发表论文精选

LF-GDPR: A Framework for Estimating Graph Metrics with Local Differential Privacy

Qingqing Ye, *Member, IEEE,* Haibo Hu, *Senior Member, IEEE,* Man Ho Au, *Member, IEEE,* Xiaofeng Meng, *Member, IEEE,* Xiaokui Xiao, *Member, IEEE*

Abstract—Local differential privacy (LDP) is an emerging technique for privacy-preserving data collection without a trusted collector. Despite its strong privacy guarantee, LDP cannot be easily applied to real-world graph analysis tasks such as community detection and centrality analysis due to its high implementation complexity and low data utility. In this paper, we address these two issues by presenting LF-GDPR, the first LDP-enabled graph metric estimation framework for graph analysis. It collects two atomic graph metrics — the adjacency bit vector and node degree — from each node locally. LF-GDPR simplifies the job of implementing LDP-related steps (e.g., local perturbation, aggregation and calibration) for a graph metric estimation task by providing either a complete or a parameterized algorithm for each step. To address low data utility of LDP, it optimally allocates privacy budget between the two atomic metrics during data collection. To demonstrate the usage of LF-GDPR, we show use cases on two common graph analysis tasks, namely, clustering coefficient estimation and community detection. The privacy and utility achieved by LF-GDPR are verified through theoretical analysis and extensive experimental results.

Index Terms—Local differential privacy; Graph metric; Privacy-preserving graph analysis.

1 Introduction

With the prevalence of big data and machine learning, graph analytics has received great attention and nurtured numerous applications in web, social network, transportation, and knowledge base. However, recent privacy incidents, particularly the Facebook privacy scandal, pose real-life threats to any **centralized** party who needs to safeguard graph data of individuals while providing graph analysis service to third parties. In that scandal, a third-party developer Cambridge Analytica retrieves the personal profiles of 87 million Facebook users through the Facebook Graph API for third-party apps [1], [2]. The main cause is that this API allows these apps to access the **friends list** of a user by a simple authorization, through which these apps propagate like virus in the social network. Unfortunately, most existing privacy models on graph assume a centralized trusted party to release the graph data that satisfies certain privacy metrics, for example, the k-neighborhood anonymity [3], k-degree anonymity [4], k-automorphism [5], k-isomorphism [6], and differential privacy [7], [8]. However, in practice even Facebook cannot be fully trusted or is in the centralized position to release graph data on behalf of each user. For decentralized graphs in which each user or party locally maintains a limited view of the graph, there is even no such a central party. These graphs, such as the World Wide Web, federated knowledge graphs, peer-to-peer (e.g., vehicular and mobile ad-hoc) and blockchain networks, and contact tracing graph for COVID-19, are in a more compelling need to find alternative privacy models without a trusted party [9].

A promising model is local differential privacy (LDP) [10], where each individual user **locally perturbs her share of graph metrics** (e.g., node degree and adjacency list, depending on the graph analysis task) before sending them to the data collector for analysis. As such, the data collector does not need to be trusted. A recent work *LDPGen* [11] has also shown the potential of LDP for graph analytics. In that work, LDP is used to collect node degree for synthetic graph generation. However, such solution is usually task specific — for different tasks, such as centrality analysis and community detection, dedicated LDP solutions must be designed from scratch. To show how complicated it is, an LDP solution usually takes four steps: (1) selecting graph metrics to collect from users for the target metric (e.g., clustering coefficient, modularity, or centrality) of this task, (2) designing a local perturbation algorithm for users to report these metrics under LDP, (3) designing a collector-side aggregation algorithm to estimate the target metric based on the perturbed data, (4) designing an optional calibration algorithm for the target metric if the estimation is biased. Step (4) is important as locally perturbed data often causes bias (i.e., deviation from the true mean) in the collector-side statistics. Obviously, working out such a solution **requires in-depth knowledge of LDP**, which hinders the embrace of LDP by more graph applications.

In this paper, we address this challenge by presenting LF-GDPR (Local Framework for Graph with Differentially Private Release), the first LDP-enabled graph metric estimation framework for general graph analysis. It simplifies

- *Qingqing Ye is with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, and the School of Information, Renmin University of China. E-mail: qqing.ye@polyu.edu.hk*
- *Haibo Hu is with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, and Polyu Shenzhen Research Institute. E-mail: haibo.hu@polyu.edu.hk*
- *Man Ho Au is with the Department of Computer Science, The University of Hong Kong. E-mail: allenau@cs.hku.hk*
- *Xiaofeng Meng is with the School of Information, Renmin University of China. E-mail: xfmeng@ruc.edu.cn*
- *Xiaokui Xiao is with the School of Computing, National University of Singapore. E-mail: xkxiao@nus.edu.sg*

Manuscript received April 19, 2005; revised August 26, 2015.

the job of a graph application to design an LDP solution for a graph metric estimation task by providing complete or parameterized algorithms for steps (2)-(4) as above. As long as the target graph metric can be derived from the two atomic metrics, namely, the adjacency bit vector and node degree, the parameterized algorithms in steps (2)-(4) can be completed with ease. Furthermore, LF-GDPR features an optimal allocation of privacy budget between the two atomic metrics. To illustrate the usage of LF-GDPR, we will also show use cases on two common graph analysis tasks, namely, clustering coefficient estimation and community detection. To summarize, our main contributions in this paper are as follows.

1) This is the first LDP-enabled graph metric estimation framework for a variety of graph analysis tasks.
2) We provide complete or parameterized algorithms for local perturbation, collector-side aggregation, and calibration.
3) We present an optimal solution to allocate the privacy budget between adjacency bit vector and node degree.
4) We show two use cases of LF-GDPR and compare their performance with existing methods on real datasets.

The rest of the paper is organized as follows. Section 2 introduces preliminaries on local differential privacy and its application in graph analytics. Section 3 presents an overview of LF-GDPR. Section 4 describes the implementation details of this framework. Sections 5 and 6 show the detailed usage of LF-GDPR in two use cases. Section 7 presents the experimental results, followed by Section 8 which reviews related work. Section 9 draws a conclusion with future work.

2 PRELIMINARIES

2.1 Local Differential Privacy

Differential privacy [12] (DP) is defined on a randomized algorithm $\mathcal{A}$ of a sensitive database. $\mathcal{A}$ is said to satisfy ϵ-differential privacy, if for any two neighboring databases D and D' that differ only in one tuple, and for any possible output s of $\mathcal{A}$, we have $\frac{\Pr[\mathcal{A}(\mathrm{D})=\mathrm{s}]}{\Pr[\mathcal{A}(\mathrm{D}')=\mathrm{s}]} \leq e^{\epsilon}$. In essence, DP guarantees that after observing any output of $\mathcal{A}$, an adversary cannot infer with high confidence whether the input database is D or D', thus hiding the existence or non-existence of any individual tuple.

Centralized DP requires the real database stored in a trusted server where the randomized algorithm $\mathcal{A}$ can execute. However, this assumption does not hold in many real-world applications. **Local differential privacy (LDP)** [10], [13] is proposed to assume each individual is responsible for her own tuple in the database. In LDP, each user locally perturbs her tuple using a randomized algorithm before sending it to the untrusted data collector. Formally, a randomized algorithm $\mathcal{A}$ satisfies ϵ-local differential privacy, if for any two input tuples t and t' and for any output t^*, $\frac{\Pr[\mathcal{A}(\mathrm{t})=\mathrm{t}^*]}{\Pr[\mathcal{A}(\mathrm{t}')=\mathrm{t}^*]} \leq e^{\epsilon}$ holds. In essence, LDP guarantees that after observing any output tuple t^*, the untrusted data collector cannot infer with high confidence whether the input tuple is t or t'.

2.2 Local Differential Privacy on Graphs

In this paper, a graph G is defined as $G = (V, E)$, where $V = \{1, 2, ..., n\}$ is the set of nodes, and $E \subseteq V \times V$ is the set of edges. For the node i, d_i denotes its degree and $\boldsymbol{B}_i = \{b_1, b_2, ..., b_n\}$ denotes its *adjacency bit vector*, where $b_j = 1$ if and only if edge $(i, j) \in E$, and otherwise $b_j = 0$. The adjacency bit vectors of all nodes constitute the *adjacency matrix* of graph G, or formally, $\boldsymbol{M}_{n\times n} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, ..., \boldsymbol{B}_n\}$.

As with existing LDP works, we concern attacks where an adversary can infer with high confidence whether an edge exists or not, which compromises a user's relation anonymity in a social network. As a graph has both nodes and edges, LDP can be applied to either of them, which leads to *node local differential privacy* [14] and *edge local differential privacy* [15]. Node LDP (resp. edge LDP) guarantees the output of a randomized algorithm does not reveal whether any individual node (resp. edge) exists in G.

Definition 2.1. *(Node local differential privacy).* A randomized algorithm $\mathcal{A}$ satisfies ϵ-node local differential privacy (a.k.a., ϵ-node LDP), if and only if for any two adjacency bit vectors $\boldsymbol{B}, \boldsymbol{B}'$ and any output $s \in range(\mathcal{A})$, $\frac{\Pr[\mathcal{A}(\boldsymbol{B})=s]}{\Pr[\mathcal{A}(\boldsymbol{B}')=s]} \leq e^{\epsilon}$ holds.

Definition 2.2. *(Edge local differential privacy).* A randomized algorithm $\mathcal{A}$ satisfies ϵ-edge local differential privacy (a.k.a., ϵ-edge LDP), if and only if for any two adjacency bit vectors $\boldsymbol{B}$ and $\boldsymbol{B}'$ that differ only in one bit, and any output $s \in range(\mathcal{A})$, $\frac{\Pr[\mathcal{A}(\boldsymbol{B})=s]}{\Pr[\mathcal{A}(\boldsymbol{B}')=s]} \leq e^{\epsilon}$ holds.

Both node and edge LDP satisfy sequential composition.

Theorem 2.3. *(Sequential Composition)* [11]. Given c randomized algorithms $\mathcal{A}_i (1 \leq i \leq c)$, each satisfying ϵ_i-node (resp. edge) LDP, the collection of these algorithms $\mathcal{A}_i (1 \leq i \leq c)$ satisfies $(\sum \epsilon_i)$-node (resp. edge) LDP.

Edge-LDP is a relaxation of node-LDP, which limits the definition of neighbors from any two adjacency bit vectors to those that differ only in one bit (i.e., one edge). Nonetheless, edge-LDP can still achieve strong indistinguishability of each edge's existence, which suffices for many graph applications such as social networks while preserving high utility [14]. As such, in this paper we assume edge-LDP as with all existing graph LDP works.

3 LF-GDPR: FRAMEWORK OVERVIEW

In this section, we first introduce the rationale behind LF-GDPR for privacy-preserving graph analytics and then overview its workflow. Finally, we introduce two use cases of LF-GDPR.

3.1 Design Principle

The core of privacy-preserving graph analytics often involves **estimating some target graph metric** without accessing the original graph. Under the DP/LDP privacy model, there are two solution paradigms, namely, generating a synthetic graph to calculate this metric [11], [16], [17], [18], [19] and designing a dedicated DP/LDP solution for such metric [7], [14], [20], [21], [22]. The former provides a general solution but suffers from low estimation accuracy as **the neighborhood information in the original graph is**

TABLE 1
Popular graph analysis tasks and metrics

Graph Analysis Task	Graph Metric Concerned	Derivation from $\boldsymbol{B}$, $\boldsymbol{M}$, and $\boldsymbol{D}$
synthetic graph generation	clustering coefficient	$cc_i = \frac{\boldsymbol{M}^3_{ii}}{d_i(d_i-1)}$
community detection, graph clustering	modularity	$Q_c = \frac{\|\boldsymbol{M}_c\|}{\|\boldsymbol{D}\|} - \frac{\|\boldsymbol{D}_c\|^2}{\|\boldsymbol{D}\|^2}$
node role, page rank	degree centrality	$c_i = d_i$
	eigenvector centrality	$c_i = \boldsymbol{B}_i\boldsymbol{M}^k$
connectivity analysis (clique / hub)	structural similarity	$\tau(i,j) = \frac{\|\boldsymbol{B}_i \cap \boldsymbol{B}_j\|}{\sqrt{d_i d_j}}$
node similarity search	cosine similarity	$\tau(i,j) = \frac{\boldsymbol{B}_i \boldsymbol{B}'_j}{\sqrt{d_i d_j}}$

missing from the synthetic graph. The latter can achieve higher estimation accuracy but cannot generalize such a dedicated solution to other problems — it works poorly or even no longer works if the target graph metric or graph type (e.g., undirected graph, attributed graph, and DAG) is changed [8], [18].

LF-GDPR is our answer to both solution generality and estimation accuracy under the LDP model. It collects from each node i two atomic graph metrics that can derive a wide range of common metrics. The first is the **adjacency bit vector** $\boldsymbol{B}$, where each element j is 1 only if j is a neighbor of i. $\boldsymbol{B}$ of all nodes collectively constitutes the adjacency matrix $\boldsymbol{M}$ of the graph. The second metric is **node degree vector** $\boldsymbol{D} = \{d_1, d_2, ..., d_n\}$, which is frequently used in graph analytics to measure the density of connectivity [21]. Table 1 lists some of the most popular graph analysis tasks in the literature [23], [24], [25] and their graph metrics, all of which can be derived from $\boldsymbol{B}$, $\boldsymbol{M}$, and $\boldsymbol{D}$.

Intuitively, for each node, d can be estimated from $\boldsymbol{B}$. However, given a large graph and limited privacy budget, the estimation accuracy could be too noisy to be meaningful. To illustrate this, let us assume each bit of the adjacency bit vector $\boldsymbol{B}$ is perturbed independently by the classic Randomized Response (RR) [26] algorithm with privacy budget ϵ. As stated in [26], the variance of the estimated node degree $\widetilde{d}$ is

$$Var[\widetilde{d}] = n \cdot \Big[\frac{1}{16(\frac{e^\epsilon}{e^\epsilon+1} - \frac{1}{2})^2} - (\frac{d}{n} - \frac{1}{2})^2\Big] \qquad (1)$$

Even for a moderate social graph with extremely large privacy budget, for example, $d = 100$, $n = 1M$, and $\epsilon = 8$ (the largest ϵ used in [11] is 7), $Var[\widetilde{d}] \approx 435 > 4d$, which means the variance of the estimated degree is over 4 times that of the degree itself. As such, we choose to spend some privacy budget on an independently perturbed degree. This further motivates us to design an optimal privacy budget allocation between adjacency bit vector $\boldsymbol{B}$ and node degree d, to minimize the distance between the target graph metric and the estimated one.

To summarize, in LF-GDPR each node sends two perturbed atomic metrics, namely, the adjacency bit vector $\widetilde{\boldsymbol{B}}$ (perturbed from $\boldsymbol{B}$) and node degree $\widetilde{d}$ (perturbed from d), to the data collector, who then aggregates them to estimate the target graph metric.

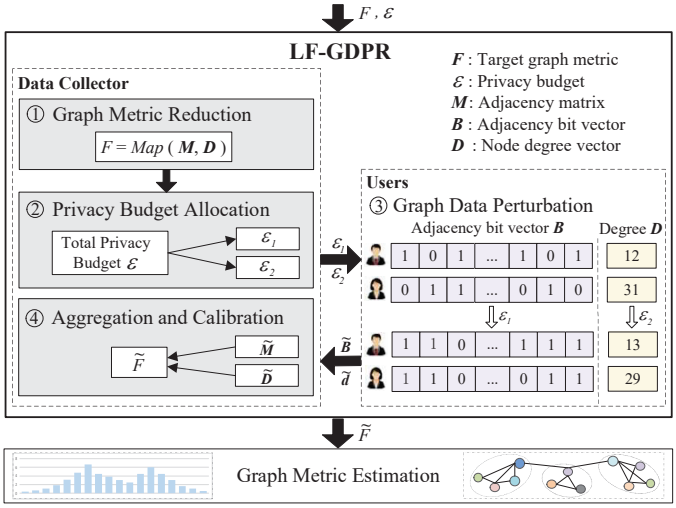


Fig. 1. An overview of LF-GDPR

3.2 LF-GDPR Overview

LF-GDPR works as shown in Fig. 1. A data collector who wishes to estimate a target graph metric F first reduces it from the adjacency matrix $\boldsymbol{M}$ and node degree vector $\boldsymbol{D}$ of all nodes by deriving a mapping function $F = Map(\boldsymbol{M}, \boldsymbol{D})$ (step ①). Based on this reduction, LF-GDPR optimally allocates the total privacy budget ϵ between $\boldsymbol{M}$ and $\boldsymbol{D}$, denoted by ϵ_1 and ϵ_2, respectively (step ②). Then each node locally perturbs its adjacency bit vector $\boldsymbol{B}$ into $\widetilde{\boldsymbol{B}}$ to satisfy ϵ_1-edge LDP, and perturbs its node degree d into $\widetilde{d}$ to satisfy ϵ_2-edge LDP (step ③). According to the composability of LDP, each node then satisfies ϵ-edge LDP. Note that this step is challenging as both $\boldsymbol{B}$ and d are correlated among nodes. For $\boldsymbol{B}$, the j-th bit of node i's adjacency bit vector is the same as the i-th bit of node j's adjacency bit vector. For d, whether i and j has an edge affects both degrees of i and j. Sections 4.2 and 4.3 solve this issue and send out the perturbed $\boldsymbol{B}$ and d, i.e., $\widetilde{\boldsymbol{B}}$ and $\widetilde{d}$. The data collector receives them from all nodes, aggregates them according to the mapping function $Map(\cdot)$ to obtain the estimated target metric $\widetilde{F}$, and further calibrates it to suppress estimation bias and improve accuracy (step ④). The resulted $\widetilde{F}$ is then used for graph analysis. The detailed implementation of LF-GDPR for steps ①②③④ will be presented in Section 4. Note that the algorithms in steps ①②④ are parameterized, which can only be determined when the target graph metric F is specified.

Example 3.2. **LF-GDPR against Facebook Privacy Scandal.** Facebook API essentially controls how a third-party app accesses the data of each individual user. To limit the access right of an average app (e.g., the one developed by Cambridge Analytica) while still supporting graph analytics, Facebook API should have a new permission rule that only allows such app to access the perturbed adjacency bit vector and degree of a user's friends list under ϵ_1 and ϵ_2-edge LDP, respectively. In the Cambridge Analytica case, the app is a personality test, so the app developer may choose structural similarity as the target graph metric and use the estimated value for the personality test. To estimate structural similarity, the app then implements steps ①②④

of LF-GDPR. On the user side, each user u has a privacy budget ϵ_u for her friends list. If $\epsilon_u \geq \epsilon_1 + \epsilon_2$, the user can grant access to this app for perturbed adjacency bit vector and degree; otherwise, the user simply ignores this access request.

3.3 Two Cases of Graph Analytics Using LF-GDPR

To illustrate LF-GDPR, we show two use cases throughout this paper. In this subsection, we introduce their background and target graph metrics F. Their usage details, including the reduction of F (step ①), the optimal privacy budget allocation (step ②), and the aggregation and calibration (step ④), are presented in Sections 5 and 6 respectively.

3.3.1 Clustering Coefficient Estimation

The clustering coefficient of a node measures the connectivity in its *neighborhood*, i.e., the subgraph of its neighbors. Formally, the clustering coefficient cc_i of node i is defined as

$$cc_i = \frac{2t_i}{d_i(d_i-1)},$$

where t_i denotes the number of edges in the neighborhood of node i, or equivalently, the number of triangles incident to node i. A clustering coefficient is in the range of $[0, 1]$, and a high value indicates its neighbors tend to directly connect to each other. It is an important measure of graph structure, and is widely used in graph analytics. For example, the graph model BTER [11], [27] needs clustering coefficient (as well as node degree) to generate a synthetic graph. As it depends on the neighborhood information and thus cannot be calculated locally in each node, existing LDP techniques for values, such as [28], [29], [30], cannot work. The detailed solution by LF-GDPR will be shown in Section 5.

3.3.2 Modularity Estimation and Community Detection

Communities (i.e., densely connected subgraphs) are commonly used in graph analytics to understand the underlying structure of a graph. The criterion of a good community is similar to a graph partition — with many intra-community edges and only a few inter-community edges. Many popular community detection methods are based on modularity maximization [31], which iteratively improves modularity, a widely-adopted metric to measure the quality of detected communities. Formally, the modularity Q of a graph is defined as the sum of individual modularities q_c of all communities $\mathcal{C}$:

$$Q = \sum_{c=1}^{r} q_c = \sum_{c=1}^{r} \left[\frac{L_c}{L} - (\frac{K_c}{2L})^2 \right], \tag{2}$$

where r is the number of communities in the graph, L is the total number of edges, L_c is the total number of edges in community $\mathcal{C}$, and K_c is the total degree of all nodes in $\mathcal{C}$. Q is in the range of $[-1, 1]$, where a higher value is more desirable. As with clustering coefficient, neither individual nor overall modularity can be estimated by dedicated LDP techniques which do not send the adjacency bit vectors. Section 6 will elaborate on how to use LF-GDPR to estimate it.

4 LF-GDPR: Implementation

In this section, we present the implementation details of LF-GDPR. We first discuss graph metric reduction (step ①), followed by the perturbation protocols for adjacency bit vector and node degree, respectively (step ③). Then we elaborate on the aggregation and calibration algorithm (step ④). Finally, we present the optimal allocation of privacy budget between adjacency bit vector and node degree (step ②).

4.1 Graph Metric Reduction

The reduction outputs a polynomial mapping function $Map(\cdot)$ from the target graph metric F to the adjacency matrix $\boldsymbol{M} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, ..., \boldsymbol{B}_n\}$ and degree vector $\boldsymbol{D} = \{d_1, d_2, ..., d_n\}$, i.e., $F = Map(\boldsymbol{M}, \boldsymbol{D})$. Without loss of generality, we assume F is a polynomial of $\boldsymbol{M}$ and $\boldsymbol{D}$. That is, F is a sum of terms F_l, each of which is a multiple of $\boldsymbol{M}$ and $\boldsymbol{D}$ of some exponents. Since F and F_l are scalars, in each term F_l, we need functions f and g to transform $\boldsymbol{M}$ and $\boldsymbol{D}$ with exponents to scalars, respectively. Formally,

$$F = \sum\nolimits_l F_l = \sum\nolimits_l f_{\phi_l}(\boldsymbol{M}^{k_l}) \cdot g_{\psi_l}(\boldsymbol{D}), \tag{3}$$

where $\boldsymbol{M}^{k_l}$ is the k_l-th power of adjacency matrix $\boldsymbol{M}$ whose cell (i, j) denotes the number of paths between node i and j of length k_l, ϕ_l projects a matrix to a cell, a row, a column or a sub-matrix, and $f_{\phi_l}(\cdot)$ denotes an aggregation function f (e.g., sum) after projection ϕ_l. Likewise, ψ_l projects a vector to a scalar or a sub-vector, and $g_{\psi_l}(\cdot)$ denotes an aggregation function g after ψ_l.

As such, the metric reduction step is to determine k_l, $f_{\phi_l}(\cdot)$, and $g_{\psi_l}(\cdot)$ for each term F_l in Eq. 3.

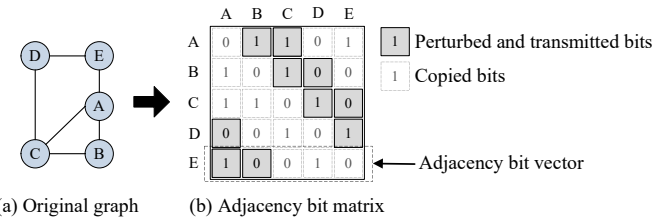
4.2 Adjacency Bit Vector Perturbation

An intuitive approach, known as *Randomized Neighbor List* (*RNL*) [11], perturbs each bit of the vector independently by the classic Randomized Response (RR) [26]. Formally, given an adjacency bit vector $\boldsymbol{B} = \{b_1, b_2, ..., b_n\}$, and privacy budget ϵ_1, the perturbed vector $\widetilde{\boldsymbol{B}} = \{\widetilde{b}_1, \widetilde{b}_2, ..., \widetilde{b}_n\}$ is obtained as follows:

$$\widetilde{b}_i = \begin{cases} b_i & \text{w.p. } \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}} \\ 1 - b_i & \text{w.p. } \frac{1}{1+e^{\epsilon_1}} \end{cases} \tag{4}$$

Note that here basic RR rather than OUE [32] is adopted. This is because adjacency bit vector is a binary vector, and according to [33], RR can achieve better accuracy than OUE.

Note that in Eq. 4, the probability of preserving an edge (bit '1') or non-edge (bit '0'), i.e., $p = \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}}$, is not proportional to the amount of edge information disclosed to the collector. In fact, the success rate of the collector inferring an observed edge is a true edge is $\frac{\gamma p}{\gamma p + (1-\gamma)(1-p)}$, where γ is the edge density in a graph. Although the edge density γ is not considered in the definition of edge LDP, but it contributes to the posterior probability for the collector to infer the truth from an observed edge or non-edge. As such, a high edge density γ also plays an important role in raising the success rate. But it is normally very small in social networks, and furthermore, such statics are generally not precisely owned by the collector.



(a) Original graph (b) Adjacency bit matrix

Fig. 2. Illustration of $RABV$ protocol

RNL is proved to satisfy ϵ_1-edge LDP for each user. However, **for undirected graphs, *RNL* can only achieve $2\epsilon_1$-edge LDP for the collector**, because the data collector witnesses the same edge perturbed twice and independently. Let $\widetilde{\boldsymbol{M}} = \{\widetilde{\boldsymbol{B}}_1, \widetilde{\boldsymbol{B}}_2, ..., \widetilde{\boldsymbol{B}}_n\}$ denote the perturbed adjacency matrix. The edge between nodes i and j appears in both $\widetilde{M}_{ij}$ and $\widetilde{M}_{ji}$, each perturbed with privacy budget ϵ_1. Then according to the theorem of composability, *RNL* becomes a $2\epsilon_1$-edge LDP algorithm for an undirected graph, which is less private. A formal proof is as follows.

For the original adjacency matrix $\boldsymbol{M}$ of an undirected graph, $M_{ij} = M_{ji}$ always holds for any two nodes i and j. By observing two perturbed bits $\widetilde{M}_{ij}$ and $\widetilde{M}_{ji}$ in the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$, the posterior probability that there exists an edge between nodes i and j can be denoted by $\Pr[M_{ij} = M_{ji} = 1 \mid \widetilde{M}_{ij}, \widetilde{M}_{ji}]$. Further, we have

$$\begin{aligned}
&\frac{\Pr[M_{ij} = M_{ji} = 1 \mid \widetilde{M}_{ij}, \widetilde{M}_{ji}]}{\Pr[M_{ij} = M_{ji} = 0 \mid \widetilde{M}_{ij}, \widetilde{M}_{ji}]} \\
&\leq \frac{\Pr[M_{ij} = M_{ji} = 1 \mid \widetilde{M}_{ij} = \widetilde{M}_{ji} = 1]}{\Pr[M_{ij} = M_{ji} = 0 \mid \widetilde{M}_{ij} = \widetilde{M}_{ji} = 1]} \\
&= \frac{\Pr[M_{ij} = 1 \mid \widetilde{M}_{ij} = 1] \cdot \Pr[M_{ji} = 1 \mid \widetilde{M}_{ji} = 1]}{\Pr[M_{ij} = 0 \mid \widetilde{M}_{ij} = 1] \cdot \Pr[M_{ji} = 0 \mid \widetilde{M}_{ji} = 1]} \\
&= \frac{\frac{e^{\epsilon_1}}{1+e^{\epsilon_1}} \cdot \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}}}{\frac{1}{1+e^{\epsilon_1}} \cdot \frac{1}{1+e^{\epsilon_1}}} = e^{2\epsilon_1},
\end{aligned}$$

which proves that *RNL* only provides $2\epsilon_1$-edge LDP.

Furthermore, *RNL* requires each user to perturb and send all n bits in the adjacency bit vector to data collector, which incurs a high computation and communication cost.

To address the problems of *RNL*, we propose a more private and efficient protocol *Randomized Adjacency Bit Vector* ($RABV$) to perturb edges in undirected graphs. As shown in Fig. 2(b), the adjacency matrix is composed of n rows, each corresponding to the adjacency bit vector of a node. For the first $1 \leq i \leq \lfloor \frac{n}{2} \rfloor$ nodes, $RABV$ uses RR as in Eq.4 to perturb and transmit $t = \lfloor \frac{n}{2} \rfloor$ bits (i.e., bits in grey) — from the $(i+1)$-th bit to the $(i+1+t \mod n)$-th bit; for the rest nodes, $RABV$ uses RR to perturb and transmit $t = \lfloor \frac{n-1}{2} \rfloor$ bits in the same way. In essence, $RABV$ **perturbs one and only one bit** for each pair of symmetric bits in the adjacency matrix. The data collector can then obtain the whole matrix by copying bits in grey to their symmetric positions.

Following the same proof of RNL, $RABV$ is guaranteed to satisfy ϵ_1-edge LDP for the collector. Further, since each node only perturbs and transmits about half of the bits in an adjacency bit vector, $RABV$ significantly reduces computation and communication cost of *RNL*.

4.3 Node Degree Perturbation

Releasing the degree of a node while satisfying edge ϵ-LDP is essentially a centralized DP problem because all edges incident to this node, or equivalently, all bits in its adjacency bit vector, form a database and the degree is a count function. In the literature, *Laplace Mechanism* [12] is the predominant technique to perturb numerical function values such as counts. As such, LF-GDPR adopts it to perturb the degree d_i of each node i. According to the definition of edge LDP, two adjacency bit vectors $\boldsymbol{B}$ and $\boldsymbol{B}'$ are two neighboring databases if they differ in only one bit. As such, the sensitivity of degree (i.e., count function) is 1, and therefore adding Laplace noise $Lap(\frac{1}{\epsilon_2})$ to the node degree can satisfy ϵ_2-LDP. That is, $\widetilde{d}_i = d_i + Lap(\frac{1}{\epsilon_2})$.

Similar to perturbing adjacency bit vector, however, in the above naive approach the data collector witnesses two node degrees d_i and d_j perturbed independently, but they share the same edge between i and j. As such, whether this edge exists or not contributes to both d_i and d_j. In the most extreme case where there are only two nodes and one edge in the graph, $d_1 = 1$ and $d_2 = 1$, both of which indicate the existence of this edge. If it is removed, both d_1 and d_2 will decrease by 1, causing the sensitivity of node degree perturbation to be 2. As DP or LDP does not refrain an adversary from possessing any background knowledge, in the worst case the collector already knows all edges except for this one. As such, witnessing the two node degrees d_i and d_j is degenerated to witnessing the edge between i and j twice and independently.

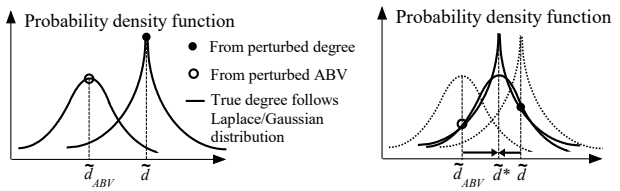
Unfortunately, the remedy that works for perturbing adjacency bit vector cannot be adopted here, as direct bit copy is not feasible for degree. As such, we take an alternative approach to increase the Laplace noise. The following theorem proves that if we add Laplace noise $Lap(\frac{2}{\epsilon_2})$ to every node degree, ϵ_2-LDP can be satisfied for the collector.

Theorem 4.1. A perturbation algorithm $\mathcal{A}$ satisfies ϵ_2-LDP for the collector if it adds Laplace noise $Lap(\frac{2}{\epsilon_2})$ to every node degree d_i, i.e., $\widetilde{d}_i = \mathcal{A}(d_i) = d_i + Lap(\frac{2}{\epsilon_2})$.

PROOF. By adding Laplace noise $Lap(\frac{2}{\epsilon_2})$ to any node degree d_i, i.e., $\widetilde{d}_i = d_i + Lap(\frac{2}{\epsilon_2})$, the perturbation algorithm $\mathcal{A}$ satisfies $\frac{\epsilon_2}{2}$-LDP for node i. For the collector, whether there is an edge between any two nodes i and j can be derived from both perturbed degrees $\widetilde{d}_i$ and $\widetilde{d}_j$. Then according to the composability property of Theorem 2.3, the perturbation algorithm $\mathcal{A}$ satisfies ϵ_2-LDP for the collector. □

The perturbed degree $\widetilde{d}$ is a coarse estimation of the true degree. Now that we have both $\widetilde{d}$ and $\widetilde{d}_{ABV}$, the degree estimated from the perturbed adjacency bit vector $\widetilde{\boldsymbol{B}}$,[1] we can use *Maximum Likelihood Estimation* (MLE) [34] to obtain a refined estimation $\widetilde{d}^*$. The rationale of this refinement is illustrated in Fig. 3. Before refinement (Fig. 3(a)), as each bit of $\boldsymbol{B}$ follows Bernoulli distribution, according to De Moivre-Laplace Central Limit Theorem, the probability density function of $\widetilde{d}_{ABV}$ can be approximated by a Gaussian

1. A naive and biased estimation is $\widetilde{d}_{ABV} = \sum_{j=1}^{n} \widetilde{b}_j$. In Example 4.4, we show a calibrated and unbiased estimation $\widetilde{d}_{ABV} = \frac{\sum_{j=1}^{n} \widetilde{b}_j}{2p-1} + \frac{(p-1)n}{2p-1}$, where $p = \frac{e^{\epsilon_1}}{e^{\epsilon_1}+1}$.



(a) Before refinement (b) After refinement

Fig. 3. Refining $\widetilde{d}$ to $\widetilde{d}^*$ by MLE

distribution $f_1(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\widetilde{d}^*)^2}{2\sigma^2}}$, where the variance $\sigma^2 = n \cdot [\frac{1}{16(p-\frac{1}{2})^2} - (\frac{d}{n} - \frac{1}{2})^2]$ is derived in Eq. 1.[2] On the other hand, as $\widetilde{d}$ is obtained by adding Laplace noise to d, the probability density function of $\widetilde{d}$ follows a Laplace distribution $f_2(x) = \frac{\epsilon_2}{4} e^{-\frac{|x-\widetilde{d}^*|\epsilon_2}{2}}$. The refinement, as shown in Fig. 3(b), shifts both distributions to share the same mean, i.e., the true degree, as they are both drawn from it. To estimate this mean $\widetilde{d}^*$ by MLE, we derive the joint likelihood of observing both $\widetilde{d}$ and $\widetilde{d}_{ABV}$, and maximize it. Since they are both independently perturbed, the joint likelihood is the multiplication of individual probabilities. Formally,

$$\begin{aligned}\widetilde{d}^* &= \arg\max_{\widetilde{d}^*} f_1(\widetilde{d}_{ABV}) \cdot f_2(\widetilde{d}) \\ &= \arg\max_{\widetilde{d}^*} \frac{\epsilon_2}{\sigma \cdot 4\sqrt{2\pi}} e^{-\frac{(\widetilde{d}_{ABV}-\widetilde{d}^*)^2+\sigma^2|\widetilde{d}-\widetilde{d}^*|\epsilon_2}{2\sigma^2}} \\ &\approx \arg\min_{\widetilde{d}^*} \left((\widetilde{d}_{ABV} - \widetilde{d}^*)^2 + \sigma^2|\widetilde{d} - \widetilde{d}^*|\epsilon_2\right)\end{aligned}$$

By solving the above equation, we have

$$\widetilde{d}^* = median(\widetilde{d}_{ABV} - \frac{\sigma^2 \cdot \epsilon_2}{2}, \widetilde{d}, \widetilde{d}_{ABV} + \frac{\sigma^2 \cdot \epsilon_2}{2}) \quad (5)$$

4.4 Aggregation and Calibration

Upon receiving the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$ and degree vector $\widetilde{\boldsymbol{D}}$,[3] the data collector can estimate the target graph metric $\widetilde{F}$ by aggregation according to Eq. 3 with a calibration function $\mathcal{R}(\cdot)$:

$$\widetilde{F} = \sum\nolimits_l \mathcal{R}\left(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})\right) \cdot g_{\psi_l}(\widetilde{\boldsymbol{D}}) \quad (6)$$

The calibration function aims to suppress the aggregation bias of $\widetilde{\boldsymbol{M}}$ propagated by f_{ϕ_l}. On the other hand, no calibration is needed for $g_{\psi_l}(\widetilde{\boldsymbol{D}})$ as $\widetilde{\boldsymbol{D}}$ is already an unbiased estimation of $\boldsymbol{D}$, thanks to the Laplace Mechanism.

To derive $\mathcal{R}(\cdot)$, we regard $\mathcal{R}$ as the mapping between $f_{\phi_l}(\boldsymbol{M}^{k_l})$ and $f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})$. In other words, $\mathcal{R}$ estimates $f_{\phi_l}(\boldsymbol{M}^{k_l})$ after observing $f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})$. Formally,

$$\mathcal{R} : f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l}) \to f_{\phi_l}(\boldsymbol{M}^{k_l})$$

The following shows a concrete example for aggregation and calibration when estimating the number of edges in a graph. The result of this example will be used in Section 6 to estimate L_c in Eq. 2 of modularity definition.

2. Here we replace d with $\widetilde{d}$ in Eq. 1 for simplicity.

3. In the sequel, $\widetilde{\boldsymbol{D}}$ denotes the refined degree $\widetilde{\boldsymbol{D}}^*$ to simplify the notation.

Example 4.4. For a graph with n nodes, there are $N = \frac{1}{2}n(n-1)$ bits in its upper/lower triangular matrix, each indicating whether an edge exists or not. Let s denote the number of edges in the original graph, i.e., the number of "1"s in these N bits. These N bits are then perturbed according to $RABV$ protocol by randomized response [26] with flipping probability p. To estimate s, the data collector takes the following two steps.

(1) **Aggregation.** It aggregates the number of "1"s in the perturbed N bits and uses it as an initial estimation $\widetilde{s}$.

(2) **Calibration.** Since the mapping between s and $\widetilde{s}$ can be captured by $\widetilde{s} = sp + (N-s)(1-p)$, the collector then calibrates $\widetilde{s}$ by $\mathcal{R}(\widetilde{s}) = \frac{\widetilde{s}}{2p-1} + \frac{p-1}{2p-1}N$, which is derived by solving the mapping function.

We can further show $\mathcal{R}(\widetilde{s})$ is an unbiased estimation of s, because $\mathbb{E}[\mathcal{R}(\widetilde{s})] = \frac{1}{2p-1}[sp + (N-s)(1-p) + (p-1)N] = s$.

If both $\mathcal{R}(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l}))$ and $g_{\psi_l}(\widetilde{\boldsymbol{D}})$ are unbiased estimation of $f_{\phi_l}(\boldsymbol{M}^{k_l})$ and $g_{\psi_l}(\boldsymbol{D})$ respectively, the following theorem guarantees $\widetilde{F}$ is an unbiased estimation of the target metric F.

Theorem 4.2. If $\mathcal{R}(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l}))$ and $g_{\psi_l}(\widetilde{\boldsymbol{D}})$ are unbiased estimation of $f_{\phi_l}(\boldsymbol{M}^{k_l})$ and $g_{\psi_l}(\boldsymbol{D})$ respectively, the estimated graph metric $\widetilde{F}$ is unbiased.

PROOF. According to the assumption of unbiased estimation, we have

$$\begin{aligned}\mathbb{E}\left[\mathcal{R}\left(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})\right)\right] &= f_{\phi_l}(\boldsymbol{M}^{k_l}) \\ \mathbb{E}\left[g_{\psi_l}(\widetilde{\boldsymbol{D}})\right] &= g_{\psi_l}(\boldsymbol{D})\end{aligned}$$

Since the adjacency bit vector and the degree of each node are perturbed independently, we have

$$\begin{aligned}\mathbb{E}\left[\widetilde{F}\right] &= \sum\nolimits_l \mathbb{E}\left[\mathcal{R}\left(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})\right) \cdot g_{\psi_l}(\widetilde{\boldsymbol{D}})\right] \\ &= \sum\nolimits_l \mathbb{E}\left[\mathcal{R}\left(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})\right)\right] \cdot \mathbb{E}\left[g_{\psi_l}(\widetilde{\boldsymbol{D}})\right] \\ &= \sum\nolimits_l f_{\phi_l}(\boldsymbol{M}^{k_l}) \cdot g_{\psi_l}(\boldsymbol{D}) \\ &= F\end{aligned}$$

Therefore, $\widetilde{F}$ is unbiased. □

4.5 Optimal Privacy Budget Allocation

The final problem in LF-GDPR is to allocate the privacy budget (step ② in Fig. 1). Formally, it divides ϵ into $\epsilon_1 = \alpha\epsilon$ and $\epsilon_2 = (1-\alpha)\epsilon$, where $\alpha \in (0,1)$, for adjacency bit vector and node degree perturbation, respectively.

Our objective is to find the optimal α that minimizes the distance between the graph metric F and our estimation $\widetilde{F}$. Without loss of generality, we adopt the L_2 distance [35] and set the loss function for optimization as the expectation of this distance, i.e., $\alpha = \arg\min_{\alpha\in(0,1)} \mathbb{E}[\|\widetilde{F} - F\|_2^2]$.

Assuming $\widetilde{F}$ is unbiased, we have

$$\begin{aligned}\mathbb{E}[\|\widetilde{F} - F\|_2^2] &= \mathbb{E}[F^2 - 2F\widetilde{F} + \widetilde{F}^2] \\ &= \mathbb{E}[F^2] - 2\mathbb{E}[F] \cdot \mathbb{E}[\widetilde{F}] + \mathbb{E}[\widetilde{F}^2] \\ &= \mathbb{E}[\widetilde{F}^2] - F^2.\end{aligned}$$

Since F^2 is constant, we only need to minimize $\mathbb{E}[\widetilde{F}^2]$ with respect to α:

$$\mathbb{E}[\widetilde{F}^2] = \mathbb{E}\left[\left(\sum\nolimits_l \mathcal{R}\left(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})\right)\cdot g_{\psi_l}(\widetilde{\boldsymbol{D}})\right)^2\right] \tag{7}$$

In the next two sections, we will demonstrate how to derive the terms in Eq. 7 with respect to α. Then we can apply numerical methods, e.g., Newton's method [36], to find α that minimizes Eq. 7. Further, the following theorem shows the accuracy guarantee of LF-GDPR.

Theorem 4.3. For a graph metric F and our estimation $\widetilde{F}$, with at least $1-\beta$ probability, we have

$$|F-\widetilde{F}| = O(\sqrt{\mathbb{E}[\widetilde{F}^2]\cdot log(1/\beta)})$$

PROOF. For a graph metric F, and its estimated one $\widetilde{F}$, the variance of $F-\widetilde{F}$ is

$$\begin{aligned} Var[F-\widetilde{F}] &= Var[\widetilde{F}] = \mathbb{E}[F^2] - (\mathbb{E}[F])^2 \\ &= \mathbb{E}[\widetilde{F}^2] - F^2 \le \mathbb{E}[\widetilde{F}^2] \end{aligned}$$

By Benstein's inequality,

$$\begin{aligned} \Pr[|F-\widetilde{F}|\ge\lambda] &\le 2\cdot\exp\left(-\frac{\lambda^2}{2Var[F-\widetilde{F}]+\frac{2}{3}\lambda}\right) \\ &\le 2\cdot\exp\left(-\frac{\lambda^2}{2\mathbb{E}[\widetilde{F}^2]+\frac{2}{3}\lambda}\right) \end{aligned}$$

By the union bound, there exists $\lambda = O(\sqrt{\mathbb{E}[\widetilde{F}^2]\cdot log(1/\beta)})$ such that $|F-\widetilde{F}|<\lambda$ holds with at least $1-\beta$ probability. □

As will be shown in the next two sections, $\mathbb{E}[\widetilde{F}^2]$ can be further expressed by ϵ, n or d for a specific graph metric.

4.6 Summary

Algorithm 1 summarizes the overall protocol of LF-GDPR. It takes three inputs — the target graph metric F, the privacy budget ϵ, and the true adjacency bit vector $\boldsymbol{B}_i$ of each node i, and returns an estimation of graph metric $\widetilde{F}$ under ϵ-LDP. In Line 1, the data collector reduces F to adjacency matrix and node degree. Based on the reduction, in Line 2 the privacy budget ϵ is divided into $\alpha\epsilon$ and $(1-\alpha)\epsilon$ by the optimal privacy budget allocation algorithm (see Section 4.5 for details), and then α is sent to each node (Line 3). On each node i, $RABV$ perturbs its adjacency bit vector (Lines 5-6, see Section 4.2 for details). For each bit to perturb, it adopts RR with privacy budget $\alpha\epsilon$. Then node i further perturbs its degree d_i by adding a Laplace noise with privacy budget $(1-\alpha)\epsilon$ (Line 7). Finally, the perturbed adjacency bit vector and node degree are sent to the data collector (Line 8). After the collector receives the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$ and degree vector $\widetilde{\boldsymbol{D}}$, it first completes the whole adjacency matrix by copying bits to their symmetric ones in $\widetilde{\boldsymbol{M}}$ (Line 9), and then refines each node degree $\widetilde{d}_i$ to $\widetilde{d}_i^*$ (Line 10, see Section 4.3 for details). Finally, it applies aggregation and calibration to estimate the graph metric $\widetilde{F}$ (Line 11).

Security of Correlation. It is known that the privacy provided by differential privacy decrease significantly under correlations [37], [38]. However, correlation between adjacency bit vectors and node degrees does not compromise LDP in LF-GDPR. First, there is pairwise correlation between the adjacency bit vectors of any two users, but the proposed RABV protocol is able to well address it by avoiding "double dose" of the same edge information. Second, there is correlation between the node degrees of two users who share an edge. But Theorem 4.1 proves that by setting sensitivity to 2 and adding $Lap(\frac{2}{\epsilon_2})$ noise, this correlation does not compromise ϵ_2-LDP. Third, there is correlation between the adjacency bit vector and node degree of the same user. But since we divide the privacy budget between them, according to sequential composition, ϵ-LDP is still achieved even if they have the strongest correlation (i.e., an equivalent or causal value).

Algorithm 1 Overall protocol of LF-GDPR framework

Input: Target graph metric F
Privacy budget ϵ
True adjacency bit vector $\{\boldsymbol{B}_1,...,\boldsymbol{B}_n\}$
Output: An estimation of the graph metric $\widetilde{F}$ under ϵ-LDP
Procedure:
*//**Collector side***
1: Reduce graph metric F to adjacency matrix $\boldsymbol{M}=\{\boldsymbol{B}_1,...,\boldsymbol{B}_n\}$ and degree vector $\boldsymbol{D}$ derived from $\boldsymbol{M}$
2: Calculate α for privacy budget allocation based on F and ϵ
3: Send α to each node
*//**User side***
4: **for** each node $i\in\{1,2,...,n\}$ **do**
5: $t = i\le\lfloor\frac{n}{2}\rfloor\ ?\ \lfloor\frac{n}{2}\rfloor : \lfloor\frac{n-1}{2}\rfloor$
6: **for** each $b_j\in\boldsymbol{B_i}$, where $i+1\le j\le(i+1+t) \mod n$ **do**
Perturb $\widetilde{b}_j = \begin{cases} b_j & \text{w.p. } \frac{e^{\alpha\epsilon}}{1+e^{\alpha\epsilon}} \\ 1-b_j & \text{w.p. } \frac{1}{1+e^{\alpha\epsilon}} \end{cases}$
7: Calculate the degree d_i from $\boldsymbol{B}_i$ and then perturb it as
$$\widetilde{d}_i = d_i + Lap\left(2/((1-\alpha)\epsilon)\right)$$
8: Send $\widetilde{\boldsymbol{B}}_i$ and $\widetilde{d}_i$ to the data collector
*//**Collector side***
9: Copy symmetric bits in $\widetilde{\boldsymbol{M}}=\{\widetilde{\boldsymbol{B}}_1,...,\widetilde{\boldsymbol{B}}_n\}$
10: Refine $\widetilde{d}_i$ to $\widetilde{d}_i^*$ of each node i according to Eq. 5
11: Apply aggregation and calibration to estimate the graph metric $\widetilde{F}$ based on $\widetilde{\boldsymbol{M}}$ and $\widetilde{\boldsymbol{D}}=\{\widetilde{d}_1^*,...,\widetilde{d}_n^*\}$
12: **return** $\widetilde{F}$

5 Clustering Coefficient Estimation with LF-GDPR

In this section, we show how to use LF-GDPR to estimate the clustering coefficients of all nodes in a graph. Based on the implementation framework in Section 4, we present the details of steps ①②④. Finally, Algorithm 2 summarizes the whole process.

5.1 Implementation Details

Graph Metric Reduction (step ① in LF-GDPR). Recall that the clustering coefficient of node i, $cc_i = \frac{2t_i}{d_i(d_i-1)}$, where t_i is the number of triangles incident to i. To count t_i, we set $k_1=3$ so that $\boldsymbol{M}^3$ denotes the number of 3-hop walks for all pairs of nodes. We then set projection ϕ_i to M_{ii}^3, the i-th diagonal element of $\boldsymbol{M}^3$ that denotes the number of 3-hop walks starting and ending at node i.[4] Note that M_{ii}^3 counts

4. The full notion of ϕ_i should be $\phi_{1,i}$. Since there is only one term in the definition of clustering coefficient, we omit the notation 1. The same applies to ψ_i.

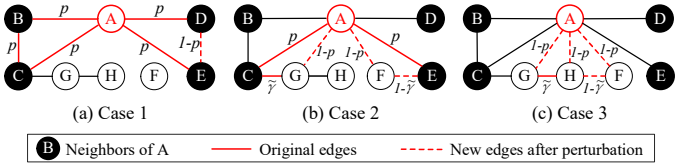


Fig. 4. Estimate number of triangles incident to node A

the triangles incident to node i twice (e.g., triangles ijk and ikj) , so $M_{ii}^3 = 2t_i$, which is exactly the numerator in the above cc_i definition. As such, the aggregation function $f(\cdot)$ can be simply set to an identity function. Formally,

$$f_{\phi_i}(\boldsymbol{M}^3) = f(M_{ii}^3) = M_{ii}^3 = 2t_i$$

To obtain the denominator in the above cc_i definition, we set the projection ψ_i to d_i, the i-th element of degree vector $\boldsymbol{D}$. And the aggregation function $g(\cdot)$ is set according to the denominator in the definition of clustering coefficient:

$$g_{\psi_i}(\boldsymbol{D}) = g(d_i) = \frac{1}{d_i(d_i-1)}$$

To sum up, the clustering coefficient of any node i, denoted by F_i, can be reduced to $\boldsymbol{M}$ and $\boldsymbol{D}$ as

$$F_i = f_{\phi_i}(\boldsymbol{M}^3) \cdot g_{\psi_i}(\boldsymbol{D}) \tag{8}$$

Aggregation and Calibration (step ④ in LF-GDPR). The data collector receives the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$ and degree vector $\widetilde{\boldsymbol{D}}$. According to Eq. 6 and 8, the estimated clustering coefficient of any node i is

$$\widetilde{F}_i = \mathcal{R}\left(f_{\phi_i}(\widetilde{\boldsymbol{M}}^3)\right) \cdot g_{\psi_i}(\widetilde{\boldsymbol{D}}), \tag{9}$$

where the calibration function $\mathcal{R}(\cdot)$ estimates $f_{\phi_i}(\boldsymbol{M}^3)$, the number of triangles incident to node i based on the perturbed number $f_{\phi_i}(\widetilde{\boldsymbol{M}}^3)$. In what follows, we derive $\mathcal{R}(\cdot)$.

According to Section 4.4, to derive $\mathcal{R}(\cdot)$ we need to estimate $f_{\phi_A}(\boldsymbol{M}^3)$, or equivalently $t_A = f_{\phi_A}(\boldsymbol{M}^3)/2$, the number of triangles incident to node A in the original graph. Figs. 4(a)-(c) enumerate all three cases of such triangles based on whether the other two nodes of this triangle are A's neighbors in the original graph. Let d denote its degree and $p = \frac{e^{\alpha\epsilon}}{1+e^{\alpha\epsilon}}$ the perturbation probability. In each case, the edges that constitute such triangles are highlighted by red color. In particular, the red solid lines denote the original edges, and each is retained in the perturbed graph with a probability of p. The red dashed lines denote the new edges after perturbation, and each appears with a probability of $1-p$.

(1) Fig. 4(a): both nodes are neighbors of A. There are two sub-cases based on whether there exists an edge between these two nodes in the original graph. For triangles such as ABC, there is an edge between B and C in the original graph. Such triangles will be retained in the perturbed graph with probability p^3. For triangles such as ADE, there is no edge between D and E in the original graph. Such triangles will be retained in the perturbed graph with probability $p^2(1-p)$. Summing up both sub-cases, the number of such triangles in the perturbed graph is $\widetilde{t}_{A,1} = t_A \cdot p^3 + \left(\frac{1}{2}d(d-1) - t_A\right) \cdot p^2(1-p)$.

(2) Fig. 4(b): only one node is a neighbor of A, for example triangles ACG and AEF. Since d nodes are adjacent to A and $n-d-1$ nodes are not adjacent, there are $d(n-d-1)$ possible triangles. In such a triangle, the two edges incident to A will be retained in the perturbed graph with probabilities $p(1-p)$. The probability of having the third edge (e.g., CG or EF) in the perturbed graph can be approximated by the overall edge density after perturbation, i.e., $\widetilde{\gamma} = \gamma p + (1-\gamma)(1-p)$, where $\gamma = \frac{\sum_{i=1}^{n} d_i}{n(n-1)}$ denote the edge density in the original graph. As such, the number of triangles in this case is $\widetilde{t}_{A,2} = d(n-d-1) \cdot p(1-p)\widetilde{\gamma}$.

(3) Fig. 4(c): neither node is a neighbor of A, for example triangles AGH and AFH. In such a triangle, the two edges incident to A will be retained in the perturbed graph with probabilities $(1-p)^2$. The probability of having the third edge (e.g., GH or FH) in the perturbed graph can also be approximated by $\widetilde{\gamma}$. Since there are $\binom{n-d-1}{2} = \frac{1}{2}(n-d-1)(n-d-2)$ possible triangles, the number of triangles in this case is $\widetilde{t}_{A,3} = \frac{1}{2}(n-d-1)(n-d-2) \cdot (1-p)^2\,\widetilde{\gamma}$.

By summing up $\widetilde{t}_{A,1}$, $\widetilde{t}_{A,2}$, and $\widetilde{t}_{A,3}$, we obtain $\widetilde{t}_A$. Since the calibration function $\mathcal{R}(\cdot)$ maps $\widetilde{t}_A$ to t_A, i.e., $\mathcal{R}(\widetilde{t}_A) = t_A$, we can solve t_A from $\widetilde{t}_A$ and derive $\mathcal{R}(\cdot)$ as [5]

$$\begin{aligned}\mathcal{R}\left(\widetilde{t}_A\right) = \frac{1}{p^2(2p-1)}\Big(\widetilde{t}_A &- \frac{1}{2}\widetilde{d}(\widetilde{d}-1)p^2(1-p) \\ &- \widetilde{d}(n-\widetilde{d}-1)p(1-p)\widetilde{\gamma} \\ &- \frac{1}{2}(n-\widetilde{d}-1)(n-\widetilde{d}-2)(1-p)^2\widetilde{\gamma}\Big)\end{aligned} \tag{10}$$

Privacy Budget Allocation (step ④ in LF-GDPR). According to Section 4.5, to solve α we derive and minimize $\mathbb{E}[\widetilde{F}^2]$ with respect to α in Eq. 7. Theorem 5.1 below shows the closed-form solution of α.

Theorem 5.1. The optimal α for clustering coefficient estimation can be approximated by:

$$\arg\min_{\alpha\in(0,1)} \frac{e^{\alpha\epsilon}+2}{e^{3\alpha\epsilon}(e^{\alpha\epsilon}-1)^2}\left(1+\frac{8(10\hat{d}^2-10\hat{d}+3)}{\hat{d}^2(\hat{d}-1)^2(1-\alpha)^2\epsilon^2}\right) \tag{11}$$

where $\hat{d}$ is a representative degree (e.g., the mean, median, or most frequent degree) of all nodes in the original graph.

PROOF. According to Eq. 7, we have

$$\begin{aligned}\mathbb{E}[\widetilde{F}^2] &= \mathbb{E}\left[\left(\sum\nolimits_l \mathcal{R}\left(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})\right) \cdot g_{\psi_l}(\widetilde{\boldsymbol{D}})\right)^2\right] \\ &= \left(f_\phi^2(\boldsymbol{M}^3) + Var\left[\mathcal{R}\left(f_\phi(\widetilde{\boldsymbol{M}}^3)\right)\right]\right) \cdot \mathbb{E}\left[g_\psi^2(\widetilde{\boldsymbol{D}})\right]\end{aligned}$$

For each node i, by setting

$$f_{\phi_i}(\boldsymbol{M}^3) = 2t_i \quad \text{and} \quad g_{\psi_i}(\boldsymbol{D}) = \frac{1}{d_i(d_i-1)},$$

we approximate $\mathbb{E}[\widetilde{F_i}^2]$ by:

$$\mathbb{E}[\widetilde{F_i}^2] = 4(t_i^2 + Var[\mathcal{R}(\widetilde{t}_i)]) \cdot \mathbb{E}\left[\frac{1}{\widetilde{d}_i^2(\widetilde{d}_i-1)^2}\right], \tag{12}$$

5. We replace d with $\widetilde{d}$, because the former is unknown to data collector and the latter is an unbiased estimation of the former.

Algorithm 2 Collector-side clustering coefficient estimation

Input: Perturbed adjacency matrix $\widetilde{\boldsymbol{M}} = \{\widetilde{\boldsymbol{B}}_1, ..., \widetilde{\boldsymbol{B}}_n\}$
Perturbed degree vector $\widetilde{\boldsymbol{D}} = \{\widetilde{d}_1, ..., \widetilde{d}_n\}$
Percentage α for privacy budget allocation
Output: Estimated clustering coefficient $\boldsymbol{cc} = \{cc_1, ..., cc_n\}$
Procedure:

1: Calculate the edge density in perturbed graph $\widetilde{\gamma} = \frac{\sum_{i=1}^{n} \widetilde{d}_i}{n(n-1)}$
2: **for** each node $i \in \{1, 2, ..., n\}$ **do**
3: Calculate the number of triangles $\widetilde{t}_i$ incident to node i
4: Calibrate $\widetilde{t}_i$ to get an unbiased one t_i according to Eq. 10, where $p = \frac{e^{\alpha\epsilon}}{1+e^{\alpha\epsilon}}$.
5: Estimate node i's clustering coefficient $cc_i = \frac{2t_i}{\widetilde{d}_i(\widetilde{d}_i-1)}$
6: **return** $\boldsymbol{cc} = \{cc_1, ..., cc_n\}$

where

$$\begin{aligned} Var\left[\mathcal{R}\left(\widetilde{t}_i\right)\right] &= \frac{Var[\widetilde{t}_i]}{p^4(2p-1)^2} \\ &= \frac{\frac{1}{2}(n-1)(n-2)Var\left[\widetilde{M}_{it_1}\widetilde{M}_{t_1t_2}\widetilde{M}_{t_2i}\right]}{p^4(2p-1)^2} \\ &\approx \frac{(n-d_i-1)^2(n-d_i-2)^2}{2(n-1)(n-2)} \cdot \frac{e^{\alpha\epsilon}+2}{e^{3\alpha\epsilon}(e^{\alpha\epsilon}-1)^2} \end{aligned} \tag{13}$$

Since $t_i^2 \ll O(n^2) \sim Var[\mathcal{R}(\widetilde{t}_i)]$ for most cases, we omit the term of t_i^2. As for $\mathbb{E}\left[\frac{1}{\widetilde{d}_i^2(\widetilde{d}_i-1)^2}\right]$, by Taylor expansion at $\mathbb{E}[\widetilde{d}_i]$, we have

$$\mathbb{E}\left[\frac{1}{\widetilde{d}_i^2(\widetilde{d}_i-1)^2}\right] \approx \frac{1}{d_i^2(d_i-1)^2} + \frac{8(10d_i^2-10d_i+3)}{d_i^4(d_i-1)^4(1-\alpha)^2\epsilon^2} \tag{14}$$

By substituting Eq. 13 and Eq. 14 into Eq. 12, we have

$$\begin{aligned} \mathbb{E}[\widetilde{F}_i^2] =& \frac{4(n-d_i-1)^2(n-d_i-2)^2}{2(n-1)(n-2)d_i^2(d_i-1)^2} \\ & \cdot \frac{e^{\alpha\epsilon}+2}{e^{3\alpha\epsilon}(e^{\alpha\epsilon}-1)^2}\left(1 + \frac{8(10d_i^2-10d_i+3)}{d_i^2(d_i-1)^2(1-\alpha)^2\epsilon^2}\right) \end{aligned}$$

To minimize $\mathbb{E}[\widetilde{F}_i^2]$, we omit the first item which is independent of α. To unify α for all nodes, we replace d_i with $\hat{d}$, a representative degree (e.g., the mean, median, or most frequent degree) of all nodes in the original graph. Therefore, we can derive α as

$$\arg\min_{\alpha\in(0,1)} \quad \frac{e^{\alpha\epsilon}+2}{e^{3\alpha\epsilon}(e^{\alpha\epsilon}-1)^2}\left(1 + \frac{8(10\hat{d}^2-10\hat{d}+3)}{\hat{d}^2(\hat{d}-1)^2(1-\alpha)^2\epsilon^2}\right)$$

□

As for the representative degree $\hat{d}$, it can be estimated by a portion of privacy budget. The data collector can ask each node to consume some of its privacy budgets for a preliminary round of node degree perturbation and send back $\widetilde{\boldsymbol{D}}$ to estimate $\hat{d}$.

5.2 Overall Algorithm

Algorithm 2 summarizes how the data collector estimates the clustering coefficients of all nodes, based on the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$ and degree vector $\widetilde{\boldsymbol{d}}$. It first computes $\widetilde{\gamma}$, the edge density in the perturbed graph from $\widetilde{\boldsymbol{d}}$ (Line 1). Then for each node the collector calculates the number of triangles incident to it (Line 3) and then further calibrates this number based on Eq. 10 (Line 4). Finally, its clustering coefficient is estimated based on Eq. 9 (Line 5).

Accuracy Guarantee. According to Theorem 5.1, with at least $1-\beta$ probability, the error of clustering coefficient estimation is bounded by $O(\frac{\sqrt{log(1/\beta)}}{d\cdot\epsilon})$.

6 Community Detection with LF-GDPR

In this section, we show how to use LF-GDPR to estimate the modularity of any community in the graph, with only a single round of $\widetilde{\boldsymbol{B}}$ and $\widetilde{\boldsymbol{D}}$ collection. Based on the implementation framework in Section 4, we present the details of steps ①②④. Finally, Algorithms 3 summarizes the process of modularity estimation, which serves for further community detection.

6.1 Implementation Details

Graph Metric Reduction (step ① in LF-GDPR). Recall in Eq. 2, the modularity of a community $\mathcal{C}$ is $q_c = \frac{L_c}{L} - \frac{K_c^2}{4L^2}$, where L_c is the number of edges in $\mathcal{C}$, K_c is the total degree of all nodes in $\mathcal{C}$, and L is the total number of edges in the whole graph. As such, we can write the graph metric $F_c = q_c$ in the form of Eq. 3 as:

$$F_c = q_c = f_{\phi_{1,c}}(\boldsymbol{M}) \cdot g_{\psi_{1,c}}(\boldsymbol{D}) - g_{\psi_{2,c}}(\boldsymbol{D}) \tag{15}$$

There are two terms in the above equation. In the first term, $\phi_{1,c}$ projects graph G to community $\mathcal{C}$, i.e., a sub-matrix $\boldsymbol{M}_c$ of nodes in $\mathcal{C}$ only, and $f_{\phi_{1,c}}(\boldsymbol{M}) = \frac{1}{2}\|\boldsymbol{M}_c\|$, half of the summation of all elements in $\boldsymbol{M}_c$. As such, $f_{\phi_{1,c}}(\boldsymbol{M}) = L_c$. Similarly, $g_{\psi_{1,c}}(\boldsymbol{D}) = \frac{1}{L} = \frac{2}{\|\boldsymbol{D}\|}$. The second term does not involve $\boldsymbol{M}$, so we set $f_{\phi_{2,c}}(\boldsymbol{M}) = -1$. To project graph G to community $\mathcal{C}$, we set $\psi_{2,c}$ to a sub-vector $\boldsymbol{D}_c$ of nodes in $\mathcal{C}$ only, and then $g_{\psi_{2,c}}(\boldsymbol{D}) = \frac{K_c^2}{4L^2} = \frac{\|\boldsymbol{D}_c\|^2}{\|\boldsymbol{D}\|^2}$.

Aggregation and Calibration (step ② in LF-GDPR). According to Eqs. 6 and 15, the data collector estimates the modularity $\widetilde{F}$ based on the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$ and degree vector $\widetilde{\boldsymbol{D}}$ as follows.

$$\widetilde{F} = \mathcal{R}\left(f_{\phi_{1,c}}(\widetilde{\boldsymbol{M}})\right) \cdot g_{\psi_{1,c}}(\widetilde{\boldsymbol{D}}) - g_{\psi_{2,c}}(\widetilde{\boldsymbol{D}})$$

Note that only the first term needs calibration $\mathcal{R}(\cdot)$ as the second term does not involve $\widetilde{\boldsymbol{M}}$. To derive $\mathcal{R}(\cdot)$, we estimate $f_{\phi_{1,c}}(\boldsymbol{M})$ from $f_{\phi_{1,c}}(\widetilde{\boldsymbol{M}})$ based on the $RABV$ algorithm and the fact that $f_{\phi_{1,c}}(\boldsymbol{M}) = \frac{1}{2}\|\boldsymbol{M}_c\| = L_c$. Example 4.4 shows the derivation of this estimation. By solving $f_{\phi_{1,c}}(\boldsymbol{M})$ in terms of $f_{\phi_{1,c}}(\widetilde{\boldsymbol{M}})$, we can derive $\mathcal{R}(\cdot)$ as

$$\mathcal{R}\left(f_{\phi_{1,c}}(\widetilde{\boldsymbol{M}})\right) = \frac{f_{\phi_{1,c}}(\widetilde{\boldsymbol{M}})}{2p-1} + \frac{1}{2}n_c(n_c-1)\frac{p-1}{2p-1}, \tag{16}$$

where $n_c = |\mathcal{C}|$ denotes the number of nodes in $\mathcal{C}$.

Privacy Budget Allocation (step ④ in LF-GDPR). Similar to clustering coefficient estimation, we derive and minimize $\mathbb{E}[\widetilde{F}^2]$ with respect to α in Eq. 7. Theorem 6.1 below shows the closed-form solution of α.

Theorem 6.1. The optimal α for modularity estimation can be approximated by:

$$\arg\min_{\alpha\in(0,1)} \frac{(1-\alpha)^2\epsilon^2L^2+6n^2}{(1-\alpha)^2\epsilon^2L^4}\left(\frac{1}{16(\frac{e^{\alpha\epsilon}}{1+e^{\alpha\epsilon}}-\frac{1}{2})^2} - (\frac{2L}{n(n-1)}-\frac{1}{2})^2\right)$$

PROOF. According to Eq. 7 and Eq. 2, we have

$$\begin{aligned}\mathbb{E}[\widetilde{F}^2] &= \mathbb{E}\left[\left(\sum\nolimits_l \mathcal{R}\left(f_{\phi l}(\widetilde{\boldsymbol{M}}^{k_l})\right)\cdot g_{\psi_l}(\widetilde{\boldsymbol{D}})\right)^2\right] \\ &= \left(f^2_{\phi_1}(\boldsymbol{M}) + Var\left[\mathcal{R}\left(f_{\phi_1}(\widetilde{\boldsymbol{M}})\right)\right]\right)\cdot \mathbb{E}\left[g^2_{\psi_1}(\widetilde{\boldsymbol{D}})\right] \\ &\quad + \mathbb{E}\left[g^2_{\psi_2}(\widetilde{\boldsymbol{D}})\right] + \mathbb{E}\left[\mathcal{R}\left(f_{\phi_1}(\widetilde{\boldsymbol{M}})\right)\right]\mathbb{E}\left[g_{\psi_1}(\widetilde{\boldsymbol{D}})g_{\psi_2}(\widetilde{\boldsymbol{D}})\right]\end{aligned}$$

For each community $\mathcal{C}$, note that $\mathbb{E}[L_c] = \frac{n_c^2}{n^2}L$ and by setting

$$f_{\phi_{1,c}}(\boldsymbol{M}) = L_c \quad \text{and} \quad g_{\psi_{1,c}}(\boldsymbol{D}) = \frac{1}{L},$$

$$f_{\phi_{2,c}}(\boldsymbol{M}) = -1 \quad \text{and} \quad g_{\psi_{2,c}}(\boldsymbol{D}) = \frac{K_c^2}{4L^2},$$

we can approximate $\mathbb{E}[\widetilde{F}_c^2]$ by:

$$\begin{aligned}\mathbb{E}[\widetilde{F}_c^2] &= \left(\left(\frac{n_c^2 L}{n^2}\right)^2 + Var\left[\mathcal{R}\left(f_{\phi_{1,c}}(\widetilde{\boldsymbol{M}})\right)\right]\right)\cdot\mathbb{E}\left[g^2_{\psi_{1,c}}(\widetilde{\boldsymbol{D}})\right] \\ &\quad + \mathbb{E}\left[g^2_{\psi_{2,c}}(\widetilde{\boldsymbol{D}})\right] - \frac{2n_c^2 L}{n^2}\cdot\mathbb{E}\left[g_{\psi_{1,c}}(\widetilde{\boldsymbol{D}})\cdot g_{\psi_{2,c}}(\widetilde{\boldsymbol{D}})\right],\end{aligned} \tag{17}$$

where

$$\begin{aligned}&Var\left[\mathcal{R}\left(f_{\phi_{1,c}}(\widetilde{\boldsymbol{M}})\right)\right] \\ &= \frac{1}{2}n_c(n_c-1)\left(\frac{1}{16(p-\frac{1}{2})^2} - \left(\frac{2L}{n(n-1)} - \frac{1}{2}\right)^2\right)\end{aligned} \tag{18}$$

By Taylor expansion at $\mathbb{E}[\widetilde{d}_i]$, we have

$$\mathbb{E}[g^2_{\psi_{1,c}}(\widetilde{\boldsymbol{D}})] = \frac{1}{L^2} + \frac{6n^2}{(1-\alpha)^2\epsilon^2 L^4} \tag{19}$$

$$\begin{aligned}\mathbb{E}[g^2_{\psi_{2,c}}(\widetilde{\boldsymbol{D}})] = n_c^2\Bigg(&\frac{1}{n^4} + \frac{32}{n^2(1-\alpha)^2\epsilon^2 L^2} \\ &+ \frac{264}{(1-\alpha)^4\epsilon^4 L^4} + \frac{480n^2}{(1-\alpha)^6\epsilon^6 L^6}\Bigg)\end{aligned} \tag{20}$$

$$\begin{aligned}\mathbb{E}[g_{\psi_{1,c}}(\widetilde{\boldsymbol{D}})\cdot g_{\psi_{2,c}}(\widetilde{\boldsymbol{D}})] = n_c^2\Bigg(&\frac{1}{n^2 L} + \frac{14}{(1-\alpha)^2\epsilon^2 L^3} \\ &+ \frac{24n^2}{(1-\alpha)^4\epsilon^4 L^5}\Bigg)\end{aligned} \tag{21}$$

By substituting Eq. 18 to 21 into Eq. 17, we have

$$\begin{aligned}\mathbb{E}[\widetilde{F}_c^2] \approx \frac{n_c(n_c-1)((1-\alpha)^2\epsilon^2 L^2 + 6n^2)}{2(1-\alpha)^2\epsilon^2 L^4}&\left(\frac{1}{16(\frac{e^{\alpha\epsilon}}{1+e^{\alpha\epsilon}} - \frac{1}{2})^2}\right. \\ &\left. - \left(\frac{2L}{n(n-1)} - \frac{1}{2}\right)^2\right) + \frac{n_c^2 - n_c^4}{n^4}\end{aligned}$$

To minimize $\mathbb{E}[\widetilde{F}_c^2]$, we omit items $\frac{n_c(n_c-1)}{2}$ and $\frac{n_c^2-n_c^4}{n^4}$ that are independent of α, and derive α as

$$\arg\min_{\alpha\in(0,1)} \frac{(1-\alpha)^2\epsilon^2 L^2 + 6n^2}{(1-\alpha)^2\epsilon^2 L^4}\left(\frac{1}{16(\frac{e^{\alpha\epsilon}}{1+e^{\alpha\epsilon}} - \frac{1}{2})^2} - \left(\frac{2L}{n(n-1)} - \frac{1}{2}\right)^2\right)$$

□

Similar to obtaining $\hat{d}$ for clustering coefficient estimation in Section 5, the total number of edges L in graph can be obtained by using a portion of privacy budget for a preliminary round of node degree perturbation to collect $\widetilde{\boldsymbol{D}}$ and estimate L.

Algorithm 3 Collector-side modularity estimation

Input: A community $\mathcal{C}$
Perturbed adjacency matrix $\widetilde{\boldsymbol{M}} = \{\widetilde{\boldsymbol{B}}_1, ..., \widetilde{\boldsymbol{B}}_n\}$
Perturbed degree vector $\widetilde{\boldsymbol{D}} = \{\widetilde{d}_1, ..., \widetilde{d}_n\}$
Percentage α for privacy budget allocation
Output: $q_c = EstMod(\cdot)$, the estimated modularity of $\mathcal{C}$
Procedure:

1: Extract a sub-matrix $\widetilde{\boldsymbol{M}}_c$ from $\widetilde{\boldsymbol{M}}$
2: Obtain $\widetilde{L}_c$ by counting and halving the number of "1"s in $\widetilde{\boldsymbol{M}}_c$
3: Calibrate $\widetilde{L}_c$ to get an unbiased one L_c according to Eq. 16, where $p = \frac{e^{\alpha\epsilon}}{1+e^{\alpha\epsilon}}$.
4: Calculate the total number of edges in the whole graph $L = \frac{1}{2}\sum_{i=1}^{n}\widetilde{d}_i$
5: Calculate the total degree of all node in $\mathcal{C}$: $K_c = \sum_{c\in\mathcal{C}} d_c$
6: Calculate the estimated modularity of $\mathcal{C}$: $q_c = \frac{L_c}{L} - \frac{K_c^2}{4L^2}$
7: **return** q_c

6.2 Overall Algorithm

Algorithm 3 summarizes how the data collector estimates the modularity of a given community $\mathcal{C}$, according to the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$ and the degree vector $\widetilde{\boldsymbol{D}}$. First, it obtains $\widetilde{L}_c$, the number of edges in $\mathcal{C}$, by counting and halving the number of "1"s in $\widetilde{\boldsymbol{M}}_c$, the sub-matrix of $\mathcal{C}$ extracted from $\widetilde{\boldsymbol{M}}$ (Lines 1-2). It then calibrates $\widetilde{L}_c$ to an unbiased estimation L_c based on Eq. 16 (Line 3). Finally, the estimated modularity is calculated according to Eq. 15 (Line 6), which is based on L_c, L (obtained from Line 4) and K_c (obtained from Line 5).

Accuracy Guarantee. According to Theorem 6.1, with at least $1-\beta$ probability, the error of modularity estimation is bounded by $O(\frac{\sqrt{log(1/\beta)}}{n^2\cdot\epsilon})$.

Now that the modularity of any community can be estimated by Algorithm 3, we can adopt existing community detection methods that are based on modularity maximization [31]. In essence, they attempt to find a graph partition with the highest overall modularity of all communities. For ease of reference, Algorithm 4 presents the detailed implementation of *Louvain* method [31], a popular community detection method under LF-GDPR, where Algorithm 3 serves as the routine for modularity estimation.

As shown in Algorithm 4, there are two iterative phases in *Louvain*. In the first phase, the data collector assigns a different community to each node and calculates its modularity by invoking $EstMod(\cdot)$, i.e., Algorithm 3 (Lines 1-2). Then for each node i, the data collector calculates the gain of modularity that would take place by moving i to the community of its neighbor j (Line 5). Here $EstMod(\cdot)$ is invoked again to estimate the modularity of community $\{i, j\}$. Node i is then moved into the community in which this gain is positive and maximum (Line 7), and then the modularity of this community is also updated (Line 8). This process is repeated for all nodes until no individual move can improve the total modularity of the graph. The result of the first phase is a new set of communities (Line 10). In the second phase, a new graph is formed from this set of communities, and the data collector repeats the process in

Algorithm 4 Community detection under LF-GDPR with *Louvain* method

Input: Perturbed adjacency matrix $\widetilde{\boldsymbol{M}} = \{\widetilde{\boldsymbol{B}}_1, ..., \widetilde{\boldsymbol{B}}_n\}$
Perturbed degree vector $\widetilde{\boldsymbol{D}} = \{\widetilde{d}_1, ..., \widetilde{d}_n\}$
Privacy budget for adjacency bit vector perturbation ϵ_1
Output: A set of detected communities $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_1, \mathcal{C}_2, ...\}$
Procedure:

1: Initialize n communities $\{\mathcal{C}_i | 1 \leq i \leq n\}$, each consisting of only one node
2: Estimate the modularity of each $\mathcal{C}_i$: $q_i = EstMod(\mathcal{C}_i, \widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{D}}, \epsilon_1)$
3: **for** each node $i \in \{1, 2, ..., n\}$ **do**
4: **for** each node j so that $\widetilde{M}_{ij} = 1$ **do**
5: Calculate gain of modularity:

$$\Delta q_{ij} = EstMod(\{i \cup \mathcal{C}_j\}, \widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{D}}, \epsilon_1) - q_i - q_j$$

6: Move i to the community of j, where $j = \arg\max\{\Delta q_{ij} | \Delta q_{ij} > 0\}$
7: Update the modularity of $\mathcal{C}_j$: $q_j = EstMod(\mathcal{C}_j, \widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{D}}, \epsilon_1)$
8: Repeat Lines 3-7 until no individual move can improve the total modularity, and obtain a new set of communities $\boldsymbol{\mathcal{C}^*}$
9: Build a graph from $\boldsymbol{\mathcal{C}^*}$, and repeat Lines 3-8 to obtain $\boldsymbol{\mathcal{C}}$
10: **return** $\boldsymbol{\mathcal{C}}$

the first phase to detect the final set of communities (Line 11).

7 Experimental Evaluation

In this section, we compare the performance of LF-GDPR with two alternative methods, i.e., *RABV-only* and *LDPGen* [11] in both use cases, namely, clustering coefficient estimation and modularity estimation for community detection. In LF-GDPR, the optimal α for clustering coefficient estimation and modularity estimation is derived Theorem 5.1 and 6.1, respectively. Since the derivation is independent of the ground-truth data, we use this optimal α unless stated otherwise. *RABV-only* is a baseline solution where each node spends all its privacy budget in the *RABV* protocol and then derives her node degree from the perturbed adjacency bit vector. As for *LDPGen*, since it needs the clustering coefficient to generate a synthetic graph, we choose the most favorable one for it, i.e., the ground truth value. To have a fair comparison with *RABV-only* and *LDPGen*, for LF-GDPR, we use 10% of the privacy budget to estimate the domain knowledge in both use cases, i.e., the representative degree in Theorem 5.1 and the total number of graph edges in Theorem 6.1. All experiments run in Java on a desktop computer with Intel Core i7-8700K CPU, 64G RAM running Windows 10. The code of LF-GDPR and datasets are available in GitHub at https://github.com/Vicky-cs/LF-GDPR.

Performance measures. For the first use case, we measure the *Mean Square Error* (MSE) of the clustering coefficients of all nodes, i.e., $\frac{1}{n}\sum_{i=1}^{n}(cc_i - \widetilde{cc}_i)^2$. For the second use case, to evaluate the modularity estimation, we measure the *Relative Error* (RE) between the ground-truth modularity q and estimated modularity $\widetilde{q}$ of one community or all communities in a graph partition, i.e., $\frac{|q-\widetilde{q}|}{q}$. To evaluate the final community detection results, we adopt the same classic metrics for cluster validation as used in [11], namely *Adjusted Random Index* (ARI) [39] and *Adjusted Mutual Information* (AMI) [40]. They measure the similarity of two clusterings, and a larger ARI or AMI value indicates more similarity between them.

Datasets. We use four public datasets [41]. The first two are used in [11], and the rest two are added to evaluate on denser and larger graphs.

(1) *Facebook* — an undirected social network of 4,039 nodes and 88,234 edges, from a survey of participants in Facebook app.
(2) *Enron* — an undirected email communication network of 36,692 nodes and 183,831 edges.
(3) *AstroPh* — an undirected collaboration network of 18,772 authors and 198,110 edges indicating collaborations between authors in arXiv, who submitted papers to Astro Physical category.
(4) *Gplus* — an undirected social network of 107,614 Google+ users and 12,238,285 edges indicating shares of social circles.[6]

7.1 Clustering Coefficient Estimation

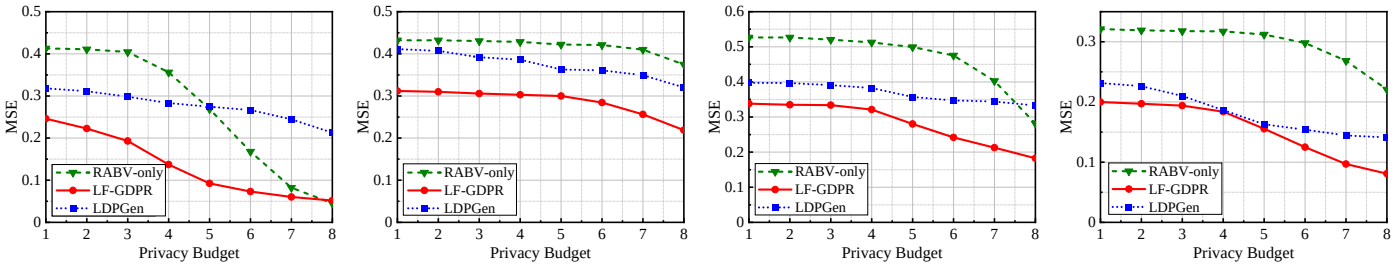
Fig. 5 shows the clustering coefficient estimation accuracy of LF-GDPR and two alternative methods over all datasets, with privacy budget ϵ varying from 1 to 8. In all cases, LF-GDPR is the most accurate. Furthermore, it always significantly outperforms *RABV-only*, which justifies our rationale in Section 3.1 that node degree derived from perturbed adjacency bit vector is too noisy. As ϵ increases, the accuracy of LF-GDPR and *RABV-only* improves significantly while *LDPGen* does not. This is because *LDPGen* is only affected by the Laplace noise added to node degree, which is already very small when $\epsilon > 2$. In other words, *LDPGen* cannot fully exploit a large privacy budget.

To evaluate the impact of privacy budget allocation on the estimation accuracy, we compare LF-GDPR with optimal allocation (derived from Eq. 11) against LF-GDPR with four constant α, namely, 0.3, 0.5, 0.7 and 0.9 in Fig. 6. Due to the space limitation, we only show the results of *Facebook*. The optimal allocation achieves the lowest MSE in most cases. As for the constant α, we observe that a large ϵ always favors a large α, which indicates that the privacy budget needed by node degree perturbation is relatively stable, and therefore surplus budget should be mostly allocated to the adjacency bit vector. However, when privacy budget is small (e.g., $\epsilon < 2$), large α (e.g., $\alpha = 0.9$) leads to high MSE. The same observation is also made in modularity estimation, which is therefore omitted in the interest of space.

7.2 Modularity Estimation and Community Detection

In this experiment, we evaluate the modularity estimation and Louvain-based community detection of LF-GDPR against *RABV-only*, *LDPGen*, and the ground truth. To allow fair comparison, we use the same algorithms for the latter three except that the modularity is estimated from the perturbed adjacency matrix only (for *RABV-only*), or directly calculated from the synthetic graph (for *LDPGen*), or directly calculated from the original graph (for ground truth). Fig. 7 plots the RE of modularity by these three methods against ground truth in all datasets. LF-GDPR always outperforms the other two and its RE approaches 0 as ϵ increases, especially in *Facebook* and *Gplus* which have

6. The original *Gplus* dataset is a directed graph, and we convert it to an undirected graph to align with the other three datasets.



(a) Facebook (b) Enron (c) AstroPh (d) Gplus

Fig. 5. Mean square error of clustering coefficient estimation

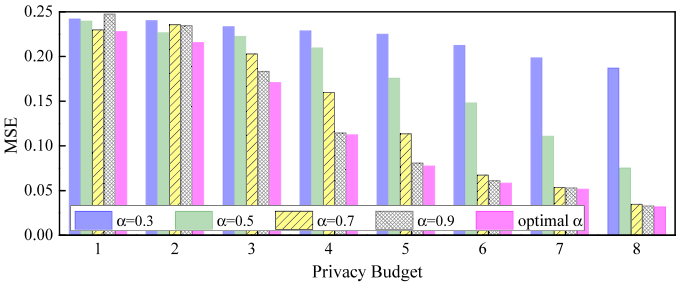


Fig. 6. Mean square error of clustering coefficient estimation, varying α

a higher mean degree than the other two datasets. *RABV-only* has the second lowest RE when ϵ is large, especially in *Facebook*, which means when the privacy budget is sufficient, adjacency bit vector alone can also estimate modularity fairly well. However, when ϵ is small, *RABV-only* has the highest RE among the three, which justifies our rationale in Section 3.1 that the estimated degree from a perturbed adjacency matrix could be too noisy to be meaningful. *LDPGen*, on the other hand, still has very high RE even when ϵ is large, which also justifies our rationale in Section 3.1 that the neighborhood information is lost in a synthetic graph.

To compare the detected communities against ground truth, we plot ARI and AMI between the estimated and ground-truth graph partitions of each method[7] in Fig. 8. Due to space limitation, we only show the results of *Facebook* and *Enron*. LF-GDPR achieves higher ARI and AMI than *LDPGen* when $\epsilon > 1$, which means the detected communities by LF-GDPR are closer to the ground truth communities detected in the original graph. Particularly, in *Facebook* both ARI and AMI of LF-GDPR approach 1 for large ϵ (e.g., $\epsilon \geq 7$), which means that the detected communities are almost identical to the ground truth communities. We can also verify this observation from a visualization tool *Gephi* in Fig. 9, which illustrates three sets of communities detected from the original graph and from LF-GDPR ($\epsilon = 8$, $\epsilon = 1$) respectively. The sizes of top-3 communities in each set are also marked.

On the other hand, as with the RE results, *LDPGen* has steady ARI/AMI curves because it does not have the neighborhood information of the original graph. As such, it becomes significantly inferior to LF-GDPR when there is a large privacy budget to spend. Dataset-wise, both LF-GDPR and *LDPGen* perform better in *Facebook* than in *Enron*. This is because *Enron* is more sparse and therefore has more communities — 1275 vs. 16 in *Facebook*.

In addition, we evaluate the accuracy of modularity estimation with respect to the size of a community. For datasets *Facebook* and *Enron*, we randomly select 500 small (5% of the total nodes) communities and 500 large (20% of the total nodes) communities. Then we apply both LF-GDPR and *RABV-only* to estimate the modularity of each community and measure its RE against the ground truth modularity of that community. Due to space limitation, Fig. 10 only shows the results of *Facebook* and *Enron*. LF-GDPR significantly outperforms *RABV-only* in both small and large communities, due to the excessive noise in the node degree introduced by *RABV-only*. We also observe that both methods work better for smaller communities and for the *Facebook* dataset (than the *Enron* dataset). We believe this indicates that LF-GDPR is more superior for denser graphs with more edges per node.

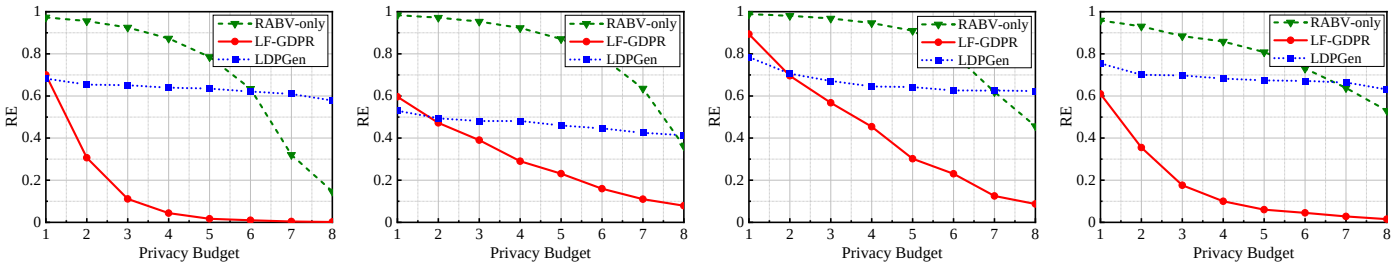
TABLE 2
Communication bandwidth cost (in kilobytes)

Dataset	LF-GDPR	*RABV-only*	*LDPGen*
Facebook	0.25	0.25	3.05
Enron	2.30	2.29	27.55
AstroPh	1.18	1.17	14.10
Gplus	6.73	6.73	80.73

To evaluate the communication bandwidth cost, we show the number of kilobytes (kB) between a node and the data collector for all datasets in Table 2. We observe that all three methods are proportional to the node size n, whereas *LDPGen* is also logarithmic to the number of groups g, an internal parameter of *LDPGen*. This coincides with the asymptotic complexity — $O(2n + \lceil \log g \rceil n)$ of *LDPGen* vs. $O(n)$ of LF-GDPR. As such, we expect *LDPGen* incurs even higher communication cost as the graph becomes larger due to an increasing g.

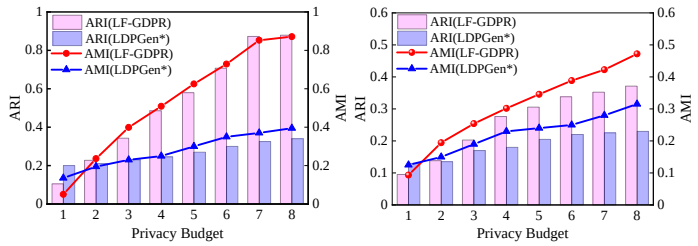
To evaluate the computation cost, we show the runtime of both metric estimation at the collector side in Table 3, with privacy budget ϵ ranging from 1 to 8. Due to the space limitation, we only show the results of *Facebook*. LF-GDPR and *RABV-only* have comparable runtime and decrease significantly with large ϵ. For small ϵ, the perturbed adjacency bit matrix is very dense, so the computation of metrics becomes time-consuming. On the other hand, *LDPGen* always needs to generate a synthetic graph with almost the same number

7. For *LDPGen**, we use the results directly from [11] because the calculation of ARI and AMI between partitions from two (similar) graphs requires an optimal node-to-cluster mapping, which is not specified in [11].



(a) Facebook (b) Enron (c) AstroPh (d) Gplus

Fig. 7. Relative error of modularity of detected communities



(a) Facebook (b) Enron

Fig. 8. Results of ARI and AMI

TABLE 3
Runtime in Data Collector Side for Facebook(in milliseconds)

Privacy Budget	Clustering Coefficient Estimation			Modularity Estimation		
	LF-GDPR	*RABV-only*	LDPGen	LF-GDPR	*RABV-only*	LDPGen
1	12686	11715	910	120	331	842
2	2273	1825	929	76	82	866
3	469	453	942	52	55	882
4	225	276	892	35	36	845
5	105	143	959	32	33	906
6	100	146	957	29	30	903
7	85	102	949	28	28	883
8	79	96	926	28	28	880

of edges as the original one, so its runtime is independent of ϵ and is outperformed by LF-GDPR and *RABV-only* except for small ϵ.

7.3 LF-GDPR VS. Dedicated LDP Solutions

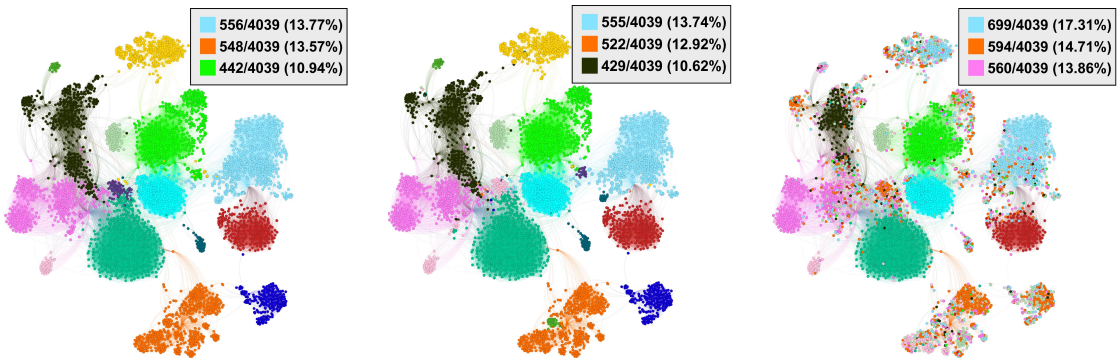
As mentioned in the introduction, dedicated LDP solutions may provide a better utility than a general framework as LF-GDPR. In this subsection, we conduct such a comparative study on clustering coefficient estimation (CCE) and modularity estimation for community detection (CD). The main challenge is the design of local perturbation mechanisms that can provide a global view which is needed for these two graph metrics. To address this, we equip each individual user with sufficient ground truth knowledge and design two optimistic dedicated solutions, i.e., *Dedicated-CCE* and *Dedicated-CD*. In Dedicated-CCE, we assume each user knows the entire ground-truth adjacency matrix, based on which her clustering coefficient is calculated and then perturbed by adding Laplace noise. In Dedicated-CD, we assume each user knows the ground-truth graph partition, based on which her number of edges linked to her community $\mathcal{C}$ is counted, perturbed by adding Laplace noise together with her node degree, and sent to the collector to calculate the modularity q_c by Eq. 2. Note that these two dedicated solutions provide the same ϵ-edge LDP guarantee as LF-GDPR. But since they optimistically assume to know the ground truth, their estimation accuracy only serves as the upper bound of dedicated solutions for CCE and CD.

Figs. 11 (a) and (b) show the mean square error (MSE) of clustering coefficient estimation of *Dedicated-CCE* and LF-GDPR on *Facebook* and *Enron* datasets. In the interest of space, the results on other datasets are omitted. We observe that *Dedicated-CCE* has very large MSE when the privacy budget is small, and it gradually outperforms LF-GDPR when $\epsilon \geq 4$ (on *Facebook*) or $\epsilon \geq 3$ (on *Enron*). But its MSE is at least 64% and 16% of the MSE of LF-GDPR on two datasets when $\epsilon = 8$. Figs. 11 (c) and (d) show the relative error (RE) of modularity estimation of *Dedicated-CD* and LF-GDPR over *Facebook* and *Enron*. Similar to CCE, there is no all-winner — LF-GDPR performs better on *Facebook* when $\epsilon \geq 3$ whereas *Dedicated-CD* gains higher accuracy (but its RE is at least 20% of the RE of LF-GDPR) in other cases. To summarize, we conclude that LF-GDPR is able to obtain comparable estimation accuracy as dedicated LDP solutions for graph metric estimation.

8 Related Work

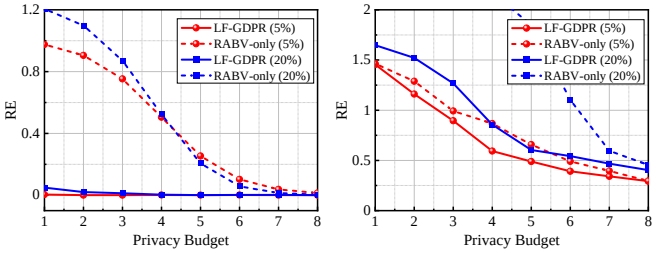
There are three related fields: privacy-preserving graph release, graph analytics with differential privacy, and local differential privacy.

Privacy-Preserving Graph Release. This field studies how a data owner publishes a privacy-preserving graph. Early works focus on anonymization techniques under those privacy models derived from *k*-anonymity [42]. Zhou *et al.* proposed *k*-neighborhood anonymity to defend against neighborhood attacks [3], Liu *et al.* proposed *k*-degree anonymity against degree attacks [4], Zou *et al.* and Cheng *et al.* proposed *k*-automorphism [5] and *k*-isomorphism [6] respectively against structural attacks, and Xue *et al.* proposed random edge perturbation against walk-based structural identification [43]. As these approaches can be vulnerable to de-anonymization techniques [44], more rigorous privacy notions are proposed, such as *L*-opacity [45] and differential privacy (DP) [12]. The former ensures an adversary cannot infer whether the distance between two nodes is equal to or less than L. The latter uses a generative graph model to fit the original graph, and then produces a synthetic graph for analytics. Common graph models include *dK-series* [16], Stochastic Kronecker Graph (SKG) model [46], Exponential Random Graph Model (ERGM) [17], Attributed Graph



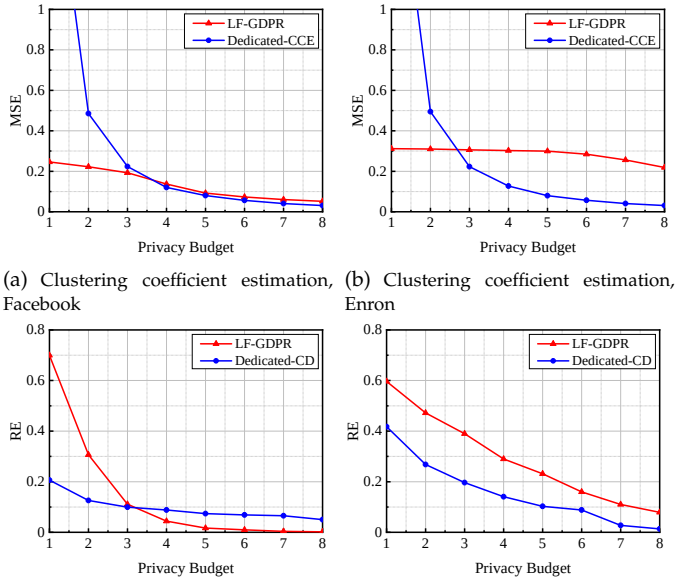
(a) Ground truth (b) $\epsilon = 8$ (c) $\epsilon = 1$

Fig. 9. Visualization of detected communities by *Gephi*



(a) Facebook (selected communities) (b) Enron (selected communities)

Fig. 10. Impact of community size on modularity estimation



(a) Clustering coefficient estimation, Facebook (b) Clustering coefficient estimation, Enron

(c) Modularity for community detection, Facebook (d) Modularity for community detection, Enron

Fig. 11. Comparison with dedicated LDP solutions

Model (AGM) [18], Hierarchical Random Graph (HRG) [19], and BTER [11] (which adopts LDP).

Graph Analytics with Differential Privacy. This field studies how to estimate graph metric and statistics with differential privacy. Most of the existing work focuses on centralized differential privacy. Nissim *et al.* estimated the cost of the minimum spanning tree and the number of triangles in a graph [7]. This technique has been extended to subgraph counting queries [20], [47] such as k-stars, k-triangles and k-cliques, and frequent subgraph mining [8], [48]. Other works estimate the distribution of node degree [14], [21] and clustering coefficient [22]. In the local setting, Sun *et al.* [47] propose to estimate subgraph counts in a decentralized graph. In our previous work [49], we briefly introduce the LF-GDPR framework that estimates generic graph metrics with local differential privacy. This work has advanced our previous work in almost all aspects. First, this work materializes all algorithms in the LF-GDPR framework. Second, it proposes a refinement strategy for degree estimation and an optimal privacy budget allocation. Third, this work shows use cases on two common graph analysis tasks, namely, clustering coefficient estimation and community detection. Last but not the least, this work comprehensively evaluates the proposed algorithms on four public datasets.

Local Differential Privacy (LDP). Due to its decentralized nature and no need of a trusted party, LDP becomes increasingly popular in privacy-preserving data collection [10], [50]. Existing works focus on estimating statistics such as frequency [32], [51], [52], mean [28], [30], heavy hitter [35], frequent itemset mining [53], k-way marginal release [54], [55], key-value data collection [29], [56] and time-series data collection [57]. Some works also focus on learning problems [30].

9 Conclusion

This paper presents a parameterized framework LF-GDPR for privacy-preserving graph metric estimation and analytics with local differential privacy. The building block is a user-side perturbation algorithm, and a collector-side aggregation and calibration algorithm. LF-GDPR simplifies the job of developing a practical LDP solution for a graph analysis task by providing a complete solution for all LDP steps. An optimal allocation of privacy budget between the two atomic metrics is also designed. Through theoretical and experimental analysis, we verify the privacy and data utility achieved by this framework.

As for future work, we plan to extend LF-GDPR to more specific graph types and graph analysis tasks, such as attributed graph and DAG, and influential node analysis, to demonstrate its wide applicability. We will also investigate some relaxation of DP, such as Gaussian Mechanism and

(ϵ, δ)-DP, to provide higher estimation accuracy and better utility. Graph-specific tighter bounds for the composition of DP [58] and the correlation of graph data will also be studied.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (Grant No: 62072390, U1636205, 91646203, 61941121 and 61972332), the Research Grants Council, Hong Kong SAR, China (Grant No: 15238116, 15222118, 15218919, 15203120 and C1008-16G), the Ministry of Education, Singapore (Grant No: MOE2018-T2-2-091).

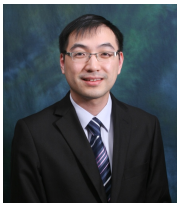
REFERENCES

[1] B. Stephanie, Facebook Scandal a 'Game Changer' in Data Privacy Regulation, *Bloomberg*, Apr 8, 2018.
[2] Facebook, https://developers.facebook.com/docs/graph-api/, graph API - Facebook for Developers.
[3] B. Zhou and J. Pei, "Preserving privacy in social networks against neighborhood attacks," in *ICDE*. IEEE, 2008, pp. 506–515.
[4] K. Liu and E. Terzi, "Towards identity anonymization on graphs," in *SIGMOD*. ACM, 2008, pp. 93–106.
[5] L. Zou, L. Chen, and M. T. Özsu, "K-automorphism: A general framework for privacy preserving network publication," *PVLDB*, vol. 2, no. 1, pp. 946–957, 2009.
[6] J. Cheng, A. W. Fu, and J. Liu, "K-isomorphism: privacy preserving network publication against structural attacks," in *SIGMOD*. ACM, 2010, pp. 459–470.
[7] K. Nissim, S. Raskhodnikova, and A. Smith, "Smooth sensitivity and sampling in private data analysis," in *STOC*. ACM, 2007, pp. 75–84.
[8] S. Xu, S. Su, L. Xiong, X. Cheng, and K. Xiao, "Differentially private frequent subgraph mining," in *ICDE*. IEEE, 2016, pp. 229–240.
[9] 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, April 2020.
[10] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *FOCS*. IEEE, 2013, pp. 429–438.
[11] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren, "Generating synthetic decentralized social graphs with local differential privacy," in *CCS*. ACM, 2017, pp. 425–438.
[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *TCC*. Springer, 2006, pp. 265–284.
[13] 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.
[14] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith, "Analyzing graphs with node differential privacy," in *TCC*. Springer, 2013, pp. 457–476.
[15] J. Blocki, A. Blum, A. Datta, and O. Sheffet, "The johnson-lindenstrauss transform itself preserves differential privacy," in *FOCS*. IEEE, 2012, pp. 410–419.
[16] A. Sala, X. Zhao, C. Wilson, H. Zheng, and B. Y. Zhao, "Sharing graphs using differentially private graph models," in *IMC*, 2011, pp. 81–98.
[17] W. Lu and G. Miklau, "Exponential random graph estimation under differential privacy," in *KDD*. ACM, 2014, pp. 921–930.
[18] Z. Jorgensen, T. Yu, and G. Cormode, "Publishing attributed social graphs with formal privacy guarantees," in *SIGMOD*, 2016, pp. 107–122.
[19] Q. Xiao, R. Chen, and K. Tan, "Differentially private network data release via structural inference," in *KDD*. ACM, 2014, pp. 911–920.
[20] V. Karwa, S. Raskhodnikova, A. Smith, and G. Yaroslavtsev, "Private analysis of graph structure," *PVLDB*, vol. 4, no. 11, pp. 1146–1157, 2011.
[21] M. Hay, C. Li, G. Miklau, and D. Jensen, "Accurate estimation of the degree distribution of private networks," in *ICDM*, 2009, pp. 169–178.
[22] Y. Wang, X. Wu, J. Zhu, and Y. Xiang, "On learning cluster coefficient of private networks," *Social network analysis and mining*, vol. 3, no. 4, pp. 925–938, 2013.
[23] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "SCAN: a structural clustering algorithm for networks," in *KDD*. ACM, 2007, pp. 824–833.
[24] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann, 2011.
[25] T. Martin, X. Zhang, and M. Newman, "Localization and centrality in networks," *Physical review E*, vol. 90, no. 5, p. 052808, 2014.
[26] 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.
[27] 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.
[28] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *NIPS*, 2017, pp. 3574–3583.
[29] Q. Ye, H. Hu, X. Meng, and H. Zheng, "PrivKV: Key-value data collection with local differential privacy," in *S&P*. IEEE, 2019, pp. 317–331.
[30] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *ICDE*. IEEE, 2019.
[31] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
[32] T. Wang, J. Blocki, N. Li, and S. Jha, "Locally differentially private protocols for frequency estimation," in *USENIX Security Symposium*, 2017, pp. 729–745.
[33] P. Kairouz, K. Bonawitz, and D. Ramage, "Discrete distribution estimation under local privacy," in *ICML*. ACM, 2016, pp. 2436–2444.
[34] S. S. Wilks, "The large-sample distribution of the likelihood ratio for testing composite hypotheses," *The Annals of Mathematical Statistics*, vol. 9, no. 1, pp. 60–62, 1938.
[35] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren, "Heavy hitter estimation over set-valued data with local differential privacy," in *CCS*. ACM, 2016, pp. 192–203.
[36] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.
[37] R. Chen, B. C. Fung, S. Y. Philip, and B. C. Desai, "Correlated network data publication via differential privacy," *The VLDB Journal*, vol. 23, no. 4, pp. 653–676, 2014.
[38] B. Yang, I. Sato, and H. Nakagawa, "Bayesian differential privacy on correlated data," in *SIGMOD*. ACM, 2015, pp. 747–762.
[39] 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.
[40] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" in *ICML*. ACM, 2009, pp. 1073–1080.
[41] L. Jure and K. Andrej, "SNAP Datasets: Stanford large network dataset collection," http://snap.stanford.edu/data, 2014.
[42] P. Samarati, "Protecting respondents identities in microdata release," *TKDE*, vol. 13, no. 6, pp. 1010–1027, 2001.
[43] M. Xue, P. Karras, R. Chedy, P. Kalnis, and H. Pung, "Delineating social network data anonymization via random edge perturbation," in *CIKM*, 2012, pp. 475–484.
[44] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *S&P*. IEEE, 2009, pp. 173–187.
[45] S. Nobari, P. Karras, H. Pang, and S. Bressan, "L-opacity: Linkage-aware graph anonymization," in *EDBT*. Springer, 2014, pp. 583–594.
[46] D. Mir and R. N. Wright, "A differentially private estimator for the stochastic kronecker graph model," in *EDBT/ICDT Workshops*. ACM, 2012, pp. 167–176.
[47] H. Sun, X. Xiao, I. Khalil, Y. Yang, Z. Qin, H. Wang, and T. Yu, "Analyzing subgraph statistics from extended local views with decentralized differential privacy," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 703–717.
[48] E. Shen and T. Yu, "Mining frequent graph patterns with differential privacy," in *KDD*. ACM, 2013, pp. 545–553.
[49] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao, "Towards locally differentially private generic graph metric estimation," in *ICDE*. IEEE, 2020, pp. 1922–1925.

[50] N. Li and Q. Ye, "Mobile data collection and analysis with local differential privacy," in *MDM*. IEEE, 2019, pp. 4–7.
[51] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *CCS*. ACM, 2014, pp. 1054–1067.
[52] R. Bassily and A. Smith, "Local, private, efficient protocols for succinct histograms," in *STOC*. ACM, 2015, pp. 127–135.
[53] T. Wang, N. Li, and S. Jha, "Locally differentially private frequent itemset mining," in *S&P*. IEEE, 2018, pp. 127–143.
[54] G. Cormode, T. Kulkarni, and D. Srivastava, "Marginal release under local differential privacy," in *SIGMOD*. ACM, 2018, pp. 131–146.
[55] Z. Zhang, T. Wang, N. Li, S. He, and J. Chen, "CALM: Consistent adaptive local marginal for marginal release under local differential privacy," in *CCS*. ACM, 2018, pp. 212–229.
[56] X. Gu, M. Li, L. Xiong, and Y. Cao, "PCKV: locally differentially private correlated key-value data collection with optimized utility," in *USENIX Security Symposium*, 2020.
[57] Q. Ye, H. Hu, N. Li, X. Meng, H. Zheng, and H. Yan, "Beyond value perturbation: Differential privacy in the temporal setting," in *INFOCOM*. IEEE, 2021.
[58] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," in *ICML*, 2015, pp. 1376–1385.

Qingqing Ye is a research assistant professor in the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University. She received her PhD degree in Computer Science from Renmin University of China in 2020. She has received several prestigious awards, including China National Scholarship, Outstanding Doctoral Dissertation Award, and IEEE S&P Student Travel Award. Her research interests include data privacy and security, and adversarial machine learning.

Haibo Hu is an associate professor in the Department of Electronic and Information Engineering, Hong Kong Polytechnic University. His research interests include cybersecurity, data privacy, internet of things, and machine learning. He has published over 80 research papers in refereed journals, international conferences, and book chapters. As principal investigator, he has received over 12 million HK dollars of external research grants from Hong Kong and mainland China. He is the recipient of a number of titles and awards, including IEEE MDM 2019 Best Paper Award, WAIM Distinguished Young Lecturer, VLDB Distinguished Reviewer, ACM-HK Best PhD Paper, Microsoft Imagine Cup, and GS1 Internet of Things Award.



Man Ho Au is an associate professor of the Department of Computer Science at the University of Hong Kong (HKU). Before joining HKU, he was an associate professor in the Department of Computing of the Hong Kong Polytechnic University. His research interests include applied cryptography, information security, blockchain technology, and related industrial applications. Dr. Au has published over 170 refereed papers in top journals and conferences, including CRYPTO, ACM CCS, ACM SIGMOD, NDSS, IEEE TIFS, TKDE. He is a recipient of the 2009 PET runner-up award for outstanding research in privacy-enhancing technologies, and best paper awards of ACISP 2016, ISPEC 2017, and ACISP 2018. He is a general chair of ASIACCS 2021, an expert member of the China delegation of ISO/IEC JTC 1/SC 27 working group 2 Cryptography and security mechanisms, and a committee member of the Hong Kong Blockchain Society R&D division.

Xiaofeng Meng is a professor in School of Information, Renmin University of China. He is a CCF Fellow and the vice chair of the Special Interesting Group on Privacy of China Confidentiality Association(CCA). He has served on the program committee SIGMOD, ICDE, CIKM, MDM, DASFAA, etc., and editorial board of JCST, FCS, JoS, CRAD, etc. His research interests include web data management, cloud data management, mobile data management, and privacy protection. He has published over 200 papers in refereed international journals and conference proceedings including IEEE TKDE,VLDBJ , VLDB, SIGMOD, ICDE, EDBT, ACM GIS etc.

Xiaokui Xiao received his PhD degree in computer science and engineering from the Chinese University of Hong Kong in 2008. He is currently an associate professor at the School of Computing, National University of Singapore (NUS). His research interests include data privacy and algorithms for large data.

Towards Locally Differentially Private Generic Graph Metric Estimation

Qingqing Ye*, Haibo Hu†, Man Ho Au†, Xiaofeng Meng*, Xiaokui Xiao♯
*Renmin University of China; †Hong Kong Polytechnic University; ♯National University of Singapore
yeqq@ruc.edu.cn; haibo.hu@polyu.edu.hk; csallen@comp.polyu.edu.hk; xfmeng@ruc.edu.cn; xkxiao@nus.edu.sg

***Abstract*—Local differential privacy (LDP) is an emerging technique for privacy-preserving data collection without a trusted collector. Despite its strong privacy guarantee, LDP cannot be easily applied to real-world graph analysis tasks such as community detection and centrality analysis due to its high implementation complexity and low data utility. In this paper, we address these two issues by presenting LF-GDPR, the first LDP-enabled graph metric estimation framework for graph analysis. It collects two atomic graph metrics — the adjacency bit vector and node degree — from each node locally. LF-GDPR simplifies the job of implementing LDP-related steps (e.g., local perturbation, aggregation and calibration) for a graph metric estimation task by providing either a complete or a parameterized algorithm for each step.**

***Index Terms*—Local differential privacy; Graph metric; Privacy-preserving graph analysis**

I. Introduction

With the prevalence of big data and machine learning, graph analytics has received great attention and nurtured numerous applications in web, social network, transportation, and knowledge base. However, recent privacy incidents, particularly the Facebook privacy scandal, pose real-life threats to any **centralized** party who needs to safeguard graph data of individuals while providing graph analysis service to third parties. In that scandal, Facebook exposed the personal profiles of 87 million users to Cambridge Analytica through Facebook API for third-party apps [11]. The main cause is that Facebook allows these apps to access the **friends list** of a user, which helps to propagate these apps easily through friends. Unfortunately, most existing privacy models assume that the trusted party cannot be compromised, which is seldom true in practice as echoed by this scandal. With General Data Protection Regulation (GDPR) enforced in EU since May 2018, there is a compelling need to find alternative privacy models without such a trusted party.

A promising model is local differential privacy (LDP) [1], [15], where each individual user **locally perturbs her share of graph metrics** (e.g., node degree and adjacency list, depending on the graph analysis task) before sending them to the data collector for analysis. As such, the data collector does not need to be trusted. A recent work *LDPGen* [10] has also shown the potential of LDP for graph analytics. In that work, LDP is used to collect node degree for synthetic graph generation. However, such solution is usually task specific — for different tasks, such as centrality analysis and community detection, dedicated LDP solutions must be designed from scratch. To show how complicated it is, an LDP solution usually takes four steps: (1) selecting graph metrics to collect from users for the target metric (e.g., clustering coefficient, modularity, or centrality) of this task, (2) designing a local perturbation algorithm for users to report these metrics under LDP, (3) designing a collector-side aggregation algorithm to estimate the target metric based on the perturbed data, (4) designing an optional calibration algorithm for the target metric if the estimation is biased. Obviously, working out such a solution **requires in-depth knowledge of LDP**, which hinders the embrace of LDP by more graph applications.

In this paper, we address this challenge by presenting LF-GDPR (Local Framework for Graph with Differentially Private Release), the first LDP-enabled graph metric estimation framework for general graph analysis. It simplifies the job of a graph application to design an LDP solution for a graph metric estimation task by providing complete or parameterized algorithms for steps (2)-(4) as above. As long as the target graph metric can be derived from the two atomic metrics, namely, the adjacency bit vector and node degree, the parameterized algorithms in steps (2)-(4) can be completed with ease. To summarize, our main contributions of this paper are as follows.

- This is the first LDP-enabled graph metric estimation framework for a variety of graph analysis tasks.
- We present efficient perturbation algorithms on adjacency bit vector and node degree, respectively, to address data correlation among nodes.
- We provide a complete solution for local perturbation, collector-side aggregation, and calibration.

The rest of the paper is organized as follows. Section II introduces preliminaries on local differential privacy and graph analytics. Section III presents an overview of LF-GDPR. Section IV describes the implementation details of this framework. Section V draws a conclusion with future work.

II. Local Differential Privacy on Graphs

In this paper, a graph G is defined as $G = (V, E)$, where $V = \{1, 2, ..., n\}$ is the set of nodes, and $E \subseteq V \times V$ is the set of edges. For the node i, d_i denotes its degree and $\boldsymbol{B}_i = \{b_1, b_2, ..., b_n\}$ denotes its *adjacency bit vector*, where $b_j = 1$ if and only if edge $(i, j) \in E$, and otherwise $b_j = 0$. The adjacency bit vectors of all nodes constitute the *adjacency matrix* of graph G, or formally, $\boldsymbol{M}_{n\times n} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, ..., \boldsymbol{B}_n\}$.

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00204

Local differential privacy (LDP) [1] is proposed to assume each individual is responsible for her own tuple in the database. In LDP, each user locally perturbs her tuple using a randomized algorithm before sending it to the untrusted data collector. Formally, a randomized algorithm $\mathcal{A}$ satisfies ϵ-local differential privacy, if for any two input tuples t and t' and for any output t^*, $\frac{\Pr[\mathcal{A}(\mathrm{t})=\mathrm{t}^*]}{\Pr[\mathcal{A}(\mathrm{t}')=\mathrm{t}^*]} \leq e^\epsilon$ holds. As with existing LDP works, we concern attacks where an adversary can infer with high confidence whether an edge exists or not, which compromises a user's relation anonymity in a social network. This directly leads to Definition 2.1.

Definition 2.1: (Edge local differential privacy). A randomized algorithm $\mathcal{A}$ satisfies ϵ-edge local differential privacy (a.k.a., ϵ-edge LDP), if and only if for any two adjacency bit vectors $\boldsymbol{B}$ and $\boldsymbol{B'}$ that differ only in one bit, and any output $s \in range(\mathcal{A})$, $\frac{\Pr[\mathcal{A}(\boldsymbol{B})=s]}{\Pr[\mathcal{A}(\boldsymbol{B'})=s]} \leq e^\epsilon$ holds.

III. LF-GDPR: Framework Overview

A. Design Principle

The core of privacy-preserving graph analytics often involves **estimating some target graph metric** without accessing the original graph. Under the DP/LDP privacy model, there are two solution paradigms, namely, generating a synthetic graph to calculate this metric [5], [7], [10] and designing a dedicated DP/LDP solution for such metric [4], [6], [9]. The former provides a general solution but suffers from low estimation accuracy as **the neighborhood information in the original graph is missing** from the synthetic graph. The latter can achieve higher estimation accuracy but cannot generalize such a dedicated solution to other problems — it works poorly or even no longer works if the target graph metric or graph type (e.g., undirected graph, attributed graph, and DAG) is changed [5].

LF-GDPR is our answer to both solution generality and estimation accuracy under the LDP model. It collects from each node i two atomic graph metrics that can derive a wide range of common metrics. The first is the **adjacency bit vector** $\boldsymbol{B}$, where each element j is 1 only if j is a neighbor of i. $\boldsymbol{B}$ of all nodes collectively constitutes the adjacency matrix $\boldsymbol{M}$ of the graph. The second metric is **node degree** d, which is frequently used in graph analytics to measure the density of connectivity [4]. Table I lists some of the most popular graph analysis tasks in the literature [3], [8], [13] and their graph metrics, all of which can be derived from $\boldsymbol{B}$, $\boldsymbol{M}$ and d.

Intuitively, d can be estimated from $\boldsymbol{B}$. However, given a large graph and limited privacy budget, the estimation accuracy could be too noisy to be meaningful. To illustrate this, let us assume each bit of the adjacency bit vector $\boldsymbol{B}$ is perturbed independently by the classic Randomized Response (RR) [12] algorithm with privacy budget ϵ. As stated in [12], the variance of the estimated node degree $\widetilde{d}$ is

$$Var[\widetilde{d}] = n \cdot \Big[\frac{1}{16(\frac{e^\epsilon}{e^\epsilon+1} - \frac{1}{2})^2} - (\frac{d}{n} - \frac{1}{2})^2\Big] \qquad (1)$$

Even for a moderate social graph with extremely large privacy budget, for example, $d = 100$, $n = 1M$, and $\epsilon = 8$ (the largest ϵ used in [10] is 7), $Var[\widetilde{d}] \approx 435 > 4d$, which means the variance of the estimated degree is over 4 times that of the degree itself. As such, we choose to spend some privacy budget on an independently perturbed degree. This further motivates us to design an optimal privacy budget allocation between adjacency bit vector $\boldsymbol{B}$ and node degree d, to minimize the distance between the target graph metric and the estimated one.

TABLE I
Popular graph analysis tasks and metrics

Graph Analysis Task	Graph Metric Concerned	Derivation from $\boldsymbol{B}$, $\boldsymbol{M}$, and d
synthetic graph generation	clustering coefficient	$cc_i = \frac{\boldsymbol{M}^3_{ii}}{d_i(d_i-1)}$
community detection, graph clustering	modularity	$Q_c = \frac{\|\|\boldsymbol{M}_c\|\|}{\sum d} - \frac{\|\|\boldsymbol{d}_c\|\|^2}{(\sum d)^2}$
node role, page rank	degree centrality	$c_i = d_i$
	eigenvector centrality	$c_i = \boldsymbol{B}_i\boldsymbol{M}^k$
connectivity analysis (clique / hub)	structural similarity	$\tau(i,j) = \frac{\|\|\boldsymbol{B}_i \bigcap \boldsymbol{B}_j\|\|}{\sqrt{d_i d_j}}$
node similarity search	cosine similarity	$\tau(i,j) = \frac{\boldsymbol{B}_i\boldsymbol{B}'_j}{\sqrt{d_i d_j}}$

To summarize, in LF-GDPR each node sends two perturbed atomic metrics, namely, the adjacency bit vector $\widetilde{\boldsymbol{B}}$ (perturbed from $\boldsymbol{B}$) and node degree $\widetilde{d}$ (perturbed from d), to the data collector, who then aggregates them to estimate the target graph metric.

B. LF-GDPR Overview

LF-GDPR works as shown in Fig. 1. A data collector who wishes to estimate a target graph metric F first reduces it from the adjacency matrix $\boldsymbol{M}$ and degree vector $\boldsymbol{d}$ of all nodes by deriving a mapping function $F = Map(\boldsymbol{M}, \boldsymbol{d})$ (step ①). Based on this reduction, LF-GDPR allocates the total privacy budget ϵ between $\boldsymbol{M}$ and $\boldsymbol{d}$, denoted by ϵ_1 and ϵ_2, respectively (step ②). Then each node locally perturbs its adjacency bit vector $\boldsymbol{B}$ into $\widetilde{\boldsymbol{B}}$ to satisfy ϵ_1-edge LDP, and perturbs its node degree d into $\widetilde{d}$ to satisfy ϵ_2-edge LDP (step ③). According to the composability of LDP, each node then satisfies ϵ-edge LDP. Note that this step is challenging as both $\boldsymbol{B}$ and d are correlated among nodes. For $\boldsymbol{B}$, the j-th bit of node i's adjacency bit vector is the same as the i-th bit of node j's adjacency bit vector. For d, whether i and j has an edge affects both degrees of i and j. Sections IV-B and IV-C solve this issue and send out the perturbed $\boldsymbol{B}$ and d, i.e., $\widetilde{\boldsymbol{B}}$ and $\widetilde{d}$. The data collector receives them from all nodes, aggregates them according to the mapping function $Map(\cdot)$ to obtain the estimated target metric $\widetilde{F}$, and further calibrates it to suppress estimation bias and improve accuracy (step ④). The resulted $\widetilde{F}$ is then used for graph analysis. The detailed implementation of LF-GDPR for steps ①③④ will be presented in Section IV. Note that the algorithms in steps ①②④ are parameterized, which can only be determined when the target graph metric F is specified.

Example III-B. **LF-GDPR against Facebook Privacy Scandal.** Facebook API essentially controls how a third-party app accesses the data of each individual user. To limit the

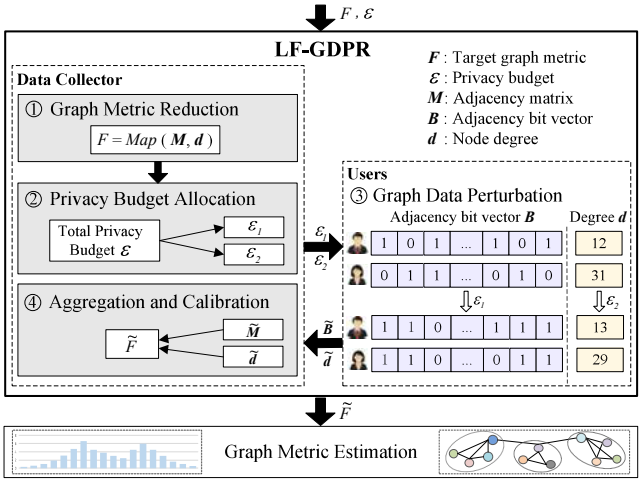


Fig. 1. An overview of LF-GDPR

access right of an average app (e.g., the one developed by Cambridge Analytica) while still supporting graph analytics, Facebook API should have a new permission rule that only allows such app to access the perturbed adjacency bit vector and degree of a user's friends list under ϵ_1 and ϵ_2-edge LDP, respectively. In the Cambridge Analytica case, the app is a personality test, so the app developer may choose structural similarity as the target graph metric and use the estimated value for the personality test. To estimate structural similarity, the app then implements steps ①②④ of LF-GDPR. On the user side, each user u has a privacy budget ϵ_u for her friends list. If $\epsilon_u \geq \epsilon_1 + \epsilon_2$, the user can grant access to this app for perturbed adjacency bit vector and degree; otherwise, the user simply ignores this access request.

IV. LF-GDPR: Implementation

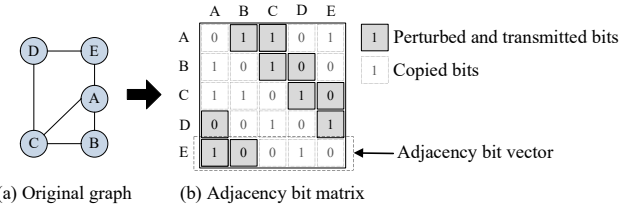
In this section, we present the implementation details of LF-GDPR. We first discuss graph metric reduction (step ①), followed by the perturbation protocols for adjacency bit vector and node degree, respectively (step ③). Then we elaborate on the aggregation and calibration algorithm (step ④).

A. Graph Metric Reduction

The reduction outputs a polynomial mapping function $Map(\cdot)$ from the target graph metric F to the adjacency matrix $\boldsymbol{M} = \{\boldsymbol{B}_1, \boldsymbol{B}_2, ..., \boldsymbol{B}_n\}$ and degree vector $\boldsymbol{d} = \{d_1, d_2, ..., d_n\}$, i.e., $F = Map(\boldsymbol{M}, \boldsymbol{d})$. Without loss of generality, we assume F is a polynomial of $\boldsymbol{M}$ and $\boldsymbol{d}$. That is, F is a sum of terms F_l, each of which is a multiple of $\boldsymbol{M}$ and $\boldsymbol{d}$ of some exponents. Since F and F_l are scalars, in each term F_l, we need functions f and g to transform $\boldsymbol{M}$ and $\boldsymbol{d}$ with exponents to scalars, respectively. Formally,

$$F = \sum\nolimits_l F_l = \sum\nolimits_l f_{\phi_l}(\boldsymbol{M}^{k_l}) \cdot g_{\psi_l}(\boldsymbol{d}), \tag{2}$$

where $\boldsymbol{M}^{k_l}$ is the k_l-th power of adjacency matrix $\boldsymbol{M}$ whose cell (i, j) denotes the number of paths between node i and j of length k_l, ϕ_l projects a matrix to a cell, a row, a column



(a) Original graph (b) Adjacency bit matrix

Fig. 2. Illustration of $RABV$ protocol

or a sub-matrix, and $f_{\phi_l}(\cdot)$ denotes an aggregation function f (e.g., sum) after projection ϕ_l. Likewise, ψ_l projects a vector to a scalar or a sub-vector, and $g_{\psi_l}(\cdot)$ denotes an aggregation function g after ψ_l.

As such, the metric reduction step is to determine k_l, $f_{\phi_l}(\cdot)$, and $g_{\psi_l}(\cdot)$ for each term F_l in Eq. 2.

B. Adjacency Bit Vector Perturbation

An intuitive approach, known as *Randomized Neighbor List* (*RNL*) [10], perturbs each bit of the vector independently by the classic Randomized Response (RR) [12]. Formally, given an adjacency bit vector $\boldsymbol{B} = \{b_1, b_2, ..., b_n\}$, and privacy budget ϵ_1, the perturbed vector $\widetilde{\boldsymbol{B}} = \{\widetilde{b}_1, \widetilde{b}_2, ..., \widetilde{b}_n\}$ is obtained as follows:

$$\widetilde{b}_i = \begin{cases} b_i & \text{w.p. } \frac{e^{\epsilon_1}}{1+e^{\epsilon_1}} \\ 1 - b_i & \text{w.p. } \frac{1}{1+e^{\epsilon_1}} \end{cases} \tag{3}$$

RNL is proved to satisfy ϵ_1-edge LDP for each user. However, **for undirected graphs, *RNL* can only achieve $2\epsilon_1$-edge LDP for the collector**, because the data collector witnesses the same edge perturbed twice and independently. Let $\widetilde{\boldsymbol{M}} = \{\widetilde{\boldsymbol{B}}_1, \widetilde{\boldsymbol{B}}_2, ..., \widetilde{\boldsymbol{B}}_n\}$ denote the perturbed adjacency matrix. The edge between node i and j appears in both $\widetilde{M}_{ij}$ and $\widetilde{M}_{ji}$, each perturbed with privacy budget ϵ_1. Then according to the theorem of composability, *RNL* becomes a $2\epsilon_1$-edge LDP algorithm for an undirected graph, which is less private. Furthermore, *RNL* requires each user to perturb and send all n bits in the adjacency bit vector to data collector, which incurs a high computation and communication cost.

To address the problems of *RNL*, we propose a more private and efficient protocol *Randomized Adjacency Bit Vector* ($RABV$) to perturb edges in undirected graphs. As shown in Fig. 2(b), the adjacency matrix is composed of n rows, each corresponding to the adjacency bit vector of a node. For the first $1 \leq i \leq \lfloor \frac{n}{2} \rfloor$ nodes, $RABV$ uses RR as in Eq.3 to perturb and transmit $t = \lfloor \frac{n}{2} \rfloor$ bits (i.e., bits in grey) — from the $(i+1)$-th bit to the $(i+1+t \mod n)$-th bit; for the rest nodes, $RABV$ uses RR to perturb and transmit $t = \lfloor \frac{n-1}{2} \rfloor$ bits in the same way. In essence, $RABV$ **perturbs one and only one bit** for each pair of symmetric bits in the adjacency matrix. The data collector can then obtain the whole matrix by copying bits in grey to their symmetric positions.

Following the same proof of RNL, $RABV$ is guaranteed to satisfy ϵ_1-edge LDP for the collector. Further, since each node only perturbs and transmits about half of the bits in an adjacency bit vector, $RABV$ significantly reduces computation and communication cost of *RNL*.

C. Node Degree Perturbation

Releasing the degree of a node while satisfying edge ϵ-LDP is essentially a centralized DP problem because all edges incident to this node, or equivalently, all bits in its adjacency bit vector, form a database and the degree is a count function. In the literature, *Laplace Mechanism* [2] is the predominant technique to perturb numerical function values such as counts. As such, LF-GDPR adopts it to perturb the degree d_i of each node i. According to the definition of edge LDP, two adjacency bit vectors $\boldsymbol{B}$ and $\boldsymbol{B'}$ are two neighboring databases if they differ in only one bit. As such, the sensitivity of degree (i.e., count function) is 1, and therefore adding Laplace noise $Lap(\frac{1}{\epsilon_2})$ to the node degree can satisfy ϵ_2-LDP. That is, $\widetilde{d}_i = d_i + Lap(\frac{1}{\epsilon_2})$.

Similar to perturbing adjacency bit vector, however, in the above naive approach the data collector witnesses two node degrees d_i and d_j perturbed independently, but they share the same edge between i and j. As DP or LDP does not refrain an adversary from possessing any background knowledge, in the worst case the collector already knows all edges except for this one. As such, witnessing the two node degrees d_i and d_j is degenerated to witnessing the edge between i and j twice and independently.

Unfortunately, the remedy that works for perturbing adjacency bit vector cannot be adopted here, as direct bit copy is not feasible for degree. As such, we take an alternative approach to increase the Laplace noise. The following theorem proves that if we add Laplace noise $Lap(\frac{2}{\epsilon_2})$ to every node degree, ϵ_2-LDP can be satisfied for the collector.

Theorem 4.1: A perturbation algorithm $\mathcal{A}$ satisfies ϵ_2-LDP for the collector if it adds Laplace noise $Lap(\frac{2}{\epsilon_2})$ to every node degree d_i, i.e., $\widetilde{d}_i = \mathcal{A}(d_i) = d_i + Lap(\frac{2}{\epsilon_2})$.

PROOF. Please refer to our technical report [14]. □

D. Aggregation and Calibration

Upon receiving the perturbed adjacency matrix $\widetilde{\boldsymbol{M}}$ and degree vector $\widetilde{\boldsymbol{d}}$,[1] the data collector can estimate the target graph metric $\widetilde{F}$ by aggregation according to Eq. 2 with a calibration function $\mathcal{R}(\cdot)$:

$$\widetilde{F} = \sum\nolimits_l \mathcal{R}\left(f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})\right) \cdot g_{\psi_l}(\widetilde{\boldsymbol{d}}) \tag{4}$$

The calibration function aims to suppress the aggregation bias of $\widetilde{\boldsymbol{M}}$ propagated by f_{ϕ_l}. On the other hand, no calibration is needed for $g_{\psi_l}(\widetilde{\boldsymbol{d}})$ as $\widetilde{\boldsymbol{d}}$ is already an unbiased estimation of $\boldsymbol{d}$, thanks to the Laplace Mechanism.

To derive $\mathcal{R}(\cdot)$, we regard $\mathcal{R}$ as the mapping between $f_{\phi_l}(\boldsymbol{M}^{k_l})$ and $f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})$. In other words, $\mathcal{R}$ estimates $f_{\phi_l}(\boldsymbol{M}^{k_l})$ after observing $f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l})$. Formally,

$$\mathcal{R} : f_{\phi_l}(\widetilde{\boldsymbol{M}}^{k_l}) \rightarrow f_{\phi_l}(\boldsymbol{M}^{k_l})$$

Further, the following theorem shows the accuracy guarantee of LF-GDPR.

[1]In the sequel, $\widetilde{\boldsymbol{d}}$ denotes the refined degree $\widetilde{\boldsymbol{d}}^*$ to simplify the notation.

Theorem 4.2: For a graph metric F and our estimation $\widetilde{F}$, with at least $1-\beta$ probability, we have

$$|F - \widetilde{F}| = O(\sqrt{\mathbb{E}[\widetilde{F}^2] \cdot log(1/\beta)}\,)$$

PROOF. Please refer to our technical report [14]. □

V. CONCLUSION

This paper presents a parameterized framework LF-GDPR for privacy-preserving graph metric estimation and analytics with local differential privacy. The building block is a user-side perturbation algorithm, and a collector-side aggregation and calibration algorithm. LF-GDPR simplifies the job of developing a practical LDP solution for a graph analysis task by providing a complete solution for all LDP steps. As for future work, we plan to extend LF-GDPR to more specific graph types, such as attributed graph and DAG. We will also evaluate the performance of LF-GDPR on other graph analysis tasks such as influential node analysis to demonstrate its wide applicability.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (Grant No: 91646203, 61941121, 61572413, U1636205, 61532010, 91846204 and 61532016), the Research Grants Council, Hong Kong SAR, China (Grant No: 15238116, 15222118 and C1008-16G) (corresponding author: Xiaofeng Meng).

REFERENCES

[1] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438. IEEE, 2013.
[2] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.
[3] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques* Morgan Kaufmann, 3 edition, 2011.
[4] M. Hay, C. Li, G. Miklau, and D. Jensen. Accurate estimation of the degree distribution of private networks. In *ICDM*, pages 169–178, 2009.
[5] Z. Jorgensen, T. Yu, and G. Cormode. Publishing attributed social graphs with formal privacy guarantees. In *SIGMOD*, pages 107–122, 2016.
[6] S. P. Kasiviswanathan, K. Nissim, S. Raskhodnikova, and A. Smith. Analyzing graphs with node differential privacy. In *TCC*, pages 457–476. Springer, 2013.
[7] W. Lu and G. Miklau. Exponential random graph estimation under differential privacy. In *KDD*, pages 921–930. ACM, 2014.
[8] T. Martin, X. Zhang, and M. Newman. Localization and centrality in networks. *Physical review E*, 90(5):052808, 2014.
[9] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84. ACM, 2007.
[10] Z. Qin, T. Yu, Y. Yang, I. Khalil, X. Xiao, and K. Ren. Generating synthetic decentralized social graphs with local differential privacy. In *CCS*, pages 425–438. ACM, 2017.
[11] B. Stephanie, Facebook Scandal a 'Game Changer' in Data Privacy Regulation. *Bloomberg*, Apr 8, 2018.
[12] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.
[13] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. SCAN: a structural clustering algorithm for networks. In *KDD*, pages 824–833. ACM, 2007.
[14] Q. Ye, H. Hu, M. H. Au, X. Meng, and X. Xiao. LF-GDPR: A framework for estimating graph metrics with local differential privacy. Technical report. http://www.eie.polyu.edu.hk/%7ehaibohu/papers/lfgdpr.pdf.
[15] Q. Ye, H. Hu, X. Meng, and H. Zheng. PrivKV: Key-value data collection with local differential privacy. In *S&P*, pages 317–331. IEEE, 2019.

A Unified Adversarial Learning Framework for Semi-supervised Multi-target Domain Adaptation

Xinle Wu[1,2], Lei Wang[2(✉)], Shuo Wang[1], Xiaofeng Meng[1], Linfeng Li[2,3], Haitao Huang[4], Xiaohong Zhang[5], and Jun Yan[2]

[1] School of Information, Renmin University of China, Beijing, China
{xinle.wu,shuowang,xfmeng}@ruc.edu.cn
[2] Yidu Cloud (Beijing) Technology Co., Ltd., Beijing, China
{lei.wang01,Linfeng.Li,jun.yan}@yiducloud.cn
[3] Institute of Information Science, Beijing Jiaotong University, Beijing, China
[4] The second Department of Neurology, Liaoning People's Hospital, Shenyang, China
[5] The Fourth Affiliated Hospital, China Medical University, Taichung, Taiwan, R.O.C.

Abstract. Machine learning algorithms have been criticized as difficult to apply to new tasks or datasets without sufficient annotations. Domain adaptation is expected to tackle this problem by establishing knowledge transfer from a labeled source domain to an unlabeled or sparsely labeled target domain. Most existing domain adaptation models focus on the single-source-single-target scenario. However, the pairwise domain adaptation approaches may lead to suboptimal performance when there are multiple target domains available, because the information from other related target domains is not being utilized. In this work, we propose a unified semi-supervised multi-target domain adaptation framework to implement knowledge transfer among multiple domains (a single source domain and multiple target domains). Specifically, we aim to learn an embedded space and minimize the marginal probability distribution differences among all domains in the space. Meanwhile, we introduce Prototypical Networks to perform classification, and extend it to semi-supervised settings. On this basis, we further align the conditional probability distributions among the domains by generating pseudo-labels for the unlabeled target data and training the model with bootstrapping method. Extensive sentiment analysis experiments show that our approach significantly outperforms several state-of-the-art methods.

Keywords: Domain adaptation · Adversarial learning · Semi-supervised · Prototypical networks · Self-training · Sentiment analysis

1 Introduction

Supervised learning algorithms have achieved great success in many fields with the availability of large quantities of labeled data. However, it is costly and time-consuming to annotate such large-scale training data for new tasks or datasets.

Y. Nah et al. (Eds.): DASFAA 2020, LNCS 12112, pp. 419–434, 2020.
https://doi.org/10.1007/978-3-030-59410-7_29

A naive idea is directly applying the model trained on a labeled source domain to the related and sparsely labeled target domain. Unfortunately, the model usually fails to perform well in the target domain due to domain shifts [24]. Domain adaptation (DA) is proposed to address this problem by transferring knowledge from a labeled source domain to a sparsely labeled target domain.

Existing DA methods can be divided into: supervised DA (SDA) [14,20,26], semi-supervised DA (SSDA) [11,21,23,31], and unsupervised DA (UDA) [4,5,9, 15]. SDA methods assume that there are some labeled data in the target domain, and perform DA algorithms only use the labeled data. Conversely, UDA methods do not need any target data labels, but they require large amounts of unlabeled target data to align the distributions between domains. Considering that it is cheap to annotate a small number of samples and a few labeled data often leads to significant performance improvements, we focus on SSDA which exploits both labeled and unlabeled data in target domains.

Typical DA methods are designed to embed the data from the source and target domains into a common embedding space, and align the marginal probability distributions between the two domains. There are two approaches to achieve this, adversarial training [4,10,20,27] and directly minimizing the distance between the two distributions [15,18,28,35]. Both of the methods can generate domain-invariant feature representations for input data, and the representations from the source domain are used to train a classifier, which is then generalized to the target domain. However, only aligning the marginal distributions is not sufficient to ensure the success of DA [4,5,12,33], because the conditional probability distributions between the source and target domains may be different.

Most DA algorithms focus on the single-source-single-target setting. However, in many practical applications, there are multiple sparsely labeled target domains. For example, in the sentiment analysis task of product reviews, we can take the reviews of Books, DVDs, Electronics and Kitchen appliances as different domains. If we only have access to sufficient labeled data of Book reviews (source domain), and hope to transfer knowledge to the other domains, then each of the other domains can be seen as a target domain. In this case, pairwise adaptation approaches may be suboptimal, especially when there are shared features between the source and multiple target domains or the source and the target domain are associated through another target domain [9]. This is due to that these methods fail to leverage the knowledge from other relevant target domains. In addition, considering the distribution differences among multiple target domains, simply merging multiple target domains into a single one may not be the optimal solution.

To address these problems, we propose semi-supervised multi-target domain adaptation networks (MTDAN). Specifically, we use a shared encoder to extract the common features shared by all domains, and a private encoder to extract the domain-specific features of each domain. For feature representations generated by the two encoders, we train a domain discriminator to distinguish which domain they come from. To ensure that the shared representation is domain-invariant, the shared encoder is encouraged to generate the representation

cannot be correctly distinguished by the domain discriminator. Given that there are only a few labeled data in each target domain, we introduce Prototypical Networks to perform classification, which is more superior than deep classifiers in few-shot scenarios [25]. We further leverage unlabeled data to refine prototypes, and extend Prototypical Networks to semi-supervised scenarios. Moreover, we utilize the self-training algorithm to exploit unlabeled target data, and we show that it can also align the class-conditional probability distributions among multiple domains.

Contributions. Our contributions are: a) We propose a unified adversarial learning framework for semi-supervised multi-target DA. b) We show that the prototype-based classifier can achieve better performance than the deep classifier when target domains have only a few labeled data and large amounts of unlabeled data. c) We show that the self-training algorithm can effectively align the class-conditional probability distributions among multiple domains. d) Our method outperforms several state-of-the-art DA approaches on sentiment analysis dataset.

2 Related Work

Domain Adaptation. Numerous domain adaptation approaches have been proposed to solve domain shift [29]. Most of them seek to learn a shared embedded space, in which the representations of source domain and target domain cannot be distinguished [27]. Based on that, the classifier trained with labeled source data can be generalized to the target domain. There are two typical ways to learn cross-domain representations: directly minimizing the distance between two distributions [15,17,18] and adversarial learning [6,7,26,27].

For the first method, several distance metrics have been proposed to measure the distance between source and target distributions. One common distance metric is the Maximum Mean Discrepancy (MMD) [2], which computes the norm of the difference between two domain means in the reproducing Kernel Hilbert Space (RKHS). Specifically, the DDC method [28] used both MMD and regular classification loss on the source to learn representations that are discriminative and domain invariant. The Deep Adaptation Network (DAN) [15] applied MMD to the last full connected layers to match higher order statistics of the two distributions. Most recently, [18] proposed to reduce domain shift in joint distributions of the network activation of multiple task-specific layers. Besides, Zellinger et al. proposed Center Moment Discrepancy (CMD) [32] to diminish the domain shift by aligning the central moment of each order across domains.

The other method is to optimize the source and target mappings using adversarial training. The idea is to train a domain discriminator to distinguish whether input features come from the source or target, whereas the feature encoder is trained to deceive the domain discriminator by generating representations that cannot be distinguished. [6] proposed the gradient reversal algorithm (ReverseGrad), which directly maximizes the loss of the domain discriminator by reversing its gradients. DRCN in [8] takes a similar approach in addition to learning to

reconstruct target domain images. [3] enforced these adversarial losses in a shared feature space, while learned a private feature space for each domain to avoid the contamination of shared representations.

[4,27,33] argued that only aligning the marginal probability distributions between the source and target is not enough to guarantee successful domain adaptation. [16] proposed to align the marginal distributions and conditional distributions between the source and target simultaneously. [20] extended the domain discriminator to predict the domain and category of the embedded representation at the same time to align the joint probability distributions of input and output, and achieved a leading effect in the supervised domain adaptive scene. [5] proposed to align the class-conditional probability distributions between the source and target.

Recently, Zhao et al. [34] introduced an adversarial framework called MDAN, which is used for multi-source-single-target domain adaption. They utilized a multi-class domain discriminator to align the distributions between multiple source and a target domain. [9] proposed an information theoretic approach to solve unsupervised multi-target domain adaptation problem, which maximizes the mutual information between the domain labels and domain-specific features, while minimizes the mutual information between the the domain labels and the domain-invariant features. Unlike their approach, we base our method on self-training rather than entropy regularization. Moreover, we introduce prototypical networks to perform classification, which is more effective than deep classifiers in SSDA scenarios.

Semi-supervised Learning. Recently, some works treat domain adaptation as a semi-supervised learning task. [11] proposed a Domain Adaptive Semi-supervised learning framework (DAS) to jointly perform feature adaptation and semi-supervised learning. [21] applied a co-training framework for semi-supervised domain adaptation, in which the shared classifier and the private classifier boost each other to achieve better performance. [22] re-evaluated classic general-purpose bootstrapping approaches under domain shift, and proved that the classic bootstrapping algorithms make strong baselines on domain adaptation tasks.

3 Preliminaries

In this section, we introduce the notations and definitions related to single-source-multi-target DA.

Notations. We use $\mathcal{D}$ to denote a domain, which consists of an m-dimensional feature space $\mathcal{X}$ and a marginal probability distribution $P(\mathrm{x})$, i.e., $\mathcal{D} = \{\mathcal{X}, P(\mathrm{x})\}$, where $\mathrm{x} \in \mathcal{X}$. We use $\mathcal{T}$ to denote a task which consists of a C-cardinality label set $\mathcal{Y}$ and a conditional probability distribution $P(y|\mathrm{x})$, i.e., $\mathcal{T} = \{\mathcal{Y}, P(y|\mathrm{x})\}$, where $y \in \mathcal{Y}$.

Problem Formulation (Single-Source-Multi-target Domain Adaptation). Let $\mathcal{D}_s = \{(\mathrm{x}_l^s, y_l^s)\}_{l=1}^{n_s}$ be a labeled source domain where n_s is the

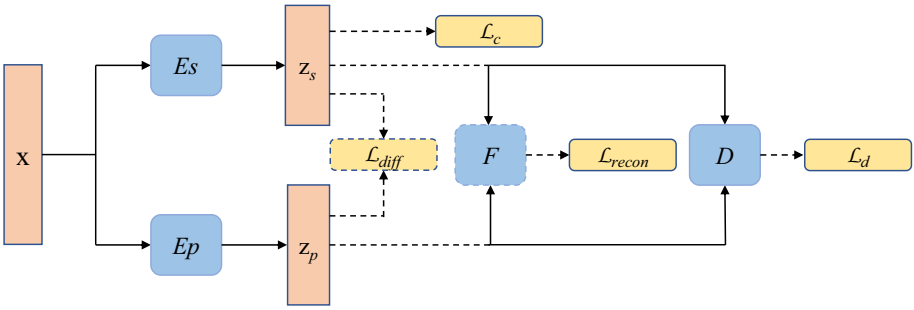


Fig. 1. The network structure of the proposed framework. The shared encoder E_s captures the common features shared among domains, while the private encoder E_p captures the domain-specific features. The shared decoder F reconstruct the input samples by using both the shared and private representations. The domain classifier D learns to distinguish which domain the input representations come from. The orthogonality constraint loss $\mathcal{L}_{diff}$ encourages E_s and E_p to encode different aspects of the inputs. The prototype-based classifier is computed on-the-fly, and the classification loss $\mathcal{L}_c$ is only used to optimize E_s.

number of labeled samples and let $\mathcal{D}_t = \{\mathcal{D}_{t_i}\}_{i=1}^{K}$ be multiple sparsely labeled target domains where $\mathcal{D}_{t_i} = \{(\mathrm{x}_l^{t_i}, y_l^{t_i})\}_{l=1}^{n_{l_i}} \bigcup \{\mathrm{x}_u^{t_i}\}_{u=1}^{n_{u_i}}$, K is the number of target domains, and $n_{l_i} (n_{l_i} \ll n_s)$ and $n_{u_i} (n_{u_i} \gg n_{l_i})$ refer to the number of labeled and unlabeled samples of i-th target domain respectively. We assume that all domains share the same feature space $\mathcal{X}$ and label space $\mathcal{Y}$, but the marginal probability distributions and the conditional probability distributions of source domain and multiple target domains are different from each other. The goal is to learn a classifier using the labeled source data and a few labeled target data, that generalizes well to the target domain.

4 Methodology

In this section, we describe each component and the corresponding loss function of the proposed framework in detail.

4.1 Proposed Approach

Our model consists of four components as shown in Fig. 1. A shared encoder E_s is trained to learn cross-domain representations, a private encoder E_p is trained to learn domain-specific representations, a shared decoder F is trained to reconstruct the input sample, and a discriminator D is trained to distinguish which domain the input sample comes from. Task classification is performed by calculating the distance from the domain-invariant representations to prototype representations of each label class.

Domain-Invariant and Domain-Specific Representations. We seek to extract domain-invariant (shared) and domain-specific (private) representations

for each input $\mathbf{x}$ simultaneously. In our model, the shared encoder E_s and the private encoder E_p learn to generate the above two representations respectively:

$$\begin{aligned} \mathbf{z}_s &= E_s(\mathbf{x}, \boldsymbol{\theta}_s) \\ \mathbf{z}_p &= E_p(\mathbf{x}, \boldsymbol{\theta}_p) \end{aligned} \tag{1}$$

Here, $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_p$ refer to the parameters of E_s and E_p respectively, $\mathbf{z}_s$ and $\mathbf{z}_p$ refer to the shared and private representations of the input $\mathbf{x}$ respectively. Note that E_s and E_p can be MLP, CNN or LSTM encoders, depending on different tasks and datasets.

Reconstruction. In order to avoid information loss during the encoding, we reconstruct input samples with both shared and private representations. We use $\hat{\mathbf{x}}$ to denote the reconstruction of the input $\mathbf{x}$, which is generated by decoder F:

$$\hat{\mathbf{x}} = F(\mathbf{z}_s + \mathbf{z}_p, \boldsymbol{\theta}_f), \tag{2}$$

where $\boldsymbol{\theta}_f$ are the parameters of F. We use mean square error to define the reconstruction loss $\mathcal{L}_{Recon}$, which is applied to all domains:

$$\mathcal{L}_{Recon} = \frac{\lambda_r}{N} \sum_{i=1}^{N} \frac{1}{C} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2, \tag{3}$$

where C is the dimension of the input $\mathbf{x}$, N is the total number of samples in all domains, $\mathbf{x}_i$ refers to the i-th sample, λ_r is the hyper-parameter controlling the weight of the loss function, and $\|\cdot\|_2^2$ is the squared L_2-norm.

Orthogonality Constraints. To minimize the redundancy between shared and private representations, we introduce orthogonality constraints to encourage the shared and private encoders to encode different aspects of inputs. Specifically, we use $\mathbf{H}_s$ to denote a matrix, each row of which corresponds to the shared representation of each input $\mathbf{x}$. Similarly, let $\mathbf{H}_p$ be a matrix, each row of which corresponds to the private representation of each input $\mathbf{x}$. The corresponding loss function is:

$$\mathcal{L}_{Diff} = \lambda_{diff} \|\mathbf{H}_s^\top \mathbf{H}_p\|_F^2, \tag{4}$$

where λ_{diff} is the scale factor, $\|\cdot\|_F^2$ is the squared Frobenius norm.

Adversarial Training. The goal of adversarial training is to regularize the learning of the shared encoder E_s, so as to minimize the distance of distributions among source and multiple target domains. After that, we can apply the source classification model directly to the target representations. Therefore, we first train a domain discriminator D with the domain labels of the shared and private representations (since we know which domain each sample comes from, it is obvious that we can generate a domain label for each sample). The discriminator D is a multi-class classifier designed to distinguish which domain the

Algorithm 1. MTDAN Algorithm

Input: labeled source domain examples L_s, labeled multi-target domain examples $L_t = \{L_{t_i}\}_{i=1}^{K}$, unlabeled multi-target domain examples $U_t = \{U_{t_i}\}_{i=1}^{K}$
Hyper-parameters: coefficients for different losses: $\lambda_r, \lambda_d, \lambda_c, \lambda_{diff}$, mini-batch size b, learning rate η

initialize $\boldsymbol{\theta}_s, \boldsymbol{\theta}_p, \boldsymbol{\theta}_f, \boldsymbol{\theta}_d$
repeat
  repeat
    Sample a mini-batch from $\{L_s, L_t\}$
    Train F by minimizing $\mathcal{L}_{Recon}$
    Train D by minimizing $\mathcal{L}_D$
    Train E_p by minimizing $\mathcal{L}_P$
    Train E_s by minimizing $\mathcal{L}_S$
  until Convergence
  Apply Eq.(9) to label U_t
  Select the most confident p positive and n negative predicted examples U_t^l from U_t
  Remove U_t^l from U_t
  Add examples U_t^l and their corresponding labels to L_t
until obtain best performance on the developing dataset

input representation comes from. Thus, D is optimized according to a standard supervised loss, defined below:

$$\mathcal{L}_D = \mathcal{L}_{D_p} + \mathcal{L}_{D_s}, \tag{5}$$

$$\mathcal{L}_{D_p} = -\frac{\lambda_d}{N} \sum_{i=1}^{N} \mathbf{d}_i^\top \log D(E_p(\mathbf{x}_i, \boldsymbol{\theta}_p), \boldsymbol{\theta}_d), \tag{6}$$

$$\mathcal{L}_{D_s} = -\frac{\lambda_d}{N} \sum_{i=1}^{N} \mathbf{d}_i^\top \log D(E_s(\mathbf{x}_i, \boldsymbol{\theta}_s), \boldsymbol{\theta}_d), \tag{7}$$

where $\mathbf{d}_i$ is the one-hot encoding of the i-th sample's domain label, $\boldsymbol{\theta}_d$ is the parameter of D, and λ_d is the scale factor.

Second, we train the shared encoder E_s to fool the discriminator D by generating cross-domain representations. We guarantee this by adding $-\mathcal{L}_{D_s}$ to the loss function of the shared encoder E_s. On the other hand, we hope the private encoder only extracts domain-specific features. Thus, we add $\mathcal{L}_{D_p}$ to the loss function of E_p to generate representations that can be distinguished by D.

Prototypical Networks for Task Classification. The simplest way to classify the target samples is to train a deep classifier, however, it may only achieve

suboptimal performance as we can see in Table 1. The reason is that there are only a few labeled samples in each target domain, which is not enough to fine-tune a deep classifier with many parameters, so that the classifier is easy to over fit the source labeled data. Although we could generate pseudo-labeled data for target domains, the correctness of the pseudo-labels can not be guaranteed due to the poor performance of the deep classifier.

To efficiently utilize the labeled samples in target domains, we refer to the idea of prototypical networks [25]. Prototypical networks assume that there is a prototype in the latent space for each class, and the projections of samples belonging to this class cluster around the prototype. The classification is then performed by computing the distances to prototype representations of each class in the latent space. By reducing parameters of the model, the prototype-based classifier can achieve better performance than the deep classifier when labeled samples are insufficient. Note that we refine prototypes during self-training by allowing unlabeled samples with pseudo-labels to update the prototypes. Specifically, we compute the average of shared representations belonging to each class in a batch as prototypes:

$$\mathbf{c}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} E_s(\mathbf{x}_i, \boldsymbol{\theta}_s), \tag{8}$$

where n_k is the number of samples belonging to class k in a batch. Then we calculate a distribution by applying softmax function to distances between a shared representation with a prototype:

$$p(y = k|\mathbf{x}) = \frac{exp(-d(\mathbf{z}_s, \mathbf{c}_k))}{\sum_{k'} exp(-d(\mathbf{z}_s, \mathbf{c}'_k))}, \tag{9}$$

where $d(\cdot)$ is a distance measure function. We use the squared Euclidean distance in this work. The classification loss is defined as:

$$\mathcal{L}_C = -\lambda_c \log p(y = k|\mathbf{x}), \tag{10}$$

where λ_c is the scale factor.

Self-training for Conditional Distribution Adaptation. As described in [5,19], only aligning the marginal probability distributions between source and target is not enough to guarantee successful domain adaptation. Because this only enforces alignment of the global domain statistics with no class specific transfer. Formally, we can achieve $P_s(E_s(\mathrm{x}_s)) \approx P_{t_i}(E_s(\mathrm{x}_{t_i}))$ by introducing adversarial training, but $P_s(y_s|E_s(\mathrm{x}_s)) \neq P_{t_i}(y_{t_i}|E_s(\mathrm{x}_{t_i}))$ may still hold, where $P_s(y_s|E_s(\mathrm{x}_s))$ can be regarded as the classifier trained with source data.

Here, we tackle this problem by further reducing the difference of conditional probability distributions among source domain and target domains. In practice, we replace conditional probability distributions with class-conditional probability distributions, because the posterior probability is quite involved [16].

However, it is nontrivial to adapt class-conditional distributions, as most of the target samples are unlabeled. We address this problem by producing pseudo-labels for unlabeled target samples, and train the whole model in a bootstrapping way. As we perform more learning iterations, the number of target samples with correct pseudo-labels grows and progressively enforces distributions to align class-conditionally.

To be specific, we first train our model on labeled source and target samples. Then, we use the model to generate a probability distribution over classes for each unlabeled target sample. If the probability of a sample on a certain class is higher than a predetermined threshold τ, the sample would be added to the training set with the class as its pseudo-label.

Loss Function and Model Training. We alternately optimize the four modules of our model.

For E_p, the goal of training is to minimize the following loss:

$$\mathcal{L}_P = \mathcal{L}_{Recon} + \mathcal{L}_{Diff} + \mathcal{L}_{D_p} \tag{11}$$

For E_s, the goal of training is to minimize the following loss:

$$\mathcal{L}_S = \mathcal{L}_{Recon} + \mathcal{L}_{Diff} - \mathcal{L}_{D_s} + \mathcal{L}_C \tag{12}$$

For F and D, the losses are $\mathcal{L}_{Recon}$ and $\mathcal{L}_D$, respectively. The detailed training process is shown in algorithm 1.

5 Experiments

5.1 Dataset

We evaluate our proposed model on the Amazon benchmark dataset [1]. It is a sentiment classification dataset[1], which contains Amazon product reviews from four different domains: Books (B), DVD (D), Electronics (E), and Kitchen appliances (K). We remove reviews with neutral labels and encode the remaining reviews into 5000 dimensional feature vectors of unigrams and bigrams with binary labels indicating sentiment.

We pick two product as the source domain and the target domain in turn, and the other two domains as the auxiliary target domains, so that we construct 12 single-source-three-target domain adaptation tasks. For each task, the source domain contains 2,000 labeled examples, and each target domain contains 50 labeled examples and 2,000 unlabeled examples. To fine-tune the hyper-parameters, we randomly select 500 labeled examples from the target domain as the developing dataset.

[1] https://www.cs.jhu.edu/mdredze/datasets/sentiment/.

5.2 Compared Method

We compare MTDAN with the following baselines:

(1) **ST:** The basic neural network classifier without any domain adaptation trained on the labeled data of the source domain and the target domain.
(2) **CoCMD:** This is the state-of-the-art pairwise SSDA method on the Amazon benchmark dataset [21]. The shared encoder, private encoder and reconstruction decoder used in this model are the same as ours.
(3) **MTDA-ITA:** This is the state-of-the-art single-source-multi-target UDA method on three benchmark datasets for image classification [9]. We implemented the framework and extend it to semi-supervised DA method. The shared encoder, private encoder, reconstruction decoder and domain classifier used in this model are the same as ours.
(4) **c-MTDAN:** We combine all the target domains into a single one, and train it using MTDAN. Similarly, we also report the performance of c-CoCMD and c-MTDA-ITA.
(5) **s-MTDAN:** We do not use any auxiliary target domains, and train MTDAN on each source-target pair.

5.3 Implementation Details

Considering that each input sample in the dataset is a tf-idf feature vector without word ordering information, we use a multilayer perceptron (MLP) with an input layer (5000 units) and one hidden layer (50 units) and sigmoid activation functions to implement both shared and private encoders. The reconstruction decoder consists of one dense hidden layer (2525 units), tanh activation functions, and relu output functions. The domain discriminator is composed of a softmax layer with n-dimensional outputs, where n is the number of the source and target domains. For MTDA-ITA, we follow the framework proposed by [9], and use the above modules to replace the original modules in the framework. Besides, the task classifier for MTDA-ITA is a fully connected layer with softmax activation functions.

The network is trained with Adam optimizer [13] and with learning rate 10^{-4}. The mini-batch size is 50. The hyper-parameters $\lambda_r, \lambda_d, \lambda_c$ and λ_{diff} are empirically set to $1.0, 0.5, 0.1$ and 1.0 respectively. The threshold τ for producing pseudo-labels is set to 0.8. Following previous studies, we use classification accuracy metric to evaluate the performances of all approaches.

5.4 Results

The performances of the proposed model and other state-of-the-art methods are shown in Table 1. Key observations are summarized as follows. (1) The proposed model MTDAN achieves the best results in almost all tasks, which proves the effectiveness of our approach. (2) c-CoCMD has worse performance in all tasks compared with CoCMD, although c-CoCMD exploits labeled and unlabeled data from auxiliary target domains for training. Similar observation can

also be observed by comparing MTDA-ITA with c-MTDA-ITA and MTDAN with c-MTDAN. This demonstrates that simply combine all target domains into a single one is not an effective method to solve the multi-target DA problem. (3) Our model outperforms CoCMD by an average of nearly 2.0%, which indicates that our model can effectively leverage the labeled and unlabeled data from multiple target domains. Similarly, our model performs better than its variant, s-MTDAN, which does not leverage the data from auxiliary target domains. This also shows that it is helpful to mine knowledge from auxiliary target domains. (4) Although MTDA-ITA is also a multi-target domain adaptation method, its performance is worse than that of MTDAN. This can be due to (i) self-training is a superior method than entropy regularization to exploit unlabeled target data, (ii) the prototype-based classifier is more efficient than the deep classifier in semi-supervised scenarios, (iii) we introduce orthogonality constraints to further reduce the redundancy between shared and private representations. (5) In the K→E task, MTDAN performs slightly worse than s-MTDAN. This can be explained that domain K is closer to domain E than the other domains as shown in Fig. 2 (a), and MTDAN leads to negative transfer when using relevant target domains to help domain adaptation. (6) s-MTDAN outperforms CoCMD in 9 of the 12 tasks, note that both of them do not use the auxiliary domains. This indicates that our model is more effective than CoCMD in pairwise domain adaptation task. (7) All models achieve better performance than the basic ST model, which demonstrates that domain adaptation methods are crucial when there exist a domain gap between the source domain and the target domain.

Table 1. Average classification accuracy with 5 runs on target domain testing dataset. The best is shown in bold. c-X: combining all target domains into a single one and performing pairwise domain adaptation with model X. s-X: performing pairwise domain adaptation between the original source and target domains with model X

Method	B→D	B→E	B→K	D→B	D→E	D→K	E→B	E→D	E→K	K→B	K→D	K→E
ST	81.6	75.8	78.2	80.0	77.0	80.4	74.7	75.4	85.7	73.8	76.6	85.3
CoCMD	83.1	83.0	85.3	81.8	83.4	85.5	76.9	78.3	87.3	77.2	79.6	87.2
c-CoCMD	82.7	82.2	84.5	80.6	83.0	84.8	76.3	77.6	87.1	75.9	79.4	86.1
MTDA-ITA	83.8	83.2	83.7	81.8	83.6	85.4	76.6	78.9	87.7	77.0	78.8	86.8
c-MTDA-ITA	83.3	82.3	83.2	81.4	83.0	85.0	76.0	79.3	87.6	76.7	78.5	87.0
s-MTDAN	83.3	83.9	84.7	81.6	83.7	84.7	78.0	80.2	87.9	78.6	79.9	**87.8**
c-MTDAN	84.0	84.0	85.5	81.7	84.3	85.9	80.2	80.7	88.1	79.8	80.5	87.0
MTDAN	**84.5**	**84.3**	**86.0**	**82.3**	**85.3**	**87.2**	**80.5**	**81.2**	**88.9**	**80.0**	**80.9**	87.4

5.5 Ablation Studies

We performed ablation experiments to verify the importance of each component of our proposed model. We report the results of removing orthogonality

constraints loss (set λ_{diff}=0), self-training process, the prototype-based classifier (replaced by the deep classifier) respectively.

As we can see from Table 2, removing each of the above components causes performance degradation. To be specific, disabling self-training degrades the performance to the greatest extent, with an average decrease of 5.1%, which shows the importance of mining information from the unlabeled data of target domains. Similarly, replacing prototype-based classifiers with deep classifiers also leads to performance degradation, with an average decrease of 1.4%, which shows that the prototype-based classifiers is more effective than deep classifiers in semi-supervised scenarios. Besides, disabling the orthogonality constraints loss leads to a performance degradation of 0.7%, which indicates that encouraging the disjoint of shared and private representations can make the shared feature space more common among all domains.

We did not test the performance degradation caused by disabling reconstruction loss and multi-class adversarial training loss, because they have been proved in previous work [3,9]. To summarize, each of the proposed components helps improve classification performance, and using all of them brings the best performance.

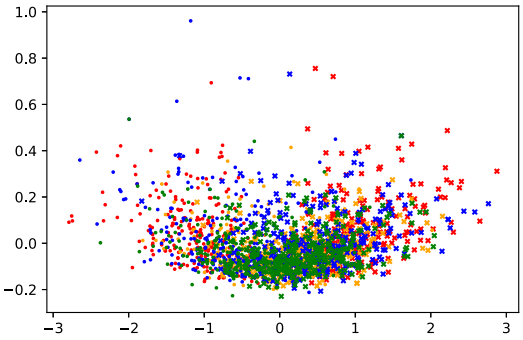
Table 2. Ablations. Performance of the proposed model when one component is removed or replaced. woDiff means without orthogonality constraints loss, woSelf means without self-training procedures, woProto means replace the prototype-based classifier with the deep classifier.

Method	B→D	B→E	B→K	D→B	D→E	D→K	E→B	E→D	E→K	K→B	K→D	K→E
MTDAN-$woDiff$	84.1	83.6	85.9	81.5	85.0	86.7	79.7	80.5	88.3	79.6	80.0	87.0
MTDAN-$woSelf$	82.8	77.6	80.0	81.6	78.8	81.5	74.8	75.6	86.7	74.5	77.3	86.7
MTDAN-$woProto$	83.3	83.8	84.1	81.8	83.8	86.9	79.2	78.9	87.9	78.0	80.3	86.4
MTDAN	**84.5**	**84.3**	**86.0**	**82.3**	**85.3**	**87.2**	**80.5**	**81.2**	**88.9**	**80.0**	**80.9**	**87.4**

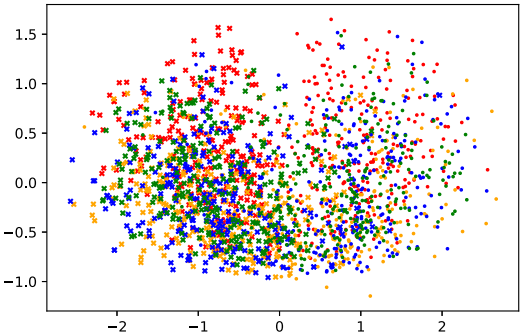
5.6 Feature Visualization

In order to understand the behavior of the proposed model intuitively, we project the shared and private encoder outputs into two-dimensional space with principle component analysis (PCA) [30] and visualize them. For comparison, we also show the visualization result of the basic ST model. Due to space constraints, we only show the visualization results of MTDAN with B as the source domain, E as the target domain, D and K as the auxiliary target domains. The results are shown in Fig. 2.

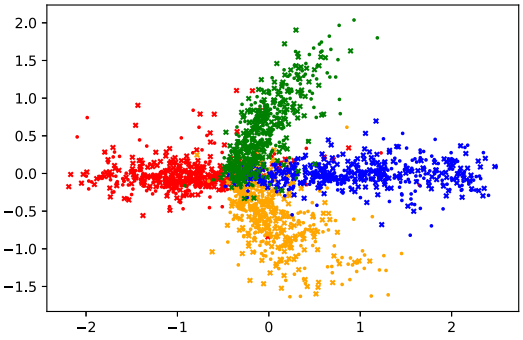
Figure 2 (a) shows the encoder output distribution of the ST model. As we can see, the distributions of domain B and domain D (called group 1) are similar and the distributions of domain E and domain D (called group 2) are similar, while the distributions of cross-group domains are relatively different. That's why the ST model gets worse classification performance when the source domain



(a) ST



(b) MTDAN-shared



(c) MTDAN-private

Fig. 2. Feature visualization for the embedding of source and target data. The red, blue, yellow and green symbols denote the samples from B, D, E and K respectively. The symbol 'x' is used for positive samples and '.' is for negative samples. (a) the distribution of the encoder output of ST, (b) the distribution of shared representations of MTDAN, (c) the distribution of private representations of MTDAN. For ST and MTDAN, we take B as the source domain and D, E and K as the target domains.

and the target domain belong to different groups. Besides, there is no obvious boundary between positive and negative samples, which is consistent with the poor performance of the ST model.

Figure 2 (b) shows the distribution of the shared encoder output of the MTDAN model. We can see that the shared representations of the source and target domains are very close, which demonstrates that our model can effectively align the marginal distributions among the source and multiple target domains. Meanwhile, for each class of samples, the shared representations of the source and target domains are also very close, which demonstrates that our model can effectively align the class-conditional distributions among multiple domains. Comparing (a) and (b), we can find that the boundary of positive and negative samples in (b) is more obvious than that in (a), which means the shared representations of MTDAN model have superior class separability.

Figure 2 (c) shows the distribution of the private encoder output of the MTDAN model. We can see that the private representations have good domain separability, partially because the domain discriminator D encourages the private encoder E_p to generate domain-specific feature representations.

6 Conclusion

In this paper, we propose MTDAN, a unified framework for semi-supervised multi-target domain adaptation. We utilize multi-class adversarial training to align the marginal probability distributions among source domain and multiple target domains. Meanwhile, we perform self-training on target unlabeled data to align the conditional probability distributions among the domains. We further introduce Prototypical Networks to replace the deep classifiers, and extend it to semi-supervised scenarios. The experimental results on sentiment analysis dataset demonstrate that our method can effectively leverage the labeled and unlabeled data of multiple target domains to help the source model achieve generalization, and is superior to the existing methods. The proposed framework could be used for other domain adaptation tasks, and we leave this as our future work.

Acknowledgment. This work was supported by National Natural Science Foundation of China (Grant No: 91646203, 91846204, 61532010, 61941121, 61532016 and 61762082). The corresponding author is Xiaofeng Meng.

References

1. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 440–447 (2007)
2. Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.P., Schölkopf, B., Smola, A.J.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics **22**(14), e49–e57 (2006)

3. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. In: Advances in Neural Information Processing Systems, pp. 343–351 (2016)
4. Cicek, S., Soatto, S.: Unsupervised domain adaptation via regularized conditional alignment. arXiv preprint arXiv:1905.10885 (2019)
5. Gabourie, A.J., Rostami, M., Pope, P.E., Kolouri, S., Kim, K.: Learning a domain-invariant embedding for unsupervised domain adaptation using class-conditioned distribution alignment. In: 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 352–359. IEEE (2019)
6. Ganin, Y., Lempitsky, V.S.: Unsupervised domain adaptation by backpropagation. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, pp. 1180–1189 (2015)
7. Ganin, Y., et al.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**(1), 2030–2096 (2016)
8. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 597–613. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_36
9. Gholami, B., Sahu, P., Rudovic, O., Bousmalis, K., Pavlovic, V.: Unsupervised multi-target domain adaptation: An information theoretic approach. arXiv preprint arXiv:1810.11547 (2018)
10. Guo, J., Shah, D.J., Barzilay, R.: Multi-source domain adaptation with mixture of experts. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018, pp. 4694–4703 (2018)
11. He, R., Lee, W.S., Ng, H.T., Dahlmeier, D.: Adaptive semi-supervised learning for cross-domain sentiment classification. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018, pp. 3467–3476 (2018)
12. Hosseini-Asl, E., Zhou, Y., Xiong, C., Socher, R.: Augmented cyclic adversarial learning for low resource domain adaptation. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019 (2019)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015, Conference Track Proceedings (2015)
14. Koniusz, P., Tas, Y., Porikli, F.: Domain adaptation by mixture of alignments of second-or higher-order scatter tensors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4478–4487 (2017)
15. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015, pp. 97–105 (2015)
16. Long, M., Wang, J., Ding, G., Sun, J., Yu, P.S.: Transfer feature learning with joint distribution adaptation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2200–2207 (2013)
17. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Unsupervised domain adaptation with residual transfer networks. In: Advances in Neural Information Processing Systems, pp. 136–144 (2016)
18. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2208–2217 (2017). JMLR. org

19. Luo, Z., Zou, Y., Hoffman, J., Fei-Fei, L.F.: Label efficient learning of transferable representations across domains and tasks. In: Advances in Neural Information Processing Systems, pp. 165–177 (2017)
20. Motiian, S., Jones, Q., Iranmanesh, S., Doretto, G.: Few-shot adversarial domain adaptation. In: Advances in Neural Information Processing Systems, pp. 6670–6680 (2017)
21. Peng, M., Zhang, Q., Jiang, Y.g., Huang, X.J.: Cross-domain sentiment classification with target domain specific information. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2505–2513 (2018)
22. Ruder, S., Plank, B.: Strong baselines for neural semi-supervised learning under domain shift. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, 15–20 July 2018, Volume 1: Long Papers, pp. 1044–1054 (2018)
23. Saito, K., Kim, D., Sclaroff, S., Darrell, T., Saenko, K.: Semi-supervised domain adaptation via minimax entropy. arXiv preprint arXiv:1904.06487 (2019)
24. Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. J. Stat. Plann. Inference **90**(2), 227–244 (2000)
25. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in Neural Information Processing Systems, pp. 4077–4087 (2017)
26. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4068–4076 (2015)
27. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7167–7176 (2017)
28. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014)
29. Wang, M., Deng, W.: Deep visual domain adaptation: a survey. Neurocomputing **312**, 135–153 (2018)
30. Wold, S., Esbensen, K., Geladi, P.: Principal component analysis. Chemometr. Intell. Lab. Syst. **2**(1–3), 37–52 (1987)
31. Yao, T., Pan, Y., Ngo, C.W., Li, H., Mei, T.: Semi-supervised domain adaptation with subspace learning for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2142–2150 (2015)
32. Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., Saminger-Platz, S.: Central moment discrepancy (CMD) for domain-invariant representation learning. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017, Conference Track Proceedings (2017)
33. Zhao, H., des Combes, R.T., Zhang, K., Gordon, G.J.: On learning invariant representations for domain adaptation. In: Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9–15 June 2019, Long Beach, California, USA, pp. 7523–7532 (2019)
34. Zhao, H., Zhang, S., Wu, G., Moura, J.M., Costeira, J.P., Gordon, G.J.: Adversarial multiple source domain adaptation. In: Advances in Neural Information Processing Systems, pp. 8559–8570 (2018)
35. Zhuang, F., Cheng, X., Luo, P., Pan, S.J., He, Q.: Supervised representation learning: Transfer learning with deep autoencoders. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)