2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Differentially Private Stochastic Convex Optimization for Network Routing Applications

Matthew Tsao, Karthik Gopalakrishnan, Kaidi Yang and Marco Pavone

Abstract— **Network routing problems are common across many engineering applications. Computing optimal routing policies requires knowledge about network demand, i.e., the origin and destination (OD) of all requests in the network. However, privacy considerations make it challenging to share individual OD data that would be required for computing optimal policies. Privacy can be particularly challenging in standard network routing problems because sources and sinks can be easily identified from flow conservation constraints, making feasibility and privacy mutually exclusive. In this paper, we present a differentially private algorithm for network routing problems. The main ingredient is a reformulation of network routing which moves all user data-dependent parameters out of the constraint set and into the objective function. We then present an algorithm for solving this formulation based on a differentially private variant of stochastic gradient descent. In this algorithm, differential privacy is achieved by injecting noise, and one may wonder if this noise injection compromises solution quality. We prove that our algorithm is both differentially private and asymptotically optimal as the size of the training set goes to infinity. We corroborate the theoretical results with numerical experiments on a road traffic network which show that our algorithm provides differentially private and near-optimal solutions in practice.**

I. INTRODUCTION

Network routing problems appear in many important topics in engineering, including traffic routing in transportation systems, power routing in electrical grids, and packet routing in distributed computer systems. Network routing problems study settings where resources must be delivered to customers through a network with limited bandwidth. The goal is typically to route resources to their respective customers as efficiently as possible, or equivalently, with as little network congestion as possible. One key challenge in network routing problems is that the requests (i.e., network demand) are often not known in advance. Since information about the demand is often necessary to develop optimal or near-optimal network routing solutions, network routing algorithms often need a way of obtaining or estimating future demand. With the advent of big data and internet-of-things systems, crowd-sourcing has gained popularity as a demand forecasting approach for network routing systems. In crowd-sourcing, customers submit their request history to the network operator. The network operator uses this historical data to train a statistical or machine learning model to predict future demand from historical demand.

While crowd-sourcing provides a bountiful supply of data for training demand forecasting models, it can also introduce potential privacy concerns. Since crowd-sourcing uses individual-level customer data to train demand forecasting models, the model's outputs may reveal sensitive user information, especially if it overfits the training data [1]. Such privacy risks are problematic because they may deter users from sharing their data with network operators, hence reducing the supply of training data for demand forecasting models.

To address such concerns, the demand forecasting pipeline should be augmented with privacy-preserving mechanisms. Differential privacy [2] is a principled and popular method to occlude the influence of a single user's data on the result of a population study while also maintaining the study's accuracy. This is done by carefully injecting noise into the desired computation so that data sets that differ by at most one data point will produce statistically indistinguishable results. Providing differential privacy guarantees for the standard formulation of network routing is difficult because, as we show in Section II-C, the constraints contain user data, meaning that in general feasibility and privacy become mutually exclusive.

A. Statement of Contributions

In this paper, we make the following contributions. First, we define a new concept of privacy for individual users in a network, called request-level differential privacy. Next, we present a reformulation of the standard network flow problem by moving all data-dependent parameters from the constraints to the objective function. To solve the reformulated problem, we adapt prior works to develop a differentially private stochastic gradient descent algorithm. We prove that the difference between a differentially private routing policy and a non-private routing policy asymptotically goes to zero with more data points. Finally, we illustrate the practicality of our algorithm in the context of road traffic routing. In particular, we show that our algorithm achieves a total travel time that is within 2% of the optimal even with less than 50 data points.

B. Related Work

Traffic assignment in transportation systems is one of the most well-known applications of network routing. Privacy works in transportation mainly focus on *location privacy*, where the objective is to prevent untrusted and/or external

M. Tsao is with Lyft Data Science. mattxtsao@gmail.com
K. Gopalakrishnan and M. Pavone are with the Dept. of Aeronautics and Astronautics at Stanford University. {kgopalakrishnan,pavone}@stanford.edu
K. Yang is with the Dept. of Civil Engineering at the National University of Singapore. kaidi.yang@nus.edu.sg
K. Yang would like to acknowledge the support from Singapore Ministry of Education under NUS Start Up Grant (A-8000404-01-00).

entities from learning geographic locations or location sequences of an individual [3]. Privacy-preserving approaches have been proposed for various location-based applications [4], e.g., trajectory publishing, mobile crowdsensing, traffic control, etc. These techniques are based on spatial cloaking [5] and differential privacy [6]. While not the setting of interest for this paper, there are many works that use Secure Multi-Party Computation [7] to achieve privacy in *distributed* mobility systems.

Spatial cloaking approaches aggregate users' exact locations into coarse information. These approaches are often based on k-anonymity [8], where a mobility dataset is divided into equivalence classes based on data attributes (e.g., geological regions, time, etc.) so that each class contains at least k records [9], [10]. These k-anonymity-based approaches can guarantee that every record in the dataset is indistinguishable from at least $k-1$ other records. However, k-anonymity is generally considered a weak privacy guarantee, especially for small k. Furthermore, coarse data aggregation is required to address outliers or sparse data, and in these cases, spatial cloaking-based approaches provide low data accuracy.

Differential privacy provides a principled privacy guarantee by producing randomized responses to queries, whereby two datasets that differ in only one entry produce statistically indistinguishable responses [2]. In other words, differential privacy ensures that an adversary with arbitrary background information (e.g., query responses, other entries) cannot infer individual entries with high confidence. Within transportation research, [11], [12], [13], [14] share noisy versions of location data for mobile crowd-sensing applications. [15], [16], [17], [18] use differential privacy to publish noisy versions of trajectory data. [19] and [20] apply differential privacy to gradient descent for federated learning in mobility systems.

Many of the works mentioned in the previous paragraph establish differential privacy of their proposed algorithms by using composition properties of differential privacy. Composition theorems for differential privacy describe how well privacy is preserved when conducting several computations one after another. In [19] and [20], composition theorems are applied as black boxes without considering the mathematical properties of the gradient descent algorithm. As a result, the privacy guarantees are overly conservative, meaning that large amounts of noise are added to the algorithm, leading to suboptimal behavior both in theory and in practice. Similarly, [15], [16], [17], [18] use composition rules as a black box, and while privacy is achieved, there are no accuracy guarantees for the algorithms presented in those works.

While blackbox applications of differential privacy can lead to impractical levels of noise injection, specialized applications of differential privacy were discovered that provide privacy with much less noise. [21] show how a simple adjustment to stochastic gradient descent can give rise to an algorithm which is both differentially private, and under reasonable regularity assumptions, is also asymptotically optimal. [22] and [23] refined this idea to develop stochastic gradient descent algorithms that are differentially private, computationally efficient, and have optimal convergence rates. These techniques cannot directly be used to solve the standard formulation of network routing because they study unconstrained optimization problems or optimization problems with public constraint sets (i.e., constraints that do not include any private data).

II. Model

In this section, we define notations, network models, and assumptions that will allow us to formulate network routing as a data-driven convex optimization problem.

A. Network, Demand, and Network Flow

In this section, we will introduce a) a graph model of a network, b) a representation of network demand, c) the standard formulation for network routing and d) privacy requirements.

Definition 1 (Network Representation): We use a directed graph $G = (V, E)$ to represent the network, where V and E represent the set of vertices and edges in the network, respectively. We will use $n := |V|$ and $m := |E|$ to denote the number of vertices and edges in the graph, respectively. For vertex pairs $(o, d) \in V \times V$ we will use $\mathcal{P}_G(o, d)$ to denote the set of simple paths (i.e., paths with no cycles) from o to d in G.

Definition 2 (Operation Period): We use $\mathcal{T} := [t_{\text{start}}, t_{\text{end}}]$ to denote the operation period within which a network operator wants to optimize its routing decisions. We will also use T to denote the number of minutes in the operation period. For example, $t_{\text{start}} = 8:00\text{am}$, $t_{\text{end}} = 9:00\text{am}$ could represent a morning commute period where $T = 60$.

Definition 3 (Demand Representation): We study a stationary demand model where demand within the operation period $\mathcal{T}$ is specified by a matrix $\Lambda \in \mathbb{R}_+^{n \times n}$. For each ordered pair of vertices $(o, d) \in V \times V$, $\Lambda(o, d)$ is the number of requests arriving per minute (i.e., the arrival rate) during $\mathcal{T}$ that need routing from vertex o to vertex d.

Remark 1 (Estimating Λ from historical data): The arrival rates from historical demand are computed empirically, i.e., if Λ_t represents the demand for day t, then $\Lambda_t(o, d)$ is calculated by counting the number of (o, d) requests appearing on day t, and then dividing it by T to obtain requests per minute.

Definition 4 (Link Latency Functions): To model congestion effects, each link $e \in E$ has a latency function $f_e : \mathbb{R}_+ \to \mathbb{R}_+$ which specifies the average travel time through the link as a function of the total flow on the link.

In this paper we study a setting where a network operator wants to route demand while minimizing the total travel time for the requests. With these definitions, the standard formulation of minimum travel time network routing is described in Definition 5.

Definition 5 (Standard Formulation of Network Flow): For a network $G = (V, E)$ with link latency functions $\{f_e\}_{e \in E}$ and demand Λ, the standard network flow problem is the following minimization program with decision variable $x = \left\{x^{(o,d)}\right\}_{(o,d) \in V \times V}$:

$$\underset{x}{\text{minimize}} \sum_{e\in E} y_e f_e(y_e) \tag{1a}$$

$$\text{subject to } y_e = \sum_{(o,d)\in V\times V} x_e^{(o,d)} \text{ for all } e \in E, \tag{1b}$$

$$\sum_{v:(u,v)\in E} x_{(u,v)}^{(o,d)} - \sum_{w:(w,u)\in E} x_{(w,u)}^{(o,d)} = \Lambda(o,d)(\mathbb{1}_{[u=o]} - \mathbb{1}_{[u=d]}) \tag{1c}$$

$$\text{for all } u \in V, (o,d) \in V \times V.$$

In (1), the decision variable x is a collection of flows $\left\{x^{(o,d)}\right\}_{(o,d)\in V\times V}$, one for each non-zero entry of Λ. $x_e^{(o,d)}$ is th amount of flow that $x^{(o,d)}$ sends on an edge e. (1c) are flow conservation constraints to ensure that $x^{(o,d)}$ sends $\Lambda(o,d)$ units of flow from o to d. Constraints (1b) ensure that $\{y_e\}_{e\in E}$ represents the total amount of flow on each edge. Finally, the objective function (1a) is to minimize the total travel time as a function of total network flow.

In Section II-B, we will describe the rigorous privacy requirements that we will mandate while designing algorithms for network flow. We then describe in Section II-C why privacy and feasibility are mutually exclusive in the standard network flow formulation.

B. Privacy Requirements

We will use differential privacy to reason about the privacy-preserving properties of our algorithms. At a high level, changing one data point of the input to a differentially private algorithm should lead to a statistically indistinguishable change in the output [2]. To make this concept concrete we will need to define data sets and adjacency.

Definition 6 (Data sets): Given a space of data points $\mathcal{Z}$, a data set L is any finite set of elements from $\mathcal{Z}$. In practice, each element of a data set is data collected from a single user, or data collected during a specific time. We use $\mathcal{L}$ to denote the space of data sets.

Definition 7 (Data Set Adjacency): Given a space of data sets $\mathcal{L}$, an adjacency relation Adj is a mapping $\text{Adj} : \mathcal{L}\times\mathcal{L} \to \{0,1\}$. Two data sets L_1, L_2 are said to be adjacent if and only if $\text{Adj}(L_1, L_2) = 1$.

For our application we will choose Adj to be Request Level Adjacency.

Definition 8 (Request Level Adjacency (RLA)): The function $\text{RLA} : \mathcal{L} \times \mathcal{L} \to \{0,1\}$ maps pairs of data sets to booleans. For two historical datasets of network demand $L := (\Lambda_1, ..., \Lambda_N)$ and $L' := (\Lambda'_1, ..., \Lambda'_N)$, we say L and L' are request-level-adjacent (RLA) if either L contains all of the requests in L', and contains one extra request that is not present in L' or vice versa. Mathematically, L, L' are request-level-adjacent, i.e., $\text{RLA}(L, L') = 1$, if and only if they satisfy all of the following relations:

- There exists t so that $\Lambda_k = \Lambda'_k$ for all $k \neq t$.
- There exists two vertices o and d so that $\Lambda_t(o', d') = \Lambda'_t(o', d')$ for all $(o', d') \neq (o, d)$.
- $|\Lambda_t(o,d) - \Lambda'_t(o,d)| \leq \frac{1}{T}$.

Indeed, these relations dictate that one of the datasets contains an extra request from o to d which happened on the tth day. A difference of 1 request within a T minute operation period leads to a change of $\frac{1}{T}$ in the arrival rate. Aside from this difference, the datasets are otherwise identical.

With these definitions in place, we are now ready to define differential privacy.

Definition 9 (Differential Privacy): For a given adjacency relation Adj, a function $M : \mathcal{L} \to \mathcal{X}$ is (ϵ, δ)-differentially private if for any $L_1, L_2 \in \mathcal{L}$ with $\text{Adj}(L_1, L_2) = 1$, the following inequality holds for every event $E \subset \mathcal{X}$:

$$\mathbb{P}\left[M(L_1) \in E\right] \leq e^{\epsilon}\mathbb{P}\left[M(L_2) \in E\right] + \delta.$$

C. Differential Privacy Challenges in Standard Network Flow

In the introduction, we mentioned that privacy and feasibility can be mutually exclusive in the standard formulation of network flow described in (1). In Theorem 1 we formally show that if Λ is constructed from a data set of trips as described in Remark 1, then trips to or from uncommon locations can be easily detected from any feasible solution to (1). As a result, announcing or releasing a feasible solution to (1) is not, in general, differentially private. Formally, we will prove the following theorem in this section.

Remark 2: The vulnerability of trips to and from uncommon locations is not a purely theoretical concern. A study on the New York City Taxi and Limousine data set was able to identify trips from residential areas to night clubs [24].

Theorem 1 (Privacy-Feasibility Contention): Let M be a function that takes as input a matrix Λ with non-negative entries and returns a feasible solution to the optimization problem (1) where Λ is used as a demand matrix. Then M cannot be (ϵ, δ)-differentially private for any $\delta < 1$.

We note that (ϵ, δ)-differential privacy only provides a meaningful privacy guarantee when $\epsilon < 1$ and $\delta < 1$ [6].

Proof: [Proof of Theorem 1] Let $G = (V, E)$ be a network, and Λ be constructed from a historical data set of requests in G. Suppose there exist uncommon locations o and d for which Λ contains no trips to or from either o or d. Mathematically, this means that

$$\Lambda(o,u) = \Lambda(u,o) = \Lambda(d,u) = \Lambda(u,d) = 0 \text{ for all } u \in V.$$

Such situations are not uncommon in transportation networks, if, for example, o and d are the homes of two different people who do not drive (perhaps they walk or bike to and from work).

If we now add a trip from o to d to the data set, and let Λ' be the resulting demand matrix, then we have

i $\Lambda(o', d') = \Lambda'(o', d')$ for all $(o', d') \neq (o, d)$, and
ii $\Lambda(o,d) = 0$, $\Lambda'(o,d) = \frac{1}{T}$.

Let $\text{Prob}_1, \text{Prob}_2$ be the optimization problem (1) using demand Λ, Λ' respectively. Because Λ, Λ' are request-level-adjacent, any differentially private algorithm must behave similarly when acting on Prob_1 and Prob_2. However, this is impossible because the feasible sets of $\text{Prob}_1, \text{Prob}_2$ are disjoint. If we look at constraint (1c) with $u = d$ and (o, d) then any feasible solution to Prob_1 satisfies

$$\sum_{u:(u,d)\in E} x_{(u,d)}^{(o,d)} - \sum_{w:(d,w)\in E} x_{(u,d)}^{(o,d)} = \Lambda(o,d)\mathbb{1}_{[d=d]} = 0.$$

However, any feasible solution to Prob_2 satisfies

$$\sum_{u:(u,d)\in E} x_{(u,d)}^{(o,d)} - \sum_{w:(d,w)\in E} x_{(u,d)}^{(o,d)} = \Lambda(o,d)\mathbb{1}_{[d=d]} = \frac{1}{T}.$$

In other words, checking the net flow leaving node d will detect the presence of any trips going to or from d. We will now show that any algorithm which returns a feasible solution to (1) cannot be differentially private. To this end, define the event E to be the event that flow is conserved at node d. Then for any algorithm M that takes as input a demand matrix and returns a feasible solution to (1) with the specified demand matrix, we have $\mathbb{P}[M(\Lambda) \in E] = 1, \mathbb{P}[M(\Lambda') \in E] = 0$. Recalling Definition 9, M is (ϵ, δ)-differentially private only if $\mathbb{P}[M(\Lambda) \in E] \leq e^{\epsilon}\mathbb{P}[M(\Lambda') \in E] + \delta$. This equation can only be satisfied if $\delta \geq 1$. ■

III. Routing Policy Formulation of Network Flow

To sidestep the impossibility result described in Theorem 1, we present an alternative formulation for the network flow problem in this section. The alternative formulation avoids the issues mentioned in Theorem 1 by moving all parameters related to user data from the constraints to the objective function, as described in (4). We note that (4) can only be solved if the demand Λ is known, which may not always be the case. For this reason, we present two variations of (4): (5) is the stochastic version of (4) where Λ is drawn from a distribution $\mathcal{Q}$, and (6) is the data driven approximation to (5) that one would solve if $\mathcal{Q}$ is unknown.

Before formally defining the model, we provide a high level description of how this formulation works. In this formulation, a feasible solution $x = \left\{x^{(o,d)}\right\}_{(o,d)\in V\times V}$ specifies, for each $(o,d) \in V \times V$, a flow $x^{(o,d)}$ that routes 1 unit of flow from o to d. We note that a flow is specified for (o,d) even if there is no demand for this origin and destination in Λ, i.e., $\Lambda(o,d) = 0$. We refer to x as a *routing policy* due to its connection to randomized routing. Given a feasible solution x, the objective function first calculates the total traffic on each edge by taking a linear combination of $\left\{x^{(o,d)}\right\}_{(o,d)\in V\times V}$ flows, where the coefficients of the linear combination are determined by the demand Λ, with high demand (o,d) pairs having larger coefficients. The total travel time can be computed from the total traffic in the same way as (1a). These ideas are formalized by the following definitions.

Definition 10 (Unit (o,d) flow): For a given origin-destination pair (o,d), we say that a flow $x^{(o,d)} \in \mathbb{R}_+^m$ is a unit (o,d) flow if and only if it routes exactly 1 unit of flow from o to d. Formally, this condition can be represented for all $u \in V$ as

$$\sum_{v\in V:(v,u)\in E} x_{(v,u)}^{(o,d)} - \sum_{v\in V:(u,v)\in E} x_{(u,v)}^{(o,d)} = \begin{cases} -1 & \text{if } u = o \\ 1 & \text{if } u = d \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Indeed, (2) requires that the net flow entering o is -1, the net flow entering d is 1, and that flow is conserved at all other vertices in the graph.

Definition 11 (Unit Network Flow): A unit network flow is a collection of flows $x = \left\{x^{(o,d)}\right\}_{(o,d)\in V\times V}$ so that $x^{(o,d)}$ is a unit (o,d) flow for each $(o,d) \in V \times V$. We use $\mathcal{X}$ to denote the set of all unit network flows.

Remark 3: We can represent x as a concatenation of the vectors $\left\{x^{(o,d)}\right\}_{(o,d)\in V\times V}$. Since each unit (o,d) flow is a m dimensional vector, we have $x \in \mathbb{R}^{n^2 m}$.

We will refer to unit network flows as routing policies, due to their connection with randomized routing, described in Remark 5 of the extended version [25].

A. Minimum Total Travel Time Network Flow

In the minimum travel time network flow problem, the network operator wants to find a stationary routing policy for each (o,d) pair that will lead to small total travel times for the requests. Due to the equivalence between stationary (o,d) routing policies and unit (o,d) flows, the network operator can instead search over the space of unit (o,d) flows.

The total travel time of a flow y through G is given by $\sum_{e\in E} y_e f_e(y_e)$. The total amount of flow on an edge $e \in E$ when serving Λ according to x is given by scaling each flow by the magnitude of its demand, i.e., $\sum_{(o,d)\in V\times V} \Lambda(o,d)x_e^{(o,d)}$. We can thus define $F(x,\Lambda)$, the total travel time experienced by requests Λ when being routed by x, as follows:

$$F(x,\Lambda) := \sum_{e\in E} \left(\sum_{\substack{(o,d)\in \\ V\times V}} \Lambda(o,d)x_e^{(o,d)} \right) f_e \left(\sum_{\substack{(o,d)\in \\ V\times V}} \Lambda(o,d)x_e^{(o,d)} \right). \tag{3}$$

With these definitions in place, the unit network flow that minimizes the total travel time when serving the demand Λ can be found by solving the following optimization problem

$$\underset{x\in\mathcal{X}}{\text{minimize}}\ F(x,\Lambda) \tag{4}$$

B. Network Flow with Stochastic Demand

In practice, demand may vary from day to day, and such variations can be modeled by Λ being a random variable with distribution $\mathcal{Q}$. If $\mathcal{Q}$ is known by the network operator, then rather than solving (4), the operator is interested in minimizing expected total travel time through the following stochastic optimization problem:

$$\underset{x\in\mathcal{X}}{\text{minimize}}\ \mathbb{E}_{\Lambda\sim\mathcal{Q}}\left[F(x,\Lambda)\right] \tag{5}$$

We note that (5) is a generalization of (1) to the case when Λ is random.

In the more realistic case that $\mathcal{Q}$ is not known, the problem (5) can be approximated from historical data. We study a situation where the network operator has demand data $\Lambda_1, \Lambda_2, ..., \Lambda_N \overset{\text{i.i.d}}{\sim} \mathcal{Q}$ collected from operations of previous days. Using this data it can solve the following empirical approximation to (5):

$$\underset{x\in\mathcal{X}}{\text{minimize}}\ \frac{1}{N}\sum_{k=1}^{N} F(x,\Lambda_k) \tag{6}$$

The optimization problem in (6) uses historical data to estimate (5). In line with Assumption 1 we will assume that $\Lambda_t(o,d) \leq \lambda_{\max}$ for all values of t, o and d. In Section IV we show how (6) can be solved in a request-level differentially private way.

C. Assumptions on Travel Time Functions

In this section, we will introduce some assumptions that will help us establish our technical results. We will make the following assumptions on the network demand:

Assumption 1 (Bounded Demand): We assume there exists a non-negative constant $\lambda_{\max}$ so that $\Lambda(o,d) \leq \lambda_{\max}$ for every $(o,d) \in V \times V$. In practice, this constant can be estimated from historical data.

The following are assumptions we make on the objective function F (see (3)). These assumptions are related to properties of the link latency functions $\{f_e\}_{e\in E}$. We present a typical model for link latency functions in Section III-D that satisfies all of the following assumptions.

Assumption 2 (Bounded Variance Gradients): We assume there exists a non-negative constant K so that for every x, the variance of $\nabla F(x,\Lambda)$ is upper bounded by K^2, i.e., $\mathbb{E}_{\Lambda\sim\mathcal{Q}}\left[||\nabla F(x,\Lambda)||_2^2\right] \leq K^2$.

Assumption 3 (Twice Differentiability): We assume that $F(\cdot,\Lambda)$ is twice-differentiable for every Λ so that the hessian $H(x,\Lambda) := \frac{\partial^2}{\partial x^2}F(x,\Lambda)$ is defined on the entire domain of x.

Assumption 4 (Strong Convexity): We assume there exists $\alpha > 0$ for which $F(\cdot,\Lambda)$ is α-strongly convex for every Λ, i.e., for every Λ, and any unit network flows x, x' we have

$$F(x',\Lambda) \geq F(x,\Lambda) + \nabla_x F(x,\Lambda)^\top (x'-x) + \frac{\alpha}{2}||x'-x||_2^2$$

Assumption 5 (Smoothness): We assume there exists $\beta > \alpha$ for which $F(\cdot,\Lambda)$ is β-smooth for every Λ, i.e., for every Λ, and any unit network flows x, x' we have

$$F(x',\Lambda) \leq F(x,\Lambda) + \nabla_x F(x,\Lambda)^\top (x'-x) + \frac{\beta}{2}||x'-x||_2^2.$$

Assumption 6 (Bounded second order partial derivative): We assume that there exists $C > 0$ so that $\left|\left|\mathcal{D}_{\Lambda(o,d)}\left[\nabla_x F\right](x,\Lambda)\right|\right|_{op} \leq C$ for all x, Λ and $(o,d) \in V \times V$.

D. Transportation Model Satisfying Assumptions from Section III-C

In this section, we present a transportation network model that satisfies Assumptions 3,4,5 and 6. We study a network where the link latency functions are all affine where for each $e \in E$, there are non-negative constants q_e, c_e so that $f_e(y) = q_e y + c_e$. Let $Q \in \mathbb{R}^{m\times m}$ be defined so that $Q_{ee} = q_e$, and let $c \in \mathbb{R}^m$ be the concatenation of all of the zero order coefficients in the link latency functions.

As mentioned in Remark 3, we will represent x as a concatenation of $\left\{x^{(o,d)}\right\}_{(o,d)\in V\times V}$. As such, $x \in \mathbb{R}^{n^2 m}$. Let $\{(o_i,d_i)\}_{i=1}^{n^2}$ be the order in which the unit (o,d) flows are concatenated to produce x so that $x = \left[x^{(o_1,d_1)\top}, x^{(o_2,d_2)\top}..., x^{(o_{n^2},d_{n^2})}\right]^\top$.

The total flow on the links in the network when serving demand Λ according to x can then be written as:

$$y := \sum_{(o,d)\in V\times V} \Lambda(o,d)x^{(o,d)} = B_\Lambda x$$

where $B_\Lambda = \text{vec}(\Lambda) \otimes I_m$. Here $\text{vec}(\Lambda)$ is a n^2 dimensional row vector whose ith entry is $\Lambda(o_i,d_i)$, and $\otimes$ represents the Kronecker product. Then when Λ is being routed according to x, the travel times on the links can be computed as $Qy + c = QB_\Lambda x + c$, which means that the total travel time can be written as

$$\widehat{F}(x,\Lambda) := \sum_{e\in E} y_e f_e(y_e) = y^\top (Qy + c) \tag{7}$$

$$= x^\top B_\Lambda^\top Q B_\Lambda x + c^\top B_\Lambda x. \tag{8}$$

Adding ℓ_2 regularization gives

$$F(x,\Lambda) = \widehat{F}(x,\Lambda) + \frac{\alpha}{2}||x||_2^2$$
$$= x^\top \left(B_\Lambda^\top Q B_\Lambda + \frac{\alpha}{2}I\right)x + c^\top B_\Lambda x.$$

Defining $\beta := n^2 \left(\max_e q_e\right)\lambda_{\max}^2 + \alpha/2$ and $C = 2\lambda_{\max}||Q||_{op}\sqrt{m}n(n+1) + ||c||_2$, it is straightforward to show that Assumptions 3,6,5,4 are satisfied. See Section 3.4 of the extended version [25] for further details.

IV. Differentially Private Network Routing Optimization

Given the setup from Section III, our objective is to design a request-level differentially private algorithm that returns a near optimal solution to (5). Since the true distribution $\mathcal{Q}$ of demand is unknown, we will design an algorithm for (6) and show that under the assumptions described in Section III-B, the algorithm's solution is also near optimal for (5).

We present a Private Projected Stochastic Gradient Descent algorithm, which is described in Algorithm 1. The algorithm conducts a single pass over the historical data, using each data point to perform a noisy gradient step (see line 6). The key difference between Algorithm 1 and standard stochastic gradient descent is in line 10, where instead of returning the final gradient descent iterate, Algorithm 1 returns a noisy version of the last iterate. Algorithm 1 has the following privacy and performance guarantees.

Theorem 2 (Privacy Guarantee for Algorithm 1): Algorithm 1 is (ϵ,δ)-differentially private under request level adjacency defined in Definition 8.

Theorem 3 (Performance Guarantee for Algorithm 1): If $\Lambda_1, ..., \Lambda_N \overset{\text{i.i.d}}{\sim} \mathcal{Q}$, and x^* is a solution to (5), then the output x_{alg} of Algorithm 1 satisfies:

$$\mathbb{E}\left[\left|\left|x_{\text{alg}} - x^*\right|\right|_2\right]$$
$$\leq \frac{1}{\alpha\sqrt{N}} KC \exp\left(\frac{\beta^2\pi^2}{12\alpha^2}\right) + \frac{n\sqrt{m}}{N}\left(\frac{\beta \exp\left(\frac{\beta^2\pi^2}{12\alpha^2}\right)}{\alpha \min(1, 2\alpha)} + \frac{C}{\epsilon\alpha T}\sqrt{2\ln\left(\frac{1.25}{\delta}\right)}\right)$$
$$\leq O\left(\frac{1}{\sqrt{N}}\right) + O\left(\frac{1}{\epsilon N}\sqrt{\ln\frac{1}{\delta}}\right)$$

In particular, x_{alg} is (ϵ,δ)-differentially private and converges to x^* as $N \to \infty$, meaning that privacy and asymptotic optimality are simultaneously achieved.

See Sections B and C of the extended version [25] for proofs of Theorem 2 and Theorem 3 respectively.

Algorithm 1: Private Projected Stochastic Gradient Descent

Input: Historical demand $\Lambda_1, ..., \Lambda_N$, privacy level (ϵ, δ);
Output: Unit network flow $x \in \mathcal{X}$;
Initialize $x_0 \in \mathcal{X}$ arbitrarily ;
for $1 \leq k \leq N$ **do**
  $\eta_{k-1} \leftarrow \min\left(\frac{1}{\alpha k}, \frac{\min(1,2\alpha)}{\beta}\right)$;
  $x_k \leftarrow \Pi_{\mathcal{X}}\left(x_{k-1} + \eta_{k-1}\nabla_x F(x_{k-1}, \Lambda_k)\right)$;
$s \leftarrow \frac{C}{T}\min\left(\frac{\min(1,2\alpha)}{\beta}, \frac{1}{\alpha N}\right)$;
$\sigma^2 \leftarrow 2\frac{s^2}{\epsilon^2}\ln\left(\frac{1.25}{\delta}\right)$;
$Z \sim \mathcal{N}\left(0, \sigma^2 I\right)$;
$x_{\text{alg}} \leftarrow \Pi_{\mathcal{X}}\left(x_N + Z\right)$;
Return x_{alg};

A. Discussion

Carefully adding noise to specific parts of existing algorithms is a principled approach for developing differentially private algorithms [2], [22], [23]. The main challenge in such an approach is determining a) where and b) how much noise to add. Suppose the goal is to (approximately) compute a query function $f(L)$ on a data set L in a differentially private way. The latter question can be addressed by measuring the sensitivity of the desired query function.

Definition 12 (ℓ_2 sensitivity): Consider a function $f : \mathcal{L} \rightarrow \mathbb{R}^d$ which maps data sets to real vectors. For a given adjacency relation Adj, the ℓ_2 sensitivity of f, denoted $s_{\text{Adj}}(f)$, is the largest achievable difference in function value between adjacent data sets. Namely,

$$s_{\text{Adj}}(f) := \max_{\substack{L_1, L_2 \in \mathcal{L} \\ \text{Adj}(L_1, L_2)=1}} ||f(L_1) - f(L_2)||_2 .$$

Once the sensitivity of the query function is known, the required noise distribution can be determined using the Gaussian mechanism as described in Theorem 4.

Theorem 4 (From [26]): Suppose $f : \mathcal{D} \rightarrow \mathbb{R}^p$ maps datasets to real vectors. If s_{Adj} is the ℓ_2 sensitivity of f, then $\widehat{f}(D) := f(D) + Z$ where $Z \sim \mathcal{N}\left(0, 2\frac{s_{\text{Adj}}^2}{\epsilon^2}\ln\left(\frac{1.25}{\delta}\right) I_p\right)$ is (ϵ, δ)-differentially private with respect to the adjacency relation Adj.

Algorithm 1 is an application of the Gaussian mechanism. Indeed, the main ingredient in the proof of Theorem 2 is the following sensitivity bound:

Lemma 1: Let $x_N(L)$ be the Nth gradient descent iterate when applied to data set L (as defined in Algorithm 1). Then $s_{\text{Adj}}(x_N) \leq \frac{C}{T}\left(\frac{\min(1,2\alpha)}{\beta}, \frac{1}{\alpha N}\right)$ where Adj is RLA.

The privacy of Algorithm 1 is achieved by calibrating the variance of the added Gaussian noise to $s_{\text{Adj}}(x_N)$. The proof of Lemma 1 can be found in Section B of the extended version [25].

Moreover, Theorem 3 shows that the suboptimality of Algorithm 1 is $O\left(\frac{1}{\sqrt{N}}\right)$. The asymptotic optimality of Algorithm 1 comes from the fact that the ℓ_2 sensitivity of the final gradient descent iterate is actually converging to zero as N approaches infinity. This fact enables us to add less noise as $N \rightarrow \infty$. Indeed, the Gaussian noise added to the final gradient descent iterate in Algorithm 1 has a standard deviation which is $O\left(\frac{1}{\epsilon N}\right)$.

V. Experiments

We present empirical studies on privacy-performance trade-offs by comparing our algorithm's performance to that of a non-private network routing approach. To this end, we simulate a transportation network to evaluate the performance of our algorithm and the non-private optimal solution to the network flow problem (6).

A. Setup

We use data for the Sioux Falls network, which is available in the Transportation Network Test Problem (TNTP) dataset [27]. This network has 24 nodes, 76 edges, and 528 OD pairs (see Figure 1 for an overview of the network topology). The distribution of mean hourly demand across different OD pairs is shown in Figure 1. The average OD has 682 requests per hour and the travel time on edges range from 2 to 10 minutes. Trip data is generated each hour, i.e., $T = 60$, from a Poisson distribution with a mean value that is given by the data. Our goal is to learn a routing policy for these trips that minimizes the total travel time. To model congestion, we use the link latency model described in Section III-D. For each $e \in E$, we estimate the free flow latency as c_e directly from the data set, and the slope of the latency function q_e is chosen such that the travel time on the link is doubled when the link flow equals the link capacity.

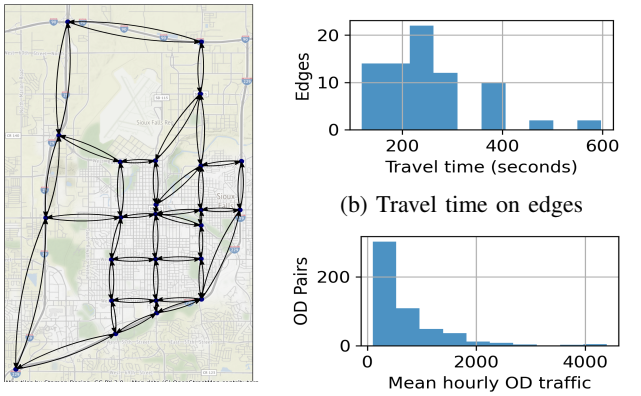
Throughout Sections V-B and V-C we compare the performance of two different algorithms, described below.

Baseline: This algorithm computes the minimum travel time solution to (6) for the Sioux Fall model described in Section III-D. Recall that in the Section III-D model, (8) describes the travel time incurred when serving demand Λ with a routing policy x. Given a data set $\Lambda_1, ..., \Lambda_N$, it computes the solution to the following minimization problem: $\min_{x \in \mathcal{X}} \frac{1}{N}\sum_{i=1}^{N} x^\top B_{\Lambda_i}^\top Q B_{\Lambda_i} x + c^\top B_{\Lambda_i} x$.

Algorithm 1: The travel time function described in (8) is not strongly convex because $B_\Lambda^\top Q B_\Lambda$ is rank deficient. In order to satisfy Assumption 4, we introduce an ℓ_2 regularization. Namely, given a data set $\Lambda_1, ..., \Lambda_N$, we apply Algorithm 1 to the following minimization problem: $\min_{x \in \mathcal{X}} \alpha\,||x||_2^2 + \frac{1}{N}\sum_{i=1}^{N} x^\top B_{\Lambda_i}^\top Q B_{\Lambda_i} x + c^\top B_{\Lambda_i} x$.

The parameters for Algorithm 1 are set as follows. The smoothness parameter β is set to be the largest eigenvalue of $B_\Lambda^\top Q B_\Lambda$, which is equal to 2.08×10^7. We set $C = \beta$, and the regularizer parameter $\alpha = 10^4$. It is easy to check that these values satisfy the assumptions described in Section III-D. Note that several different values of α could have been used to convexify the latency function. However, our choice is governed by two factors. A small value of α will ensure that the regularized objective is a good estimate of the true objective, which is desirable. However, smaller values of

α will lead to a larger condition number (which is β/α), resulting in slower convergence and necessitating more data for achieving a similar performance. Thus, the particular value of $\alpha = 10^4$ balances both these factors for our problem instance. In section V-C, we present a sensitivity analysis with different values of α.



(a) Network topology

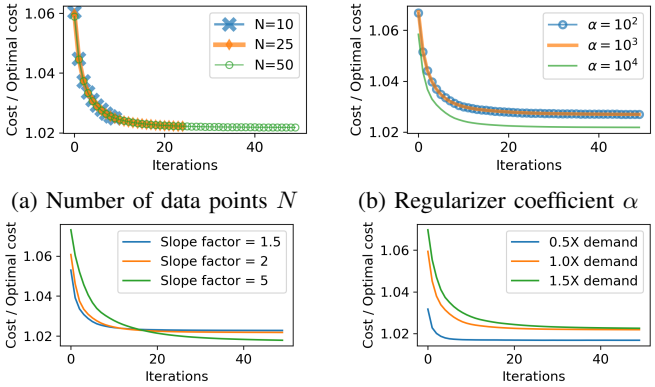
(b) Travel time on edges

(c) Mean demand between OD pairs

Fig. 1: Description of the Sioux Falls dataset

δ \ ϵ	0.01	0.1	0.5
0.1	7.83×10^{-2}	9.06×10^{-3}	2.44×10^{-3}
0.5	3.97×10^{-3}	5.96×10^{-3}	2.05×10^{-3}

TABLE I: Percentage increase in total travel time due to incorporating privacy.



(a) Number of data points N

(b) Regularizer coefficient α

(c) Slope of latency function

(d) Traffic demand

Fig. 2: Convergence dynamics of Algorithm 1 for different parameters

B. Performance of Algorithm 1

We study the convergence of the routing policy with each step of the gradient descent performed by Algorithm 1. Since the impact of a routing policy is directly reflected in the total travel time, we plot the travel cost induced by the learned routing policy as a function of the iterates. In our experiments, we use the average demand to both evaluate a routing policy and solve the benchmark optimization (instead of using the sample average approximation) for improved computational performance. Section D of the extended version [25] shows that this approximation induces at most $10^{-4}\%$ error while giving a 30X computational speedup.

For our first set of experiments, we compare the objective values obtained by Algorithm 1 and the Baseline as a function of sample size N. In Figure 2a, we plot the ratio of Algorithm 1's cost to Baseline's cost over the course of iterations for different values of N. For this set of experiments, we set the privacy parameters to $\epsilon = 0.1$ and $\delta = 0.1$. Note that Baseline's cost is fixed for a given N and is computed offline to serve as a benchmark. For a given N, we only have N iterations since each data point is only used once in Algorithm 1 to maintain privacy. For all three experiments ($N = 10$, $N = 25$, $N = 50$), the cost decreases monotonically with additional iterations. We note that Algorithm 1 finds solutions that are 2% suboptimal, and we suspect that this gap is due to differences in objective functions arising from Algorithm 1's regularization. Thus our results show good performance for small sample sizes, complementing the theoretical result from Theorem 3.

In the next set of experiments, we study the effect of different privacy parameters on total travel time. To this end, we compare the costs of the pre and post-noise solutions x_N, x_{alg} from Algorithm 1. We conduct this comparison for $\epsilon \in \{0.01, 0.1, 0.5\}$ and $\delta \in \{0.1, 0.5\}$. Table I presents the percentage increase in total travel time due to the addition of privacy noise.

The results indicate that the price of privacy, i.e., the increase in total travel time due to the introduction of differential privacy noise is less than $7.8 \times 10^{-2}\%$ in the worst case. In fact, for more commonly used privacy parameters of $\epsilon = 0.1$ and $\delta = 0.1$, the cost of privacy is even smaller. One reason for this low cost of privacy is the high demand in the traffic network, which leads to a high signal to noise ratio for the Gaussian mechanism. Indeed, Figure 1c shows that every OD pair typically has a few hundred trips.

C. Sensitivity

We now discuss the sensitivity of our algorithm to input parameters. We fix the number of data points to $N = 50$ since prior results suggest that most benefits are obtained with 50 data points. First, we study the effect of the regularizer term by setting $\alpha \in \{10^2, 10^3, 10^4\}$ in Figure 2b. We see that for larger values of α, the costs decrease faster. This makes sense because larger α results in a lower condition number $\frac{\beta}{\alpha}$, which leads to faster convergence. We also note that the cost ratio does not go to 1 because Algorithm 1 is minimizing a regularized objective, while the Baseline has no regularization.

In Figure 2c, we vary the slope for the latency function q_e. We consider three cases where the slope is chosen such that the link latency at capacity is 1.5, 2, and 3 times the free flow latency. Note that changing the latency sensitivity factor (ratio of latency at capacity to latency at free flow) changes the matrix Q. Thus, for each of these experiments, we recompute the value of β and C and set it to be equal

to the largest eigenvalue of the appropriate $B_\Lambda^\top Q B_\Lambda$. We set $\alpha = 10^4$ and recompute the optimal costs for all three cases. We observe that when the latency function is steeper, i.e., the sensitivity factor is higher, the algorithm takes more iterations to reduce the costs, but eventually ends up with the lowest costs. This is because a larger β leads to a larger condition number $\frac{\beta}{\alpha}$, which makes convergence slow. However, a larger β means that the ℓ_2 regularizer is a smaller proportion of the total cost, meaning that the objectives of Algorithm 1 and Baseline become more similar, causing the cost ratio to improve.

Finally, we vary the traffic demand in Figure 2d and compare three cases where the demand is 0.5X, 1X, and 1.5X the nominal demand. As the demand changes, B changes, and we recompute β and C. We observe that for the same value of α, higher demand leads to better convergence and lower costs. This is because higher demand increases the travel time, making the ℓ_2 regularizer a smaller proportion of Algorithm 1's objective. The objective functions becoming more similar leads to the cost ratio being closer to 1.

VI. Conclusion

In this paper, we study the problem of learning network routing policies from privacy-sensitive user data. We presented a new asymptotically optimal algorithm to learn privacy-preserving routing policies by solving a reformulated network flow problem using a differentially private variant of stochastic gradient descent. Finally, our simulations on a Sioux Falls road network suggests that for realistic travel demands, we can learn differentially private routing policies that result in only a 2% suboptimality in terms of total travel time. There are several interesting directions for future work. First, our algorithm can be naturally used for tracking non-stationary demand distributions (as opposed to the stationary demand model uses in this paper). Second, we can extend our formulation of request-level differential privacy, where the goal is to occlude the influence of a single trip on the algorithm's output, to a broader notion of user-level differential privacy, where the goal is to occlude the influence of all trips belonging to the same user on the algorithm's output. Finally, we chose to design a private version of gradient descent since its properties lead to a clean analysis, but similar ideas can be used for algorithms like Frank-Wolfe which are more specialized for network flow problems.

References

[1] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, "Extracting training data from large language models," in *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. USENIX Association, 2021, pp. 2633–2650.

[2] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*, vol. 3876. Springer, 2006, pp. 265–284.

[3] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive computing*, vol. 2, no. 1, pp. 46–55, 2003.

[4] J. W. Kim, K. Edemacu, and B. Jang, "Privacy-preserving mechanisms for location privacy in mobile crowdsensing: A survey," *Journal of Network and Computer Applications*, p. 103315, 2022.

[5] C.-Y. Chow, M. F. Mokbel, and X. Liu, "Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments," *GeoInformatica*, vol. 15, no. 2, pp. 351–380, 2011.

[6] C. Dwork, "Differential privacy: A survey of results," in *International conference on theory and applications of models of computation*. Springer, 2008, pp. 1–19.

[7] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *STOC 1987, New York, New York, USA*, A. V. Aho, Ed. ACM, 1987, pp. 218–229.

[8] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[9] M. Ghasemzadeh, B. C. Fung, R. Chen, and A. Awasthi, "Anonymizing trajectory data for passenger flow analysis," *Transportation research part C: emerging technologies*, vol. 39, pp. 63–79, 2014.

[10] B. Y. He and J. Y. Chow, "Optimal privacy control for transport network data sharing," *Transportation Research Part C: Emerging Technologies*, vol. 113, 2020.

[11] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1330–1341, 2018.

[12] K. Yan, G. Lu, G. Luo, X. Zheng, L. Tian, and A. M. V. V. Sai, "Location privacy-aware task bidding and assignment for mobile crowd-sensing," *IEEE Access*, vol. 7, pp. 131 929–131 943, 2019.

[13] S. Cai, X. Lyu, X. Li, D. Ban, and T. Zeng, "A trajectory released scheme for the internet of vehicles based on differential privacy," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[14] C. Xu, Y. Ding, C. Chen, Y. Ding, W. Zhou, and S. Wen, "Personalized location privacy protection for location-based services in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[15] Y. Gong, C. Zhang, Y. Fang, and J. Sun, "Protecting location privacy for task allocation in ad hoc mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 110–121, 2015.

[16] M. E. Gursoy, L. Liu, S. Truex, and L. Yu, "Differentially private and utility preserving publication of trajectory data," *IEEE Transactions on Mobile Computing*, vol. 18, no. 10, pp. 2315–2329, 2018.

[17] K. Al-Hussaeni, B. C. Fung, F. Iqbal, G. G. Dagher, and E. G. Park, "Safepath: Differentially-private publishing of passenger trajectories in transportation systems," *Computer Networks*, vol. 143, pp. 126–139, 2018.

[18] Y. Li, D. Yang, and X. Hu, "A differential privacy-based privacy-preserving data publishing algorithm for transit smart card data," *Transportation Research Part C: Emerging Technologies*, vol. 115, p. 102634, 2020.

[19] R. Dong, W. Krichene, A. M. Bayen, and S. S. Sastry, "Differential privacy of populations in routing games," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 2798–2803.

[20] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 50–64, 2017.

[21] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. F. Naughton, "Bolt-on differential privacy for scalable stochastic gradient descent-based analytics," in *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. ACM, 2017, pp. 1307–1322.

[22] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta, "Privacy amplification by iteration," in *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 2018, pp. 521–532.

[23] V. Feldman, T. Koren, and K. Talwar, "Private stochastic convex optimization: optimal rates in linear time," in *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*. ACM, 2020, pp. 439–449.

[24] V. Pandurangan, "Riding with the stars: Passenger privacy in the nyc taxicab dataset," Available Online, 2014.

[25] M. Tsao, K. Gopalakrishnan, K. Yang, and M. Pavone, "Differentially private stochastic convex optimization for network routing applications," *arXiv preprint*, 2022, extended version. Available at https://arxiv.org/abs/2210.14489.

[26] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014.

[27] "Transportation networks for research," github.com/bstabler/TransportationNetworks, 2016, accessed January 20, 2021.