Distributed Experimental Design Networks

Yuanyuan Li,[1] Lili Su,[1] Carlee Joe-Wong,[2] Edmund Yeh,[1] and Stratis Ioannidis,[1]
[1]Northeastern University, [2]Carnegie Mellon University,
Email: yuanyuanli@ece.neu.edu, l.su@northeastern.edu, cjoewong@andrew.cmu.edu, {eyeh, ioannidis}@ece.neu.edu

***Abstract*—As edge computing capabilities increase, model learning deployments in diverse edge environments have emerged. In experimental design networks, introduced recently, network routing and rate allocation are designed to aid the transfer of data from sensors to heterogeneous learners. We design efficient experimental design network algorithms that are (a) distributed and (b) use multicast transmissions. This setting poses significant challenges as classic decentralization approaches often operate on (strictly) concave objectives under differentiable constraints. In contrast, the problem we study here has a non-convex, continuous DR-submodular objective, while multicast transmissions naturally result in non-differentiable constraints. From a technical standpoint, we propose a distributed Frank-Wolfe and a distributed projected gradient ascent algorithm that, coupled with a relaxation of non-differentiable constraints, yield allocations within a $1 - 1/e$ factor from the optimal. Numerical evaluations show that our proposed algorithms outperform competitors with respect to model learning quality.**

***Index Terms*—Experimental Design, DR-submodularity, Bayesian linear regression, Distributed algorithm.**

I. Introduction

We study *experimental design networks*, as introduced by Liu at al. [1]. In these networks, illustrated in Fig. 1, learners and data sources are dispersed across different locations in a network. Learners receive streams of data collected from the sources, and subsequently use them to train models. We are interested in rate allocation strategies that maximize the quality of model training at the learners, subject to network constraints. This problem is of practical significance. For instance, in a smart city [2], [3], various sensors capture, e.g., image, temperature, humidity, traffic, and seismic measurements, which can help forecast transportation traffic, the spread of disease, pollution levels, the weather, etc. Distinct, dispersed public service entities, e.g., a transportation authority, an energy company, the fire department, etc., may perform different training and prediction tasks on these data streams.

Even though the resulting optimization of rate allocations is non-convex, Liu et al. [1] provide a polynomial-time $(1-1/e)$-approximation algorithm, exploiting a useful property of the learning objective, namely, continuous DR-submodularity [4], [5]. Though [1] lays a solid foundation for studying this problem, the algorithm proposed suffers from several limitations. First, it is centralized, and requires a full view of network congestion conditions, demand, and learner utilities. This significantly reduces scalability when the number of sources and learners are large. In addition, it uses unicast transmissions between sources and learners. In practice, this significantly under-utilizes network resources when learner interests in data streams overlap.

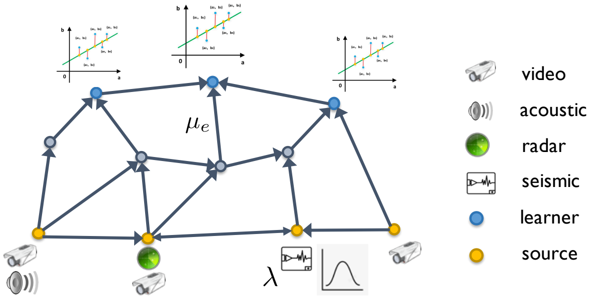


Fig. 1: An *experimental design network* [1]. Sources (yellow) generate streams of data from diverse sensors, e.g., cameras, microphones, seismic sensors, etc. Learners (blue) train distinct models over (possibly overlapping) received data. We wish to allocate bandwidth to data traffic in a manner that maximizes the social welfare, i.e., the aggregate quality of models across learners.

In this paper, we aim to design efficient algorithms that are (a) distributed and (b) use multicast transmission. Achieving this goal is far from trivial. First, classic decentralization approaches, such as, primal-dual algorithms [6]–[8], often operate on (strictly) concave objectives. In contrast, the problem we study here has a non-concave, continuous DR-submodular objective. Second, multicast transmissions naturally result in non-differentiable constraints (see [7], [9], but also Eqs. (4) and (6) in Sec. III). This further hinders standard decentralization techniques. Our **contributions** are as follows:

- We incorporate multicast transmissions to experimental design networks. This is more realistic when learning jointly from common sensors, and yields a significantly higher throughput in comparison to unicast transmissions.
- We prove that, assuming Poisson data streams in steady state and Bayesian linear regression as a learning task, as in [1], our experimental design objective remains continuous DR-submodular.
- We construct *both* centralized *and* distributed algorithms within a $1 - 1/e$ factor from the optimal in this setting. For the latter, we make use of a primal dual technique that addresses both the non-differentiability of constituent multicast constraints, as well as the lack of strict convexity exhibited by our problem.
- We conduct extensive simulations over both synthetic and backbone network topologies. Our proposed algorithms outperform all competitors w.r.t. the quality of model estimation, and our distributed algorithms perform closely to their centralized versions.

From a technical standpoint, we couple the Frank-Wolfe algorithm by Bian et al. [4], also used by Liu et al. [1], with a nested distributed step; the latter deals with the non-differentiability of multicast constraints through an l_p relaxation of the max norm. We also implement two additional extensions sketched out by Liu et al. [1]: we consider (a) Gaussian data sources, that are (b) subject to heterogeneous noise. We incorporate both in our mathematical formulation and theoretically and experimentally characterize performance under these extensions. Gaussianity requires revisiting how gradient estimation is performed, compared to Liu et al., as well as devising new estimation bounds.

The remainder of this paper is organized as follows. Sec. II provides a literature review. We introduce our distributed model in Sec. III. Sec. IV describes our analysis of the problem and proposed centralized algorithm, while Sec. V describes our distributed algorithm. We propose additional distributed algorithms in Sec. VI. We present numerical experiments in Sec. VII, and conclude in Sec. VIII.

II. Related Work

Experimental Design. As discussed by Liu et al. [1], experimental design is classic under a single user with an experiment budget constraint [10], [11], while the so-called D-Optimality criterion is a popular objective [12]–[16]. Liu et al. [1] are the first to extend this objective to the context of experimental design networks. As discussed in the introduction, we deviate from Liu et al. by proposing a decentralized algorithm and considering multicast transmissions; both are practically important and come with technical challenges. Liu et al. make additional restrictive assumptions, including, e.g., that data samples come from a finite set and that labeling noise is homogeneous across sources, but mention that their analysis could be extended to amend these assumptions. We implement this extension by considering Gaussian sources and noise heteroskedasticity, and proving gradient estimation bounds using appropriate Chernoff inequalities (see, e.g., Lem. 1).

Submodular Maximization. Submodularity is traditionally explored within the context of set functions [17], but can also be extended to functions over the integer lattice [5] and the continuous domain [4]. Maximizing a monotone submodular function subject to a matroid constraint is classic. Krause and Golovin [18] show that the greedy algorithm achieves a $1/2$ approximation ratio. Calinescu et al. [17] propose a *continuous greedy* algorithm improving the ratio to $1-1/e$ that applies a Frank-Wolfe (FW) [19] variant to the multilinear extension of the submodular objective. With the help of auxiliary potential functions, Bian et al. [4] show that the same FW variant can be used to maximize continuous DR-submodular functions within a $1-1/e$ ratio. The centralized algorithm by Liu et al. [1], and ours, are applications of the FW variant [4]; in both cases, recovering their guarantees requires devising novel gradient estimators and bounding their estimation accuracy. We also depart by considering a distributed version of this algorithm, where each node accesses only neighborhood knowledge.

Convergence of Primal-Dual Algorithms. Nedić and Ozdaglar [20] propose a subgradient algorithm for generating approximate saddle-point solutions for a convex-concave function. Assuming Slater's condition and bounded Lagrangian gradients, they provide bounds on the primal objective function. Alghunaim and Sayed [21] prove linear convergence for primal-dual gradient methods. The methods apply to augmented Lagrangian formulations, whose primal objective is smooth and strongly convex under equality constraints (ours are inequality constraints). Lyapunov equations are usually employed for a continuous version of the primal-dual gradient algorithm [7]; this requires objectives to be strictly concave. Feijer and Paganini [22] prove the stability of primal–dual gradient dynamics with concave objectives through Krasovskii's method and the LaSalle invariance principle. We follow [22] to ensure convergence of our decentralized algorithm.

Distributed Algorithms. Distributed algorithms for the maximization of strictly concave objectives under separable constraints are classic (see, e.g., [6], [7], [9], [23]). Our objective is continuous DR-submodular; thus, these methods do not directly extend to our setting. Tychogiorgos et al. [24] provide the theoretical foundations for distributed dual algorithm solution of non-convex problems. Mokhtari et al. [25] propose a partially decentralized continuous greedy algorithm for DR-submodular maximization subject to down-closed convex constraints and prove a $1-1/e$ guarantee. However, the conditional gradient update step they propose requires global information and remains centralized. Our analysis requires combining above techniques, with the (centralized) Frank-Wolfe variant by Bian et al. [4], which yields a $1-1/e$ approximation guarantee. Doing so requires dealing with both the lack of strict convexity of the constituent problem, as well as the non-differentiability of multicast constraints.

III. Problem Formulation

Our model, and its exposition below, follows closely Liu et al. [1]: for consistency, we use the same notation and terminology. We depart in multiple ways. First and foremost, we (a) seek a *distributed algorithm* determining the rate allocation, requiring knowledge only from itself and its neighbourhoods, while the centralized algorithm requires global information, and (b) we extend the analysis from unicast to multicast, allowing sharing of the same traffic across learners, thereby increasing throughput. We also implement two potential extensions drafted by Liu et al. [1]: we (c) generate features from Gaussian sources instead of a finite set, changing subscripts of variables from feature $\boldsymbol{x}$ to source s, (d) adopt source instead of hop-by-hop routing, which changes the optimized random variables, leading to a general directed graph instead of a DAG (directed acyclic graph), and (f) incorporate heterogeneous noise over source s and types t instead of homogeneous constant noise.

Network. We model the system as a multi-hop network with a topology represented by a *directed graph* $\mathcal{G}(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$ is the set of links. Each link $e=(u,v)\in\mathcal{E}$ has a link capacity $\mu^e\geq 0$. Sources $\mathcal{S}\subset\mathcal{V}$ generate data streams, while learners $\mathcal{L}\subset\mathcal{V}$ reside at sinks.

Data Sources. Each data source $s \in \mathcal{S}$ generates a sequence of labeled pairs $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ of type t according to a Poisson process of rate $\lambda_{s,t} \geq 0$, corresponding to measurements or experiments the source conducts (each pair is a new measurement). Intuitively, features $\boldsymbol{x}$ correspond to covariates in an experiment (e.g., pixel values in an image, etc.), label types $t \in \mathcal{T}$ correspond to possible measurements (e.g., temperature, radiation level, etc.), and labels y correspond to the actual measurement value collected (e.g., 23°C).

The generated data follows a linear regression model [26], [27], i.e., for every type $t \in \mathcal{T}$ from source $s \in \mathcal{S}$, there exists a $\boldsymbol{\beta}_t \in \mathbb{R}^d$ such that $y = \boldsymbol{x}^\top \boldsymbol{\beta}_t + \epsilon_{s,t}$ where $\epsilon_{s,t} \in \mathbb{R}$ are i.i.d. zero mean normal noise variables with variance $\sigma^2_{s,t} > 0$. Departing from Liu et al. [1], but also from classic experimental design [10], where s samples feature vectors $\boldsymbol{x} \in \mathbb{R}^d$ from a finite set, we assume that they are sampled from a *Gaussian* distribution $N(\boldsymbol{0}, \boldsymbol{\Sigma}_s)$. Again in contrast to [1], we allow for *heterogeneous* (also known as heteroskedastic) noise levels $\sigma_{s,t}$ across *both* sources and experiment types.

Learners and Bayesian Linear Regression. Each learner $\ell \in \mathcal{L}$ wishes to learn a model $\boldsymbol{\beta}_{t^\ell}$ for some type $t^\ell \in \mathcal{T}$, via *Bayesian linear regression.* In particular, each ℓ has a Gaussian prior $N(\boldsymbol{\beta}_0^\ell, \boldsymbol{\Sigma}_0^\ell)$ on the model $\boldsymbol{\beta}_{t^\ell}$ it wishes to estimate. We assume that the system operates for a data acquisition time period T. Let $n_s^\ell \in \mathbb{N}$ be the cumulative number of pairs $(\boldsymbol{x}, y)$ from source s collected by learner ℓ during this period, and $\boldsymbol{n}^\ell = [n_s^\ell]_{s \in \mathcal{S}}$ the vector of arrivals at learner ℓ from all sources. We denote by $\boldsymbol{x}^\ell_{s,i}$, $y^\ell_{s,i}$ the i-th feature and label generated from source s to reach learner ℓ, and $\boldsymbol{X}^\ell = [[\boldsymbol{x}^\ell_{s,i}]_{i=1}^{n_s^\ell}]_{s \in \mathcal{S}}$, $\boldsymbol{y}^\ell = [[y^\ell_{s,i}]_{i=1}^{n_s^\ell}]_{s \in \mathcal{S}}$ the feature matrix and label vector, respectively, received at learner ℓ up to time T. Then, maximum a-posterior (MAP) estimation [1], [27] at learner ℓ amounts to:

$$\begin{aligned}\hat{\boldsymbol{\beta}}^\ell_{\texttt{MAP}} = ((\boldsymbol{X}^\ell)^\top (\tilde{\boldsymbol{\Sigma}}^\ell)^{-1} \boldsymbol{X}^\ell + (\boldsymbol{\Sigma}_0^\ell)^{-1})^{-1} \cdot \\ ((\boldsymbol{X}^\ell)^\top (\tilde{\boldsymbol{\Sigma}}^\ell)^{-1} \boldsymbol{y}^\ell + (\boldsymbol{\Sigma}_0^\ell)^{-1} \boldsymbol{\beta}_0^\ell),\end{aligned} \tag{1}$$

where $\tilde{\boldsymbol{\Sigma}}^\ell = \mathbb{R}^{n^\ell \times n^\ell}$ is a diagonal noise covariance matrix, containing corresponding source noise covariances σ^2_{s,t^ℓ} in its diagonal. The quality of this estimator is determined by the error covariance [27], i.e., the $d \times d$ matrix:

$$\texttt{cov}(\hat{\boldsymbol{\beta}}^\ell_{\texttt{MAP}} - \boldsymbol{\beta}_{t^\ell}) = \left((\boldsymbol{X}^\ell)^\top (\tilde{\boldsymbol{\Sigma}}^\ell)^{-1} \boldsymbol{X}^\ell + (\boldsymbol{\Sigma}_0^\ell)^{-1}\right)^{-1}. \tag{2}$$

The covariance summarizes the estimator quality in all directions in $\mathbb{R}^d$: directions $\boldsymbol{x} \in \mathbb{R}^d$ of high variance (e.g., eigenvectors in which eigenvalues of $\texttt{cov}(\hat{\boldsymbol{\beta}}^\ell_{\texttt{MAP}} - \boldsymbol{\beta}_{t_\ell})$ are high) are directions in which the prediction $\hat{y} = \boldsymbol{x}^\top \hat{\boldsymbol{\beta}}^\ell_{\texttt{MAP}}$ will have the highest prediction error.

Network Constraints. Data pairs $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ of type $t \in \mathcal{T}$ generated by sources are transmitted over paths in the network and eventually delivered to learners. Furthermore, we consider the *multirate multicast* transmission [7], which saves network resources compared to unicast. That is, we assume that each source s has a set of paths $\mathcal{P}_{s,t}$ over which data pairs of type t are routed: each path $p \in \mathcal{P}_{s,t}$ links to a different learner. Note that $|\mathcal{P}_{s,t}|$, the size of $\mathcal{P}_{s,t}$, equals the number of learners with type t. We denote the *virtual* rate [7], with which data pairs of type t from source s are transmitted through path $p \in \mathcal{P}_{s,t}$, as $\lambda^p_{s,t} \geq 0$. Let $P_{\texttt{TOT}} = \sum_{s \in \mathcal{S}, t \in \mathcal{T}} |\mathcal{P}_{s,t}|$ be the total number of paths. We refer to the vector

$$\boldsymbol{\lambda} = [\lambda^p_{s,t}]_{s \in \mathcal{S}, t \in \mathcal{T}, p \in \mathcal{P}_{s,t}} \in \mathbb{R}_+,^{P_{\texttt{TOT}}} \tag{3}$$

as the global rates allocation. To satisfy multicast link capacity constraints, for each link $e \in \mathcal{E}$, we must have

$$\sum_{s \in \mathcal{S}, t \in \mathcal{T}} \max_{p \in \mathcal{P}_{s,t} : e \in p} \lambda^p_{s,t} \leq \mu^e. \tag{4}$$

Note that only data pairs of the same type generated and same source can be multicast together. For learner l, we denote by

$$\lambda^\ell_s = \lambda^p_{s,t^\ell} \tag{5}$$

the incoming traffic rate of type t^ℓ at $\ell \in \mathcal{L}$ from source $s \in \mathcal{S}$. Note that $p \in \mathcal{P}_{s,t^\ell}$ and ℓ is the last node of p. At source s, for each $t \in \mathcal{T}$, we have the constraints:

$$\max_{p \in \mathcal{P}_{s,t}} \lambda^p_{s,t} \leq \lambda_{s,t}. \tag{6}$$

Note that the left hand sides of both constraints in (4) and (6) are non-differentiable. We adopt the following assumption on the network substrate (Asm. 1 in [1]):

Assumption 1. *For $\boldsymbol{\lambda} \in \mathcal{D}$, the system is stable and, in steady state, pairs $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ of type t^ℓ arrive at learner $\ell \in \mathcal{L}$ according to $|\mathcal{S}|$ independent Poisson processes with rate λ^ℓ_s.*

As discussed in [1], this is satisfied if, e.g., the network is a Kelly network [28] where Burke's theorem holds [27].

D-Optimal Design Objective. The so-called D-optimal design objective [10] for learner ℓ is given by:

$$G^\ell(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell) = \log\det(\texttt{cov}(\hat{\boldsymbol{\beta}}^\ell_{\texttt{MAP}} - \boldsymbol{\beta}_{t^\ell})), \tag{7}$$

where the covariance is given by Eq. (2). As the latter summarises the expected prediction error in all directions, minimizing the $\log\det$ (i.e., the sum of logs of eigenvalues of the covariance) imposes an overall bound on this error.

Aggregate Expected Utility Optimization. Under Asm. 1, the arrivals of pertinent data pairs at learner ℓ from source s form a Poisson process with rate λ^ℓ_s. The PMF of arrivals is:

$$\mathbf{P}[\boldsymbol{n}^\ell = \boldsymbol{n}] = \prod_{s \in \mathcal{S}} \frac{(\lambda^\ell_s T)^{n_s} e^{-\lambda^\ell_s T}}{n^\ell_s!}, \tag{8}$$

for all $\boldsymbol{n} = [n_s]_{s \in \mathcal{S}} \in \mathbb{N}^{|\mathcal{S}|}$ and $\ell \in \mathcal{L}$. Then, the PDF (probability distribution function) of features is:

$$f(\boldsymbol{X}^\ell = \boldsymbol{X}) = \prod_{s \in \mathcal{S}} \prod_{i=1}^{n^\ell_s} \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{|\boldsymbol{\Sigma}_s|}} e^{-\frac{1}{2} \boldsymbol{x}_{s,i}^\top \boldsymbol{\Sigma}_s^{-1} \boldsymbol{x}_{s,i}}, \tag{9}$$

for all $\boldsymbol{X} = [[\boldsymbol{x}_{s,i}]_{i=1}^{n_s}]_{s\in\mathcal{S}} \in \mathbb{R}^{\sum_s n_s}$ and $\ell \in \mathcal{L}$. We define the utility at learner $\ell \in \mathcal{L}$ as its expected D-optimal design objective, namely:

$$\begin{aligned}U^\ell(\boldsymbol{\lambda}^\ell) &= \mathbb{E}_{\boldsymbol{n}^\ell}\left[\mathbb{E}_{\boldsymbol{X}^\ell}[G^\ell(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell)|\boldsymbol{n}^\ell]\right]\\ &= \sum_{\boldsymbol{n}\in\mathbb{N}^{|\mathcal{S}|}} \mathbf{P}[\boldsymbol{n}^\ell = \boldsymbol{n}] \int_{\boldsymbol{X}\in\mathbb{R}^{\sum_s n_s}} G^\ell(\boldsymbol{X}, \boldsymbol{n}) f(\boldsymbol{X}^\ell = \boldsymbol{X})\mathrm{d}\boldsymbol{X},\end{aligned}$$

where $\boldsymbol{\lambda}^\ell = [\lambda_s^\ell]_{s\in\mathcal{S}}$, and D-optimal design objective G^ℓ is given by Eq. (7). We wish to solve the following problem:

$$\text{Maximize:} \quad U(\boldsymbol{\lambda}) = \sum_{\ell\in\mathcal{L}} (U^\ell(\boldsymbol{\lambda}^\ell) - U^\ell(\mathbf{0})), \tag{10a}$$

$$\text{s.t.} \quad \boldsymbol{\lambda} \in \mathcal{D}, \tag{10b}$$

where $U^\ell(\mathbf{0})$ is a lower bound[1] for $U^\ell(\boldsymbol{\lambda}^\ell)$ and feasible set $\mathcal{D}$ is defined by constraints (3)-(6). Note that the feasible set is a down-closed convex set [4]. However, this problem is not convex, as the objective is non-concave.

IV. Centralized Algorithm

In this section, we propose a centralized polynomial-time algorithm with a new gradient estimation, as required by the presence of Gaussian sources, to solve Prob. (10). By establishing submodularity, we achieve an optimality guarantee of $1 - 1/e$.

A. DR-submodularity

To solve this non-convex problem, we utilize a key property here: *diminishing-returns submodularity*, defined as follows:

Definition 1 (DR-Submodularity [4], [5])**.** *A function $f : \mathbb{N}^p \to \mathbb{R}$ is called* diminishing-returns (DR) submodular *iff for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{N}^p$ such that $\boldsymbol{x} \leq \boldsymbol{y}$ and all $k \in \mathbb{N}$,*

$$f(\boldsymbol{x} + k\boldsymbol{e}_j) - f(\boldsymbol{x}) \geq f(\boldsymbol{y} + k\boldsymbol{e}_j) - f(\boldsymbol{y}), \tag{11}$$

for all $j = 1, \ldots, p$, where $\boldsymbol{e}_j$ is the j-th standard basis vector. Moreover, if Eq. (11) *holds for a real valued function $f : \mathbb{R}_+^p \to \mathbb{R}$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ s.t. $\boldsymbol{x} \leq \boldsymbol{y}$ and all $k \in \mathbb{R}_+$, the function is called* continuous DR-submodular.

The following theorem establishes that objective (10a) is a continuous DR-submodular function.

Theorem 1. *Objective $U(\boldsymbol{\lambda})$ is (a) monotone-increasing and (b) continuous DR-submodular with respect to $\boldsymbol{\lambda}$. Moreover, the partial derivative of U is:*

$$\frac{\partial U}{\partial \lambda_{s,t}^p} = \sum_{n=0}^{\infty} \Delta_s^\ell(\boldsymbol{\lambda}^\ell, n) \cdot \mathbf{P}[n_s^\ell = n] \cdot T, \tag{12}$$

where $t = t^\ell$, ℓ is the last node of p, the distribution P *is Poisson described by* (8)*, with parameters governed by $\lambda_s^\ell T$, and*

$$\begin{aligned}\Delta_s^\ell(\boldsymbol{\lambda}^\ell, n) = \mathbb{E}_{\boldsymbol{n}^\ell}&\left[\mathbb{E}_{\boldsymbol{X}^\ell}[G^\ell(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell)|\boldsymbol{n}^\ell]|n_s^\ell = n+1\right]\\ -\mathbb{E}_{\boldsymbol{n}^\ell}&\left[\mathbb{E}_{\boldsymbol{X}^\ell}[G(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell)|\boldsymbol{n}^\ell]|n_s^\ell = n\right].\end{aligned} \tag{13}$$

Algorithm 1: Frank-Wolfe Variant

```
Input: $U : \mathcal{D} \to \mathbb{R}_+$, $\mathcal{D}$, stepsize $\delta \in (0, 1]$.
$\boldsymbol{\lambda}^0 = 0, \eta = 0, k = 0$
while $\eta < 1$ do
    find direction $\boldsymbol{v}(k)$, s.t.
        $\boldsymbol{v}(k) = \arg\max_{\boldsymbol{v}\in\mathcal{D}} \langle \boldsymbol{v}, \widehat{\nabla U(\boldsymbol{\lambda}(k))} \rangle$
    $\gamma_k = \min\{\delta, 1 - \eta\}$
    $\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \gamma_k \boldsymbol{v}(k)$, $\eta = \eta + \gamma_k$, $k = k + 1$
return $\boldsymbol{\lambda}(K)$
```

The proof is in App. A. Obj. (10a) contains two layers of expectations and a different D-optimal design objective from [1]. This is a consequence of the Gaussianity and heterogeneity of sources. In turn, this also requires a different argument in establishing the continuous DR-submodularity.

B. Algorithm Overview

We follow the Frank-Wolfe variant for monotone continuous DR-submodular function maximization by Bian et al. [4] and Liu et al. [1], but deviate in estimating the gradients of objective U. The proposed algorithm is summarized in Alg. 1. **Frank-Wolfe Variant.** Starting from $\boldsymbol{\lambda}(0) = \mathbf{0}$, FW iterates:

$$\boldsymbol{v}(k) = \arg\max_{\boldsymbol{v}\in\mathcal{D}} \langle \boldsymbol{v}, \widehat{\nabla U(\boldsymbol{\lambda}(k))} \rangle, \tag{14a}$$

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \gamma \boldsymbol{v}(k), \tag{14b}$$

where $\widehat{\nabla U(\cdot)}$ is an estimator of the gradient ∇U, and γ is an appropriate stepsize. We will further discuss how to estimate the gradient in Sec. IV-C. This algorithm achieves a $1 - \frac{1}{e}$ approximation guarantee, characterized by:

Theorem 2. *Let $\lambda_{\text{MAX}} = \max_{\boldsymbol{\lambda}\in\mathcal{D}} \|\boldsymbol{\lambda}\|_1$. Then, for any $0 < \epsilon_0, \epsilon_1 < 1$, there exists $K = O(\frac{\epsilon_0}{P_{\texttt{TOT}}(|\mathcal{S}|-1)}\epsilon_1)$, $n' = O(\lambda_{\text{MAX}} T + \ln\frac{1}{\epsilon_1})$, $N_1 = N_2 = \Omega(\sqrt{\ln\frac{P_{\texttt{TOT}}K}{\epsilon_0} \cdot (n'+1)TK})$, s.t., the FW variant algorithm terminates in K iterations, and uses n' terms in the sum, N_1 samples for $\boldsymbol{n}$, and N_2 samples for $\boldsymbol{X}$ in estimator* (17)*. Thus, with probability greater than $1 - \epsilon_0$, the output solution $\boldsymbol{\lambda}(K) \in \mathcal{D}$ of Alg. 1 satisfies:*

$$U(\boldsymbol{\lambda}(K)) \geq (1 - e^{\epsilon_1 - 1}) \max_{\boldsymbol{\lambda}\in\mathcal{D}} U(\boldsymbol{\lambda}) - \epsilon_2, \tag{15}$$

where ϵ_2 determined by ϵ_0, ϵ_1 and network parameters: $\epsilon_2 = (T^2 P_{\texttt{TOT}} \lambda_{\text{MAX}}^2 + 2\lambda_{\text{MAX}})\frac{1}{K}\max_{\ell\in\mathcal{L}, s\in\mathcal{S}} \log\left(1 + \frac{\lambda_{\text{MAX}}(\boldsymbol{\Sigma}_0^\ell)c_s^2}{\sigma_{s,t}^2}\right) > 0$, $c_s = 4\sqrt{\lambda_{\text{MAX}}(\boldsymbol{\Sigma}_s)}\sqrt{d} + 2\sqrt{\lambda_{\text{MAX}}(\boldsymbol{\Sigma}_s)}\sqrt{\log\frac{1}{\delta}}$, and $\delta = O\left(\frac{\epsilon_0}{P_{\texttt{TOT}}K|\mathcal{S}|n'}\right)$.

The proof is in App. C. Our algorithm is based on the Frank-Wolfe variant from Bian et al. [4]. Similar to Thm. 2 in [1], our guarantee involves gradient estimation through truncating and sampling, due to Poisson arrivals (see Asm. 1). However, incorporating Gaussian sources, we need to also sample from the Gaussian distribution to ensure a polynomial-time estimator. This requires combining a sub-Gaussian norm

[1] This is added to ensure the non-negativity of the objective, which is needed to state guarantees in terms of an approximation ratio (c.f. Thm. 2) [1].

bound [29] with the aforementioned truncating and sampling techniques.

C. Gradient Estimation

We describe here how to produce an unbiased, polynomial-time estimator of our gradient $\widehat{\nabla U}$, which is accessed by Eq. (14a). There are three challenges in computing the true gradient (12): (a) the outer sum involves infinite summation over $n_s^\ell \in \mathbb{N}$; (b) the outer expectation involves an exponential sum in $|\mathcal{S}| - 1$; and (c) the inner expectation involves an exponential sum in $\|\boldsymbol{n}^\ell\|$. The last arises from Gaussian sources, which differs from gradient estimation in [1]; this requires the use of a different bound (see Lem. 7 in the Appendix), as well as a decoupling argument (as features and arrivals are jointly distributed).

To address these challenges, we (a) truncate the infinite summation while maintaining the quality of estimation through a Poisson tail bound:

$$\mathrm{HEAD}_{s,t}^p(n') = \sum_{n=0}^{n'} \Delta_s^\ell(\boldsymbol{\lambda}^\ell, n) \cdot \mathbf{P}[n_s^\ell = n] \cdot T, \tag{16}$$

where $t = t^\ell$, l is the last node of p, $\Delta_s^\ell(\boldsymbol{\lambda}^\ell, n)$ is defined in Eq. (13), and n' is the truncating parameter. We then (b) sample $\boldsymbol{n}^\ell$ (N_1 samples) according to the Poisson distribution, parameterized by $\boldsymbol{\lambda}^\ell T$; and (c) sample $\boldsymbol{X}^\ell$ (N_2 samples) according to the Gaussian distribution. When $n' \geq \lambda_s^\ell T$, we estimate the gradient by polynomial-time sampling:

$$\widehat{\frac{\partial U}{\partial \lambda_{s,t}^p}} = \sum_{n=0}^{n'} \widehat{\Delta_s^\ell(\boldsymbol{\lambda}^\ell, n)} \cdot \mathbf{P}[n_s^\ell = n] \cdot T, \tag{17}$$

where

$$\widehat{\Delta_s^\ell(\boldsymbol{\lambda}^\ell, n)} = \frac{1}{N_1 N_2} \sum_{j=1}^{N_1} \sum_{k=1}^{N_2} (G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n_s^{\ell,j}=n+1}) - \\ G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n_s^{\ell,j}=n})),$$

$\boldsymbol{n}^j|_{n_s^{\ell,j}=n}$ indicates vector $\boldsymbol{n}^j$ with $n_s^{\ell,j} = n$, and N_1, N_2 are sampling parameters. At each iteration, we generate N_1 samples $\boldsymbol{n}^{\ell,j}$, $j = 1, \ldots, N_1$ of the random vector $\boldsymbol{n}^\ell$ according to the Poisson distribution in Eq. (8), parameterized by the current solution vector $\boldsymbol{\lambda}^\ell T$. Having a sample $\boldsymbol{n}^{\ell,j}$, we could sample N_2 samples $\boldsymbol{X}^{\ell,j,k}$, $k = 1, \ldots, N_2$ of random matrix $\boldsymbol{X}^{\ell,j} = [[\boldsymbol{x}_{s,i}^\ell]_{i=1}^{n_s^{\ell,j}}]_{s\in\mathcal{S}}$ according to the Gaussian distribution in Eq. (9). We bound the distance between the estimated and true gradient as follows:

Lemma 1. *For any $\delta \in (0,1)$, and $n' \geq \lambda_s^\ell T$,*

$$-\gamma \max_{\ell\in\mathcal{L}, s\in\mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_0^\ell) c_s^2}{\sigma_{s,t}^2}) \leq \frac{\partial U}{\partial \lambda_{s,t}^p} - \widehat{\frac{\partial U}{\partial \lambda_{s,t}^p}} \leq \\ \gamma \max_{\ell\in\mathcal{L}, s\in\mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_0^\ell) c_s^2}{\sigma_{s,t}^2}) + \mathbf{P}[n_s^\ell \geq n' + 1] \frac{\partial U}{\partial \lambda_{s,t}^p},$$

with probability greater than $1 - 2 \cdot e^{-\gamma^2 N_1 N_2 / 2T^2 (n'+1)} - |\mathcal{S}| n' \delta - (|\mathcal{S}| - 1)\delta_{s,t}^p$, where $\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_0^\ell)$ is the maximum eigenvalue of matrix $\boldsymbol{\Sigma}_0^\ell$.

The proof is in App. B. Combining this estimated gradient in Eq. (17) with classic Frank-Wolfe variant [4], we propose Alg. (14) and establish Thm. 2.

V. Distributed Algorithm

Implementing Alg. (14) in our distributed learning network is hard, as it requires the full knowledge of the network. We thus present our distributed algorithm for solving Prob. (10). The algorithm performs a *primal dual gradient algorithm* over a modified Lagrangian to effectively find direction $\boldsymbol{v}$, defined in Eq. (14a), in a distributed fashion. The linearity of Eq. (14a) ensures convergence, while Thm. 2 ensures the aggregate utility attained in steady state is within an $1 - \frac{1}{e}$ factor from the optimal.

A. Algorithm Overview

Solving Prob. (10) in a distributed fashion requires decentralizing Eqs. (14a) and (14b). Decentralizing the latter is easy, as Eq. (14b) can be executed across sources via:

$$\lambda_{s,t}^p(k+1) = \lambda_{s,t}^p(k) + \gamma v_{s,t}^p(k), \tag{18}$$

across all $s \in \mathcal{S}$, and for all $t \in \mathcal{T}$, $p \in \mathcal{P}_{s,t}$. We thus turn our attention to decentralizing Eq. (14a).

Eq. (14a) is a linear program. Standard primal-dual distributed algorithms (see, e.g., [7], [9]) typically require strictly concave objectives, as they otherwise would yield to harmonic oscillations and not converge to an optimal point [22] in linear programs. An additional challenge arises from the multicast constraints in Eqs. (4) and (6): the $\max$ function is non-differentiable. The maximum could be replaced by a set of multiple inequality constraints, but this approach does not scale well, introducing a new dual variable per additional constraint.

To address the first challenge, we follow Feijer and Paganini [22] and replace constraints of the form $u \leq 0$ with $\phi(u) \leq 0$, where $\phi(u) = e^u - 1$. In order to obtain a scalable differentiable Lagrangian, we use the approach in [7], [9]: we replace the multicast constraints (4) and (6) by

$$\sum_{s\in\mathcal{S}, t\in\mathcal{T}} \left(\sum_{p\in\mathcal{P}_{s,t}: e\in p} (v_{s,t}^p)^\theta \right)^{\frac{1}{\theta}} \leq \mu^e, \tag{19}$$

for each link $e \in \mathcal{E}$, and

$$\left(\sum_{p\in\mathcal{P}_{s,t}: e\in p} (v_{s,t}^p)^\theta \right)^{\frac{1}{\theta}} \leq \lambda_{s,t}, \tag{20}$$

for each source $s \in \mathcal{S}$ and each type $t \in \mathcal{T}$. Note that this is tantamount to approximating $\|\cdot\|_\infty$ with $\|\cdot\|_\theta$. Combining these two approaches together, the Lagrangian for the modified problem is:

$$L(\boldsymbol{v}, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{u}) = \langle \boldsymbol{v}, \widehat{\nabla U(\boldsymbol{\lambda})} \rangle - \sum_{e\in\mathcal{E}} q_e (e^{g_e(\boldsymbol{v})} - 1) - \sum_{s\in\mathcal{S}, t\in\mathcal{T}} \\ r_{s,t}(e^{g_{s,t}}(\boldsymbol{v}) - 1) - \sum_{s\in\mathcal{S}, t\in\mathcal{T}} \sum_{p\in\mathcal{P}_{s,t}} u_{s,t}^p (e^{g_{s,t}^p(\boldsymbol{v})} - 1),$$

where

$$g_e(\boldsymbol{v}) = \sum_{s\in\mathcal{S},t\in\mathcal{T}} \left(\sum_{p\in\mathcal{P}_{s,t}:e\in p} (v^p_{s,t})^\theta \right)^{\frac{1}{\theta}} - \mu^e,$$

$$g_{s,t}(\boldsymbol{v}) = \sum_{p\in\mathcal{P}_{s,t}} \left(\sum_{p\in\mathcal{P}_{s,t}:e\in p} (v^p_{s,t})^\theta \right)^{\frac{1}{\theta}} - \lambda_{s,t},$$

$$g^p_{s,t}(\boldsymbol{v}) = -v^p_{s,t},$$

and $\boldsymbol{q} = [q_e]_{e\in\mathcal{E}}$, $\boldsymbol{r} = [r_{s,t}]_{s\in\mathcal{S},t\in\mathcal{T}}$, and $\boldsymbol{u} = [u^p_{s,t}]_{s\in\mathcal{S},t\in\mathcal{T},p\in\mathcal{P}_{s,t}}$ are non-negative dual variables. Intuitively, L penalizes the infeasibility of network constraints.

We apply a primal dual gradient algorithm over this modified Lagrangian to decentralize Eq. (14a). In particular, at iteration $\tau + 1$, the primal variables are adjusted via gradient ascent:

$$v^p_{s,t}(\tau+1) = v^p_{s,t}(\tau) + m^p_{s,t}\nabla_{v^p_{s,t}} L(\tau), \tag{21}$$

and the dual variables are adjusted via gradient descent:

$$q_e(\tau+1) = q_e(\tau) - k_e(\nabla_{q_e} L(\tau))^+_{q_e(\tau)}, \tag{22a}$$

$$r_{s,t}(\tau+1) = r_{s,t}(\tau) - h_{s,t}(\nabla_{r_{s,t}} L(\tau))_{q_e(\tau)})^+_{r_{s,t}(\tau)}, \tag{22b}$$

$$u^p_{s,t}(\tau+1) = u^p_{s,t}(\tau) - w^p_{s,t}(\nabla_{u^p_{s,t}} L(\tau))_{q_e(\tau)})^+_{u^p_{s,t}(\tau)}, \tag{22c}$$

for each edge $e \in \mathcal{E}$, source $s \in \mathcal{S}$, type $t \in \mathcal{T}$, and path $p \in \mathcal{P}_{s,t}$, where $k_e > 0$, $h_{s,t} > 0$, $w^p_{s,t} > 0$, and $m^p_{s,t} > 0$ are stepsize for q_e, $r_{s,t}$, $u^p_{s,t}$ and $v^p_{s,t}$, respectively, and $(y)^+_x = \begin{cases} y, & x > 0, \\ \max(y,0), & x \leq 0. \end{cases}$ These operations can indeed be distributed across the network, as we describe in Sec. V-B. The following theorem states the convergence of this modified primal dual gradient algorithm, according to Thm. 11 in [22]:

Theorem 3. *The trajectories of the modified primal–dual gradient algorithm (Eqs.* (21) *and* (22)*), with constant stepsize, converge to* $\boldsymbol{v}^*_\theta$*. The* $\boldsymbol{v}^*_\theta$ *is an optimum of Prob.* (14a) *over* $\mathcal{D}_\theta$*, where* $\mathcal{D}_\theta$ *is* $\mathcal{D}$ *with* (4) *replaced by* (19)*.*

For $\boldsymbol{v}^*$ be the optimum of Eq. (14a), $\lim_{\theta\to\infty} \boldsymbol{v}^*_\theta \to \boldsymbol{v}^*$, as $\mathcal{D}_\theta \to \mathcal{D}$. Thus, our distributed algorithm preserves a $1 - \frac{1}{e}$ approximation factor from the optimal objective value as stated in Thm. 2, for large enough θ.

B. Distributed FW Implementation Details

We conclude by giving the full implementation details of the distributed FW algorithm and, in particular, the primal dual steps, describing the state maintained by every node, the messages exchanged, and the constituent state adaptations. Starting from $\boldsymbol{\lambda}(0) = \boldsymbol{0}$, the algorithm iterates over:

1) Each source node $s \in \mathcal{S}$ finds direction $v^p_{s,t}(k)$ for all $t \in \mathcal{T}$ and $p \in \mathcal{P}_{s,t}$ by the primal dual gradient algorithm.
2) Each source node s updates $\lambda^p_{s,t}(k+1)$ using $v^p_{s,t}(k)$ for all $t \in \mathcal{T}$, and $p \in \mathcal{P}_{s,t}$ by executing Eq. (18).

Algorithm 2: Distributed Frank-Wolfe Variant

```
Input: $U : \mathcal{D} \to \mathbb{R}_+$, $\mathcal{D}$, stepsize $\delta \in (0,1]$.
$\boldsymbol{\lambda}^0 = 0, \eta = 0, k = 0$
foreach source $s \in \mathcal{S}$ do
    while $\eta < 1$ do
        find direction $v^p_{s,t}(k)$ by Alg. 3
        $\gamma_k = \min\{\delta, 1-\eta\}$
        $\lambda^p_{s,t}(k+1) = \lambda^p_{s,t}(k) + \gamma v^p_{s,t}(k)$, $\eta = \eta + \gamma_k$,
          $k = k + 1$
return $\boldsymbol{\lambda}(K)$
```

Algorithm 3: Primal Dual Gradient Algorithm

```
Input: Rates $\boldsymbol{\lambda}$.
Output: Directions $\boldsymbol{v}$.
Initialize direction $\boldsymbol{v}(0) = \boldsymbol{0}$, dual variables
  $\boldsymbol{q}(0), \boldsymbol{r}(0), \boldsymbol{u}(0) = \boldsymbol{0}$.
foreach learner $\ell \in \mathcal{L}$ do
    Send control messages carrying $\widehat{\nabla_{\lambda^p_{s,t}} U(\boldsymbol{\lambda}^\ell)}$
      calculated by Eq. (17) downstream over $p$.
for $\tau = 1, 2, ...$ do
    foreach source $s \in \mathcal{S}$ do
        Generate features carrying $v^p_{s,t}$ upstream.
    foreach edge $e \in \mathcal{E}$ do
        Calculate $v^e_{s,t}$ using fetched $v^p_{s,t}$ by Eq. (I.2).
    foreach learner $\ell \in \mathcal{L}$ do
        Send control messages downstream and collect
          $q_e$ and $v^e_{s,t}$ from traversed edges.
    foreach edge $e \in \mathcal{E}$ do
        Update $q_e$ using calculated $v^e_{s,t}$ by Eq. (I.3).
    foreach source $s \in \mathcal{S}$ do
        Update $r_{s,t}$ using maintained $v^p_{s,t}$ by Eq. (I.4).
        Update $u^p_{s,t}$ using maintained $v^p_{s,t}$ by Eq. (I.5).
        Update $v^p_{s,t}$ using received $\widehat{\nabla_{\lambda^p_{s,t}} U(\boldsymbol{\lambda}^\ell)}$, $q_e$,
          $v^e_{s,t}$, and maintained $v^p_{s,t}$ by Eq. (I.1).
return $v^p_{s,t}$ from each source $s$
```

We describe the first step in more detail. Every edge $e \in \mathcal{E}$ maintains (a) Lagrange multiplier q_e, and (b) auxiliary variable $v^e_{s,t}$ for all $s \in \mathcal{S}$, $t \in \mathcal{T}$. Every source $s \in \mathcal{S}$ maintains (a) direction $v^p_{s,t}$ and (b) Lagrange multipliers $u^p_{s,t}$, for all $t \in \mathcal{T}$, $p \in \mathcal{P}_{s,t}$, and $r_{s,t}$, for all $t \in \mathcal{T}$. The algorithm initializes all above variables by $\boldsymbol{0}$. Given the rates $\boldsymbol{\lambda}^\ell$, each learner estimates the gradient $\widehat{\nabla_{\lambda^p_{s,t}} U(\boldsymbol{\lambda}^\ell)}$ by Eq. (17). Control messages carrying $\widehat{\nabla_{\lambda^p_{s,t}} U(\boldsymbol{\lambda}^\ell)}$ are generated and propagated over the path p in the reverse direction to sources. Note that this algorithm is a synchronous algorithm where information needs to be exchanged within a specified intervals. Thus, the algorithm proceeds as follows during iteration $\tau + 1$.

1) When feature $\boldsymbol{x}$ is generated from source $s \in \mathcal{S}$, it is propagated over the path p to learner carrying direction $v^p_{s,t}$. Every time it traverses an edge $e \in \mathcal{E}$, edge e

$$v^p_{s,t}(\tau+1) = v^p_{s,t}(\tau) + m^p_{s,t}\left(\nabla_{\lambda^p_{s,t}}U(\boldsymbol{\lambda}) - \textstyle\sum_{e\in p} q_e(\tau)\cdot\, e^{\sum_{s'\in\mathcal{S},t'\in\mathcal{T}}(v^e_{s',t'}(\tau))^{\frac{1}{\theta}}-\mu^e}(v^e_{s,t}(\tau))^{\frac{1-\theta}{\theta}}(v^p_{s,t}(\tau))^{\theta-1} - \right. \tag{I.1}$$

$$\left. r_{s,t}e^{\left(\sum_{p\in\mathcal{P}_{s,t}}(v^p_{s,t}(\tau))^\theta\right)^{\frac{1}{\theta}}-\lambda_{s,t}}\left(\textstyle\sum_{p\in\mathcal{P}_{s,t}}(v^p_{s,t}(\tau))^\theta\right)^{\frac{1-\theta}{\theta}}(v^p_{s,t}(\tau))^{\theta-1} + u^p_{s,t}\exp(-v^p_{s,t}(\tau))\right).$$

$$v^e_{s,t}(\tau) = \textstyle\sum_{p\in\mathcal{P}_{s,t}:e\in p}(v^p_{s,t}(\tau))^\theta. \tag{I.2}$$

$$q_e(\tau+1) = q_e(\tau) + k_e\big(e^{\sum_{s\in\mathcal{S},t\in\mathcal{T}}(v^e_{s,t}(\tau))^{\frac{1}{\theta}}-\mu^e}-1\big)^+_{q_e(\tau)}. \tag{I.3}$$

$$r_{s,t}(\tau+1) = r_{s,t}(\tau) + h_{s,t}\left(e^{\left(\sum_{p\in\mathcal{P}_{s,t}}(v^p_{s,t}(\tau))^\theta\right)^{\frac{1}{\theta}}-\lambda_{s,t}} - 1\right)^+_{r_{s,t}(\tau)}. \tag{I.4}$$

$$u^p_{s,t}(\tau+1) = u^p_{s,t}(\tau) + w^p_{s,t}\left(e^{-v^p_{s,t}(\tau)}) - 1\right)^+_{u^p_{s,t}(\tau)}. \tag{I.5}$$

TABLE I: Expanding primal and dual steps in Eqs. (21) and (22), so that we can execute FW algorithm distributively.

fetches $v^p_{s,t}$.

2) After fetching all $v^p_{s,t}$, each edge $e \in \mathcal{E}$ calculates the auxiliary variables $v^e_{s,t}(\tau)$ for all $s \in \mathcal{S}$ and $t \in \mathcal{T}$ by executing (I.2).
3) Learner $\ell \in \mathcal{L}$ generates a control message, sent over path p in the reverse direction until reaching the source. When traversing edge $e \in \mathcal{E}$, the control message collects q_e and $v^e_{s,t}$. The source obtains these q_e and $v^e_{s,t}$.
4) After receiving all control messages, the edge $e \in \mathcal{E}$ updates the Lagrangian multiplier $q_e(\tau+1)$ using calculated $v^e_{s,t}$ by executing (I.3).
5) Upon obtaining $\nabla_{\lambda^p_{s,t}}\widehat{U(\boldsymbol{\lambda}^\ell)}$, q_e and $v^e_{s,t}$, the source updates the Lagrangian multiplier $r_{s,t}(\tau+1)$ by executing Eq. (I.4), for all $t \in \mathcal{T}$, updates $u^p_{s,t}(\tau+1)$ by executing Eq. (I.5), for all $t \in \mathcal{T}$, $p \in \mathcal{P}_{s,t}$, and updates the direction $v^p_{s,t}(\tau+1)$ using the received $\nabla_{\lambda^p_{s,t}}\widehat{U(\boldsymbol{\lambda}^\ell)}$ q_e, $v^e_{s,t}$ by executing Eq. (I.1), for all $t \in \mathcal{T}$, $p \in \mathcal{P}_{s,t}$.

The above steps are summarized in Alg. 2. This implementation is indeed in a decentralized form: the updates happening on sources and edges require only the knowledge of entities linked to them.

VI. Projected Gradient Ascent

We can also solve Prob. (10) by *projected gradient ascent* (PGA) [30]. Decentralization reduces then to a primal-dual algorithm [7], [22] over a strictly convex objective, which is easier than the FW variant we studied; however, PGA comes with a worse approximation guarantee. We briefly outline this below. Starting from $\boldsymbol{\lambda}(0) = \mathbf{0}$, PGA iterates over:

$$\boldsymbol{v}(k) = \boldsymbol{\lambda}(k) + \gamma\widehat{\nabla U(\boldsymbol{\lambda}(k)})) \tag{23a}$$

$$\boldsymbol{\lambda}(k+1) = \Pi_{\mathcal{D}}(\boldsymbol{v}(k)) \tag{23b}$$

where $\widehat{\nabla U(\cdot)}$ is an estimator of the gradient ∇U, γ is the stepsize, and $\Pi_{\mathcal{D}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}\in\mathcal{D}}(\boldsymbol{y}-\boldsymbol{x})^2$ is the orthogonal projection. Our gradient estimator in Sec. IV-C would again be used here to compute $\widehat{\nabla U(\boldsymbol{\lambda}^k})$. Note that, to achieve the same quality of gradient estimator, PGA usually takes a longer time compared to the FW algorithm. This comes from larger $\lambda^\ell_s(k)$, thus, larger truncating parameter n', during the iteration k (see also Sec. VII-A for how we set algorithm parameters). Furthermore, PGA comes with a worse approximation guarantee compared to the FW algorithm, namely, $1/2$ instead of $1-1/e \approx 0.63$; this would follow by combining the guarantee in [30] with the gradient estimation bounds in Sec. IV-C. Similar to distributed FW, we can easily decentralize Eq. (23a). Eq. (23b) has a strictly convex objective, so we can directly decentralize it through a standard primal-dual algorithm with approximated multicast link capacity constraints, as in Eq. (19). Convergence then is directly implied by Thm. 5 in [22].

TABLE II: Graph Topologies and Experiment Parameters

Graph	$\lvert V\rvert$	$\lvert E\rvert$	μ^e	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{T}\rvert$	U_{DFW}	U_{DPGA}
				synthetic topologies				
ER	100	1042	5-10	5	10	3	351.8	357.3
BT	341	680	5-10	5	10	3	163.3	180.6
HC	128	896	5-10	5	10	3	320.4	343.7
star	100	198	5-10	5	10	3	187.1	206.0
grid	100	360	5-10	5	10	3	213.6	236.9
SW	100	491	5-10	5	10	3	269.5	328.4
				real backbone networks				
GEANT	22	66	5-8	3	3	2	116.4	117.4
Abilene	9	26	5-8	3	3	2	141.3	139.6
Dtelekom	68	546	5-8	3	3	2	125.1	142.3

VII. Numerical Evaluation

A. Experimental Setup

Topologies. We perform experiments over five synthetic graphs, namely, Erdős-Rényi (ER), balanced tree (BT), hypercube (HC), grid_2d (grid), and small-world (SW) [31], and three backbone network topologies: Deutsche Telekom (DT), GEANT, and Abilene [32]. The graph parameters of different topologies are shown in Tab. II.

Network Parameter Settings. For each network, we uniformly at random (u.a.r.) select $|\mathcal{L}|$ learners and $|\mathcal{S}|$ data sources. Each edge $e \in \mathcal{E}$ has a link capacity μ^e and types $\mathcal{T}$ as indicated in Tab. II. Sources generate feature vectors with dimension $d = 100$ within data acquisition time $T = 1$. Each source s generates the data $(\boldsymbol{x}, y)$ of type t label with rate $\lambda_{s,t}$, uniformly distributed over [5,8]. Features $\boldsymbol{x}$ from source s are generated following a zero mean Gaussian distribution, whose covariance is generated as follows. First, we separate features into two classes: well-known and poorly-known. Then, we set the corresponding Gaussian covariance (i.e., the diagonal elements in $\boldsymbol{\Sigma}_s$) to low (uniformly from 0 to 0.01) and high (uniformly from 10 to 20) values, for well-known and poorly-known features, respectively. Source

s labels y of type t using ground-truth models, as discussed below, with Gaussian noise, whose variance $\sigma_{s,t}$ is chosen u.a.r. (uniformly at random) from 0.5 to 1. For each source, the paths set consists of the shortest paths between the source and every learner in $\mathcal{L}$. Each learner has a target model $\boldsymbol{\beta}_{t^\ell}$, which is sampled from a prior normal distribution as follows. Similarly to sources, we separate features into interested and indifferent. Then, we set the corresponding prior covariance (i.e., the diagonal elements in $\boldsymbol{\Sigma}_0^\ell$) to low (uniformly from 0 to 0.01) and high (uniformly from 1 to 2) values, and set the corresponding prior mean to 1 and 0, for interested and indifferent features, respectively.

Algorithms. We implement our algorithm and several competitors. First, there are four centralized algorithms:

- `MaxTP`: This maximizes the aggregate incoming traffic rates (throughput) of learners, i.e.:

$$\max_{\boldsymbol{\lambda}\in\mathcal{D}} : U_{\texttt{MaxTP}}(\boldsymbol{\lambda}) = \sum_{\ell\in\mathcal{L}}\sum_{s\in\mathcal{S}} \lambda_s^\ell. \tag{24}$$

- `MaxFair`: This maximizes the aggregate α-fair utilities [7] of the incoming traffic at learners, i.e.:

$$\max_{\boldsymbol{\lambda}\in\mathcal{D}} : U_{\texttt{MaxFair}}(\boldsymbol{\lambda}) = \sum_{\ell\in\mathcal{L}}(\sum_{s\in\mathcal{S}}\lambda_s^\ell)^{1-\alpha}/(1-\alpha). \tag{25}$$

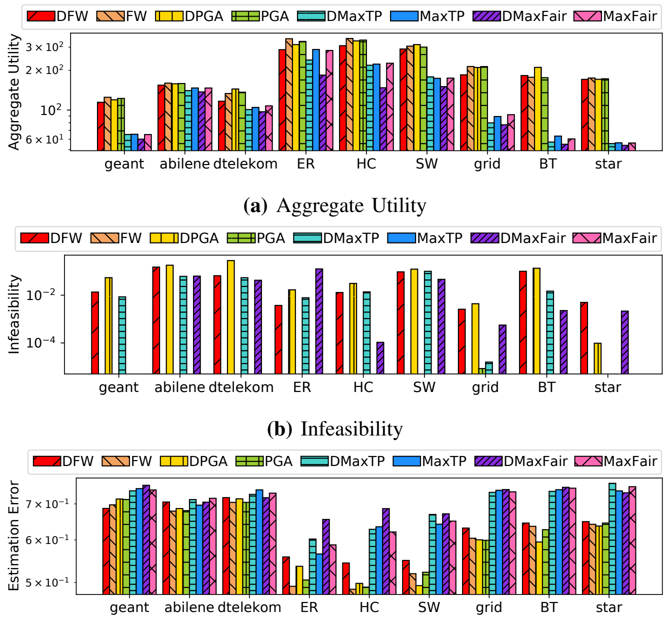
  We set $\alpha = 2$.
- `FW`: This is Alg. (14), as proposed in Sec. IV.
- `PGA`: This is the algorithm we proposed in Sec. VI.

We also implement their corresponding distributed versions: `DMaxTP`, `DMaxFair`, `DFW` (algorithm in Sec. V-B), and `DPGA` (see Sec. VI). The objectives of `MaxTP` Eq. (24) and `MaxFair` Eq. (25) are linear and strictly concave, respectively. The modified primal dual gradient algorithm, used in `DFW`, and basic primal dual gradient algorithm, used in `DPGA`, directly apply to `DMaxTP` and `DMaxFair`, respectively.

Algorithm Parameter Settings. We run `FW`/`DFW` and `PGA`/`DPGA` for $K = 50$ iterations, i.e. stepsize $\gamma = 0.02$, with respect to the outer iteration. In each iteration, we estimate the gradient according to Eq. (17) with sampling parameters $N_1 = 50$, $N_2 = 50$, and truncating parameters $n' = \max\{\lceil 2\max_{\ell,s}\lambda_s^\ell T\rceil, 10\}$, where λ_s^ℓ is given by the current solution. We run the inner primal-dual gradient algorithm for 1000 iterations and set parameter $\theta = 10$ when approximating the max function via Eqs. (19) and (20). We compare the performance metrics (Aggregate utility and Infeasibility, defined in Sec. VII-B), between centralized and distributed versions of each algorithm under different stepsizes, and we choose the best stepsize for distributed primal-dual algorithms. We further discuss the impact of the stepsizes in Sec. VII-C.

B. Performance Metrics

To evaluate the performance of the algorithms, we use the *Aggregate Utility*, defined in Eq. (10a) as one metric. Note that as the aggregate utility involves a summation with infinite support, we thus need to resort to sampling to estimate it; we set $N_1 = 100$ and $N_2 = 100$. Also, we define an *Estimation Error* to measure the model learning/estimation quality. Formally, it is defined as: $\frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}}\frac{\|\hat{\boldsymbol{\beta}}^\ell_{\texttt{MAP}}-\boldsymbol{\beta}^\ell\|}{\|\boldsymbol{\beta}^\ell\|}$, following the equation of MAP estimation Eq. (1). We average over 2500 realizations of the number of data arrived at the learner $\{\boldsymbol{n}^\ell\}_{\ell\in\mathcal{L}}$ and features $\{\boldsymbol{X}^\ell\}_{\ell\in\mathcal{L}}$, and 20 realizations of ground-truth models $\{\boldsymbol{\beta}^\ell\}_{\ell\in\mathcal{L}}$. Finally, we define an *Infeasibility* to measure the feasibility of solutions, as primal dual gradient algorithm used in distributed algorithms does not guarantee feasibility. It averages the total violations of constraints (3)-(6) over the number of constraints.



(a) Aggregate Utility

(b) Infeasibility

(c) Estimation Error

Fig. 2: Aggregate utility, infeasibility and estimation error across networks. `DFW` and `DPGA` perform very well in terms of maximizing the utility and minimizing the estimation error in all networks. The aggregate utilities of `DFW` and `DPGA` are also listed in Tab. II. Furthermore, their performances are close to their centralized versions: `FW` and `PGA`, with an acceptable infeasibility ∼ 0.1.

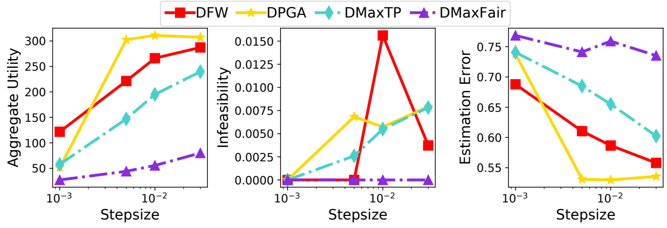
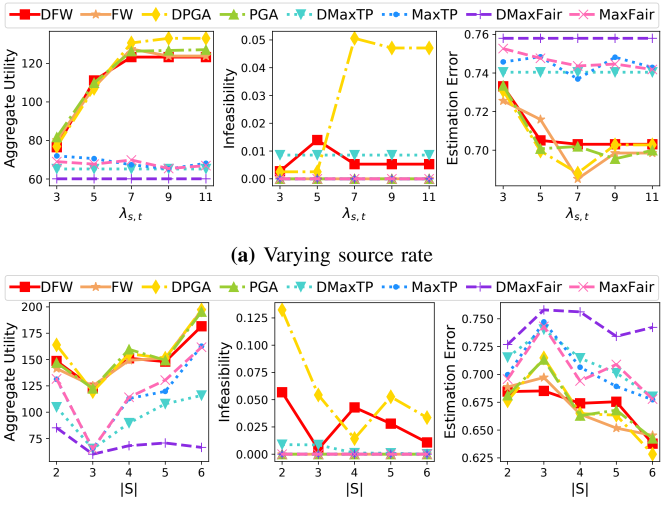


Fig. 3: Stepsize effect on primal dual gradient algorithms over topology `ER`. Larger stepsizes lead to better performance, and `DFW` and `DPGA` are always the best in terms of both utility and estimation error. However, stepsizes above 0.03 lead to numerical instability.

C. Results

Different Topologies. We first compare the proposed algorithms with several baselines in terms of aggregate utility, infeasibility and estimation error over several network topologies, shown in Fig. 2. Our proposed algorithms dramatically



(a) Varying source rate

(b) Varying source set size

Fig. 4: Varying source rates and source set size over GEANT. When increasing source rates and source set sizes, learners receive more data. This leads to higher aggregate utility, and lower estimation error. Our algorithms, DFW and DPGA, stay close to their centralized versions (FW and PGA) and outperform competitors in both metrics, with a small change in feasibility.

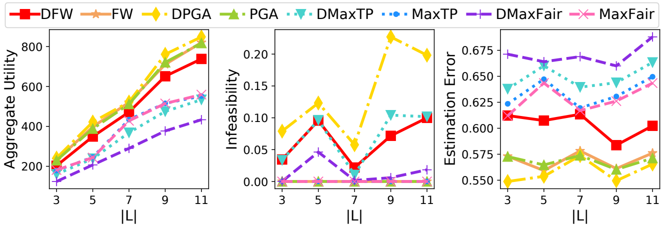


Fig. 5: Varying learner set size over topology SW. The aggregate utility increases, while the estimation error remains essentially unchanged, as the number of learners increases. DFW and DPGA again stay close to their centralized versions and outperform competitors.

outperform all competitors, and our distributed algorithms perform closed to their corresponding centralized algorithms (c.f. Thm. 3). All of the distributed algorithms have a low infeasibility, which is less than 0.1. Such violations over the constraints are expected by primal-dual gradient algorithms, since they employ soft constraints.

Effect of Stepsizes. We study the effect of the stepsizes in the primal dual gradient algorithms used in the distributed algorithms DFW, DPGA, DMaxTP and DMaxFair over ER, shown in Fig. 3. When the algorithms are stable, larger stepsizes achieve better performance w.r.t. both aggregate utility and estimation error, while worse performance with respect to infeasibility. However, if the stepsize is too large, the algorithms do not converge and become numerically unstable. It is crucial to choose an appropriate stepsize for better performance, while maintaining convergence. In all convergent cases, DFW and DPGA outperform competitors.

Varying Source Rates and Source Set Size. Next, we evaluate how algorithm performance is affected by varying the (common) source rates $\lambda_{s,t}$ over topology GEANT. As shown in Fig. 4a, when source rates increase, the aggregate utility first increases very fast and then tapers off. Higher source rates indicate more data received at the learners, hence the greater utility. However, due to DR-submodularity, the marginal gain decreases as the number of sources increases. Furthermore, under limited bandwidth, if link capacities saturate, there will be no further utility increase. The same interpretation applies to the estimation error. We observe similarly changing patterns when varying the source set size $|\mathcal{S}|$ over topology GEANT, shown in Fig. 4b, since more sources also indicates learners receive more data. However, the curve changes are not as smooth as those for increasing the source rates. This is because varying source sets also changes the available paths, corresponding link bandwidth utilization, indexes of well-known features, etc. Overall, we observe that our algorithms, DFW and DPGA, stay close to their centralized versions (FW and PGA) and outperform competitors in both metrics, with a small change in feasibility.

Varying Learner Set Size. Finally, we evaluate the effect of the learner set size $|\mathcal{L}|$ over topology SW. Fig. 5 shows that as the number of learners increases so does the aggregate utility, while the estimation errors essentially remain the same. With multicast transmissions, increasing the number of learners barely affects the amount of data received by each learner. Thus, the aggregate utility increases as expected, while the *average* utility per learner (the aggregate utility divided by the number of learners) and, consequently, the estimation error hardly change, when more learners are in the network. Again, our algorithms, DFW and DPGA, stay close to their centralized versions (FW and PGA) and outperform competitors.

VIII. Conclusion

We generalize the experimental design networks by considering Gaussian sources and multicast transmissions. A polytime distributed algorithm with $1 - 1/e$ approximation guarantee is proposed to facilitate heterogeneous model learning across networks. One limitation of our distributed algorithm is its synchronization. It is natural to extend the model to an asynchronous setting, which better resembles the reality of large networks. One possible solution is that sources and links compute outdated gradients [6]. Another interesting direction is to estimate gradients through shadow prices [33], [34], instead of sampling. Furthermore, how a model trained by one learner benefits other training tasks in experimental design networks is also a worthwhile topic to study. The authors have provided public access to their code and data.[2]

Acknowledgment

The authors gratefully acknowledge support from the National Science Foundation (grants 1718355, 2106891, 2107062, and 2112471).

[2] https://github.com/neu-spiral/DistributedNetworkLearning

References

[1] Y. Liu, Y. Li, L. Su, E. Yeh, and S. Ioannidis, "Experimental design networks: A paradigm for serving heterogeneous learners under networking constraints," in *IEEE INFOCOM 2022*. IEEE, 2022, pp. 210–219.

[2] M. Mohammadi and A. Al-Fuqaha, "Enabling cognitive smart cities using big data and machine learning: Approaches and challenges," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 94–101, 2018.

[3] V. Albino, U. Berardi, and R. M. Dangelico, "Smart cities: Definitions, dimensions, performance, and initiatives," *Journal of urban technology*, vol. 22, no. 1, pp. 3–21, 2015.

[4] A. A. Bian, B. Mirzasoleiman, J. Buhmann, and A. Krause, "Guaranteed non-convex optimization: Submodular maximization over continuous domains," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 111–120.

[5] T. Soma and Y. Yoshida, "A generalization of submodular cover via the diminishing return property on the integer lattice," *Advances in neural information processing systems*, vol. 28, 2015.

[6] S. H. Low and D. E. Lapsley, "Optimization flow control. i. basic algorithm and convergence," *IEEE/ACM Transactions on networking*, vol. 7, no. 6, pp. 861–874, 1999.

[7] R. Srikant and T. Başar, *The mathematics of Internet congestion control*. Springer, 2004.

[8] D. S. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3. IEEE, 2005, pp. 1607–1617.

[9] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, "Minimum-cost multicast over coded packet networks," *IEEE Transactions on information theory*, vol. 52, no. 6, pp. 2608–2623, 2006.

[10] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[11] F. Pukelsheim, *Optimal design of experiments*. Society for Industrial and Applied Mathematics, 2006.

[12] T. Horel, S. Ioannidis, and S. Muthukrishnan, "Budget feasible mechanisms for experimental design," in *Latin American Symposium on Theoretical Informatics*. Springer, 2014, pp. 719–730.

[13] Y. Guo, J. Dy, D. Erdogmus, J. Kalpathy-Cramer, S. Ostmo, J. P. Campbell, M. F. Chiang, and S. Ioannidis, "Accelerated experimental design for pairwise comparisons," in *SDM*. SIAM, 2019, pp. 432–440.

[14] N. Gast, S. Ioannidis, P. Loiseau, and B. Roussillon, "Linear regression from strategic data sources," *ACM Transactions on Economics and Computation (TEAC)*, vol. 8, no. 2, pp. 1–24, 2020.

[15] Y. Guo, P. Tian, J. Kalpathy-Cramer, S. Ostmo, J. P. Campbell, M. F. Chiang, D. Erdogmus, J. G. Dy, and S. Ioannidis, "Experimental design under the bradley-terry model." in *IJCAI*, 2018, pp. 2198–2204.

[16] X. Huan and Y. M. Marzouk, "Simulation-based optimal bayesian experimental design for nonlinear systems," *Journal of Computational Physics*, vol. 232, no. 1, pp. 288–317, 2013.

[17] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[18] A. Krause and D. Golovin, "Submodular function maximization." 2014.

[19] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[20] A. Nedić and A. Ozdaglar, "Subgradient methods for saddle-point problems," *Journal of optimization theory and applications*, vol. 142, no. 1, pp. 205–228, 2009.

[21] S. A. Alghunaim and A. H. Sayed, "Linear convergence of primal–dual gradient methods and their performance in distributed optimization," *Automatica*, vol. 117, p. 109003, 2020.

[22] D. Feijer and F. Paganini, "Stability of primal–dual gradient dynamics and applications to network optimization," *Automatica*, vol. 46, no. 12, pp. 1974–1981, 2010.

[23] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

[24] G. Tychogiorgos, A. Gkelias, and K. K. Leung, "A non-convex distributed optimization framework and its application to wireless ad-hoc networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4286–4296, 2013.

[25] A. Mokhtari, H. Hassani, and A. Karbasi, "Decentralized submodular maximization: Bridging discrete and continuous settings," in *International conference on machine learning*. PMLR, 2018, pp. 3616–3625.

[26] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[27] R. G. Gallager, *Stochastic Processes: Theory for Applications*. Cambridge University Press, 2013.

[28] F. P. Kelly, *Reversibility and stochastic networks*. Cambridge University Press, 2011.

[29] A. Rinaldo, "Sub-gaussian vectors and bound for the their norm." 2019. [Online]. Available: https://www.stat.cmu.edu/~arinaldo/Teaching/36709/S19/Scribed_Lectures/Feb21_Shenghao.pdf

[30] H. Hassani, M. Soltanolkotabi, and A. Karbasi, "Gradient methods for submodular maximization," in *NeurIPS*, 2017, pp. 5843–5853.

[31] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *STOC*, 2000.

[32] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.

[33] S. Ioannidis, A. Chaintreau, and L. Massoulié, "Optimal and scalable distribution of content updates over a mobile social network," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 1422–1430.

[34] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, vol. 49, no. 3, pp. 237–252, 1998.

[35] A. G. Akritas, E. K. Akritas, and G. I. Malaschonok, "Various proofs of sylvester's (determinant) identity," *Mathematics and Computers in Simulation*, vol. 42, no. 4-6, pp. 585–593, 1996.

[36] C. L. Canonne, "A short note on poisson tail bounds," 2017. [Online]. Available: http://www.cs.columbia.edu/~ccanonne/files/misc/2017-poissonconcentration.pdf

[37] N. Alon and J. H. Spencer, *The probabilistic method*. John Wiley & Sons, 2004.

[38] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

Appendix A
Proof of Theorem 1

Consider an equivalent data generation process, in which infinite sequences of independent Gaussian variables $\widetilde{\boldsymbol{X}}^{\ell} = [[\boldsymbol{x}_{s,i}^{\ell}]_{i=1}^{\infty}]_{s\in\mathcal{S}}$ from different sources s are received by the learner, but only an initial part of each (as governed by $\boldsymbol{n}^{\ell}$) is observed. Note that the statistics of this process are identical to the actual arrival process, where samples are independent conditioned on $\boldsymbol{n}^{\ell}$. Then, combining the proof of Lem. 1 in [1] using Sylvester's determinant identity [12], [35] and telescoping sum, we get:

Lemma 2. *Function $G^{\ell}(\widetilde{\boldsymbol{X}}^{\ell}, \boldsymbol{n}^{\ell})$ is (a) monotone-increasing and (b) DR-submodular w.r.t. $\boldsymbol{n}^{\ell}$, where $\widetilde{\boldsymbol{X}}^{\ell} = [[\boldsymbol{x}_{s,i}^{\ell}]_{i=1}^{\infty}]_{s\in\mathcal{S}}$.*

Proof. For $\boldsymbol{n} \in \mathbb{N}^{|\mathcal{S}|}$, we have:

$$\begin{aligned}
G(\boldsymbol{X}, \boldsymbol{n}+\boldsymbol{e}_{s'})) - G(\boldsymbol{X}, \boldsymbol{n}) =& \log\det\big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^{\top}}{\sigma_s^2} + \frac{\boldsymbol{x}_{s',n_{s'}+1}\boldsymbol{x}_{s',n_{s'}+1}^{\top}}{\sigma_{s'}^2} + \boldsymbol{\Sigma}_0^{-1}\big) - \log\det\big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^{\top}}{\sigma_s^2} + \boldsymbol{\Sigma}_0^{-1}\big)\\
=& \log\det\big(\boldsymbol{I}_d + \frac{\boldsymbol{x}_{s',n_{s'}+1}\boldsymbol{x}_{s',n_{s'}+1}^{\top}}{\sigma_{s'}^2}\big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^{\top}}{\sigma_s^2} + \boldsymbol{\Sigma}_0^{-1}\big)^{-1}\big)\\
=& \log\det\big(\boldsymbol{I}_1 + \frac{1}{\sigma_{s'}^2}\boldsymbol{x}_{s',n_{s'}+1}^{\top}\boldsymbol{A}(\boldsymbol{n})\boldsymbol{x}_{s',n_{s'}+1}\big)\\
=& \log\big(1 + \frac{1}{\sigma_{s'}^2}\boldsymbol{x}_{s',n_{s'}+1}^{\top}\boldsymbol{A}(\boldsymbol{n})\boldsymbol{x}_{s',n_{s'}+1}\big)
\end{aligned}$$

where $\boldsymbol{A}(\boldsymbol{n}) = \big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^{\top}}{\sigma_s^2} + \boldsymbol{\Sigma}_0^{-1}\big)^{-1}$ and the second last equality follows Sylvester's determinant identity. Then for $\boldsymbol{n} \in \mathbb{N}^{|\mathcal{S}|}$ and $k \in \mathbb{N}$, by telescoping sum, we have

$$\begin{aligned}
&G(\boldsymbol{X}, \boldsymbol{n}+k\boldsymbol{e}_{s'})) - G(\boldsymbol{X}, \boldsymbol{n})\\
=&G(\boldsymbol{X}, \boldsymbol{n}+k\boldsymbol{e}_{s'})) - G(\boldsymbol{X}, \boldsymbol{n}+(k-1)\boldsymbol{e}_{s'})) + \cdots + (G(\boldsymbol{X}, \boldsymbol{n}+\boldsymbol{e}_{s'})) - G(\boldsymbol{X}, \boldsymbol{n}))\\
=&\log\big(1 + \frac{1}{\sigma^2}\boldsymbol{x}_{s',n_{s'}+k}^{\top}\boldsymbol{A}(\boldsymbol{n}+(k-1)\boldsymbol{e}_{s'})\boldsymbol{x}_{s',n_{s'}+k}\big) + \cdots + \log\big(1 + \frac{1}{\sigma^2}\boldsymbol{x}_{s',n_{s'}+1}^{\top}\boldsymbol{A}(\boldsymbol{n})\boldsymbol{x}_{s',n_{s'}+1}\big)
\end{aligned}$$

where $\boldsymbol{A}(\boldsymbol{n}) = \big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^{\top}}{\sigma_s^2} + \boldsymbol{\Sigma}_0^{-1}\big)^{-1}$ and the second last equality follows Sylvester's determinant identity. The monotonicity of G follows because $A(\boldsymbol{n})$ is positive semidefinite. Finally, since the matrix inverse is decreasing over the positive semi-definite order, we have $A(\boldsymbol{n}) \succeq A(\boldsymbol{m})$, $\forall\ \boldsymbol{n}, \boldsymbol{m} \in \mathbb{N}^{|\mathcal{S}|}$, $k \in \mathbb{N}$ and $\boldsymbol{n} \leq \boldsymbol{m}$, which leads to $G(\boldsymbol{X}, \boldsymbol{n}+k\boldsymbol{e}_s) - G(\boldsymbol{X}, \boldsymbol{n}) \geq G(\boldsymbol{X}, \boldsymbol{m}+k\boldsymbol{e}_s) - G(\boldsymbol{X}, \boldsymbol{m})$. □

Armed with this result, we use a truncation argument to prove that taking expectations w.r.t. the (infinite) sequences $\widetilde{\boldsymbol{X}}^{\ell} = [[\boldsymbol{x}_{s,i}^{\ell}]_{i=1}^{\infty}]_{s\in\mathcal{S}}$ preserves submodularity:

Lemma 3. *Function $g^{\ell}(\boldsymbol{n}^{\ell}) = \mathbb{E}_{\boldsymbol{X}^{\ell}}[G(\widetilde{\boldsymbol{X}}^{\ell}, \boldsymbol{n}^{\ell})|\boldsymbol{n}^{\ell}]$ is (a) monotone-increasing and (b) DR-submodular w.r.t. $\boldsymbol{n}^{\ell}$.*

Proof. Let's consider a projection:

$$\Pi_{n_0}(\boldsymbol{n}) = \tilde{\boldsymbol{n}} = [\tilde{n}_s]_{s\in\mathcal{S}},$$

where

$$\tilde{n}_s = \begin{cases} n_s, & \text{if } n_s \leq n_0\\ n_0, & \text{otherwise.}\end{cases}$$

Then, consider a function:

$$G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{n}) = G(\widetilde{\boldsymbol{X}}, \Pi_{n_0}(\boldsymbol{n})) = \begin{cases} G(\widetilde{\boldsymbol{X}}, \boldsymbol{n}), & \text{if } \|\boldsymbol{n}\|_{\infty} \leq n_0\\ G(\widetilde{\boldsymbol{X}}, \Pi_{n_0}(\boldsymbol{n})), & \text{otherwise.}\end{cases}$$

It is easy to verify that $\forall\ \boldsymbol{n}, \boldsymbol{m} \in \mathbb{N}^{|\mathcal{S}|}$ and $\boldsymbol{n} \leq \boldsymbol{m}$, we have $G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{n}+k\boldsymbol{e}_s) - G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{n}) \geq G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{m}+k\boldsymbol{e}_s) - G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{m})$. Thus, $G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{n})$ is DR-submodular w.r.t. $\boldsymbol{n}$.

Consider $g_{n_0}(\boldsymbol{n}) = \mathbb{E}_{\widetilde{\boldsymbol{X}}}[G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{n})|\boldsymbol{n}]$. As an expectation of 'finite $\widetilde{\boldsymbol{X}}$', $g_{n_0}(\boldsymbol{n})$ is still DR-submodular, i.e., $\forall\ \boldsymbol{n}, \boldsymbol{m} \in \mathbb{N}^{|\mathcal{S}|}$, $\boldsymbol{n} \leq \boldsymbol{m}$ and $\forall\ n_0 \in \mathbb{N}$, we have $g_{n_0}(\boldsymbol{n}+k\boldsymbol{e}_s) - g_{n_0}(\boldsymbol{n}) \geq g_{n_0}(\boldsymbol{m}+k\boldsymbol{e}_s) - g_{n_0}(\boldsymbol{m})$. Observe that $\lim_{n_0\to\infty} g_{n_0}(\boldsymbol{n}) = g(\boldsymbol{n})$, $\forall\ \boldsymbol{n}' \in \mathbb{N}^{|\mathcal{S}|}$. In fact, $\forall\ \boldsymbol{n}' \in \mathbb{N}^{|\mathcal{S}|}$, $\exists\ n_0(\boldsymbol{n}') = \|\boldsymbol{n}'\|_{\infty}$, s.t. $g_{n_0}(\boldsymbol{n}') = g(\boldsymbol{n}')$. Furthermore, $\exists n_0 = \max\{\|\boldsymbol{n}\|_{\infty}, \|\boldsymbol{n} +$

$k\boldsymbol{e}_s\|_\infty, \|\boldsymbol{m}\|_\infty, \|\boldsymbol{m}+k\boldsymbol{e}_s\|_\infty\}$, s.t. $g_{n_0}(\boldsymbol{n}) = g(\boldsymbol{n})$, $g_{n_0}(\boldsymbol{n}+k\boldsymbol{e}_s) = g(\boldsymbol{n}+k\boldsymbol{e}_s)$, $g_{n_0}(\boldsymbol{m}) = g(\boldsymbol{m})$, $g_{n_0}(\boldsymbol{m}+k\boldsymbol{e}_s) = g(\boldsymbol{m}+k\boldsymbol{e}_s)$, and $g(\boldsymbol{n}+k\boldsymbol{e}_s) - g(\boldsymbol{n}) \geq g(\boldsymbol{m}+k\boldsymbol{e}_s) - g(\boldsymbol{m})$.

Similarly, we could obtain the monotonicity. □

Then, we establish the positivity of the gradient and non-positivity of the Hessian of U, following Thm. 1 in [1] and utilizing Lem. 3 to prove our Thm. 1.

Proof. By the law of total expectation:

$$U^\ell(\boldsymbol{\lambda}^\ell) = \mathbb{E}[g^\ell(\boldsymbol{n}^\ell)] = \sum_{t=0}^{\infty} \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t\right] \cdot \frac{(\lambda_s^\ell T)^t e^{-\lambda_s^\ell T}}{t!}.$$

Thus the first partial derivatives are:

$$\begin{aligned}\frac{\partial U}{\partial \lambda_{s,t}^p} =& \frac{\partial U^\ell}{\partial \lambda_{s,t}^p} = \sum_{t=0}^{\infty} \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t\right] \cdot (\frac{t}{\lambda_s^\ell} - T)\frac{(\lambda_s^\ell T)^t e^{-\lambda_s^\ell T}}{t!} \\ =& \sum_{t=0}^{\infty}(\mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t+1\right] - \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t\right]) \cdot \frac{(\lambda_s^\ell)^t T^{t+1}}{t!} e^{-\lambda_s^\ell T} \geq 0\end{aligned}$$

by monotonicity of g. Note that, here $t = t^\ell$ and $\ell = p[-1]$. Next, we compute the second partial derivatives $\frac{\partial^2 U}{\partial \lambda_{s,t}^p \partial \lambda_{s',t'}^{p'}}$. It is easy to see that for $\ell \neq \ell'$, we have:

$$\frac{\partial^2 U}{\partial \lambda_{s,t}^p \partial \lambda_{s',t'}^{p'}} = 0.$$

For $\ell = \ell'$, which also implies $t = t'$, and $s = s'$. No matter p equals to p' or not, it holds that

$$\begin{aligned}&\frac{\partial^2 U}{\partial \lambda_{s,t}^p \partial \lambda_{s,t}^{p'}} = \frac{\partial^2 U^\ell}{\partial (\lambda_{s,t}^p)^2} \\ =&(\mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = 1\right] - \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = 0\right]) \cdot (-T^2) e^{-\lambda_s^\ell T} + \sum_{t=1}^{\infty}(\mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t+1\right] - \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t\right]) \cdot \ldots \\ &\quad (\frac{(\lambda_s^\ell)^{t-1} T^{t+1}}{(t-1)!} - \frac{(\lambda_s^\ell)^t T^{t+2}}{t!}) e^{-\lambda_s^\ell T} \\ =& \sum_{t=1}^{\infty}\left((\mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t+1\right] - \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t\right]) - (\mathbb{E}\left[g(\boldsymbol{n})|\ \ n_s^\ell = t\right] - \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t-1\right])\right) \cdot \frac{(\lambda_s^\ell)^{t-1} T^{t+1}}{(t-1)!} e^{-\lambda_s^\ell T} \\ \leq& 0,\end{aligned}$$

by the submodularity of G. For $s \neq s'$, which also implies $p \neq p'$

$$\begin{aligned}\frac{\partial^2 U}{\partial \lambda_{s,t}^p \partial \lambda_{s',t}^{p'}} =& \frac{\partial^2 U^\ell}{\partial \lambda_{s,t}^p \partial \lambda_{s',t}^{p'}} = \sum_{t=0}^{\infty}\sum_{k=0}^{\infty} \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t, n_{s'}^\ell = k\right] \cdot \frac{\partial}{\partial \lambda_s^\ell} \mathrm{P}(n_s^\ell = t) \cdot \frac{\partial}{\partial \lambda_{s'}^\ell} \mathrm{P}(n_{s'}^\ell = k) \\ =& \sum_{t=0}^{\infty}\sum_{k=0}^{\infty}((\mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t+1, n_{s'}^\ell = k+1\right] - \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t, n_{s'}^\ell = k+1\right]) - \ldots \\ &\quad (\mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t+1,\ n_{s'}^\ell = k\right] - \mathbb{E}\left[g(\boldsymbol{n})|n_s^\ell = t, n_{s'}^\ell = k\right])) \cdot \frac{(\lambda_s^\ell)^t (\lambda_{s'}^\ell)^k T^{t+k+2}}{t!k!} e^{-(\lambda_s^\ell + \lambda_{s'}^\ell)T} \\ \leq& 0\end{aligned}$$

also by the submodularity of g. □

Appendix B
Proof of Lemma 1

We know that Poisson has a subexponential tail bound [36], then, after truncating, the HEAD, defined in Eq. (16) is guaranteed to be within a constant factor from the true partial derivative:

Lemma 4 (Lemma 4 in [1]). *For $h(u) = 2\frac{(1+u)\ln(1+u)-u}{u^2}$ and $n' \geq \lambda_s^\ell T$, we have:*

$$\frac{\partial U}{\partial \lambda_{s,t}^p} \geq \mathrm{HEAD}_{s,t}^p(n') \geq (1 - \mathbf{P}[n_s^\ell \geq n'+1]) \frac{\partial U}{\partial \lambda_{s,t}^p}.$$

To quantify the distance between the estimated gradient and HEAD, we first introduce two auxiliary lemmas as follows. By Chernoff bounds described by Thm A.1.16 in [37], we get:

Lemma 5. *If there exists a constant vector $\boldsymbol{c} = [c_s]_{s\in\mathcal{S}} \in \mathbb{R}^{|\mathcal{S}|}$, such that for any $s \in \mathcal{S}$ and $i \leq n_s^\ell$, $\|\boldsymbol{x}_{s,i}^\ell\|_2 \leq c_s$, then:*

$$\left|\widehat{\frac{\partial U}{\partial \lambda_{s,t}^p}} - \mathrm{HEAD}_{s,t}^p\right| \leq \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_0^\ell)c_s^2}{\sigma_{s,t}^2}),$$

with probability greater than $1 - 2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)}$, where $\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_0^\ell)$ is the maximum eigenvalue of matrix $\boldsymbol{\Sigma}_0^\ell$.

Proof. We define

$$X^{j,k}(n) = \frac{G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n_s^{\ell,j}=n+1}) - G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n_s^{\ell,j}=n}) - \Delta_s^\ell(\boldsymbol{\lambda}^\ell, n)}{G_{\mathrm{MAX}}},$$

where $G_{\mathrm{MAX}} = \max_{\ell\in\mathcal{L},s\in\mathcal{S},\boldsymbol{x}\in B(\mathbf{0};c)}(G^\ell(\boldsymbol{x},\boldsymbol{e}_s) - G^\ell(\boldsymbol{x},\mathbf{0})) = \max_{\ell\in\mathcal{L},s\in\mathcal{S},\boldsymbol{x}\in B(\mathbf{0};c)} \log\left(1 + \frac{1}{\sigma_{s,t}^2}\boldsymbol{x}_s^\top \boldsymbol{\Sigma}_{\boldsymbol{\ell}}\boldsymbol{x}_s\right)$. We have $|X^{j,k}(n)| \leq 1$, because:

$$G_{\mathrm{MAX}} \geq (G^\ell(\boldsymbol{x},\boldsymbol{e}_s) - G^\ell(\boldsymbol{x},\mathbf{0})) \geq G(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^j|_{n_s^{\ell,j}=n+1}) - G(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^j|_{n_s^{\ell,j}=n}),$$

for any $\ell \in \mathcal{L}$, $s \in \mathcal{S}$, $\boldsymbol{x} \in B(\mathbf{0};c)$, $n \geq 0$. By Chernoff bounds described by Theorem A.1.16 in [37], we have

$$\mathbf{P}\left[\left|\sum_{j=1}^{N_1}\sum_{k=1}^{N_2}\sum_{n=0}^{n=n'} X^{j,k}(n)\right| > c\right] \leq 2e^{-c^2/2N_1N_2(n'+1)}.$$

Suppose we let $c = \gamma \cdot N_1 N_2/T$, where γ is the step size, then we have

$$\begin{aligned}
\left|\widehat{\frac{\partial U}{\partial \lambda_{s,t}^p}} - \mathrm{HEAD}_{s,t}^p\right| \leq & \left|\sum_{n=0}^{n'}\sum_{j=1}^{N_1}\sum_{k=1}^{N_2} \frac{G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n_s^{\ell,j}=n+1}) - G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n_s^{\ell,j}=n}) - \Delta_s^\ell(\boldsymbol{\lambda}^\ell, n)}{N_1 N_2} T\right| \\
= & \left|\sum_{t=0}^{n'}\sum_{j=1}^{N_1}\sum_{k=1}^{N_2} X^{j,k}(n)\right| \cdot \frac{T}{N_1 N_2} \cdot G_{\mathrm{MAX}} \leq \gamma \cdot G_{\mathrm{MAX}} \\
= & \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S},\boldsymbol{x}\in B(\mathbf{0};c)} \log\left(1 + \frac{1}{\sigma_{s,t}^2}\boldsymbol{x}_s^\top \boldsymbol{\Sigma}_{\boldsymbol{\ell}}\boldsymbol{x}_s\right) \\
\leq & \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S},\boldsymbol{x}\in B(\mathbf{0};c)} \log(1 + \frac{1}{\sigma_{s,t}^2}\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_\ell)\|\boldsymbol{x}_s\|_2^2) \\
= & \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_\ell)c_s^2}{\sigma_{s,t}^2}),
\end{aligned}$$

with probability greater than $1 - 2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)}$, and the last inequality holds because $\lambda_{\mathrm{MIN}}(\Sigma_\ell)|\boldsymbol{x}|^2 \leq \boldsymbol{x}^\top \Sigma_\ell \boldsymbol{x} \leq \lambda_{\mathrm{MAX}}(\Sigma_\ell)|\boldsymbol{x}|^2$, for any $\boldsymbol{x}$, by [38]. □

By sub-Gaussian norm bound [29], we get:

Lemma 6 (Theorem 8.3 in [29])**.** *If $\boldsymbol{x} \sim N(0, \boldsymbol{\Sigma})$, for any $\delta \in (0,1)$:*

$$\mathbf{P}\left[\|\boldsymbol{x}\|_2 \leq 4\sqrt{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma})}\sqrt{d} + 2\sqrt{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma})}\sqrt{\log\frac{1}{\delta}}\right] \geq 1 - \delta,$$

where d is the dimension of $\boldsymbol{x}$.

Then, we bound the distance between the estimated gradient and HEAD, and prove it by law of total probability theorem.

Lemma 7. *For any $\delta \in (0,1)$, and $n' \geq \lambda_s^\ell T$,*

$$|\widehat{\frac{\partial U}{\partial \lambda_{s,t}^p}} - \mathrm{HEAD}_{s,t}^p| \leq \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_0^\ell)c_s^2}{\sigma_{s,t}^2}),$$

where $c_s = 4\sqrt{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_s)}\sqrt{d} + 2\sqrt{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_s)}\sqrt{\log\frac{1}{\delta}}$, γ is stepsize, and d is the dimension of feature $\boldsymbol{x}$, with probability greater than $1 - 2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)} - |\mathcal{S}|n'\delta - (|\mathcal{S}| - 1)\delta_{s,t}^p$, and $\delta_{s,t}^p = \max_{s'\in\mathcal{S}\setminus\{s\}} \mathbf{P}[n_{s'}^\ell \geq n' + 1]$.

Proof. According to law of total probability theorem, where event $A = \left|\widehat{\frac{\partial U}{\partial \lambda^p_{s,t}}} - \mathrm{HEAD}^p_{s,t}\right| > \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_\ell)c_s^2}{\sigma^2_{s,t}})$, event $B = \{\|\boldsymbol{x}^\ell_{s,i}\|_2 \leq c_s : \forall s \in \mathcal{S}, i \leq n^\ell_s\}$, and event $C = \{n^\ell_s \leq n' : \forall s \in \mathcal{S}\}$, the probability of event A is bounded by:

$$\mathbf{P}(A) =\mathbf{P}(A|B\cap C)\mathbf{P}(B\cap C) + \mathbf{P}(A|\overline{B\cap C})\mathbf{P}(\overline{B\cap C}) \leq \mathbf{P}(A|B\cap C)\mathbf{P}(B\cap C) + \mathbf{P}(\overline{B\cap C}).$$

Then, the probability of its complement is bounded by:

$$\begin{aligned}\mathbf{P}(\overline{A}) \geq& 1 - \big(\mathbf{P}(A|B\cap C)\mathbf{P}(B\cap C) + \mathbf{P}(\overline{B\cap C})\big) = \mathbf{P}(B\cap C) - \mathbf{P}(A|B\cap C)\mathbf{P}(B\cap C)\\ =& \ \mathbf{P}(\overline{A}|B\cap C)\mathbf{P}(B\cap C) \geq (1 - 2\cdot e^{-\gamma^2 N_1N_2/2T^2(n'+1)})\cdot(1-\delta)^{|\mathcal{S}|n'}\cdot \prod_{s'\in\mathcal{S}\setminus\{s\}}(1-\mathbf{P}[n^\ell_{s'} \geq n'+1]),\end{aligned}$$

where the first term comes from Lem. 5, the second term is by Lem. 6, and the last term derives from the condition, where the number of arrivals n^ℓ_s from source s is less then or equal to n'. Since truncating has forced one coordinate (coordinate s) to be less then or equal to n' already, the condition is applied to the left sources.

The probability can be further simplified by Bernoulli's inequality:

$$\begin{aligned}\mathbf{P}(\overline{A}) \geq& (1 - 2\cdot e^{-\gamma^2 N_1N_2/2T^2(n'+1)})\cdot(1-|\mathcal{S}|n'\delta)\cdot(1-(|\mathcal{S}|-1)\delta^p_{s,t})\\ \geq& 1 - 2\cdot e^{-\gamma^2 N_1N_2/2T^2(n'+1)} - |\mathcal{S}|n'\delta - (|\mathcal{S}|-1)\delta^p_{s,t},\end{aligned}$$

where $\delta^p_{s,t} = \max_{s'\in\mathcal{S}\setminus\{s\}}\{\mathbf{P}[n^\ell_{s'} \geq n'+1]\}$. □

By Lems. 4 and 7, we instantly get the distance between the estimated and true gradient, shown in Lem. 1

Appendix C
Proof of Theorem 2

We bound the quality of our gradient estimator by Lem. 1:

Lemma 8. *At each iteration k, with probability greater than* $1 - (2\cdot e^{-\gamma^2 N_1N_2/2T^2(n'+1)} + |\mathcal{S}|n'\delta + (|\mathcal{S}|-1)\mathrm{P}_{\mathrm{MAX}})\cdot P_{\mathtt{TOT}}$,

$$\langle \boldsymbol{v}(k), \nabla U(\boldsymbol{\lambda}(k))\rangle \geq a\cdot \max_{\boldsymbol{v}\in\mathcal{D}}\langle \boldsymbol{v}, \nabla U(\boldsymbol{\lambda}(k))\rangle - b, \tag{26}$$

where

$$a = 1 - \mathrm{P}_{\mathrm{MAX}}, \quad \textit{and} \tag{27}$$

$$b = 2\lambda_{\mathrm{MAX}}\cdot\gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}^\ell_0)c_s^2}{\sigma^2_{s,t}}), \tag{28}$$

for $\mathrm{P}_{\mathrm{MAX}} = \max_{k=1,\dots,K}\mathrm{P}_{\mathrm{MAX}}(k) = \max_{l\in\mathcal{L},s\in\mathcal{S}}\mathrm{P}[n^\ell(k)_s \geq n'+1]$ *(*$n^\ell_s(k)$ *is a Poisson r.v. with parameter* $\lambda^\ell_s(k)T$*), and* $\lambda_{\mathrm{MAX}} = \max_{\boldsymbol{\lambda}\in\mathcal{D}}\|\boldsymbol{\lambda}\|_1$.

Proof. We now use the superscript k to represent the parameters for the kth iteration: we find $\boldsymbol{v}^k \in \mathcal{D}$ that maximizes $\langle \boldsymbol{v}^k, \widehat{\nabla U(\boldsymbol{\lambda}^k)}\rangle$. Let $\boldsymbol{u}^k \in \mathcal{D}$ be the vector that maximizes $\langle \boldsymbol{u}^k, \nabla U(\boldsymbol{\lambda}^k)\rangle$ instead. We have

$$\begin{aligned}\langle \boldsymbol{v}^k, \nabla U(\boldsymbol{\lambda}^k)\rangle \geq& \langle \boldsymbol{v}^k, \widehat{\nabla U(\boldsymbol{\lambda}^k)}\rangle - \lambda_{\mathrm{MAX}}\cdot\gamma\max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_\ell)c_s^2}{\sigma^2_{s,t}})\\ \geq& \langle \boldsymbol{u}^k, \widehat{\nabla U(\boldsymbol{\lambda}^k)}\rangle - \lambda_{\mathrm{MAX}}\cdot\gamma\max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_\ell)c_s^2}{\sigma^2_{s,t}})\\ \geq& (1-\mathrm{P}_{\mathrm{MAX}})\cdot\langle \boldsymbol{u}^k, \nabla U(\boldsymbol{\lambda}^k)\rangle - 2\lambda_{\mathrm{MAX}}\cdot\gamma\max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_\ell)c_s^2}{\sigma^2_{s,t}}),\end{aligned}$$

where the first and last inequalities are due to Lem. 1 and the second inequality is because $\boldsymbol{v}^k$ maximizes $\langle \boldsymbol{v}^k, \widehat{\nabla U(\boldsymbol{\lambda}^k)}\rangle$. The above inequality requires the satisfaction of Lem. 1 for every partial derivative. By union bound, the above inequality satisfies with probability greater than $1 - (2\cdot e^{-\gamma^2 N_1N_2/2T^2(n'+1)} + |\mathcal{S}|n'\delta + \sum_{s\in\mathcal{S},t\in\mathcal{T}}\sum_{p\in\mathcal{P}_{s,t}}(|\mathcal{S}|-1)\delta^p_{s,t})\cdot P_{\mathtt{TOT}}$, and thus greater than $1 - (2\cdot e^{-\gamma^2 N_1N_2/2T^2(n'+1)} + |\mathcal{S}|n'\delta + (|\mathcal{S}|-1)\mathrm{P}_{\mathrm{MAX}})\cdot P_{\mathtt{TOT}}$. □

By Lemma 2 in [1], and Lipschitz constant of ∇U: $L = 2T^2 P_{\mathtt{TOT}}\cdot\max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}^\ell_0)c_s^2}{\sigma^2_{s,t}})$, we get:

Lemma 9. *With probability greater than* $1 - (2 \cdot e^{-\gamma^2 N_1 N_2 / 2T^2(n'+1)}$ $- |\mathcal{S}|n'\delta - (|\mathcal{S}| - 1)\mathrm{P}_{\mathrm{MAX}}) \cdot P_{\mathtt{TOT}}K$*, the output solution* $\boldsymbol{\lambda}(K) \in \mathcal{D}$ *satisfies:*

$$U(\boldsymbol{\lambda}(K)) \geq (1 - e^{\mathrm{P}_{\mathrm{MAX}}-1}) \max_{\boldsymbol{\lambda} \in \mathcal{D}} U(\boldsymbol{\lambda}) - (T^2 P_{\mathtt{TOT}} \lambda_{\mathrm{MAX}}^2 + 2\lambda_{\mathrm{MAX}})\gamma \max_{\ell \in \mathcal{L}, s \in \mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_0^\ell) c_s^2}{\sigma_{s,t}^2}),$$

where $K = \frac{1}{\gamma}$ *is the number of iterations.*

Then, we utilize subexponential Poisson tail bound described by Thm. 1 in [36], and organize constants in above lemma to obtain Thm. 2.

Proof. From Thm. 1 in [36], the probability is an increasing function w.r.t. λ_s^ℓ, and λ_{MAX} is an upper bound for λ_s^ℓ. Letting $u = \frac{n' - \lambda_{\mathrm{MAX}} T}{\lambda_{\mathrm{MAX}} T}$, we have:

$$\mathrm{P}_{\mathrm{MAX}} < e^{-\frac{(n' - \lambda_{\mathrm{MAX}} T)^2}{2\lambda_{\mathrm{MAX}} T} h(\frac{n' - \lambda_{\mathrm{MAX}} T}{\lambda_{\mathrm{MAX}} T})} = e^{-\lambda_{\mathrm{MAX}} T((1+u)\ln(1+u) - u)} \leq \Omega(e^{-\lambda_{\mathrm{MAX}} T u}) = \epsilon_1,$$

where the last inequality holds because $(1 + u)\ln(1 + u) - u > u$ when u is large enough, e.g., $u \geq 4$. Thus, $n' = O(\lambda_{\mathrm{MAX}} T + \ln \frac{1}{\epsilon_1})$. Then, we have

$$(2 \cdot e^{-\gamma^2 N_1 N_2 / 2T^2(n'+1)} - |\mathcal{S}|n'\delta - (|\mathcal{S}| - 1)\mathrm{P}_{\mathrm{MAX}}) \cdot P_{\mathtt{TOT}} K = \epsilon_0,$$

and assume that

$$\begin{cases} 2 \cdot e^{-\gamma^2 N_1 N_2 / 2T^2(n'+1)} \leq \frac{\epsilon_0}{3P_{\mathtt{TOT}} K} \\ |\mathcal{S}|n'\delta \leq \frac{\epsilon_0}{3P_{\mathtt{TOT}} K} \\ (|\mathcal{S}| - 1)\mathrm{P}_{\mathrm{MAX}} \leq \frac{\epsilon_0}{3P_{\mathtt{TOT}} K} \end{cases}$$

Then, we first get: $K = O(\frac{\epsilon_0}{P_{\mathtt{TOT}}(|\mathcal{S}|-1)}\epsilon_1)$, then $\delta = O(\frac{\epsilon_0}{P_{\mathtt{TOT}} K |\mathcal{S}| n'})$, and $N_1 N_2 = \Omega(\ln \frac{P_{\mathtt{TOT}} K}{\epsilon_0} \cdot T^2(n' + 1)K^2)$. Finally, we have

$$\epsilon_2 = (T^2 P_{\mathtt{TOT}} \lambda_{\mathrm{MAX}}^2 + 2\lambda_{\mathrm{MAX}}) \frac{1}{K} \cdot \max_{\ell \in \mathcal{L}, s \in \mathcal{S}} \log(1 + \frac{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_\ell) c_s^2}{\sigma_{s,t}^2}),$$

where $c_s = 4\sqrt{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_s)}\sqrt{d} + 2\sqrt{\lambda_{\mathrm{MAX}}(\boldsymbol{\Sigma}_s)}\sqrt{\log \frac{1}{\delta}}$. □