Travel Time Estimation in the Age of Big Data

Dimitris Bertsimas, Arthur Delarue, Patrick Jaillet, Sébastien Martin
Operations Research Center, Massachusetts Institute of Technology

May 2018*

Abstract

Twenty-first century urban planners have identified the understanding of complex city traffic patterns as a major priority, leading to a sharp increase in the amount and the diversity of traffic data being collected. For instance, taxi companies in an increasing number of major cities have started recording metadata for every individual car ride, such as its origin, destination and travel time. In this paper, we show that we can leverage network optimization insights to extract accurate travel time estimations from such origin-destination data, using information from a large number of taxi trips to reconstruct the traffic patterns in an entire city. We develop a method that tractably exploits origin-destination data, which, because of its optimization framework, could also take advantage of other sources of traffic information. Using synthetic data, we establish the robustness of our algorithm to high variance data, and the interpretability of its results. We then use hundreds of thousands of taxi travel times observations in Manhattan to show that our algorithm can provide insights about urban traffic patterns on different scales and accurate travel time estimations throughout the network.

1 Introduction

In today's increasingly dense urban landscapes, traffic congestion is an ever more prevalent issue. As flows of goods and people increase, billions of dollars in potential savings are at stake, making the understanding of traffic patterns a major urban planning priority.

*Accepted for publication in Operations Research

A main goal of traffic studies is *travel time estimation*, which in the broadest sense consists of evaluating the time necessary to travel from any origin O to any destination D. This goal is difficult to achieve because travel times depend on a range of effects at different timescales, from the structure of the network (number of lanes on each road, speed limit, etc.) over the long term, to the state of congestion of the network over the medium term, to a host of small random events (missed lights, etc.) over the very short term. Because of the sheer number and diversity of these sources of uncertainty, most general approaches to travel time estimation consist of finding parameters that describe a distribution or set of distributions from which the travel time from O to D is sampled.

Travel time estimation is often combined with the related goal of *routing*, especially in the short to medium-term planning case. In this case, the goal is to evaluate the time necessary to travel from O to D and find at least one path that drivers could use to achieve this estimate, relating the travel time estimate to interpretable network properties. In this paper, we present a novel method to estimate typical travel times for each road in a city network using taxi data, thus providing reasonable paths and total trip time estimates for any origin and destination in the network.

1.1 The Need for a Generalized Approach to Travel Time Estimation

The problem of inferring traffic patterns from diverse measurements is a fundamental step behind the resolution of many complex questions in transportation and logistics. A simple cost function on the individual arcs of the network can often form a building block of a more complex network study, such as recent work by Pióro et al. (2016) presenting a novel understanding of resilient networks. Furthermore, many network problems specifically require a travel time estimate for each arc: for instance, Nikolova and Stier-Moses (2014), who develop a new model for traffic assignment that takes into account network uncertainty, present an approach starting from a prior estimate of the expected travel times of individual arcs in the network. Even in examples such as the aforementioned work or that of Jaillet et al. (2016), both of which generally consider travel time to be a stochastic quantity, a good estimate for the network travel times is a valuable asset in order to define a prior or an uncertainty set for this uncertain quantity, laying the groundwork to answer more complex questions about the network.

In a real-world setting, there are different ways to obtain traffic data in a network, each leading to different travel time estimation methods. A pop-

ular approach uses fixed detectors that provide information about traffic at particular points in the network, most commonly loop sensors as in Coifman (2002), or more advanced methods as in Li and Rose (2011) that exploit communication between sensors to identify the same vehicle at different locations. Another popular approach, as in Jenelius and Koutsopoulos (2013), uses so-called "floating-car" data, where GPS-equipped vehicles record their location and speed at fixed time intervals, which can range from a few seconds to a few minutes. The path followed by the vehicle between "pings" of the GPS device can be inferred in a variety of ways, from probabilistic models in Hofleitner et al. (2012) to tensor decomposition in Wang et al. (2014).

A third area of study involves easily gatherable "origin-destination" (OD) data, that only records the time and location at the beginning and at the end of a trip, as, for example, collected by taxis or cellphone towers. Logging this data instead of high-density floating-car data increases the privacy of the taxi driver and passenger because the details of the followed route are not recorded. It also treats the network as a black box, only making measurements when the user enters and exits. OD data can be gathered for different purposes, and the methods we develop here in the context of vehicle traffic can be extended to other types of networks, including railways, subways, and bicycle and pedestrian networks (see recent studies such as Hänseler et al. (2017)), or combinations of such networks. Nevertheless, this generality makes the travel time estimation harder: the problem of simultaneously determining paths and travel times based on origin-destination data only is close to the Inverse Shortest Path Length Problem (ISPL), an NP-hard problem which has also received some attention by Hung (2003).

In recent years, the New York City Taxi and Limousine Commission (NYCTLC (2016)) has maintained a complete public record of yellow cab rides within the city. The database contains relevant metadata such as the origin, destination, fare, distance and time traveled for over 170 million rides per year, and has been exploited for a variety of purposes, as shown in Yang (2015). Despite the data's size and availability, however, it has not been used very much for travel time estimation. Wang et al. (2015) develop a machine learning method based on k-nearest neighbors matching, while Santi et al. (2014) describe a very simple smoothing heuristic. Meanwhile, Zhan et al. (2013)'s more model-oriented approach develops a full probabilistic path selection scheme.

1.2 Our Contributions

The main contribution of this paper is a tractable methodology to solve the travel time estimation and routing problem in a real-world setting on a large network, which has a number of desirable properties.

First, we use very few assumptions about the data: to provide travel time estimates, we only ask for a set of trips within a known network for which an origin, a destination and a travel time are recorded. In particular, we do not require information about the demand structure in the network. We design a simple static model of traffic based on shortest path theory. This simplicity allows us to develop a multipurpose network optimization method that can leverage large amounts of high-variance origin-destination data to build an estimate of city travel times that is accurate both in and out of sample. Moreover, this method is general enough to be able to handle other sources of data, including floating cars and loop sensors.

Furthermore, the method also recovers interpretable city traffic and routing information from this potentially noisy and incomplete data. We estimate a single parameter for each edge, which enhances the interpretability of the results (see Figure 1 for an example). In order to avoid overfitting, particularly in regions of the city where little data is available, we add a simple regularization term to the model. The method provides insight on traffic patterns at the scale of a few city blocks, as well as at the scale of the entire network, and also allows us to quickly find viable paths associated with our travel time estimates.

Solving this estimation problem to optimality at an impactful scale is generally intractable. For this reason, we develop a novel iterative algorithm that provides good solutions, by solving a sequence of large second-order cone problems (SOCPs), which modern solvers can tractably handle. We verify the accuracy of this algorithm in a variety of settings and show that it provides high-quality solutions. The method is tractable, determining the typical paths and travel times in the 4300-node Manhattan network over a three-hour time window in under 20 minutes.

In Section 2, we formulate an optimization problem that gives both accurate origin-destination travel time estimates and interpretable link travel times and routing paths. In Section 3, we introduce an iterative algorithm that can compute solutions to this optimization in large scale settings. In Section 4, using synthetic data we show that the solutions of this algorithm are near-optimal and that the simplifications we made for tractability did not impact accuracy and interpretability. In Section 5, we show that this also extends to real-world situations and we present results on Manhattan taxi data.

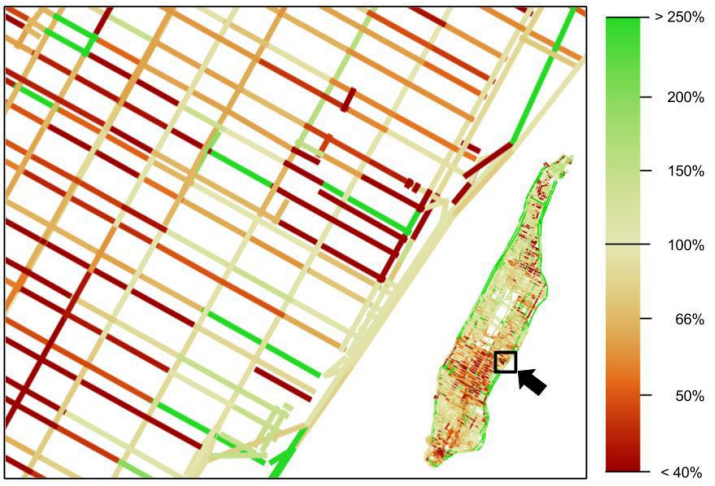


Figure 1: Close-up of Manhattan with the arc travel times estimated by our method between 9 and 11 AM. The color of each arc represents the speed along that arc as a percentage of the reference velocity $v_0 = 13.85$ kph (average velocity in Manhattan on weekdays). We can identify traffic effects at the scale of the city (Midtown congestion) and at the scale of a single street (the ramp onto the highway on the eastern shore of Manhattan is congested).

2 Methodology

In this section, we define the probabilistic setting of travel time estimation, and introduce a simple traffic model that leverages the knowledge of the routing network to represent travel time estimates in the lower dimensional space of network arc travel times. This allows us to create an optimization formulation that uses origin-destination data to build an interpretable image of the network travel times, and at the same time provide accurate travel time estimates.

2.1 Problem Statement: Estimating Travel Times From Data

Data. We consider a road network, represented as a directed graph $G = (V, E)$. On this graph, we are given a data set of origin-destination travel time values in the network, of the form (o, d, T) with $(o, d) \in V \times V$ the origin and destination nodes and $T \in \mathbb{R}^+$ the corresponding observed travel time. Data corresponding to this general description can be obtained in many different ways. For example, the set of observed travel times for taxi

trips that started between 12pm and 1pm on 2016 Wednesdays in Manhattan would be a valid example of such a data set, as would the set of stop-to-stop travel times for Boston school buses in the academic year 2016-17.

Some origin-destination pairs have several travel time observations in the data set, while others have none. We can therefore define $W \subset V \times V$ as the subset of origin-destination pairs for which we have data: for each $(o, d) \in W$ we are given travel times $\{T_{od}^i\}_{i=1}^{n_{od}}$, realized on n_{od} distinct trips from o to d.

Probabilistic Setting. We would like to estimate the times of trips that are "similar" to the trips that are given in the data set, but may not have been observed in the data. In other words, given any origin-destination pair $(o, d) \in V^2$, we would like to provide a point estimate $\hat{T}_{od}$ of the time it takes to go from o to d. To properly define these estimates for all origin-destination pairs, we describe a simple probabilistic setting.

Each observation of the data-set is assumed to be independently sampled from the same probability distribution. This sampling process goes as follows: the origin and destination nodes (o, d) are sampled from a discrete distribution $\mathcal{D}$ in V^2. Then, conditioned on having an origin o and a destination d, the observed travel times $\{T_{od}^i\}_{i=1}^{n_{od}}$ are assumed to be sampled independently from the distribution $\mathcal{D}_{od}$. Note that $\mathcal{D}_{od}$ can be different for each (o, d). We assume that our data set was built by following this sampling process, but that we do not know the distributions $\mathcal{D}$ or $\mathcal{D}_{od}$. We will hold these probabilistic assumptions to be true throughout this paper, including our experiments on synthetic data in Section 4. In Section 5, we will show that our results extend to real-world data that does not necessarily verify our probabilistic assumptions.

We want to obtain a point estimate $\hat{T}_{od}$ of the distribution $\mathcal{D}_{od}$ for every pair $(o, d) \in V \times V$. Specifically, we would like to estimate the geometric mean of the distribution $\mathcal{D}_{od}$: $\exp(\mathrm{E}_{T_{od} \sim \mathcal{D}_{od}}[\log(T_{od})])$. We choose the geometric mean instead of the standard mean because we think that the quality of travel time estimations is perceived on a multiplicative rather than an additive scale, as we discuss in the next paragraph.

To understand the choice of estimating the geometric mean, note that the geometric means of all the distributions $\mathcal{D}_{od}$ are estimates that minimize

the overall mean squared log error (MSLE) :

$$\text{MSLE}((\hat{T}_{od})_{(o,d)\in V^2}) = \mathbb{E}_{(o,d)\sim\mathcal{D},T_{od}\sim\mathcal{D}_{od}}\left[\left(\log(\hat{T}_{od}) - \log(T_{od})\right)^2\right] \tag{1}$$

$$= \mathbb{E}_{(o,d)\sim\mathcal{D},T_{od}\sim\mathcal{D}_{od}}\left[\left(\log(T_{od}) - \mathbb{E}_{T_{od}\sim\mathcal{D}_{od}}\left[\log(T_{od})\right]\right)^2\right] \tag{2}$$

$$+ \mathbb{E}_{(o,d)\sim\mathcal{D}}\left[\left(\log(\hat{T}_{od}) - \mathbb{E}_{T_{od}\sim\mathcal{D}_{od}}\left[\log(T_{od})\right]\right)^2\right]. \tag{3}$$

Note that the expectations are taken with respect to the distributions $\mathcal{D}$ and $\mathcal{D}_{od}$. The MSLE decomposes into the mean log variance of the data (2) which is independent of our estimates, and the mean squared log bias (MSLB) (3) which is a measure of the distance of each estimate $\hat{T}_{od}$ from the geometric mean of $\mathcal{D}_{od}$. Using the MSLE implies that an estimate that is twice an observed value is equally bad as an estimate that is half of it. Additionally, a 30-second estimation error is a lot worse for a trip that last 2 minutes than for a trip that lasts 15 minutes. This is what we want and why we chose the log scale and geometric mean estimates.

Model. In practice, we do not observe all the possible (o, d) pairs, which makes it hard to estimate the geometric mean of $\mathcal{D}_{od}$ using only the data that has origin o and destination d. Nonetheless, the estimates of the distributions $\mathcal{D}_{od}$ are typically related: for example, a trip from o to d and a trip from o' to d where o and o' are geographically close will have similar travel time estimates. Therefore, we leverage the network structure by introducing parameters t_{ij} that represent the typical travel time along any arc $(i, j) \in E$, and use them to compute our estimates $\hat{T}_{od}$.

We define a path P_{od} from o to d as a series of consecutive arcs (without cycles), starting at o and ending at d, and $\mathcal{P}_{od}$ to be the finite set of all possible paths from o to d. For each possible path $P_{od} \in \mathcal{P}_{od}$, we model the point estimate of the total travel time along this path to be $\hat{T}_{P_{od}} = \sum_{(i,j)\in P_{od}} t_{ij}$. Because our data set provides no information as to which path was followed to realize a given travel time, we assume that drivers use the fastest paths available. We thus select $\hat{P}_{od} = \operatorname{argmin}_{P_{od}\in\mathcal{P}_{od}} \sum_{(i,j)\in P_{od}} t_{ij}$, and define our point estimate to be $\hat{T}_{od} = \hat{T}_{\hat{P}_{od}} = \sum_{(i,j)\in\hat{P}_{od}} t_{ij}$. As a consequence, given the parameters t_{ij}, our model chooses the point estimates $\hat{T}_{od}(\mathbf{t}) = \min_{P_{od}\in\mathcal{P}_{od}} \sum_{(i,j)\in P_{od}} t_{ij}$, where $\mathbf{t}$ is simply shorthand for the vector $(t_{ij})_{(i,j)\in E}$ (following standard boldfaced vector notation).

To use this model, we must only provide $|E|$ parameters, which is generally much less than the $|V|^2$ estimates we want to obtain. The model is

also interpretable, as we expect the values t_{ij} to be representative of the typical travel-times along arc $(i,j) \in E$. We acknowledge that the shortest-path assumption itself can be questioned. From a behavioral standpoint, taxi drivers may have other objectives in mind, such as maximizing revenue or minimizing fuel consumption; in addition, Dial (1971) showed that shortest paths can be sensitive to changes in travel time. However, we find that despite this modeling assumption, our results on real data are interpretable and reasonably accurate.

Parameter Estimation. We want to use the observed travel times T^i_{od} to estimate the model parameters t_{ij}. Following our goal to have estimates as close as possible to the geometric mean of $\mathcal{D}_{od}$, we want to find the values of t_{ij} that minimize the MSLE of the estimates $\hat{T}_{od}$. Because the distributions $\mathcal{D}_{od}$ and $\mathcal{D}$ are unknown, we approximate them with the empirical distribution of our observations and we obtain the following minimization problem:

$$\min_{\mathbf{t}} \sum_{(o,d)\in W} \sum_{i=1}^{n_{od}} (\log \hat{T}_{od}(\mathbf{t}) - \log T^i_{od})^2, \tag{4}$$

which is equivalent to

$$\min_{\mathbf{t}} \sum_{(o,d)\in W} n_{od} (\log \hat{T}_{od}(\mathbf{t}) - \log T_{od})^2, \tag{5}$$

where $T_{od} = \left(\prod_{i=1}^{n_{od}} T^i_{od}\right)^{1/n_{od}}$, the geometric mean of all the observed travel times from o to d.

Regularization. In order to generalize well out of sample, we need to add a regularization term to the empirical MSLE. This is important because we may not have sampled enough data from $\mathcal{D}$ and $\mathcal{D}_{od}$, and the empirical MSLE (4) may not be a good approximation of the MSLE (1). Leveraging our knowledge of the city network, we hypothesize that two similar intersecting or consecutive roads should have similar traffic speeds by default. Two arcs (i,j) and (k,l) are called *neighboring* when they represent consecutive or intersecting roads with the same "type". These types are defined through our knowledge of the routing network, and differentiate highways, major arteries and smaller roads. The neighboring relationship is written as $(i,j) \leftrightarrow (k,l)$. This regularization is somewhat unusual in traffic studies, but it is effective in practice and will only influence our estimation when we do not have enough data. Adding the regularization term to our objective yields:

$$\sum_{(o,d)\in W} n_{od} \left(\log \hat{T}_{od} - \log T_{od}\right)^2 + \lambda \sum_{(i,j)\leftrightarrow(k,l)} \left| \frac{t_{ij}}{d_{ij}} - \frac{t_{kl}}{d_{kl}} \right| \frac{2}{d_{ij} + d_{kl}}, \tag{6}$$

where d_{ij} corresponds to the length in meters of the arc (i, j) in the routing network, $\sum_{(i,j)\leftrightarrow(k,l)}$ represent the sum over all pairs of neighboring arcs (i, j) and (k, l) and the parameter λ represents the strength of the regularization. In other words, we minimize the difference in speed of neighboring roads, with the weighting factor $2/(d_{ij}+d_{kl})$ ensuring that continuity is more important in shorter neighboring roads (where constant velocity is a better approximation) than in longer ones.

2.2 MIO Formulation

We can now estimate the parameters t_{ij} from the data by solving the following mixed-integer formulation with linear constraints and a non-linear objective:

$$\min_{\hat{\mathbf{T}},\,\mathbf{t},\mathbf{z}} \quad \sum_{(o,d)\in W} n_{od}\left(\log \hat{T}_{od} - \log T_{od}\right)^2 + \quad \lambda \sum_{(i,j)\leftrightarrow(k,l)} \left|\frac{t_{ij}}{d_{ij}} - \frac{t_{kl}}{d_{kl}}\right| \frac{2}{d_{ij}+d_{kl}} \tag{7a}$$

$$\text{s.t.} \quad \hat{T}_{od} \le \sum_{(i,j)\in P^\ell_{od}} t_{ij} \qquad \forall (o,d)\in W,\ P^\ell_{od}\in\mathcal{K}_{od} \tag{7b}$$

$$\hat{T}_{od} \ge \sum_{(i,j)\in P^\ell_{od}} t_{ij} - M(1-z^\ell_{od}) \qquad \forall (o,d)\in W,\ P^\ell_{od}\in\mathcal{K}_{od} \tag{7c}$$

$$\sum_{\ell} z^\ell_{od} = 1 \qquad \forall (o,d)\in W \tag{7d}$$

$$z^\ell_{od}\in\{0,1\} \qquad \forall (o,d)\in W, \ell\in\{1,\dots,|\mathcal{K}_{od}|\} \tag{7e}$$

$$t_{ij} \ge a_{ij} \qquad \forall (i,j)\in E. \tag{7f}$$

The objective (7a) is the parameter estimation cost introduced in (6). For each $(o, d) \in W$, the constraints enforce that $\hat{T}_{od} = \min_{P_{od}\in\mathcal{K}_{od}} \sum_{(i,j)\in P_{od}} t_{ij}$, i.e. $\hat{T}_{od}$ is the time of the shortest path from o to d out of all the paths in $\mathcal{K}_{od}$. This non-linear shortest path constraint is enforced using the binary variables z^l_{od} that represent which path $P^\ell_{od} \in \mathcal{K}_{od}$ is the shortest path, together with the constraints (7b), (7d) and the big-M constraints (7c). Typically, $\mathcal{K}_{od} = \mathcal{P}_{od}$ is the set of all paths from o to d, but the formulation generalizes to any other subset $\mathcal{K}_{od} \subset \mathcal{P}_{od}$. Finally, the constraints (7f) introduce the bounds a_{ij} to enforce a speed limit on the arc travel times t_{ij}.

2.3 Iterative Path Generation

For each $(o, d) \in W$, formulation (7) requires one binary variable for each path going from o to d. The number of paths is typically exponential in the size of the graph, so we need to reduce the number of paths to consider if we want to be able to solve (7). It turns out the formulation is naturally suited for an iterative approach. Assume we start with a small set of paths $\mathcal{P}_{od}^0$ for every origin-destination pair in the dataset. We can solve the problem in (7) by considering the set of paths $\mathcal{K}_{od} = \mathcal{P}_{od}^0$ instead of the much larger $\mathcal{K}_{od} = \mathcal{P}_{od}$. This yields values of t_{ij}, for which we can recompute new shortest paths in the network using any shortest-path algorithm. If for a given (o, d), the new shortest path has length less than $\hat{T}_{od}$, then we know that the minimum path length computed over $\mathcal{P}_{od}^0$ is not equal to the minimum path length over $\mathcal{P}_{od}$. In this case, we add the new shortest path P_{od}^1 to our set of paths, obtaining the set $\mathcal{P}_{od}^1 = \mathcal{P}_{od}^0 \cup \{P_{od}^1\}$. We can then re-solve (7) using $\mathcal{P}_{od}^1$ instead of $\mathcal{P}_{od}^0$, and iterate this process. If instead the new shortest path for each (o, d) has length equal to $\hat{T}_{od}$, then we know we have already found reasonable paths, reaching a stopping point for the algorithm. The algorithm thus generates an increasing list of path candidates $\mathcal{P}_{od}^k$ for each iteration k and $(o, d) \in W$, so that the shortest paths P_{od}^k are added to the path candidates of the next iteration, e.g. $\mathcal{P}_{od}^{k+1} = \mathcal{P}_{od}^k \cup \{P_{od}^k\}$.

This iterative approach is inspired by cutting plane algorithms in linear optimization. In practice, most paths between o and d are not remotely close to being the shortest and would never even be considered by drivers looking to travel from o to d. Although this iterative method does not necessarily converge to the global optimum of (7) with $\mathcal{K}_{od} = \mathcal{P}_{od}$, we will show empirically that it yields good results for large problems, does not exhibit pathological local optima when used with appropriate regularization and typically converges in a few steps. Additionally the algorithm is always interpretable: the solution at any iteration k corresponds to the optimal solution of the problem if the drivers only consider the paths in $\mathcal{P}_{od}^k$.

3 Solving Large-Scale Problems

Even with the iterative path generation presented in 2.3, the optimization problem (7) cannot be tractably solved for most problems of interest. The main reasons are that the objective is non-convex, and that there are at least $O(|W|)$ binary variables, which makes it impossible for state-of-the-art solvers to give interesting solutions in a reasonable time for problems with more than 1000 data-points and routing networks that represent real cities.

Actually, solving this problem to optimality relates to the problem of path reconstruction in a graph (sometimes called the Inverse Shortest Path Length problem), an NP-hard problem, as discussed in Hung (2003). We present a tractable approach that produces good solutions, allowing us to handle hundreds of thousands of data points in networks with tens of thousands of arcs.

3.1 Adapting the shortest path constraint

In order to handle a large number of data points, we need to discard the binary variables z_{od}^{ℓ} introduced in (7e). One way to do this is to modify the constraint $\hat{T}_{od} = \min_{P_{od} \in \mathcal{K}_{od}} \sum_{(i,j) \in P_{od}} t_{ij}$. An interesting solution can be built by fixing the values of the binary variable, i.e., choosing which path should be the shortest for each $(o, d) \in W$. Indeed, if the shortest path in $\mathcal{K}_{od}$ is chosen to be P_{od}^{*}, then the shortest path constraints (7b)-(7e) trivially become:

$$\hat{T}_{od} = \sum_{(i,j) \in P_{od}^{*}} t_{ij} \qquad \forall (o, d) \in W, \tag{8a}$$

$$\hat{T}_{od} \geq \sum_{(i,j) \in P_{od}} t_{ij} \qquad \forall (o, d) \in W,\ P_{od} \in \mathcal{K}_{od}. \tag{8b}$$

For this formulation to become useful, we need a clever way to choose P_{od}^{*} for each $(o, d) \in W$. Our iterative path generation algorithm introduced in Section 2.3 provides a good candidate. At iteration k, the algorithm computes the shortest path P_{od}^{k} for each $(o, d) \in W$. This path can be viewed as our "best estimate" of the true path at iteration k, and is one of the paths we consider at iteration $k + 1$. For this reason, we choose to use this path as the chosen shortest path for the next iteration $k + 1$, setting $P_{od}^{*} = P_{od}^{k}$.

In the end, the results on synthetic data in Section 4 and on real data in Section 5 show that this method, appropriately regularized, yields interpretable high-quality solutions and empirically converges. Our intuition is the following: this path estimation may not seem perfect, but the tractability gains allow us to use orders of magnitude more data, which will improve the accuracy of the t_{ij} parameters and the $\hat{T}_{od}$ estimates, thus allowing us to compute better paths at each iteration.

3.2 Towards a Convex Objective

The left term in the minimization objective (7a) is nonconvex and not easily optimized by traditional optimization solvers. We want to find a surrogate

that is convex, tractable, and a good approximation of the original squared log cost. More specifically, we want to find a convex loss function ℓ such that $\ell(\hat{T}_{od}, T_{od}) = (\log(\hat{T}_{od}) - \log(T_{od}))^2 + o((\hat{T}_{od} - T_{od})^2)$, and such that ℓ is also unbiased in the multiplicative space, i.e. $\ell(aT_{od}, T_{od}) = \ell(\frac{T_{od}}{a}, T_{od})$ for any scalar $a > 0$.

A good candidate is the maximum ratio loss: $\ell(\hat{T}_{od}, T_{od}) = \left(\max\left(\frac{T_{od}}{\hat{T}_{od}}, \frac{\hat{T}_{od}}{T_{od}}\right) - 1\right)^2$. It is a convex function of the variable $\hat{T}_{od}$ that has all the desired properties. Our objective thus becomes:

$$\sum_{(o,d)\in W} n_{od} \left(\max\left(\frac{T_{od}}{\hat{T}_{od}}, \frac{\hat{T}_{od}}{T_{od}}\right) - 1\right)^2 + \lambda \sum_{(i,j)\leftrightarrow(k,l)} \left|\frac{t_{ij}}{d_{ij}} - \frac{t_{kl}}{d_{kl}}\right| \frac{2}{d_{ij} + d_{kl}} \tag{9}$$

We want to be able to solve the corresponding optimization with hundreds of thousands of data-points. To the best of our knowledge, only state-of-the-art LP and SOCP solvers are able to handle formulations with hundreds of thousands of variables and constraints. As a consequence, we would like to slightly modify our formulation to be able to formulate it as an SOCP. All we need to do is replace the squared losses by absolute values, yielding the modified objective:

$$\sum_{(o,d)\in W} n_{od} \max\left(\frac{T_{od}}{\hat{T}_{od}}, \frac{\hat{T}_{od}}{T_{od}}\right) + \lambda \sum_{(i,j)\leftrightarrow(k,l)} \left|\frac{t_{ij}}{d_{ij}} - \frac{t_{kl}}{d_{kl}}\right| \frac{2}{d_{ij} + d_{kl}} \tag{10}$$

This new objective allows us to reformulate each iteration as an SOCP:

$$\min_{\hat{\mathbf{T}}, \mathbf{t}, \mathbf{x}} \quad \sum_{(o,d)\in W} n_{od} x_{od} + \lambda \sum_{(i,j)\leftrightarrow(k,l)} \left|\frac{t_{ij}}{d_{ij}} - \frac{t_{kl}}{d_{kl}}\right| \frac{2}{d_{ij} + d_{kl}} \tag{11a}$$

$$\text{s.t.} \quad \hat{T}_{od} = \sum_{(i,j)\in P^*_{od}} t_{ij} \qquad \forall (o,d) \in W, \tag{11b}$$

$$\hat{T}_{od} \geq \sum_{(i,j)\in P_{od}} t_{ij} \qquad \forall (o,d) \in W,\ P_{od} \in \mathcal{K}_{od}, \tag{11c}$$

$$x_{od} \geq \frac{\hat{T}_{od}}{T_{od}} \qquad \forall (o,d) \in W, \tag{11d}$$

$$x_{od} \geq \frac{T_{od}}{\hat{T}_{od}} \qquad \forall (o,d) \in W, \tag{11e}$$

$$t_{ij} \geq a_{ij} \qquad \forall (i,j) \in E. \tag{11f}$$

where $x_{od} = \max\left(\frac{T_{od}}{\hat{T}_{od}}, \frac{\hat{T}_{od}}{T_{od}}\right)$. The objective can be formulated as linear, and all the constraints are linear except (11e), which can be reformulated as the following second-order cone constraint:

$$\left(x_{od} + \hat{T}_{od}\right) \geq \left\| \begin{pmatrix} \hat{T}_{od} - x_{od} \\ 2\sqrt{T_{od}} \end{pmatrix} \right\|. \tag{12}$$

Replacing the squared losses by absolute values makes our new formulation more robust to outliers and more tractable, but weakens the case for replacing the observations T^i_{od} that share the same (o, d) with their geometric mean T_{od}. Once more, we trade some modeling rigor for the ability to use more data, and we will show that this choice is empirically justified.

3.3 A Tractable Algorithm

We now summarize our tractable algorithm for large-scale static travel time estimation.

1. Choose a regularization parameter λ and an initial set of arc travel-times: $(t^0_{ij})_{(i,j)\in E}$. We will show in the next sections that our results are not sensitive to these choices. For each $(o, d) \in K$, start with an empty set of paths $\mathcal{P}_{od} = \emptyset$. Then start Step 2 with iteration $k = 1$.

2. For each iteration k, do the following:

3. Use an efficient, parallelized shortest-path algorithm to compute all the shortest paths $(P^k_{od})_{(o,d)\in W}$, using the arc travel-times $(t^{k-1}_{ij})_{(i,j)\in E}$. Add these paths to the previous set of paths $\mathcal{P}^k_{od} = \mathcal{P}^{k-1}_{od} \cup \{P^k_{od}\}$. If there is a limit Π on the number of paths we can store (for memory or tractability reasons), remove the path of $\mathcal{P}^k_{od}$ with the longest travel-time to make sure that $|\mathcal{P}^k_{od}| \leq \Pi$.

4. Solve the optimization problem (11), using the newly computed shortest paths $P^*_{od} = P^k_{od}$, to obtain the new arc travel-times $(t^k_{ij})_{(i,j)\in E}$.

5. If a convergence criterion is met, stop the algorithm and return the times $(t^k_{ij})_{(i,j)\in E}$. Else, start iteration $k + 1$ and go to Step 2.

In the end, our algorithm returns a set of arc travel-times, that can be used to compute shortest paths and travel time estimations $\hat{T}_{od}$ for any origin-destination pair in the network. We propose a convergence criterion based on path differences.

Definition 1 (Path difference)**.** Given a node pair (o, d) and two paths P_{od}^A and P_{od}^B, the path difference $d(P_{od}^A, P_{od}^B)$ is defined as the average of the number of arcs in P_{od}^A that are not in P_{od}^B and the number of arcs in P_{od}^B that are not in P_{od}^A.

At each iteration k, we can compute the path difference between the new path P_{od}^k and the path of the previous iteration P_{od}^{k-1} for each $(o, d) \in W$. We stop our algorithm when the mean path difference across all $(o, d) \in W$ is less than a threshold δ, i.e. $\frac{1}{|W|} \sum_{(o,d)\in W} d(P_{od}^k, P_{od}^{k-1}) < \delta$. In this paper, we fix δ to a small value: $\delta = 0.5$. We chose this value because we noticed that our estimates $\hat{T}_{od}$ were not improving in subsequent iterations, for the specific applications of this paper. In this situation, the algorithm tends to converge in less than 10 iterations.

3.4 A General Model

The ability to solve the travel time estimation and routing problem using only origin-destination data is useful because it makes minimal assumptions on the format of the data. However, in some cases more data is available, for instance from loop sensors or floating car probes (see Section 1.1). Due to its optimization-based framework, our method is flexible enough to handle many additional forms of data.

The method presented in the previous section is designed under the assumption that for every (o, d) in the set of input node pairs $W \subset V \times V$, we are only given a finite number of sample travel times, from which we compute a geometric mean T_{od}, with no information about the path taken by the drivers. Constraint (11b) reflects the algorithm's attempt to guess the correct path, assuming that the drivers are trying to minimize driving times. If we assume now that for some $(o, d) \in W$, we are given not only a travel time T_{od}^{obs}, but also the used path P_{od}^{obs}, then we can add a term in the objective penalizing the distance between the observation T_{od}^{obs} and the sum $\sum_{(i,j)\in P_{od}^{\text{obs}}} t_{ij}$ of link travel times along the path P_{od}^{obs}.

Another form of traffic data that is commonly available comes from loop sensors/traffic cameras, which can sometimes measure traffic velocity on a given set of arcs $L \subseteq E$. For example, Wang and Nihan (2000) shows that a single loop detector on a highway is enough to provide accurate speed estimates. A velocity measurement on arc (i, j) is easily integrated into our model, by adding a term in the objective that penalizes the distance of t_{ij} from its measurement.

Thus, though our method is designed with minimal data in mind, it can easily incorporate additional information about the network. In a world

where more and more data is available, but formats may differ greatly from source to source, an optimization-based approach allows for the easy integration of complementary information, yielding a multipurpose method to solve the problem of travel time estimation and routing.

4 Performance on Synthetic Data

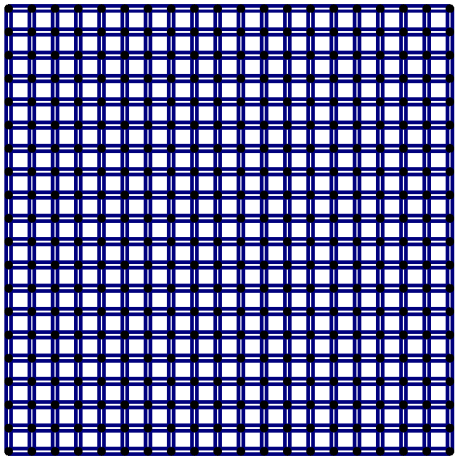
When developing a tractable algorithm in Section 2 and 3, we made several simplifying assumptions about driver behavior and network properties, and the complexity of our optimization formulation led to several heuristic simplifications. It is hard to verify if the tractable iterative algorithm presented in Section 3 provides good solutions to our original problem presented in Section 2.2 using real-world travel time data. Indeed, real data does not always follow our model's assumptions. This is why we first use synthetic data verifying our model's assumptions to study the convergence of our tractable algorithm as an approximation of the original formulation presented in Section 2, and then show in Section 5 that our model generalizes well to real-world data in terms of interpretability and accuracy.

Therefore, the goal of this section is twofold: first, we show that despite its heuristic steps, our approach to solving the optimization problem in Section 3 converges to a good estimate $\hat{T}_{od}$ of $\mathcal{D}_{od}$ in the log space, while recovering interpretable parameters t_{ij} that represent the local congestion states in the city. Second, we show that even with high variance travel time distributions $\mathcal{D}_{od}$ and very incomplete observations ($|W| << |V|^2$), we are still able to generalize well and recover good estimates $\hat{T}_{od}$ for all $(o, d) \in V^2$, when the synthetic data is generated following our modeling assumptions.

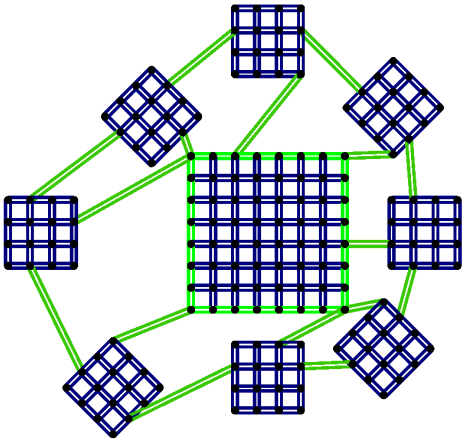
4.1 Synthetic Networks and Virtual Data

In order to test our method on synthetic data, we create simple model networks in which we attempt to reconstruct traffic patterns. One model reproduces some features of a city, with a central "downtown" area (8×8 square grid), surrounded by suburbs (4×4 square grids) and circled by highways (with higher speed limits) that connect each suburb to the central area and to the two closest neighboring suburbs. This network is shown in Figure 2b. For more advanced testing, we use a larger 20×20 square grid, with which we investigate a range of traffic patterns.

Once we have constructed the routing graphs, we create the synthetic travel time distributions $\mathcal{D}$ and $\mathcal{D}_{od}$. Each observation (o, d, T) is generated as follows: first, the distribution $\mathcal{D}$ is chosen to be uniform over all origin-



(a) Simple square network, with 400 nodes and 1520 arcs. There are two arcs between any adjacent nodes (one in each direction). All arcs are of the same type, and consequently they all have the same maximum speed.



(b) Toy model of a city, with 192 nodes and 640 arcs. The green roads are highways, with much higher speed limits (and consequently lower travel times proportional to their length).

Figure 2: Model networks used to test our travel time estimation and routing algorithm.

destination pairs in V^2. In practice, taxi trips are not uniformly distributed over the city network; however, we will see in the Section 5 that our model performs well with real-world observations that are far from being uniformly distributed. Then, $\mathcal{D}_{od}$ is chosen to be lognormal with log-mean parameter $\mu = \log(T_{od}^{\text{real}})$ and a second parameter σ that controls the randomness of travel times from o to d. For context, $\mathcal{D}_{od}$ has geometric mean T_{od}^{real}, and a value of $\sigma = \log 2 \approx 0.7$ means that a sampled time T_{od}^{i} is within one geometric standard deviation of the T_{od}^{real} if it is between $0.5T_{od}^{\text{real}}$ and $2T_{od}^{\text{real}}$. The values T_{od}^{real} are chosen to follow our shortest-path model. Therefore, we set a deterministic value of the parameter t_{ij}^{real} for each arc (i, j), and define $T_{od}^{\text{real}} = \min_{P_{od} \in \mathcal{P}_{od}} \sum_{(i,j) \in P_{od}} t_{ij}^{\text{real}}$. As a consequence, T_{od}^{real} are the best estimates of the distributions $\mathcal{D}_{od}$ given our shortest path model and the estimation loss introduced in Section 2.1.

We then use this process to sample N observations of origin-destination travel-times. For each (o, d) independently, we would need several samples to be able to estimate T_{od}^{real} (because of the randomness σ), but the routing network model of our algorithm allows us to be able to use much less samples to provide accurate point estimates for $\mathcal{D}_{od}$ (i.e. close to T_{od}^{real}), even when $(o, d) \notin W$.

4.2 Results

We evaluate the quality of our estimation using the Root Mean Squared Log Error (RMSLE) of our estimates $\hat{T}_{od}$, defined as the square root of the MSLE (1). Interestingly, the formulation simplifies when using the lognormal distributions:

$$\text{RMSLE}(\hat{T}_{od}) = \sqrt{\sigma^2 + \mathbb{E}_{(o,d)\sim\mathcal{D}}\left[\left(\log(\hat{T}_{od}) - \log(T_{od}^{real})\right)^2\right]}. \tag{13}$$

To make it easier to compare our estimations across different values of σ, we focus on the square root of the MSLB (RMSLB), removing the contribution of the log variance σ^2 of the data (see (3)).

$$\text{RMSLB}(\hat{T}_{od}) = \sqrt{\sum_{(o,d)\in V^2}\left(\log(\hat{T}_{od}) - \log(T_{od}^{real})\right)^2}, \tag{14}$$

where we used that $\mathcal{D}$ is uniform over V^2. Note that RMSLB $= 0$ means that we recover the geometric expectation of the travel times exactly.

We present the effects of our method when run on the city models described in the previous section, with a few different travel time functions and

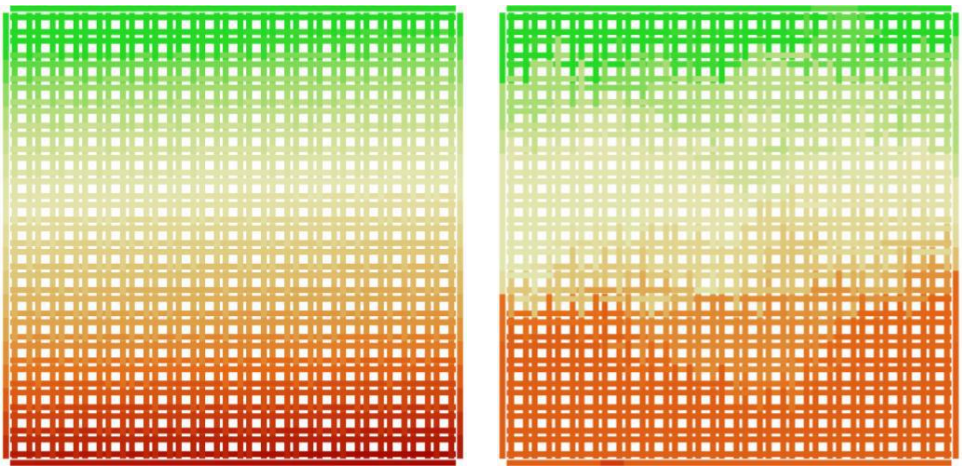
	Available amount of data			
	$N = 100$	$N = 500$	$N = 1,000$	$N = 10,000$
Randomness of input data	RMSLB of estimation			
$\sigma = 0.0$	0.08	0.03	0.01	0.01
$\sigma = 0.1$	0.09	0.06	0.03	0.02
$\sigma = 0.5$	0.23	0.12	0.15	0.05
$\sigma = 1.0$	0.48	0.22	0.20	0.07
$\sigma = 2.0$	0.62	0.43	0.30	0.15

Table 1: RMSLB (Root Mean Squared Log Bias) of estimation for a varying amount of data and randomness σ. RMSLB of estimation for a varying amount of data N and a varying amount of travel time randomness on the toy city model (see Fig. 2b). The toy city used has just under 37,000 node pairs (192 nodes), but notice that we need very little data to create an estimate with small bias.

data generated as described above. We begin by studying the toy model of a city introduced in Figure 2b. The values t_{ij}^{real} are chosen by road type, with one speed for regular streets and another for the highways. We sample N travel time observations as described in Section 4.1, and we start with random arc travel times to define the initial path P_{od}^0 for each (o, d) in W. In Table 1 we present results of our method for different values of N and σ.

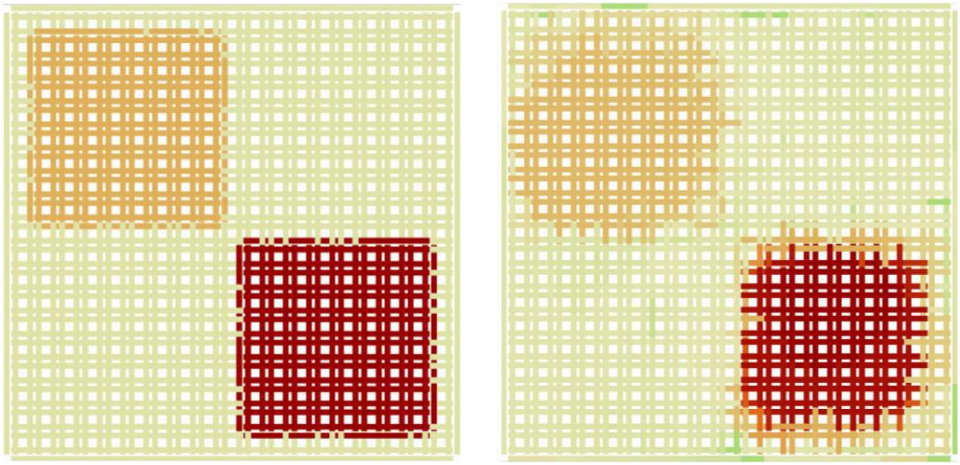
For all values of σ, when setting $\delta = 0.5$ we find that the method tends to converge in under 10 iterations. Each iteration on this small network (192 nodes, 640 arcs) takes less than 10 seconds for a total run-time of less than two minutes. We noticed that the regularization term in the objective greatly speeds up convergence by reducing the relevance of tiny path differences. In addition, Table 1 confirms the rather obvious fact that results are more accurate with more data and less randomness in the data (top left corner). However, it also reveals that when the input has a high log variance σ^2 it is possible to obtain an estimate with comparatively small bias with very little data. For example, when $\sigma = 2$, it is possible to obtain an estimation bias that is smaller by more than a factor of two with only 100 observations, i.e., less than 2% of the total origin-destination pairs.

The results in Table 1 should be taken with a grain of salt, however, as the toy model in Fig. 2b is intentionally suited to the assumptions with which we developed our model (especially the velocity continuity assumption of our regularization). The goal of this experiment is simply to confirm that the method converges as intended and produces sensible results. The next step



(a) True congestion. (b) Reconstruction.

Figure 3: Results of our algorithm on a square network for a congestion gradient. Green arcs have a higher velocity and thus a lower travel time, while red arcs are more congested and thus have a lower velocity and a higher travel time. All arcs are of the same type, with a maximum velocity of 50 kph. The estimates $\hat{T}_{od}$ are computed using $N = 5,000$ observations. The Root mean squared log bias (RMSLB) of the estimates is 0.041, which is over eight times smaller than the input log standard deviation $\sigma = 0.35$. The algorithm effectively reconstructs high-level traffic patterns and provides accurate travel time estimates despite extremely noisy data. For arcs in each of the four quarters from the top, the true velocity is respectively 60%, 30%, 20%, and 15% of the maximum velocity. The gradient from Figure 3a is clearly visible despite some noise.



(a) True congestions (b) Reconstruction

Figure 4: Results of our algorithm on a square network with two congested neighborhoods. Green arcs have a higher velocity and thus a lower travel time, while red arcs are more congested and thus have a lower velocity and a higher travel time. All arcs are of the same type, with a maximum velocity of 50 kph. The true velocity is 30% of the maximum velocity for arcs in the upper-left neighborhood, 15% for arcs in the lower-right neighborhood, and 60% for arcs outside these neighborhoods. The estimates $\hat{T}_{od}$ are computed using $N = 5,000$ observations. The RMSLB of the estimates is 0.069, which is over eight times smaller than the input log standard deviation $\sigma = 0.35$.

is to consider a model that does not follow our traffic assumptions as closely. We therefore focus on the square network (400 nodes), which we associate with two traffic configurations, presented in Figures 3 and 4, corresponding to semi-realistic scenarios, including a gradual north-to-south increase in travel time (Fig. 3a), and two congested neighborhoods where arc travel times are doubled and quadrupled (Fig. 4a) as compared to the rest of the city. Readers will note that these scenarios break our road velocity continuity assumption in different ways: the first because the velocities of neighboring vertical arcs are never equal, and the second because the borders between congested and uncongested areas are strongly discontinuous in terms of velocity.

For each of the two scenarios thus described, we sample $N = 5,000$ observations as described in Section 4.1, for $\sigma = 0.35$ (we choose this value because it is approximately our estimate of the log standard deviation of Manhattan taxi travel time data).

The arc travel times found by our algorithm are shown in Figures 3b and 4b. The algorithm does a remarkable job reconstructing the travel times in the network given limited data. As noted above, the data provided was noisy ($\sigma = 0.35$), yet the RMSLB for the estimated travel times T_{od} over all (o, d) in W is 0.07 in one case and 0.04 in the other. Therefore, the algorithm not only produces accurate travel times estimates for the origin-destination pairs for which no data was available, it does so with minimal bias when compared to the high randomness and sparsity of the travel time data. Notice that the regularization term in the objective, though based on a questionable traffic assumption, does not preclude us from reconstructing the arc travel times as desired in both cases, though it does make it difficult to find the exact border of the congested neighborhoods in one case, and the precise velocity gradient in the other.

All in all, the method developed in this paper is able to extract useful information from high-variance inputs, and produces a network cost function, in the form of arc travel times, that is interpretable and can in turn be used for other applications in the network. In the following section, we show that all the algorithm properties displayed in this section, namely low estimation bias despite inputs with high randomness, and the production of an interpretable final solution, also hold in a real-world setting at much larger scales.

5 Performance on Real-World Data

So far, we have described, implemented and tested a methodology to solve the travel time estimation and routing problem. We use a network formulation because we assume the only allowed origins and destinations are nodes in the

graph. However, real origin-destination data records vehicles' starting and ending points using GPS coordinates, which are continuous variables.

In this section, we first provide a bridge between the continuous and discrete problems in order to apply our method to real-world OD data, and present the results on data from New York City taxis. We then display the results of our method for varying amounts of available data, showing that our method provides both accurate travel time estimations throughout the network and sensible routing information, for an interpretable understanding of traffic in the city.

5.1 A Large-Scale Data Framework

A major contribution of this paper is the ability to exploit a large dataset to solve the travel time estimation and routing problem for the real-world network of a large city. Providing a tractable method at this scale requires the construction of a substantial framework to handle large amounts of network information and origin-destination data, allowing us to leverage big data insights in solving a complex problem.

In order to solve the travel time estimation and routing problem in a real-world setting, it is necessary to overcome two major challenges. The first difficulty is to extract a network structure from a complex urban landscape, and specifically to identify a graph that is elaborate enough to capture most of the details of the city under study, but simple enough to tractably support our network optimization methods. For this purpose, we use open-source geographical data from the OpenStreetMap project. Its database provides a highly-detailed map of New York City, which we simplify by excluding walkways and service roads, and removing nodes that do not represent the intersection of two or more roads. For the island of Manhattan, to which we restrict our problem, we obtain a strongly connected graph with 4324 nodes and 9518 arcs. This network is quite large, and the tractability of our method on a map of this size is itself a significant contribution of this paper: readers will realize that an algorithm seeking to estimate travel times in this network must consider over 18 million origin-destination pairs of nodes and at least that many shortest paths.

The second challenge is obtaining and cleaning real OD data. Data from the New York City Taxi and Limousine Commission for the years 2009-2016 is freely available from NYCTLC (2016). A month's worth of data (approximately 2GB) contains information for over 12 million taxi trips (over 400,000 a day). We perform all computations, network and data handling using the Julia programming language. Our method's tractability is enhanced by the use of the cutting-edge Julia for Mathematical Programming

(JuMP) interface by Lubin and Dunning (2015), which enables us to take advantage of top-of-the-line linear and second-order programming methods, as implemented by the commercial solvers Gurobi and Mosek. Therefore, our framework can handle problem instances considering hundreds of thousands of data points in the entirety of Manhattan.

The results presented in Sections 5.3 and 5.4 use taxi data from weekdays in May 2016, in the time windows 9-11AM (morning), 6-8PM (evening), and 3-6AM (night). We restrict the data to a single month to reduce the taxi trip variance. Smaller time windows also guarantee less noisy data, but at the cost of fewer data points, and so we opt for a medium-sized window of a few hours. Our method therefore seeks to capture network patterns that are averaged over the considered time window.

In order to eliminate extreme outliers, we ignore trips shorter than 30s and longer than 3 hours, trips connecting points that are less than 250m or more than 200km away, and trips which would require an average speed greater than 110 kph or less than 2 kph to make sense. The existence of such unrealistic trips is a consequence of the imperfection of the GPS sensors inside the taxi meters. After this filtering step, we split the data into a training set containing about 415,000 trips and a testing set containing about 275,000 trips. In the next section, we explain how we adapt this taxi data to our discrete network-based framework.

5.2 Applying a Discrete Model to Real-World Data

The model described in Section 2 is discrete in space and static in time: it considers fixed traffic patterns during a given time window in a network where the only possible start and end locations are intersections. In contrast, real-world data is continuous in time and space: a given taxi trip is associated with a start time and an end time recorded by a clock within the taxi meter, and with start and end locations that are recorded using often noisy GPS sensors. We therefore need to process the data a bit further for it to work with our model.

In the database, each taxi trip is represented as a 6-tuple $(x_O, y_O, x_D, y_D, t_{\text{start}}, tt_{OD})$, where x_O and y_O are the GPS coordinates of the origin, x_D and y_D are the GPS coordinates of the destination, t_{start} is the date and time of the beginning of the ride, and tt_{OD} is the travel time of the taxi from its origin to its destination. We use t_{start} to assign taxi trips to time intervals of length τ, and consider only this time window, discarding all taxi trips that do not start inside this interval. For taxi trips that do start within the interval, we do not differentiate between different t_{start} values, so each trip is reduced to the 5-tuple $(x_O, y_O, x_D, y_D, tt_{OD})$.

The length τ of the time interval should be chosen based on the scope of the application. If the goal is static planning, we can select a large value of τ (from a few hours to a few months), which will allow us to consider a large amount of data, and estimate fixed travel time parameters over the interval as accurately as possible. If the goal is short-term dynamic planning, we can pick a small value of τ, say 5 minutes, and use the small amount of data in this interval to quickly estimate the travel time parameters, which we will only assume to be valid for the next time interval.

In the discrete formulation presented in the previous sections, the input of the method is a set of node pairs W, with a known (but possibly noisy) travel time T_{od} for each (o, d) in the set W. In the continuous problem, the inputs are position vectors (x_O, y_O, x_D, y_D) and associated travel times tt_{OD}. We therefore need to project the continuous origin (x_O, y_O) and destination (x_D, y_D) onto the network to be able to use our discrete methods in this real-world setting.

There exist many methods of projecting continuous data onto a discrete network model (see recent work by Quddus and Washington (2015), Chen et al. (2014)); all our results were obtained by projecting each continuous origin-destination pair (x_O, y_O, x_D, y_D) to the nearest node pair (o, d) using the Euclidean metric in $\mathbb{R}^4$. We can now apply our algorithm to real taxi data in Manhattan.

5.3 Evaluating Results at the Scale of the City

Accuracy. We have explained in this paper that real-world OD data has significant variance, originating from several main sources: the imprecision of the data-gathering protocol, including potent "urban canyoning" effects in GPS data, as well as the inherent variance of traffic patterns (see Section 2.1). The latter source is especially important when the time window is long, as a consequence of our static traffic modeling assumption.

As seen in Section 2, the mean squared log error (MSLE) decomposes into the sum of the mean log variance of the data and the mean squared log bias of our estimate. On empirical data, we can only evaluate the MSLE using (4). In order to be able to evaluate the performance of our estimation, we need to estimate the log variance of the travel time observations (2). Indeed, it is a lower bound on the MSLE of our estimate, and a low-bias estimate must have an MSLE as close as possible to this lower bound.

For this purpose, we simply compare each taxi trip in the data to the average of the k trips closest to it, and compute the empirical log variance between the two values. This gives us an upper bound on the log variance term of the MSLE of our estimate. Because the dataset is quite large, this

bound is indicative, especially when we choose the value of k that minimizes it. For instance, this approach yields an input log variance of $0.31^2 = 0.10$ for the time window 9-11 AM. To understand the magnitude of this variance, consider a mean time of 20 minutes. A trip within one standard deviation of the mean could last any amount of time between $20e^{-0.31} = 14.7$ minutes and $20e^{0.31} = 27.3$ minutes.

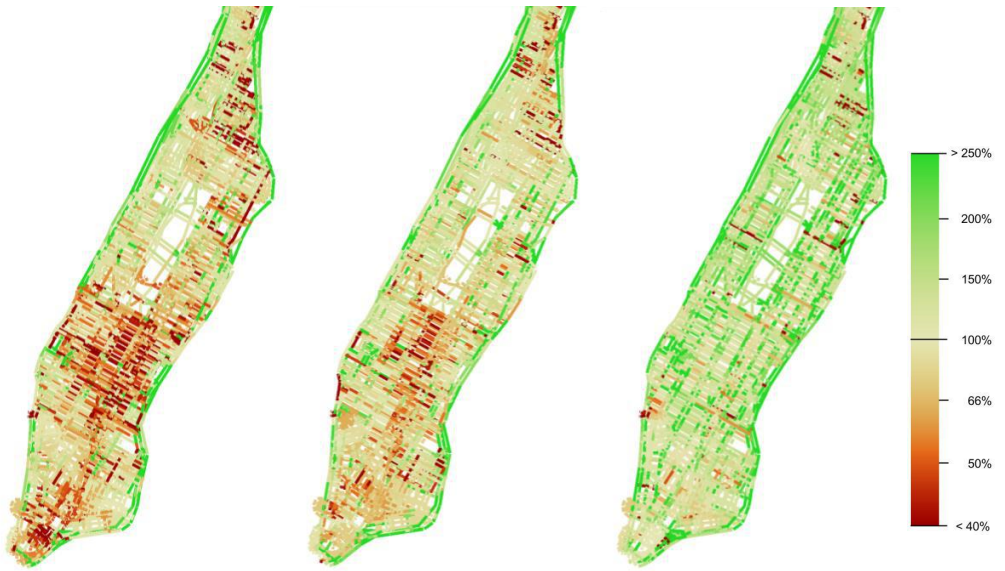
We will measure the accuracy of our method by how close the MSLE of our estimate is to the log variance (2) of the data. This is a proxy for minimizing the mean squared log bias (3), which is what we did in Section 4 when we new the distributions $\mathcal{D}_{od}$. If the difference between the MSLE of our estimations and the input log variance of the data is small, it means that our estimations are very close to the geometric mean of the network travel times, and most of our error comes from the inherent variability of the taxi trips.

For tractability reasons, we restrict the size of the input node pairs set W to 100,000 (o, d) pairs. With $|W| = 100,000$, the total computation at the scale of Manhattan takes less than 2 hours. We choose the regularization parameter $\lambda = 1000$, which is the value that minimized the MSLE in cross-validation. We note that the algorithm converges in 10 iterations without showing noticeable cycling (the out-of-sample improves at each iteration).

It turns out that between 9 and 11 AM, the out-of-sample RMSLE of our estimations is just over 0.36. This result means that our travel time estimation error is barely worse than the inherent noise in the data, and our estimated travel times must therefore be very close to the geometric expectation of the travel times throughout the network.

Interpretability. In addition to its accuracy, we argue that our method provides global insights about traffic patterns in New York. To show this, we compare our results in Manhattan in the morning (9-11AM), in the evening (6-8PM) and at night (3-6AM). We show the edge travel times for these time windows in Figures 5 and 6. The overall traffic patterns are easily identifiable in Figure 5, in particular the effect of the morning (and to a lesser extent, evening) commute in Midtown, as well as the congestion in the northern part of the island near the bridges connecting it to the mainland.

The results of our algorithm provide insights at a variety of scales: in addition to displaying citywide effects such as the daily commute, they also reveal more subtle realities about traffic in New York: For example, when looking at Figures 6a and 6b, it is clear that crosstown (east-west) travelers are much more exposed to congestion than uptown-downtown (north-south) travelers, and that the highways on Manhattan's eastern and western shores



(a) Morning (9-11AM) (b) Evening (6-8PM) (c) Night (3-6AM)

Figure 5: Edge travel times in Manhattan estimated by our algorithm on weekdays in May 2016 in the morning, evening and at night. The color of each edge represents the speed along that edge as a percentage of the reference velocity $v_0 = 13.85$ kph (average velocity in Manhattan on weekdays). At the scale of the city, the algorithm clearly identifies morning commute congestion in Midtown and the Financial District, while at the scale of individual city blocks, it confirms the empirically known fact that crosstown (east-west) traffic in Manhattan is much more congested than uptown-downtown (north-south) traffic.

(a) 9-11AM (close-up) (b) 6-8PM (close-up) (c) 3-6AM (close-up)

Figure 6: Zoom on edge travel times in Manhattan estimated by our algorithm on weekdays in May 2016 for 9-11AM and 3-6AM. Detail of Figure 5. The color of each edge represents the speed along that edge as a percentage of the reference velocity $v_0 = 13.85$ kph (average velocity in Manhattan on weekdays). The gap in congestion between crosstown traffic and uptown-downtown traffic is also visible at this much smaller scale.

Time of the day	Training trips	Mean trip time	Out-of-sample RMSLE
09-11 AM	415,106	13m54s	0.31
06-08 PM	545,965	11m54s	0.30
03-06 AM	75,339	07m38s	0.28

Table 2: Effect of the time of the day on the taxi-trip dataset and the estimation power of our method. Note that the number of trips available and the root mean squared log error (RMSLE) depends on the time of the day. As a consequence, the time-window choice plays an important role in the quality of our estimation. Figures 5 and 6 represents the corresponding network travel times.

(FDR Drive and Riverside Drive) are much faster routes than Manhattan's inner streets.

More detailed error results regarding the morning, evening and night time windows are available in Table 2.

A Robust and Sensible Path Estimation. At each iteration of the algorithm, the total path difference decreases, which means the algorithm finds a stable solution to the travel time estimation and routing problem. Moreover, we also note that the algorithm always converges to a similar choice of path, independently of the choice of initial paths and arc travel times. To support this claim, we show in Figures 7 and 8 the evolution of a path between a given origin and destination over random restarts of the

algorithm. Specifically, we initiate the algorithm with random times: each edge has a velocity drawn randomly between 1 and 130 kph. This results in the random initial paths shown in panes 7a, 7b, and 7c. After 5 iterations, we consider the paths obtained by our method, in panes 8a, 8b, and 8c.

We see that in all three cases, the algorithm made the justifiable decision of using the freeway on the western edge of Manhattan. In addition, despite stark differences in the starting point, the final paths found by the method are eerily similar. One can quibble about the exact level of similarity between these final paths, but it is wise to remember that our method does not seek to obtain the "optimal" path between an origin o and a destination d (and indeed Dial (1971) questions the existence of such an optimal path in a noisy environment), but simply a reasonable path that achieves the estimated travel time. Figure 8 is an example of our method accomplishing this stated purpose.

To provide intuition for why our routes seem sensible, note that we have empirically established that the travel time estimation accuracy in Manhattan has low bias, as we showed that the MSLE was close to our estimate of the mean log variance of the data. Additionally, the regularization allows us to generalize well to parts of the city with few observations. Furthermore, the obtained arc travel time parameters t_{ij} are good estimations on synthetic data and seem reasonable in NYC. All these observations lead us to hypothesize that the obtained paths are sensible.

This result means that in a network with almost ten thousand nodes, with only a few OD pairs relative to the possible 18 million pairs, using high variance data, we are able to reconstruct the all-pairs shortest paths that minimize the error between the shortest path lengths and the input data. Our optimization-based algorithm thus exhibits a certain number of important properties: it is tractable at the scale of a large and complex city, produces accurate travel time estimations and sensible routing information despite high variance data, and produces an overall traffic map of the city that can be used for numerous other applications. These results are obtained with a large number of data points, and in fact we operate at the limit of what our solvers can handle. In the following section, we explore the effect of reducing available data on our method's accuracy.

5.4 Impact of Data Density and Comparison with Data-Driven Methods

The results presented in Section 5.3 show that, when run with a large number of data points, our method tractably estimates travel times in Manhattan.

(a) Random initial path 1.

(b) Random initial path 2.

(c) Random initial path 3.

Figure 7: Evolution of paths studied by our algorithm : original paths. Three random starting point paths are presented. See Figure 8 for the results of the algorithm.

Given this performance with a wealth of data, it is natural to wonder how our algorithm fares when the data is much more sparse. Good performance in data-poor environments is important for two reasons: first, few cities have as much demand for taxis as New York, so extending the method to other networks would necessitate good behavior with only minimal amounts of data. Second, taxis do not necessarily explore networks in a uniform manner: even in cities such as New York where they represent a significant fraction of traffic, taxis only seldom visit certain neighborhoods, creating data-rich and data-poor settings within a single network.

Nearest Neighbor Travel Time Estimation. In this section, we choose to compare the performance of our method to simple purely data-driven schemes, which are expected to work very well in a data-rich setting and comparatively less well in a data-poor setting. Indeed, the formulation of the real-world travel time estimation problem as the estimation of tt_{OD} as a function of the four variables x_O, y_O, x_D, and y_D suggests simple solution approaches based solely on the data. With no knowledge of the network or the underlying behavior of taxi drivers, it is possible to use machine learning to infer travel times.

For instance, a simple k-nearest neighbors scheme would match an input

(a) Path 1 after 5 iterations.

(b) Path 2 after 5 iterations.

(c) Path 3 after 5 iterations.

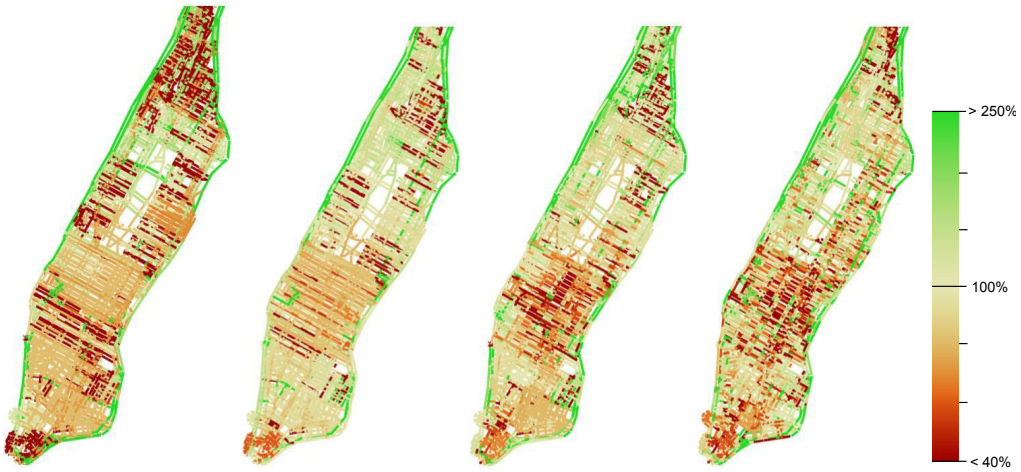
Figure 8: Evolution of paths studied by our algorithm : path convergence. Shows the resulting path to which the algorithm converges after 5 iterations, starting from the initial paths and times presented in Figure 7. The reader will notice that despite strong differences in the starting paths, the algorithm eventually converges to a very reasonable solution, a path that makes use of the freeway on the western shore of Manhattan.

origin and destination $(x_O, y_O, x_D, y_D) \in \mathbb{R}^4$ to the k taxi trips in the database closest to it (for some choice of metric) and compute the geometric average of their travel times to produce an estimate for the travel time between the provided origin and destination.

This scheme has the advantage of being extremely simple and allowing for quick travel time estimations. In addition, it is easy to see that the bias of this travel time estimate would converge to zero as the number of observation increase (if k is scaled appropriately). Indeed, this approach is not limited by the low-dimensional model assumption of our algorithm. However, in practice it has several drawbacks: first, it is not particularly well suited to travel time estimation for origins and destinations for which little data is available. This is a particularly damaging flaw because, as stated before, origin-destination data is not very complete and is concentrated in regions with more taxi traffic.

Second, this pure data-driven approach does not address the routing aspect of the problem: with no knowledge of the network it cannot possibly provide information as to which path should be used. These two drawbacks justify the use of our more complicated network optimization approach, but the k-nearest neighbors scheme remains a useful benchmark of our performance. Of course, we do not expect to produce more accurate estimates than a k-nearest neighbors scheme when a wealth of taxi trips is available. With a good method, however, we should be able to obtain more accurate travel times than k-nearest neighbors in zones without much data, and only slightly less accurate in zones where data is plentiful.

High Accuracy in Data-Poor Environments. To evaluate the impact of the dataset size on our method, we compute travel times and paths for varying amounts of training data and compare the obtained RMSLE values on the testing set with those produced by the k-nearest neighbors approach. We also compare our results to those produced by the travel time estimation method in Zhan et al. (2013) (for the small amounts of data where it is tractable). The results are presented in Table 3: though, as expected, the k-nearest neighbors scheme outperforms the optimization method for high amounts of data, it is significantly less accurate in a data-poor setting. Meanwhile, Zhan et al. (2013)'s method is less accurate and also untractable for more than a small number of trips, as the runtime was 10-20 times longer than ours (due to the much larger size of the network as compared to the one used by the authors). Looking at the results, it seems that with our method, simply recording the origin, destination and travel time of 100 taxi trips is enough to accurately estimate the traffic patterns in an entire city.



(a) $N = 10^2$ (b) $N = 10^3$ (c) $N = 10^4$ (d) $N = 10^5$

Figure 9: Edge travel times in Manhattan (9-11AM) estimated by our algorithm for an increasing number of input taxi trips N. The color of each edge represents the speed along that edge as a percentage of the reference velocity $v_0 = 13.85$ kph (average velocity in Manhattan on weekdays). With just 100 taxi trips, the algorithm is able to identify that Midtown is generally congested, especially in the area around Times Square and Penn Station, and that the shoreline highways are very fast. As N increases, congestion patterns become more precise, and smaller congested areas become apparent, for example around freeway ramps. For $N = 100,000$ (the largest size that allows our algorithm to converge in less than 2 hours), we obtain a detailed, edge-by-edge description of Manhattan traffic, without losing sight of global congestion patterns.

	k-NN		Optimization		Zhan et al. (2013)
Training trips	Best k	RMSLE	Best λ	RMSLE	RMSLE
100,000	16	0.3243	1e3	0.3595	-
10,000	11	0.3636	1e3	0.3775	-
1,000	7	0.4296	1e3	0.4019	0.8228
100	6	0.5556	1e3	0.4495	0.8822

Table 3: Effect of data density on k-nearest neighbors (k-NN), our optimization method, and Zhan et al. (2013)'s method on the out-of-sample RMSLE. The best values of k and the continuity parameter λ are chosen. As before, the convergence threshold δ is set to 0.5. For large amounts of data, k-nearest neighbors is unsurprisingly more accurate than our optimization-based method (although not by much), but it performs much worse in a low-data environment. Zhan et al. (2013)'s method is 10-20 times slower than ours (untractable for 10,000 trips or more), and is less accurate.

The accuracy gap between our method and k-nearest neighbors is noticeable, especially when you consider that our method also provides a path for any (o, d) pair in the network, which a simple k-nearest neighbors scheme can never provide since it has no knowledge of the network. Therefore, in a data-poor environment, our scheme is superior to a purely data-driven one in terms of accuracy and routing, and both methods have a running time that is appropriate for the application (a few seconds for k-nearest neighbors, a few minutes for our method). In a higher-data environment we pay for the added routing information with a decrease in accuracy of just fractions of a minute and an increase in computational time.

6 Conclusions

The method proposed in this paper leverages a simple approach to tractably yield accurate solutions to the travel time estimation and routing problem in a real-world setting. Given trip times for any number of origin-destination pairs, from a few hundred to a few hundred thousand, we can estimate the travel time from any origin to any destination, as well as provide a sensible path associated with this travel time. Furthermore, our algorithm is robust to a high degree of input uncertainty, successfully exploiting very noisy data to provide results characterized by their accuracy.

Providing travel times for each arc in the city effectively augments the network with a cost function based on real traffic information, which can be

of use both for city planners and for further network-based research. Using our optimization-based framework, we can estimate traffic patterns in a real-world network, providing insight at every scale, from a few blocks to the entire city, and extracting global meaning from the observed data.

Acknowledgement

We would like to thank the area editor Prof. Anton Kleywegt, the associate editor and three reviewers for many careful and insightful comments that improved the paper significantly. Research funded in part by ONR grants N00014-12-1-0999 and N00014-16-1-2786. Geographical data for New York City is copyrighted to OpenStreetMap contributors and available from OSM (2015).

References

Chen BY, Yuan H, Li Q, Lam WHK, Shaw SL, Yan K (2014) Map-matching algorithm for large-scale low-frequency floating car data. *Int. J. Geogr. Inf. Sci.* 28(1):22–38.

Coifman B (2002) Estimating travel times and vehicle trajectories on freeways using dual loop detectors. *Transportation Research Part A: Policy and Practice* 36(4):351 – 364.

Dial RB (1971) A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation Research* 5(2):83–111.

Hänseler F, Molyneaux N, Bierlaire M (2017) Estimation of pedestrian origin-destination demand in train stations. *Transportation Science* 51(3):981–997.

Hofleitner A, Herring R, Abbeel P, Bayen A (2012) Learning the dynamics of arterial traffic from probe data using a dynamic bayesian network. *IEEE Transactions on Intelligent Transportation Systems* 13(4):1679–1693.

Hung CH (2003) *On the Inverse Shortest Path Length Problem*. Ph.D. thesis, Georgia Tech ISyE.

Jaillet P, Qi J, Sim M (2016) Routing optimization under uncertainty. *Operations Research* 64(1):186–200.

Jenelius E, Koutsopoulos HN (2013) Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological* 53:64 – 81.

Li R, Rose G (2011) Incorporating uncertainty into short-term travel time predictions. *Transportation Research Part C: Emerging Technologies* 19(6):1006 – 1018.

Lubin M, Dunning I (2015) Computing in operations research using julia. *INFORMS Journal on Computing* 27(2):238–248.

Nikolova E, Stier-Moses NE (2014) A mean-risk model for the traffic assignment problem with stochastic travel times. *Operations Research* 62(2):366–382.

NYCTLC (2016) Trip record data. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`.

OSM (2015) OpenStreetMap Project Database. `http://planet.openstreetmap.org`.

Pióro M, Fouquet Y, Nace D, Poss M (2016) Optimizing flow thinning protection in multicommodity networks with variable link capacity. *Operations Research* 64(2):273–289.

Quddus M, Washington S (2015) Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transportation Research Part C: Emerging Technologies* 55:328 – 339.

Santi P, Resta G, Szell M, Sobolevsky S, Strogatz S, Ratti C (2014) Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences* 111(37):13290–13294.

Wang H, Li Z, Kuo Y, Kifer D (2015) A simple baseline for travel time estimation using large-scale trip data. *CoRR* abs/1512.08580.

Wang Y, Nihan NL (2000) Freeway Traffic Speed Estimation Using Single Loop Outputs. *Transportation Research Record: Journal of the Transportation Research Board* 1727(1):9.

Wang Y, Zheng Y, Xue Y (2014) Travel time estimation of a path using sparse trajectories. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 25–34 (KDD 2014).

Yang C (2015) *Data-driven modeling of taxi trip demand and supply in New York City*. Ph.D. thesis, Rutgers University.

Zhan X, Hasan S, Ukkusuri SV, Kamga C (2013) Urban link travel time estimation using large-scale taxi data with partial information. *Transportation Research Part C: Emerging Technologies* 33:37 – 49.