Federated Submodular Maximization with Differential Privacy

Anonymous Author(s)

ABSTRACT

Submodular maximization is a fundamental problem for many web applications, such as data summarization, influence maximization, and recommendation. Despite being intensively studied over the last two decades, it has not yet been considered in an emerging federated computation setting. In this paper, we comprehensively study the federated submodular maximization problem, where a set of clients aims to cooperate in finding a feasible set of items to maximize a monotone submodular function under the orchestration of a central server while providing strong privacy guarantees for their sensitive data. We consider this problem in the *client-level differential privacy* setting that does not assume a trusted server and each client should perturb its results locally before sending to the server for computation. Specifically, we propose a novel approximation algorithm for federated submodular maximization by incorporating client-level differential privacy mechanisms and decomposed function evaluations into the greedy algorithm for submodular maximization, along with two heuristics to further reduce the privacy budget and time complexity. Finally, we perform extensive experiments to demonstrate the effectiveness and efficiency of our proposed algorithms.

CCS CONCEPTS

• **Theory of computation** → **Approximation algorithms analysis**; • **Security and privacy** → **Privacy-preserving protocols**.

KEYWORDS

submodular maximization, differential privacy

ACM Reference Format:

Anonymous Author(s). 2022. Federated Submodular Maximization with Differential Privacy. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Federated computation [4] is a promising solution to large-scale, distributed data analytics with strong privacy guarantees for sensitive user data. For a set of clients that want to perform a particular analysis task collaboratively, federated computation allows each client to process data locally without sharing them with the server and other clients. Typically, only privacy-preserving summary information is sent to the server for computation. Although it is primarily adopted to train machine learning models (known as *federated learning* [24, 33]), federated computation has been applied in a wide variety of Web applications, e.g., frequency estimation [23], heavy hitters [61], and recommender systems [35, 58], due to its high scalability in dealing with massive data sets, versatility in handling different application scenarios, and flexibility in integrating with various privacy-preserving mechanisms [13].

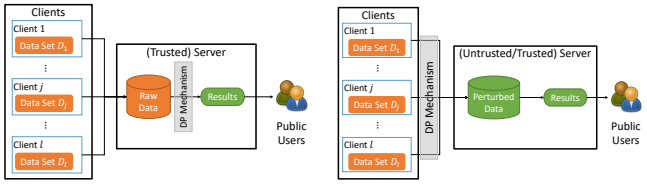
In this paper, we focus on a fundamental problem in combinatorial optimization, namely *submodular maximization* [28], in the federated setting. In particular, submodular maximization aims at finding a set S of items from a ground set V under a specific constraint to maximize the utility of S for a data set D as measured by a submodular function $f_D(S)$. It has found application to numerous problems on the Web, including data summarization [36, 45], recommendation [21, 37], and influence maximization in social networks [26, 34]. Although submodular maximization has been intensively studied over the last two decades and a wide range of efficient approximation algorithms have been designed, to our best knowledge, there has not been any prior work for this problem in the federated setting.

However, the needs for federated submodular maximization in real-world applications are compelling. For example, let us consider the problem of submodular facility location [37, 49], where each client holds sensitive user location information. The collaborative goal is to find a subset of locations from a ground set of candidate locations for facility deployment such that their overall benefits to all users, as computed by the distances from users' locations to their nearest facilities, are maximized. In this problem, user locations are stored on local clients, i.e., a participant's server or a user's device. For legal and ethical issues, sensitive location information should not be transferred directly or leaked by any result released to the public. Thus, the collection of clients would aim to find the best choice of locations based on their data without compromising privacy. In that case, it is essential to propose efficient algorithms that fit well for the computations on data sets distributed across clients and provide essential mechanisms for privacy protection. But unfortunately, most of the existing submodular maximization algorithms neither work efficiently for distributed computation nor accommodate privacy concerns.

To this end, we systematically study the federated submodular maximization problem subject to *differential privacy* [11, 13] (DP) to enable submodular maximization applications in the federated setting with provable guarantees on both privacy and utility. Differential privacy is the de facto standard for privacy-preserving computation. Generally, it guarantees that no individual's information can be inferred from the analysis procedure by ensuring that any computation result should be statistically indistinguishable with or without one individual's data. Typical DP mechanisms introduce randomness into computations to achieve such privacy guarantees. Specifically for submodular maximization problems, since they are NP-hard [15] in general, the dominant approach for designing efficient approximation algorithms is to use the greedy strategy [17, 50]. Starting from an empty solution $S = \emptyset$, a greedy algorithm iteratively adds the item that maximally increases the

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



(a) Centralized DP **(b) Client-level DP**

Figure 1: Illustration of centralized and client-level differential privacy settings for federated data analytics.

utility (called "marginal gain") of S until no more items can be added anymore with the given constraint. Theoretically, it achieves the best possible approximation factor of $1 - \frac{1}{e}$ [50] for maximizing a monotone submodular function with a cardinality constraint that limits the solution size to at most k. It can also be applied to more general constraints, e.g., matroids and p-systems, with approximation guarantee [17]. The incorporation of differential privacy into greedy algorithms has been considered in [20, 49]. A typical way is to perform a randomized selection procedure such as *exponential mechanism* [42] based on the marginal gain, instead of the greedy selection, for achieving privacy guarantee while ensuring that the utility of the solution is no much worse than that of the greedy algorithm. Nevertheless, these existing DP algorithms never consider the cases when the marginal gains should be computed across different clients. Although they can be extended to a so-called centralized DP (CDP) setting (cf. Figure 1a) by performing function evaluations on clients and applying DP mechanisms on the server only, they fail to prevent privacy leakage from a client's data set to the (possibly untrusted) server and other clients, as raw information about the data set are sent without perturbation.

Our Results. We focus on the problem of maximizing a decomposable, monotone, and submodular function subject to differential privacy. Let D be a data set of n tuples (i.e., user information) and V be a ground set of m items. We consider that each tuple $x \in D$ is associated with a monotone submodular function $f_x : 2^V \to [0, \lambda]$, where $\lambda > 0$, to measure the utility of $S \subseteq V$ for tuple x and the goal is to find a set S with certain constraint to maximize the overall utility, as computed by the sum of utilities for all tuples, i.e., $f_D(S) = \sum_{x \in D} f_x(S)$. The tuples in D are distributed across a data federation of clients, and the ground set V is kept by a central server and public to all clients. Specifically for our problem, we adopt a *client-level differential privacy* setting as shown in Figure 1b. In such a setting, each client would perturb any raw information subject to differential privacy before sending it to the server for computation.

For federated submodular maximization with client-level differential privacy, we first show that an extension of the greedy algorithm [50] called FDP-Greedy, which adds noise to marginal gains of items based on the Poisson subsampled Laplace mechanism [12] on each client and selects the item with maximum (noisy) marginal gain at each iteration on the server, yields both approximation and privacy guarantees. Specifically, FDP-Greedy achieves an approximation factor of $1 - \frac{1}{e}$ with an additive error of $\tilde{O}\big(k^{3/2} \cdot (\sqrt{n \log m} + \sqrt{lm})\big)$ while guaranteeing (ε, δ)-differential privacy on the data set of every client for any $\varepsilon, \delta > 0$, where n and m are the sizes of the data set and the ground set, k is the solution size, and l is the number of clients.[1] Moreover, as FDP-Greedy suffers from high privacy budgets and utility losses due to $O(mk)$ adaptive rounds in privacy composition, we introduce a lazy-forward [31] optimization into FDP-Greedy to decrease the number of adaptive rounds to $O(m + kc)$ for any $c \geq 1$. We further exploit using the *Permute-and-Flip* mechanism [40] along with the Laplace mechanism to reduce the adaptive complexity to $O(kc)$. Although strict approximation bounds for utilities cannot be guaranteed without additional assumptions, both heuristics lower the privacy budgets, thus leading to solutions of higher quality in practice.

Finally, we conduct extensive experiments for two submodular maximization applications, i.e., *maximum coverage* and *facility location*, on real-world data sets to evaluate the performance of our proposed algorithms. The results show that our proposed algorithms provide high-utility solutions close to those of non-private algorithms for federated submodular maximization in a typical setting for (ε, δ)-DP (i.e., $\varepsilon = 2$, $\delta = 1/n^{1.5}$) while running up to two orders of magnitude faster. Our main contributions are:

- We formally define the federated submodular maximization problem subject to differential privacy. (Section 3)
- We design a novel approximation algorithm with two efficient heuristics for federated submodular maximization together with theoretical analyses. (Section 4)
- We conduct extensive experiments to demonstrate the effectiveness and efficiency of our proposed algorithms. (Section 5)

2 RELATED WORK

Submodular Maximization. There has been a vast amount of literature on submodular maximization (see [28] for a survey). The greedy algorithm proposed by Nemhauser et al. [50] is the most widely used approach to maximizing monotone submodular functions. It achieves a best possible $(1 - \frac{1}{e})$-approximation for a cardinality constraint [50], a $\frac{1}{2}$-approximation for a matroid constraint, and a $\frac{1}{p+1}$-approximation for a p-system constraint [17]. *Lazy-forward* [31] and *subsampling* [1, 46] have been considered to improve the efficiency of the greedy algorithm while retaining the approximation factors in non-private settings.

Since the greedy algorithm works only for centralized computation, many efforts have been devoted to distributed submodular maximization [10, 14, 30, 44, 47] where data are distributed across networks. Although the federated computation we consider seems similar to the distributed computation, they are essentially different, and the methods designed for one cannot be trivially used for the other. In the distributed computation, either the function values are directly provided by an oracle or the data set D for function evaluation is accessible by all the participants. But in the federated computation, each client holds a subset of D that cannot be accessed from outside; the server has a public ground set V and maximizes the function on V by issuing queries to clients. Thus, unlike distributed computations, we can deploy DP mechanisms at the client level in federated analytics.

Differentially Private Submodular Optimization. There have been several differentially private algorithms proposed for submodular optimization problems [7, 8, 20, 49, 51–53, 56]. The seminal

[1] Here, the notation $\tilde{O}$ denotes that all parameters related to ε and δ are ignored.

work of Gupta et al. [20] first proposed to extend the greedy algorithm for maximizing decomposable, monotone, and submodular functions with cardinality constraints subject to differential privacy using the exponential mechanism [42]. Mitrovic et al. [49] generalized the results in [20] for general submodular functions with different constraints beyond cardinality. Then, several DP algorithms [7, 51] were proposed for submodular minimization/maximization in an online setting. In addition, continuous greedy algorithms [6, 16, 29] were also used to design differentially private algorithms for submodular maximization problems with matroid and knapsack constraints [8, 52, 53, 56]. Although continuous greedy algorithms achieve better approximation factors and stronger privacy guarantees for more complex constraints (e.g., matroid and knapsack), they suffer from low efficiency and cannot scale to large data. Thus, our experiments only compare our algorithms with those in [20, 49].

Federated Analytics with Differential Privacy. Federated computation [4] is an emerging area that seeks to perform large-scale, distributed data analytics while providing strong privacy protection for sensitive user data. Its most prominent application is to train machine learning models, which is referred to as *federated learning* [24, 33]. Specifically, federated learning approaches subject to differential privacy have attracted much attention recently [18, 38, 39, 55, 57, 59]. Besides training ML models, many other differentially private data analytics problems have also been studied in the federated setting, e.g., frequency estimation [23], heavy hitters [61], histograms [2, 9], recommender systems [35, 58], data summarization [54], and principle component analysis [19]. Although the above approaches are promising, they are designed for different problems and are not directly comparable to our work.

3 PRELIMINARIES

This section introduces the preliminary concepts about submodular maximization, federated computation, and differential privacy, which will be used for our algorithms in Section 4.

Submodular Maximization. Let V be a finite set referred to as the *ground set* and X be a finite set referred to as the *data universe*. A data set is a n-tuple set $D = \{x_1, \cdots, x_n\} \in X^n$. Suppose that D is associated with a set function $f_D : 2^V \to \mathbb{R}$ to measure the utility of any subset $S \subseteq V$ for the data set D. The definition of f_D depends on the specific application, but it is assumed that the association between D and f_D is public information. In this paper, we consider that the set function f_D is *normalized*, i.e., $f_D(\emptyset) = 0$, *monotone*, i.e., $f_D(S) \leq f_D(T)$ for any $S \subseteq T \subseteq V$, and *submodular*, i.e., $f_D(S \cup \{v\}) - f_D(S) \geq f_D(T \cup \{v\}) - f_D(T)$ for any $S \subseteq T \subseteq V$ and $v \in V \setminus T$. If for any data set $D = \{x_1, \cdots, x_n\}$, we have $f_D(S) = \sum_{i=1}^{n} f_{x_i}(S)$, where a normalized, monotone and submodular function $f_{x_i} : 2^V \to [0, \lambda]$ measures the utility of any $S \subseteq V$ for each $x_i \in D$, we say f_D is λ-*decomposable*. We investigate the problem of approximately maximizing a λ-decomposable utility function f_D over a set $\mathcal{I} \subseteq 2^V$ of feasible sets, i.e., $S^* = \arg\max_{S \in \mathcal{I}} f_D(S)$. We focus on the simplest case when $\mathcal{I}$ is defined by a cardinality constraint k, i.e., $\mathcal{I} = \{S \subseteq V : |S| \leq k\}$. Note that all our results are still valid when $\mathcal{I}$ is defined by more general matroid [27] and p-system [43] constraints, coming at the expense of lower approximation ratios.

Federated Submodular Maximization. We study the privacy-preserving submodular maximization problem in the federated setting. We have a data federation consisting of l clients, in which each client j holds a private n_j-tuple subset $D_j = \{x_{j1}, \cdots, x_{jn_j}\}$ of data set D. For simplicity, we assume that the data sets of different clients are disjoint from each other, i.e., $D_j \cap D_{j'} = \emptyset$ for any $j \neq j'$. Based on a public ground set V, the central server and clients want to select a feasible subset $S \in \mathcal{I}$ from V collaboratively to maximize the utility for D subject to *differential privacy*. In this work, we consider a *client-level differential privacy* setting, a generalization of the de facto standard for privacy-preserving computation, i.e., *differential privacy* [11–13] (DP), for federated data analytics. In the client-level DP setting, every client must protect its data set with differential privacy by performing data perturbation before sending any result to the server. The server computes a solution on perturbed data and releases the differentially private results to the public (cf. Figure 1b).

Differential Privacy. Next, we introduce several fundamental concepts we will use to build our algorithms for federated submodular maximization in the client-level differential privacy setting. At a high level, a differentially private algorithm should ensure that any two neighboring data sets $D, D' \in X^n$ (denoted as $D \sim D'$) that differ in at most one tuple do not produce output that is very different statistically. Formally,

Definition 3.1 (Differential Privacy [12]). A randomized mechanism $\mathcal{M} : X^n \to \mathcal{R}$ satisfies (ε, δ)-*differential privacy* if for all measurable sets $R \subseteq \mathcal{R}$ and all neighboring data sets $D \sim D'$,

$$\Pr[\mathcal{M}(D) \in R] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in R] + \delta.$$

An important property of the function of interest for differentially private algorithms is that the function value does not change significantly with respect to small changes in the input data set, which is formalized by *sensitivity* as follows:

Definition 3.2 (Sensitivity [20, 49]). The *sensitivity* of a set function $f_D : 2^V \to \mathbb{R}$ with respect to a constraint $\mathcal{I} \in 2^V$ is

$$\max_{D \sim D'} \max_{S \in \mathcal{I}} |f_D(S) - f_{D'}(S)|.$$

It is already known that the sensitivity of any λ-decomposable function f_D is at most λ [8, 20]. For ease of analysis, we assume that λ is normalized to 1 by dividing all the function values by λ in the rest of this paper. For federated submodular maximization, the server issues a query $q_{D_j}(v, S)$ to client j for the *marginal gain* $f_{D_j}(S \cup \{v\}) - f_{D_j}(S)$ of v w.r.t. S on D_j; the client j computes and provides the perturbed $\tilde{q}_{D_j}(v, S)$ to the server. After the server collects the answers for a limited number of queries from clients, it should produce an approximate and (ε, δ)-*differentially private* solution $\tilde{S} \in \mathcal{I}$ to maximize f_D for release.

One basic result for designing DP algorithms to solve such optimization problems is *adaptive composition* [5, 13]. In particular, when a $(\varepsilon_0, \delta_0)$-DP algorithm is repeated used on the same input D in k rounds, where the i-th round of computation depends on the output of previous $i' < i$ rounds, the k-fold adaptive composition satisfies (ε, δ)-DP, where

$$\varepsilon = k\varepsilon_0,\ \delta = k\delta_0 \text{ (basic composition)}; \quad (1)$$

$$\varepsilon = k\varepsilon_0^2/2 + \sqrt{2k\ln(1/\delta')}\varepsilon_0,\ \delta = \delta' + k\delta_0 \text{ (adv. composition)} \quad (2)$$

for any $\delta' > 0$. Intuitively, the *adaptive composition* quantifies the accumulated privacy loss when a DP algorithm is repeated and adaptively used. Another two important results for differential privacy are "*post-processing immunity*" [13], i.e., if $\mathcal{M}$ is (ε, δ)-DP on D, then any algorithm without additional knowledge about D except the output of $\mathcal{M}$ will be (ε, δ)-DP on D and "*parallel composition*" [41], i.e., if $\mathcal{M}$ is (ε, δ)-DP on each disjoint subset D_j of D, then $\mathcal{M}$ will also provide (ε, δ)-DP on D.

Next, we introduce two basic DP mechanisms that serve as building blocks of our algorithms, namely *Laplace mechanism* [12] and *Permute-and-Flip mechanism* [40]. The Laplace mechanism is a basic DP mechanism for answering numeric queries. Its basic idea is to perturb the original query result by adding noise drawn from the Laplace distribution. Here, we only consider the one-dimensional case. Let a query $q : X^n \to \mathbb{R}$ of sensitivity Δq map a data set $D \in X^n$ to a real number. The Laplace mechanism is defined as $\mathcal{M}_L(q, D, \varepsilon) = q_D + \mathcal{L}(0, \frac{\Delta q}{\varepsilon})$, where q_D is the original query result of q on D and $\mathcal{L}(0, \frac{\Delta q}{\varepsilon})$ is a random variable drawn from a Laplace distribution with 0 mean and $\frac{\Delta q}{\varepsilon}$ scale. According to [12], the Laplace mechanism preserves $(\varepsilon, 0)$-differential privacy. The Permute-and-Flip mechanism is designed for differentially privately selecting a high-utility item from the ground set. The basic idea is to pick each item in the ground set sequentially and randomly based on utility. Formally, given a function $q : V \times X^n \to \mathbb{R}$ of sensitivity Δq which measures the utility $q_D(v)$ of each $v \in V$ on data set $D \in X^n$, the Permute-and-Flip mechanism $\mathcal{M}_{PF}(q, D, \varepsilon)$ proceeds as follows: It first finds the maximum utility $q^* = \max_{v \in V} q_D(v)$ among all items. Then, it generates a random permutation $\pi(V)$ of V. Each $v \in \pi(V)$ is checked one by one: it returns the current v as the output and terminates the selection with probability $\exp\left(\frac{\varepsilon \cdot (q_D(v) - q^*)}{2\Delta q}\right)$ or moves on to the next item otherwise. According to [40], the Permute-and-Flip mechanism preserves $(\varepsilon, 0)$-differential privacy. Moreover, it guarantees that

$$\mathbb{E}[q_D(\tilde{v})] \geq \max_{v \in V} q_D(v) - \frac{2\Delta q}{\varepsilon} \cdot (\ln m + t) \tag{3}$$

with probability at least $1 - e^{-t}$ for the output $\tilde{v}$ of the Permute-and-Flip mechanism $\mathcal{M}_{PF}(q, D, \varepsilon)$.

4 OUR ALGORITHMS

In this section, we present our algorithms for the problem of federated submodular maximization. We first propose the FDP-Greedy algorithm based on the greedy algorithm [50] and the Laplace mechanism [12]. We then incorporate two heuristics into the FDP-Greedy algorithm to reduce the privacy cost and time complexity.

4.1 The FDP-Greedy Algorithm

In this subsection, we present our FDP-Greedy algorithm for maximizing decomposable, monotone, and submodular functions in the federated setting. Specifically, FDP-Greedy is an extension of the original greedy algorithm [50] in the following three aspects. First, as a basic requirement for federated data analytics, it computes the marginal gain of each item $v \in V$ on data set D_j at client j and aggregates the marginal gains from clients on the server to avoid directly transferring the original data set. Second, it requires each client to perturb the marginal gains subject to differential privacy before sending them to the server. We use the Laplace mechanism [12] to guarantee (ε, δ)-DP for real-valued queries in our algorithm. Third, a subsampling strategy is introduced to further improve the performance of our algorithm. The benefits of subsampling are two-fold: *(i)* The privacy guarantee of a differentially private mechanism can be amplified by subsampling [3, 25, 32, 62]. *(ii)* In decomposable submodular maximization, subsampling [1] is an effective approach to improve the time efficiency while retaining the approximation factor. In particular, we use Poisson sampling in our algorithm.

Algorithm 1: FDP-Greedy

Input: Data set $D = \bigcup_{j=1}^{l} D_j$ across l clients, ground set V, utility function $f_D : 2^V \to \mathbb{R}$, feasible sets $\mathcal{I} \subseteq 2^V$, sampling rate $\gamma \in (0, 1]$, privacy parameter $\varepsilon_0 > 0$

Output: Set $\tilde{S}$ such that $\tilde{S} \in \mathcal{I}$

Initialize $\tilde{S} \leftarrow \emptyset$;
while *there exists any item* $v \in V \setminus \tilde{S}$ *such that* $\tilde{S} \cup \{v\} \in \mathcal{I}$ **do**
  foreach *client* $j = 1, \ldots, l$ **do**
    foreach $v \in V \setminus \tilde{S}$ *such that* $\tilde{S} \cup \{v\} \in \mathcal{I}$ **do**
      $\widehat{D_j} \leftarrow \mathsf{PossionSample}(\gamma, D_j)$;
      $q_{\widehat{D_j}}(v, \tilde{S}) \leftarrow f_{\widehat{D_j}}(\tilde{S} \cup \{v\}) - f_{\widehat{D_j}}(\tilde{S})$;
      $\tilde{q}_{\widehat{D_j}}(v, \tilde{S}) \leftarrow \mathcal{M}_L(q_{\widehat{D_j}}(v, \tilde{S}), \widehat{D_j}, \varepsilon_0)$;
  $\tilde{q}_{\widehat{D}}(v, \tilde{S}) \leftarrow \sum_{j=1}^{l} \tilde{q}_{\widehat{D_j}}(v, \tilde{S})$ for each item $v \in V \setminus \tilde{S}$;
  $\tilde{v}^* \leftarrow \arg\max_{v \in V \setminus \tilde{S} : \tilde{S} \cup \{v\} \in \mathcal{I}} \tilde{q}_{\widehat{D}}(v, \tilde{S})$;
  Update $\tilde{S} \leftarrow \tilde{S} \cup \{\tilde{v}^*\}$;
return $\tilde{S}$

Before presenting our algorithm, we first give a formal definition of Poisson sampling and its privacy amplification bound [3, 32].

Definition 4.1 (Possion Sampling). Given a data set $D \in X^n$, the Possion sampling (PossionSample) procedure outputs a subset $\widehat{D} = \{x_i : Y_i = 1, i \in [n]\}$ of D, where each Y_i is an i.i.d. variable drawn from a Bernoulli distribution $\mathrm{Ber}(\gamma)$ for any parameter $\gamma \in (0, 1]$.

Lemma 4.2 ([3, 32]). *If $\mathcal{M}$ is (ε, δ)-DP, then $\mathcal{M}' = \mathcal{M}(\widehat{D})$, i.e., applying $\mathcal{M}$ on $\widehat{D}$ output by* PossionSample *with parameter γ, satisfies (ε', δ')-DP, where $\varepsilon' = \ln(1 + \gamma(e^{\varepsilon} - 1))$ and $\delta' = \gamma\delta$.*

Algorithm Description. Our FDP-Greedy algorithm is described in Algorithm 1. Generally, it starts from $\tilde{S} = \emptyset$ and iteratively adds an item $\tilde{v}^*$ with the largest (noisy) marginal gain with respect to $\tilde{S}$ until no more items can be added. At each iteration, the server issues a query for the marginal gain of each item $v \in V \setminus \tilde{S}$ w.r.t. the current $\tilde{S}$ to each client. After receiving a query, each client j first performs a Poisson sampling procedure with sampling rate γ on its data set D_j and then computes the marginal gain based on the sampled data set $\widehat{D_j}$. Before sending the marginal gains (i.e., query results) back to the server, the client perturbs them via the Laplace mechanism. The server sums up the (noisy) marginal gains of all items from every client and adds the item with the largest marginal gain to $\tilde{S}$. Finally, if $\tilde{S}$ is a maximal feasible set of $\mathcal{I}$, i.e., there does not exist any $v \in V \setminus \tilde{S}$ such that $\tilde{S} \cup \{v\} \in \mathcal{I}$, $\tilde{S}$ will be returned as the (differentially private) solution to federated submodular maximization.

Theoretical Analysis. Next, we will analyze the privacy, approximation, and complexity of Algorithm 1.

First of all, according to Lemma 4.2, the Laplace mechanism together with the Poisson sampling procedure preserves $(\varepsilon_0', 0)$-differential privacy, where $\varepsilon_0' = \ln(1+\gamma(e^{\varepsilon_0}-1))$. Since the number of adaptive rounds (i.e., queries) on each client is at most mk, Algorithm 1 guarantees $(mk\varepsilon_0', 0)$-differential privacy or $(mk\varepsilon_0'^2/2 + \sqrt{2mk\ln(1/\delta)}\cdot\varepsilon_0', \delta)$-differential privacy on data set D_j for each $j \in [l]$ (as well as D due to *parallel composition*) based on the basic or advanced composition in Equations (1) and (2), respectively.

Then, we analyze the upper bound of the difference between the original and perturbed marginal gains $q_D(v,\tilde{S})$, $\tilde{q}_{\widehat{D}}(v,\tilde{S})$ for any item v and set $\tilde{S}$ in the following lemma.

Lemma 4.3. *For any item $v \in V$, any subset $\tilde{S} \subset V$ of size at most k, and a given data set D, it holds with a high probability that*

$$\left| q_D(v,\tilde{S}) - \frac{\tilde{q}_{\widehat{D}}(v,\tilde{S})}{\gamma} \right| \leq \frac{1}{\gamma}\cdot\left(\sqrt{nk\ln m} + \frac{3\sqrt{2l}}{\varepsilon_0}\right). \tag{4}$$

The proof of Lemma 4.3 is included in Appendix C.1.

Finally, we derive the privacy guarantee, approximation factor, and time complexity of Algorithm 1 in the following theorem based on all the above results.

Theorem 4.4. *Suppose $f_D : 2^V \to \mathbb{R}$ is a decomposable, monotone, and submodular function. For any parameter $\varepsilon_0, \gamma > 0$, Algorithm 1 provides (ε,δ)-differential privacy, where $\varepsilon = mk\varepsilon_0'$ and $\delta = 0$ (basic composition), or $\varepsilon = mk\varepsilon_0'^2/2 + \sqrt{2mk\ln(1/\delta)}\cdot\varepsilon_0'$ and $\delta > 0$ (advanced composition), and $\varepsilon_0' = \ln(1+\gamma(e^{\varepsilon_0}-1))$. Moreover, for every $D \in X^n$, it holds with a high probability that*

$$\mathbb{E}\left[f_D(\tilde{S})\right] \geq \left(1-\frac{1}{e}\right)\cdot \mathtt{OPT} - O\left(\frac{k^{3/2}}{\gamma}\cdot\left(\sqrt{n\log m} + \frac{\sqrt{lm}}{\varepsilon}\right)\right),$$

where $\mathtt{OPT} = \max_{S\in\mathcal{I}} f(S)$, $\mathcal{I}$ is defined by a cardinality constraint $k \in \mathbb{Z}^+$, i.e., $\mathcal{I} = \{S \in 2^V : |S| \leq k\}$, and $m = |V|$. Finally, the time complexity of Algorithm 1 is $O(nmk\gamma)$.

The proof of Theorem 4.4 is included in Appendix C.2.

4.2 Lazy-Forward Optimization

The main drawback of FDP-Greedy is that it requires $\Omega(mk)$ queries on each client for solution computation. This leads to significant privacy budgets due to adaptive composition, which incurs large utility losses in the higher private regime (i.e., $\varepsilon \leq 2$). In this subsection, we incorporate the lazy-forward strategy [31], a widely used approach to improving the performance of greedy submodular maximization, into FDP-Greedy to reduce the number of queries.

We apply the lazy-forward optimization technique, inspired by the prior work [31] that uses the same approach in the original greedy algorithm [50]. The basic idea is to exploit submodularity to reduce the number of queries for the marginal gains of items. Assume that we have evaluated the marginal gain $\Delta_{f_D}(v,\tilde{S})$ of any item $v \in V$ w.r.t. $\tilde{S}$. A key observation is that $\Delta_{f_D}(v,\tilde{S})$ will never increase, as $\tilde{S}$ only grows over time. The above result still holds with high probability when the marginal gains are computed on subsampled data sets and perturbed with Laplace noise because the sampling rate and privacy parameter are always the same for all items at every iteration. Thus, instead of recomputing the marginal gains for all items at each iteration, we can utilize the previously evaluated marginal gain $\delta(v)$ of each $v \in V$ to prune unnecessary queries. We keep all items $v \in V$ in descending order of $\delta(v)$ (e.g., by using a priority queue). We set all $\delta(v)$'s as *outdated* at the start of each iteration, as they are computed based on an old $\tilde{S}$. Then, we always extracts the item v' with the largest $\delta(v')$ at one time. If $\delta(v')$ is outdated, we re-evaluate $\delta(v')$ w.r.t. the up-to-date $\tilde{S}$ by issuing queries to all clients, insert v' back to the queue, and extract the top item from the queue again; otherwise, we will skip the remaining items and add v' to $\tilde{S}$. Furthermore, we limit the maximum number of items to re-evaluate per iteration by a threshold $c \geq 1$. Once the first c $\delta(v')$'s extracted from the queue are all outdated, we will add the item with the largest marginal gain among the c re-evaluated items to $\tilde{S}$. Algorithm 2 (FDP-LF-Greedy) in Appendix B provides pseudocode for the above procedure.

Theoretically, the number of queries on each client in FDP-LF-Greedy decreases from $O(mk)$ to $O(m+kc)$ because it queries each client m times at the first iteration but up to c times at the remaining $k-1$ iterations. Thus, the privacy cost and time complexity of FDP-LF-Greedy improve upon those of FDP-Greedy by replacing a factor of mk with $m+kc$ accordingly. Moreover, FDP-LF-Greedy can no longer provide strict theoretical bounds on utilities because it cannot guarantee to find the item with the maximum marginal gain w.r.t. $\tilde{S}$ after c re-evaluations. Nevertheless, when the value of c is large enough such that an up-to-date $\delta(v')$ can be found across all iterations, the approximation factor of FDP-LF-Greedy will keep the same as FDP-Greedy.

4.3 Permute-and-Flip Based Heuristic

In this subsection, we consider how to reduce the number of queries to be sublinear with m for lower privacy budgets. As depicted in Section 4.2, by applying the lazy-forward optimization, unnecessary queries on low-utility items that are less likely to be included in the solution are pruned. However, the lazy-forward strategy works only when the marginal gains of all items have been computed by each client at least once so that the server can decide which items to query at subsequent iterations. Thus, its query complexity is at least $\Omega(m)$. For further improvement, each client should report the marginal gains of only a few high-utility items to the server. As shown in Section 3, the *exponential mechanism* [42] and its variant, i.e., the *permute-and-flip mechanism* [40], can find an item based on a utility function with both approximation and privacy guarantees. Next, we will present the procedure of integrating the permute-and-flip mechanism into FDP-Greedy for federated submodular maximization.

Generally, we follow the same procedure as the original FDP-Greedy algorithm, i.e., starting with an empty set $\tilde{S}$ and running iteratively to add the items with the maximum marginal gains to $\tilde{S}$. The difference from the original FDP-Greedy lies in the method to evaluate the marginal gains of items. At the start of each iteration, it initializes the marginal gain of each item to 0. Then, to apply the permute-and-flip mechanism, instead of querying the clients for the marginal gain of a specific item, the server asks each client to provide the item with the maximum marginal gain w.r.t. the set $\tilde{S}$ as well as the value of its marginal gain. Each client computes the marginal gains of all items locally, performs the permute-and-flip mechanism to select an item, and sends the selected item along with

its marginal gain perturbed with the Laplace noise to the server. Then, the server updates the marginal gains of the items returned by all clients. Similar to the lazy-forward optimization, we also limit the number of queries to each client per iteration by a threshold $c \geq 1$. As such, the server adds an item with the largest marginal gain, obtained from the results of c queries for the top-c items of all clients, to $\tilde{S}$. Algorithm 3 (FDP-PF-Greedy) in Appendix B provides pseudocode for the above procedure.

Intuitively, the number of queries on each client in FDP-PF-Greedy is $O(kc)$. To answer each query, a client uses the *permute-and-flip mechanism* [40] and *Laplace mechanism* [12] sequentially to find an item and its marginal gain. Let ε_1 and ε_2 be the privacy parameters for the two mechanisms. We apply the basic composition in Equation (1) along with the privacy amplification bound in Lemma 4.2 to guarantee that each query in FDP-PF-Greedy is $(\varepsilon_0', 0)$-differentially private, where $\varepsilon_0' = \varepsilon_1' + \varepsilon_2'$, $\varepsilon_1' = \ln(1 + \gamma(e^{\varepsilon_1} - 1))$, and $\varepsilon_2' = \ln(1 + \gamma(e^{\varepsilon_2} - 1))$. Then, FDP-PF-Greedy ensures (ε, δ)-differential privacy when the value of ε_0' is computed based on the advanced composition in Equation (2) for kc adaptive rounds. Moreover, each client j can provide an item whose marginal gain w.r.t. $\tilde{S}$ over D_j is close to the maximum marginal gain within bounded errors by expectation according to the following lemma.

Lemma 4.5. *For a subset $\tilde{S} \subset V$ of size at most k and a data set D_j at client j, it holds with a high probability that*

$$\mathbb{E}\left[\frac{q_j(\tilde{v}_{ij})}{\gamma}\right] \geq \max_{v \in V_j} q_{D_j}(v, \tilde{S}) - \frac{1}{\gamma} \cdot \left(2\sqrt{nk \ln m} + \frac{4 \ln m}{\varepsilon_1} + \frac{\ln m}{\varepsilon_2}\right),$$

where $\tilde{v}_{ij}$ is the item returned by client j for any $i \in [c]$.

The proof of Lemma 4.5 is included in Appendix C.3.

Nevertheless, despite the result of Lemma 4.5, FDP-PF-Greedy cannot provide any approximation guarantee for utility in general because the item with the largest marginal gain w.r.t. $\tilde{S}$ may not be among the top-c items on each client. Finally, the time complexity of FDP-PF-Greedy is $O(cnmk\gamma)$ because it issues $O(kc)$ queries to each client, and the number of function evaluations to answer each query across all clients is $O(nm\gamma)$.

5 EXPERIMENTS

In this section, we compare the performance of our proposed algorithms with several baselines in terms of the effectiveness and efficiency on real-world submodular maximization problems, namely *maximum coverage* and *facility location*.

5.1 Experimental Setup

Algorithms. The following five algorithms for submodular maximization are compared in our experiments.

- **Greedy:** A non-private $(1 - 1/e)$-approximation algorithm for submodular maximization under cardinality constraints [50].
- **CDP-Greedy:** An algorithm for submodular maximization in the centralized DP setting from [20, 49]. We use the *Permute-and-Flip* mechanism [40] (PFM) instead of the *exponential* mechanism [42] (EM) for item selection because PFM achieves better performance than EM in practice. For a fair comparison, we also apply the Poisson sampling to CDP-Greedy.
- **FDP-Greedy:** Our basic greedy algorithm for submodular maximization in the client-level DP setting (cf. Algorithm 1).
- **FDP-LF-Greedy:** An improved version of FDP-Greedy with the Lazy-Forward [31] optimization (cf. Algorithm 2).
- **FDP-PF-Greedy:** An extension of FDP-Greedy by incorporating the the *Permute-and-Flip* mechanism [40] (cf. Algorithm 3).

Note that continuous greedy algorithms [8, 52, 53, 56] for differential private submodular maximization are not compared since their efficiencies are too low for practical applications.

Applications and Data Sets. We consider the following two real-world applications of federated submodular maximization.

Maximum coverage is a well-known NP-Hard submodular maximization problem with wide applications in document summarization [36] and influence maximization [26, 34]. Given a data set D of n tuples and a ground set V of m sets, where each $V' \in V$ is a subset of D, i.e., $V' \subseteq D$, the goal is to select a subset $S \in \mathcal{I}$ of V such that the union of the sets in S contains as many tuples in D as possible. The utility function is defined as $f_D(S) = |\bigcup_{V' \in S} V'|$. Here, we consider that $\mathcal{I}$ is defined by a cardinality constraint k. Furthermore, We should protect the *membership privacy* of tuples (i.e., individuals). Specifically, each tuple $x \in D$ as well as its membership set $S(x) = \{V' \in V : x \in V'\}$ are stored on one of the l clients. The index of each $V' \in V$ is public, but the members in V' are private. We must compute an approximate solution to the maximum coverage problem without leaking membership information.

We use two public data sets for maximum coverage. The *DBLP* data set is downloaded from https://dblp.org/xml/release/. It is a bipartite network between $n = 704,738$ researchers and $m = 2,675$ venues. Each researcher's membership set contains all venues she/he published at least one paper between 2018 and 2021. The goal is to select a set of venues that covers the most number of researchers under differential privacy. The *Flickr* data set is obtain from [48]. It is also a bipartite network constructed from group memberships between $n = 242,364$ users and $m = 2,000$ groups. The goal is to pick a set of groups that contains the most number of users under differential privacy.

Facility location is another well-known NP-Hard submodular maximization problem which have been widely used for data summarization [37] and sensor deployment [49]. Given a data set D of n tuples, each denotes an individual's location, and a ground set V of m items, each denotes a candidate location, we define a (nonnegative) benefit matrix $B \subseteq \mathbb{R}^{n \times m}$, where each entry $b_{xv} \in B$ denotes the benefit of location $v \in V$ on individual $x \in D$. The goal is to select a subset $S \in \mathcal{I}$ from candidate locations to maximize the overall benefits of all individuals. The utility function is defined as $f_D(S) = \sum_{x \in D} \max_{v \in S} b_{xv}$. Here, $\mathcal{I}$ is also defined by a cardinality constraint k. In the facility location problem, we should protect the location privacy of individuals. Specifically, the tuples in D are private and kept across clients, but the locations of candidate facilities are public.

We use two public data sets for facility location. The *Uber* data set is downloaded from https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city, which consists of a set of 564,516 pickups in New York City. We pick a set of 1,000 popular locations as facilities. The *FourSquare* data set [60] is a set of 100,000 check-ins in Tokyo. We choose a set of $m = 1,131$ medical centers

Table 1: Overview of the performance of different algorithms on each data set in the default setting (i.e., $\varepsilon = 2, \delta = 1/n^{1.5}, k = 10, l = 20$). The best results among all FDP algorithms are highlighted in bold fonts.

Algorithm	DBLP			Flickr			Uber			FourSquare		
	Utility ($\times 10^5$)	Time (s)	Comm. (MB)	Utility ($\times 10^4$)	Time (s)	Comm. (MB)	Utility ($\times 10^6$)	Time (s)	Comm. (MB)	Utility ($\times 10^4$)	Time (s)	Comm. (MB)
(Non-Private) Greedy	1.106	7502.3	5.39	9.763	2010.8	3.64	4.6331	9945.9	5.19	5.984	1524.5	6.40
CDP-Greedy	1.102 (±0.004)	75.70	5.39	9.721 (±0.038)	20.75	3.64	4.6139 (±0.0018)	99.17	5.19	5.877 (± 0.045)	15.14	6.40
FDP-Greedy	0.883 (±0.096)	79.95	14.07	7.746 (±0.433)	22.19	10.13	4.5482 (±0.0038)	**99.21**	5.54	5.376 (±0.135)	**16.02**	6.51
FDP-LF-Greedy	1.083 (±0.015)	**39.01**	2.14	9.127 (±0.191)	**15.73**	1.29	4.5790 (±0.0022)	167.8	1.57	5.555 (±0.108)	23.10	1.44
FDP-PF-Greedy	**1.093 (±0.010)**	146.6	**1.02**	**9.386 (±0.159)**	39.05	**0.42**	**4.5920 (±0.0026)**	183.8	**0.83**	**5.673 (±0.126)**	28.17	**0.70**

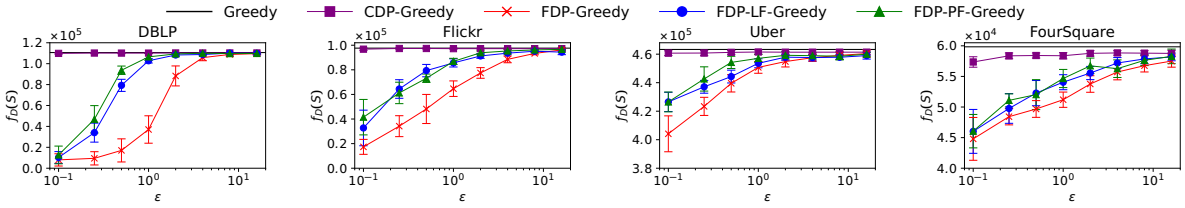


Figure 2: Solution quality of different algorithms with varying the privacy parameter ε from 0.1 to 16.

locations as facilities. We adopt the *RBF kernel function* following [37] to calculate the benefit of facility v on individual x, i.e., $b_{xv} = \exp(-\gamma \|\mathbf{p}_x - \mathbf{p}_v\|_2^2)$, where $\mathbf{p}_x$ and $\mathbf{p}_v$ are the coordinates of v and x, respectively, and the parameter γ is calculated based on the inverse of the average squared distance between facility and user locations. We aim to select a set of facilities to serve all individuals best while keeping their locations private.

Evaluation Metrics and Default Settings. Each algorithm we compare is run ten times with different seeds. The effectiveness is measured by the average and variance of the utility $f_D(S)$ of the solution S. The efficiency is measured by the average running time and the amount of data transferred between the server and clients. By default, we set the privacy parameters $\varepsilon = 2$ and $\delta = n^{-1.5}$ following the typical choice for DP deployment in industry, e.g., Apple[2]. In the experiments on varying ε, we test $\varepsilon \in \{0.1, 0.25, 0.5, \ldots, 16\}$. The solution size k is set to 10 by default and is varied over $\{1, 2, 4, \ldots, 20\}$ to evaluate its impact on each algorithm. We use $l = 20$ clients by default. We also vary the number l of clients over $\{1, 4, \ldots, 1024\}$ for scalability tests. We set the sampling rate $\gamma = 0.01$ in CDP-Greedy and our algorithms. In FDP-LF-Greedy, we set the cut-off threshold c to 16 and 256 for *maximum coverage* and *facility location*, respectively. In FDP-PF-Greedy, the cut-off threshold c is fixed to 2 and the ratio $\varepsilon_1/\varepsilon_2$ is set to 4. Also, the values of γ, c, and $\varepsilon_1/\varepsilon_2$ are varied over the ranges $[0.0001, \ldots, 1]$, $[1, \ldots, 1024]$, and $[1/16, \ldots, 16]$, respectively, for parameter tuning. Due to space limitations, all the results for tuning the parameters γ, c, and $\varepsilon_1/\varepsilon_2$ in our algorithms are deferred to the full version of this paper in https://anonymous.4open.science/r/5F42.

Hardware Configuration and Implementation. We conducted all experiments on a server with an Intel Xeon E5-2650v4 2.2GHz processor and 64GB memory running Ubuntu 18.04 LTS. All algorithms were implemented in Python 3. Our code and data are released on https://anonymous.4open.science/r/5F42.

[2]https://machinelearning.apple.com/research/learning-with-privacy-at-scale

5.2 Experimental Results

Overview. We present the performance of different algorithms in the default setting (i.e., $\varepsilon = 2, \delta = 1/n^{1.5}, k = 10, l = 20$) in Table 1. Among our federated algorithms, FDP-PF-Greedy always provides the solutions with the highest utilities because its number of rounds for privacy composition is the lowest. In particular, the utilities of its solutions reach at least 96% of those of non-private greedy solutions on all data sets. Furthermore, FDP-LF-Greedy also shows better solution quality than FDP-Greedy. Moreover, in terms of efficiency, all the algorithms that apply the Poisson subsampling with parameter $\gamma = 0.01$ are nearly two orders of magnitude faster than the non-private greedy algorithm. Because of fewer function evaluations, the lazy-forward optimization improves the time efficiency upon FDP-Greedy for maximum coverage (when $c = 16$). However, it reduces the time efficiency for facility location (when $c = 256$) due to extra queue maintenance and communication costs. Specifically, the server communicates with each client c times per iteration in FDP-LF-Greedy but only 1 time per iteration in FDP-Greedy. The extra communication overhead can overwhelm the benefit of reducing function evaluations when c is large. The running time of FDP-PF-Greedy is near c times longer than that of FDP-Greedy, as indicated by its time complexity. In terms of communication cost, the amount of transferred data is proportional to the number of queries for all algorithms. Naturally, FDP-PF-Greedy has the lowest communication cost in all cases.

Impact of Privacy Parameter ε. The solution quality of different algorithms with varying ε is illustrated in Figure 2. The smaller the value of ε is, the stronger privacy guarantee an algorithm provides, yet the more noise it adds to results. All algorithms in the client-level DP setting cannot provide high-quality solutions when $\varepsilon < 1$ because of more strict privacy requirements than the centralized DP setting. FDP-LF-Greedy and FDP-PF-Greedy consistently achieve better solution quality than FDP-Greedy across different ε's due to smaller privacy losses. Moreover, their utilities are close to those of non-private greedy solutions and are highly stable when $\varepsilon \geq 2$. These results confirm the effectiveness of our algorithms in common scenarios for privacy-preserving computation.

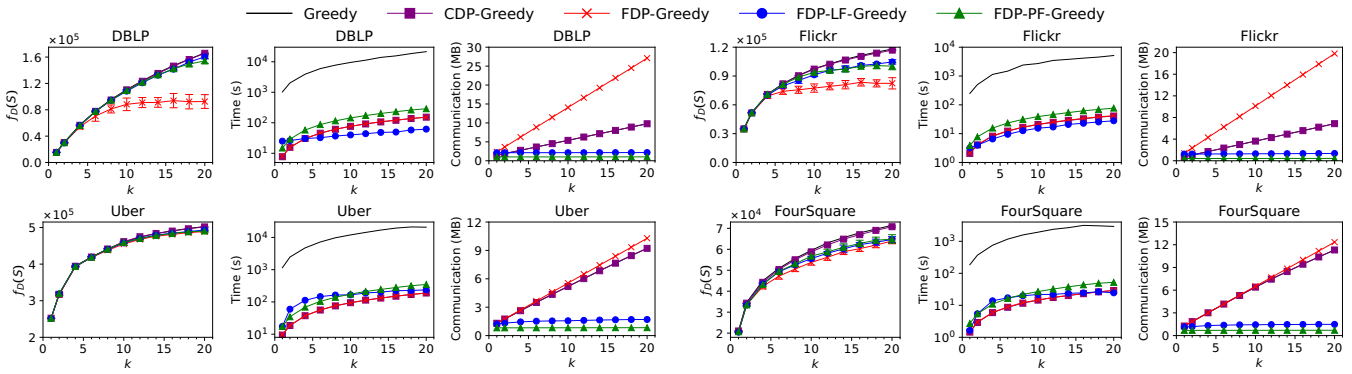


Figure 3: Performance of different algorithms with varying the solution size k from 1 to 20.

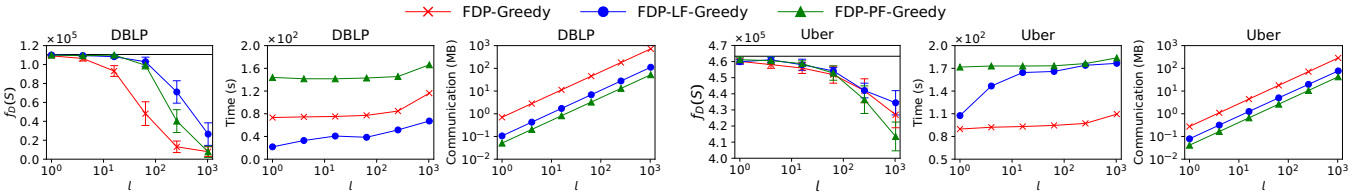


Figure 4: Scalability of FDP algorithms with varying the number of clients l.

Impact of Solution Size k. The performance of different algorithms with varying k is presented in Figure 3. Not surprisingly, each algorithm's utility, running time, and communication cost increase with k. Moreover, the utilities of the solutions of all privacy-preserving algorithms slightly degrade compared to non-private greedy solutions when k is larger. As analyzed in Section 4, all of them run in k iterations, and thus their privacy bounds are linear with k via adaptive composition. Consequently, each algorithm should add more noise with an increasing k to guarantee the same privacy budget of $\varepsilon = 2$. Moreover, the marginal gains of items decrease when k grows due to submodularity. In such cases, after introducing randomness with the Laplace and/or Permute-and-Flip mechanism, the greedy procedure cannot precisely identify the items with the largest marginal gains at later iterations. Nevertheless, FDP-LF-Greedy and FDP-PF-Greedy still provide solutions with at least 85% utilities compared with non-private ones when $k = 20$. In terms of efficiency, all algorithms with the Poisson subsampling also run about two orders of magnitude faster than Greedy. FDP-LF-Greedy runs slower than other privacy-preserving algorithms when k is small because of additional communication and queue maintenance costs. But it shows higher efficiencies when k is larger due to fewer function evaluations. Finally, the communication costs are nearly linear with k for CDP-Greedy and FDP-Greedy because their numbers of queries are proportional to k. In FDP-LF-Greedy and FDP-PF-Greedy, the communication costs are dominated by sending the data set to each client in the first iteration and only slightly increase with k.

Impact of Number l of Clients. Finally, we evaluate the scalability of different algorithms by varying the number of clients l. Note that CDP-Greedy is hardly affected by l and thus is not included in the experiments. The performance of each algorithm for $l \in \{1, 4, \ldots, 1024\}$ on the DBLP and Uber datasets are shown in Figure 4. The results for the remaining two data sets are omitted due to space limitations. We observe that the solution utilities of all algorithms are very close to those of Greedy when l ranges from 1 to 64 but begin to drop drastically when $l > 64$. According to the algorithm description, the same noise scale will be added by every client, no matter the number of clients, given that the privacy parameter is decided. As l increases, the average volume of data on each client decreases, but the noise scale remains unchanged. Thus, the greedy selection procedure is significantly disturbed when l is large. In terms of efficiency, the overall computational cost of the server and all clients slightly increases with l due to extra communication overhead since the total amount of transferred data is strictly linear with l. To summarize, our algorithms are scalable to massive data sets with an adequate number of clients (typically up to $l \approx 50$). We suggest that each client should contain enough data to ensure the effectiveness of federated submodular maximization.

6 CONCLUSION

In this paper, we systemically investigated the problem of constrained submodular function maximization in the federated setting. We adopted a widely used *client-level differential privacy* for the federated submodular maximization problem. Then, we proposed an approximation algorithm for federated submodular maximization, along with two heuristic methods with improved performance. Finally, extensive experiments on several real-world datasets confirmed the effectiveness and efficiency of our proposed algorithms.

While a first step towards differentially private submodular maximization in the federated setting, our work leaves several open problems for future exploration. For example, it would be interesting to extend our results to non-decomposable and non-monotone submodular functions. Another possible direction is to reduce the privacy budget by improving the privacy accounting schemes.

A LIST OF FREQUENT NOTATIONS

We summarize the frequently used notations in this paper in Table 2.

Table 2: List of frequently used notations.

Symbol	Description
D, D_j	A data set and its subset at client j
$f_D : 2^V \to \mathbb{R}$	A submodular set function on ground set V for data set D
n, m	The sizes of data set D and ground set V
l	The number of clients in data federation
$\mathcal{I} \subseteq 2^V$	A set of feasible solutions defined by any given constraint
k	The solution size in a cardinality constraint
$\mathcal{M}$	A (ε, δ)-differential privacy mechanism with $\varepsilon, \delta \geq 0$
$q_D(v, S)$	A query for the marginal gain of item v w.r.t. S on D
$q_{D_j}(v, S)$	A query to client j for the marginal gain of v w.r.t. S on D_j
γ	The Poisson sampling rate in $(0, 1]$
$\widehat{D}, \widehat{D_j}$	The Poisson sampled subsets of D, D_j
$\tilde{q}_D(v, S), \tilde{q}_{D_j}(v, S)$	The perturbed results for queries $q_D(v, S), q_{D_j}(v, S)$
c	The cut-off parameter in the LF or PF heuristics

B PSEUDOCODE

In this section, we provide the pseudocode of each algorithm omitted in Section 4 due to space limitations. Specifically, Algorithms 2 and 3 depict the procedures of the extended versions of FDP-Greedy where the lazy-forward optimization and the Permute-and-Flip mechanism are incorporated, respectively.

C OMITTED PROOFS

In this section, we provide the proofs of theorems and lemmas omitted in Section 4 due to space limitations.

C.1 Proof of Lemma 4.3

Proof. First of all, in the Poisson sampling procedure, whether one tuple is sampled is independent of each other. Therefore, $\widehat{D} = \bigcup_{j=1}^{l} \widehat{D_j}$ is a Poisson sample of D (with the same sampling rate γ). Next, we analyze the impact of γ on the difference between the original marginal gain $q_D(v, \tilde{S})$ and the marginal gain $q_{\widehat{D}}(v, \tilde{S})$ on the subsampled data set (without Laplace noise). Let X_i be a random variable $Y_i \cdot \Delta_{f_{x_i}}(v, \tilde{S})$. Obviously, $\mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \gamma \cdot q_D(v, \tilde{S})$ and $\sum_{i=1}^{n} X_i = q_{\widehat{D}}(v, \tilde{S})$. By applying Hoeffding's inequality [22], we have the following result:

$$\Pr\left(\left|\gamma \cdot q_D(v, \tilde{S}) - q_{\widehat{D}}(v, \tilde{S})\right| > t\right) \leq 2\exp\left(-\frac{2t^2}{n}\right).$$

Or equivalently, we have $\left|q_D(v, \tilde{S}) - \frac{q_{\widehat{D}}(v, \tilde{S})}{\gamma}\right| \leq \frac{1}{\gamma} \cdot \sqrt{\frac{n \ln m}{2}}$ with probability at least $1 - \frac{2}{m}$. By using Boole's inequality (known as the union bound), it holds with probability at least $1 - \frac{1}{m}$ that

$$\left|q_D(v, \tilde{S}) - \frac{q_{\widehat{D}}(v, \tilde{S})}{\gamma}\right| \leq \frac{\sqrt{nk \ln m}}{\gamma} \tag{5}$$

for all m^k sets $\tilde{S} \subseteq V$ of size at most k and all m items $v \in V$.

Furthermore, we observe that each client j perturbs every marginal gain via the Laplace mechanism by adding an i.i.d. random variable $Y_j \sim \mathcal{L}(\frac{1}{\varepsilon_0})$ (recall that $\Delta q = 1$ for Δ_f). Thus, we have $\tilde{q}_{\widehat{D}}(v, \tilde{S}) - q_{\widehat{D}}(v, \tilde{S}) = \sum_{j=1}^{l} Y_j$, where Y_j is i.i.d. from $\mathcal{L}(\frac{1}{\varepsilon_0})$. According to the Central Limit Theorem, the sum of l Laplace variables $\sum_{j=1}^{l} Y_j$ is approximately normally distributed with 0 mean and $\frac{2l}{\varepsilon_0^2}$ variance, because the variance of $\mathcal{L}(b)$ is $2b^2$. Therefore, we have $|\tilde{q}_{\widehat{D}}(v, \tilde{S}) - q_{\widehat{D}}(v, \tilde{S})| \leq \frac{3\sqrt{2l}}{\varepsilon_0}$ with probability at least 99.7% according to the PDF of the normal distribution. Given the above results, for any sampling rate $\gamma \in (0, 1)$, it holds with probability at least $0.997 - \frac{1}{m}$ that

$$\left|q_D(v, \tilde{S}) - \frac{\tilde{q}_{\widehat{D}}(v, \tilde{S})}{\gamma}\right| \leq \frac{1}{\gamma} \cdot \left(\sqrt{nk \ln m} + \frac{3\sqrt{2l}}{\varepsilon_0}\right),$$

and we conclude the proof. □

Algorithm 2: FDP-LF-Greedy

Input: Data set $D = \bigcup_{j=1}^{l} D_j$ across l clients, ground set V, utility function $f_D : 2^V \to \mathbb{R}$, feasible sets $\mathcal{I} \subseteq 2^V$, sampling rate $\gamma \in (0, 1]$, privacy parameter $\varepsilon_0 > 0$, cut-off $c \in \mathbb{Z}^+$
Output: Set $\tilde{S}$ such that $\tilde{S} \in \mathcal{I}$

Initialize $\tilde{S} \leftarrow \emptyset$ and $r \leftarrow 1$;
Create an empty priority queue Q, where all the items $v \in V$ are sorted in descending order of $\delta(v)$;
foreach $v \in V$ **do**
  foreach *client* $j = 1, \ldots, l$ **do**
    $\widehat{D_j} \leftarrow \text{PossionSample}(\gamma, D_j)$;
    $q_{\widehat{D_j}}(v, \tilde{S}) \leftarrow f_{\widehat{D_j}}(\{v\})$;
    $\tilde{q}_{\widehat{D_j}}(v, \tilde{S}) \leftarrow \mathcal{M}_L(q_{\widehat{D_j}}(v, \tilde{S}), \widehat{D_j}, \varepsilon_0)$;
  $\tilde{q}_{\widehat{D}}(v, \tilde{S}) \leftarrow \sum_{j=1}^{l} \tilde{q}_{\widehat{D_j}}(v, \tilde{S})$;
  Insert v into Q with $\delta(v) = \tilde{q}_{\widehat{D}}(v, \tilde{S})$ and $r(v) = r$;
$\tilde{v}^* \leftarrow \arg\max_{v \in V} \tilde{q}_{\widehat{D}}(v, \tilde{S})$;
Update $\tilde{S} \leftarrow \tilde{S} \cup \{\tilde{v}^*\}$ and $r \leftarrow r + 1$;
while *there exists any item* $v \in V \setminus \tilde{S}$ *such that* $\tilde{S} \cup \{v\} \in \mathcal{I}$ **do**
  Reset $\tilde{v}^* \leftarrow \emptyset$;
  for $i \leftarrow 0, 1, \ldots, c$ **do**
    Pop the top item from Q as v';
    if $r(v') = r$ **then**
      $\tilde{v}^* \leftarrow v'$ and **break**;
    else
      foreach *client* $j = 1, \ldots, l$ **do**
        $\widehat{D_j} \leftarrow \text{PossionSample}(\gamma, D_j)$;
        $q_{\widehat{D_j}}(v', \tilde{S}) \leftarrow f_{\widehat{D_j}}(\tilde{S} \cup \{v'\}) - f_{\widehat{D_j}}(\tilde{S})$;
        $\tilde{q}_{\widehat{D_j}}(v', \tilde{S}) \leftarrow \mathcal{M}_L(q_{\widehat{D_j}}(v', \tilde{S}), \widehat{D_j}, \varepsilon_0)$;
      $\tilde{q}_{\widehat{D}}(v', \tilde{S}) \leftarrow \sum_{j=1}^{l} \tilde{q}_{\widehat{D_j}}(v', \tilde{S})$;
      if $\tilde{q}_{\widehat{D}}(v', \tilde{S}) > \tilde{q}_{\widehat{D}}(\tilde{v}^*, \tilde{S})$ **then** $\tilde{v}^* \leftarrow v'$;
      Insert v' into Q with $\delta(v) = \tilde{q}_{\widehat{D}}(v, \tilde{S}_r)$ and $r(v) = r$;
  Update $\tilde{S} \leftarrow \tilde{S} \cup \{\tilde{v}^*\}$ and $r \leftarrow r + 1$;
return $\tilde{S}$

C.2 Proof of Theorem 4.4

Proof. First of all, the privacy bound is a simple combination of Equations (1) and (2) with Lemma 4.2.

Then, the approximation result can be adapted from [49, 50] as follows. Let $\tilde{v}_i$ and $\tilde{S}_i$ denote the item added and partial solution obtained at the i-th iteration. Suppose that $\alpha = O\left(\frac{\sqrt{nk \log m}}{\gamma} + \frac{\sqrt{l}}{\gamma\varepsilon_0}\right)$.

Algorithm 3: FDP-PF-Greedy

Input: Data set $D = \bigcup_{j=1}^{l} D_j$ across l clients, ground set V, utility function $f_D : 2^V \to \mathbb{R}$, feasible sets $\mathcal{I} \subseteq 2^V$, sampling rate $\gamma \in (0, 1]$, privacy parameter $\varepsilon_1, \varepsilon_2 > 0$, cut-off $c \in \mathbb{Z}^+$
Output: Set $\tilde{S}$ such that $\tilde{S} \in \mathcal{I}$

Initialize $\tilde{S} \leftarrow \emptyset$;
while *there exists any item* $v \in V \setminus \tilde{S}$ *such that* $\tilde{S} \cup \{v\} \in \mathcal{I}$ **do**
  $\tilde{q}(v) \leftarrow 0$ for each $v \in V \setminus \tilde{S}$;
  $V_j \leftarrow \{v \in V \setminus \tilde{S} : \tilde{S} \cup \{v\} \in \mathcal{I}\}$ for each $j \in [l]$;
  for $i \leftarrow 0, 1, \ldots, c$ **do**
    foreach *client* $j = 1, \ldots, l$ **do**
      $\widehat{D_j} \leftarrow \text{PossionSample}(\gamma, D_j)$;
      $\tilde{v}_{ij} \leftarrow \mathcal{M}_{PF}(q_{\widehat{D_j}}(\cdot, \tilde{S}), \widehat{D_j}, \varepsilon_1)$;
      $\tilde{q}_j(\tilde{v}_{ij}) \leftarrow \mathcal{M}_L(q_{\widehat{D_j}}(\tilde{v}_{ij}, \tilde{S}), \widehat{D_j}, \varepsilon_2)$;
    for $j = 1, 2, \ldots, l$ **do**
      $\tilde{q}(\tilde{v}_{ij}) \leftarrow \tilde{q}(\tilde{v}_{ij}) + \tilde{q}_j(\tilde{v}_{ij})$;
      $V_j \leftarrow V_j \setminus \{\tilde{v}_{ij}\}$;
  $\tilde{v}^* \leftarrow \arg\max_{v \in V \setminus \tilde{S} : \tilde{S} \cup \{v\} \in \mathcal{I}} \tilde{q}(v)$;
  Update $\tilde{S} \leftarrow \tilde{S} \cup \{\tilde{v}^*\}$;
return $\tilde{S}$

According to Lemma 4.3, we have $\left| q_D(v, \tilde{S}) - \frac{\tilde{q}_{\widehat{D}}(v,\tilde{S})}{\gamma} \right| \leq \alpha$ for every $v \in V$ and $\tilde{S} \in V$ with $|\tilde{S}| \leq k$ with probability at least $0.997 - \frac{1}{m}$. Thus, $\mathbb{E}\left[\Delta_{f_D}(\tilde{v}_i, \tilde{S}_{i-1})\right] \geq \max_{v \in V \setminus \tilde{S}_{i-1}} \Delta_{f_D}(v, \tilde{S}_{i-1}) - \alpha$ with probability at least $0.994 - \frac{2}{m}$. Let $S^* = \arg\max_{S \subseteq V : |S| = k} f_D(S)$ and $\mathsf{OPT} = f_D(S^*)$. We have

$$\begin{aligned}
\mathbb{E}\left[\Delta_{f_D}(\tilde{v}_i, \tilde{S}_{i-1})\right] &\geq \max_{v \in V \setminus \tilde{S}_{i-1}} \Delta_{f_D}(v, \tilde{S}_{i-1}) - \alpha \\
&\geq \frac{1}{k} \sum_{v \in S^*} \Delta_{f_D}(v, \tilde{S}_{i-1}) - \alpha \\
&\geq \frac{1}{k} \cdot \left(f_D(S^* \cup \tilde{S}_{i-1}) - f_D(\tilde{S}_{i-1}) \right) - \alpha \\
&\geq \frac{1}{k} \cdot \left(\mathsf{OPT} - f_D(\tilde{S}_{i-1}) \right) - \alpha \\
&= \frac{1}{k} \cdot \left(\mathsf{OPT} - \mathbb{E}\left[f_D(\tilde{S}_{i-1}) \right] \right) - \alpha
\end{aligned}$$

where the second inequality holds for the fact that the item to compare has the largest marginal gain w.r.t. $\tilde{S}_{i-1}$, the third inequality is based on the submodularity of f_D, the fourth inequality is according to the monotonicity of f_D, and the last equality is obtained by taking the expectation. Since $\Delta_{f_D}(\tilde{v}_i, \tilde{S}_{i-1}) = f_D(\tilde{S}_i) - f_D(\tilde{S}_{i-1})$, by rearranging the above inequalities, we have

$$\mathsf{OPT} - \mathbb{E}\left[f_D(\tilde{S}_i) \right] \leq \left(1 - \frac{1}{k}\right) \cdot \left(\mathsf{OPT} - \mathbb{E}\left[f_D(\tilde{S}_{i-1}) \right] \right) + \alpha \quad (6)$$

Using Equation (6) recursively for $i' = i, i-1, \cdots, 1$, we have

$$\begin{aligned}
\mathsf{OPT} - \mathbb{E}\left[f_D(\tilde{S}_i) \right] &\leq \left(1 - \frac{1}{k}\right)^i \cdot \left(\mathsf{OPT} - \mathbb{E}\left[f_D(\tilde{S}_0) \right] \right) + i\alpha \\
&= \left(1 - \frac{1}{k}\right)^i \cdot \mathsf{OPT} + i\alpha
\end{aligned}$$

Hence, by setting $i = k$, we obtain that

$$\mathbb{E}\left[f_D(\tilde{S}) \right] \leq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) \cdot \mathsf{OPT} - k\alpha \leq \left(1 - \frac{1}{e}\right) \cdot \mathsf{OPT} - k\alpha \quad (7)$$

Based on Equation (2), we have $\varepsilon_0 = O(\sqrt{\varepsilon/mk})$. In addition, Equation (7) will hold only if Equation (4) holds for each $i \in [k]$ and thus has a probability of at least $0.994^k - \frac{2k}{m}$. Therefore, we conclude the proof by taking them into Equation (7).

Finally, the number of function calls to compute $q_{\widehat{D}}(v, \tilde{S})$ is $O(n\gamma)$. In addition, Algorithm 1 has k iterations and there are $O(m)$ items to evaluate per iteration. Therefore, the time complexity of Algorithm 1 is $O(nmk\gamma)$. □

C.3 Proof of Lemma 4.5

Proof. First of all, by applying the same analyses for Poisson subsampling as shown in Lemma 4.3, it holds with probability at least $1 - \frac{1}{m}$ that

$$\left| q_{D_j}(v, \tilde{S}) - \frac{q_{\widehat{D_j}}(v, \tilde{S})}{\gamma} \right| \leq \frac{\sqrt{nk \ln m}}{\gamma} \quad (8)$$

for all m^k sets $\tilde{S} \subseteq V$ of size at most k and all m items $v \in V_j$. By applying Equation (3) when $t = \ln m$, we have

$$\mathbb{E}[q_{\widehat{D_j}}(\tilde{v}_{ij}, \tilde{S})] \geq \max_{v \in V_j} q_{\widehat{D_j}}(v, \tilde{S}) - \frac{4 \ln m}{\varepsilon_1} \quad (9)$$

with probability at least $1 - \frac{1}{m}$. Let $v^* = \arg\max_{v \in V_j} q_{D_j}(v, \tilde{S})$ and $\tilde{v}^* = \arg\max_{v \in V_j} q_{\widehat{D_j}}(v, \tilde{S})$. Based on the one-sided version of Equation (8), we have $q_{D_j}(v^*) - \frac{q_{\widehat{D_j}}(\tilde{v}^*)}{\gamma} \leq \frac{2\sqrt{nk \ln m}}{\gamma}$ with probability at least $1 - \frac{1}{m}$. According to this result along with Equation (9), we have

$$\mathbb{E}\left[\frac{q_{\widehat{D_j}}(\tilde{v}_{ij}, \tilde{S})}{\gamma} \right] \geq \max_{v \in V_j} q_{D_j}(v, \tilde{S}) - \frac{1}{\gamma} \cdot \left(2\sqrt{nk \ln m} + \frac{4 \ln m}{\varepsilon_1} \right) \quad (10)$$

with probability at least $1 - \frac{2}{m}$. Furthermore, due to the fact that $\Pr[|Y| \geq tb] = \exp(-t)$ if $Y \sim \mathcal{L}(b)$, we have

$$|q_j(\tilde{v}_{ij}) - q_{\widehat{D_j}}(\tilde{v}_{ij}, \tilde{S})| \leq \frac{\ln m}{\varepsilon_2} \quad (11)$$

with probability at least $1 - \frac{1}{m}$. Finally, by combining Equations (10) and (11), we have

$$\mathbb{E}\left[\frac{q_j(\tilde{v}_{ij})}{\gamma} \right] \geq \max_{v \in V_j} q_{D_j}(v, \tilde{S}) - \frac{1}{\gamma} \cdot \left(2\sqrt{nk \ln m} + \frac{4 \ln m}{\varepsilon_1} + \frac{\ln m}{\varepsilon_2} \right)$$

with probability at least $1 - \frac{3}{m}$ and conclude the proof. □

D ADDITIONAL EXPERIMENTS

Parameter Tuning. We first show the performance of CDP-Greedy and FDP-Greedy with varying γ from 0.001 to 1 in Figure 5. We observe that the running time of each algorithm is nearly linear with γ, as their time complexities are both $O(nmk\gamma)$. In addition, the utilities of the solutions of both algorithms keep steady when $\gamma \geq 0.01$. On the one hand, the error in the estimation of $f_D(S)$ increases for smaller values of γ; on the other hand, the privacy amplification by subsampling allows less noise to be added to the

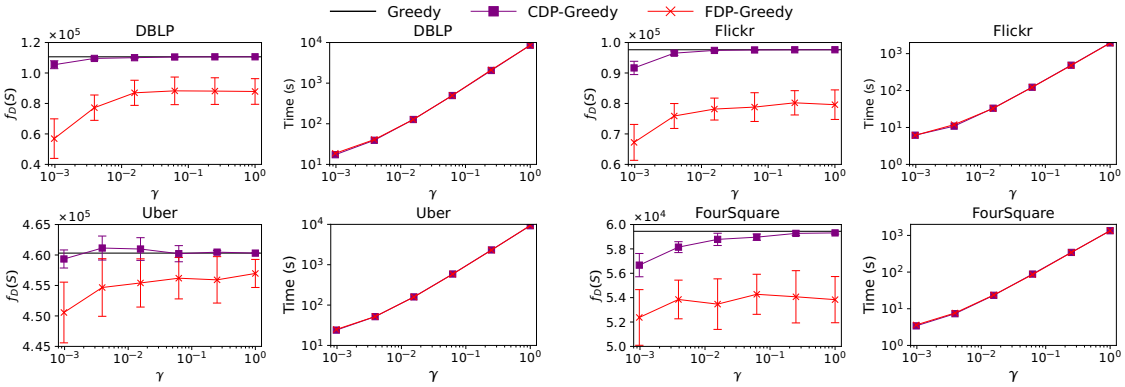


Figure 5: Effect of sampling rate γ on the performance of CDP-Greedy and FDP-Greedy.

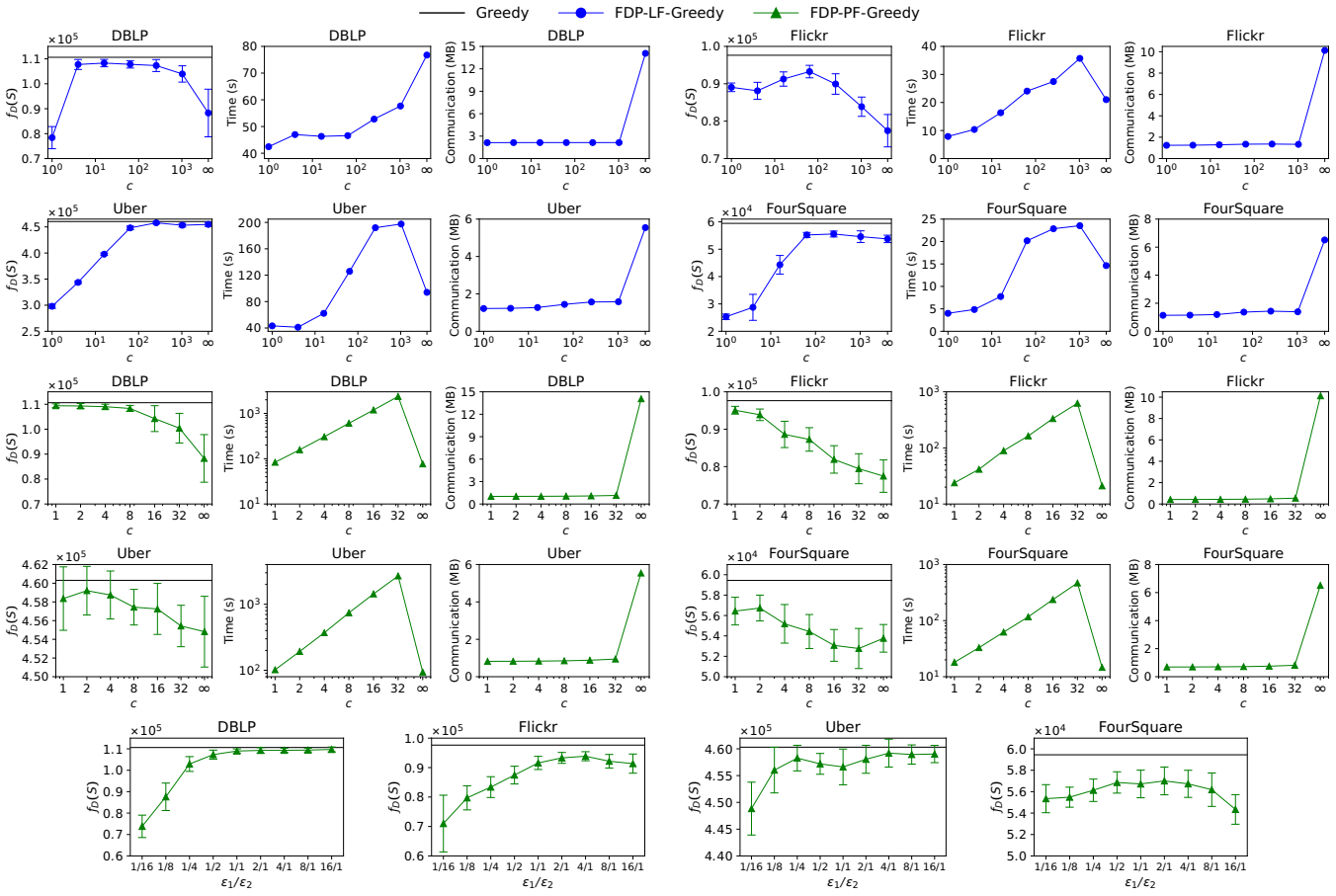


Figure 6: Impact of parameters c and $\varepsilon_1/\varepsilon_2$ on the performance of FDP-LF-Greedy and FDP-PF-Greedy.

result. The two factors cancel out when γ drops from 1 to 0.01, yet the former overweighs the latter when γ is smaller. Thus, we set $\gamma = 0.01$ for CDP-Greedy and FDP-Greedy by default in all remaining experiments.

The performance of FDP-LF-Greedy when the cut-off threshold c is varied over $\{1, 4, \ldots, 1024\}$ and ∞ (i.e., no cut-off) is shown in the first two rows of Figure 6. We observe different trends in two applications. For maximum coverage, the quality of solutions first grows with increasing c but then drops when c is large due to the high privacy budget. For facility location, it cannot provide a good solution until $c \geq 64$, but the solution quality keeps stable for larger c. Such a phenomenon can be explained by the distribution of marginal gains across items. For maximum coverage, the marginal gains of a few items are much larger than the remaining ones, and a small cut-off c can filter them out. For facility location, the marginal gain of any facility decreases significantly when a facility close to it

is included in the solution and thus a large cut-off c is necessary to find a new facility that is distant from all existing ones. The running time of FDP-LF-Greedy increases with c on both applications. When c is large, it often runs slower than FDP-Greedy (without cut-offs) due to additional queue maintenance and communication costs. In the remaining experiments, we set $c = 16$ for maximum coverage and $c = 256$ for facility location by default.

The performance of FDP-PF-Greedy when the cut-off threshold c is varied over $\{1, 2, \ldots, 32\}$ and ∞ (i.e., no cut-off) is shown in the third and fourth rows of Figure 6. In general, we find that the running time increases nearly linearly with c due to its time complexity, while the utilities of the solutions drop significantly when $c > 4$ because of high privacy budgets. Therefore, we set $c = 2$ for FDP-PF-Greedy in the remaining experiments. Finally, the ratio of ε_1 and ε_2 for the Permute-and-Flip and Laplace mechanisms can also affect the performance of FDP-PF-Greedy, as shown in the fifth row of Figure 6. In most cases, it achieves high utilities when $\varepsilon_1/\varepsilon_2$ is between 2 and 8, and thus we fix the ratio $\varepsilon_1/\varepsilon_2 = 4$ in the remaining experiments.

REFERENCES

[1] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. 2014. Streaming Submodular Maximization: Massive Data Summarization on the Fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 671–680.

[2] Eugene Bagdasaryan, Peter Kairouz, Stefan Mellem, Adrià Gascón, Kallista A. Bonawitz, Deborah Estrin, and Marco Gruteser. 2022. Towards Sparse Federated Analytics: Location Heatmaps under Distributed Differential Privacy with Secure Aggregation. *Proc. Priv. Enhancing Technol.* 2022, 4 (2022), 162–182.

[3] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS '18)*. Curran Associates Inc., Red Hook, NY, USA, 6280–6290.

[4] Akash Bharadwaj and Graham Cormode. 2022. An Introduction to Federated Computation. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. ACM, New York, NY, USA, 2448–2451.

[5] Mark Bun and Thomas Steinke. 2016. Concentrated Differential Privacy: Simplifications, Extensions, and Lower Bounds. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*. Springer, Berlin, Heidelberg, 635–658.

[6] Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2011. Maximizing a Monotone Submodular Function Subject to a Matroid Constraint. *SIAM J. Comput.* 40, 6 (2011), 1740–1766.

[7] Adrian Rivera Cardoso and Rachel Cummings. 2019. Differentially Private Online Submodular Minimization. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics (AISTATS '19)*. PMLR, 1650–1658.

[8] Anamay Chaturvedi, Huy Le Nguyen, and Lydia Zakynthinou. 2021. Differentially Private Decomposable Submodular Maximization. *Proc. AAAI Conf. Artif. Intell.* 35, 8 (2021), 6984–6992.

[9] Graham Cormode and Akash Bharadwaj. 2022. Sample-and-threshold differential privacy: Histograms and applications. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (AISTATS '22)*. PMLR, 1420–1431.

[10] Rafael da Ponte Barbosa, Alina Ene, Huy L. Nguyen, and Justin Ward. 2015. The Power of Randomization: Distributed Submodular Maximization on Massive Datasets. In *Proceedings of the 32nd International Conference on Machine Learning (ICML '15)*. PMLR, 1236–1244.

[11] Cynthia Dwork. 2008. Differential Privacy: A Survey of Results. In *Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings*. Springer, Berlin, Heidelberg, 1–19.

[12] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. Springer, Berlin, Heidelberg, 265–284.

[13] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407.

[14] Alessandro Epasto, Vahab S. Mirrokni, and Morteza Zadimoghaddam. 2017. Bicriteria Distributed Submodular Maximization in a Few Rounds. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '17)*. ACM, New York, NY, USA, 25–33.

[15] Uriel Feige. 1998. A Threshold of ln n for Approximating Set Cover. *J. ACM* 45, 4 (1998), 634–652.

[16] Moran Feldman, Joseph Naor, and Roy Schwartz. 2011. A Unified Continuous Greedy Algorithm for Submodular Maximization. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 570–579.

[17] Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. 1978. An analysis of approximations for maximizing submodular set functions—II. In *Polyhedral Combinatorics: Dedicated to the memory of D.R. Fulkerson*, Michel L. Balinski and Alan J. Hoffman (Eds.). Springer, Berlin, Heidelberg, 73–87.

[18] Antonious M. Girgis, Deepesh Data, Suhas N. Diggavi, Peter Kairouz, and Ananda Theertha Suresh. 2021. Shuffled Model of Differential Privacy in Federated Learning. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS '21)*. PMLR, 2521–2529.

[19] Andreas Grammenos, Rodrigo Mendoza-Smith, Jon Crowcroft, and Cecilia Mascolo. 2020. Federated Principal Component Analysis. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, 6453–6464.

[20] Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. 2010. Differentially Private Combinatorial Optimization. In *Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 1106–1125.

[21] Marwa El Halabi, Slobodan Mitrovic, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. 2020. Fairness in Streaming Submodular Maximization: Algorithms and Hardness. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, 13609–13622.

[22] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Am. Stat. Assoc.* 58, 301 (1963), 13–30.

[23] Ziyue Huang, Yuan Qiu, Ke Yi, and Graham Cormode. 2022. Frequency Estimation Under Multiparty Differential Privacy: One-shot and Streaming. *Proc. VLDB Endow.* 15, 10 (2022), 2058–2070.

[24] Peter Kairouz *et al.* 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210.

[25] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2011. What Can We Learn Privately? *SIAM J. Comput.* 40, 3 (2011), 793–826.

[26] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*. ACM, New York, NY, USA, 137–146.

[27] Bernhard Korte and Jens Vygen. 2012. *Combinatorial Optimization: Theory and Algorithms*. Springer, Berlin, Heidelberg.

[28] Andreas Krause and Daniel Golovin. 2014. Submodular Function Maximization. In *Tractability: Practical Approaches to Hard Problems*, Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli (Eds.). Cambridge University Press, Cambridge, UK, 71–104.

[29] Ariel Kulik, Hadas Shachnai, and Tami Tamir. 2009. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '09)*. SIAM, 545–554.

[30] Ravi Kumar, Benjamin Moseley, Sergei Vassilvitskii, and Andrea Vattani. 2015. Fast Greedy Algorithms in MapReduce and Streaming. *ACM Trans. Parallel Comput.* 2, 3, Article 14 (2015), 22 pages.

[31] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne M. Van Briesen, and Natalie S. Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*. ACM, New York, NY, USA, 420–429.

[32] Ninghui Li, Wahbeh H. Qardaji, and Dong Su. 2012. On Sampling, Anonymization, and Differential Privacy or, k-Anonymization Meets Differential Privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS '12)*. ACM, New York, NY, USA, 32–33.

[33] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* 37, 3 (2020), 50–60.

[34] Yuchen Li, Ju Fan, Yanhao Wang, and Kian-Lee Tan. 2018. Influence Maximization on Social Graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* 30, 10 (2018), 1852–1872.

[35] Zitao Li, Bolin Ding, Ce Zhang, Ninghui Li, and Jingren Zhou. 2021. Federated Matrix Factorization with Privacy Guarantee. *Proc. VLDB Endow.* 15, 4 (2021), 900–913.

[36] Hui Lin and Jeff A. Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. ACL, Portland, OR, USA, 510–520.

[37] Erik M. Lindgren, Shanshan Wu, and Alexandros G. Dimakis. 2016. Leveraging Sparsity for Efficient Submodular Data Summarization. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS '16)*. Curran Associates Inc., Red Hook, NY, USA, 3414–3422.

[38] Junxu Liu, Jian Lou, Li Xiong, Jinfei Liu, and Xiaofeng Meng. 2021. Projected Federated Averaging with Heterogeneous Differential Privacy. *Proc. VLDB Endow.* 15, 4 (2021), 828–840.

[39] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. 2020. FedSel: Federated SGD Under Local Differential Privacy with Top-k Dimension Selection. In *Database Systems for Advanced Applications - 25th International Conference, DASFAA 2020, Jeju, South Korea, September 24-27, 2020, Proceedings, Part I*. Springer, Berlin, Heidelberg, 485–501.

[40] Ryan McKenna and Daniel Sheldon. 2020. Permute-and-Flip: A new mechanism for differentially private selection. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, 193–203.

[41] Frank McSherry. 2009. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data (SIGMOD '09)*. ACM, New York, NY, USA, 19–30.

[42] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS '07)*. IEEE, 94–103.

[43] Julián Mestre. 2006. Greedy in Approximation Algorithms. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*. Springer, Berlin, Heidelberg, 528–539.

[44] Vahab S. Mirrokni and Morteza Zadimoghaddam. 2015. Randomized Composable Core-sets for Distributed Submodular Maximization. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing (STOC '15)*. ACM, New York, NY, USA, 153–162.

[45] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Amin Karbasi. 2016. Fast Constrained Submodular Maximization: Personalized Data Summarization. In *Proceedings of the 33nd International Conference on Machine Learning (ICML '16)*. PMLR, 1358–1367.

[46] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. 2015. Lazier Than Lazy Greedy. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI '15)*. AAAI Press, 1812–1818.

[47] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. 2016. Distributed Submodular Maximization. *J. Mach. Learn. Res.* 17 (2016), 238:1–238:44.

[48] Alan Mislove, Massimiliano Marcon, P. Krishna Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (IMC '07)*. ACM, New York, NY, USA, 29–42.

[49] Marko Mitrovic, Mark Bun, Andreas Krause, and Amin Karbasi. 2017. Differentially Private Submodular Maximization: Data Summarization in Disguise. In *Proceedings of the 34th International Conference on Machine Learning (ICML '17)*. PMLR, 2478–2487.

[50] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Math. Program.* 14, 1 (1978), 265–294.

[51] Sebastian Perez-Salazar and Rachel Cummings. 2021. Differentially Private Online Submodular Maximization. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS '21)*. PMLR, 1279–1287.

[52] Akbar Rafiey and Yuichi Yoshida. 2020. Fast and Private Submodular and k-Submodular Functions Maximization with Matroid Constraints. In *Proceedings of the 37th International Conference on Machine Learning (ICML '20)*. PMLR, 7887–7897.

[53] Omid Sadeghi and Maryam Fazel. 2021. Differentially Private Monotone Submodular Maximization Under Matroid and Knapsack Constraints. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS '21)*. PMLR, 2908–2916.

[54] Kanthi K. Sarpatwar, Karthikeyan Shanmugam, Venkata Sitaramagiridharganesh Ganapavarapu, Ashish Jagmohan, and Roman Vaculín. 2019. Differentially Private Distributed Data Summarization under Covariate Shift. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS '19)*. Curran Associates Inc., Red Hook, NY, USA, 14432–14442.

[55] Lichao Sun, Jianwei Qian, and Xun Chen. 2021. LDP-FL: Practical Private Aggregation in Federated Learning with Local Differential Privacy. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. IJCAI Organization, 1571–1578.

[56] Xin Sun, Gaidi Li, Yapu Zhang, and Zhenning Zhang. 2021. Measured Continuous Greedy with Differential Privacy. In *Algorithmic Aspects in Information and Management - 15th International Conference, AAIM 2021, Virtual Event, December 20-22, 2021, Proceedings*. Springer, Berlin, Heidelberg, 212–226.

[57] Stacey Truex, Ling Liu, Ka Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20)*. ACM, New York, NY, USA, 61–66.

[58] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2022. Fast-adapting and Privacy-preserving Federated Recommender System. *VLDB J.* 31 (2022), 877–896.

[59] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Trans. Inf. Forensics Secur.* 15 (2020), 3454–3469.

[60] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. 2015. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Trans. Syst. Man Cybern. Syst.* 45, 1 (2015), 129–142.

[61] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. 2020. Federated Heavy Hitters Discovery with Differential Privacy. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS '20)*. PMLR, 3837–3847.

[62] Yuqing Zhu and Yu-Xiang Wang. 2019. Poission Subsampled Rényi Differential Privacy. In *Proceedings of the 36th International Conference on Machine Learning (ICML '19)*. PMLR, 7634–7642.