ denote the EVC scores of the i-th most influential node in the original and synthetic graphs, respectively, then calculate as follows:

$$MAE_{EVC} = \frac{1}{T} \sum_{i=1}^{T} |\hat{s_i} - s_i|$$

- **Degree Distribution** We measure the difference of the degree distribution of the original graph, $P(x)$, and that of the synthetic graph, $\hat{P}(x)$, using Kullback-Leibler (KL) [10] divergence.

$$D_{KL}(P||\hat{P} = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{\hat{P}(x)}\right)$$

[30].
- **Path Condition** We measure the diameter as the maximum distance among all path-connected pairs of nodes, which we label as D for the original graph and $\hat{D}$ for the synthetic graph. Setting δ to be a small positive constant to ensure a non-zero denominator, we find the Relative Error (RE) of the diameters as follows:

$$RE_{Diam} = \frac{|\hat{D} - D|}{max(\delta, D)}$$

[30].
- **Topology Structure** We compute the clustering coefficients and the modularities of the original and synthetic generated graph and compute the relative error between the two. We define modularity as follows:

$$Q = \sum_{C} [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2]$$

where $\sum_{in}$ is the sum of the weights of the edges inside the community C, $\sum_{tot}$ is the sum of the weights of the edges incident nodes in C, and m represents the sum of the weights of all edges. Similar to above, we find the Relative Error of modularity for the original graph Q and the modularity of the synthetic graph $\hat{Q}$ as follows:

$$RE_{Mod} = \frac{|\hat{Q} - Q|}{max(\delta, Q)}$$

[30].

5.1 Normalization

To ensure that the metrics weighted equally, we apply Z-score normalization to them ($Z = \frac{x-\mu}{\sigma}$), before calculating the overall score by summing the normalized metrics. We then used `LinearRegression` from the `sklearn` Python package to fit the model by minimizing the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. This means solving the equation

$$\min_{\beta} ||X\beta - y||_2^2$$

where β is the vector of coefficients [19].

5.2 Feature Selection

We experimented with manual and automatic feature selection. We start with $\varepsilon, N, \varepsilon_1, \varepsilon_2$, number of nodes ($n$), number of edges ($m$), as well as average degree ($2 * \frac{n}{m}$) as the basic building blocks for higher- power terms. In manual feature selection, we construct features that we believe should play a role in the performance of the hyperparameter set. For example, in Section E.1 of the PrivGraph paper, the authors note that when the privacy budget is small, a larger N is more beneficial since it reduces the number of initial communities and reduces the relative impact of the injected Laplacian noise. Thus we added an interaction term $\varepsilon * N$ which would capture this relationship. We include features that lead to a sizable improvement in the mean squared error and the R^2 of the model when evaluated on a randomized test set.

For automatic feature selection, we tried several approaches that consider all the possible polynomial terms (including interaction terms) up to degree 2. We tried using recursive feature elimination (RFE) from the `sklearn` package to iteratively prune features with a small enough coefficient. We tried RFE up to a predetermined number of features, as well as RFE with five-fold cross validation that chooses the number of features. We found that the out-of-the-box RFE approaches don't fit the model as well as the manual one as measured by R^2 and mean squared error, which could be due to the fact the magnitude of the values of the features is not taken into account. Finally, we started off by including all polynomial features in the model, and eliminated those that have a small `coefficient * mean of the feature` to avoid overfitting. This last approach has the lowest R^2 and MSE, can be easily adapted to different types of graphs, and is the approach we will evaluate and is less prone to producing extreme predictions on perturbed data.

5.3 Extra Budget Consumption

In order to extract information about the graphs for hyperparameter tuning, extra privacy budget has to be used. We chose coarse-grained features of the graph that would be robust to noise such as the number of nodes and the number of edges, instead of finer-grained features such as the adjacency matrix or the eigenvector. This helps us take away as little budget as possible while using graph-specific knowledge to tune hyperparameters.

To compute the n and m in the graph while preserving differential privacy, we first find the actual n and m by summing over the datapoints in our database D:

$$sum(D) = \sum_{i=1}^{n} D_i$$

Then we find the sensitivity of sum. The L1 sensitivity of a function $f : D^n \to \mathbb{R}^d$ is the smallest number $S(f)$ such that for all $x, x' \in D^n$ which differ in a single entry [5]:

$$||f(x) - f(x')||_1 \leq S(f)$$

Put another way, this tells us the most any one data point can change the output of f, and helps us calibrate the standard deviation of the noise. The sensitivity of sum is just 1 because removing a node or an edge can change the sum by at most one. We can now create a differentially private version of the sum function [1].

$$dpSum(D) = sum(D) + Lap(0|\tfrac{S(f)}{\varepsilon})$$
$$dpSum(D) = sum(D) + Lap(0|\tfrac{1}{\varepsilon})$$

Here, $Lap(x|b) = \frac{1}{2b}e^{-\frac{|x|}{b}}$ and ε is the privacy budget allocated to the extraction of n or m in the graph.

The final coefficients for our MLR model can be seen in Table 2.

6 Evaluation

Of course, our modified PrivGraph algorithm must also satisfy ε-edge DP to be useful in application. Unmodified, PrivGraph satisfies ε-edge DP, where $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$[30].

Our proposed modifications to the PrivGraph algorithm will maintain that $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3 + \varepsilon^*$ where ε^* is the extra budget we use to extract graph information. Subsequently, by the ε-edge DP property of PrivGraph, our modified PrivGraph also satisfies ε-edge DP.

6.1 Comparison

To evaluate how our model would perform across different privacy budgets, we predicted hyperparameters for each value $\varepsilon \in 0.5, 2, 3.5$ for both seen and

Table 2: *Model Coefficients*

Coefficient	Value
m^2	$6.939 \cdot 10^{-9}$
$m \cdot d_{\text{avg}}$	$-1.750 \cdot 10^{-5}$
n^2	$-8.815 \cdot 10^{-7}$
ε	9.877
N	0.5206
ε_2	16.05
ε_1	12.63
$n \cdot m$	$4.190 \cdot 10^{-8}$
$\varepsilon \cdot m$	$6.632 \cdot 10^{-5}$
ε^2	-1.061
$\varepsilon \cdot n$	$-6.522 \cdot 10^{-4}$
$\varepsilon \cdot d_{\text{avg}}$	$-7.670 \cdot 10^{-2}$
$N \cdot d_{\text{avg}}$	$-3.831 \cdot 10^{-3}$
N^2	$-4.138 \cdot 10^{-4}$
ε_2^2	-11.33
$\varepsilon \cdot N$	$-5.642 \cdot 10^{-2}$
$\varepsilon_1 \cdot m$	$1.510 \cdot 10^{-4}$
$\varepsilon_1 \cdot n$	$-1.333 \cdot 10^{-3}$
$N \cdot \varepsilon_1$	$-1.948 \cdot 10^{-1}$
$\varepsilon_1 \cdot d_{\text{avg}}$	$-1.845 \cdot 10^{-1}$

unseen datasets. We first take 2% of the overall privacy budget ε to obtain the number of nodes and edges. Then, using the predicted hyperparameter N and privacy budget allocation ratios ε_1 and ε_2 (implying $\varepsilon_3 = 1 - \varepsilon_1 - \varepsilon_2$) from our model, we ran PrivGraph on the datasets and evaluated the performance by comparing the metrics with the baseline of $N = 20$ and $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \frac{1}{3}\varepsilon$. The authors of PrivGraph found these baseline hyperparameters to be a reasonable default across the datasets through their evaluation [30]. Our model's predicted hyperparameters for each dataset and ε can be seen in Table 4.

Figure 3 illustrates a visual representation of the comparison between the performance of PrivGraph implemented with baseline hyperparameters versus our predicted hyperparameters. The first row illustrates the NMI results, where a higher value indicates a better performance. We see that across all datasets, community discovery had the highest performance improvement. Clearly, our prediction of N, the number of nodes in community initialization, aids the PrivGraph algorithm to more accurately detect communities within a network.

Although we do see consistent improvement in performance across all metrics as noted in Table 5, there is also high variance in the percentage of improvement between the baseline and our optimized hyperparameter models. Notably, the error bounds lean toward indefinite improvement using our model's

Table 3: *Notation*

Variable	Description
n	Number of Nodes in the Network
m	Number of Edges in the Network
ε	Privacy Budget
ε_1	Community Division Privacy budget
ε_2	Graph Reconstruction Privacy budget
ε_3	Information Extraction Privacy budget
d_{avg}	$\frac{n}{m} \cdot 2$, Average Degree
nmi	Community Discovery Metric
evc_overlap	EVC Node Information Overlap
evc_mae	EVC Node Info Mean Average Error
deg_kl	KL Divergence of Degree Distribution
diam_rel	Relative Error of Diameters
cc_rel	Relative Error of Clustering Coefficients
mod_rel	Relative Error of Modularity

hyperparameters rather than the baseline hyperparameters. Although the percentage of improvement is variable, all means are still above 0% which implies that if one were to use the optimized hyperparameters that our model suggests, they could expect to see improvement in at least one of the metrics.

Table 4: *Predicted Optimal Hyperparameters*

Dataset	ε	ε_1 ratio	ε_2 ratio	N
Bitcoin	0.5	0.1	0.7	55
	2	0.3	0.6	35
	3.5	0.3	0.6	55
Chameleon	0.5	0.3	0.6	40
	2	0.3	0.6	30
	3.5	0.5	0.4	15
CA-HepPh	0.5	0.3	0.6	45
	2	0.4	0.5	30
	3.5	0.5	0.4	20
Congress	0.5	0.1	0.7	35
	2	0.2	0.7	25
	3.5	0.4	0.5	10
Facebook	0.5	0.5	0.4	30
	2	0.6	0.3	15
	3.5	0.7	0.2	5
Enron	0.5	0.1	0.7	50
	2	0.1	0.7	50
	3.5	0.1	0.7	30

The average improvement of our selected hyperparameters over the seven metrics is about 13%, excluding zeroes which cannot be divided by. That said, the improvement is uneven and our hyperparameters performed worse on three of them. We believe

Figure 3: *Comparison of PrivGraph's performance with default hyperparameters and our predicted hyperparameters as seen in Table 4. Each column represents a dataset, and each row represents an evaluation metric. In each plot, the x-axis denotes the privacy budget ε, and the y-axis denotes the performance metric (normalized across the datasets). For the first 2 rows, a higher value is better; a lower value is better in the other five rows.*

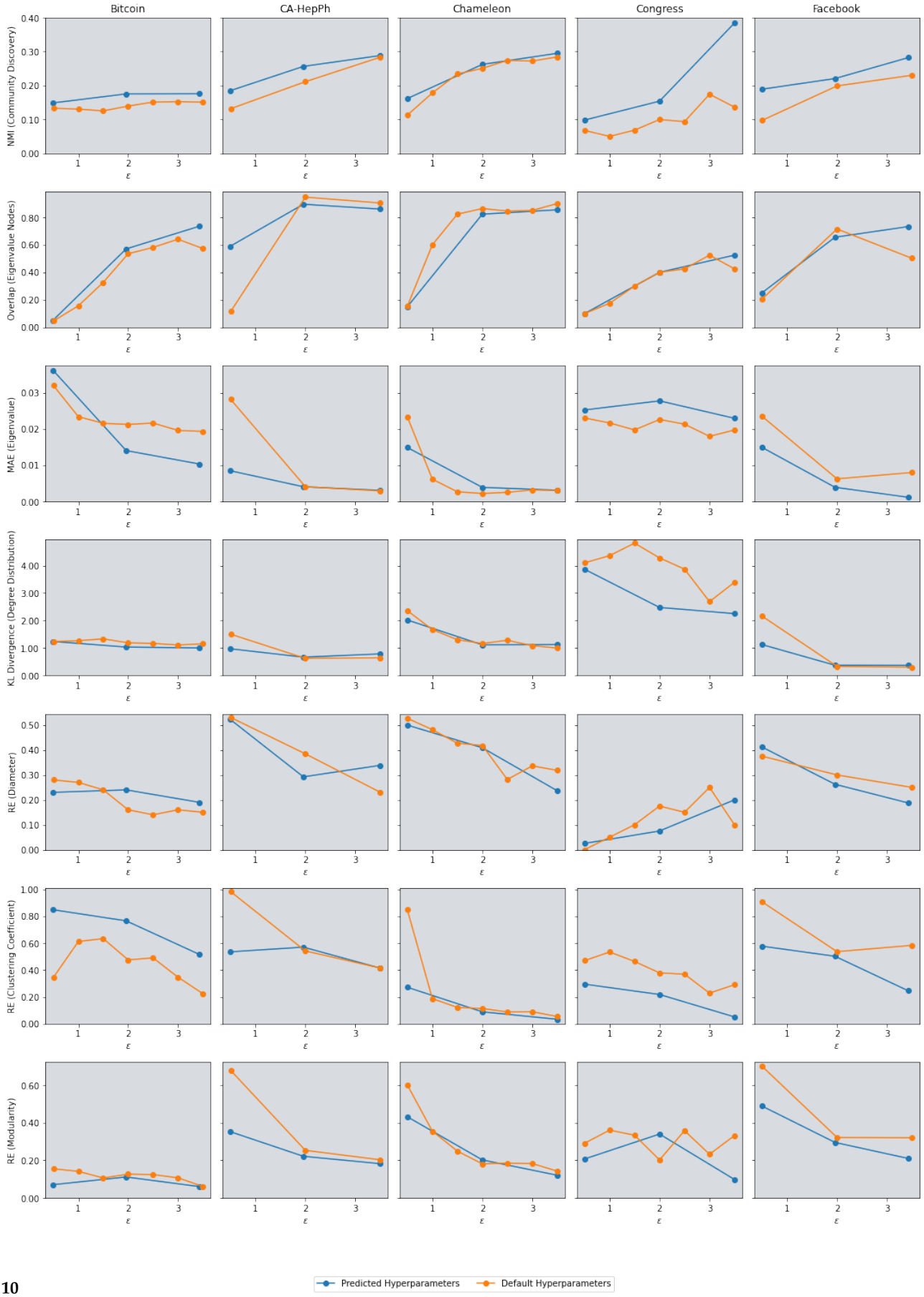

Table 5: ***Metric Comparison:*** *The Performance % Improvement reflects the percent improvement across a variety of metrics. The performance percentage improvement is calculated by comparing the metric of the baseline hyperparameters ($\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \frac{1}{3}, N = 20$) to the metrics from our predicted optimal hyperparameters outlined in Table 4. A positive value reflects that our model's predicted hyperparameters outperforms the baseline by that percentage. These values are averaged across all of our datasets. We also provide a bound on the error of our performance improvement by measuring the standard deviation of the percent improvement.*

Metric	Performance % Improvement
nmi	35.92 ± 36.05
evc_overlap	12.55 ± 21.79
evc_MAE	9.23 ± 40.08
deg_kl	8.80 ± 20.69
diam_rel	4.29 ± 21.98
cc_rel	0.39 ± 69.07
mod_rel	13.77 ± 24.54

this is due to the fact that we attempt to capture the performance across all metrics with a single score, but this lacks granularity needed to improve a specific metric. We have observed in our experiments that it is difficult for a hyperparameter set to score highly on all the metrics, since each metric captures a different aspect of how well the synthetic graph preserves the original structure. In the future, users could modify our notebooks to weigh the metrics differently according to their needs.

Our hyperparameters performed better than we expected on the unseen dataset *Enron*, with an overall improvement of 8%. We did not expect there to be any improvement because it is significantly larger than all the other datasets the model is based on, which was the reason we did not include it in the first place. Granted, the improvement on Enron is lower than on other datasets, but it shows that the model could work for larger and unseen datasets, and potentially improved with more experiments in the future.

Furthermore, we observe that the improvement is larger for lower privacy budget: 25% for $\epsilon = 0.5$, 7% for $\epsilon = 2$ and 5% for $\epsilon = 3.5$. This is likely due to the fact that the model often suggested a larger N for smaller budgets, which is more resilient to noise.

7 Future Work

PrivGraph is only able to guarantee ε-edge DP, so more work must be done to create an algorithm that satisfies ε-node DP. This is a much stricter guarantee, and is harder to achieve. As aforementioned, the addition or removal of an node often has a larger effect on the overall graph structure than the addition or removal of an edge, thus introducing increased complexity. PrivGraph and its hyperparameters could also be modified to guarantee k-edge DP. This would require further work in determining how to allocate the privacy budget in way that enforces this stronger guarantee while retaining crucial network structure information. We are interested in providing mathematical analysis on running such an ε-node DP algorithm with varying values of hyperparameters to see the effects of such changes on their corresponding levels of protection. Likewise, further work lies in analyzing values of k in k-edge DP, and how those values correspond to levels of protection.

Another topic of interest is applying a Bayesian Optimization approach to hyperparameter selection while maintaining differential privacy guarantees [12, 27]. This approach would minimize the amount of time for hyperparemeter selection such as on the large dataset of *Enron*, while still providing differential privacy with the publication and calculation of optimal hyperparameters. This application could save essential time in the computation of publishable differentially private networks using PrivGraph.

We also found that in order to come up with a more effective model, or test different estimation approaches, we require significantly more data. In the future, we could provide data across many different networks and more fine-grained hyperparameter selections. This would allow us to build a more robust model that could ease the ability of prediction for a large variety of networks.

Similarly, our model is limited to providing overall improvement of all metrics. However, a data publisher may have interest in maintaining some graph structures, but less interest in others. Thus, we could provide additional analysis of our model so users can generate hyperparameters that may tailor to their specific metrics of interest.

8 Conclusion

The intricate structure, properties, and sensitivities of datasets vary by their content. Given these complexities and the significant risks of data leakage, it is crucial to tailor the data privatization method to the specific needs of each dataset. Our analysis has shed light on how granular modifications to the PrivGraph algorithm can significantly influence the accuracy and privacy of its output.

Using our estimation model, which maps hyperparameters of PrivGraph to expected results, we provided analysis of the implications of changes to varying hyperparameters on the resulting differentially private graph. Our experiment considers a range of

network types and how their differing structures and uses could play a role in finding their optimal hyperparameters. We compare our model's predicted hyperparameters to those initially proposed by the PrivGraph[30] authors and analyse the differences in results. Finally, we address some of the limitations of PrivGraph and the application of differential privacy on a wide variety of networks.

Our model and its analysis highlight the complexities of creating a differentially private publishable network that maintains the initial properties of the original graph. With our findings, we conclude that optimizing the hyperparameters of the PrivGraph algorithm can yield significant benefits on the retention of crucial properties of the initial network.

References

[1] Aaroth. "Lecture 1: Introduction to Differential Privacy and the Laplace Mechanism", *University of Pennsylvania*, https://www.cis.upenn.edu/ aaroth/ chatgpt_lecture_notes.pdf.

[2] "Attorney General Becerra Announces 8.69 Million Settlement against Anthem, Inc., over Failure to Protect Patients' Personal Data." *State of California - Department of Justice - Office of the Attorney General*, 1 Oct. 2020, oag.ca.gov/news/press-releases/attorney-general-becerra-announces-869-million-settlement-against-anthem-inc.

[3] Biden. "Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence." *The White House*, The United States Government, 30 Oct. 2023, www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/.

[4] Dwork, Cynthia. "Differential privacy". ICALP 2006: automata, languages and programming, pp 1–12. 2006. 10.1007/11787006.1

[5] Dwork, McSherry, Nissim, Smith. "Calibrating Noise to Sensitivity in Private Data Analysis." *Theory of Cryptography Conference (TCC)*, Springer, 2006.

[6] "Family Educational Rights and Privacy Act (FERPA)." *US Department of Education (ED)*, ed.gov.

[7] Fink, Omodt, Zinnecker, and Sprint: "A Congressional Twitter network dataset quantifying pairwise probability of influence." *Data in Brief*, 2023.

[8] Grayer, Annie, and Sean Lyngaas. "Hundreds of US Lawmakers and Staff Affected by Data Breach | CNN Politics." *CNN*, Cable News Network, 9 Mar. 2023, www.cnn.com/2023/03/08/politics/data-breach-us-lawmakers/index.html.

[9] Hunter, Benjamin et al. "The Role of Artificial Intelligence in Early Cancer Diagnosis." *Cancers* vol. 14, no.6, 2022, 1524, doi:10.3390/cancers14061524.

[10] Kullback, "Information Theory and Statistics." *Courier Corporation*, 1997.

[11] Kumar, S., et al. "REV2: Fraudulent User Prediction in Rating Platforms." *11th ACM International Conference on Web Searchand Data Mining (WSDM)*, 2018.

[12] Kusner, Matt J., et al. Differentially Private Bayesian Optimization. 2015.

[13] Kumar, Spezzano, Subrahmanian, and Faloutsos. "Edge Weight Prediction in Weighted Signed Networks." *IEEE International Conference on Data Mining (ICDM)*, 2016.

[14] Leskovec, Kleinberg, and Faloutsos. "Graph Evolution: Densi- fication and Shrinking Diameters." *ACM transactions on Knowledge Discovery from Data (TKDD*, vol. 1, no.1, 2007

[15] Leskovec and Mcauley. "Learning to Discover Social Circles in Ego Networks." *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[16] McAuley and Leskovec. "Learning to Discover Social Circles in Ego Networks." *NIPS*, 2012.

[17] Pan, Di, and Kapil, Rajwani. "Implementation of Simulation Training During the COVID-19 Pandemic: A New York Hospital Experience." *Simulation in healthcare : journal of the Society for Simulation in Healthcare*, vol. 16, no.1, 2021, pp. 46-51, doi:10.1097/SIH.0000000000000535

[18] Pasquale et al. "Lost and found: applying network analysis to public health contact tracing for HIV." *Applied Network Science*, vol. 6, no. 13, 2021, doi:10.1007/s41109-021-00355-w

[19] Pedregosa, Fabian, et al. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research*, vol.12, 2011, pp.2823-2830.

[20] Qin, Zhan, et al. "Generating Synthetic Decentralized Social Graphs with Local Differential Privacy." *Proceedings of the ACM on Interactive, Mobile, Wearable, and Ubiquitous Technologies*, 2017, pp. 425-438. doi:10.1145/3133956.3134086.

[21] R. A. Rossi and N. K. Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization. In AAAI, 2015.

[22] Raskhodnikova and Smith. "Differentially Private Analysis of Graphs." *Boston University*.

[23] Ritchie, John Newman; Amy and Nick Jones. "Equifax to Pay 575 Million as Part of Settlement with FTC, CFPB, and States Related to 2017 Data Breach." *Federal Trade Commission*, 2 June 2023, www.ftc.gov/news-events/news/press-releases/2019/07/equifax-pay-575-million-part-settlement-ftc-cfpb-states-related-2017-data-breach.

[24] Rozemberczki, Allen, and Sarkar. "Multi-Scale Attributed Node Embedding." *Journal of Complex Networks*, vol.9, no.2, 2021, cnab014.

[25] Sealfon, Ullman. "Efficiently Estimating Erdos-Rényi Graphs with Node Differential Privacy." *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada.

[26] "Snap for C++: Stanford Network Analysis Platform." *SNAP*, snap.stanford.edu/.

[27] Snoek, Jasper, et al. Practical Bayesian Optimization of Machine Learning Algorithms. 2012.

[28] The White House. "Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence". October 30, 2023.

[29] Wooldridge. "Introductory Econometrics: A Modern Approach."

[30] Yuan, Quan, et al. "PrivGraph: Differentially Private Graph Data Publication by Exploiting Community Information." *2023 USENIX Security*, 13 Oct. 2023, doi:10.48550/arXiv.2304.02401.

[31] Yuan, Quan, et al. "PrivGraph: Differentially Private Graph Data Publication by Exploiting Community Information." *USENIX Security*, 2023, https://github.com/Privacy-Graph/PrivGraph.

[32] Yang, Zhong, Li, and Jie. "Bi-Directional Joint Inference for User Links and Attributes on Large Social Graphs." *Proceedings of the 26th International Conference on World Wide Web Companion*, pages, 2017, pp. 564–573.