Distributed Experimental Design Networks

Yuanyuan Li
Northeastern University
Boston, USA
yuanyuanli@ece.neu.edu

Lili Su
Northeastern University
Boston, USA
l.su@northeastern.edu

Carlee Joe-Wong
Carnegie Mellon University
Pittsburgh, USA
cjoewong@andrew.cmu.edu

Edmund Yeh
Northeastern
Boston, USA
eyeh@ece.neu.edu

Stratis Ioannidis
Northeastern
Boston, USA
ioannidis@ece.neu.edu

ABSTRACT

As edge computing capabilities increase, deployments to learn models in a heterogeneous edge environment have emerged. We consider an experimental design network, as introduced by Liu et al., in which network routing and rate allocation are designed to aid the transfer of data from sensors to heterogeneous learners. We generalize this setting by considering Gaussian sources, incorporating multicast, but also–crucially–studying distributed algorithms in this setting. From a technical standpoint, we show that, assuming Gaussian sensor sources still yields a continuous DR-submodular experimental design objective. We also propose a distributed Frank-Wolfe algorithm yielding allocations within a $1-1/e$ factor from the optimal. Numerical evaluations show that our proposed algorithm outperforms competitors w.r.t. both objective maximization and model learning quality.

KEYWORDS

Experimental Design, DR-submodularity, Distributed algorithm

1 INTRODUCTION

The proliferation of mobile IoT devices and mobile services has resulted in a tremendous amount of data and ever-increasing deployment of machine learning tasks. We study a network in which heterogeneous learners dispersed at different geographical locations perform local learning tasks by consuming data collected from diverse data sources, as illustrated in Fig. 1. These data sources are deployed across the network, and generate data streams containing both features and labels. Data is transmitted over the network (potentially through several intermediate router nodes) towards learner nodes for training purposes. Networks have limited computing/transmission resources; we are thus interested in the design of rate allocation strategies so that the quality of model training at learners is maximized, subject to these network constraints.

Studying this problem is of practical significance. For instance, in a mobile edge computing network [1, 39], end devices such as mobile phones act as data sources and transmit data to edge servers, which serve as learners for intensive model training. Additionally, in a smart city [2, 30], various sensors capture data, such as image, temperature, humidity, traffic, and seismic measurements, that can aid the forecasting of transportation traffic, the spread of disease, pollution levels, the weather, etc. Training for each task could happen at distinct public service entities, e.g., a transportation authority, an energy company, the fire department, etc., each aiming to perform a different prediction task.

Liu et al. [27] recently proposed a mathematical model of such a network. Crucially, they quantify the impact that data samples have on model training accuracy by leveraging objectives motivated from *experimental design* [8]. They show that the problem has a continuous DR-submodular objective and propose a Frank-Wolfe like algorithm that yields an approximation within $1-1/e$ from the optimal. However, their proposed algorithm is centralized, and assumes a full view of network conditions. Establishing a coordinator for monitoring network conditions counters the advantage of distributed learning on local devices. It also poses significant scalability problems, when the number of sensors/data sources, learners, or both, is large. Liu at al. also model unicast transmissions between sources and learners, thereby under-utilizing network resources. This is important in practice, when desired training datasets may overlap and sending data between them is bandwidth-intensive. This will indeed be the case in, e.g., a smart city scenario, where data such as, e.g., video captures from multiple locations in the city can collectively amount to terabytes of data collected in a few seconds. Finally, Liu et al. make additional restrictive assumptions, including, e.g., that data samples come from a finite set, which is also not realistic as features are usually continuous values, and labeling noise is homogeneous across sources; nevertheless, they mention possible ways through which these assumptions can be extended to, e.g., Gaussian and heterogeneous noise sources.

We extend the model from Liu et al. [27] by (a) proposing a distributed algorithm for the optimization of this problem, and (b) considering multicast transmissions. Multicast transmissions result in non-differentiable constraints, which makes standard decentralization techniques inapplicable. We prove that the problem maintains continuous DR-submodularity in our setting, and use this property, along with a primal dual approach, to construct a distributed algorithm. We also implement the extensions suggested by Liu et al., including the assumption of Gaussian sources and noise heterogeneity: we formally incorporate the latter in our mathematical formulation, and further characterize performance under these extensions both theoretically and experimentally. From a technical standpoint, the Gaussianity of sources requires revisiting how gradient estimation is performed, compared to Liu et al. [27], as well as devising new estimation bounds. We overcome both challenges,

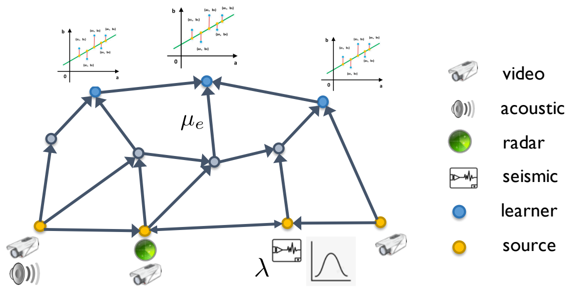


Figure 1: An example of a experimental design network, adapted from Liu et al. [27]. Yellow nodes are sources and blue nodes are learners. Sources generate streams of data from diverse sensors, e.g., cameras, microphones, seismic sensors, etc. Learners train distinct models over (possibly overlapping) received data. Our network problem amounts to routing and allocating bandwidth to data traffic in a manner that maximizes the social welfare, i.e., the aggregate quality of the learned models trained across learners.

as well as challenges related to the non-differentiability of multicast constraints, when constructing our distributed optimization algorithm. Overall, our contributions are as follows:

- We generalize and extend experimental design networks with Gaussian sources and multicast transmission, which is more realistic and yields higher throughputs.
- We prove that, as in Liu et al. [27], assuming Poisson data streams in steady state, and training Bayesian linear regression model as a learning task, our experimental design objective maintains its continuous DR-submodularity.
- 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.
- Finally, we conduct extensive simulations over both synthetic and backbone network topologies. Our proposed algorithms outperform all competitors in both utility maximization and the quality of model estimation, and our distributed algorithm's performance is close to the centralized version.

The remainder of this paper is organized as follows. Sec. 2 provides a literature review. We review experimental design networks in Sec. 3, and introduce our extended extdistributed model in Sec. 4. Sec. 5 describes our analysis of the problem and proposed centralized algorithm, while Sec. 6 describes our distributed algorithm. We propose additional distributed algorithms in Sec. 7. We present numerical experiments in Sec. 8, and conclude in Sec. 9.

2 RELATED WORK

Experimental Design. The experimental design problem is classic and well-studied [8, 32]. Several works consider the D-optimality objective [14–16, 18, 20] for a single learner subject to budget constraints on the cost for conducting the experiments. Liu et al. [27] are the first to formalize experimental design networks. We review their formulation in Sec. 3. They propose a Frank-Wolfe variant algorithm with a novel gradient estimator to give an optimality guarantee of $1-1/e$. We follow and extend this work but, as discussed in the introduction, deviate from it by proposing a decentralized algorithm and considering multicast transmissions.

Submodular Maximization. Submodularity is traditionally studied in the context of set functions [9], but can also be extended to functions over the integer lattice [35] and the continuous domain [7]. Maximizing a monotone submodular function subject to a matroid constraint is classic. Krause and Golovin [26] show that the greedy algorithm achieves a $1/2$ approximation ratio. Calinescu et al. [9] propose a *continuous greedy* algorithm improving the ratio to $1-1/e$, that applies a Frank-Wolfe [6] variant to the multilinear extension of the submodular objective. With the help of auxiliary potential functions, Filmus and Ward [12] run a non-oblivious local search after the greedy algorithm, and also produce a $1-1/e$ approximation ratio. Further improvements are made by Sviridenko et al. [37] for a more restricted class of submodular functions with bounded curvature. Bian et al. [7] show that the same Frank-Wolfe variant can be used to maximize continuous DR-submodular functions within a $1-1/e$ ratio. Both the centralized algorithm by Liu et al. [27] and by this paper are applications of the algorithm by Bian et al. [7] to our setting; in both cases (Liu et al. and ours) recovering their guarantees requires devising novel gradient estimators and bounding their estimation accuracy. We further depart by also considering a distributed version of this algorithm, where each node accesses to only neighborhood knowledge. This requires distributing constituent steps of the algorithm by Bian et al. via a primal-dual approach; as discussed below, the latter is a priori designed for convex optimization problems, a property that does not hold in our setting.

Convergence of Primal-Dual Algorithm. Nedić and Ozdaglar [31] 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 [3] 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 constraint (ours are inequality constraints). Lyapunov equations are usually employed for a continuous version of primal-dual gradient algorithm [36]; this requires objectives to be strictly concave. Feijer and Paganini [11] provide stability proofs for primal–dual gradient dynamics with concave objectives through Krasovskii's method and the LaSalle invariance principle. We follow this approach to guarantee the convergence of our decentralized algorithm.

Distributed Algorithms. Low and Lapsley [28] consider a unicast multiple source one sink network, and maximize a strictly concave utility w.r.t. rate allocation under bandwidth constraints. They solve the dual problem using a gradient projection (dual) algorithm, and prove optimality guarantees. Another approach for solving network resource allocation problems exactly is the classic primal-dual algorithm (see, e.g., [36]): intuitively, a primal algorithm is implemented at the sources and a dual algorithm at the links. Lun et al. [29] consider a single-source/multiple-sinks setting, combined with hop-to-hop multicast. They again maximize concave utilities w.r.t. rate allocation under bandwidth constraints.

They decentralize problems with both linear and strictly concave utilities through two different techniques: subgradient optimization with ϵ-relaxation method [5] and primal-dual algorithm mentioned before, respectively.

All of the above frameworks pertain to convex optimization problems. Our objective is continuous DR-submodular; thus, these methods do not directly extend to our setting. Tychogiorgos et al. [38] provide the theoretical foundations for distributed dual algorithm solution of non-convex problems. Our analysis requires combining such techniques, particularly the Feijer and Paganini [11] primal-dual algorithm described above, with the (centralized) Frank-Wolfe variant by Bian et al. [7], 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.

3 BACKGROUND

We assume that learners train Bayesian linear regression models [13, 22] in our setting, and quantify the model quality improvement from data through experimental design [8]. We first review these classic components, and then a centralized experimental design networks, formulated by Liu et al. [27], before introducing our distributed experimental design networks.

3.1 Bayesian Linear Regression

In the standard linear regression setting, a learner observes n samples $(\boldsymbol{x}_i, y_i)$, $i=1,\dots,n$, where $\boldsymbol{x}_i\in\mathbb{R}^d$ and $y_i\in\mathbb{R}$ are the feature vector and the label of sample i, respectively. Labels are assumed to be linearly related to the features; particularly, there exists a model parameter vector $\boldsymbol{\beta}\in\mathbb{R}^d$ such that

$$y_i=\boldsymbol{x}_i^\top\boldsymbol{\beta}+\epsilon_i,\quad \text{for all } i\in\{1,\dots,n\}, \tag{1}$$

and ϵ_i are i.i.d. zero mean normal noise variables with variance σ^2 (i.e., $\epsilon_i\sim N(0,\sigma^2)$).

The learner's goal is to estimate the model parameter $\boldsymbol{\beta}$ from samples $\{(\boldsymbol{x}_i,y_i)\}_{i=1}^n$. In Bayesian linear regression, it is additionaly assumed that $\boldsymbol{\beta}$ is sampled from a prior normal distribution with mean $\boldsymbol{\beta}_0\in\mathbb{R}^d$ and covariance $\Sigma_0\in\mathbb{R}^{d\times d}$ (i.e., $\boldsymbol{\beta}\sim N(\boldsymbol{\beta}_0,\Sigma_0)$). Under this prior, given dataset $\{(\boldsymbol{x}_i,y_i)\}_{i=1}^n$, maximum a posteriori (MAP) estimation of $\boldsymbol{\beta}$ amounts to [13]:

$$\hat{\boldsymbol{\beta}}_{\mathsf{MAP}}=(\boldsymbol{X}^\top\Sigma^{-1}\boldsymbol{X}+\Sigma_0^{-1})^{-1}(\boldsymbol{X}^\top\Sigma^{-1}\boldsymbol{y}+\Sigma_0^{-1}\boldsymbol{\beta}_0), \tag{2}$$

where $\boldsymbol{X}=[\boldsymbol{x}_i^\top]_{i=1}^n\in\mathbb{R}^{n\times d}$ is the matrix of features, $\boldsymbol{y}\in\mathbb{R}^n$ is the vector of labels, Σ is a diagonal matrix, and $\Sigma_{ii}=\sigma^2$, and $\boldsymbol{\beta}_0,\Sigma_0$ are the mean and covariance of the prior, respectively. The quality of this estimator can be characterized by the covariance of the estimation error difference $\hat{\boldsymbol{\beta}}_{\mathsf{MAP}}-\boldsymbol{\beta}$ [13]:

$$\mathsf{cov}(\hat{\boldsymbol{\beta}}_{\mathsf{MAP}}-\boldsymbol{\beta})=\left(\boldsymbol{X}^\top\Sigma^{-1}\boldsymbol{X}+\Sigma_0^{-1}\right)^{-1}\in\mathbb{R}^{d\times d}. \tag{3}$$

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 $\mathsf{cov}(\hat{\boldsymbol{\beta}}_{\mathsf{MAP}}-\boldsymbol{\beta})$ are high) are directions in which the prediction $\hat{y}=\boldsymbol{x}^\top\hat{\boldsymbol{\beta}}_{\mathsf{MAP}}$ will have the highest prediction error (see also [27]).

3.2 Experimental Design

In experimental design, a learner with prior $N(\boldsymbol{\beta}_0,\Sigma_0)$ on parameters $\boldsymbol{\beta}$ determines which experiments to conduct to learn the most accurate linear model. Formally, given p possible experiment settings, each described by feature vectors $\boldsymbol{x}_i\in\mathbb{R}^d$, $i=1,\dots,p$, the learner selects a total of n experiments to conduct with these feature vectors, possibly with repetitions, collects associated labels, and then performs linear regression on these sample pairs. In classic experimental design (see, e.g., Section 7.5 in [8]), the selection is formulated as an optimization problem minimizing a scalarization of the covariance, given by Eq. (3). For example, in D-optimal design, the vector $\boldsymbol{n}=[n_i]_{i=1}^p\in\mathbb{N}^p$ of the number of times each experiment is to be performed is determined by maximizing:

$$\text{Max.: } G(\boldsymbol{n};\sigma,\Sigma_0)=\log\det\Big(\sum_{i=1}^p\frac{n_i}{\sigma^2}\boldsymbol{x}_i\boldsymbol{x}_i^\top+\Sigma_0^{-1}\Big), \tag{4a}$$

$$\text{s.t.: } \sum_{i=1}^p n_i=n. \tag{4b}$$

Function $G:\mathbb{N}^p\to\mathbb{R}_+$ in (4a) is (a) monotone-increasing and (b) DR-submodular w.r.t. $\boldsymbol{n}$ [27]. Moreover, as the covariance summarises the expected prediction error in all directions, minimizing the log det (i.e., the sum of the logs of eigenvalues of the covariance) imposes an overall bound on the magnitude of this error.

3.3 Experimental Design Networks

Liu et al. [27] consider a network that aims to facilitate a distributed learning task, illustrated in Fig. 1. The network comprises the following components.

Graph. They model the above system as a general multi-hop network with a topology represented by a *directed acyclic graph* (DAG) $\mathcal{G}(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}\subset\mathcal{V}\times\mathcal{V}$ is the set of links. Each link $e=(u,v)\in\mathcal{E}$ can transmit data at a maximum rate (i.e., link capacity) $\mu^e\geq 0$.

Data Sources. They assume that there exists a *finite set* $\mathcal{X}\subset\mathbb{R}^d$ of experiments every source can conduct. Once an experiment with features $\boldsymbol{x}\in\mathcal{X}$ is conducted, the source labels it with $y\in\mathbb{R}$ of type t out of a set of possible types $\mathcal{T}$. They assume that every source generates labeled pairs $(\boldsymbol{x},y)\in\mathbb{R}^d\times\mathbb{R}$ of type t according to a Poisson process of rate $\lambda_{\boldsymbol{x},t}^s\geq 0$. Moreover, they assume that generated data follows the linear model in Eq. (1); that is, for every type $t\in\mathcal{T}$, there exists a $\boldsymbol{\beta}_t\in\mathbb{R}^d$ s.t. $y=\boldsymbol{x}^\top\boldsymbol{\beta}_t+\epsilon_t$ where $\epsilon_t\in\mathbb{R}$ are i.i.d. zero mean normal noise variables with variance $\sigma_t^2>0$, independent across experiments and sources $s\in\mathcal{S}$.

Learners. Each learner $\ell\in\mathcal{L}$ wishes to learn a model $\boldsymbol{\beta}_{t^\ell}$ for some type $t^\ell\in\mathcal{T}$. They assume that each learner has a prior $N(\boldsymbol{\beta}_\ell,\Sigma_\ell)$ on $\boldsymbol{\beta}_{t^\ell}$. The learner wishes to use the network to receive data pairs $(\boldsymbol{x},y)$ of type t^ℓ, and subsequently estimate $\boldsymbol{\beta}_{t^\ell}$ through the MAP estimator in Eq. (2).

Network Constraints. Their network design aims at allocating network capacity to different flows to meet learner needs. The network adopts *hop-by-hop routing* and *unicast* for transmission. In particular, at the end of a data acquisition time period T each learner $\ell\in\mathcal{L}$ estimates $\boldsymbol{\beta}_{t^\ell}$ via MAP estimation, based on the data it has received during this period. They denote by $\boldsymbol{\lambda}^\ell=[\lambda_{\boldsymbol{x}}^\ell]_{\boldsymbol{x}\in\mathcal{X}}\in$

$\mathbb{R}_+^{|\mathcal{X}|}$, for all $\ell \in \mathcal{L}$, the incoming traffic with different features $\boldsymbol{x} \in \mathcal{X}$ of type t^ℓ at $\ell \in \mathcal{L}$, and $\boldsymbol{\lambda} = \big[[\lambda^e_{\boldsymbol{x},t}]_{\boldsymbol{x}\in\mathcal{X},t\in\mathcal{T},e\in\mathcal{E}}; [\boldsymbol{\lambda}^\ell]_{\ell\in\mathcal{L}}\big]$ the vector comprising edge and learner rates. Let $n^\ell_{\boldsymbol{x}} \in \mathbb{N}$ be the cumulative number of times that a pair $(\boldsymbol{x}, y)$ of type t^ℓ was collected by learner ℓ during this period, and $\boldsymbol{n}^\ell = [n^\ell_{\boldsymbol{x}}]_{\boldsymbol{x}\in\mathcal{X}}$ the vector of arrivals across all experiments. Motivated by standard experimental design (see Sec. 3.2), they define the utility at learner $\ell \in \mathcal{L}$ as the following expectation:

$$U^\ell(\boldsymbol{\lambda}^\ell) = \mathbb{E}_{\boldsymbol{\lambda}^\ell}\big[G^\ell(\boldsymbol{n}^\ell)\big] = \sum_{\boldsymbol{n}\in\mathbb{N}^{|\mathcal{X}|}} G^\ell(\boldsymbol{n}) \cdot \Pr[\boldsymbol{n}^\ell = \boldsymbol{n}], \quad (5)$$

where $G^\ell(\boldsymbol{n}^\ell) \equiv G(\boldsymbol{n}^\ell; \sigma_{t^\ell}, \Sigma_\ell)$, G is given by (4a), and the distribution Pr is Poisson, with parameters governed by $\boldsymbol{\lambda}^\ell T$ (see also Ass. 1). Liu et al. [27] solve the following problem:

$$\text{Maximize:} \quad U(\boldsymbol{\lambda}) = \sum_{\ell\in\mathcal{L}} (U^\ell(\boldsymbol{\lambda}^\ell) - U^\ell(\boldsymbol{0})), \quad (6a)$$

$$\text{s.t.} \quad \boldsymbol{\lambda} \in \mathcal{D}, \quad (6b)$$

where feasible set $\mathcal{D}$ defines (a) flow conservation rules, and (b) unicast link capacity constraints. They prove that, assuming Poisson data streams in steady state, the global objective (6a) is a continuous DR-submodular function (see Defn. 5.1). A *centralized* Frank-Wolfe type algorithm is proposed that outputs a solution within a $1 - 1/e$ factor from the optimal. This requires a novel gradient estimation component, which is designed based on Poisson tail bounds and sampling.

4 PROBLEM FORMULATION

As a distributed learning network, it is desirable and necessary to (a) seek a *distributed algorithm* determining the rate allocation, whereby each node's actions require knowledge only from itself and its neighbourhood. Furthermore, to fully utilize the limited transmission bandwidth and increase throughput, we (b) assume *multicast* instead of unicast transmissions. Multicast transmissions result in non-differentiable link capacity constraints, which makes decentralization challenging. Finally, we also implement several additional changes proposed by Liu et al. [27] as possible extensions: (c) our sources generate features from *a uncountable set* instead a finite set, which is not realistic; (d) we remove the DAG assumption in [27] and consider a *general directed graph*, by using source routing instead of hop-by-hop routing, also making multicast analysis tractable; and (e) we consider *heterogeneous noise* instead of homogeneous noise at different sources, which is again more realistic.

4.1 Network Model

We model the system as a general multi-hop network with a topology represented by a (general) *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 $e = (u, v) \in \mathcal{E}$ can transmit data at a maximum rate (i.e., 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 stream of labeled data. In particular, we assume that source s generates feature $\boldsymbol{x}$ following *Gaussian* distribution $N(\boldsymbol{0}, \Sigma_s)$, over $\mathbb{R}^d$. Once an experiment with features $\boldsymbol{x}$ is conducted, the source labels it with $y \in \mathbb{R}$ of type t out of a set of possible types $\mathcal{T}$. Intuitively, features $\boldsymbol{x}$ correspond to parameters set 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).

We also assume that every source s generates labeled pairs $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ of type t according to a Poisson process of rate $\lambda_{s,t} \geq 0$. Moreover, we assume that generated data follows again the linear model (1); that is, for every type $t \in \mathcal{T}$ from source $s \in \mathcal{S}$, there exists a $\boldsymbol{\beta}_t \in \mathbb{R}^d$ s.t. $y = \boldsymbol{x}^\top \boldsymbol{\beta}_t + \epsilon_{s,t}$ where $\epsilon_{s,t} \in \mathbb{R}$ are i.i.d. zero mean normal noise variables with variance $\sigma^2_{s,t} > 0$: in contrast to [27], the noise level is *heterogeneous* (a.k.a. heteroskedastic) across both sources and experiment types.

Learners. We make exactly the same assumptions about learners as Liu et al. [27], summarized in Sec. 3.3. In short, each learner $\ell \in \mathcal{L}$ wishes to learn a model $\boldsymbol{\beta}_{t^\ell}$ for some type $t^\ell \in \mathcal{T}$. Assume that each learner has a prior $N(\boldsymbol{\beta}_\ell, \Sigma_\ell)$ on $\boldsymbol{\beta}_{t^\ell}$.

Network Constraints. The different data pairs $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \mathbb{R}$ generated by sources are transmitted over edges in the network along with their types $t \in \mathcal{T}$ and eventually delivered to learners. Our network design aims to allocate network capacity to different flows to meet learner needs. We assume *source routing* and denote the paths set from source s with which data pairs of type t as $\mathcal{P}_{s,t}$. Data pairs are transmitted through the path $p \in \mathcal{P}_{s,t}$ from source s to learner ℓ. In each path set $\mathcal{P}_{s,t}$, no two paths reach to the same learner. That is, the size of $\mathcal{P}_{s,t}$, i.e., $|\mathcal{P}_{s,t}|$, equals to the number of learners with type t. Furthermore, we consider the *multirate multicast* transmission [36], which saves network resources compared to unicast. Here, we denote the *virtual* rate [36], 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 $P_{\text{TOT}} = \sum_{s\in\mathcal{S},t\in\mathcal{T}} |\mathcal{P}_{s,t}|$ be the total number of paths. We refer to the vector

$$\boldsymbol{\lambda} = [\lambda^p_{s,t}]_{s\in\mathcal{S},t\in\mathcal{T},p\in\mathcal{P}_{s,t}} \in \mathbb{R}_+,^{P_{\text{TOT}}} \quad (7)$$

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

$$\sum_{s\in\mathcal{S},t\in\mathcal{T}} \max_{p\in\mathcal{P}_{s,t}:e\in p} \lambda^p_{s,t} \leq \mu^e. \quad (8)$$

(only data pair of same type generated from same source can be multicast). For learner l, we denote by

$$\lambda^\ell_s = \lambda^p_{s,t^\ell} \quad (9)$$

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

$$\sum_{p\in\mathcal{P}_{s,t}} \lambda^p_{s,t} \leq \lambda_{s,t}. \quad (10)$$

We make the similar assumption as [27] on the network substrate:

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.*

This is satisfied, if, e.g., the network is a Kelly network [23] where Burke's theorem holds [13].

4.2 Gaussian Experimental Design Networks

Different from the classic experimental design problem (4), as well as Liu et al. [27], where the set of features is assumed to be finite, we consider a uncountable feature set. Let $n_s^\ell \in \mathbb{N}$ be the cumulative number of times that a pair $(\boldsymbol{x}, y)$ from source s was 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 $\boldsymbol{x}_{s,i}^\ell$ the i-th feature generated from source s to learner ℓ, and $\boldsymbol{X}^\ell = [[\boldsymbol{x}_{s,i}^\ell]_{i=1}^{n_s^\ell}]_{s\in\mathcal{S}}$ the feature matrix received at learner ℓ. The D-optimal design objective for learner ℓ is to maximize $G^\ell(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell) \equiv G(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell; \boldsymbol{\sigma}_{t^\ell}, \Sigma_\ell)$, and:

$$G(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell; \boldsymbol{\sigma}_{t^\ell}, \Sigma_\ell) = \log\det\Big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s^\ell}\frac{\boldsymbol{x}_{s,i}^\ell(\boldsymbol{x}_{s,i}^\ell)^\top}{\sigma_{s,t^\ell}^2} + \Sigma_\ell^{-1}\Big). \tag{11}$$

We consider a data acquisition time period T, at the end of which each learner $\ell \in \mathcal{L}$ estimates $\boldsymbol{\beta}_t$ based on the data it has received during this period via MAP estimation. Under Asm. 1, the arrivals of pertinent data pairs at learner ℓ from source s form a Poisson process with rate λ_s^ℓ. The PMF of arrivals is:

$$\Pr[\boldsymbol{n}^\ell = \boldsymbol{n}] = \prod_{s\in\mathcal{S}}\frac{(\lambda_s^\ell T)^{n_s^\ell}e^{-\lambda_s^\ell T}}{n_s^\ell!}, \tag{12}$$

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

$$f(\boldsymbol{X}^\ell = \boldsymbol{X}) = \prod_{s\in\mathcal{S}}\prod_{i=1}^{n_s^\ell}\frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{|\Sigma_s|}}e^{-\frac{1}{2}\boldsymbol{x}_{s,i}^\top\Sigma_s^{-1}\boldsymbol{x}_{s,i}}, \tag{13}$$

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 the following expectation:

$$\begin{aligned} U^\ell(\boldsymbol{\lambda}^\ell) &= \mathbb{E}_{\boldsymbol{n}^\ell}\big[\mathbb{E}_{\boldsymbol{X}^\ell}[G^\ell(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell)|\boldsymbol{n}^\ell]\big] \\ &= \sum_{\boldsymbol{n}\in\mathbb{N}^{|\mathcal{S}|}}\Pr[\boldsymbol{n}^\ell = \boldsymbol{n}]\int_{\boldsymbol{X}\in\mathbb{R}^{\sum_s n_s}} G^\ell(\boldsymbol{X}, \boldsymbol{n})f(\boldsymbol{X}^\ell = \boldsymbol{X})\mathrm{d}\boldsymbol{X}, \end{aligned} \tag{14}$$

where $\boldsymbol{\lambda}^\ell = [\lambda_s^\ell]_{s\in\mathcal{S}}$. We wish to solve the following problem:

$$\text{Maximize:} \quad U(\boldsymbol{\lambda}) = \sum_{\ell\in\mathcal{L}}(U^\ell(\boldsymbol{\lambda}^\ell) - U^\ell(\boldsymbol{0})), \tag{15a}$$

$$\text{s.t.} \quad \boldsymbol{\lambda} \in \mathcal{D}, \tag{15b}$$

where $U^\ell(\boldsymbol{0})$ is lower bound for $U^\ell(\boldsymbol{\lambda}^\ell)$, added to ensure the non-negativity of the objective which is needed to state guarantees in terms of an approximation ratio (c.f. Thm. 5.3), and feasible set $\mathcal{D}$ is defined by constraints (7)-(10). The feasible set is a down-closed convex set. However, this problem is s non-convex in general.

5 CENTRALIZED ALGORITHM

In this section, we propose a centralized poly-time algorithm with a new gradient estimation, due to Gaussian sources, to solve Prob. (15). By establishing a submodular objective, we achieve an optimality guarantee of $1 - 1/e$.

5.1 DR-submodularity

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

Algorithm 1: Frank-Wolfe Variant

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

Definition 5.1 (DR-Submodularity [7, 35]). 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 $\boldsymbol{x} \le \boldsymbol{y}$ and all $k \in \mathbb{N}$,

$$f(\boldsymbol{x} + k\boldsymbol{e}_j) - f(\boldsymbol{x}) \ge f(\boldsymbol{y} + k\boldsymbol{e}_j) - f(\boldsymbol{y}), \text{ for all } j = 1, \dots, p, \tag{16}$$

where $\boldsymbol{e}_j$ is the j-th standard basis vector. Moreover, if (16) holds for a real valued function $f : \mathbb{R}_+^p \to \mathbb{R}$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^p$ such that $\boldsymbol{x} \le \boldsymbol{y}$ and all $k \in \mathbb{R}_+$, the function is called *continuous DR-submodular.*

The following theorem establishes that objective (15a) is a continuous DR-submodular function.

Theorem 5.2. *Objective $U(\boldsymbol{\lambda})$ is (a) monotone-increasing and (b) continuous DR-submodular w.r.t. $\boldsymbol{\lambda}$. Moreover, the partial derivative of U is:*

$$\frac{\partial U}{\partial \lambda_{s,t}^p} = \sum_{n=0}^{\infty}\Delta_s^\ell(\boldsymbol{\lambda}^\ell, n)\cdot\Pr[n_s^\ell = n]\cdot T, \tag{17}$$

where $t = t^\ell$, ℓ is the last node of p, the distribution $\Pr$ is Poisson, with parameters governed by $\lambda_s^\ell T$, and

$$\begin{aligned} \Delta_s^\ell(\boldsymbol{\lambda}^\ell, n) = &\,\mathbb{E}_{\boldsymbol{n}^\ell}\left[\mathbb{E}_{\boldsymbol{X}^\ell}[G^\ell(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell)|\boldsymbol{n}^\ell]|n_s^\ell = n + 1\right] \\ &-\mathbb{E}_{\boldsymbol{n}^\ell}\left[\mathbb{E}_{\boldsymbol{X}^\ell}[G(\boldsymbol{X}^\ell, \boldsymbol{n}^\ell)|\boldsymbol{n}^\ell]|n_s^\ell = n\right]. \end{aligned} \tag{18}$$

The proof is in App. A. Compared to Obj. (6a), Obj. (15a) contains two layers of expectations and a different D-optimal design objective. 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.

5.2 Algorithm Overview

We follow the Frank-Wolfe variant for monotone continuous DR-submodular functionn maximization by Bian et al. [7] and Liu et al. [27], but deviate in estimating the gradients of objective U. The proposed algorithm is summarized in Alg. 1.

Frank-Wolfe Variant. Starting from $\boldsymbol{\lambda}(0) = \boldsymbol{0}$, the algorithm iterates over:

$$\boldsymbol{v}(k) = \arg\max_{\boldsymbol{v}\in\mathcal{D}}\langle \boldsymbol{v}, \widehat{\nabla U(\boldsymbol{\lambda}(k))}\rangle, \tag{19a}$$

$$\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \gamma\boldsymbol{v}(k), \tag{19b}$$

where $\widehat{\nabla U(\cdot)}$ is an estimator of the gradient ∇U, and γ is an appropriate stepsize, selected as in Alg. 1. We will further discuss how to estimate the gradient in Section 5.3. This algorithm achieves a $1 - \frac{1}{e}$ approximation guarantee, characterized by the following theorem:

Theorem 5.3. *Let $\lambda_{\text{MAX}} = \max_{\boldsymbol{\lambda}\in\mathcal{D}} \|\boldsymbol{\lambda}\|_1$. Then, for any $0 < \epsilon_0, \epsilon_1 < 1$, there exists $K = O(\frac{\epsilon_0}{P_{\text{TOT}}(|\mathcal{S}|-1)}\epsilon_1)$, $n' = O(\lambda_{\text{MAX}}T + \ln\frac{1}{\epsilon_1})$, $N_1 = N_2 = \Omega(\sqrt{\ln\frac{P_{\text{TOT}}K}{\epsilon_0}}\cdot(n'+1)TK)$, s.t., FW variant algorithm terminates in K iterations, use n' terms, and N_1 samples for $\boldsymbol{n}$, N_2 samples for $\boldsymbol{X}$ in estimator (23). Thus, with probability greater than $1-\epsilon_0$, the output solution $\boldsymbol{\lambda}(K)\in\mathcal{D}$ satisfies:*

$$U(\boldsymbol{\lambda}(K)) \geq (1-e^{\epsilon_1 - 1})\max_{\boldsymbol{\lambda}\in\mathcal{D}} U(\boldsymbol{\lambda}) - \epsilon_2, \tag{20}$$

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

The proof is in App. C. Our algorithm is based on the Frank-Wolfe variant from Bian et al. [7]. Similar to Thm. 2 in [27], our guarantee involves gradient estimation through truncating and sampling, due to our Poisson arrivals (see Asm. 1). However, to ensure our estimator has polynomial time complexity, we need to also sample from the Gaussian distribution. This requires leveraging a sub-Gaussian norm bound [33], and combining it with the aforementioned truncation and sampling techniques.

5.3 Gradient Estimation

We describe here how to produced an unbiased, polynomial-time estimator of our gradient $\widehat{\nabla U}$, which is presumably accessed by Eq. (19a). There are three difficulties to precisely compute the true gradient (17). (a) The outer sum involves infinite summation over $n^\ell_s \in \mathbb{N}$. (b) The outer expectation involves exponential sum in $|\mathcal{S}|-1$. (c) The inner expectation involves exponential sum in $\|\boldsymbol{n}^\ell\|$. The last difficulty comes from Gaussian sources, which differs from gradient estimation in [27].

To solve these difficulties, we (a) truncate the infinite summation while maintaining the quality of estimation through Poisson tail bound; (b) sample $\boldsymbol{n}^\ell$ (N_1 samples) according to Poisson distribution, parameterized by $\boldsymbol{\lambda}^\ell T$; and (c) sample $\boldsymbol{X}^\ell$ (N_2 samples) according to Gaussian distribution, which is determined by $\boldsymbol{n}^\ell$ sampled in (b).

Truncating. We truncate the infinite sum and utilize partial sum to estimate partial derivatives

$$\text{HEAD}^p_{s,t}(n') = \sum_{n=0}^{n'} \Delta^\ell_s(\boldsymbol{\lambda}^\ell, n)\cdot \Pr[n^\ell_s = n]\cdot T, \tag{21}$$

where $t = t^\ell$, l is the last node of p, $\Delta^\ell_s(\boldsymbol{\lambda}^\ell, n)$ is defined in Eq. (18), and n' is the truncating parameter. We know that Poisson has a subexponential tail bound, then, this HEAD is guaranteed to be within a constant factor from the true partial derivative by:

Lemma 5.4 (Lemma 4 in [27]). *For $h(u) = 2\frac{(1+u)\ln(1+u)-u}{u^2}$ and $n' \geq \lambda^\ell_s T$, we have:*

$$\frac{\partial U}{\partial \lambda^p_{s,t}} \geq \text{HEAD}^p_{s,t}(n') \geq (1-\Pr[n^\ell_s \geq n'+1])\frac{\partial U}{\partial \lambda^p_{s,t}}. \tag{22}$$

Sampling. When $n' \geq \lambda^\ell_s T$, we estimate $\Delta^\ell_s(\boldsymbol{\lambda}^\ell, n)$ by polynomial-time sampling:

$$\widehat{\frac{\partial U}{\partial \lambda^p_{s,t}}} = \sum_{n=0}^{n'}\widehat{\Delta^\ell_s(\boldsymbol{\lambda}^\ell, n)}\cdot\Pr[n^\ell_s = n]\cdot T, \tag{23}$$

where

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

$\boldsymbol{n}^j|_{n^{\ell,j}_s = n}$ means vector $\boldsymbol{n}^j$ with $n^{\ell,j}_s = n$, and N_1, N_2 are sampling parameters. At each iteration, we generate N_1 samples $\boldsymbol{n}^{\ell,j}$, $j = 1,\dots,N_1$ of the random vector $\boldsymbol{n}^\ell$ according to the Poisson distribution in Eq. (12), parameterized by the current solution vector $\boldsymbol{\lambda}^\ell T$. Having a sample $\boldsymbol{n}^{\ell,j}$, we could sample N_2 samples $\boldsymbol{X}^{\ell,j,k}$, $k = 1,\dots,N_2$ of random matrix $\boldsymbol{X}^{\ell,j} = [[\boldsymbol{x}^\ell_{s,i}]^{n^{\ell,j}_s}_{i=1}]_{s\in\mathcal{S}}$ according to the Gaussian distribution in Eq. (13). The distance between estimated partial derivative $\widehat{\frac{\partial U}{\partial \lambda}}$ and HEAD has the following guarantee:

Lemma 5.5. *For any $\delta \in (0,1)$, and $n' \geq \lambda^\ell_s T$,*

$$\left|\widehat{\frac{\partial U}{\partial \lambda^p_{s,t}}} - \text{HEAD}^p_{s,t}\right| \leq \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\text{MAX}}(\Sigma_\ell)c_s^2}{\sigma^2_{s,t}}), \tag{24}$$

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

The proof is in App. B. By Lems. 5.4 and 5.5, we instantly get the distance between the estimated and true gradient:

Lemma 5.6. *For any $\delta \in (0,1)$, and $n' \geq \lambda^\ell_s T$,*

$$\begin{aligned} -\gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\text{MAX}}(\Sigma_\ell)c_s^2}{\sigma^2_{s,t}}) &\leq \frac{\partial U}{\partial \lambda^p_{s,t}} - \widehat{\frac{\partial U}{\partial \lambda^p_{s,t}}} \leq \\ \gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\text{MAX}}(\Sigma_\ell)c_s^2}{\sigma^2_{s,t}}) &+ \Pr[n^\ell_s \geq n'+1]\frac{\partial U}{\partial \lambda^p_{s,t}}, \end{aligned} \tag{25}$$

with probability greater than $1 - 2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)} - |\mathcal{S}|n'\delta - (|\mathcal{S}|-1)\delta^p_{s,t}$,

Combining this estimated gradient in Eq. (23) with classic Frank-Wolfe variant [7], we propose Alg. 1 and establish Thm. 5.3.

6 DISTRIBUTED ALGORITHM

Implementing Alg. 1 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. (15). The algorithm performs a *primal dual gradient algorithm* over a modified Lagrangian to effectively find direction $\boldsymbol{v}$, defined in Eq. (19a), in a distributed fashion. The linearity of Eq. (19a) ensures convergence, while Thm. 5.3 ensures that the aggregate utility attained in steady state is within an $1-\frac{1}{e}$ factor from the optimal.

6.1 Algorithm Overview

Solving Prob. (15) in a distributed fashion requires decentralizing Eqs. (19a) and (19b). Decentralizing the latter is easy, Eq. (19b) can be executed via:

$$\lambda^p_{s,t}(k+1) = \lambda^p_{s,t}(k) + \gamma v^p_{s,t}(k), \tag{26}$$

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. (19a).

This is a linear program. Standard primal-dual distributed algorithms (see, e.g. [29, 36]) typically require strictly concave objectives. As our objective Eq. (19a) is not strictly concave, standard primal-dual algorithms would yield to harmonic oscillations and will not converge to a optimal point [11]. An additional challenge arises from the multicast link capacity constraints in Eq. (8): these involve a max function, that is non-differentiable. The maximum could be replaced by a set of multiple inequality constraints, but this approach would not scale well, introducing a new dual variable per additional constraint.

To address the first challenge, we follow Feijer and Paganini [11], and replacing constraints of the form $u \le 0$ with $\phi(u) \le 0$, where $\phi(u) = e^u - 1$. In order to obtain a scalable differentiable Lagrangian, we use the approach in [29, 36]: we replace the multicast link capacity constraints (8) by

$$\sum_{s\in\mathcal{S},t\in\mathcal{T}} \left(\sum_{p\in\mathcal{P}_{s,t}:e\in p} (v^p_{s,t})^\theta \right)^{\frac{1}{\theta}} \le \mu^e, \tag{27}$$

for each link $e \in \mathcal{E}$. Note that this is tantamount to approximating $\|\cdot\|_\infty$ with $\|\cdot\|_\theta$; the latter indeed converges to $\|\cdot\|_\infty$ as $\theta \to \infty$, making the approximation arbitrarily accurate.

Combining these two approaches together, the Lagrangian for the modified problem is:

$$\begin{aligned} L(\boldsymbol{v},\boldsymbol{q},\boldsymbol{r},\boldsymbol{u}) = \langle \boldsymbol{v}, \widehat{\nabla U(\boldsymbol{\lambda})} \rangle - \sum_{e\in\mathcal{E}} q_e(e^{g_e(\boldsymbol{v})} - 1) - \sum_{s\in\mathcal{S},t\in\mathcal{T}} \\ r_{s,t}(e^{g_{s,t}(\boldsymbol{v})} - 1) - \sum_{s\in\mathcal{S},t\in\mathcal{T}} \sum_{p\in\mathcal{P}_{s,t}} u^p_{s,t}(e^{g^p_{s,t}(\boldsymbol{v})} - 1), \end{aligned} \tag{28}$$

where

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

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 Lagrange multipliers. Intuitively, L penalizes the infeasibility of network constraints.

We apply a primal dual gradient algorithm over this modified Lagrangian to decentralize the Eq. (19a). The algorithm iteratively maximizes primal variable $\boldsymbol{v}$ by gradient ascent on the modified Lagrangian, and minimizes dual variables $\boldsymbol{q}$, $\boldsymbol{r}$ and $\boldsymbol{u}$ by gradient descent. In particular, at iteration $\tau+1$, primal variables are adjusted via:

$$v^p_{s,t}(\tau+1) = v^p_{s,t}(\tau) + m^p_{s,t} \nabla_{v^p_{s,t}} L(\tau), \tag{29}$$

and dual variables are adjusted via:

$$q_e(\tau+1) = q_e(\tau) + k_e(\nabla_{q_e} L(\tau))^+_{q_e(\tau)}, \tag{30a}$$

$$r_{s,t}(\tau+1) = r_{s,t}(\tau) + h_{s,t}(\nabla_{r_{s,t}} L(\tau))_{q_e(\tau)}{}^+_{r_{s,t}(\tau)}, \tag{30b}$$

$$u^p_{s,t}(\tau+1) = u^p_{s,t}(\tau) + w^p_{s,t}(\nabla_{u^p_{s,t}} L(\tau))_{q_e(\tau)}{}^+_{u^p_{s,t}(\tau)}, \tag{30c}$$

for all 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

$$(y)^+_x = \begin{cases} y, & x > 0, \\ \max(y,0), & x \le 0. \end{cases} \tag{31}$$

Thes operations can indeed be distributed across the network, as we describe below. The following theorem states the convergence of this modified primal dual gradient algorithm.

Theorem 6.1. *[Theorem 11 in [11]] Let $\boldsymbol{v}^*_\theta$ be the optima to Eq.* (19a) *over $\mathcal{D}_\theta$, where $\mathcal{D}_\theta$ is $\mathcal{D}$ with* (8) *replaced by* (27). *The trajectories of the modified primal–dual gradient algorithm (Alg. 3), with constant stepsize, converges to the optima $\boldsymbol{v}^*_\theta$.*

Let $\boldsymbol{v}^*$ be the optima to Eq. (19a). $\lim_{\theta\to\infty} \boldsymbol{v}^*_\theta \to \boldsymbol{v}^*$, as $\mathcal{D}_\theta \to \mathcal{D}$ when $\theta \to \infty$. Thus, our distributed algorithm preserves $1 - \frac{1}{e}$ approximation factor from the optimal as stated in Thm. 5.3, when choosing an appropriate θ.

6.2 Expanding Primal and Dual Steps

Before we describe how to distribute the execution of the primal dual algorithm in Eqs. (29) and (30), we first expand the primal dual steps, to indicate the specific operations they entail.

At iteration $\tau+1$, for each primal variable direction v of source $s \in \mathcal{S}$, type $t \in \mathcal{T}$, and path $p \in \mathcal{P}_{s,t}$:

$$\begin{aligned} v^p_{s,t}(\tau+1) = v^p_{s,t}(\tau) + m^p_{s,t} \Bigg(\nabla_{\lambda^p_{s,t}} U(\boldsymbol{\lambda}) - \sum_{e\in p} q_e(\tau) \cdot \\ \exp\left(\sum_{s'\in\mathcal{S},t'\in\mathcal{T}} (v^e_{s',t'}(\tau))^{\frac{1}{\theta}} - \mu^e \right) (v^e_{s,t}(\tau))^{\frac{1-\theta}{\theta}} (v^p_{s,t}(\tau))^{\theta-1} \\ -r_{s,t} \exp\left(\sum_{p\in\mathcal{P}_{s,t}} v^p_{s,t}(\tau) - \lambda_{s,t} \right) + u^p_{s,t} \exp(-v^p_{s,t}(\tau)) \Bigg). \end{aligned} \tag{32}$$

For each dual variable q of edge $e \in \mathcal{E}$:

$$q_e(\tau+1) = q_e(\tau) + k_e \left(\exp\left(\sum_{s\in\mathcal{S},t\in\mathcal{T}} (v^e_{s,t}(\tau))^{\frac{1}{\theta}} - \mu^e \right) - 1 \right)^+_{q_e(\tau)}, \tag{33}$$

where

$$v^e_{s,t}(\tau) = \sum_{p\in\mathcal{P}_{s,t}:e\in p} (v^p_{s,t}(\tau))^\theta. \tag{34}$$

For each dual variable r of source $s \in \mathcal{S}$, and type $t \in \mathcal{T}$:

$$r_{s,t}(\tau+1) = r_{s,t}(\tau) + h_{s,t} \left(\exp(\sum_{p\in\mathcal{P}_{s,t}} v^p_{s,t}(\tau) - \lambda_{s,t}) - 1 \right)^+_{r_{s,t}(\tau)}, \tag{35}$$

For each dual variable u of source $s \in \mathcal{S}$, and type $t \in \mathcal{T}$, and path $p \in \mathcal{P}_{s,t}$:

$$u^p_{s,t}(\tau+1) = u^p_{s,t}(\tau) + w^p_{s,t} \left(\exp(-v^p_{s,t}(\tau)) - 1 \right)^+_{u^p_{s,t}(\tau)}. \tag{36}$$

With these primal dual gradient updates, we now implement the overall distributed algorithm.

Algorithm 2: Distributed Frank-Wolfe Variant

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

Algorithm 3: Primal Dual Gradient Algorithm

```
Input: Rates $\boldsymbol{\lambda}$.
Output: Direction $\boldsymbol{v}$.
1 Initialize direction $\boldsymbol{v}(0) = \mathbf{0}$, and Lagrange multipliers $\boldsymbol{q}(0)$, $\boldsymbol{r}(0), \boldsymbol{u}(0) = \mathbf{0}$.
2 foreach learner $\ell \in \mathcal{L}$ do
3     Send control messages carrying $\widehat{\nabla_{\lambda_{s,t}^p} U(\boldsymbol{\lambda}^\ell)}$ calculated by Eq. (23) downstream over $p$.
4 for $\tau = 1, 2, \ldots$ do
5     foreach source $s \in \mathcal{S}$ do
6         Generate features carrying $v_{s,t}^p$ upstream.
7     foreach edge $e \in \mathcal{E}$ do
8         Calculate $v_{s,t}^e$ using fetched $v_{s,t}^p$ by Eq. (34).
9     foreach learner $\ell \in \mathcal{L}$ do
10        Send control messages downstream and collect $q_e$ and $v_{s,t}^e$ from traversed edges.
11    foreach edge $e \in \mathcal{E}$ do
12        Update $q_e$ using calculated $v_{s,t}^e$ by Eq. (33).
13    foreach source $s \in \mathcal{S}$ do
14        Update $r_{s,t}$ using maintained $v_{s,t}^p$ by Eq. (35).
15        Update $u_{s,t}^p$ using maintained $v_{s,t}^p$ by Eq. (36).
16        Update $v_{s,t}^p$ using received $\widehat{\nabla_{\lambda_{s,t}^p} U(\boldsymbol{\lambda}^\ell)}$, $q_e$, $v_{s,t}^e$, and maintained $v_{s,t}^p$ by Eq. (32).
17 return $v_{s,t}^p$ from each source $s$
```

6.3 Distributed FW Implementation Details

We conclude by giving the full implementation details of the FW algorithm and, in particular, the primal dual steps in Eqs. (29) and (30), describing the state maintained by every node, the messages exchanged, and the constituent state adaptations.

Starting from $\boldsymbol{\lambda}(0) = \mathbf{0}$, the algorithm iterates over:

(1) Each source node $s \in \mathcal{S}$ finds direction $v_{s,t}^p(k)$ for all $t \in \mathcal{T}$, and $p \in \mathcal{P}_{s,t}$ by primal dual gradient algorithm.
(2) Each source node s updates $\lambda_{s,t}^p(k+1)$ using $v_{s,t}^p(k)$ for all $t \in \mathcal{T}$, and $p \in \mathcal{P}_{s,t}$ by executing Eq. (26).

We describe the first step in more detail. Every edge $e \in \mathcal{E}$ maintains (a) Lagrange multiplier q_e, and (b) auxiliary variable $v_{s,t}^e$ for all $s \in \mathcal{S}$, $t \in \mathcal{T}$. Every source $s \in \mathcal{S}$ maintains (a) direction $v_{s,t}^p$, for all $t \in \mathcal{T}$, $p \in \mathcal{P}_{s,t}$, (b) Lagrange multipliers $u_{s,t}^p$, 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 $\mathbf{0}$. Given the rates $\boldsymbol{\lambda}^\ell$, each learner estimates the gradient $\widehat{\nabla_{\lambda_{s,t}^p} U(\boldsymbol{\lambda}^\ell)}$ by Eq. (23). Control messages carrying $\widehat{\nabla_{\lambda_{s,t}^p} U(\boldsymbol{\lambda}^\ell)}$ are generated and propagated over the path p in the reverse direction to sources. Sources finally obtain $\widehat{\nabla_{\lambda_{s,t}^p} U(\boldsymbol{\lambda}^\ell)}$. 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 $v_{s,t}^p$. Every time it traverses an edge $e \in \mathcal{E}$, edge e fetches $v_{s,t}^p$.
(2) After fetching all needed $v_{s,t}^p$, the edge $e \in \mathcal{E}$ calculates the auxiliary variables $v_{s,t}^e(\tau)$ for all $s \in \mathcal{S}$, and $t \in \mathcal{T}$ by executing (34).
(3) A control message is generated from learner $\ell \in \mathcal{L}$ and backpropoagted over the path p in the reverse direction until reaching the source. Every time traversing an edge $e \in \mathcal{E}$, the control message collects q_e and $v_{s,t}^e$. The source will obtain these q_e and $v_{s,t}^e$.
(4) After visited by all control messages, the edge $e \in \mathcal{E}$ updates the Lagrangian multiplier $q_e(\tau+1)$ using calculated $v_{s,t}^e$ by executing (33).
(5) Once obtaining $\widehat{\nabla_{\lambda_{s,t}^p} U(\boldsymbol{\lambda}^\ell)}$, q_e and $v_{s,t}^e$, the source updates the Lagrangian multiplier $r_{s,t}(\tau+1)$ using maintained $v_{s,t}^p$ by executing Eq. (35), for all $t \in \mathcal{T}$, updates $u_{s,t}^p(\tau+1)$ using maintained $v_{s,t}^p$ by executing Eq. (36), for all $t \in \mathcal{T}$, and $p \in \mathcal{P}_{s,t}$, and updates the direction $v_{s,t}^p(\tau+1)$ using received $\widehat{\nabla_{\lambda_{s,t}^p} U(\boldsymbol{\lambda}^\ell)}$, q_e, $v_{s,t}^e$, and maintained $v_{s,t}^p$ by executing Eq. (32), for all $t \in \mathcal{T}$, and $p \in \mathcal{P}_{s,t}$.

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

7 EXTENSIONS

We can also solve Prob. (15) by *projected gradient ascent* (PGA) [17]. Decentralization reduces then to a primal-dual algorithm [11, 36] 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 how this could be done below.

Projected Gradient Ascent. Starting from $\boldsymbol{\lambda}(0) = \mathbf{0}$, the algorithm iterates over:

$$\boldsymbol{v}(k) = \boldsymbol{\lambda}(k) + \gamma \widehat{\nabla U(\boldsymbol{\lambda}}(k) \tag{37a}$$
$$\boldsymbol{\lambda}(k+1) = \Pi_{\mathcal{D}}(\boldsymbol{v}(k)) \tag{37b}$$

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 projection. Our gradient estimator in Sec. 5.3 would again be used here to compute $\widehat{\nabla U(\boldsymbol{\lambda}^k)}$. Note that, to achieve the same quality of gradient estimator, the PGA usually takes longer time compared to FW algorithm. This comes from larger $\lambda_s^\ell(k)$, thus, larger truncating parameter n', during the iteration k (see also Sec. 8.2 for how we set algorithm parameters).

Table 1: 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

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 of [17] with the gradient estimation bounds in Sec. 5.3.

Distributed Projected Gradient Ascent. Similar to distributed FW, we can easily decentralize Eq. (37a). Eq. (37b) has a strictly convex objective, so that we can directly decentralize it through a standard primal-dual algorithm with approximated multicast link capacity constraints, as in Eq. (27). Convergence then is directly implied by Thm. 5 in [11].

8 NUMERICAL EVALUATION

To evaluate our proposed algorithms and competitors, we perform experiments over five synthetic graphs, namely, Erdős-Rényi (ER), balanced tree (BT), hypercube (HC), grid_2d (grid), small-world (SW) [25], and three backbone network topologies: Deutsche Telekom (DT), GEANT, Abilene [34]. The graph parameters of different topologies are shown in Tab. 1.

8.1 Settings

For each network, we uniformly at random (u.a.r.) select $|\mathcal{L}|$ learners and $|\mathcal{S}|$ data sources. Each edge $e = (u, v) \in \mathcal{E}$ has a link capacity μ^e and types $\mathcal{T}$ as indicated in Tab. 1. 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 Σ_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. Labels y of each type and each source are generated using ground-truth models per type, as discussed below, with Gaussian noise, whose variance $\sigma_{s,t}$ is chosen u.a.r. from 0.5 to 1. For each source, the paths set consists the 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 with mean and covariance generated 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 Σ_{t^ℓ}) 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.

8.2 Competitor Algorithms

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

- MaxTP: This maximizes the aggregate total useful incoming traffic rates (throughput) of learners, i.e.:
$$\max_{\boldsymbol{\lambda}\in\mathcal{D}} : U_{\text{MaxTP}}(\boldsymbol{\lambda}) = \sum_{\ell\in\mathcal{L}}\sum_{s\in\mathcal{S}} \lambda_s^\ell. \tag{38}$$
- MaxFair: This maximizes the aggregate α-fair utilities [36] of the total useful incoming traffic at learners, i.e.:
$$\max_{\boldsymbol{\lambda}\in\mathcal{D}} : U_{\text{MaxFair}}(\boldsymbol{\lambda}) = \sum_{\ell\in\mathcal{L}}\Big(\sum_{s\in\mathcal{S}} \lambda_s^\ell\Big)^{1-\alpha}/(1-\alpha). \tag{39}$$
We set $\alpha = 2$.
- FW: This is Alg. 1, as proposed in Sec. 5.
- PGA: This is the algorithm we proposed in Sec. 7.

We also implement their corresponding distributed version: DMaxTP, DMaxFair, DFW (Alg. 2 in Sec. 6), and DPGA (see Sec. 7). Note that, the objectives of MaxTP Eq. (38) and MaxFair Eq. (39) 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.

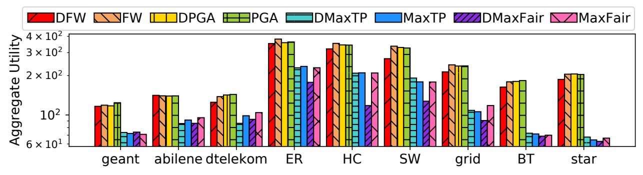
Parameters. We run FW/DFW and PGA/DPGA for $K = 50$ iterations with step size $\delta = 0.02$ w.r.t. the outer iteration. In each iteration, we estimate the gradient according to Eq. (23) with $N_1 = 50$, $N_2 = 50$, and $n' = \max\{\lceil 2\max_{\ell,s}\lambda_s^\ell T\rceil, 10\}$, where λ_s^ℓ is given by the current solution. We run inner primal-dual gradient algorithm for 1000 iterations. We set parameter $\theta = 10$ when approximating the max function via Eq. (27). We choose the best stepsize for distributed primal-dual algorithms by comparing the performance metrics (Aggregate Utility and Infeasibility, defined in Sec. 8.3), between cantralized and distributed versions of each algorithm under different stepsizes. We further discuss the impact of stepsizes in Sec. 8.4.

8.3 Performance Metrics

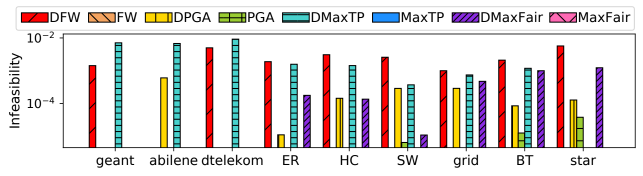
To evaluate the performance of algorithms, we use *Aggregate Utility*, defined in Eq. (15a) as one metric. Note that, aggregate utility involves summation with infinite support, we thus need to resort to sampling to estimate it; we set $N_1 = 100$ and $N_2 = 100$. Also, we define an *Estimation Error* to measure the model estimation quality. Formally, it is defined as:

$$\frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}}\frac{\|\hat{\boldsymbol{\beta}}^\ell_{\text{MAP}} - \boldsymbol{\beta}^\ell\|}{\|\boldsymbol{\beta}^\ell\|} = \frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}}\|(\boldsymbol{X}^{\ell\top}\Sigma^{\ell-1}\boldsymbol{X}^\ell + \Sigma_\ell^{-1})^{-1}\cdot (\boldsymbol{X}^{\ell\top}\Sigma^{\ell-1}\boldsymbol{y}^\ell + \Sigma_\ell^{-1}\boldsymbol{\beta}_0^\ell) - \boldsymbol{\beta}^\ell\|/\|\boldsymbol{\beta}^\ell\|, \tag{40}$$

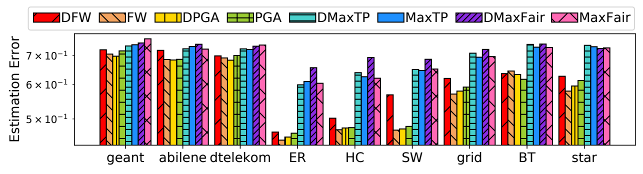
following the equation of MAP estimation Eq. (2). We average over 2500 realizations of the number of data arrived at the learner $\{\boldsymbol{n}^\ell\}_{\ell\in\mathcal{L}}$ and features $\{\boldsymbol{X}^\ell\}_{\ell\in\mathcal{L}}$, and 20 realizations of $\{\boldsymbol{\beta}^\ell\}_{\ell\in\mathcal{L}}$ to calculate this average. Finally, we define an *Infeasibility* to measure the feasibility of solutions, as primal dual gradient algorithm



(a) Aggregate Utility



(b) Infeasibility



(c) Estimation Error

Figure 2: Aggregate utility, infeasibility and estimation error in different networks. We can observe that 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. 1. Furthermore, their performances are close to their centralized version: FW and PGA, with an acceptable Infeasibility ~ 0.01.

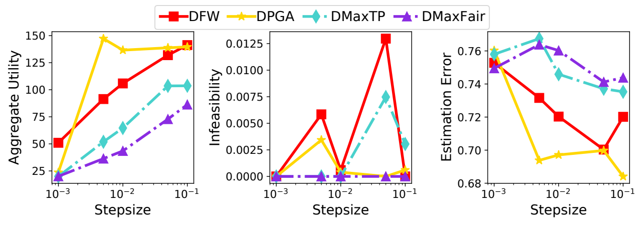
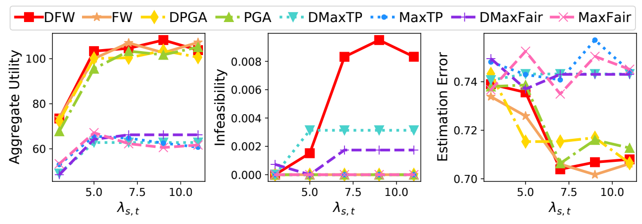


Figure 3: Effect of stepsize in primal dual gradient algorithm over topology Abilene. Larger stepsizes lead to better performance, and DFW and DPGA are always the best in terms of both utility and estimation error. However, stepsizes above 10^{-1} lead to numerical instability.

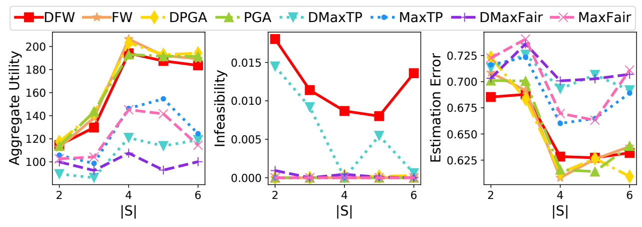
used in distributed algorithms do not guarantee feasibility. It averages violations of constraints (7)-(10) over the total number of constraints.

8.4 Results

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



(a) Varying source rate



(b) Varying source set size

Figure 4: Varying source rate over GEANT, and source set size over Dtelekom. Greater source rate and source set size both reflect learners receive more data. This leads to higher aggregate utility, and lower estimation error.

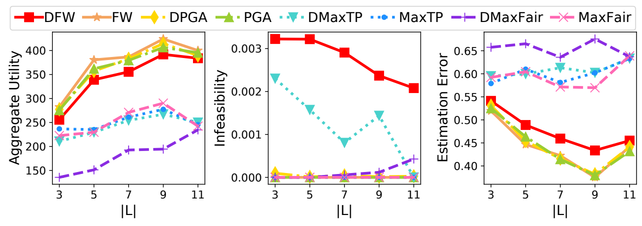


Figure 5: Varying learner set size over topology ER. Aggregate utility increases, while estimation error decreases, with more learners.

distributed algorithms perform closed to their corresponding centralized algorithms (c.f. Thm. 6.1). All of algorithms have a low Infeasibility, which is less than 0.01. Such violations over the constraints, which incurred by primal-dual gradient algorithm, are expected in distributed algorithms, since Lagrangian employs soft constraints.

Effect of Stepsizes. We study the effect of stepsize in primal dual gradient algorithm for distributed algorithms over Abilene, 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 w.r.t. infeasibility. However, if the stepsize is too large, the algorithm 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 source rate $\lambda_{s,t}$ over topology GEANT. As shown in Fig. 4a, when sources rates increase, the aggregate utility first increases very fast and then tapers off. Higher source rates indicate more data received at learners, and thus 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 increment. The same interpretation applies to the estimation error and infeasibility. We observe the same changing patterns when varying source set size $|\mathcal{S}|$ over topology `Dtelekom`, shown in Fig. 4b, since increasing sources also indicates learners receive more data. Curve changes are not as smooth as increasing rates, because varying source sets also changes available paths, corresponding link bandwidth utilizations, etc.

Varying Learner Set Size. Then, we evaluate how learner set size $|\mathcal{L}|$ affect over topology `ER`. The Fig. 5 shows that when increasing the number of learners, the aggregate utility and infeasibility increase and estimation error decreases. This is because D-optimal design is a DR-submodular function, which has a similar effect as concavity: if the total data arrival is fixed (source is fixed), even distribution of the data features leads to larger aggregate utility.

9 CONCLUSION

We generalize the experimental design networks by considering Gaussian sources and multicast transmissions. A poly-time 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 gradient based on outdated information, so that they may communicate at different times and with different frequencies [28]. Another interesting direction is to estimate gradient through shadow price [21, 24], instead of sampling. Furthermore, how model trained by one learner benefits other training tasks in experimental design networks is a worthwhile topic to study.

REFERENCES

[1] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. 2017. Mobile edge computing: A survey. *IEEE Internet of Things Journal* 5, 1 (2017), 450–465.
[2] Vito Albino, Umberto Berardi, and Rosa Maria Dangelico. 2015. Smart cities: Definitions, dimensions, performance, and initiatives. *Journal of urban technology* 22, 1 (2015), 3–21.
[3] Sulaiman A Alghunaim and Ali H Sayed. 2020. Linear convergence of primal–dual gradient methods and their performance in distributed optimization. *Automatica* 117 (2020), 109003.
[4] Noga Alon and Joel H Spencer. 2004. *The probabilistic method.* John Wiley & Sons.
[5] Dimitri Bertsekas and John Tsitsiklis. 2015. *Parallel and distributed computation: numerical methods.* Athena Scientific.
[6] Dimitri P Bertsekas. 1999. *Nonlinear programming.* Athena scientific Belmont.
[7] Andrew An Bian, Baharan Mirzasoleiman, Joachim Buhmann, and Andreas Krause. 2017. Guaranteed non-convex optimization: Submodular maximization over continuous domains. In *Artificial Intelligence and Statistics.* PMLR, 111–120.
[8] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge university press.
[9] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* 40, 6 (2011), 1740–1766.
[10] Clément L Canonne. 2017. A short note on Poisson tail bounds. http://www.cs.columbia.edu/~ccanonne/files/misc/2017-poissonconcentration.pdf
[11] Diego Feijer and Fernando Paganini. 2010. Stability of primal–dual gradient dynamics and applications to network optimization. *Automatica* 46, 12 (2010), 1974–1981.
[12] Yuval Filmus and Justin Ward. 2014. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM J. Comput.* 43, 2 (2014), 514–542.
[13] Robert G Gallager. 2013. *Stochastic Processes: Theory for Applications.* Cambridge University Press.
[14] Nicolas Gast, Stratis Ioannidis, Patrick Loiseau, and Benjamin Roussillon. 2020. Linear regression from strategic data sources. *ACM Transactions on Economics and Computation (TEAC)* 8, 2 (2020), 1–24.
[15] Yuan Guo, Jennifer Dy, Deniz Erdogmus, Jayashree Kalpathy-Cramer, Susan Ostmo, J Peter Campbell, Michael F Chiang, and Stratis Ioannidis. 2019. Accelerated experimental design for pairwise comparisons. In *Proceedings of the 2019 SIAM International Conference on Data Mining.* SIAM, 432–440.
[16] Yuan Guo, Peng Tian, Jayashree Kalpathy-Cramer, Susan Ostmo, J Peter Campbell, Michael F Chiang, Deniz Erdogmus, Jennifer G Dy, and Stratis Ioannidis. 2018. Experimental Design under the Bradley-Terry Model.. In *IJCAI.* 2198–2204.
[17] Hamed Hassani, Mahdi Soltanolkotabi, and Amin Karbasi. 2017. Gradient methods for submodular maximization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 5843–5853.
[18] Thibaut Horel, Stratis Ioannidis, and S Muthukrishnan. 2014. Budget feasible mechanisms for experimental design. In *Latin American Symposium on Theoretical Informatics.* Springer, 719–730.
[19] Roger A Horn and Charles R Johnson. 2012. *Matrix analysis.* Cambridge university press.
[20] Xun Huan and Youssef M Marzouk. 2013. Simulation-based optimal Bayesian experimental design for nonlinear systems. *J. Comput. Phys.* 232, 1 (2013), 288–317.
[21] Stratis Ioannidis, Augustin Chaintreau, and Laurent Massoulié. 2009. Optimal and scalable distribution of content updates over a mobile social network. In *IEEE INFOCOM 2009.* IEEE, 1422–1430.
[22] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning.* Vol. 112. Springer.
[23] Frank P Kelly. 2011. *Reversibility and stochastic networks.* Cambridge University Press.
[24] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* 49, 3 (1998), 237–252.
[25] Jon Kleinberg. 2000. The small-world phenomenon: An algorithmic perspective. In *STOC.*
[26] Andreas Krause and Daniel Golovin. 2014. Submodular function maximization.
[27] Yuezhou Liu, Yuanyuan Li, Lili Su, Edmund Yeh, and Stratis Ioannidis. 2022. Experimental design networks: A paradigm for serving heterogeneous learners under networking constraints. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications.* IEEE, 210–219.
[28] Steven H Low and David E Lapsley. 1999. Optimization flow control. I. Basic algorithm and convergence. *IEEE/ACM Transactions on networking* 7, 6 (1999), 861–874.
[29] Desmond S Lun, Niranjan Ratnakar, Muriel Médard, Ralf Koetter, David R Karger, Tracey Ho, Ebad Ahmed, and Fang Zhao. 2006. Minimum-cost multicast over coded packet networks. *IEEE Transactions on information theory* 52, 6 (2006), 2608–2623.
[30] Mehdi Mohammadi and Ala Al-Fuqaha. 2018. Enabling cognitive smart cities using big data and machine learning: Approaches and challenges. *IEEE Communications Magazine* 56, 2 (2018), 94–101.
[31] Angelia Nedić and Asuman Ozdaglar. 2009. Subgradient methods for saddle-point problems. *Journal of optimization theory and applications* 142, 1 (2009), 205–228.
[32] Friedrich Pukelsheim. 2006. *Optimal design of experiments.* Society for Industrial and Applied Mathematics.
[33] Alessandro Rinaldo. 2019. Sub-Gaussian vectors and bound for the their norm. https://www.stat.cmu.edu/~arinaldo/Teaching/36709/S19/Scribed_Lectures/Feb21_Shenghao.pdf
[34] Dario Rossi and Giuseppe Rossini. 2011. *Caching performance of content centric networks under multi-path routing (and more).* Technical Report. Telecom ParisTech.
[35] Tasuku Soma and Yuichi Yoshida. 2015. A Generalization of Submodular Cover via The Diminishing Return Property On The Integer Lattice. *Advances in Neural Information Processing Systems* 28 (2015), 847–855.
[36] Rayadurgam Srikant and Tamer Başar. 2004. *The mathematics of Internet congestion control.* Springer.
[37] Maxim Sviridenko, Jan Vondrák, and Justin Ward. 2017. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Mathematics of Operations Research* 42, 4 (2017), 1197–1218.
[38] Georgios Tychogiorgos, Athanasios Gkelias, and Kin K Leung. 2013. A non-convex distributed optimization framework and its application to wireless ad-hoc networks. *IEEE Transactions on Wireless Communications* 12, 9 (2013), 4286–4296.
[39] Bo Yang, Xuelin Cao, Xiangfang Li, Qinqing Zhang, and Lijun Qian. 2019. Mobile-edge-computing-based hierarchical machine learning tasks distribution for IIoT. *IEEE Internet of Things Journal* 7, 3 (2019), 2169–2180.

A PROOF OF THEOREM 5.2

In general, linear combinations of submodular functions with positive weights are also submodular. However, proving the DR-submodularity of the objective poses a challenge precisely because the set of features $\boldsymbol{X}^\ell$ observed by a learner depends on $\boldsymbol{n}^\ell$, the number of samples received. As a result, such an "averaging" argument cannot be directly applied here.

To resolve this, we consider an equivalent 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 [27] and telescoping sum, we get:

Lemma A.1. *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. We eliminate the superscript ℓ of the notations for brevity. For $\boldsymbol{n} \in \mathbb{N}^{|\mathcal{S}|}$, we have:

$$\begin{aligned}
&G(\widetilde{\boldsymbol{X}}, \boldsymbol{n}+\boldsymbol{e}_{s'})) - G(\widetilde{\boldsymbol{X}}, \boldsymbol{n}) \\
&= \log\det\Big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^\top}{\sigma_s^2} + \frac{\boldsymbol{x}_{s',n_{s'}+1}\boldsymbol{x}_{s',n_{s'}+1}^\top}{\sigma_{s'}^2} + \Sigma_0^{-1}\Big) - \\
&\quad \log\det\Big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^\top}{\sigma_s^2} + \Sigma_0^{-1}\Big) \\
&= \log\det\Big(\boldsymbol{I}_d + \frac{\boldsymbol{x}_{s',n_{s'}+1}\boldsymbol{x}_{s',n_{s'}+1}^\top}{\sigma_{s'}^2}\Big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^\top}{\sigma_s^2} + \Sigma_0^{-1}\Big)^{-1}\Big) \\
&= \log\det\Big(\boldsymbol{I}_1 + \frac{1}{\sigma_{s'}^2}\boldsymbol{x}_{s',n_{s'}+1}^\top\boldsymbol{A}(\boldsymbol{n})\boldsymbol{x}_{s',n_{s'}+1}\Big) \\
&= \log\Big(1 + \frac{1}{\sigma_{s'}^2}\boldsymbol{x}_{s',n_{s'}+1}^\top\boldsymbol{A}(\boldsymbol{n})\boldsymbol{x}_{s',n_{s'}+1}\Big)
\end{aligned}$$

where $\boldsymbol{A}(\boldsymbol{n}) = \big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^\top}{\sigma_s^2} + \Sigma_0^{-1}\big)^{-1}$ 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

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

where $\boldsymbol{A}(\boldsymbol{n}) = \big(\sum_{s\in\mathcal{S}}\sum_{i=1}^{n_s}\frac{\boldsymbol{x}_{s,i}\boldsymbol{x}_{s,i}^\top}{\sigma_s^2} + \Sigma_0^{-1}\big)^{-1}$ 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 $\boldsymbol{n} \le \boldsymbol{m}$, which leads to $G(\widetilde{\boldsymbol{X}}, \boldsymbol{n}+k\boldsymbol{e}_s) - G(\widetilde{\boldsymbol{X}}, \boldsymbol{n}) \ge G(\widetilde{\boldsymbol{X}}, \boldsymbol{m}+k\boldsymbol{e}_s) - G(\widetilde{\boldsymbol{X}}, \boldsymbol{m})$. □

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 A.2. *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. We again eliminate the superscript ℓ of the notations for brevity. Let's consider a projection

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

where

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

Then, consider a function

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

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

Consider $g_{n_0}(\boldsymbol{n}) = \mathbb{E}_{\boldsymbol{X}}[G_{n_0}(\widetilde{\boldsymbol{X}}, \boldsymbol{n})|\boldsymbol{n}]$. As an expectation of 'finite $\boldsymbol{X}$', $g_{n_0}(\boldsymbol{n})$ is still DR-submodular, i.e., $\forall\, \boldsymbol{n}, \boldsymbol{m} \in \mathbb{N}^{|\mathcal{S}|}$, $\boldsymbol{n} \le \boldsymbol{m}$ and $\forall\, n_0 \in \mathbb{N}$, we have $g_{n_0}(\boldsymbol{n}+k\boldsymbol{e}_s) - g_{n_0}(\boldsymbol{n}) \ge g_{n_0}(\boldsymbol{m}+k\boldsymbol{e}_s) - g_{n_0}(\boldsymbol{m})$. 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 $g(\boldsymbol{n}+k\boldsymbol{e}_s) - g(\boldsymbol{n}) \ge g(\boldsymbol{m}+k\boldsymbol{e}_s) - g(\boldsymbol{m})$. Monotonicity follows similarly. □

Then, we establish the positivity of the gradient and non-positivity of the Hessian of U following Thm. 1 in [27] to prove Thm. 5.2.

Proof. By the law of total expectation:

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

Thus the first partial derivatives are:

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

where $t = t^\ell$ and ℓ is the last node of p, holds by monotonicity of g. Thus U is monotone-increasing. 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 $\ell \ne \ell'$, we have:

$$\frac{\partial^2 U}{\partial\lambda_{s,t}^p\partial\lambda_{s',t'}^{p'}} = 0. \tag{41}$$

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

$$\frac{\partial^2 U}{\partial \lambda^p_{s,t} \partial \lambda^{p'}_{s,t}} = \frac{\partial^2 U^\ell}{\partial (\lambda^p_{s,t})^2} = \Delta^\ell_s(\boldsymbol{\lambda}^\ell, 0) \cdot (-T^2) e^{-\lambda^\ell_s T} + \sum_{n=1}^{\infty} \Delta^\ell_s(\boldsymbol{\lambda}^\ell, n) \cdot$$
$$\left(\frac{(\lambda^\ell_s)^{n-1} T^{n+1}}{(n-1)!} - \frac{(\lambda^\ell_s)^n T^{n+2}}{n!}\right) e^{-\lambda^\ell_s T}$$
$$= \sum_{n=0}^{\infty} \Big(\Delta^\ell_s(\boldsymbol{\lambda}^\ell, n+1) - \Delta^\ell_s(\boldsymbol{\lambda}^\ell, n)\Big) \cdot \Pr[n^\ell_s = n] \cdot T^2 \le 0,$$

by the submodularity of G. For $\ell = \ell'$ and $s \ne s'$, which implies $t = t'$ and $p \ne p'$:

$$\frac{\partial^2 U}{\partial \lambda^p_{s,t} \partial \lambda^{p'}_{s',t}} = \frac{\partial^2 U^\ell}{\partial \lambda^p_{s,t} \partial \lambda^{p'}_{s',t}} = \mathbb{E}\left[g(\boldsymbol{n}) | n^\ell_s = 0, n^\ell_{s'} = 0\right] \cdot (-T)^2 e^{-\lambda^\ell_s T}$$
$$e^{-\lambda^\ell_{s'} T} + \sum_{k=1}^{\infty} \mathbb{E}\left[g(\boldsymbol{n}) | n^\ell_s = 0, n^\ell_{s'} = k\right] (-T) e^{-\lambda^\ell_s T} \Big(\frac{(\lambda^\ell_{s'})^{k-1} T^k}{(k-1)!} -$$
$$\frac{(\lambda^\ell_{s'})^k T^{k+1}}{k!}\Big) e^{-\lambda^\ell_{s'} T} + \sum_{n=1}^{\infty} \mathbb{E}\left[g(\boldsymbol{n}) | n^\ell_s = n, n^\ell_{s'} = 0\right] \Big(\frac{(\lambda^\ell_s)^{n-1} T^n}{(n-1)!} -$$
$$\frac{(\lambda^\ell_s)^n T^{n+1}}{n!}\Big) e^{-\lambda^\ell_s T} (-T) e^{-\lambda^\ell_{s'} T} + \sum_{n=1}^{\infty} \sum_{k=1}^{\infty} \mathbb{E}\left[g(\boldsymbol{n}) | n^\ell_s = n, n^\ell_{s'} = k\right] \cdot$$
$$\Big(\frac{(\lambda^\ell_s)^{n-1} T^n}{(n-1)!} - \frac{(\lambda^\ell_s)^n T^{n+1}}{n!}\Big) e^{-\lambda^\ell_s T} \Big(\frac{(\lambda^\ell_{s'})^{k-1} T^n}{(k-1)!} - \frac{(\lambda^\ell_{s'})^n T^{k+1}}{k!}\Big) e^{-\lambda^\ell_{s'} T}$$
$$= \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} \Big(\Delta^\ell_{s,s'}(\boldsymbol{\lambda}^\ell, n, k+1) - \Delta^\ell_{s,s'}(\boldsymbol{\lambda}^\ell, n, k)\Big) \cdot \Pr[n^\ell_s = n] \cdot$$
$$\Pr[n^\ell_{s'} = n] \cdot T^2 \le 0,$$

where

$$\Delta^\ell_{s,s'}(\boldsymbol{\lambda}^\ell, n, k) = \mathbb{E}\left[g(\boldsymbol{n}) | n^\ell_s = n+1, n^\ell_{s'} = k\right] - \mathbb{E}\left[g(\boldsymbol{n}) | n^\ell_s = n, n^\ell_{s'} = k\right],$$

holds by the submodularity of g. Thus U is continuous DR-submodular. □

B PROOF OF LEMMA 5.5

By Chernoff bounds described by Theorem A.1.16 in [4], we get:

Lemma B.1. *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 $i \le n^\ell_s$, $\|\boldsymbol{x}^\ell_{s,i}\|_2 \le c_s$, then:*

$$\left|\widehat{\frac{\partial U}{\partial \lambda^p_{s,t}}} - \mathrm{HEAD}^p_{s,t}\right| \le \gamma \max_{s \in \mathcal{S}} \log\Big(1 + \frac{\lambda_{\mathrm{MAX}}(\Sigma_\ell) c_s^2}{\sigma^2_{s,t}}\Big), \tag{42}$$

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

Proof. We define

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

where

$$G_{\mathrm{MAX}} = \max_{\ell \in \mathcal{L}, s \in \mathcal{S}, \boldsymbol{x} \in B(\boldsymbol{0}; c)} \big(G^\ell(\boldsymbol{x}, \boldsymbol{e}_s) - G^\ell(\boldsymbol{x}, \boldsymbol{0})\big)$$
$$= \max_{\ell \in \mathcal{L}, s \in \mathcal{S}, \boldsymbol{x} \in B(\boldsymbol{0}; c)} \log\Big(1 + \frac{1}{\sigma^2_{s,t}} \boldsymbol{x}_s^\top \Sigma_\ell \boldsymbol{x}_s\Big)$$

We have $|X^{j,k}(n)| \le 1$, because:

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

for any $\ell \in \mathcal{L}$, $s \in \mathcal{S}$, $\boldsymbol{x} \in B(\boldsymbol{0}; c)$, $n \ge 0$. By Chernoff bounds described by Theorem A.1.16 in [4], we have

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

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

$$\left|\widehat{\frac{\partial U}{\partial \lambda^p_{s,t}}} - \mathrm{HEAD}^p_{s,t}\right|$$
$$\le \left|\sum_{n=0}^{n'} \sum_{j=1}^{N_1} \sum_{k=1}^{N_2} \frac{G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n^{\ell,j}_s = n+1}) - G^\ell(\boldsymbol{X}^{\ell,j,k}, \boldsymbol{n}^{\ell,j}|_{n^{\ell,j}_s = n}) - \Delta^\ell_s(\boldsymbol{\lambda}^\ell, n)}{N_1 N_2} T\right|$$
$$= \left|\sum_{t=0}^{n'} \sum_{j=1}^{N_1} \sum_{k=1}^{N_2} X^{j,k}(n)\right| \cdot \frac{T}{N_1 N_2} \cdot G_{\mathrm{MAX}} \le \gamma \cdot G_{\mathrm{MAX}}$$
$$= \gamma \max_{\ell \in \mathcal{L}, s \in \mathcal{S}, \boldsymbol{x} \in B(\boldsymbol{0}; c)} \log\Big(1 + \frac{1}{\sigma^2_{s,t}} \boldsymbol{x}_s^\top \Sigma_\ell \boldsymbol{x}_s\Big)$$
$$\le \gamma \max_{\ell \in \mathcal{L}, s \in \mathcal{S}, \boldsymbol{x} \in B(\boldsymbol{0}; c)} \log\Big(1 + \frac{1}{\sigma^2_{s,t}} \lambda_{\mathrm{MAX}}(\Sigma_\ell) \|\boldsymbol{x}_s\|_2^2\Big)$$
$$= \gamma \max_{\ell \in \mathcal{L}, s \in \mathcal{S}} \log\Big(1 + \frac{\lambda_{\mathrm{MAX}}(\Sigma_\ell) c_s^2}{\sigma^2_{s,t}}\Big),$$

with probability greater than $1 - 2 \cdot e^{-\gamma^2 N_1 N_2 / 2T^2 (n'+1)}$, and the last inequality holds because $\lambda_{\mathrm{MIN}}(\Sigma_\ell) |\boldsymbol{x}|^2 \le \boldsymbol{x}^\top \Sigma_\ell \boldsymbol{x} \le \lambda_{\mathrm{MAX}}(\Sigma_\ell) |\boldsymbol{x}|^2$, for any $\boldsymbol{x}$, by [19]. □

Lemma B.2 (Theorem 8.3 in [33]). *If $\boldsymbol{x} \sim N(0, \Sigma)$, for any $\delta \in (0, 1)$:*

$$\Pr\left[\|\boldsymbol{x}\|_2 \le 4\sqrt{\lambda_{\mathrm{MAX}}(\Sigma)} \sqrt{d} + 2\sqrt{\lambda_{\mathrm{MAX}}(\Sigma)} \sqrt{\log \frac{1}{\delta}}\right] \ge 1 - \delta, \tag{43}$$

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

Then, we prove Lem. 5.5 by law of total probability theorem.

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

$$\begin{aligned}
&\Pr(A) = \Pr(A|B\cap C)\Pr(B\cap C) + \Pr(A|\overline{B\cap C})\Pr(\overline{B\cap C})\\
&\geq \Pr(A|B\cap C)\Pr(B\cap C)\\
&\geq (1-2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)})\cdot(1-\delta)^{|\mathcal{S}|n'}\cdot \prod_{s'\in\mathcal{S}\setminus\{s\}}\\
&\quad (1-\Pr[n_{s'}^{\ell}\geq n'+1]),\\
&\geq (1-2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)})\cdot(1-|\mathcal{S}|n'\delta)\cdot(1-(|\mathcal{S}|-1)\delta_{s,t}^{p})\\
&\geq 1-2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)} - |\mathcal{S}|n'\delta - (|\mathcal{S}|-1)\delta_{s,t}^{p},
\end{aligned}$$

where $\delta_{s,t}^{p} = \max_{s'\in\mathcal{S}\setminus\{s\}}\{\Pr[n_{s'}^{\ell}\geq n'+1]\}$. In the first inequality, we drop $\Pr(A|\overline{B\cap C})\Pr(\overline{B\cap C})$, because we do not access to their values. The second inequality is by Lem. B.1, Lem. B.2 and truncating of coordinate s. The third inequality holds by Bernoulli's inequality. □

C PROOF OF THEOREM 5.3

We can bound the quality of our gradient estimator by Lem. 5.6:

Lemma C.1. *At each iteration k, with probability greater than $1-(2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)} + |\mathcal{S}|n'\delta + (|\mathcal{S}|-1)\mathrm{P}_{\mathrm{MAX}})\cdot P_{\mathrm{TOT}}$,*

$$\langle \boldsymbol{v}(k), \nabla U(\boldsymbol{\lambda}(k))\rangle \geq a\cdot \max_{\boldsymbol{v}\in\mathcal{D}}\langle \boldsymbol{v}, \nabla U(\boldsymbol{\lambda}(k))\rangle - b, \tag{44}$$

where

$$a = 1-\mathrm{P}_{\mathrm{MAX}}, \quad \textit{and} \tag{45}$$

$$b = 2\lambda_{\mathrm{MAX}}\cdot\gamma \max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log\Big(1+\frac{\lambda_{\mathrm{MAX}}(\Sigma_\ell)c_s^2}{\sigma_{s,t}^2}\Big), \tag{46}$$

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

Proof. Here, we use the parameter k to represent variables in the k-th iteration: let $\boldsymbol{u}(k)\in\mathcal{D}$ be the vector that maximizes $\langle \boldsymbol{u}(k), \nabla U(\boldsymbol{\lambda}(k))\rangle$. We have

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

where the first and last inequalities hold due to Lem. 5.6, and the second inequality holds because $\boldsymbol{v}(k)$ maximizes $\langle \boldsymbol{v}(k), \widehat{\nabla U(\boldsymbol{\lambda}(k))}\rangle$ by Eq. (19a). The above inequality requires the satisfaction of Eq. (25) for every partial derivative. By union bound, the above inequality satisfies with probability greater than $1-(2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)} + |\mathcal{S}|n'\delta + \sum_{s\in\mathcal{S},t\in\mathcal{T}}\sum_{p\in\mathcal{P}_{s,t}}(|\mathcal{S}|-1)\delta_{s,t}^{p})\cdot P_{\mathrm{TOT}}$, and thus greater than $1-(2\cdot e^{-\gamma^2 N_1 N_2/2T^2(n'+1)} + |\mathcal{S}|n'\delta + (|\mathcal{S}|-1)\mathrm{P}_{\mathrm{MAX}})\cdot P_{\mathrm{TOT}}$ by the definition of $\mathrm{P}_{\mathrm{MAX}}$. □

By Lemma 2 in [27], and Lipschitz constant of ∇U: $L = 2T^2 P_{\mathrm{TOT}}\cdot\max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\mathrm{MAX}}(\Sigma_\ell)c_s^2}{\sigma_{s,t}^2})$, we get:

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

$$\begin{aligned}
U(\boldsymbol{\lambda}(K)) \geq (1-e^{\mathrm{P}_{\mathrm{MAX}}-1})\max_{\boldsymbol{\lambda}\in\mathcal{D}}U(\boldsymbol{\lambda}) - (T^2P_{\mathrm{TOT}}\lambda_{\mathrm{MAX}}^2 +\\
2\lambda_{\mathrm{MAX}})\gamma\max_{\ell\in\mathcal{L},s\in\mathcal{S}}\log(1+\frac{\lambda_{\mathrm{MAX}}(\Sigma_\ell)c_s^2}{\sigma_{s,t}^2}),
\end{aligned}\tag{47}$$

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

Then, we organize constants in above lemma to obtain Thm. 5.3.

Proof. We know that Poisson has a subexponential tail bound:

Lemma C.3 (Thm. 1 in [10]). *Let $n_s^{\ell}\sim\mathrm{Poisson}(\lambda_s^{\ell}T)$, for $\lambda_s^{\ell}, T>0$. Then, for any $z>\lambda_s^{\ell}T$, we have*

$$\Pr[n_s^{\ell}\geq z] \leq e^{-\frac{(z-\lambda_s^{\ell}T)^2}{2\lambda_s^{\ell}T}h(\frac{z-\lambda_s^{\ell}T}{\lambda_s^{\ell}T})}, \tag{48}$$

where $h: [-1,\infty)\rightarrow\mathbb{R}$ is the function defined by $h(u) = 2\frac{(1+u)\ln(1+u)-u}{u^2}$.

This probability Pr is increasing 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:

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

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

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

and assume that

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

Then, we first get: $K = O(\frac{\epsilon_0}{P_{\mathrm{TOT}}(|\mathcal{S}|-1)}\epsilon_1)$, then $\delta = O(\frac{\epsilon_0}{P_{\mathrm{TOT}}K|\mathcal{S}|n'})$, and $N_1N_2 = \Omega(\ln\frac{P_{\mathrm{TOT}}K}{\epsilon_0}\cdot T^2(n'+1)K^2)$. Finally, we have

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

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