Federated Submodular Maximization: Improved Communication Rounds and Bit Complexity

Akbar Rafiey [1] **Sreeharsh Namani** [1] **Neophytos Charalambides** [2]

Abstract

Subset selection is a fundamental problem in machine learning, data mining, game theory, and economics, where the objective is to select a subset of elements from a collection that maximizes a given submodular function. This renders submodular maximization an important and widely studied problem. However, in massively distributed settings with sensitive data, centralized iterative optimization is often infeasible and disfavored due to privacy concerns and computational constraints. To address these challenges, we propose a novel federated algorithm for maximizing a monotone submodular function where the data and computations are distributed across multiple clients. Our approach enables optimization without requiring a central server to directly access the local functions. To handle scalability, we employ client subsampling for aggregation and low-bit communication updates, reducing communication complexity – which is critical in large scale and high-dimensional settings. Our federated algorithm guarantees a $\frac{1}{2}$-approximation, and effectively captures the trade-off between the number of participating clients and bit complexity. We then propose a second algorithm for improved efficiency, which requires only logarithmically many communication rounds between the clients and the central server, while preserving the $\frac{1}{2}$-approximation guarantee. The approximation guarantees of our algorithms can be improved to $1 - \frac{1}{e}$ through amplification techniques. Our methods are validated numerically, demonstrating their effectiveness in real-world settings and corroborating our guarantees.

[1]Department of Computer Science and Engineering, New York University, Brooklyn, NY, USA [2]School of Computing Information and Data Sciences, University of California San Diego, CA, USA. Correspondence to: Akbar Rafiey <ar9530@nyu.edu>.

Accepted at the ICML 2025 Workshop on Collaborative and Federated Agentic Workflows (CFAgentic@ICML'25), Vancouver, Canada. July 19, 2025.

1. Introduction

The problem of selecting a subset of elements from a discrete universe is central in machine learning and data-mining and it essentially seeks to select a subset from a collection of elements, such that a prespecified objective is optimized. This is a highly comprehensive framework that arises in numerous application. Some of the most relevant applications to this work are those related to recommender systems (Mehrotra & Vishnoi, 2023), resource and facility allocation (Abbasi et al., 2024; Elhamifar, 2019), coverage functions (Karimi et al., 2017). For example, in recommender systems the goal is to pick a subset of items from a collection (e.g., restaurants or movies) such that the selected items best summarize the collection or best match the users' interests.

In many of the above examples the objective function can be expressed as a set function with nice structural properties. One of the properties that naturally appears is the diminishing return property. That is, as the set of selected elements increases; the marginal contribution of the new ones decreases. Specifically, for a collection of elements E, a set function $F : 2^E \to \mathbb{R}$ has the diminishing return property if for any $S \subseteq T \subseteq E$ and $e \in E \setminus T$ we have

$$F(S \cup \{e\}) - F(S) \geq F(T \cup \{e\}) - F(T). \quad (1)$$

These functions comprise the class of submodular functions and many of the subset selection tasks can be formulated as optimizing a submodular function. In our work, the goal is to pick a subset of elements S under some feasibility constraints from a collection E such that $F(S)$ is maximized, where $F(S)$ is a monotone and non-negative submodular function. Submodularity has an amazing algorithmic consequence that leads to an efficient and easily implementable iterative greedy algorithm (Nemhauser et al., 1978). This algorithm initializes S as the empty set and at each iteration greedily selects an element with maximum marginal contribution from the remaining elements, for a fixed number of iterations.

While appealing, this greedy algorithm has several drawbacks. Firstly, it is sequential and centralized, and is not emendable to real-world applications that involve distributed data across multiple devices or organizations. We elaborate

on this through an example of movie recommendations, in which service providers often deal with millions, if not billions of users, hence collecting their data (e.g., interest and ratings) in a one central server is computationally infeasible and requires substantial memory allocation. Moreover, individual users may be reluctant to share their raw data and interests with an untrusted/curious central server. This is one example which prevents *centralized* optimizations techniques and procedures from being performed by one central server, due to privacy concerns, communication constraints; and the sheer volume of data.

In an era where data is generated at unprecedented rates across distributed sources, such as smartphones, IoT devices, and organizational networks, it is challenging to perform meaningful optimization approaches without compromising privacy or efficiency. Federated settings offer an attractive alternative by enabling decentralized computations while keeping sensitive data local to its source. This paradigm aligns with privacy-preserving principles, reduces communication overheads, and supports scalable optimization in distributed environments, rendering it particularly applicable to large-scale submodular optimization tasks.

Problem definition and setting. With this in mind, we consider the problem of maximizing a submodular function in the federated setting. Let E be a ground set of size m and $c_1, \ldots, c_n$ be n clients, each of whom has a (*private*) interest over $E = \{e_1, \ldots, e_m\}$. Each client's interest is expressed as a submodular function $f_i : 2^E \to \mathbb{R}_{\geq 0}$, where i indicates the client's index. A central server wants to solve the following constrained distributed optimization problem

$$\max_{S \in \mathcal{I}} F(S), \quad \text{with} \quad F(S) = \sum_{i=1}^{n} p_i f_i(S) \tag{2}$$

where $\mathcal{I}$ is the set of *feasible solutions* (e.g., independent sets of a matroid $\mathcal{M}$ with ground set E), and $p_1, \ldots, p_n$ are pre-specified weights such that $0 < p_i < 1$ and $\sum_{i=1}^{n} p_i = 1$. For instance, they can be set to $1/n$. To simplify our presentation, we assume $p_i = \frac{1}{n}$ for all i. All our results and techniques generalize for an arbitrary simplex $p_1, \ldots, p_n$. In the optimization problem (2) the data can be massively distributed over the number of clients n, meaning that n itself can be very large. We emphasize that the ground set E is common to all clients, though f_i may differ. It is possible that different clients may have different subsets of E as their domain. What we present covers this scenario also.

Generally speaking, in federated learning (FL) settings (McMahan et al., 2017; Konečnỳ et al., 2016; Rafiey, 2024), a central server asks for client's local updates through rounds of communications with the active clients. Upon receiving the clients' updates, the central server decides on a global update. This process is repeated several times until we converge to a satisfactory solution. We emphasize, in our setting clients do not trust the central server and for privacy reasons, we assume the server has access only to the **aggregated updates** of clients; and **not individual updates**.

Federated settings come with their own challenges and considerations. Next, we discuss the most relatable to our work.

Client participation and heterogeneity. In our federated setting, unlike traditional distributed settings (Mirzasoleiman et al., 2016; Barbosa et al., 2015; Bateni et al., 2018), the central server does not have control over the clients' devices, nor on how the data is distributed. For instance, when a mobile phone is turned off or WiFi access is unavailable, it becomes unavailable to the server. The clients' objectives can differ vastly, which may depend on their respective local datasets; an issue known as heterogeneity (Qin et al., 2023). Scalability and performance of federated settings also depend significantly on the number of participating clients and the algorithm's communication efficiency. Real-world federated systems often involve billions of devices. While including more clients improves the diversity and optimization quality, it introduces practical challenges (Chen et al., 2022). Communication is a bottleneck under bandwidth constraints, and relying on all clients causes stragglers to slow down training (Lee et al., 2018; Li et al., 2020).

Bit complexity. Beyond client participation and heterogeneity, the bit complexity of the iterative and distributed updates is another factor which can prevent scalability (Basu et al., 2019; Haddadpour et al., 2021). Even when working with a limited number of clients, transmitting uncompressed updates can overwhelm bandwidth resources, and strain energy-constrained devices such as smartphones or IoT sensors. Conversely, carefully designed mechanisms which reduce bit complexity, can significantly alleviate communication costs without sacrificing performance.

1.1. Our contributions

We address the challenges of scaling to many clients, reducing bit complexity of the updates, and ensuring good approximation quality, enabling efficient and scalable submodular maximization in federated settings.

• We propose a federated algorithm for optimizing Equation (2), by requiring only r rounds of communication between the central server and clients. (Here r depends on constraints.) Our algorithm obtains $\frac{1}{2}$-approximation guarantees and effectively captures the trade-off between the number of participating clients and bit complexity. It is versatile and applicable to scenarios involving both a large number of clients and a limited number of clients.

• We also design a federated algorithm which requires a reduced number of communication rounds. Specifically, we present a method requiring only $O(\log |E|)$ communication

Method	Approximation	# clients per round	# bit per client	# communication rounds
Algorithm 1	$1/2 - O(\varepsilon)$	$\tilde{O}(m^2r^2/(d^2\varepsilon^2))$	$\tilde{O}(d)$	r
Algorithm 2	$1/2 - O(\varepsilon)$	$\tilde{O}(m^2r^4/(d^2\varepsilon^2))$	$\tilde{O}(d)$	$O(\log m \log(r/\varepsilon)\varepsilon^{-2})$
Algorithm 3 (Theorem B.2)	$1 - 1/\mathrm{e} - O(\varepsilon)$	$\tilde{O}(m^2r^2/(d^2\varepsilon^2))$	$\tilde{O}(d)$	$O(r/\varepsilon^2)$
Algorithm 3 (Theorem B.1)	$1 - 1/\mathrm{e} - O(\varepsilon)$	$\tilde{O}(m^2r^4/(d^2\varepsilon^2))$	$\tilde{O}(d)$	$O(\log m \log(r/\varepsilon^3)\varepsilon^{-3})$
Proposition C.1 (local DP)	$1/2 - O(\varepsilon)$	$K = \tilde{O}(m^2r^2/(d^2\varepsilon^2))$	$\tilde{O}(\varepsilon\, d\sqrt{K})$	r
Wang et al. (2023)	$1/2 - O(\varepsilon)$	n	$\tilde{O}(m)$	r
Rafiey (2024) (Algorithm 3, assuming $n \gg m$)	$1/2 - O(\varepsilon)$	$O(m^2r/\varepsilon^2)$	$\tilde{O}(m)$	r

Table 1: Summary of our results under a matroid constraint. All results listed in this table achieve multiplicative approximation guarantees with no additive error. Consequently, we exclude results such as Algorithms 1 and 2 from (Rafiey, 2024), which incur additive error and involve parameters not discussed in this paper. Our local DP result, Proposition C.1, has $(\varepsilon' m r, \delta' m r)$-DP guarantee with $\varepsilon' = O\left(\frac{r}{\varepsilon K} \log \frac{1}{\delta'} \sqrt{\log \frac{m r}{\delta}}\right)$. Its approximation guarantee can be improved to $1 - 1/\mathrm{e}$ using the amplification technique.

rounds; which achieves a $\frac{1}{2}$-approximation to Equation (2). Our analysis rigorously explores the interplay between the number of participating clients and the bit complexity, providing deeper insights into this trade-off.

- We further improve the approximation guarantee to $(1-1/e)$ using an amplification technique (Chekuri & Quanrud, 2019; Balkanski et al., 2019; Badanidiyuru & Vondrák, 2014), requiring only a constant number of additional communication rounds. We also extend our framework to incorporate local differential privacy, at the cost of higher bit complexity. Full details are provided in the appendix.

See Table 1 for a summary of our results.

1.2. Related work

The main approach to submodular maximization is the greedy approach, which in the centralized setting yields a tight approximation guarantee under various scenarios and constraints e.g., see (Călinescu et al., 2011; Chen & Kuhnle, 2023; Nemhauser et al., 1978; Nikolakaki et al., 2021; Vondrak, 2008). However the sequential nature of the greedy approach deems it prohibitive to scale to massive datasets. This issue is partially addressed by means of Map-Reduce style algorithms (Kumar et al., 2015), and several elegant distributed algorithms (Bateni et al., 2018; Mirzasoleiman et al., 2016; Barbosa et al., 2015), as well as in the adaptive and parallel frameworks (Balkanski et al., 2019; Balkanski & Singer, 2018; Chekuri & Quanrud, 2019).

Rafiey (2024) considered solving Equation (2) under matroid constraints and designed algorithms, Algorithms 1 and 2 in (Rafiey, 2024), based on the Continuous Greedy Algorithm (Călinescu et al., 2011) that achieves $1-1/e$ multiplicative approximation guarantee plus an additive error. The additive error depends inversely on the number of communication rounds. Rafiey's algorithmic framework, under mild heterogeneity assumptions, is very flexible and applicable to many practical settings. Algorithm 1 in (Rafiey, 2024) can handle partial client participation and requires only $\tilde{O}(r)$ bit complexity per round per active client, where $r \leq m$ is the rank of the matroid, and often $r \ll m$. They further improve this algorithm by proposing Algorithm 2 in (Rafiey, 2024), which incorporates a client's local step to reduce the number of communication rounds at the cost of a larger additive error and $\tilde{O}(m)$ bit complexity per round per active client. We point out that their algorithm requires clients to locally estimate gradients of the multilinear extension, which in turns requires clients to do roughly $O(\log(T K))$ local function evaluations, were K is the number of clients participating in a communication round and $T \geq r$ is the number of communication rounds. Assuming $n \gg m$, they also proposed a more computationally efficient algorithm, Algorithm 3 in (Rafiey, 2024), that avoids multilinear extension, leveraging sparsification techniques from (Rafiey & Yoshida, 2022). This algorithm achieves a $\frac{1}{2}$-approximation (with no additive error) under matroid constraints and requires r communication rounds, $O(m^2r/\varepsilon^2)$ active clients per round, and $\tilde{O}(m)$ bit complexity per round per active client, along with estimation of each client's importance factor.

Wang et al. (2023) considered the problem of maximizing a non-negative and monotone submodular function in federated settings, where they focus on DP. The idea is that clients add DP-noise to their updates before sharing them with the server, and they achieve a $1 - 1/e$ multiplicative approximation guarantee under a cardinality constraint; plus an additive error that is an artifact of their DP mechanism. Their algorithm requires all clients to participate in each communication round, with bit complexity per client per round of $\tilde{O}(|E|)$.

2. Preliminaries

Let E be the ground set with $|E| = m$. A submodular function $f : 2^E \to \mathbb{R}$ is monotone if $f(S) \leq f(T)$ for every $S \subseteq T \subseteq E$. Throughout the paper, we assume $f(\emptyset) = 0$. The marginal contribution of e to set S is denoted by $f(e \mid S) \coloneqq f(S \cup \{e\}) - f(S)$.

Matroids. A pair $\mathcal{M} = (E, \mathcal{I})$ of a set E and $\mathcal{I} \subseteq 2^E$ is called a *matroid* if 1) $\emptyset \in \mathcal{I}$, 2) $A \in \mathcal{I}$ for any $A \subseteq B \in \mathcal{I}$, 3) for any $A, B \in \mathcal{I}$ with $|A| < |B|$; there exists $e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$. Sets in $\mathcal{I}$ are called *independent sets*. We sometimes abuse notation to denote sets in $\mathcal{I}$ by $S \in \mathcal{M}$. The *rank function* $\texttt{rank}_{\mathcal{M}} \colon 2^E \to \mathbb{Z}_+$ of $\mathcal{M}$ is $\texttt{rank}_{\mathcal{M}}(S) = \max\{|I| : I \subseteq S, I \in \mathcal{I}\}$. An independent set $S \in \mathcal{I}$ is called a *basis* if $\texttt{rank}_{\mathcal{M}}(S) = \texttt{rank}_{\mathcal{M}}(E)$. Denote the rank of $\mathcal{M}$ by $\texttt{rank}(\mathcal{M})$; or r when it is clear from the context, and $\texttt{OPT} = \max_{S \in \mathcal{I}} F(S)$.

3. Low bit complexity with unbiased clients

In this section, we present our federated algorithm, Algorithm 1, in the low bit complexity regime. The central server initializes S as the empty set $S = \emptyset$ and after each communication round adds an element to the current set S. The element selection is greedy, and is based on the information received from the clients. Since the objective is to find a subset $S \in \mathcal{I}$ to maximize $F(S)$ for monotone F, it is sensible to include as many elements as possible, as long as we do not violate the constraint $S \in \mathcal{I}$. Hence, the number of communication rounds is set to be $r = \texttt{rank}(\mathcal{M})$.

At each communication round the server selects K active clients uniformly at random; this selection is independent of the clients' local functions and any other local information of theirs. Let $A^{(t)}$ denote the set of sampled clients selected in iteration t. The central server broadcasts the current set S to all the clients with indices in $A^{(t)}$, each of which does a local computation in parallel. These clients then sample d *distinct* elements $D_i = \{e_{i1}, \ldots, e_{id}\}$ from $E \setminus S$ uniformly at random, for each $i \in A^{(t)}$. We emphasize that this sampling does not require any function evaluation, nor does it require any matroid oracle queries. The marginal contribution of each $e_{ij} \in D_i$ is computed with respect to the local function f_i, and each client i sends $f_i(S \cup \{e_{ij}\}) - f_i(S)$ for all $j \in [d]$, which requires only $\tilde{O}(d)$ bits to encode.

Ultimately, the goal of each communication round is for the central server to *learn* an estimate of marginal contributions of the remaining elements. The clients' updates are scaled by a factor of $\frac{|E \setminus S|}{d\,K}$, to obtain *unbiased estimates* of the marginal contributions of all elements in $E \setminus S$. Let $\widehat{F}\big(e \mid S\big) = \frac{|E \setminus S|}{d\,K} \sum_{i \in A^{(t)}, e \in D_i} f_i\big(e \mid S\big)$ denote the (securely) aggregated updates of clients for $e \in E \setminus S$.

Algorithm 1 FedSM with d-bits and K clients

Input: Matroid $\mathcal{M} = (E, \mathcal{I})$ of rank r, $\delta, \varepsilon \in (0, 1)$.
$d \leftarrow$ an integer in $[1, m - r]$, and $\lambda = \max_{e,i} f_i(\{e\})$
$K \leftarrow \frac{\left(\frac{m}{d}\right)^2 r^2\, \lambda^2\, \ln\left(\frac{2\,r\,m}{\delta}\right)}{8\,\varepsilon^2}$, $S \leftarrow \emptyset$.
for $t = 0$ to $r - 1$ **do**
  Server selects a subset of K *active* clients $A^{(t)}$ uniformly at random, and sends S to them.
  for each client $i \in A^{(t)}$ in parallel **do**
    Sample d *distinct* elements $D_i = \{e_{i1}, \ldots, e_{id}\}$ from $E \setminus S$ uniformly at random.
    for each $e \in D_i$ send back to the (secure) aggregator **do**
      / Scale by $\frac{|E \setminus S|}{d\,K}$ to remain unbiased */*
      $\big(\frac{|E \setminus S|}{d\,K}(f_i(S \cup \{e\}) - f_i(S)), e\big)$
    end for
  end for
  / (Securely) aggregate the reports to form estimates: */*
  For each element $e \in E \setminus S$ define

$$\widehat{F}\big(e \mid S\big) = \frac{|E \setminus S|}{d\,K} \sum_{i \in A^{(t)},\ e \in D_i} f_i\big(e \mid S\big). \qquad (3)$$

  Server updates: $S \leftarrow S \cup \Big\{\operatorname{argmax}_{e : S \cup \{e\} \in \mathcal{I}} \widehat{F}\big(e \mid S\big)\Big\}$.
end for
Output: S

Then, the central server performs a greedy selection step $S \leftarrow S \cup \Big\{\operatorname{argmax}_{e :\, S \cup \{e\} \in \mathcal{I}} \widehat{F}\big(e \mid S\big)\Big\}$.

The utility of S returned by our algorithm depends on how accurately the central server estimates the marginal contributions from client feedback. Due to randomness in client selection at the server and element sampling at the clients, the server's estimates of marginal contributions may not be accurate. To mitigate this issue, the central server can sample a larger number of clients at each round. Additionally, each client may need to sample a substantial number of elements D_i. Next, we outline the trade-offs associated with different parameter choices.

Different choices of K and d. Depending on the central server's computational limitations and the dataset e.g., number of clients n and $|E|$, different choices of K and d may be more suitable. For instance, the scenario where $K = n$ and $d = O(m)$, realizes the centralized greedy algorithm (Nemhauser et al., 1978). In this scenario, at each communication round the server learns the exact marginal contribution of each element $F(e \mid S) = \frac{1}{n} \sum_{i=1}^{n} f_i(e \mid S)$. In scenarios where the number of clients is much larger than the size of the ground set i.e., $n \gg m$, setting $d = 1$ and $K = \tilde{O}(m^2\, r^2\, \lambda^2 / \varepsilon^2)$ suffices to achieve a $\frac{1}{2}$ multiplicative approximation guarantee. However, when it is not possible to have such a large K, we need to compromise by increasing d. This proportional trade-off between K and d is also verified in our experiments.

Definition and role of λ. We define λ as $\lambda = \max_{e,i} f_i(\{e\})$. This is not restrictive, as upper bounds on the evaluations of f_is often exist in practice, and λ is typically a constant in most cases. This value imposes an upper bound on the extent to which distinct local functions can differ. By the submodulaity of the f_i's we have $\big|F(e \mid S) - f_i(e \mid S)\big| \leq 2\lambda$, $\forall e, S$. (This is similar to Assumption 3.1 in (Rafiey, 2024)) Moreover, by submodularity and monotonicity we have $f_i(e \mid S) \leq \lambda$ for all e and S. This is important for us to derive a concentration bound over the updates received from the clients at each communication round. The value of $\widehat{F}\big(e \mid S\big) = \frac{|E \setminus S|}{d\,K} \sum_{i \in A^{(t)}, e \in D_i} f_i\big(e \mid S\big)$ is bounded within a range that depends on λ. This, in turn, allows us to apply Hoeffding's inequality to show that for sufficiently large K, the estimates of marginal contributions are, with high probability, within a small additive error of the true value $F\big(e \mid S\big)$.

Theorem 3.1 (Correctness and Approximation Guarantee)**.** *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid of rank r, and $\lambda = \max_{e,i} f_i(\{e\})$. For every $\varepsilon, \delta > 0$ there exist integers K and $d \in [1, m - r]$ such that, w.p. at least $1 - \delta$, Algorithm 1 returns $S \in \mathcal{I}$ which satisfies $F(S) \geq (\frac{1}{2} - \varepsilon)OPT$.*

Moreover, Algorithm 1 has r communication rounds between the server and clients, and it has $\tilde{O}(d)$ bit complexity per client per round.

We point out that the best approximation guarantee for maximizing a monotone submodular function under a matroid constraint is $1 - 1/e$ (in the centralized setting). In Appendix B, we discuss an amplification technique that improves our approximation guarantee to $1 - 1/e$. Moreover, under a cardinality constraint, the approximation guarantee of Algorithm 1 is also $1 - 1/e$.

4. $O(\log m)$ communication rounds

Recall that Algorithm 1 consists of r communication rounds, where only one element is added to S per round. To reduce the number of communication rounds, we consider adding a *block* of elements that provide a large marginal contribution to the current set S. In this section, we propose an algorithm that requires $\tilde{O}(\log m)$ communication rounds between the central server and the clients. Our algorithm, Algorithm 2, extends the approach proposed by Balkanski et al. (2019) for maximizing a monotone submodular function under a matroid constraint in the adaptive setting, which we have specifically adapted for our federated setting.

We note that Algorithm 2 in (Rafiey, 2024) improves the number of communication rounds, but the extent of this improvement is not specified and comes at the cost of an additive error in the approximation. Furthermore their algorithm relies on evaluating multilinear extension and has bit complexity $\tilde{O}(|E|)$. In contrast, our algorithm introduces no additive error and offers a clear trade-off between solution quality, communication rounds, and bit complexity.

Algorithm 2 FedSM with reduced communication rounds

Input: $\mathcal{M} = (E, \mathcal{I})$ of rank r, $\delta, \varepsilon \in (0,1)$, bit complexity $d \geq 1$.
$S \leftarrow \emptyset$, $\quad \tau = \lambda = \max_{e,i} f_i(\{e\})$
for $O\big(\frac{1}{\varepsilon}\log(\frac{r}{\varepsilon})\big)$ iterations **do**
  $X \leftarrow E$
  while $X \neq \emptyset$ **do**
    $r' \leftarrow \mathrm{rank}\big(\mathcal{M}(S, X)\big)$
    $K \leftarrow \frac{|X|^2 \left(r'\right)^4 \lambda^2}{2\,d^2\,\varepsilon^2} \ln\Big(\frac{2\,|X|\,r'\,\ln(m)\,\ln\left(\frac{r}{\varepsilon}\right)}{\delta\,\varepsilon^2}\Big)$
    Server selects a subset of K *active clients* A uniformly at random
    Server generates a random feasible sequence $\{a_l\}_{l=1}^{r'}$.
    for each active client $i \in A$ in parallel **do**
      Client i: sample d *distinct* pairs $D_i \subseteq X \times \{1, \ldots, r'\}$ of size d, chosen uniformly among all $\binom{|X| \cdot r'}{d}$ ways (i.e. *without* replacement).
      For each pair $(e, j) \in D_i$ send back to the server: $\big(f_i(e \mid S \cup \{a_1, \ldots, a_j\}),\ (e, j)\big)$
    end for
    / Aggregate the reports to form estimates: */*
    For each pair $(e, j) \in X \times \{1, \ldots, r'\}$ define

$$\widehat{F}\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big) = \frac{|X|\,r'}{dK} \sum_{i \in A,\, (e,j) \in D_i} f_i\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big).$$

    Let X_j be $\Big\{e \in X \mid S \cup \{a_l\}_{l=1}^{j} \cup \{e\} \in \mathcal{I}, \widehat{F}\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big) \geq \tau\Big\}$
    $j^* \leftarrow \min\{\, j : |X_j| \leq (1 - \varepsilon)\,|X|\}$.
    $S \leftarrow S \cup \{a_1, \ldots, a_{j^*}\}$, $X \leftarrow X_{j^*}$.
  end while
  Server updates: $\tau \leftarrow (1 - \varepsilon)\,\tau$.
end for
Output: S

Full description of the algorithm is given in Algorithm 2. At the start let $X = E$. At a high level, during each communication round, the central server has two objectives: (i) to remove at least an ε-fraction of the elements from X that can potentially be added to the current set S, and (ii) to add as many good elements as possible to S. Ideally, the elements that are removed should have small marginal contributions, while the elements that are added should have large marginal contributions. In order to achieve these two goals, simply estimating the marginal contribution of the remaining elements with respect to the current set S is not sufficient.

More specifically, let X be the current set of available elements to be added to S. Let $\{a_l\}_{l=1}^{r'}$ (r' to be specified later) be *a random feasible sequence* such that $S \cup \{a_j\}_{l=1}^{j}$, for every $j \leq r'$, is a feasible set from the matroid. The goal is to remove at least an ε-fraction of elements from X, and to identify an index j^* to update S to $S \cup \{a_1, \ldots, a_{j^*}\}$.

Each active client samples d distinct pairs (e, j) and performs the following local function evaluation: $f_i(e \mid S \cup \{a_1, \ldots, a_j\})$. These function evaluation results are aggregated and scaled by a factor of $\frac{|X|r'}{dK}$ to ensure unbiasedness. The index j^* is chosen as the smallest j for which $|X_j| \leq (1-\varepsilon)|X|$, where

$$X_j = \left\{ e \in X \mid S \cup \{a_l\}_{l=1}^{j} \cup \{e\} \in \mathcal{I}, \widehat{F}\left(e \mid S \cup \{a_l\}_{l=1}^{j}\right) \geq \tau \right\} \tag{4}$$

where the term $\widehat{F}\left(e \mid S \cup \{a_l\}_{l=1}^{j}\right)$ in Equation (4) is an unbiased estimation of $F\left(e \mid S \cup \{a_l\}_{l=1}^{j}\right)$. The set X is then updated to X_{j^*}. As a result, X decreases by at least an ε-fraction in each round. This process is repeated until X becomes empty.

Theorem 4.1. *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid of rank r, and $\lambda = \max_{e,i} f_i(\{e\})$. For every $\varepsilon, \delta > 0$ there exist integers K and d for each round of Algorithm 2 such that, w.p. at least $1-\delta$, Algorithm 2 returns $S \in \mathcal{I}$ which satisfies $\mathbb{E}[F(S)] \geq (\frac{1}{2} - O(\varepsilon))OPT$.*

Moreover, Algorithm 2 has $O(\log(m)\log(r/\varepsilon)\varepsilon^{-2})$ communication rounds between the server and clients, and it has $\tilde{O}(d)$ bit complexity per client per round.

5. Experimental results

In this section, we experimentally evaluate our algorithms on real-world datasets by optimizing two fundamental submodular problems: facility location and maximum coverage. Due to space constraints, the results for the maximum coverage problem are presented in the appendix.

Facility Location. We study a movie recommendation task (Stan et al., 2017), where each client i has a private utility function f_i over sets of movies. The goal is to select a set of r movies that best satisfy all clients. We use the MovieLens dataset with $n = 6041$ clients and $m = 4000$ movies (1 million ratings). As users prefer to keep their ratings private, we assume ratings are not shared. Let $r_{i,j}$ be the rating of client i for movie j, set to 0 if missing. Each client uses a facility location objective: $f_i(S) = \max_{j \in S} r(i,j)$ for $S \subseteq E$. Ratings are integers in $[0, 5]$, so $\lambda = 5$. The server aims to estimate $\max_{S \subseteq E, |S| \leq r} \frac{1}{n} \sum f_i(S)$.

In Figure 1, we evaluate Algorithm 1 under two settings: varying K with fixed d, and varying d with fixed K. Here, K denotes the percentage of sampled clients per round, and d the percentage of elements each client samples. We compare against the centralized greedy algorithm (Nemhauser et al., 1978), averaging results over 3 random seeds. Performance improves with larger d when $K = 1.0\%$, and with larger K when $d = 10\%$, aligning with our theoretical findings.

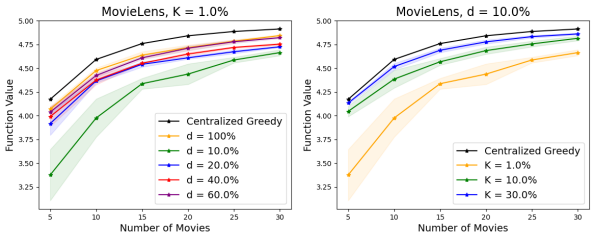

Figure 1: Bit complexity and client participation trade-off on MovieLens. Left: varying d at fixed $K = \lfloor n/100 \rfloor$. Right: varying K at fixed $d = \lfloor m/10 \rfloor$.

Notably, in the MovieLens dataset, the number of clients n is comparable to the number of movies m. In such cases, choosing a sufficiently large d is crucial to compensate for the limited number of clients. As seen in Appendix, when $d = 1$, even with full participation ($K = 100\%$), there is a significant performance gap between the centralized greedy algorithm and our approach, underscoring the need for a larger d.

Maximum Coverage. Let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be a set of clients and $E = \{G_1, \ldots, G_m\}$ be a family of sets where each $G_i \subseteq \mathcal{C}$ is a group of clients. Given a positive integer r, in the `Max Coverage` problem the objective is to select at most r groups of clients from E such that the maximum number of clients are covered i.e., the union of the selected groups has maximal size. One can formulate this problem as follows. For every $i \in [n]$ and $A \subseteq [m]$ define

$$f_i(A) = \begin{cases} 1 & \text{if there exists } a \in A \text{ such that } c_i \in G_a, \\ 0 & \text{otherwise.} \end{cases}$$

which f_i's are monotone and submodular. Furthermore, define $F(A) = \sum_{i \in [n]} f_i(A)$, which is also monotone and submodular, and $\lambda = 1$. All in all, the formulation of the `Max Coverage` problem is $\max_{A \subseteq [n], |A| \leq k} \sum_{i \in [N]} f_i(A)$.

We use a public dataset for `Max Coverage`. The DBLP dataset (`https://dblp.org/xml/release/`) is a bipartite network comprised of $n = 704738$ researchers and $m = 2675$ venues. Each researcher's membership set contains all venues where the researcher published at least one paper between 2018 and 2021. The objective of `Max Coverage` is to select a set of venues covering the most researchers. Following the same setup as for the MovieLens dataset, we compare the performance of Algorithm 1 under different choices of K and d against the centralized greedy algorithm (Nemhauser et al., 1978). The reported results, Figure 2, are averaged over 3 runs with different random seeds.

Analogous conclusions were drawn i.e., increasing K or d improved the overall performance. A key difference be-

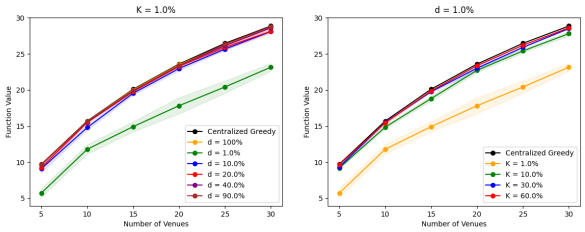

Figure 2: Bit complexity and client participation trade-off on DBLP. Left: varying d at fixed $K = \lfloor n/100 \rfloor$. Right: varying K at fixed $d = \lfloor m/100 \rfloor$.

tween the DBLP and MovieLens datasets is that the number of clients in DBLP ($n = 704,738$ researchers) is significantly larger than the ground set size ($m = 2,675$). As a result, the performance gap between the centralized greedy algorithm and our approach is much smaller than in the MovieLens case. This is evident in Figure 6 (Appendix), where even for $d = 1$ and varying K, our algorithm closely approaches the performance of the centralized greedy algorithm.

Algorithm 2 and the Number of Communication Rounds. We evaluate the performance of Algorithm 2 against the centralized greedy algorithm on the MovieLens dataset, which Algorithm 2 is applicable when $O(\log m) \leq r$. In these experiments, we set $r = 200$ i.e., the goal is to select a subset of 200 movies that maximizes the objective. In Figure 3, we plot the number of communication rounds against the function value.

This experiment requires larger K and d than Algorithm 1, due to a larger sampling space—consistent with the theoretical bound in Lemma A.4. Initially, S grows nearly exponentially, yielding faster convergence than centralized greedy. As shown in Figure 3, growth slows to linear beyond a saturation point, reflecting diminishing return property and the effect of threshold τ. The interaction between K and d also strongly influences convergence speed and the growth of S.

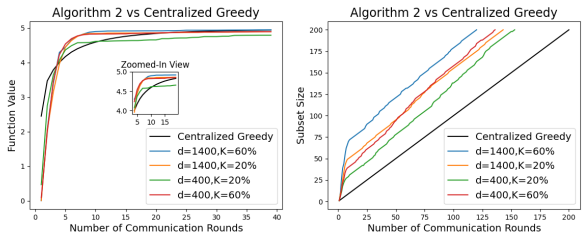

Figure 3: Left: Algorithm 2 vs. Centralized Greedy (up to 40 rounds). Right: growth of S with respect to K and d.

6. Conclusion and discussion

In this work, we studied the problem of maximizing a monotone submodular function under a matroid constraint in a federated setting. We proposes an efficient and easy-to-implement algorithm with low bit complexity and partial client participation. Our theoretical analysis captures the trade-off between client sampling and bit complexity, making our approach flexible and applicable to various practical scenarios. Furthermore, we improved the communication round complexity by introducing an algorithm that requires only a logarithmic number of communication rounds.

Differential Privacy. In the Appendix we discussed how a differentially private mechanism can be integrated into our approach. By leveraging the algorithms from (Cheu et al., 2019) (specifically, Algorithms 5 and 6), we can ensure privacy in each communication round of our algorithms at the cost of an additional $O(\sqrt{K})$ factor in the bit complexity. Specifically, there exists a (ε', δ') differentially private algorithm, $\texttt{DPAlg}$, with $\varepsilon' = O\left(\frac{r}{\varepsilon K} \log \frac{1}{\delta'} \sqrt{\log \frac{m\,r}{\delta}}\right)$ and $e^{-\Omega(K^{1/4})} < \delta' < \frac{1}{K}$, which requires every participating client to encode each real-valued number using $O(\varepsilon \sqrt{K})$ bits, and for any $e \in E \setminus S$ it guarantees $\Pr\left[\left|\text{Output of } \texttt{DPAlg} - \widehat{F}(e \mid S)\right| > \varepsilon/r\right] \leq \delta/(m\,r)$.

Amplification. We point out that the best approximation guarantee for maximizing a monotone submodular function under a matroid constraint is $1 - 1/e$. In Appendix, we discuss an amplification technique that improves our $1/2$-approximation guarantee to $1 - 1/e$. Specifically, using Algorithm 2 as a subroutine in our amplification algorithm, we obtain $1 - 1/e - O(\varepsilon)$-approximation using $O(\log(m) \log(\frac{r}{\varepsilon\,\eta}) \frac{1}{\varepsilon\,\eta})$ communication rounds.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abbasi, F., Adamczyk, M., Bosch-Calvo, M., Byrka, J., Grandoni, F., Sornat, K., and Tinguely, A. An O(loglog n)-Approximation for Submodular Facility Location. In *ICALP*, 2024.

Badanidiyuru, A. and Vondrák, J. Fast algorithms for maximizing submodular functions. In *SODA*, 2014.

Balkanski, E. and Singer, Y. The adaptive complexity of maximizing a submodular function. In *STOC*, 2018.

Balkanski, E., Rubinstein, A., and Singer, Y. An optimal

approximation for submodular maximization under a matroid constraint in the adaptive complexity model. In *STOC*, 2019.

Barbosa, R., Ene, A., Nguyen, H., and Ward, J. The Power of Randomization: Distributed Submodular Maximization on Massive Datasets. In *ICML*, 2015.

Basu, D., Data, D., Karakus, C., and Diggavi, S. Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations. *NeurIPS*, 2019.

Bateni, M., Esfandiari, H., and Mirrokni, V. Optimal distributed submodular optimization via sketching. In *KDD*, 2018.

Beimel, A., Nissim, K., and Omri, E. Distributed private data analysis: Simultaneously solving how and what. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference*, 2008.

Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., and Raykova, M. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical Secure Aggregation for Federated Learning on User-Held Data. *arXiv preprint arXiv:1611.04482*, 2016.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of cryptography conference*, 2016.

Călinescu, G., Chekuri, C., Pál, M., and Vondrák, J. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.*, 40(6):1740–1766, 2011.

Chan, T. H., Shi, E., and Song, D. Optimal lower bound for differentially private multi-party aggregation. In *ESA*, 2012.

Chaturvedi, A., Nguyen, H. L., and Zakynthinou, L. Differentially private decomposable submodular maximization. In *AAAI*, 2021.

Chekuri, C. and Quanrud, K. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *STOC*, 2019.

Chekuri, C., Vondrak, J., and Zenklusen, R. Dependent randomized rounding via exchange properties of combinatorial structures. In *FOCS*, 2010.

Chen, W., Horváth, S., and Richtárik, P. Optimal client sampling for federated learning. *Transactions on Machine Learning Research*, 2022.

Chen, Y. and Kuhnle, A. Approximation algorithms for size-constrained non-monotone submodular maximization in deterministic linear time. In *KDD*, 2023.

Cheu, A., Smith, A., Ullman, J., Zeber, D., and Zhilyaev, M. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019*, 2019.

Dwork, C. and Lei, J. Differential privacy and robust statistics. In *STOC*, 2009.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2006.

Dwork, C., Rothblum, G. N., and Vadhan, S. Boosting and differential privacy. In *FOCS*, 2010.

Elhamifar, E. Sequential Facility Location: Approximate Submodularity and Greedy Algorithm. In *ICML*, 2019.

Gupta, A., Ligett, K., McSherry, F., Roth, A., and Talwar, K. Differentially private combinatorial optimization. In *SODA*, 2010.

Haddadpour, F., Kamani, M. M., Mokhtari, A., and Mahdavi, M. Federated learning with compression: Unified analysis and sharp guarantees. In *AISTAT*, 2021.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.

Karimi, M., Lucic, M., Hassani, H., and Krause, A. Stochastic submodular maximization: The case of coverage functions. In *NeurIPS*, 2017.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Kumar, R., Moseley, B., Vassilvitskii, S., and Vattani, A. Fast greedy algorithms in mapreduce and streaming. *ACM Trans. Parallel Comput.*, 2(3):14:1–14:22, 2015.

Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., and Ramchandran, K. Speeding Up Distributed Machine Learning Using Codes. *IEEE Transactions on Information Theory*, 64(3):1514–1529, 2018.

Li, S., Avestimehr, S., et al. Coded computing: Mitigating fundamental bottlenecks in large-scale distributed computing and machine learning. *Foundations and Trends® in Communications and Information Theory*, 17(1):1–148, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTAT*, 2017.

Mehrotra, A. and Vishnoi, N. K. Maximizing Submodular Functions for Recommendation in the Presence of Biases. In *WWW*, 2023.

Mirzasoleiman, B., Karbasi, A., Sarkar, R., and Krause, A. Distributed Submodular Maximization. *J. Mach. Learn. Res.*, 17:238:1–238:44, 2016.

Mitrovic, M., Bun, M., Krause, A., and Karbasi, A. Differentially private submodular maximization: Data summarization in disguise. In *ICML*, 2017.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978.

Nikolakaki, S. M., Ene, A., and Terzi, E. An efficient framework for balancing submodularity and cost. In *KDD*, 2021.

Qin, Z., Deng, S., Zhao, M., and Yan, X. Fedapen: Personalized cross-silo federated learning with adaptability to statistical heterogeneity. In *KDD*, 2023.

Rafiey, A. Decomposable submodular maximization in federated setting. In *ICML*, 2024.

Rafiey, A. and Yoshida, Y. Fast and private submodular and k-submodular functions maximization with matroid constraints. In *ICML*, 2020.

Rafiey, A. and Yoshida, Y. Sparsification of decomposable submodular functions. In *AAAI*, 2022.

Sadeghi, O. and Fazel, M. Differentially private monotone submodular maximization under matroid and knapsack constraints. In *AISTAT*, 2021.

Stan, S., Zadimoghaddam, M., Krause, A., and Karbasi, A. Probabilistic Submodular Maximization in Sub-Linear Time. In *ICML*, 2017.

Vondrak, J. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, 2008.

Wang, Y., Zhou, T., Chen, C., and Wang, Y. Federated submodular maximization with differential privacy. *IEEE Internet of Things Journal*, 2023.

A. Missing proofs

We use the following concentration bound in our proofs.

Lemma A.1 (Hoeffding's inequality (Hoeffding, 1994)). *Let $\mathcal{X} = (x_1, \ldots, x_n)$ be a finite population of n points and $X_1, \ldots, X_K$ be a random sample drawn without replacement from $\mathcal{X}$. Let*

$$a = \min_{1 \le i \le n} x_i \quad \text{and} \quad b = \max_{1 \le i \le n} x_i \tag{5}$$

Then, for all $\varepsilon > 0$,

$$\Pr\Big[\Big|\frac{1}{K}\sum_{i=1}^{K} X_i - \mu\Big| \ge \varepsilon\Big] \le \exp\big(-\tfrac{2K\varepsilon^2}{(b-a)^2}\big) \tag{6}$$

where $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the mean of $\mathcal{X}$.

A.1. Missing proofs from Section 3

Proof of Theorem 3.1. At each iteration, the server adds an element that does not violate the matroid constraint, ensuring that the feasibility of the updated set S is preserved. We will focus on the approximation guarantee. Let $S_0 \subseteq S_1 \subseteq \cdots \subseteq S_{r-1}$ be the sets chosen by Algorithm 1 at each iteration t. Now, we show three key facts regarding the algorithm.

1. Unbiasedness of each estimate. Fix a particular element $e \in E \setminus S_t$, for which we want to show unbiasedness: $\mathbb{E}\left[\widehat{F}\left(e \mid S_t\right)\right] = F\big(e \mid S_t\big)$.

The server picks exactly K out of n clients uniformly. Hence, $\Pr\left[i \in A^{(t)}\right] = \frac{K}{n}$. This follows from the fact that among the $\binom{n}{K}$ equally likely K-subsets, the fraction containing a fixed i is $\frac{K}{n}$. Conditioned on $i \in A^{(t)}$, client i picks a d-element subset $D_i \subset E \setminus S_t$ uniformly at random from all $\binom{|E\setminus S_t|}{d}$ possible d-element subsets. Therefore, $\Pr\left[e \in D_i\right] = \frac{d}{|E\setminus S_t|}$, since among all possible ways to form a d-element subset, those containing e comprise a fraction $\frac{d}{|E\setminus S_t|}$. Thus, for each $i \in \{1, \ldots, n\}$,

$$\begin{aligned}
&\Pr\Big[i \in A^{(t)} \text{ and } e \in D_i\Big] \\
&= \Pr[i \in A^{(t)}] \cdot \Pr\big[e \in D_i \mid i \in A^{(t)}\big] \\
&= \tfrac{K}{n} \cdot \tfrac{d}{|E\setminus S_t|}.
\end{aligned}$$

If $e \in D_i$, then client i contribution to the sum defining $\widehat{F}(e \mid S_t)$, amounts to $\frac{|E\setminus S_t|}{d\,K} \cdot f_i\big(e \mid S_t\big)$. Otherwise, if $e \notin D_i$ there is no corresponding contribution from client i. Therefore, the expected contribution from client i is

$$\tfrac{|E\setminus S_t|}{d\,K} \cdot f_i\big(e \mid S_t\big) \cdot \Pr\big[e \in D_i \text{ and } i \in A^{(t)}\big].$$

Summing over all clients, by linearity of expectation

$$\begin{aligned}
\mathbb{E}\big[\widehat{F}(e \mid S_t)\big] &= \sum_{i=1}^{n} \tfrac{|E\setminus S_t|}{d\,K} \cdot \tfrac{K}{n} \cdot \tfrac{d}{|E\setminus S_t|} \cdot f_i(e \mid S_t) \\
&= \tfrac{1}{n}\sum_{i=1}^{n} f_i(e \mid S_t),
\end{aligned}$$

which is precisely $F\big(e \mid S_t\big)$. We therefore conclude that $\widehat{F}(e \mid S_t)$ is an unbiased estimator of $F(e \mid S_t)$.

2. Concentration via Hoeffding. Note that $\max_{e,i} f_i(\{e\}) = \lambda$, hence by submodularity and monotonicity we have $f_i(e \mid S_t) \le \lambda$ for all e, S_t, thus $f_i(e \mid S_t) \in [0, \lambda]$. After scaling $f_i(e \mid S_t)$ by $\frac{|E\setminus S_t|}{d\,K}$, since $\frac{|E\setminus S_t|}{d\,K} \le \frac{m}{d\,K}$ its magnitude is at most $\frac{m\,\lambda}{d\,K}$. Summing over K clients then gives us $\widehat{F}\big(e \mid S_t\big) \le \frac{m}{d}\,\lambda$ in the worst case, therefore $0 \le f_i(e \mid S_t) \le \frac{m}{d}\,\lambda$ for each i. For our purposes, we can view $f_i(e \mid S_t)$ as a random variable with range $[0, \frac{m}{d}\lambda]$.

Given this range, we can now apply Hoeffding's inequality. Let

$$K \;\ge\; \frac{\big(m/d\big)^2\, r^2\, \lambda^2\, \ln\big(2\,r\,m/\delta\big)}{8\,\varepsilon^2},$$

and by Lemma A.1, for a fixed e and S_t we have

$$\begin{aligned}
&\Pr\Big[\Big|\widehat{F}(e \mid S_t) - F(e \mid S_t)\Big| \ge 2\varepsilon/r\Big] \\
&\qquad \le \exp\Big(-\tfrac{2K^2\,(2\varepsilon/r)^2}{(m\lambda/d)^2}\Big) \le \tfrac{\delta}{m\,r}.
\end{aligned}$$

Using a union bound over all e and iterations $t \in \{0, \ldots, r-1\}$, w.p. at least $1-\delta$; *all* the estimates $\widehat{F}(e \mid S_t)$ are within $\frac{2\varepsilon}{r}$ of $F(e \mid S_t)$ simultaneously, that is for all e and S_t:

$$\Pr\Big[\big|\widehat{F}(e \mid S_t) - F(e \mid S_t)\big| \;\le\; 2\varepsilon/r\Big] \ge 1 - \delta.$$

We henceforth assume this good event occurs.

3. Matroid exchange & submodularity. Let $O = \{o_1, \ldots, o_r\}$ be an optimal solution, i.e. a basis of $\mathcal{M}$ with $F(O) = \texttt{OPT}$. Let $S = \{e_1, \ldots, e_r\}$ be the set returned by Algorithm 1. By matroid properties, for each t we know $S_{t-1} \cup \{o_t\} \in \mathcal{I}$, thus

$$F(O) - F(S) \;\le\; \sum_{o \in O} F\big(o \mid S\big) \;\le\; \sum_{t=1}^{r} F\big(o_t \mid S_{t-1}\big).$$

But Algorithm 1 picks e_t as an approximate maximizer of $F(e \mid S_{t-1})$, meaning

$$F\big(o_t \mid S_{t-1}\big) \;\le\; F\big(e_t \mid S_{t-1}\big) + \tfrac{2\varepsilon}{r},$$

because $\widehat{F}(o_t \mid S_{t-1})$ and $\widehat{F}(e_t \mid S_{t-1})$ differ (with high probability) by at most $\frac{2\varepsilon}{r}$. Summing over $t = 1, \ldots, r$ yields

$$F(O) - F(S) \;\le\; \sum_{t=1}^{r} \Big(F\big(e_t \mid S_{t-1}\big) + \frac{2\varepsilon}{r} \Big) \;=\; F(S) + 2\varepsilon,$$

which implies $F(S) \;\ge\; \frac{1}{2}\,\mathtt{OPT} - \varepsilon$. This holds under the event that all the estimates $\widehat{F}(e \mid S_t)$ are within $\frac{2\varepsilon}{r}$ of $F(e \mid S_t)$ simultaneously, which happens w.p. $1 - \delta$.

Choosing ε small enough [1] gives the multiplicative bound $F(S) \;\ge\; (\frac{1}{2} - O(\varepsilon))\,\mathtt{OPT}$.

□

A.2. Missing proofs from Section 4

We start with analyzing the number of communication rounds in Algorithm 2.

Lemma A.2. *Algorithm 2 has $O(\log(m)\log(r/\varepsilon)\varepsilon^{-2})$ communication rounds between the server and clients.*

Proof of Lemma A.2. The outer for loop has $O(\log(r/\varepsilon)\varepsilon^{-1})$ iterations. The `while` loop runs for at most $O(\log(m)\varepsilon^{-1})$ iterations, , as, according to the definition of j^*, at least an ε-fraction of the remaining elements in X is eliminated in each iteration. We can find j^* by computing X_j for each j in one communication round. □

Next we will establish the approximation quality of Algorithm 2. Some notations are in order. For a random feasible sequence $\{a_l\}_{l=1}^{r'}$, r' is equal to $\mathtt{rank}(\mathcal{M}(S, X))$. Here $\mathcal{M}(S, X) \coloneqq \{T \subseteq X \colon S \cup T \in \mathcal{M}\}$ represents the matroid over elements X, where a subset is feasible in $\mathcal{M}(S, X)$ if its union with the current S remains feasible according to $\mathcal{M}$.

We first show the aggregated updates at the central server ar in fact unbiased estimation. The term $\widehat{F}\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big)$ in (4) is an estimation of $F\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big)$ and is computed from the aggregation of the clients' feedback. As in Algorithm 1, the estimate must be unbiased with a small additive error. In Lemma A.3, we show that the resulting estimate of Algorithm 2 is indeed unbiased, and that for a sufficiently large K we are close to the actual value with high confidence.

Lemma A.3 (Unbiasedness and Concentration)**.** *Let $S \subseteq E$ be fixed, and let $r' = \mathrm{rank}\big(\mathcal{M}(S, X)\big)$. Assume each $f_i(\{e\}) \le \lambda$ for all e, i. Suppose the server picks K active clients indexed by $A \subseteq [n]$ uniformly at random, where each subset of size K is equally likely. Conditioned on this event, each active client $i \in A$ draws a subset $D_i \subseteq X \times \{1, \ldots, r'\}$ of size d* without replacement *(each d-subset is equally likely). Then, for each pair (e, j) where $S \cup \{a_1, \ldots, a_j, e\} \in \mathcal{I}$, define*

$$\widehat{F}\big(e \;\mid\; S \cup \{a_l\}_{l=1}^{j}\big) = \tfrac{|X|\,r'}{d\,K} \sum_{i \in A,\,(e,j) \in D_i} f_i\big(e \;\mid\; S \cup \{a_l\}_{l=1}^{j}\big).$$

We then have the unbiasedness and concentration, defined below.

1. *Unbiasedness. For every (e, j), we have $\mathbb{E}\Big[\widehat{F}\big(e \;\mid\; S \cup \{a_l\}_{l=1}^{j}\big)\Big] \;=\; F\big(e \;\mid\; S \cup \{a_l\}_{l=1}^{j}\big)$*

   where the expectation is over all the random choices, i.e. which K clients and subsequently d-subsets are chosen.

2. *Concentration. If $K \ge \frac{\left(|X|\,r'\right)^2 \lambda^2}{2\,d^2\,\varepsilon^2} \ln\Big(\frac{2\,|X|\,r'}{\delta}\Big)$, then for* all *pairs (e, j)*

   $$\Pr\left[\left|\widehat{F}\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big) - F\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big)\right| \le \frac{\varepsilon}{r'}\right] \ge 1 - \delta.$$

Proof of Lemma A.3. We first show unbiasedness. Fix a particular pair (e, j). The server picks exactly K out of n clients uniformly at random i.e., $\Pr[i \in A] = \frac{K}{n}$, which follows from the fact discussed in the proof of Theorem 3.1. Conditioned on $i \in A$, client i picks a d-element subset $D_i \subseteq X \times \{1, \ldots, r'\}$ uniformly at random, from all $\binom{|X| \cdot r'}{d}$ possible subsets of size d. Hence, $\Pr\left[(e, j) \in D_i\right] = \frac{d}{|X| \cdot r'}$, since among all ways to form a d-element subset, exactly those that contain (e, j) form a fraction of $\frac{d}{|X| \cdot r'}$. Thus, for each $i \in \{1, \ldots, n\}$

$$\begin{aligned} &\Pr\Big[i \in A \text{ and } (e, j) \in D_i\Big] \\ &= \Pr[i \in A] \cdot \Pr\big[(e, j) \in D_i \;\mid\; i \in A\big] \\ &= \frac{K}{n} \;\cdot\; \frac{d}{|X|\,r'}. \end{aligned}$$

If $(e, j) \in D_i$, then the contribution of the i-th client to $\widehat{F}\big(e \;\mid\; S \cup \{a_1, \ldots, a_j\}\big)$ is precisely

$$\frac{|X|\,r'}{d\,K}\, f_i\big(e \;\mid\; S \cup \{a_1, \ldots, a_j\}\big).$$

Otherwise, it contributes 0. Hence the expected contribution from client i is

$$\frac{|X|\,r'}{d\,K} \times \Pr\big[(e, j) \in D_i \text{ and } i \in A\big] \times f_i\big(e \;\mid\; S \cup \{a_1, \ldots, a_j\}\big).$$

[1] Note that by submodularity and monotonicity we have $\lambda \le \mathtt{OPT} \le \lambda r$, thus the choice of $\varepsilon = \varepsilon' \lambda r$ implies $\varepsilon \le \varepsilon'\,\mathtt{OPT}$ for any ε'.

Considering all n clients, by linearity of expectation we deduce that

$$\begin{aligned}&\mathbb{E}\big[\widehat{F}(e \mid S \cup \{a_l\}_{l=1}^{j})\big] \\ &= \sum_{i=1}^{n} \frac{|X|\, r'}{d\, K} \cdot \frac{K}{n} \cdot \frac{d}{|X|\, r'} \cdot f_i(e \mid S\{a_l\}_{l=1}^{j}) \\ &= \frac{1}{n} \sum_{i=1}^{n} f_i(e \mid S \cup \{a_l\}_{l=1}^{j}) = F\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big).\end{aligned}$$

Therefore, $\widehat{F}$ is an unbiased estimator of F.

Proof of concentration. Since $\max_{e,i} f_i(\{e\}) \leq \lambda$, by monotonicity and submodularity, we have $f_i(e \mid S) \leq \lambda$. Each active client i that has $(e, j) \in D_i$ contributes at most $\frac{|X|\, r'}{d\, K}\lambda$ to $\widehat{F}\big(e \mid S \cup \{a_l\}_{l=1}^{j}\big)$. We can view each of the K active client's contribution as a bounded random variable in $[0,\ \frac{|X|\, r'}{d}\lambda]$. By then invoking Hoeffding's inequality, for $z = 2(\frac{\varepsilon}{r'})^2\, K \big/ \big(\frac{|X|\, r'}{d}\lambda\big)^2$ we get

$$\begin{aligned}&\Pr\Big[\big|\widehat{F}(e \mid S \cup \{a_l\}_{l=1}^{j}) - \mathbb{E}[\widehat{F}(e \mid S \cup \{a_l\}_{l=1}^{j})]\big| \ \geq \tfrac{\varepsilon}{r'}\Big] \\ &\leq 2e^{-z}.\end{aligned}$$

To ensure that this holds with probability is at most $\frac{\delta}{|X|\, r'}$, we require $\frac{2\,\varepsilon^2\, K\, d^2}{|X|^2\, r'^4\, \lambda^2} \geq \ln\Big(\frac{2\,|X|\, r'}{\delta}\Big)$, i.e. $K \geq \frac{|X|^2\, r'^4\, \lambda^2}{2\, d^2\, \varepsilon^2} \ln\Big(\frac{2\,|X|\, r'}{\delta}\Big)$.

Finally, we apply a union bound over all $(e, j) \in X \times \{1, \ldots, r'\}$, which amount for a total of $|X|\, r'$ pairs. This yields an overall failure probability of at most δ, thus we have

$$\begin{aligned}&\Pr\Big[\big|\widehat{F}(e \mid S \cup \{a_l\}_{l=1}^{j}) - F(e \mid S \cup \{a_l\}_{l=1}^{j})\big| \leq \frac{\varepsilon}{r'}\Big] \\ &\geq 1 - \delta\end{aligned}$$

for *all* (e, j) simultaneously. □

The following lemma, shows that at each iteration the threshold τ in Algorithm 2 is not very restrictive, and it is unlikely that elements with high marginal contribution will be removed.

Lemma A.4. *For every $\varepsilon, \delta > 0$, at each iteration let*

$$K = O\left(\frac{m^2 \cdot r^4 \cdot \lambda^2}{\varepsilon^2} \log\left(\frac{m \log(m) \cdot r \log(r/\varepsilon)}{\delta \varepsilon^2}\right)\right).$$

Then $\Pr\big[\tau \geq (1-\varepsilon) \max_{e:S\cup\{e\}\in\mathcal{M}} F(e \mid S) - \varepsilon/r\big] \geq 1 - \delta$.

Proof of Lemma A.4. The lemma is proven by induction. First note that

$$\max_{e \in E} F(e) = \max_{e \in E} \sum_{i=1}^{n} \frac{1}{n} f_i(e) \leq \sum_{i} \frac{1}{n} \lambda = \lambda. \quad (7)$$

Thus (7) holds at the first iteration by the initial instantiations $\tau = \lambda = \max_{e,i} f_i(\{e\})$, $S = \emptyset$ and $X = E$. We show that the lemma holds throughout the execution of the algorithm when either S or τ are updated.

First, suppose at some iteration of the algorithm we have $\tau \geq (1-\varepsilon) \max_{e:S\cup\{e\}\in\mathcal{M}} F(e \mid S) - \varepsilon/r$, and that S is updated to $S \cup \{a_1, \ldots, a_{j^*}\}$. Then, for all e for which $S \cup \{e\} \in \mathcal{M}$ we have

$$\begin{aligned}&(1-\varepsilon) F(e \mid S \cup \{a_1, \ldots, a_{j^*}\}) - \varepsilon/r \\ &\qquad \leq (1-\varepsilon) F(e \mid S) - \varepsilon/r \leq \tau\end{aligned}$$

where the first inequality is by submodularity, and the second is by the inductive hypothesis. Since $\{e : S \cup \{a_1, \ldots, a_{j^*}, e\} \in \mathcal{M}\} \subseteq \{e : S \cup \{e\} \in \mathcal{M}\}$ by the downward closed property of $\mathcal{M}$, we have

$$\begin{aligned}&\max_{e:S\cup\{a_1,\ldots,a_{j^*},e\}\in\mathcal{M}} F(e \mid S \cup \{a_l\}_{l=1}^{j^*}) \\ &\qquad \leq \max_{e:S\cup\{e\}\in\mathcal{M}} F(e \mid S \cup \{a_l\}_{l=1}^{j^*}).\end{aligned}$$

Thus, when S is updated to $S \cup \{a_l\}_{l=1}^{j^*}$, we have

$$\tau \geq (1-\varepsilon) \max_{e:S\cup\{a_1,\ldots,a_{j^*},e\}\in\mathcal{M}} F(e \mid S) - \varepsilon/r.$$

Next, consider an iteration where τ is updated to $\tau' = (1-\varepsilon)\tau$. At that iteration, we have $X = \emptyset$ and current solution S. Therefore, by the algorithm, each $e \in E$ were discarded from X at a previous iteration with corresponding (iteration) solution S' for which $S' \cup \{a_1, \ldots, a_{j^*}\} \subseteq S$. Since e was discarded, it is either the case that $S' \cup \{a_1, \ldots, a_{j^*}\} \cup \{e\} \notin \mathcal{M}$ or $\widehat{F}\big(e \mid S' \cup \{a_1, \ldots, a_{j^*}\}\big) < \tau$. If $S' \cup \{a_1, \ldots, a_{j^*}\} \cup \{e\} \notin \mathcal{M}$, by the downward closed property of $\mathcal{M}$ and the fact that $S' \cup \{a_1, \ldots, a_{j^*}\} \subseteq S$, we deduce that $S \cup \{e\} \notin \mathcal{M}$. Otherwise, $\tau > \widehat{F}\big(e \mid S' \cup \{a_1, \ldots, a_{j^*}\}\big)$, which yields

$$\begin{aligned}\tau' = (1-\varepsilon)\tau &> (1-\varepsilon)\widehat{F}\big(e \mid S' \cup \{a_1, \ldots, a_{j^*}\}\big) \\ &\geq (1-\varepsilon)[F(e \mid S' \cup \{a_1, \ldots, a_{j^*}\}) - \varepsilon/r] \quad (8) \\ &\geq (1-\varepsilon) F(e \mid S' \cup \{a_1, \ldots, a_{j^*}\}) - \varepsilon/r \\ &\geq (1-\varepsilon) F(e \mid S) - \varepsilon/r.\end{aligned}$$

where the last inequality follows by submodularity and the fact that $S' \cup \{a_l\}_{l=1}^{j^*} \subseteq S$. Note that (8) holds with some probability. Specifically, by Lemma A.3 and applying the union bound over $O(\log(m)\log(r/\varepsilon)\varepsilon^{-2})$ iterations (Lemma A.2), for

$$\begin{aligned}K &= \frac{|X|^2 \cdot r^4 \cdot \lambda^2}{2\varepsilon^2} \log\Big(\frac{2|X| \cdot \mathtt{rank}(\mathcal{M}(S,X)) \log(m) \log(r/\varepsilon)}{\delta \varepsilon^2}\Big) \\ &= O\left(\frac{m^2 \cdot r^4 \cdot \lambda^2}{\varepsilon^2} \log\left(\frac{m \log(m) \cdot r \log(r/\varepsilon)}{\delta \varepsilon^2}\right)\right)\end{aligned}$$

with probability at least $1-\delta$ (8) holds for all cases. Thus, with probability at least $1-\delta$, for all e such that $S\cup\{e\}\in\mathcal{M}$, we have $\tau' \geq (1-\varepsilon)F(e \mid S) - \varepsilon/r$. This finishes the proof. □

Generating a random feasible sequence $a_1, a_2, \ldots, a_{r'}$ does not require function evaluations and communication with the clients. We employ the same procedure as (Balkanski et al., 2019). Given a matroid $\mathcal{M}$ we say that $(a_1, \ldots, a_{\texttt{rank}(\mathcal{M})})$ is a *random feasible sequence* if for all $i \in \{1, 2, \ldots, \texttt{rank}(\mathcal{M})\}$, a_i is an element chosen uniformly at random from the set of feasible elements $\{a : \{a_1, \ldots, a_{i-1}, a\} \in \mathcal{M}\}$. One simple way to generate a random feasible sequence is by sampling feasible elements sequentially. The following lemma shows that even though a_i's are sampled uniformly at random, in expectation, the resulting marginal contribution is close to optimal.

Lemma A.5. *Assume $a_1, \ldots, a_{r'}$, for $r' = \texttt{rank}(\mathcal{M}(S, X))$, is a random feasible sequence. Then, for all $j \leq j^*$*

$$\mathbb{E}_{a_j}\left[F(a_j \mid S\cup\{a_l\}_{l=1}^{j-1})\right] \geq (1-\varepsilon)^2 \max_{e:S\cup\{a_l\}_{l=1}^{j-1}\cup\{e\}\in\mathcal{M}} F(e \mid S) - \varepsilon/r.$$

Proof of Lemma A.5. First note that by unbiasedness in Lemma A.3 we have $\mathbb{E}\Big[\widehat{F}\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)\Big] = F\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)$ where the expectation is over all the random choices, i.e. which K clients and subsequently d-subsets are chosen. Hence,

$$\begin{aligned}\mathbb{E}_{a_j}[F(a_j \mid S\cup\{a_l\}_{l=1}^{j-1})] &= \mathbb{E}_{a_j}\left[\mathbb{E}\Big[\widehat{F}\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)\Big]\right]\\ &= \mathbb{E}\,\mathbb{E}_{a_j}\left[\widehat{F}\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)\right]\end{aligned}$$

Note that a_j is sampled uniformly at random from $X_{j-1}^{\mathcal{M}} = \{e \in X : S\cup\{a_l\}_{l=1}^{j-1}\cup\{e\}\in\mathcal{M}\}$. Then

$$\begin{aligned}&\Pr_{a_j}\left[\widehat{F}\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)\geq\tau\right]\cdot\tau\\ &= \frac{|X_{j-1}^{\mathcal{M}}|}{|X_{j-1}|}\cdot\tau\\ &\geq \frac{|X_{j-1}^{\mathcal{M}}|}{|X|}\cdot\tau\\ &\geq (1-\varepsilon)^2\max_{e:\in X_{j-1}^{\mathcal{M}}} F(e \mid S_{j-1}) - \varepsilon/r\end{aligned}$$

where the equality follows from the definition of X_{j-1}. The first inequality holds since $X_{j-1}^{\mathcal{M}} \subseteq X$, and the second by Lemma A.4 and the fact that $j \leq j^*$. Finally, by Markov's inequality

$$\begin{aligned}&\mathbb{E}_{a_j}\left[\widehat{F}\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)\right]\\ &\geq \Pr_{a_j}\left[\widehat{F}\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)\geq\tau\right]\cdot\tau\\ &\geq (1-\varepsilon)^2\max_{e:\in X_{j-1}^{\mathcal{M}}} F(e \mid S_{j-1}) - \varepsilon/r \qquad (9)\end{aligned}$$

Therefore, by (9)

$$\begin{aligned}&\mathbb{E}_{a_j}\left[F(a_j \mid S\cup\{a_l\}_{l=1}^{j-1})\right]\\ &= \mathbb{E}\,\mathbb{E}_{a_j}\left[\widehat{F}\big(a_j \mid S\cup\{a_l\}_{l=1}^{j-1}\big)\right]\\ &\geq \mathbb{E}\left[(1-\varepsilon)^2\max_{e:\in X_{j-1}^{\mathcal{M}}} F(e \mid S_{j-1}) - \varepsilon/r\right]\\ &= (1-\varepsilon)^2\max_{e:\in X_{j-1}^{\mathcal{M}}} F(e \mid S_{j-1}) - \varepsilon/r\end{aligned}$$

which completes the proof. □

Lastly, we show that if a_j in a solution $S = \{a_1, \ldots a_{\texttt{rank}}(\mathcal{M})\}$ provides an expected marginal contribution to its preceding set $S_{j-1} \coloneqq \{a_1, \ldots, a_{j-1}\}$ which is close to optimal, then the expected approximation ratio of $F(S)$ approaches $\frac{1}{2}$.

Lemma A.6. *Assume that $S = \{a_1, \ldots, a_r\}$ is such that $\mathbb{E}_{a_j}[F(a_j \mid S_{j-1})] \geq (1-\varepsilon)\max_{e:S_{j-1}\cup\{e\}\in\mathcal{M}} F(e \mid S_{j-1})$ where $S_{j-1} = \{a_l\}_{l=1}^{j-1}$. Then, for a matroid constraint $\mathcal{M}$, we have*

$$\mathbb{E}[F(S)] \geq (1/2 - O(\varepsilon))\texttt{OPT}.$$

Proof of Lemma A.6. Let $O = \{o_1, \ldots, o_r\}$ be an optimal solution, for which $\{o_1, \ldots, o_{i-1}, a_i\} \in \mathcal{I}$ for all i, which as we saw is always possible because of matroid exchange property. Then

$$\begin{aligned}&\mathbb{E}[F(S)]\\ &= \sum_{j=1}^{r}\mathbb{E}[F(a_j \mid S_{j-1})] \geq (1-\varepsilon)\sum_{j=1}^{r}\mathbb{E}[F(o_j \mid S_{j-1})]\\ &\geq (1-\varepsilon)F(O \mid S) \geq (1/2 - O(\varepsilon))\texttt{OPT}.\end{aligned}$$ □

To summarize, by Lemmas A.2 and A.6, Algorithm 2 requires only $O(\log(m)\log(r/\varepsilon)\varepsilon^{-2})$ communication rounds between the server and clients to provide an approximation that is arbitrarily close to $1/2$, in expectation.

Theorem A.7. *Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid of rank r, and $\lambda = \max_{e,i} f_i(\{e\})$. For every $\varepsilon, \delta > 0$ there exist integers K and d for each round of Algorithm 2 such that, w.p. at least $1-\delta$, Algorithm 2 returns $S \in \mathcal{I}$ which satisfies*

$\mathbb{E}[F(S)] \geq (\frac{1}{2} - O(\varepsilon))\texttt{OPT}$. *Moreover, Algorithm 2 has* $O(\log(m)\ \log(r/\varepsilon)\varepsilon^{-2})$ *communication rounds between the server and clients, and it has* $\tilde{O}(d)$ *bit complexity per client per round.*

B. Amplification

The best known multiplicative approximation factor for monotone submodular function maximization under a matroid constraint is $1 - \frac{1}{e}$, which was first achieved by the continuous greedy algorithm in (Călinescu et al., 2011). Both our Algorithms 1 and 2, achieve a $\frac{1}{2}$ approximation factor guarantee, which is short of $1 - \frac{1}{e}$. In order to discuss this technique, we first need to define a continuous extension of submodular functions.

Multilinear extension. The multilinear extension $f^{mle} : [0,1]^m \to \mathbb{R}$ of a set function $f : \{0,1\}^m \to \mathbb{R}$ is

$$\begin{aligned} f^{mle}(\mathbf{x}) &= \sum_{S \subseteq E} f(S) \prod_{e \in S} \mathbf{x}(e) \prod_{e \notin S} (1 - \mathbf{x}(e)) \\ &= \mathbb{E}_{R \sim \mathbf{x}}[f(R)] \end{aligned}$$

where $R \subseteq E$ is a random set that contains or excludes each $e \in E$, with probability (w.p.) $\mathbf{x}(e)$ and $1 - \mathbf{x}(e)$, respectively. By $R \sim \mathbf{x}$ we denote that R is randomly sampled according to $\mathbf{x}$.

As was mentioned earlier, we apply a novel amplification technique introduced in (Badanidiyuru & Vondrák, 2014) and later on extended in (Balkanski et al., 2019; Chekuri & Quanrud, 2019), to boost our approximation guarantee. The amplification technique builds upon the same fundamental idea as in the standard continuous greedy algorithm (Călinescu et al., 2011), which constructs a continuous solution that approximately maximizes the multilinear extension F^{mle} of a monotone submodular function F. That is, at each iteration, the solution $\mathbf{x} \in [0,1]^m$ is updated in the direction of a feasible set $S \in \mathcal{I}$. The key distinction between the continuous greedy algorithm and the amplification procedure (shown in Algorithm 3) lies in the selection of S, which determines the direction in which $\mathbf{x}$ should be updated.

The amplification technique (Algorithm 3) starts by initializing $\mathbf{x} = \mathbf{0}$, and performs $1/\eta$ updates; for $\eta \in (0,1)$ a fixed step-size which is given as input. At each iteration, Algorithm 2 is employed on a surrogate function g which quantifies the marginal contribution to $\mathbf{x}$, when taking a step of size η in the direction of T. Specifically, g is defined as

$$\begin{aligned} g(T) &= F^{mle}(\mathbf{x} + \eta \mathbf{1}_T) - F^{mle}(\mathbf{x}) \\ &= \mathbb{E}_{R \sim (\mathbf{x} + \eta \mathbf{1}_T)}[F(R)] - \mathbb{E}_{R \sim \mathbf{x}}[F(R)] \\ &= \frac{1}{n} \sum_{i=1}^{n} \left[\mathbb{E}_{R \sim (\mathbf{x} + \eta\, \mathbf{1}_T)}[f_i(R)] - \mathbb{E}_{R \sim \mathbf{x}}[f_i(R)] \right] \\ &= \frac{1}{n} \sum_{i=1}^{n} g_i(T). \end{aligned}$$

Algorithm 2 is carried out as a subroutine, with inputs the monotone submodular function g and matroid $\mathcal{M} = (E, \mathcal{I})$, and returns a (random) set $S \in \mathcal{I}$ satisfying $\mathbb{E}[g(S)] \geq (\frac{1}{2} - \varepsilon)g(T)$ for all $T \in I$ (Lemma A.6). Then, the results of (Badanidiyuru & Vondrák, 2014; Balkanski et al., 2019; Chekuri & Quanrud, 2019) imply that Algorithm 3 returns a vector $\mathbf{x}$ which satisfies

$$\begin{aligned} \mathcal{F}(\mathbf{x}) = \mathbb{E}_{R \sim \mathbf{x}}[F(R)] &= \tfrac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{R \sim \mathbf{x}}[f_i(R)] \\ &\geq (1 - \tfrac{1}{e} - O(\varepsilon))\texttt{OPT} \end{aligned}$$

where $\texttt{OPT} = \max_{S \in \mathcal{I}} F(S)$. Note that $\mathbf{x}$ is a convex combination of $1/\eta$ one-hot encoding of sets of rank at most r from $\mathcal{I}$, thus we have $\mathbf{x} \in [0,1]^m$ and $\|\mathbf{x}\|_1 \leq r$. (Note that $\mathbf{x}$ is inside what is known as the matroid polytope of $\mathcal{M}$.)

Finally, given $\mathbf{x} \in [0,1]^m$ with $\|\mathbf{x}\|_1 \leq r$, a discrete solution $S \in \mathcal{I}$ can be obtained using appropriate rounding techniques. Methods such as pipage rounding (Călinescu et al., 2011) and swap rounding (Chekuri et al., 2010) operate independently of the objective function, so they do not require direct access to it. As a result, the central server can apply any of these rounding schemes without knowledge of the underlying client functions. The loss in the approximation guarantee can be arbitrarily small when using these rounding techniques.

Theorem B.1. *Algorithm 3, with Algorithm 2 as a subroutine, consists of* $O(\log(m)\log(\frac{r}{\varepsilon\,\eta})\frac{1}{\varepsilon\,\eta})$ *communication rounds between the central server and clients and w.p. at least* $1 - \delta$ *obtains a* $1 - 1/e - O(\varepsilon)$ *approximation for Problem 2, with step-size* $\eta = O(\varepsilon^2 \log^{-1}(\frac{1}{\delta}))$.

We point out that Algorithm 3 can also use Algorithm 1 as a subroutine, leading to the following result.

Theorem B.2. *Algorithm 3, with Algorithm 1 as a subroutine, consists of* $O(r/\eta)$ *communication rounds between the central server and clients and w.p. at least* $1 - \delta$ *obtains a* $1 - 1/e - O(\varepsilon)$ *approximation for Problem 2, with step-size* $\eta = O(\varepsilon^2 \log^{-1}(\frac{1}{\delta}))$.

Algorithm 3 Improved approximations through amplification

Input: Matroid $\mathcal{M} = (E, \mathcal{I})$, step-size η, accuracy parameters $\delta, \varepsilon \in (0, 1)$.
$\mathbf{x} = \mathbf{0}$
for $1/\eta$ iterations **do**
  Define $g(T) = \mathbb{E}_{R \sim (\mathbf{x} + \eta \mathbf{1}_T)}[F(R)] - \mathbb{E}_{R \sim \mathbf{x}}[F(R)]$
  Run Algorithm 1 or 2 with $(g, \mathcal{M}, \delta, \varepsilon)$, and return S.
  $\mathbf{x} \leftarrow \mathbf{x} + \eta \, \mathbf{1}_S$
end for
Output: $\mathbf{x}$ /* *Apply an appropriate rounding on* $\mathbf{x}$ *to obtain* S.*/

C. Differential privacy via shuffling

The primary focus of this work is not on ensuring privacy guarantees, but developing a framework that provides strong approximation guarantees while being adaptable to various privacy settings. In this section, we discuss the privacy aspect of our approach and highlight that ensuring privacy comes at the cost of increased bit complexity. In our setting, the central server issues *sum queries* to selected clients, i.e. each client responds with *bounded* real-valued numbers, and the server aims to compute the sum of their responses. The clients do not trust the central server, and are therefore unwilling to send their responses in the clear. This has been addressed by several models of privacy and security in the literature, e.g. multiparty computation (MPC); differential privacy and secret sharing.

One approach to addressing privacy concerns is to leverage powerful tools of modern cryptography, such as MPC and secure function evaluation, to emulate centralized algorithms in a setting without a trusted server (Dwork et al., 2006). Another common method for achieving a secure aggregation mechanism is through additive masking (Bell et al., 2020; Bonawitz et al., 2016), which is based on secret sharing and requires participating clients to communicate with each other. These techniques currently introduce significant bandwidth constraints and delays, making them impractical for large-scale deployments.

Another solution to the privacy concern is for each participating client to apply a differentially private algorithm locally to their data and share only the output of the algorithm with the central server that aggregates clients' responses. However, the utility guarantee of such approaches is far from optimal. Beimel et al. (2008) and Chan et al. (2012) show that estimating the sum of bits, one held by each client, requires error $\Omega(\sqrt{K}/\varepsilon)$, where K is the number of participating clients. We note that the utility guarantee of the approach from (Wang et al., 2023) at each iteration per element is $O(\sqrt{r\, n\, \log m}/\gamma)$, where γ is their Poisson sampling rate.

A suitable approach for our setting which is scalable and has strong privacy guarantees, is the work of Cheu et al. (2019). They proposed a differentially private model for real-valued sum queries that eliminates the need for a trusted central server. Their *shuffled differentially private* model leverages an anonymous shuffler to permute client messages before processing, and it requires a relatively high bit complexity to encode client massages. The technical details of their method are not in the scope of this paper. Instead, we focus on their final results and implications to our framework.

By leveraging the algorithms from (Cheu et al., 2019) (specifically, Algorithms 5 and 6), we can ensure privacy in each communication round of our algorithms at the cost of an additional $O(\sqrt{K})$ factor in the bit complexity. Furthermore, by using composition of differential privacy (Bun & Steinke, 2016; Dwork & Lei, 2009; Dwork et al., 2010), we can guarantee differential privacy across all communication rounds of Algorithms 1 and 2. The following result follows directly from Theorem 22 of (Cheu et al., 2019).

Proposition C.1. *Let $\varepsilon, \delta > 0$. Suppose that in Algorithm 1, at each communication round, K clients (K sufficiently large) are sampled, and each client sends d bounded real-valued numbers to the central server. There exists a (ε', δ') differentially private algorithm, `DPAlg` ((Cheu et al., 2019) Algorithm 6), with $\varepsilon' = O\left(\frac{r}{\varepsilon K} \log \frac{1}{\delta'} \sqrt{\log \frac{m\, r}{\delta}}\right)$ and $e^{-\Omega(K^{1/4})} < \delta' < \frac{1}{K}$, which requires every participating client to encode each real-valued number using $O(\varepsilon \sqrt{K})$ bits, and for any $e \in E \setminus S$ it guarantees*

$$\Pr\left[\left|\textit{Output of } \texttt{DPAlg} - \widehat{F}\big(e \mid S\big)\right| > \varepsilon/r\right] \leq \delta/(m\, r).$$

Furthermore, using composition of differential privacy, `DPAlg` in every communication round of Algorithm 1, results in an $(\varepsilon'\, m\, r, \delta'\, m\, r)$-DP algorithm with $O(\varepsilon\, d \sqrt{K})$ bit complexity per round per client, and the approximation guarantee of Algorithm 1 is maintained.

Note that the DP guarantee comes at the cost of significantly increased bit complexity. Finally, we note that the same differentially private subroutine can also be applied in Algorithm 2.

We further note that the differentially private algorithms developed for the centralized setting (Gupta et al., 2010; Mitrovic et al., 2017; Rafiey & Yoshida, 2020; Chaturvedi et al., 2021; Sadeghi & Fazel, 2021) are not applicable in the federated setting.

D. Further experimental results on Algorithm 1

We provide further numerical results on Algorithm 1 on the same datasets and different choices for K and d.

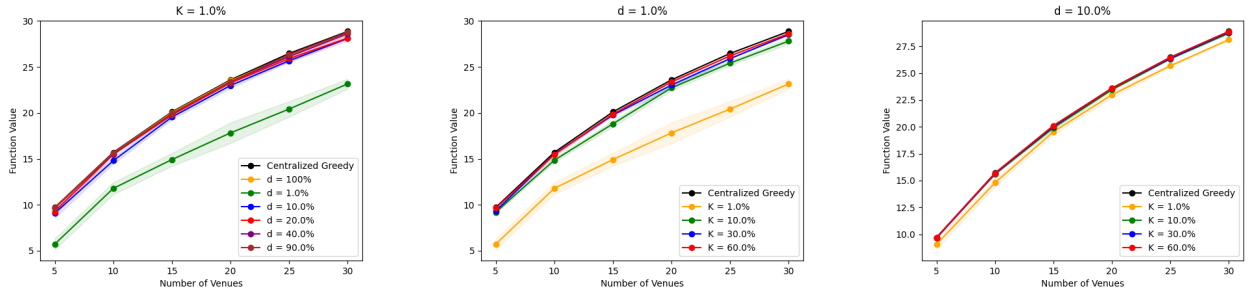

Figure 4: Results on the `Max Coverage` problem. Performance comparison between Algorithm 1 and the Centralized Greedy algorithm across different values of K and d on the DBLP dataset.

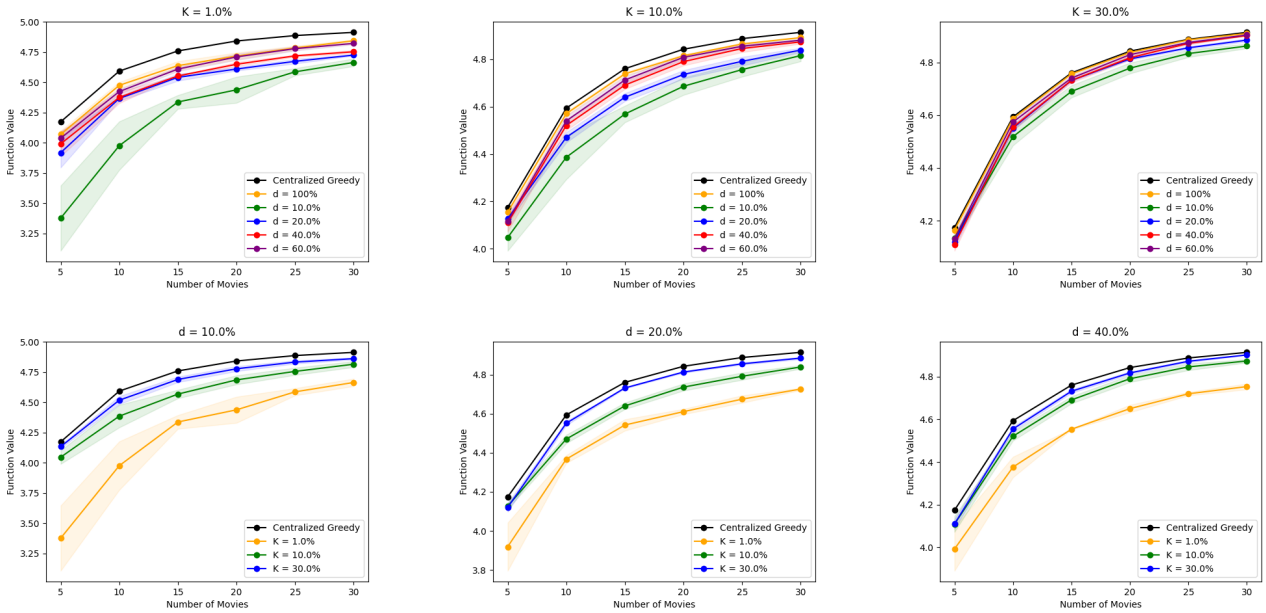

Figure 5: Results on the `Facility Location` problem. Performance comparison between Algorithm 1 and the Centralized Greedy algorithm across different values of K and d on the MovieLens dataset.

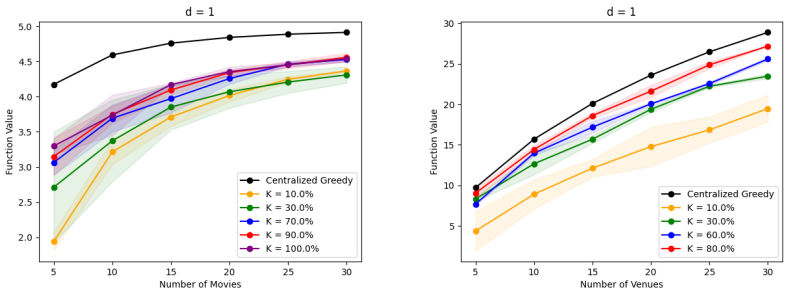

Figure 6: Effect of client sampling ratio K on the performance of Algorithm 1 and the Centralized Greedy baseline for $d = 1$ on DBLP and MovieLens datasets. Due to the much larger client-to-ground-set ratio in DBLP ($n \gg m$), our algorithm achieves performance closer to the centralized greedy approach on DBLP compared to MovieLens.

E. Parameters of Algorithm 2

The parameters present in the algorithm are τ, ε, r, d, and K. Since the sampling space of this algorithm is larger, it is crucial to use the optimal values for these parameters and analyze the impact of each parameter on the number of rounds and function value.

Analysis of r We first point out that Algorithm 2 is applicable to the cases where r is relatively large. Moreover, unlike Algorithm 1, r is directly involved in the core selection process of the algorithm. The quality of inclusion of elements depends on the quality of the feasible random sequence $a_1, \ldots, a_{r'}$, which is influenced by r.

- A higher r improves the quality of the $a_1, \ldots, a_{r'}$, leading to better overall results.
- We chose $r = 200$, which is approximately 5% of the movies present in the MovieLens dataset. Figure 7 shows the performance of the Algorithm 2 with different parameters and as we see it reaches subset of size 200 before the communication round 200.

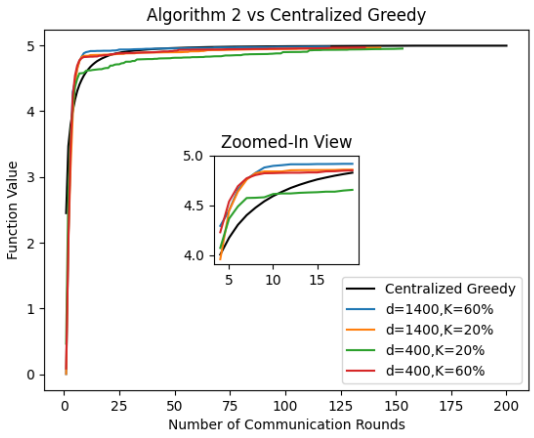

Figure 7: Full execution of Algorithm 2 vs. Centralized Greedy.

Analysis of K and d We analyzed the impact of K and d on both the number of rounds and function value by fixing one parameter and varying the other Figures 7 to 9.

- We observed that increasing either K or d resulted in a decrease in communication rounds and an improvement in function value.
- The results demonstrated a faster convergence over the centralized greedy algorithm when appropriately tuning K and d.

Analysis of τ and ε To determine the appropriate range for these parameters, we first analyzed initial τ (initial threshold) and ε (threshold decay rate) in isolation from the other parameters. We tested various combinations of ε and τ in the centralized version of the algorithm, where all combinations of (e, j) are considered from all clients without any sampling—thus not involving d and K.

- We examined the impact of ε on the function value by fixing the initial threshold τ and drawing multiple plots for different fixed τ values Figure 11.
- We observed that for a fixed τ, increasing ε led to a decrease in function value and a reduction in the number of communication rounds required to add 200 elements to set S.
- This is consistent with theoretical expectations since increasing ε allows more j values satisfying $|X_j| \leq (1-\varepsilon)|X|$ to be selected, resulting in more elements being added in each round. However, this also reduces the quality of element inclusion.
- An exception occurs when τ is lower and ε is higher, where the number of rounds becomes highly random. In such cases, $|X_j|$ for all j may be high because τ is lower, leading to no X_j satisfying $|X_j| \leq (1-\varepsilon)|X|$, resulting in unexpectedly more communication rounds.

Next, we analyzed the impact of τ on function value by fixing ε and increasing τ, generating plots for various fixed ε values Figure 10:

- We found that for a fixed ε, increasing τ led to an increase in function value and an increase in the number of communication rounds required to add 200 elements to S.
- This aligns with theoretical expectations since increasing τ reduces $|X_j|$ (as per line 16 of the pseudocode). This, in turn, reduces the count of $e \in X$ that satisfy the condition, pushing j toward 0.
- As fewer elements are added per round, the quality of inclusion improves, leading to higher function values.

Choosing τ and ε for Algorithm 2 Since Algorithm 2 is not centralized and involves more samplings, the best function value and number of rounds are achieved in the centralized variant with higher τ and ε. However, due to the sampling process in the federated setting, we must reduce τ and ε to accommodate for the loss of (e, j) tuples.

- In our experiments, we found that setting $\varepsilon = 0.4$ and $\tau = 0.006$ yielded **better function values** and an optimal number of communication rounds across different values of K and d.

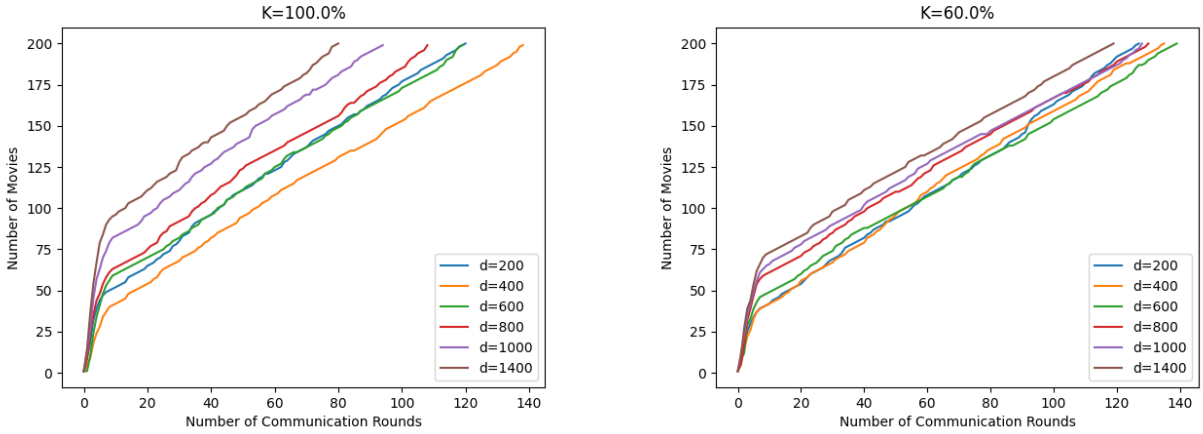

Figure 8: Growth of S with respect to d in Algorithm 2

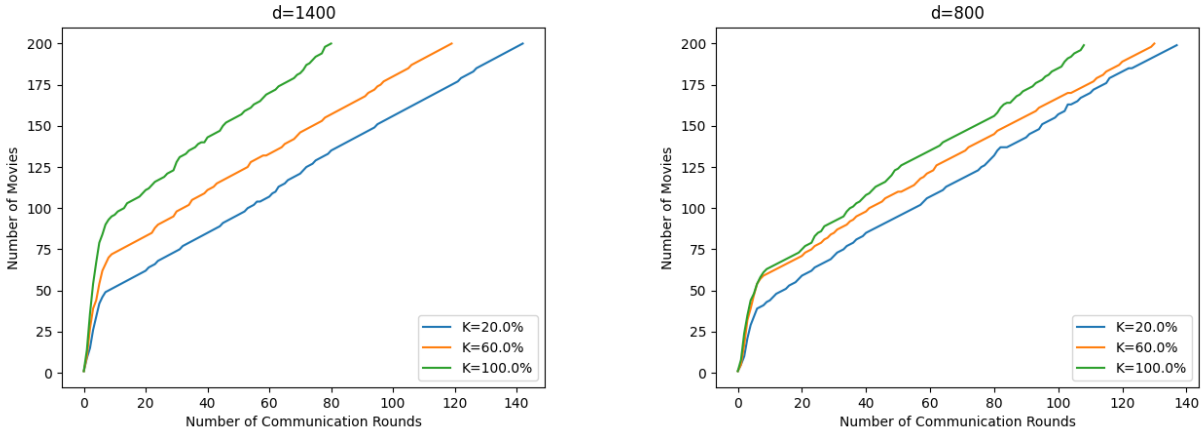

Figure 9: Growth of S with respect to K in Algorithm 2

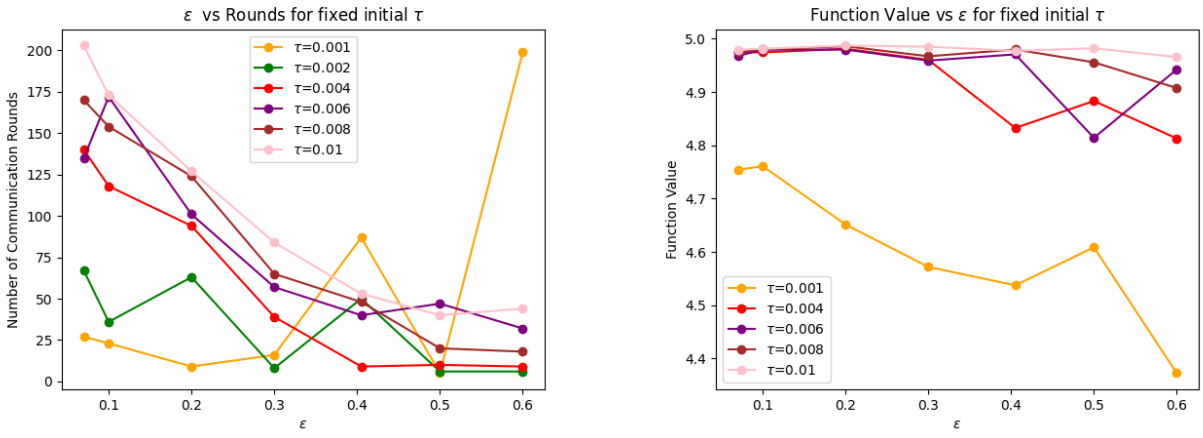

Figure 10: Fixed initial τ

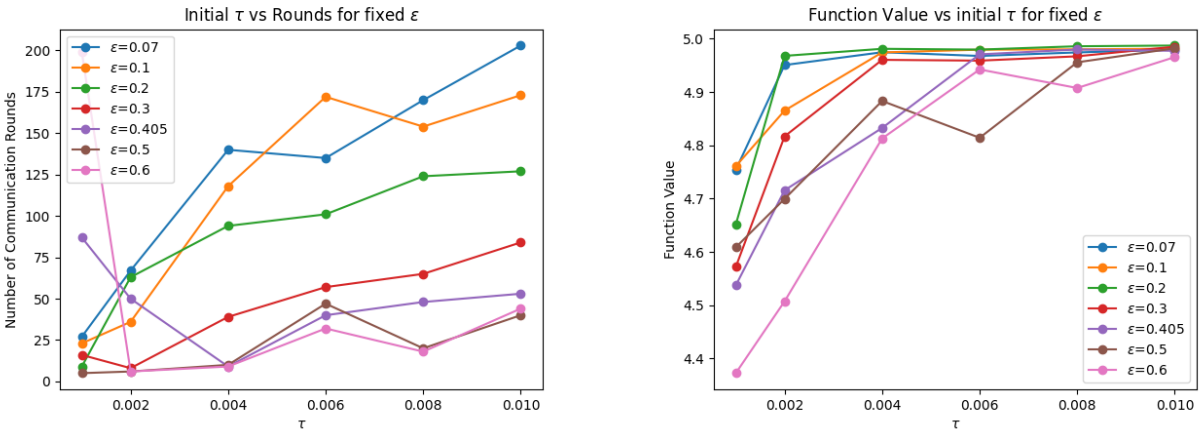

Figure 11: Fixed ϵ