A Simple Baseline for Travel Time Estimation using Large-Scale Trip Data

HONGJIAN WANG, Twitter Inc., USA
XIANFENG TANG, Pennsylvania State University, USA
YU-HSUAN KUO, Pennsylvania State University, USA
DANIEL KIFER, Pennsylvania State University, USA
ZHENHUI LI, Pennsylvania State University, USA

The increased availability of large-scale trajectory data provides rich information for the study of urban dynamics. For example, New York City Taxi & Limousine Commission regularly releases source/destination information of taxi trips, where 173 million taxi trips released for Year 2013 [29]. Such a big dataset provides us potential new perspectives to address the traditional traffic problems. In this paper, we study the travel time estimation problem. Instead of following the traditional route-based travel time estimation, we propose to simply use a large amount of taxi trips without using the intermediate trajectory points to estimate the travel time between source and destination. Our experiments show very promising results. The proposed big data-driven approach significantly outperforms both state-of-the-art route-based method and online map services. Our study indicates that novel simple approaches could be empowered by the big data and these approaches could serve as new baselines for some traditional computational problems.

CCS Concepts: • **Information systems** → *Nearest-neighbor search*; • **Computing methodologies** → *Unsupervised learning*; • **Applied computing** → *Transportation*;

Additional Key Words and Phrases: Travel time estimation, big data, baseline, trajectory data

ACM Reference Format:
Hongjian Wang, Xianfeng Tang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2018. A Simple Baseline for Travel Time Estimation using Large-Scale Trip Data. *ACM Trans. Intell. Syst. Technol.* 1, 1 (October 2018), 22 pages. https://doi.org/0000001.0000000

1 INTRODUCTION

The positioning technology is widely adopted into our daily life. The on-board GPS devices track the operation of vehicles and provide navigation service, meanwhile a significant amount of trajectory data are collected. For example, a huge amount of taxi trip data is accumulated and enables us to study urban dynamics. While many existing methods try to outdo each other in terms of complexity and algorithmic sophistication, in the spirit of "big data beats algorithms", we study whether some simple baseline methods can be empowered by taking the benefit of big data.

Authors' addresses: Hongjian Wang, Twitter Inc. 1355 Market St. #900, San Francisco, CA, 94103, USA, hongjianw@twitter.com; Xianfeng Tang, Pennsylvania State University, Westgate Building, University Park, PA, 16802, USA, xut10@ist.psu.edu; Yu-Hsuan Kuo, Pennsylvania State University, Westgate Building, University Park, PA, 16802, USA, yzk5145@cse.psu.edu; Daniel Kifer, Pennsylvania State University, W333 Westgate Building, University Park, PA, 16802, USA, dkifer@cse.psu.edu; Zhenhui Li, Pennsylvania State University, E331 Westgate Building, University Park, PA, 16802, USA, jessieli@ist.psu.edu.

2157-6904/2018/10-ART $15.00
https://doi.org/0000001.0000000

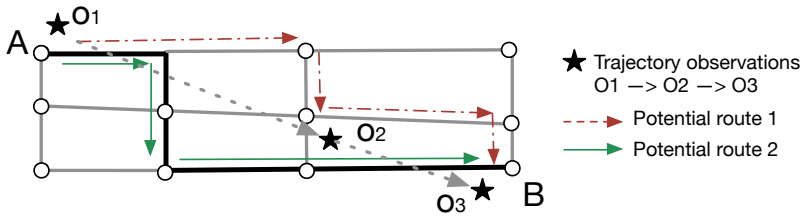


Fig. 1. Use historical trajectory $o_1 \rightarrow o_2 \rightarrow o_3$ to estimate the travel time from A to B.

In this paper, we revisit a traditional travel time estimation problem, which estimates the travel time between an origin and a destination. Existing travel time estimation approaches mostly fall into the line of *route-based* method [4, 5, 13, 31]. Given a source and a destination, these methods first identify a route and then estimate the travel time for this route by aggregating the travel time spent on each segment (or subpath) based on historical trajectories. Such a route-based method faces many complicated challenges in implementation. In particular, there are two major challenges:

- *Route mapping.* The route-based method needs to obtain the travel time from historical trajectories. But the actual trajectory data may not map well onto the roads due to GPS positioning errors, or imprecise road network data, or the time gap between two adjacent observations (for example, in the Shanghai taxi data used in our experiments, the GPS sample interval ranges from 60 to 90 seconds). We illustrate this problem in Figure 1. Suppose we are estimating the travel time from A to B. We need to first identify a route (the black road segments). Next, to estimate the travel time for this route, we need to know the travel time spent on each segment. And the time spent on each segment is obtained from historical trajectories. One historical trajectory $o_1 \rightarrow o_2 \rightarrow o_3$ is shown on Figure 1. However, it is non-trivial to map these GPS points to the actual road segments. Two potential routes are plotted in Figure 1 and both could be taken by the observed trajectory. In addition, we often have missing data between two consecutive data points. Therefore, it is not clear what is the actual route taken by an observed trajectory.
- *Data sparsity.* Trajectory data are sparse. Even with millions of trip observations, there are a lot of road segments not covered by any trajectory because of the non-uniform spatial distribution of GPS locations in the city. For example, in the Shanghai taxi data, even with over 300 million GPS records, more than 50% of road segments are not covered by any trajectory. The inference of travel time is not accurate due to limited number of trajectories covering the road segment. In addition to the spatial sparseness, the trajectory data are temporally sparse too. Even if a road segment is covered by a few trajectories, these trajectories may not be sufficient to estimate the dynamic travel time that varies under different conditions (e.g., peak time and weekends).

To avoid these challenges in the route-based method, we propose a simple baseline method to estimate the travel time. Suppose we are to predict travel time from A to B and we have hundreds of historical trips from A to B. Even if we do not have the intermediate points for those trips, the average travel time of these historical trips could provide us a reasonable estimation of the travel time from A to B. This is similar to the way we humans tackle this problem. For example, if someone asks you how long it takes to drive from your office to home, you do not calculate the time based

on the travel time on each road segments. Instead, you would estimate an approximate time based on the historical travel times.

But in the real-world setting, even with a large database of millions of trips, it is not easy to get sufficient number of trips from exactly the same origin A to exactly the same destination B. Therefore, we look for neighboring trips with a nearby origin and destination to estimate the travel time and we call our method a *neighbor-based method*. However, simply taking the average time of neighboring trips will yield a poor performance because of the variance in traffic at different times. For example, the travel time during the peak hour could be much longer than that in the midnight; there could also be abnormal traffic changes due to holidays or traffic jams. Therefore, we further propose to consider such traffic dynamics by exploiting traffic periodicity and making timely adjustments based on recent traffic patterns.

One may argue that our neighbor-based method does not provide the specific route information, which limits its application. While this is true, it does not keep our method from being a good baseline for travel time estimation problem. Besides, there are many real scenarios where the route information is not important. One example is the trip planning [14]. For example, a person will take a taxi to catch a flight, which leaves at 5pm. Since he is not driving, the specific route is less of a concern and he only needs to know the trip time. The second example is to estimate the city commuting efficiency [19] [17]. In the field of urban transportation research, the commuting efficiency measure is a function of observed commuting time and expected minimum commuting time of given origin and destination areas. To estimate the commuting efficiency of a city for a future time, the ability to predict the pairwise travel time for any two areas is crucial. In this scenario, the specific path between two areas is not important, either.

We conduct experiments on two large real-world datasets. We evaluate our method using more than 450 million NYC trips, and our method is able to outperform Bing Maps [16] by almost 30% (even when Bing uses traffic conditions during the same relative time within a week as query trips). Since NYC taxi data only contain the information on endpoints of the trips (i.e., pick-up and drop-off locations), to compare our method with route-based method, we use Shanghai taxi data with more than 5 million trips with complete intermediate points of trajectories. On this dataset, our method also significantly outperforms the state-of-the-art route-based method [27] by 19% and Baidu Maps by 17%. It is noteworthy that our method runs 40 times faster than state-of-the-art route-based method.

In summary, the contributions of this paper are as follows:

- We propose to estimate the travel time using neighboring trips from the large-scale historical data. To the best of our knowledge, this is the first work to estimate travel time without computing the routes.
- We improve our neighbor-based approach by addressing the dynamics of traffic conditions.
- Our experiments are conducted on large-scale real data. We show that our method can outperform state-of-the-art methods as well as online map services (Bing Maps and Baidu Maps).

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem and provides an overview of the proposed approach. Sections 4 discusses our method to weight the neighboring trips. We present our experimental results in Section 5 and conclude the paper in Section 6.

2 RELATED WORK

To the best of our knowledge, most of the studies in literature focus on the problem of estimating travel time for *a route* (i.e., a sequence of locations). There are two types of approaches for this problem: *segment-based method* and *path-based method*.

Segment-based method. A straightforward travel time estimation approach is to estimate the travel time on *individual road segments* first and then take the sum over all the road segments of the query route as the travel time estimation. There are two types of data that are used to estimate the travel time on road segments: *loop detector data* and *floating-car data* (or probe data) [23]. Loop detector can sense whether a vehicle is passing above the sensor. Various methods [12, 20, 22, 26] have been proposed to infer vehicle speed from the loop sensor readings and then infer travel time on individual road segments.

Floating cars collect timestamped GPS coordinates via GPS receivers on the cars. The speed of individual road segments at a time t can be inferred if a floating car is passing through the road segment at that time point [2, 30]. Due to the low GPS sampling rate, a vehicle typically goes through multiple road segments between two consecutive GPS samplings. A few methods have been proposed to overcome the low sampling rate issue [3, 8, 9, 11, 28]. Another issue with floating-car data is data sparsity – not all road segments are covered by vehicles all the time. Wang et al. [27] proposes to use matrix factorization to estimate the missing values on individual road segments. In our problem setting, we assume the input is a special type of the floating-car data, where we only know the origin and destination points of the trips.

Path-based method. Segment-based method does not consider the transition time between road segments such as waiting for traffic lights and making left/right turns. Recent research works start considering the time spent on intersections [6–8, 15]. However, these methods do not directly use sub-paths to estimate travel time. Rahmani et al. [21] and Wang et al. [27] propose to concatenate sub-paths to give a more accurate estimation of the query route. Our proposed solution can be considered as an extreme case of the path-based method, where we use the travel time of the *full paths* in historical data to estimate the travel time between an origin and a destination. Full paths are better at capturing the time spent on all intersections along the routes, assuming we have enough full paths (i.e., a reasonable support).

Origin-destination travel time estimation. In our problem setting, instead of giving a route as the input query, *we only take the origin and destination as the input query*. One could argue that, we could turn the problem into the trajectory travel time estimation by first finding a route (e.g., shortest route or fastest route) [4, 5, 13, 31] and then estimating the travel time for that route. This alternative solution is similar to those provided by online maps services, where users input a starting point and an ending point, and the service generates a few route options and their corresponding times. Such solution eventually leads to travel time estimation of a query route. However, the historical trajectory data may only have the starting and ending points of the trips, such as the public NYC taxi data [29] used in our experiment. Different from the data used in literature which has the complete trajectories of floating cars, such data have limited information and are not suitable for the segment-based method or path-based method discussed above. In addition, route-based approach introduces more expensive computations because we need to find the route first and then compute travel time on segments or sub-paths. More importantly, such expensive computation does not necessarily lead to better performance compared with our method using limited information, as we will demonstrate in the experiment section.

3 PROBLEM DEFINITION

A **trip** $\mathbf{p}_i$ is defined as a 5-tuple $(o_i, d_i, s_i, l_i, t_i)$, which consists of the origin location o_i, the destination location d_i and the starting time s_i. Both origin and destination locations are GPS coordinates. We further use l_i and t_i to denote the distance and travel time for this trip, respectively. Note that, here we assume the intermediate locations of the trips are not available. In real world applications, it is quite possible that we can only obtain such limited information about trips due to privacy concerns and tracking costs. For example, the largest public taxi data released by New York City [29] does not contain any intermediate GPS points.

Problem 1 (OD Travel Time Estimation). *Suppose we have a database of trips,* $\mathcal{D} = \{\mathbf{p}_i\}_{i=1}^{N}$. *Given a query* $\mathbf{q} = (o_q, d_q, s_q)$, *our goal is to estimate the travel time* t_q *with given origin* o_q, *destination* d_q, *and departure time* s_q, *using the historical trips in* $\mathcal{D}$.

3.1 Approach Overview

An intuitive solution is that we should find similar trips as the query trip $\mathbf{q}$ and use the travel time of those similar trips to estimate the travel time for $\mathbf{q}$. The problem can be decomposed to two sub-problems: (1) how to define similar trips; and (2) how to aggregate the travel time of similar trips. Here, we name similar trips as **neighboring trips** (or simply **neighbors**) of query trip $\mathbf{q}$. The naive solution to travel time estimation is to find neighboring trips with exact origin/destination and trip starting time. However, even with millions of trips due to the distribution skewness, this naive solution faces with serious sparsity issue. Therefore, we have to loose the condition for neighboring trips by considering trips with varying starting times, and thus the aggregating is not trivial because of varying traffic conditions.

We call trip $\mathbf{p}_i$ a neighbor of trip $\mathbf{q}$ if the origin (and destination) of $\mathbf{p}_i$ are spatially close to the origin (and destination) of $\mathbf{q}$. Thus, the set of neighbors of $\mathbf{q}$ is defined as:

$$\mathcal{N}(\mathbf{q}) = \{\mathbf{p}_i \in \mathcal{D} | dist(o_i, o_q) \leq \tau \text{ and } dist(d_i, d_q) \leq \tau\}, \tag{1}$$

where $dist()$ is a distance measure of two given points. For simplicity we use the Euclidean distance in our paper, but it is better to use the road network distance.

With the definition of neighbors, a baseline approach is to take the average travel time of these trips as the estimation:

$$\widehat{t}_q = \frac{1}{|\mathcal{N}(\mathbf{q})|} \sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{q})} t_i. \tag{2}$$

An alternative definition of neighbors is to weight neighboring trips based on the origin distance (i.e., $dist(o_i, o_q)$) and destination distance (i.e., $dist(d_i, d_q)$). However, such definition will make the computation more expensive since we typically need to consider more neighbors and also does not lead to a better performance (refer to Section 5.3.4).

With the model in Eq. (2), there is a major issue with the baseline approach: it does not model the dynamic traffic conditions across different time. To remedy this issue, for each neighboring trip $\mathbf{p}_i$ of $\mathbf{q}$ we define the *scaling factor* w_i calculated from the speed reference, so that $w_i t_i \approx t_q$. Thus, our estimation model becomes

$$\widehat{t}_q = \frac{1}{|\mathcal{N}(\mathbf{q})|} \sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{q})} w_i t_i. \tag{3}$$

In the following sections, we will discuss how to obtain scaling factor w_i by using the temporal regularity (Section 4.1), adjusting with respect to irregular changes (Section 4.2), and further considering the geographical differences (Section 4.3).

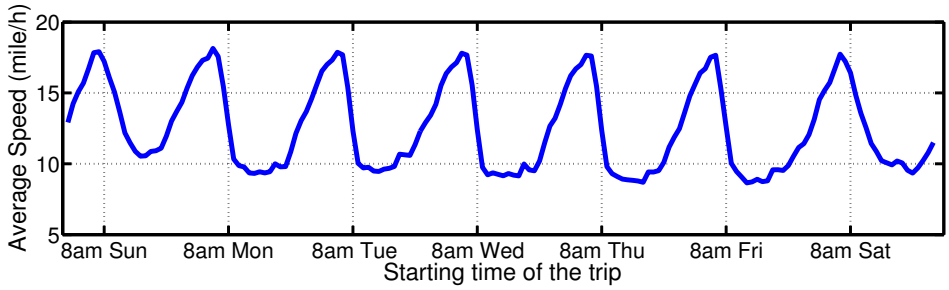
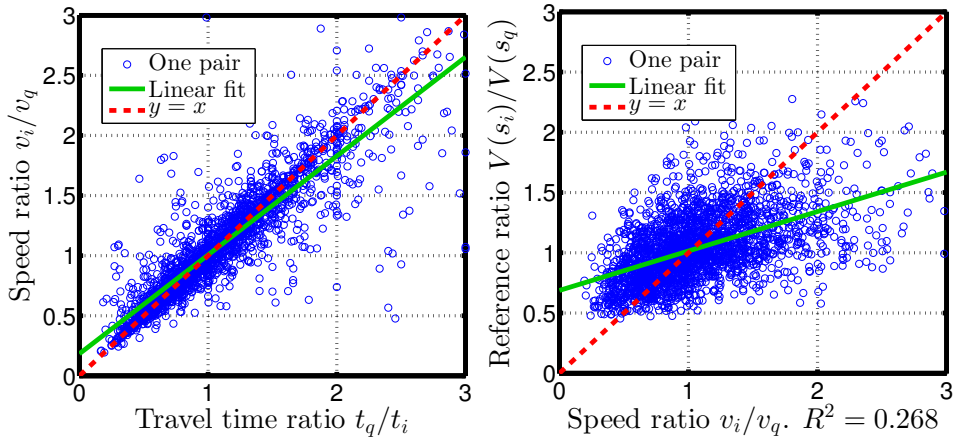


Fig. 2. Average travel speed w.r.t. trip starting time.



(a) $\frac{t_q}{t_i} \approx \frac{v_i}{v_q}$ (b) $\frac{v_i}{v_q} \approx \frac{V(s_i)}{V(s_q)}$

Fig. 3. The validity of our assumptions. We randomly sample a set of neighboring trip pairs, and calculate the different ratios for each pair. We plot each trip pair as a blue point. The solid green line is the linear regression fit of all the points. The dotted red line is $y = x$. In (a), we can see that fitted line has a slope approximating 1, which verifies our assumption $\forall \mathbf{p}_i \in \mathcal{N}(q), l_i \simeq l_q$. Similarly, in (b), we conclude that ratio of speed reference approximates the ratio of actual speed.

4 CAPTURING THE TEMPORAL DYNAMICS OF TRAFFIC CONDITIONS

As we discussed earlier, it is not appropriate to simply take the average of all the neighbors of $\mathbf{q}$ because of traffic conditions vary at different times. Figure 2 shows the average speed of all NYC taxi trips at different times in a week. Apparently, the average speed is much faster at the midnight compared with the speed during the peak hours. Thus, if we wish to estimate the travel time of a trip $\mathbf{q}$ at 2 a.m. using a neighboring trip $\mathbf{p}$ at 5 p.m., we should proportionally decrease the travel time of $\mathbf{p}$, because a 2 a.m. trip is often much faster than a 5 p.m. trip.

Now the question is, how can we derive a *temporal scaling reference* to correspondingly adjust travel time on the neighboring trips. We first define the **scaling factor** of a neighboring trip $\mathbf{p}_i$ on query trip $\mathbf{q}$ as:

$$w_i = \frac{t_q}{t_i}. \tag{4}$$

One way to estimate w_i is using the speed of $\mathbf{p}_i$ and $\mathbf{q}$. Let v_i and v_q be the speed of trip $\mathbf{p}_i$ and $\mathbf{q}$. Since we pick a small τ to extract neighboring trips of $\mathbf{q}$, it is safe to assume that $\forall \mathbf{p}_i \in \mathcal{N}(q)$, $l_i \simeq l_\mathbf{q}$. In Figure 3(a), on a sample set of neighboring trip pairs, we plot the inverse speed ratio against the travel time ratio. The solid line shows the actual relation between $\frac{t_q}{t_i}$ and $\frac{v_i}{v_q}$, which is very close to the dotted line $y = x$, which means that two ratios are approximately equivalent. With this assumption, we have:

$$w_i = \frac{t_q}{t_i} = \frac{l_q/v_q}{l_i/v_i} \approx \frac{v_i}{v_q}.$$

However, v_q is unknown, so we need to estimate $\frac{v_i}{v_q}$. Since the average speed of all trips are stable and readily available, we try to build a bridge from the actual speed ratio to the corresponding average speed ratio for any given two trips. One solution is to assume the ratio between v_q and v_i approximately equals to the ratio between the average speed of all trips at s_q and s_i. Formally, let $V(s)$ denote the average speed of all trips at timestamp s, we have an approximation of w_i as

$$w_i \approx \frac{v_i}{v_q} \approx \frac{V(s_i)}{V(s_q)}. \tag{5}$$

This assumption is validated in Figure 3(b). For the sample set of neighboring trip pairs, the average speed ratio is plotted against actual speed ratio. Since the points are approximately distributed along the line $y = x$, we conclude that average speed ratio is a feasible approximation of the scaling factor. We notice that the fitted line has a slope less than 1, which is mainly due to the anomaly in individual trips. Specifically, the speed of individual trips v has a higher variance and some individual trip pairs have an extreme large ratio. On the other hand, the average speed $V(s)$ has a smaller variance and the ratio is confined to a more reasonable range $[0.5, 2]$.

Next, we show the effectiveness of the scaling factor calculated from Eq. (5). In Figure 4, we present two specific trips to demonstrate the intuition of how the scaling factor works. For each trip, we retrieve its historical neighboring trips, and then plot the actual travel time in circle together with the scaled travel time in triangle. From the Figure 4(a), we can see that the scaling factor helps reduce the variance in the travel time of neighboring trips. In Figure 4(b), trip $\mathbf{q}$ happened in rush hour, which takes longer time than most of its neighboring trips. This gap is successfully filled in by the scaling factor.

Considering such scaling factors using speed as the reference, we can estimate the travel time of $\mathbf{q}$ using the neighboring trips as follows:

$$\widehat{t}_q = \frac{1}{|\mathcal{N}(\mathbf{q})|} \sum_{\mathbf{p}_i \in \mathcal{N}(\mathbf{q})} t_i \cdot \frac{V(s_i)}{V(s_q)}. \tag{6}$$

We show the effectiveness of this predictor in Figure 5. Each point in the figure is a target trip. The prediction is plotted against the actual trip travel time. We can see that the prediction is close to the actual value. This indirectly implies the validity of assumptions we made previously.

In order to compute the average speed $V(s_i)$, we need to collect all the trips in $\mathcal{D}$ which start at time s_i. However, for the query trip $\mathbf{q}$, the starting time s_q may be the current time or some time in the future. Therefore, no trips in $\mathcal{D}$ have the same starting time as $\mathbf{q}$. In the following, we discuss two approaches to predict $V(s_q)$ using the available data.

4.1 Relative Speed Reference

In this section, we assume that $V(s)$ exhibits a regular daily or weekly pattern. We fold the time into a relative time window $\mathcal{T}_{rela} = \{1, 2, \cdots, T\}$, where T is the assumed periodicity. For example,

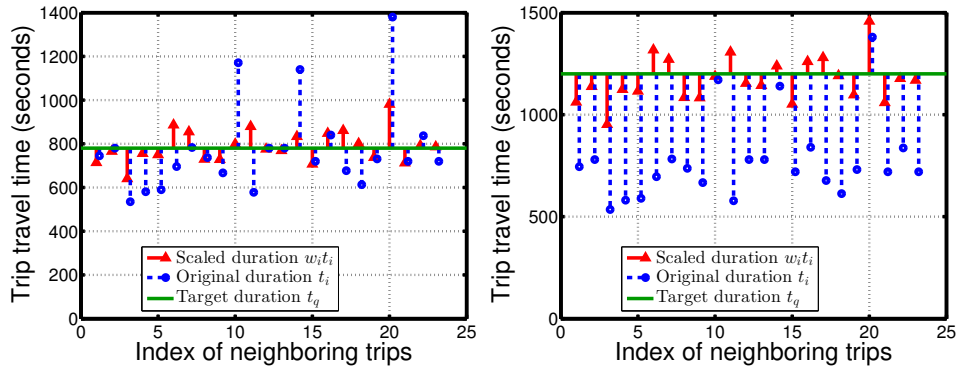


Fig. 4. We pick two particular trips to demonstrate the effectiveness of the scaling factor. The horizontal green line shows the travel time of target trip $\mathbf{q}$. The actual travel time t_i of neighboring trips is plotted as blue circle, while the scaled travel time $w_i t_i$ is plotted as red triangle. The index of neighboring trips refers to the ID of neighboring trip. In (a), the actual travel times of historical trips have very large variance, because they are from different time slots with distinct traffic conditions. The scaling factor successfully reduces the variance in the actual travel time t_i. In (b), the target trip $\mathbf{q}$ occurred in rush hour, and thus t_q is much longer than most of its neighboring trips. The scaling factor successfully fill in the gap between historical trips and $\mathbf{q}$.

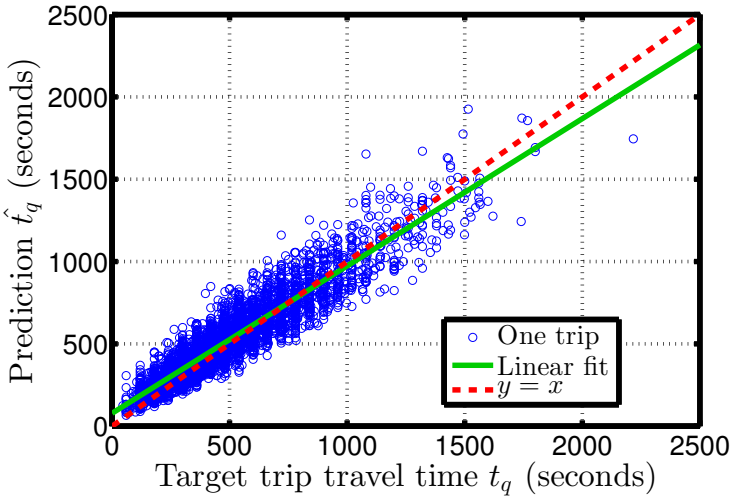


Fig. 5. Estimated travel time against actual travel time. Each point is a trip, where the estimation $\hat{t_q}$ is plotted against t_q. The green line is the linear regression fit of the points, which is closer to the red line $y = x$. This means the prediction is close to the actual value.

using a weekly pattern with 1 hour as the basic unit, we have $T = 7 \times 24 = 168$. Using this relative time window, we represent the average speed of the k-th time slot as $V_k, \forall k \in \mathcal{T}_{rela}$. We call $\{V_k | k \in \mathcal{T}_{rela}\}$ *relative reference*.

We use k_i to denote the time slot to which s_i belongs. As a result, we can write $V(s_i) = V_{k_i}$. To compute V_{k_i}, we collect all the trips in $\mathcal{D}$ which fall into the same time slot as $\mathbf{p}_i$ and denote the

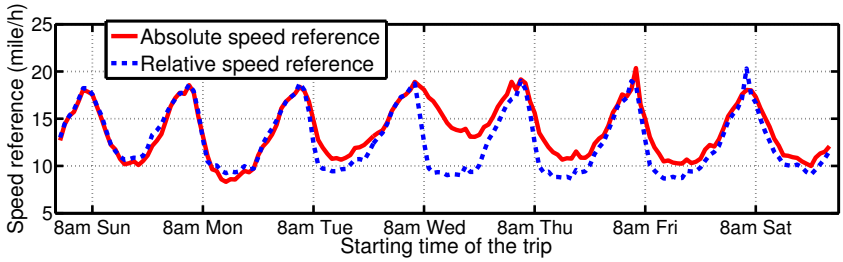


Fig. 6. Comparison between the absolute speed reference and the relative speed reference in the Christmas week (Dec 22, 2015 – Dec 28, 2013). We can see the traffic condition is much better during Wednesday daytime than usual, because people are celebrating Christmas.

set as $S(\mathbf{p}_i)$. Then, we have

$$V_{k_i} = \frac{1}{|S(\mathbf{p}_i)|} \sum_{\mathbf{p}_j \in S(\mathbf{p}_i)} \frac{l_j}{t_j} \tag{7}$$

In Figure 2, we present the weekly relative speed reference on all trips. We can see all the weekdays share similar patterns the rush hour started from 8:00 in the morning. Meanwhile, during the weekends, the traffic is not as much as usual during 8:00 in the morning.

The relative speed reference mainly has the following two advantages. First, the relative speed reference is able to alleviate the data sparsity issue. By folding the data into a relative window, we will have more trips to estimate an average speed with a higher confidence. Second, the computation overhead of relative speed reference is small, and we could do it offline.

4.2 Absolute Speed Reference

In the previous section, the relative speed reference assumes the speed follows daily or weekly regularity. However, in real scenario, there are always irregularities in the traffic condition. For example, during national holidays the traffic condition will significantly deviate from the usual days. In Figure 6, we show the actual average traffic speed during the Christmas week (Dec 22, 2013 – Dec 28, 2013). Compared with the relative speed reference, we can see that on Dec 25th, 2013, the traffic condition is better than usual during the day, since people are celebrating Christmas. Therefore, assuming we have enough data, it would be more accurate if we could directly infer the average speed at any time slot t from the historical data.

In this section, we propose an alternative approach to directly capture the traffic condition at different time slots. We extend our relative reference from Section 4.1 to an *absolute speed reference*. To this end, we partition the original timeline into time slots based on a certain time interval (i.e., 1 hour). All historical trips are mapped to the absolute time slots $\mathcal{T}_{abs} = \{1, 2, 3, \cdots\}$ accordingly, and the average speed $\{V_k | k \in \mathcal{T}_{abs}\}$ are calculated as the absolute speed reference.

The challenge in absolute speed reference is that for a given query trip with starting time s_q in the near future (e.g. next hour), we need to estimate the speed reference $V(s_q)$. We estimate $V(s_q)$ by taking into account factors such as the average speed of previous hours, seasonality, and random noise. Formally, given the time series of average speed: $\{V_1, V_2, \ldots, V_M\}$, our goal is to compute V_{M+1} as follows:

$$V_{M+1} = f(V_1, \ldots, V_M). \tag{8}$$

In order to predict current speed reference based on historical speed time series, we adopt the auto-regressive integrated moving average (ARIMA) model [10] for time series forecasting. We will discuss the ARIMA model and seasonal ARIMA model in the following section.

4.2.1 Overview of the ARIMA Model. In statistical analysis of time series, ARIMA is a popular tool for understanding and predicting future values in a time series. Mathematically, for any time series $\{X_t\}$, let L be the *lag operator*:

$$LX_t = X_{t-1}, \forall t > 1. \tag{9}$$

Then, the ARIMA(p, d, q) model is given by

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d X_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\epsilon_t, \tag{10}$$

where parameters p, d and q are non-negative integers that refer to the order of the auto-regressive, integrated, and moving average parts of the model, respectively. In addition, ϵ_t is a white noise process.

In practice, the autocorrelation function (ACF) and partial autocorrelation function (APCF) is frequently used to estimate the order parameters (p, d, q) from a time series of observations $\{X_1, X_2, \ldots, X_M\}$. Then, the coefficients $\{\phi_i\}_{i=1}^{p}$ and $\{\theta_i\}_{i=1}^{q}$ of the ARIMA(p, d, q) can be learned using standard statistical methods such as the least squares.

4.2.2 Incorporating the Seasonality. In our problem, the average speed V_t exhibits a strong weekly pattern. Thus, instead of directly applying the ARIMA model to $\{V_t\}$, we first compute the sequence of seasonal difference $\{Y_t\}$:

$$Y_t = V_t - V_{t-T}, \tag{11}$$

where T is the period (e.g., one week). Then, we apply the ARIMA model to $\{Y_t\}$:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d Y_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\epsilon_t. \tag{12}$$

Note that our ARIMA model with the seasonal difference is a special case of the more general class of Seasonal ARIMA (SARIMA) model for time series analysis. We refer interested readers to [10] for detailed discussion about the model.

Suppose we use first order difference of Y_t, namely $d = 1$ and $(1 - L)^d Y_t = Y_t - Y_{t-1}$. Then we have

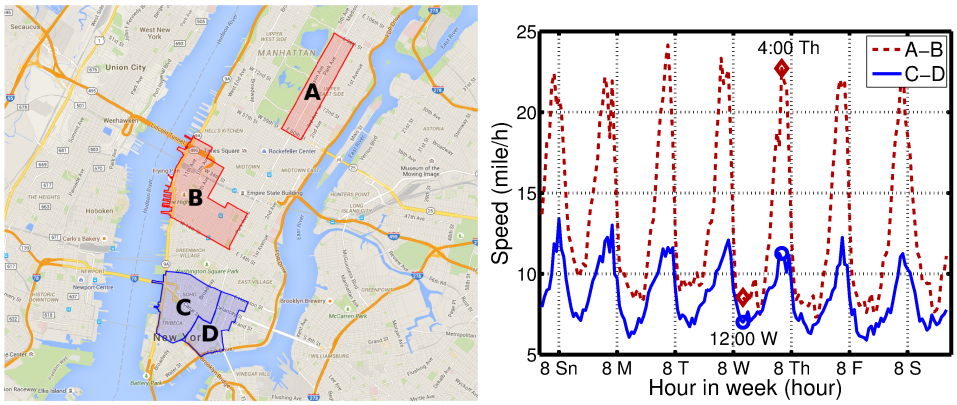
$$Y_t = Y_{t-1} + \sum_{i=1}^{p} \phi_i L^i (Y_t - Y_{t-1}) + \sum_{i=1}^{q} \theta_i L^i \epsilon_t + \epsilon_t \tag{13}$$

Since the last term ϵ_t in Eq. (13) is white noise, whose value is unknown but the expectation $E(\epsilon_t) = 0$, we have estimator $\hat{Y}_t = Y_t - \epsilon_t$. Together with Eq. (11), we have

$$\hat{V}_t = \hat{Y}_t + V_{t-T} \tag{14}$$

4.3 The Effect of Geographic Regions

So far, we have assumed that the traffic condition follows the same temporal pattern across all geographic locations. However, in practice, trips within a large geographic area (e.g., New York City) may have different traffic patterns, depending on the spatial locations. For example, in Figure 7 we show the speed references of two different pairs of regions. The speed reference in region pairs (A, B) has a larger variation than (C, D). In particular, for (A, B) pair, the average speed is 22.7 mph at 4:00 a.m. on Thursday and 8.5 mph at 12:00 p.m. on Wednesday. But for (C, D) pair, the average speed is only 11.2 mph and 7.0 mph at these two corresponding times. The reason could be that A is a residential area, whereas B is a business district. Therefore, the traffic pattern between A and B exhibits a very strong daily peak-hour pattern. On the other hand, regions C and D are popular tourist areas, the speeds are constantly slower compared with that of (A, B).



(a) Manhattan (b) Speed reference

Fig. 7. Different region pairs have different traffic patterns.

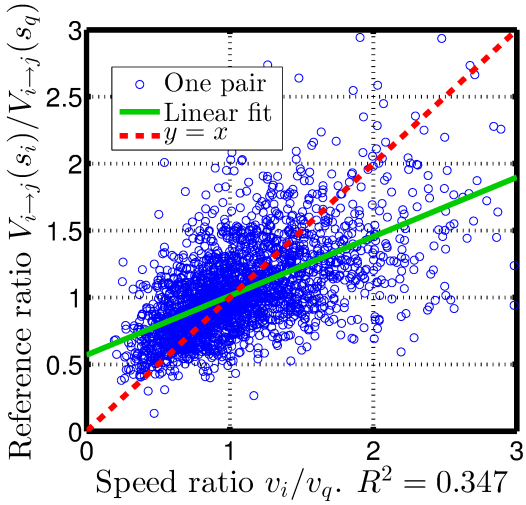


Fig. 8. Refining speed reference with regions improves the performance. We use the same set of trip pairs sampled in Figure 3(b) to validate $\frac{V_{i\to j}(s_i)}{V_{i\to j}(s_q)} \approx \frac{v_i}{v_q}$. Compared with Figure 3(b), the slope of linear regression fit is closer to 1 and the R^2 is larger, which means the region-based speed reference approximates the actual speed ratio better.

The real case above suggests that refining the speed references for different spatial regions could help us further improve the travel time estimation. Therefore, we propose to divide the map into a set of K neighborhoods $\{R_1, R_2, \ldots, R_K\}$. For each pair of neighborhoods (R_i, R_j), $1 \le i, j \le K$, we use $V_{i\to j}(s)$ to denote the average speed of all trips from R_i to R_j at starting time s. Let $\mathcal{D}_{i\to j}$ be the subset of trips in $\mathcal{D}$ whose origin and destination fall in R_i and R_j, respectively. Then, $V_{i\to j}(s)$ can be estimated using $\mathcal{D}_{i\to j}$ in the same way as described in Sections 4.1 and 4.2.

We use Figure 8 to show the effectiveness of refining the speed reference for each region pairs. In Figure 8, we plot the refined speed reference ratio against the actual speed ratio of a pair of trips

using the same set of sampled trip pairs in Figure 3(b). Comparing with the Figure 3(b), we see that the linear regression fit has a slope closer to 1, which means the two ratios are closer, and the R^2 is larger, which means the line fits better.

4.4 Time Complexity Analysis

Our approach has three steps: 1) mapping training trips into grids; 2) extracting neighbors of a given OD pair; and 3) estimating travel time based on the neighbors.

Step 1. In order to quickly retrieve the neighboring trips, we employ a raster partition of the city (e.g., 50 meters by 50 meters grid in our experiment), and preparing N training trips takes $O(N)$ time. This step can be preprocessed offline.

Step 2. Given a testing trip $\mathbf{q}$, we find its corresponding origin gird in $O(1)$ time. Then, retrieving all the neighboring trips in the same grid would take $O(N/h)$ time if the trips are uniformly distributed, where h is the total number of grids. In practice, however, the number of trips in each grid follows a long tail distribution. Therefore, the worst case time complexity of retrieving neighboring trips would be $O(\alpha \cdot N)$, where $\alpha \cdot N$ is the number of trips in the most dense grid ($\alpha = 0.01$ in NYC taxi data).

Step 3. After retrieving the neighboring trips, the time complexity of calculating travel time of $\mathbf{q}$ is $O(|\mathcal{N}(\mathbf{q})|)$, where $|\mathcal{N}(\mathbf{q})|$ is the number of neighbors of trip $\mathbf{q}$. The speed references can be computed offline. The relative speed reference takes $O(N)$ time, and ARIMA model can be trained in $O(M^2)$ time, where M is the number of time slots. During the online estimation, looking-up relative speed reference takes $O(1)$ time, whereas using ARIMA to predict the current speed reference takes $O((p+q) \cdot d)$ time, where p, q, and d are the parameters learned during model training ($p = 2$, $q = 0$, and $d = 1$ in our experiment).

Therefore, the time complexity of online computation is $O(\alpha \cdot N)$ in the worst case. In order to serve a large amount of batch queries, we can use multithreading to further boost the estimation time.

5 EXPERIMENT

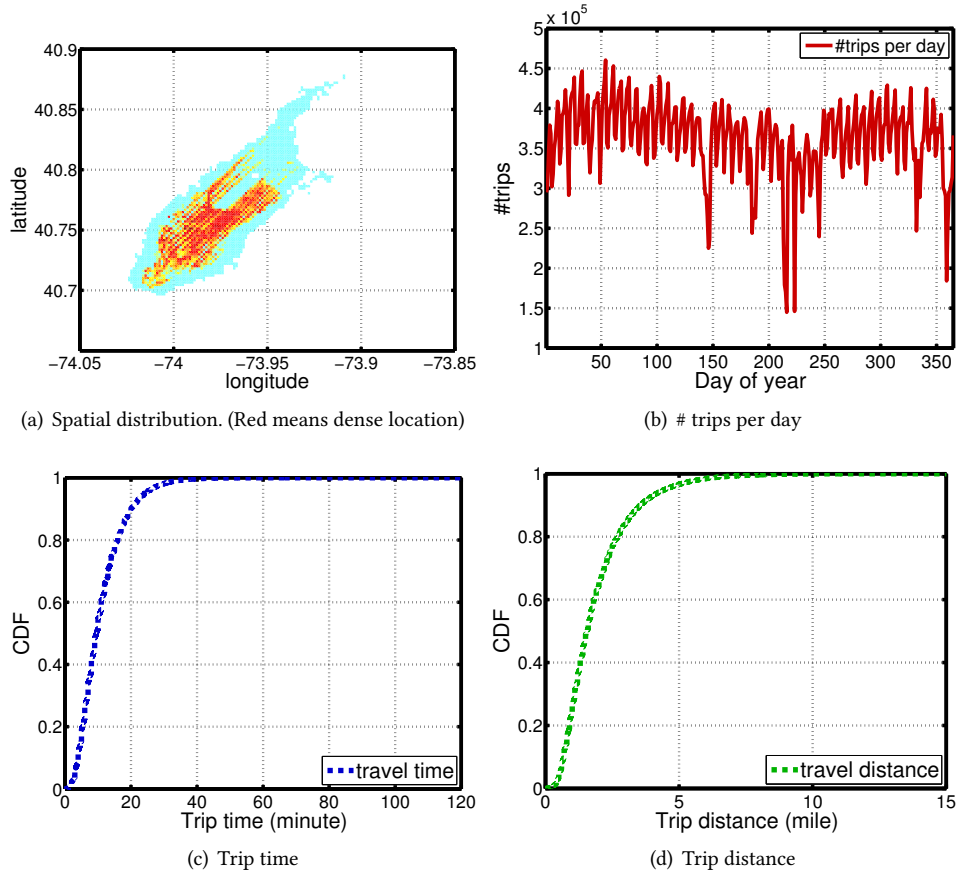
In this section, we present a comprehensive experimental study on two real datasets. All the experiments are conducted on a 8-core 3.4 GHz Intel Core i7 system with 16 GB memory. We parallelize the computation, which will be discussed in Section 5.6 in detail.

5.1 Dataset

We conduct experiments on datasets from two different countries to show the generality of our approach.

5.1.1 NYC Taxi. A large-scale New York City taxi dataset has been made public online [29]. The dataset contains more than 160 million taxi trips per year. Each trip contains information about pick-up location and time, drop-off location and time, trip distance, fare amount, etc. We use the subset of trips within the borough of Manhattan (the boundary is obtained from wikimapia.org) from 2013 to 2015, which has roughly over 127 million trips per year. On average, we have 349, 410 trips per day.

Figure 9(a) shows the distribution of GPS points of the pick-up and drop-off locations. Figure 9(b) shows the number of trips per day over the year 2013. Figure 9(c) and Figure 9(d) show the empirical CDF plot of the trip distance and trip time. About 56% trips have trip time less than 10 minutes, and about 99% trips have trip time less than 30 minutes. The mean and median trip time are 636 (second) and 546 (second). The mean and median trip distance are 1.935 (mile) and 1.6 (mile). We



(a) Spatial distribution. (Red means dense location)

(b) # trips per day

(c) Trip time

(d) Trip distance

Fig. 9. NYC data statistics. See text for explanations.

noticed that most of trips in our dataset is short, however to the best of our knowledge the NYC taxi dataset is the largest publicly accessible dataset.

5.1.2 Shanghai Taxi. In order to compare with the existing route-based methods, we use the Shanghai taxi dataset with the trajectories of 2, 600 taxis during two months in 2006 [25]. A GPS record has following fields: vehicle ID, speed, longitude, latitude, occupancy, and timestamp. In total we have over 300 million GPS records. We extract the geographical information of Shanghai road networks from OpenStreetMap.

To retrieve taxi trips in this dataset, we rely on the occupancy bit of GPS records. This occupancy bit is 1 if there are passengers on board, and 0 otherwise. For each taxi, we define a trip as consecutive GPS records with occupancy equal to 1. We get **5, 815, 470** trips after processing the raw data. The distribution of trip travel time is similar to that of NYC taxi data in Figure 9(c). Specifically, about half of the trips have travel time less than 10 minutes.

5.2 Evaluation Protocol

Methods for evaluation. We systematically compare the following methods:

- Linear regression (LR). We train a linear regression model with travel time as a function of the L1 distance between pick-up and drop-off locations. This simple linear regression serves as a baseline for comparison.
- Neighbor average (AVG). This method simply takes the average of travel times of all neighboring trips as the estimation.
- Temporally weighted neighbors (TEMP). This is our proposed method using temporal speed reference to assign weights on neighboring trips. We name the method using relative-time speed reference as $\texttt{TEMP}_{\texttt{rel}}$ (Section 4.1). And we call the method using ARIMA model to predict absolute temporal reference as $\texttt{TEMP}_{\texttt{abs}}$ (Section 4.2).
- Temporal speed reference by region (TEMP+R). This is our improved method based on TEMP by considering the temporal reference for different region pairs (Section 4.3).
- Segment-based estimator (SEGMENT). This method estimates the travel time of each road segment individually, and then aggregate them to get the estimation of a complete trip (a baseline method used in [27]).
- Subpath-based estimator (SUBPATH). One drawback of SEGMENT is that the transition time at intersections cannot be captured. Therefore, Wang et al. [27] propose to concatenate subpaths to estimate the target route, where each sub-path is consisted of multiple road segments. For each sub-path, SUBPATH estimates its travel time by searching all the trips that contain this sub-path.
- Online map service (BING and BAIDU). We also compare our methods with online map services. We use Bing Maps [16] for NYC taxi dataset and Baidu Maps [1] for Shanghai dataset. We use Bing Maps instead of Google Maps because Bing Maps API allows query with current traffic for free whereas Google does not provide that. To consider traffic, we send queries to Bing Maps at the same time of the same day (in a weekly window) as the starting time of the testing trip. Due to national security concerns, the mapping of raw GPS data is restricted in China [18]. So we use Baidu Maps instead of Bing Maps for Shanghai taxi dataset.

Evaluation metrics. Similar to [27], we use mean absolute error (MAE) and mean relative error (MRE) to evaluate the travel time estimation methods:

$$MAE = \frac{\sum_i |y_i - \hat{y}_i|}{n}, MRE = \frac{\sum_i |y_i - \hat{y}_i|}{\sum_i y_i},$$

where $\hat{y}_i$ is the travel time estimation of test trip i and y_i is the ground truth. Since there are anomalous trips, we also use the median absolute error (MedAE) and median relative error (MedRE) to evaluate the methods:

$$MedAE = median(|y_i - \hat{y}_i|), MedRE = median\left(\frac{|y_i - \hat{y}_i|}{y_i}\right),$$

where *median* returns the median value of a vector.

5.3 Performance on NYC Data

5.3.1 Overall Performance on NYC Data. In our experiment, we evaluate our methods independently within each of the three years. The trips from the first 11 months (i.e., January to November) are used as training and the ones in the last month (i.e., December) are used as testing. As for the BING method, due to the limited quota, we sample $260k$ trips in December as testing of each year. Bing Map API does not support travel time query of a past trip. Therefore we assume that the traffic condition has strong weekly periodicity, and we use the same relative time within a week to query the Bing prediction for all the testing trips. The Bing predictions of all testing trips are queried in the third week of August 2017. The overall accuracy comparisons are shown in Table 1. Since NYC

data only have endpoints of trips, we cannot compare our method with route-based methods. We will compare with these methods using Shanghai data in Section 5.4.

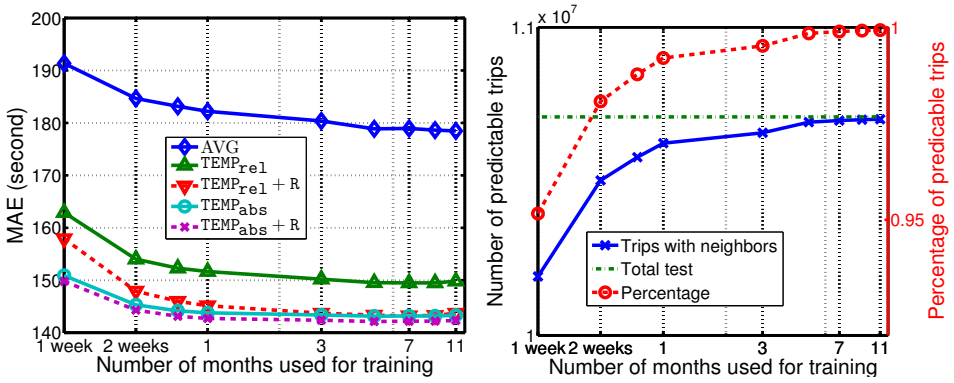
Table 1. Overall Performance on NYC Data

Year	Method	MAE (s)	MRE	MedAE (s)	MedRE
2013	LR	194.60	0.2949	164.82	0.3017
	AVG	178.46	0.2704	120.83	0.2345
	TEMP_{rel}	149.81	0.2270	97.36	0.1907
	TEMP_{abs}	143.31	0.2171	98.78	0.1890
	TEMP_{rel} + R	143.72	0.2178	**92.07**	**0.1805**
	TEMP_{abs} + R	**142.33**	**0.2157**	98.05	0.1874
2014	LR	291.76	0.3552	224.574	0.3338
	AVG	237.09	0.2887	142.17	0.2397
	TEMP_{rel}	204.43	0.2489	117.31	0.1993
	TEMP_{abs}	195.05	0.2375	118.90	0.1975
	TEMP_{rel} + R	194.81	0.2371	**110.179**	**0.1867**
	TEMP_{abs} + R	**192.69**	**0.2346**	118.37	0.1956
2015	LR	300.67	0.3567	227.87	0.3348
	AVG	244.63	0.2902	147.10	0.2432
	TEMP_{rel}	207.64	0.2463	120.06	0.2000
	TEMP_{abs}	196.62	0.2332	116.78	0.1928
	TEMP_{rel} + R	**196.54**	**0.2331**	**111.91**	**0.1859**
	TEMP_{abs} + R	196.61	0.2332	116.78	0.1928
Comparison with Bing Maps (260K testing trips per year)					
2013	BING	233.81	0.3642	153.00	0.3180
	BING(traffic)	193.53	0.3015	140.00	0.2678
	TEMP_{abs} + R	**135.36**	**0.2108**	**94.94**	**0.1839**
2014	BING	343.69	0.4118	205.00	0.3615
	BING(traffic)	240.41	0.2915	153.00	0.2517
	TEMP_{abs} + R	**192.75**	**0.2337**	**117.59**	**0.1947**
2015	BING	381.92	0.4410	231.00	0.3866
	BING(traffic)	250.49	0.2892	152.00	0.2453
	TEMP_{abs} + R	**204.12**	**0.2357**	**119.93**	**0.1938**

We first observe that our method is better than linear regression (LR) baseline. This is expected because LR is a simple baseline which does not consider the origin and destination locations.

Considering temporal variations of traffic condition improves the estimation performance. All the TEMP methods have significantly lower errors compared with AVG method. The improvement of TEMP_{abs} over AVG is about 35 - 50 seconds in MAE. In other words, the MAE is decreased by nearly 20% by considering the temporal factor. Furthermore, using the absolute speed reference (TEMP_{abs}) is better than using the relative (i.e., weekly) speed reference (TEMP_{rel}). This is because the traffic condition does not strictly follow the weekly pattern, but has some irregular days such as holidays.

By adding the region factor, the performance is further improved. This means the traffic patterns between regions are actually different. However, we observe that the improvement of TEMP_{abs} + R over TEMP_{abs} is not as significant as the improvement of TEMP_{rel} + R over TEMP_{rel}. This is due to data sparsity when we compute the absolute time reference for each region pair. If two regions



(a) MAE w.r.t. training set size (b) Coverage of testing set

Fig. 10. Performance with regard to the training set size on 2013 data.

have little traffic flow, the absolute time reference might not be accurate. In this case, using the relative time reference between regions make the data more dense and produces better results.

In Table 1, the MAE of $\texttt{TEMP}_{\texttt{abs}}$ + R is significantly lower than BING by more than 100 seconds. BING underestimates the travel times without considering traffic, where 81.37% testing trips are underestimated. When considering traffic, BING(traffic) is able to better predict the travel time. Notice that this result is different from our conference paper [24], where BING(traffic) has a higher prediction error than BING. The reason is that the previous Bing results are queried in October 2015. In the this paper, we queried the Bing Map predictions for testing trips in the week of August 2017. There might be an improvement on the Bing Map service.

5.3.2 Performance w.r.t. the Size of Historical Data. We expect that, by using more historical data (i.e., training data), the estimation will be more accurate. To verify this, we study how performance changes w.r.t. the size of training data. We choose different time durations (from 1 week, 2 weeks, to 11 months) before December as the training data. Figure 10(a) shows the accuracy w.r.t. the size of the training data. The result meets with our expectation that MAE drops when using more historical data. But the gain of using more data is not obvious when using more than 1 month data. This indicates that using 1-month training data is enough to achieve a stable performance in our experimental setting. Such indication will save running time in real applications as we do not need to search for neighboring trips more than 1 month ago.

In addition, using more historical data, we are able to cover more testing trips. Because a testing trip should have at least one neighboring trip in order to be estimated. Figure 10(b) shows the coverage of testing data by using more historical data. With 1-week data, we can estimate 95.16% testing trips; with 1-month data, we can cover 99.20% testing trips.

5.3.3 Performance w.r.t. Trip Features. Next, we study the performance of all methods with respect to various trip features.

Performance w.r.t. trip time. In Figure 11(a) and 11(b), we plot the MAE and MRE with respect to the trip time. As expected, the longer-time trips have higher MAEs. It is interesting to observe that the MREs are high for both short-time trips and long-time trips. Because the short-time trips (less than 500 seconds) are more sensitive to dynamic conditions of the trips, such as traffic light. On the other hand, the long-time trips usually have less neighbors, which leads to higher MREs.

Performance w.r.t. trip distance. The MAE and MRE against trip distance are shown in Figure 11(c) and 11(d). Overall, we have consistent observations as Figure 11(a) and Figure 11(b). However, we notice that the error of TEMP_{rel} + R increases faster than other methods. TEMP_{rel} + R becomes even worse than AVG method when the trip distance is longer than 8 miles. This is because the longer-distance trips have fewer neighboring trips. However, we do not observe the same phenomenon in Figure 11(a) and Figure 11(b). After further inspection, we find that long-*distance* trips have even less neighboring trips than long-*time* trips. The reason is that long-distance trips usually have long travel time; however, long-time trips may not have long trip distance (e.g., the long travel time may be due to traffic).

Performance w.r.t. number of neighboring trips. The error against number of neighbors per trip is shown in Figure 11(e) and Figure 11(f). MAEs decrease for the trips with more neighbors. However, MREs increase as the number of neighbors increases. The reason is that the short trips usually have more neighbors and short trips have higher relative errors, as shown in Figure 11(b) and Figure 11(d).

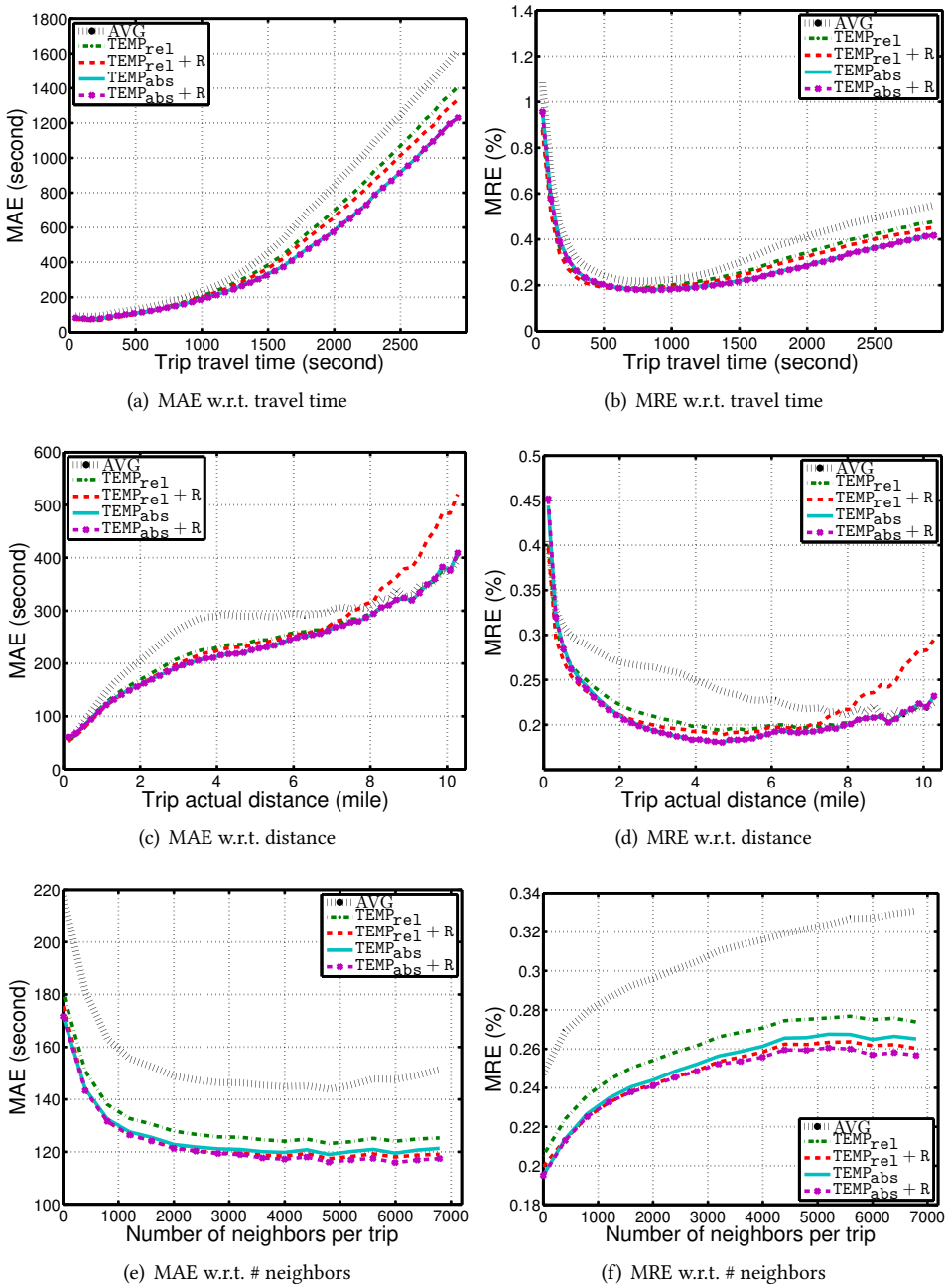
5.3.4 An Alternative Definition of Neighbors. We further study the performance by defining *soft* neighbors. We give neighboring trips spatial weights based on the sum of corresponding end-points' distance, and evaluate the performance on 500k trips. The MAE of TEMP_{abs} + R is 107.705s on this testing set. Adding spatial weights on the neighbors degenerates the performance to 112.392s. It suggests that it is non trivial to assign weights on neighbors based on the distances of end-points. It will require more careful consideration of this factor.

5.4 Performance on Shanghai Taxi Data

In this section, we conduct experiments on Shanghai taxi data. This dataset provides the trajectories of the taxi trips so we can compare our method with SEGMENT and SUBPATH methods. Both SEGMENT and SUBPATH methods use the travel time on individual road segments or subpaths to estimate the travel time for a query trip. Due to data sparsity, we cannot obtain travel time for every road segments. [27] propose a tensor decomposition method to estimate the missing values. In our experiment, to avoid the missing value issue, we only select the testing trips that have values on every segments of the trip. However, by doing such selection, the testing trips are biased towards shorter trips. Because the shorter the trip is, the less likely the trip has missing values on the segment. To alleviate this bias issue, we further sampled the biased trip dataset to make the travel time distribution similar to the distribution of the original whole dataset. We randomly sample 2, 138 trips in total as testing, and the rest are used as training. We use the same training set and testing set for different methods.

Table 2. Overall Performance on Shanghai Taxi Data

Method	MAE (s)	MRE	MedAE (s)	MedRE
LR	130.710	0.6399	138.796	0.6173
BAIDU	111.484	0.5451	73.001	0.5001
SEGMENT	119.833	0.5866	84.704	0.4947
SUBPATH	113.566	0.5560	75.913	0.4820
AVG	94.202	0.4615	60.183	0.3739
TEMP_{rel}	**92.428**	**0.4527**	**55.317**	**0.3678**



(a) MAE w.r.t. travel time

(b) MRE w.r.t. travel time

(c) MAE w.r.t. distance

(d) MRE w.r.t. distance

(e) MAE w.r.t. # neighbors

(f) MRE w.r.t. # neighbors

Fig. 11. Performance with regard to the trip features.

5.4.1 Overall Performance on Shanghai Taxi Data. The comparison among different methods is shown in Table 2. Our neighbor-based method significantly outperform other methods. The simple method AVG is 17 seconds better than BAIDU in terms of MAE. By considering temporal factors, our

method $\texttt{TEMP}_{\texttt{rel}}$ further outperforms AVG method. Here, we do not report the results of $\texttt{TEMP}_{\texttt{abs}}$, because the two-month Shanghai data have several days' trip missing, but the ARIMA model requires complete time series. In addition, SUBPATH method outperforms SEGMENT method, which is consistent with previous work [27]. SEGMENT simply adds travel time of individual segments, whereas SUBPATH considers the transition time between segments by concatenating subpaths. However, SUBPATH is still 21 seconds worse than our method $\texttt{TEMP}_{\texttt{rel}}$ in terms of MAE. Such result is also expected. Because our method can be considered as a special case of SUBPATH method, where we use the whole paths from the training data to estimate the travel time for a testing trip. If we have enough number of whole paths, it is better to use the whole paths instead of subpaths.

Since the Shanghai taxi data are much more sparse than NYC data, we do not show the results of $\texttt{TEMP}_{\texttt{rel}}$ + R and $\texttt{TEMP}_{\texttt{abs}}$ + R on this dataset. Note that the main goal of this experiment is to show that $\texttt{TEMP}_{\texttt{rel}}$ outperforms the existing methods. With more data, our other approaches could potentially outperform the $\texttt{TEMP}_{\texttt{rel}}$ as well as shown in NYC data.

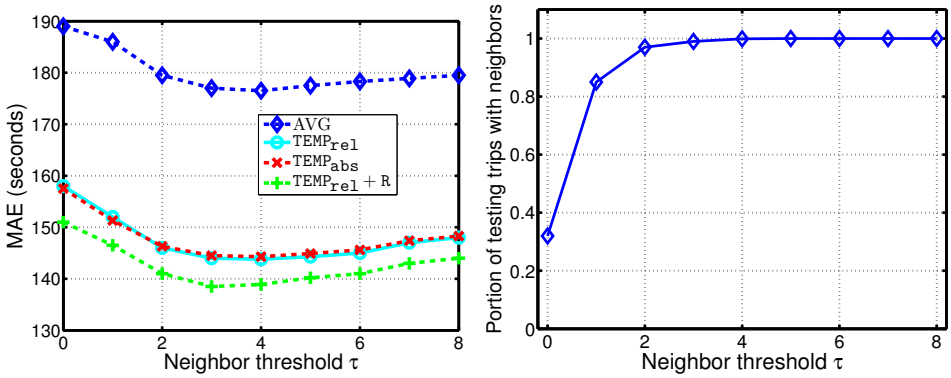
5.4.2 Applicability of Segment-based Method and Neighbor-based Method. Segment-based method requires every individual road segment of the testing trip has at least one historical data point to estimate the travel time. On the other hand, our neighbor-based method requires a testing trip has at least one neighboring trip. Among 435, 887 trips in Shanghai dataset, only 54, 530 trips have values on every road segment, but 217, 585 trips have at least one neighbor. Therefore, in this experimental setting, our method not only outperforms segment-based method in terms of accuracy, it is also more applicable to answer more queries (49.9%) compared with segment-based method (12.5%).

5.5 Parameter Sensitivity

In this section we study the parameter sensitivity of our proposed method. There is only one parameter in our proposed method, the distance threshold τ to define the neighbors. We use the first 11 months of 2013 from NYC dataset to study the performance w.r.t. parameter τ. For computational efficiency, we partition the map into small grids of 50 meters by 50 meters. The distance in Eq. (1) is now defined as the L1 distance between two grids. For example, if p_1 is a neighboring trip of q for $\tau = 0$, it means that the pick-up (and drop-off) location of p_1 is in the same grid as the pick-up (and drop-off) location of q. If $\tau = 1$, a neighboring trip has endpoints in the same or adjacent grids.

A larger τ will retrieve more neighboring trips for a testing trip and thus could give an estimation with a higher confidence. However, a larger τ also implies that the neighboring trips are less similar to the testing trip, which could introduce prediction errors. Therefore, it is crucial to identify the optimal τ that balance estimation confidence and neighbors' similarity. In Figure 12(a), we plot the empirical relationship between the MAE and τ. We can see that MAE is the lowest when τ is in the range of $[3, 6]$, striking a good balance between confidence and similarity. It is worthy to mention that our method is not sensitive to the different selection of τ.

We also want to point out that we need to have at least one neighboring trip in order to give an estimation of a testing trip. Obviously, the larger τ is, the more testing trips we can cover. Figure 12(b) plots the percentage of testing trips with at least one neighbor. The figure shows that if $\tau = 1$, there are only 32% trips are predictable. When $\tau = 3$, 99% trips are predictable. Considering the both aspects, we use $\tau = 3$ in our experiments, which considers two trips are neighboring if their starting and ending points are within 150 meters. Our method can also adapt to different cities by controlling neighborhood parameter τ.



(a) Performance (b) Coverage

Fig. 12. Estimation error and testing coverage w.r.t. neighbor threshold τ.

5.6 Running Time

Table 3 presents the running time comparison between TEMP_{abs} + R and SUBPATH. We use a multi-threading implementation of our method, since it is easy to parallel. We profile the running time of TEMP_{abs} + R on last two months' trips in NYC dataset, and SUBPATH on whole Shanghai dataset. Note that NYC dataset is bigger than Shanghai dataset and TEMP_{abs} + R is the most computationally expensive method, and also the most accurate method among our proposed methods. The pre-processing time for TEMP_{abs} + R consists of mapping trips to grids, calculating the speed reference, and learning the ARIMA parameters. The preparation time for SUBPATH includes indexing trajectory by road segments.

Table 3. Running time comparison

Method	Process	#trips	time	time/trip
Our method	Prep. training trip	11M	42s	0.004ms
	Learn ARIMA	11M	147s	0.013ms
	Find neighbor	100K	24s	0.24ms
	Estimate	10.5M	229s	0.021ms
SUBPATH	Prep. training trip	4.35M	547s	0.126ms
	Estimate	54K	2297s	42ms

In TEMP_{abs} + R, both training trip preparation and learning ARIMA can be done off-line or updated every hour with the new data. The online query part includes querying neighboring trips and estimation using neighbors. The neighbor finding is the most time consuming part. We get 1.09 ms/trip if using one thread and 0.24 ms/trip if using 8 threads. The estimation part only takes 0.02 ms with 8 threads. So the total online query time for our method is 1.505 ms/trip on average and it will be even faster (i.e., 0.26 ms/trip) if using 8 threads. The estimation of SUBPATH includes finding optimal concatenation of segments, which has $O(n^2 \cdot m)$ time complexity, where n is the number of road segments and m is number of trips going through each segment. It takes 42ms on average to estimate a testing trip. Our neighbor-based method is more efficient in answering travel time queries because we avoid computing the route and estimating the time for the subpaths.

6 CONCLUSION

This paper demonstrates that one can use large-scale trip data to estimate the travel time between an origin and a destination in a very efficient and effective way. Our proposed method retrieves all the neighboring historical trips with the similar origin and estimation locations and estimate the travel time using those neighboring trips. We further improve our method by considering the traffic conditions w.r.t. different temporal granularity and spatial dynamics. We conduct experiments on two large-scale real-world datasets and show that our method can greatly outperform the state-of-the-art methods and online map services.

In our method, the spatial region is an important factor which greatly improves the estimation accuracy. However, for the purpose of simplicity, we directly employ the administrative boundary to partition city into regions. One could improve this baseline partitioning by learning a better partitioning from the taxi data, because the goal of region partition should be finding neighborhood, within which temporal traffic pattern is similar. Deriving those neighborhoods with similar traffic patterns is actually non-trivial and worth further investigation.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

The work was supported in part by NSF awards #1054389, #1228669, #1544455, #1652525, #1618448, and #1702760. Zhenhui Li would like to acknowledge the support from Haile Family Early Career Professorship. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

REFERENCES

[1] Baidu 2016. Baidu Map: http://map.baidu.com.

[2] Paolo Cintia, Roberto Trasarti, Livia Almada Cruz, Camila F Costa, and Jose Antônio F de Macedo. 2013. A Gravity Model for Speed Estimation over Road Network. In *IEEE MDM*, Vol. 2. IEEE, 136–141. https://doi.org/10.1109/MDM.2013.83

[3] Corrado De Fabritiis, Roberto Ragona, and Gaetano Valenti. 2008. Traffic estimation and prediction based on real time floating car data. In *IEEE International Conference on Intelligent Transportation Systems*. 197–203.

[4] Brian C Dean. 1999. *Continuous-time dynamic shortest path algorithms*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[5] Hector Gonzalez, Jiawei Han, Xiaolei Li, Margaret Myslinska, and John Paul Sondag. 2007. Adaptive fastest path computation on a road network: a traffic mining approach. In *International Conference on Very large Data Bases*. 794–805.

[6] Ryan Herring, Aude Hofleitner, Saurabh Amin, T Nasr, A Khalek, Pieter Abbeel, and Alexandre Bayen. 2010. Using mobile phones to forecast arterial traffic through statistical learning. In *89th Transportation Research Board Annual Meeting*. 1–22.

[7] Aude Hofleitner and Alexandre Bayen. 2011. Optimal decomposition of travel times measured by probe vehicles using a statistical traffic flow model. In *IEEE International Conference on Intelligent Transportation Systems*. 815–821.

[8] Aude Hofleitner, Ryan Herring, Pieter Abbeel, and Alexandre Bayen. 2012. Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network. *IEEE Transactions on Intelligent Transportation Systems* 13, 4 (2012), 1679–1693.

[9] Timothy Hunter, Ryan Herring, Pieter Abbeel, and Alexandre Bayen. 2009. Path and travel time inference from GPS probe vehicle data. *NIPS Analyzing Networks and Learning with Graphs* 12, 1 (2009).

[10] Rob J Hyndman and George Athanasopoulos. 2014. *Forecasting: principles and practice*. OTexts.

[11] Erik Jenelius and Haris N Koutsopoulos. 2013. Travel time estimation for urban road networks using low frequency probe vehicle data. *Transportation Research Part B: Methodological* 53 (2013), 64–81.

[12] Zhanfeng Jia, Chao Chen, Ben Coifman, and Pravin Varaiya. 2001. The PeMS algorithms for accurate, real-time estimates of g-factors and speeds from single-loop detectors. In *IEEE International Conference on Intelligent Transportation Systems*. 536–541.

[13] Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang. 2006. Finding fastest paths on a road network with speed patterns. In *IEEE International Conference on Data Engineering*. IEEE, 10–10.

[14] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. 2005. On trip planning queries in spatial databases. In *Advances in Spatial and Temporal Databases*. Springer, 273–290.
[15] Mu Li, Amr Ahmed, and Alexander J. Smola. 2015. Inferring Movement Trajectories from GPS Snippets. In *ACM WSDM*. 325–334.
[16] Microsoft Bing 2017. Bing Maps: https://www.bing.com/maps/.
[17] Enda Murphy and James E Killen. 2010. Commuting economy: an alternative approach for assessing regional commuting efficiency. *Urban studies* (2010).
[18] National Administration of Surveying, Mapping and Geoinformation of China. 2016. Surveying and Mapping Law of the People's Republic of China.
[19] Michael A Niedzielski. 2006. A spatially disaggregated approach to commuting efficiency. *Urban Studies* 43, 13 (2006), 2485–2502.
[20] Karl F Petty, Peter Bickel, Michael Ostland, John Rice, Frederic Schoenberg, Jiming Jiang, and Ya'acov Ritov. 1998. Accurate estimation of travel times from single-loop detectors. *Transportation Research Part A: Policy and Practice* 32, 1 (1998), 1–17.
[21] Mahmood Rahmani, Erik Jenelius, and Haris N Koutsopoulos. 2013. Route travel time estimation using low-frequency floating car data. *IEEE International Conference on Intelligent Transportation Systems* (2013).
[22] John Rice and Erik Van Zwet. 2004. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems* 5, 3 (2004), 200–207.
[23] Martin Treiber and Arne Kesting. 2013. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (2013).
[24] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2016. A Simple Baseline for Travel Time Estimation using Large-Scale Trip Data. In *Proceedings of the 24th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '16)*. ACM, New York, NY, USA.
[25] Hongjian Wang, Yanmin Zhu, and Qian Zhang. 2013. Compressive sensing based monitoring with vehicular networks. In *INFOCOM, 2013 Proceedings IEEE*. 2823–2831. https://doi.org/10.1109/INFCOM.2013.6567092
[26] Yibing Wang and Markos Papageorgiou. 2005. Real-time freeway traffic state estimation based on extended Kalman filter: a general approach. *Transportation Research Part B: Methodological* 39, 2 (2005), 141–167.
[27] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 25–34.
[28] Bradford S Westgate, Dawn B Woodard, David S Matteson, and Shane G Henderson. 2013. Travel time estimation for ambulances using Bayesian data augmentation. *The Annals of Applied Statistics* 7, 2 (2013), 1139–1161.
[29] Chris Whong. 2014. Foiling NYC's Taxi Trip Data: http://chriswhong.com/open-data/foil_nyc_taxi/.
[30] Daniel B Work, O-P Tossavainen, Sébastien Blandin, Alexandre M Bayen, Toch Iwuchukwu, and Ken Tracton. 2008. An ensemble Kalman filtering approach to highway traffic estimation using GPS enabled mobile devices. In *IEEE Conference on Decision and Control*. 5062–5068.
[31] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *ACM SIGSPATIAL*. 99–108.

Received November 2016; revised August 2017; accepted October 2018