Efficient Traffic Estimation With Multi-Sourced Data by Parallel Coupled Hidden Markov Model

Senzhang Wang, Xiaoming Zhang, Fengxiang Li, Philip S. Yu, *Fellow, IEEE*, and Zhiqiu Huang

***Abstract*— Traffic congestion estimation in arterial networks with sparse GPS probe data is a practically important while substantially challenging research issue. The effectiveness and efficiency of the existing GPS probe data-based traffic estimation models are largely limited due to the following two challenges. First, due to the low sampling frequency of GPS probes, probe data are usually sparse, especially for some road links not located in the central urban areas. Second, due to the very complex temporal and spatial dependencies among the road links, the variable space of the existing traffic estimation models is huge. It is time consuming to get an accurate estimation of a large arterial road network with thousands of road links. To address the abovementioned issues, this paper proposes to extract traffic event signals from social media and incorporate them with GPS probe data to alleviate the data sparse issue. We first collect traffic-related posts that report various traffic events, including traffic jam, accident, and road construction from Twitter. By considering the GPS probe readings and the traffic event tweets as two types of observations, we next extend the conventional coupled hidden Markov model for integrating the two types of data to obtain a more accurate estimation of traffic conditions. To address the computational challenge, a parallel importance sampling-based electromagnetic algorithm is further introduced. We evaluate our model on the arterial network of downtown Chicago. The experimental results demonstrate the superior performance of the model in both effectiveness and efficiency.**

***Index Terms*— Social media, traffic estimation, CHMM.**

Manuscript received December 7, 2016; revised September 16, 2017 and May 9, 2018; accepted September 14, 2018. Date of publication November 8, 2018; date of current version July 31, 2019. This work was supported in part by the National Key Research and Development Program under Grants 2016YFB1000802 and 2018YFB1003902, in part by the NSFC under Grants 61602237, 61772270, and 61672313, in part by the Director's Project Fund of the Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, under Grant 2017KF03, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20171420, and in part by NSF under Grants IIS-1526499, IIS-1763325, and CNS-1626432. This paper was presented in part at the ECML-PKDD 2016. The Associate Editor for this paper was D. Work. *(Corresponding author: Xiaoming Zhang.)*

S. Wang and Z. Huang are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China.

X. Zhang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: yolixs@buaa.edu.cn).

F. Li is with the College of Computer and Information Science, Northeastern University, Boston, MA 02115 USA.

P. S. Yu is with the Department of Computer Science, The University of Illinois at Chicago, Chicago, IL 60607 USA, and also with the Institute for Data Science, Tsinghua University, Beijing 100084, China.

Digital Object Identifier 10.1109/TITS.2018.2870948

I. Introduction

CURRENTLY, GPS based probe vehicle data have become a significant data source available for traffic monitoring and sensing. As such, there is considerable research interests in exploring GPS probe data for conducting various traffic related applications such as traffic states estimation and prediction [1], [2]. However, the characteristics of probe data, including the low sampling frequency, and the randomness of its spatiotemporal coverage, make it insufficient for fully estimating traffic conditions of a large transportation network [3]. There are two major issues that largely limit the usability of the GPS probe data in traffic states estimation. First, the sampling frequency of GPS probes is usually relatively low, making the probe readings on some road links sparse, especially for the road links far away from the central urban areas. Second, due to the complex spatial and temporal dependencies among the road links, the variable space of the traffic estimation model can be huge and thus the computational complex is high.

Currently, it is a common practice for pedestrians, drivers and official transportation departments to share instant traffic information through social media like Twitter [4], [5]. A considerable amount of tweets that report traffic events like congestion and accident are posted instantly every day. Previous study [6] showed that there are thousands of traffic event tweets posted every day in some big cities of United States like Chicago and Los Angeles. Many such tweets, like "*Harrison St: accident at Kilbourn Ave, 2:04-4/2/2015,*" explicitly give the type of traffic event, time, and location information. Motivated by the rich traffic information available in social media, rising research efforts have been made to explore social media data for facilitating traffic related applications, including traffic event location identification [7], [8], traffic event detection [9], [10], and traffic congestion estimation [6], [11]–[13]. Although the utility of social media data has attracted much research attention, less attention is paid on developing both effective and efficient models to combine the multi-source data, particularly the social media data and the GPS probe readings for improving traffic states estimation.

The aims of this paper are 1) incorporating traffic event signals extracted from Twitter with GPS probe readings for more accurately estimating urban traffic conditions, and 2) propose an efficient solution to make the algorithm more scalable such that it can work on large transportation networks with thousands of road links. The challenges of the studied problem are three-fold. Firstly, the traffic information extracted from Twitter can be associated to multi-typed traffic events including congestion, accident, road construction, etc. It is difficult to model the potential impacts of the diverse traffic events on traffic congestion. For example, given a tweet that reports a traffic accident, how can we quantitively measure its impact on traffic congestion? Secondly, it is also difficult to combine the two types of data with totally different data formats seamlessly. A piece of GPS probe reading normally contains the time, speed, heading, and the exact location (longitude, latitude) information of a vehicle; while a tweet that reports a particular traffic event typically will mention the traffic event type, the time, and the road or road link information. It is non-trivial to integrate the two types of data directly to feed them into a traditional traffic estimation model. Thirdly, CHMM is shown to be effective to capture the spatial and temporal dependencies among the road links in traffic estimation, but the high computational complexity makes it infeasible to handle large road networks [1], [14].

To address the above challenges, we take Twitter as a complementary data source and collect traffic related tweets. We first extract the traffic event type, time, and location information from the tweets by data processing. To effectively fuse Twitter data and GPS probe data, we next extend the conventional CHMM [1], [15]. The extended model considers the GPS probe data and the traffic event tweets as two types of observations generated from two different distributions independently. The final estimation result is a weighted combination of the estimation results based on the two types of data. As the exact solution of CHMM is infeasible for a large network due to the exponential space and time consumption, we propose a parallel particle filtering method to more efficiently estimation the variables in the E-step of the EM algorithm. The proposed parallel algorithm evenly distributes the particles into multiple processors and conduct the resampling procedure in parallel. A rebalancing strategy is also introduced to guarantee the new sampled particles evenly distributed in the processors. Meanwhile, in the M-step we formulate the original optimization problem decomposable into smaller problems that can also be optimized in parallel.

To summarize, this paper makes the following contributions.

- An extended Coupled Hidden Markov Model is proposed to effectively combine the traffic event related tweets and traditional probe GPS readings for more accurately estimating traffic congestions.
- To reduce the computational complexity, a parallel EM-algorithm is proposed to more efficiently estimate the large number of variables in the model.
- The evaluation on both the arterial network of downtown Chicago and the synthetic datasets shows that 1) incorporating the social media data can significantly improve the performance of traffic estimation compared with existing methods, and 2) the proposed parallel algorithm achieves nearly linear speedup.

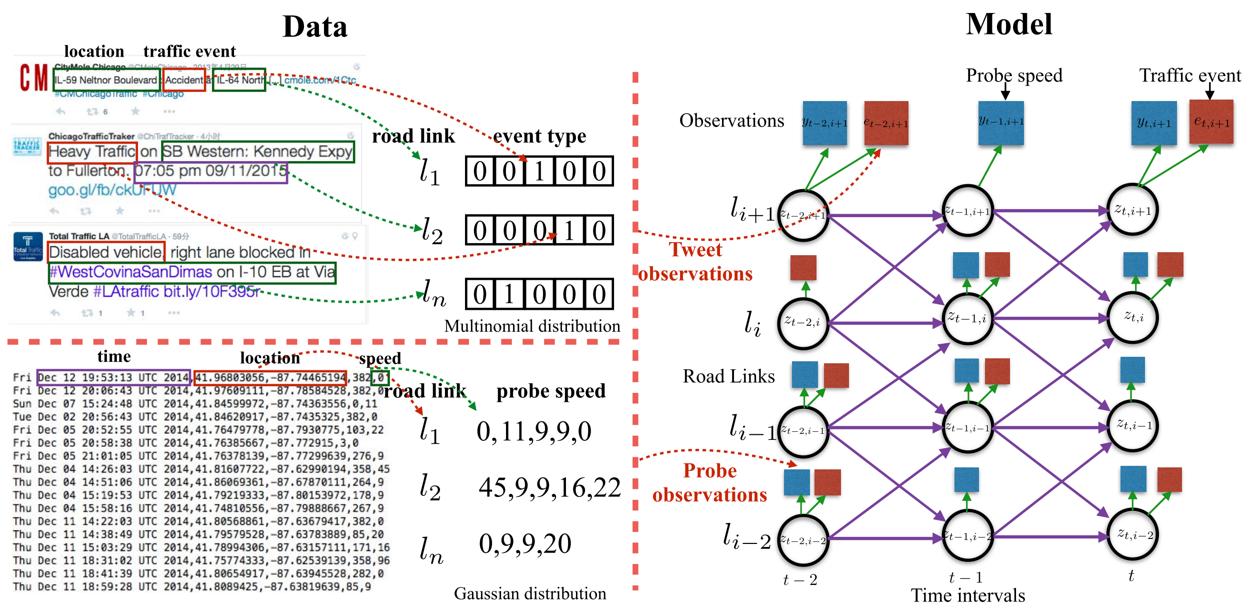
The remainder of the paper is organized as follows. Section II discusses related work. In Section III, we introduce the preliminary and show the framework of our method. Section IV introduces Twitter data collection. Section V elaborates the proposed CHMM. We describe the paralleled parameter inference in Section VI and evaluate our model in Section VII. Finally, we conclude this paper in Section VIII.

II. Related Work

Traditionally, traffic monitoring and estimation mainly rely on various road sensors, and can be roughly categorized into traffic modeling on individual roads [16]–[18] and on a road network [1], [2], [19], [20]. Helbing [17] employed a Fundamental Diagram to learn the relations among vehicle speed, traffic density, and volume for a particular road to estimate traffic condition on an individual road. Muñoz *et al.* [16] proposed a macroscopic traffic flow model SMM by utilizing the loop detector data to estimate the traffic density at unmonitored locations along a highway. Porikli and Li [18] proposed a Gaussian Mixture Hidden Markov Models to detect traffic condition with the MPEG video data. Gahrooei and Work [21] studied how to use hidden Markov Model to infer the traffic signal phases from the turning movement counters. Researches on traffic monitoring on a road network usually need to capture and model the correlations of the traffic conditions among the road links connected to each other [1], [2], [22]. Such models mainly utilized the Floating Car Data (FCD) or probe data generated by the GPS sensors equipped in vehicles. Fabritiis *et al.* [22] studied to use FCD data based on the traces of GPS positions to predict the traffic on Italian motorway network.

Herring *et al.* [1] have pioneered to apply a coupled Hidden Markov Model to capture the spatiotemporal dependencies among the road links of a road network in traffic state estimation. Although CHMM is shown to be effective in traffic state estimation, a major limitation is that CHMM is very time consuming to solve, and thus it is infeasible to be applied to large transportation networks. Herring *et al.* [1] and Wang *et al.* [23] proposed to apply particle filtering algorithm to estimate the parameters of CHMM. An important issue in particle filtering is the particle degeneracy. To address this issue, Polson and Sokolov [24] and Sokolov and Polson [25] proposed a Bayesian particle filter for tracking traffic flows that is capable of capturing nonlinearities and discontinuities present in flow dynamics. They showed that their proposed Bayesian particle filter is less prone to the degeneracy issue. However, particle filtering is still very time consuming because it needs to conduct a large number of simulations. To reduce the computational complexity, Mihaylova *et al.* [26] proposed a parallel particle filtering algorithm by dividing a large traffic network into several subnetworks. However, the issue is that it is not clear how to decompose the large traffic network. Unlike other types of networks that usually present community structures, it is hard to divide a road network of a city into a group of small road networks with very few connections among them. Thus a new parallel particle filtering algorithm that does not need to decompose the traffic network is necessary.

As the traffic sensor data such as GPS probe data are usually sparse, some researchers tried to collect traffic information from social media. Recently exploring traffic



CityMole Chicago @CMoleChicago · 2013年4月29日
IL-59 Neltnor Boulevard : Accident at IL-64 North [...] cmole.com/4Ctc #CMChicagoTraffic #Chicago

ChicagoTrafficTraker @ChiTrafTracker · 4小时
Heavy Traffic on SB Western: Kennedy Expy to Fullerton. 07:05 pm 09/11/2015 goo.gl/fb/ckUFUW

Total Traffic LA @TotalTrafficLA · 59分
Disabled vehicle, right lane blocked in #WestCovinaSanDimas on I-10 EB at Via Verde #LAtraffic bit.ly/10F395r

```
Fri Dec 12 19:53:13 UTC 2014,41.96803056,-87.74465194,382,0
Fri Dec 12 20:06:43 UTC 2014,41.97609111,-87.78584528,382,0
Sun Dec 07 15:24:48 UTC 2014,41.84599972,-87.74363556,0,11
Tue Dec 02 20:56:43 UTC 2014,41.84620917,-87.7435325,382,0
Fri Dec 05 20:52:55 UTC 2014,41.76479778,-87.7930775,103,22
Fri Dec 05 20:58:38 UTC 2014,41.76385667,-87.772915,3,0
Fri Dec 05 21:01:05 UTC 2014,41.76378139,-87.77299639,276,9
Thu Dec 04 14:26:03 UTC 2014,41.81607722,-87.62990194,358,45
Thu Dec 04 14:51:06 UTC 2014,41.86069361,-87.67870111,264,9
Thu Dec 04 15:19:53 UTC 2014,41.79219333,-87.80153972,178,9
Thu Dec 04 15:58:16 UTC 2014,41.74810556,-87.79888667,267,9
Thu Dec 11 14:22:03 UTC 2014,41.80568861,-87.63679417,382,0
Thu Dec 11 14:38:49 UTC 2014,41.79579528,-87.63783889,85,20
Thu Dec 11 15:03:29 UTC 2014,41.78994306,-87.63157111,171,16
Thu Dec 11 18:31:02 UTC 2014,41.75774333,-87.62539139,358,96
Thu Dec 11 18:41:39 UTC 2014,41.80654917,-87.63945528,282,0
Thu Dec 11 18:59:28 UTC 2014,41.8089425,-87.63819639,85,9
```

Fig. 1. Framework of the proposed model.

related information from Twitter to detect traffic events or monitor traffic conditions has been a hot research topic [4], [6], [9], [11], [13]. Most previous works focused on investigating either how to extract and visualize the traffic event information from tweets [4], [9], [11] or how to locate the traffic events mentioned in the tweets [7], [8]. As traffic event data are usually sparse and imbalanced, imbalanced learning techniques are usually explored [27]. The work in [13] is the first to estimate traffic congestion of an arterial network by collecting traffic related tweets from Twitter. Wang *et al.* [6] further incorporated other information such as social events and road features with social media data to more effectively estimate citywide traffic congestions. However, as the probe data are not explored, the performance are usually not desirable due to the sparse and noisy Twitter data [13]. Our previous work [14] proposed an enhanced CHMM model namely E_CHMM to integrate the GPS probe readings and traffic event tweets. However, E_CHMM uses traditional sequential parameter estimation algorithms, and thus it is still less efficient when dealing with large transportation networks. About traffic state estimation, Seo *et al.* [28] provided a comprehensive survey that one can refer to for more information on this research problem.

III. Framework and Problem Definition

In this section, we will first give some definitions. Then we will introduce the framework of our model followed by a formal problem definition. Next we will make some basic assumptions to model the problem.

Definition 1 (An Observation of Traffic Event Tweet $e_{t,l,i}$ [14]): We represent a tweet observation of traffic event occurring on the road link l at time t as such a tuple $e_{t,l,i} = (c, loc, t)$, where c is the traffic event category, loc represents the location or road link, and t denotes the time.

Definition 2 (An Observation of GPS Probe Reading $y_{t,l,i}$ [14]): We represent an observation of GPS probe reading on the road link l at time t as a vector $y_{t,l,i} = (s, lat, lon, head, t)$, where s is the vehicle speed, lat is the latitude, lon is the longitude, $head$ is the heading of the probe, and t is the time.

Definition 3 (A Road Link l [14]): We use the intersections to partition an arterial road R into several road links $R = \{l_1, l_2, \ldots\}$. Each road link l can be represented as a tuple $l = (Link\_ID, Start\_Inter, End\_Inter)$, where $link\_ID$ is the ID of the road link, $Start\_Inter$ is the start intersection, and End_Inter is the end intersection.

Definition 4 (Neighbor Road Links [14]): Two road links l_1 and l_2 are called neighbor road links if they connect to each other, namely they share an intersection. Particularly, the road link l is also considered as a neighbor road link of itself. We denote all the neighbor links of road link l as N_l.

A. Traffic Congestion

Traffic congestion refers to a condition on transportation networks that occurs as use increases, and it is characterized by slower speeds, long trip times, and increasing vehicular queueing. As a fuzzy concept, the definition of traffic congestion can vary significantly from time to time and place to place. For example, the concept of congestion in highway can be very different from it on an arterial road in urban area. Therefore, congestion is difficult to define precisely in a mathematical sense [29]. A straightforward definition of traffic congestion, which is also used in this paper is that the average vehicle speed is lower than a threshold. In this paper, we use the 5-state traffic condition system based on the average vehicle speed defined by Chicago Transit Authority (CTA), and we will introduce it in detail later.

Fig.1 shows the framework of our method. It contains two parts: the data collection and processing part, and the model part. One can see that there are two types of data in the model, traffic event tweets and GPS probe readings. From each traffic event tweet, we first extract the traffic event type, location, and time information, and then map it to the corresponding road link by geocoding. Similarly, we extract the exact location and travel speed information from each GPS probe reading, and then map it to the corresponding road link. For each road link, we assume the occurrence of traffic events on it follows a multinomial distribution, and the vehicle speed in a particular time interval follows a Gaussian distribution [1].

We model the spatiotemporal conditional dependencies of arterial traffic using a probabilistic graphical model *Coupled Hidden Markov Model* (CHMM). A CHMM models a system of multiple interaction processes which are assumed to be a Markov process with unobserved states. In our model, the multiple processes evolving over time are the discrete traffic states of each link in the road network (the circles in the model part of Fig.1). Since we do not observe the state of each link for all times, we consider them as *hidden*. We can observe the vehicle speed and traffic events from GPS probe and tweets (the blue and red squares in the model part of Fig.1), and the traffic speed and event on each link are conditioned on its hidden state. In addition, a coupled structure to the HMM specifies the local dependencies between adjacent links of the arterial network. Given the problem framework illustrated in Fig.1, we formally define the studied problem as follows.

B. Problem Definition

Given an arterial road network $\mathcal{G}$ with L road links in T time intervals, the tweet observations of traffic events $\{e_{t,l,i}\}_{t=1:T}^{l=1:L}$, and the GPS probe readings $\{y_{t,l,i}\}_{t=1:T}^{l=1:L}$ on $\mathcal{G}$, our goal is to accurately and efficiently infer the hidden congestion states $\{z_{t,l}\}_{t=1:T}^{l=1:L}$ for each road link l in each time interval t.

Following previous traffic estimation models [1], [15], we make the following assumptions for computational tractability.

- *Discrete traffic states:* For each time interval t, the traffic condition on link l is represented by a discrete value s_t^l, which indicates the level of congestion.
- *Conditional independence of link travel speed:* Conditioned on the state s_t^l of a link l, the travel speed distribution on l is independent from all the other traffic variables.
- *Conditional independence of traffic events:* Conditioned on the state s_t^l of a link l, the probability of traffic event $e_{t,l,i}$ occurring on link l is independent from all the other traffic variables.
- *Conditional independence of state transitions:* Conditioned on the states of link l and its neighbor links in time interval t, the state of link l at time $t+1$ is independent from all the other current link states, all the past link states and observations.

The second and third assumptions show that the two types of observations are conditionally independent and only determined by the current traffic states of the road links. The fourth assumption implies that the traffic state of each link is only related to its neighbors in the last time interval, but independent of the states of other links in earlier time intervals.

IV. Data Collection

A. Twitter Data Collection

This paper focuses on studying the traffic conditions in Chicago, and we collect tweets in Chicago from two types of accounts as in [14]: *traffic authority Twitter accounts* and *regular Twitter user accounts*.

1) Traffic Authority Twitter Accounts: Traffic authority Twitter accounts refer to the Twitter accounts that specialize in posting traffic related information. Such accounts are mostly operated by official transportation departments. Tweets posted by these accounts are formal and easy to process, and the exact location and time information are explicitly given such as the tweet "*Heavy Traffic on NB Western: Fullerton to Kennedy Expy. 06:15 pm 02/13/2015*." We identify 10 such Twitter accounts that report real-time traffic information of Chicago: *ChicagoDrives*, *ChiTraTracker*, *roadnowChicago*, *traffic_Chicago*, *IDOT_Illinois*, *WGNtraffic*, *TotalTrafficCHI*, *GeoTrafficChi*, *roadnowil*, and *rosalindrossi*.

2) Regular Twitter User Accounts: We also collect traffic related tweets from regular users. We selected 100,000 Twitter users registered in Chicago, and crawled more than 32.3 million tweets posted by these users. Next, two major steps are conducted for data preprocessing. 1) *Traffic Event Tweets Identification.* We select traffic event tweets from all the crawled tweets which match at least one term of the predefined vocabularies: "*stuck*," "*congestion*," "*jam*," "*crowded*," "*pedestrian*," "*driver*," "*accident*," "*crash*," "*road blocked*," "*road construction*," "*slow traffic*," "*heavy traffic*," "*bad traffic*" and "*disabled vehicle*." We first select the tweets that contain at least one of the above keywords by keywords matching. Based on the keywords, we can also identify the traffic event category. 2) *Tweet Geocoding.* We then geocode tweets to the road links.

For the tweets collected from traffic authority accounts, we can very easily locate the road link where the traffic event occurs as such tweets usually have fixed formats. Taking the tweet "*Heavy Traffic on NB Western: Fullerton to Kennedy Expy. 06:15 pm 02/13/2015*" as an example again, we can extract the road link through the pattern *{on R1: R2 to R3}*, where $R1$, $R2$, and $R3$ are street names. For the tweets collected from general sensor users, it is harder to map them to the exact road links. A small proportion (less than 5%) of such tweets are geo-tagged. Thus we can correctly map them to the corresponding road link. For most tweets without geotags, we extract the name of the streets and landmarks information from the tweets. Taking the tweet "*Bad traffic at Roosevelt this morning*" as an example, we can extract the street name "Roosevelt" by matching it with all the Chicago street names. Thus we can map the traffic event to the road link "Roosevelt." If only one street name is mentioned in the tweet as in the example, we roughly consider the traffic event will influence the entire street. If two street names are mentioned and forms the pattern *{on R1…R2}*, we consider the traffic event happen on the road link of street $R1$ that is close to the intersection between $R1$ and $R2$.

As shown in Table I, in all we obtain 245,568 traffic event tweets from April 2014 to December 2014, around 80% of

TABLE I
STATISTICS OF THE DATA IN CHICAGO

Traffic related tweets			
#Traffic related tweets	Congestion	Accident	Others
245,568	163,742	77,454	4,372
GPS probe data			
# of probe readings		# of readings per link per hour	
2,351,647		2.6	

which are collected from traffic authority accounts. Each tweet reports a traffic event. 163,742 of them are related to traffic congestion, 77,454 are related to accident, and 4,372 are reporting other traffic events such as road construction and road closure. We categorize these tweets into three types: congestion, accident and others. Other traffic events include road construction, road closure, etc.

B. GPS Probe Data

We have two parts of GPS probe data collected in downtown Chicago. The first part pf probe data is generated by more than 2,000 Chicago Transit Authority (CTA for short) public passenger buses from 11/25/2014 to 12/30/2014. Every 10 minutes the probe will produce a probe reading including the speed and location information of the CTA bus. This data contains about 5 million GPS probe readings in total, and it is publicly available[1]. We use the CTA bus probe data as ground truth (introduce in detail later). The second part of probe data is collect from probes installed in normal vehicles including cars and trucks. In this part of data, probes produce readings in every 15 minutes. There are more than 10 million readings in total, and we select more than 2.35 million of them that are produced in the time period from 11/25/2014 to 12/30/2014. Each such probe reading contains the following information: time, latitude, longitude, heading, and speed. For each probe reading, we first map it to the corresponding road link based on its latitude and longitude information through geocoding. Table I shows the statistics of our traffic event tweets and GPS probe reading data. One can see that on average there are only 2.6 probe readings per link per hour, which is sparse.

V. PE_CHMM: PARALLEL COUPLED HIDDEN MARKOV MODEL TO INCORPORATE MULTI-SOURCED DATA

Before introducing the method, we first give some notations and their meanings in Table II. We use boldface capital letters to denote the observations or hidden state matrixes on all the road links in all the time intervals. For example, **Y** denotes all the GPS probe observations. We use capital letters with subscripts to denote the observations or hidden state vectors in a particular time interval. For example, Y_t denotes the GPS probe observations on all the road links in time interval t.

There are three groups of variables that we need to estimate in our model, the initial traffic state probability matrix Π,

[1]https://data.cityofchicago.org/Transportation/Chicago-Traffic-Tracker-Historical-Congestion-Esti/77hq-huss/data

TABLE II
NOTATIONS AND MEANINGS

Notations	Meanings
L	Number of road links
T	Number of time intervals
M	Number of traffic event types
S	Number of traffic states
O	Number of processors
N_l	The set of all the neighbor links of road link l
N_{li}	The $i\text{-}th$ neighbor based on the lexicographical order of $link\_ID$, $link\_ID \in N_l$
$y_{t,l}$	The set of probe observations for link l in time slot t
$y_{t,l,i}$	One probe observation for link l in t, $y_{t,l,i} \in y_{t,l}$
$e_{t,l}$	The set of traffic event observations for link l in t
$e_{t,l,i}$	One traffic event observation for link l in t, $e_{t,l,i} \in e_{t,l}$
π_l^s	The initial probability that link l begins in state s
A_l	The state transition probability matrix for link l with respect to its neighbors N_l
$g_l^s(\cdot)$	The probability density function of travel speed for link l in traffic state s
$f_l^s(\cdot)$	The distribution function of traffic event for link l in traffic state s
P_l^s	The parameters of the probability density function $g_l^s(\cdot)$ and $f_l^s(\cdot)$
$z_{t,l}^s$	The probability of link l being in traffic state s in t
$q_{t,l}^{R_i,s}$	The probability of link l being in traffic state s for time period t given that its neighboring links N_l are in states $R_i = (r_{i1}, r_{i2}, ...r_{i\|N_l\|})$ in $t-1$

the traffic state transition probability matrix **A**, and the distribution parameters **P** of the observations. Next we introduce these variables in details. Each element π_l^s in Π denotes the initial probability of road link l in traffic state s. A_l of **A** is the traffic state transition probability matrix for link l. It is a matrix of size $S^{|N_l|} \times S$, where $S^{|N_l|}$ represents the number of all possible states of the neighbors N_l of link l. Each element $A_l(R_i, s)$ represents the probability of link l transiting to state s given that its neighbors N_l are in states $R_i = (r_{i1}, r_{i2}, \dots r_{i|N_l|})$. We give an example in Table II to further explain the transition matrix of a road link. Assume the neighbor links of l_1 are (l_1, l_2), and there are two traffic states 1 (congestion) and 0 (flow). The first column in Table III gives the current traffic states of all the neighbor links, and the second and third columns are the probabilities of the road link l_1 transiting to traffic state 0 and 1 in the next time slot, respectively. One can see that the sum of the two probability in each row is 1, and these probabilities are unknown and need to estimate. For the observation distribution parameters **P**, it contains the Gaussian distribution parameters $g(\cdot)$ for the probe speed observations and the multinomial distribution parameters $f(\cdot)$ for the traffic event observations. $g_l^s(\cdot)$ is the probability density function of vehicle speed for link l in state s. We assume it follows Gaussian distribution [1]. $f_l^s(\cdot)$ represents the distribution of traffic event number for link l in state s. We assume it follows Multinomial distribution.

With the above notations, we give the complete log likelihood of the observation data and hidden variables. Typically, the log likelihood of the hidden variables and observations can

TABLE III
AN EXAMPLE OF THE TRANSITION MATRIX OF ROAD LINK l_1 WITH (l_1, l_2) AS ITS NEIGHBOR LINKS ON A 2-STATE TRAFFIC STATE SYSTEM

Current traffic states of the links (l_1, l_2)	Probability of l_1 transiting to state 0	Probability of l_1 transiting to state 1
(0, 0)	0.1	0.9
(0, 1)	0.4	0.6
(1, 0)	0.7	0.3
(1, 1)	0.6	0.4

be written as follows,

$$\begin{aligned} &lnP(\mathbf{Y}, \mathbf{E}, \mathbf{Z}) \\ &\quad = lnP(Z_1) + \sum_{t=2}^{T} lnP(Z_t|Z_{t-1}) + \sum_{t=1}^{T} lnP(Y_t, E_t|Z_t) \\ &\quad = lnP(Z_1) + \sum_{t=2}^{T} lnP(Z_t|Z_{t-1}) + \sum_{t=1}^{T} lnP(Y_t|Z_t) \\ &\qquad + \sum_{t=1}^{T} lnP(E_t|Z_t) \end{aligned} \tag{1}$$

On the second line, the first term of the formula (1) represents the initial probability of traffic states Z_1 for all the road links, the second term represents the probability that traffic states Z_{t-1} in time interval $t-1$ transit to the states Z_t in the next time interval t, and the third term is the probability of observations Y_t and E_t conditioned on the traffic states Z_t. Since the GPS probe observations are conditionally independent from the traffic event observations, we can further decompose $\sum_{t=1}^{T} lnP(Y_t, E_t|Z_t)$ as shown in the third line of formula (1).

The initial probability of the congestion states in the first time interval is

$$lnP(Z_1) = \sum_{l=1}^{L}\sum_{s=1}^{S} z_{1,l}^{s} ln\pi_l^s \tag{2}$$

The log probability of congestion state transiting from time interval $t-1$ to t can be further represented as follows,

$$lnP(Z_t|Z_{t-1}) = \sum_{l=1}^{L}\sum_{s=1}^{S}\sum_{i=1}^{S^{|N_l|}}\Big(\prod_{N_{lj}\in N_l} z_{t-1,N_{lj}}^{r_{ij}} z_{t,l}^{s} lnA_l(R_i, s)\Big) \tag{3}$$

The third summation of formula (3) is over all the possible traffic states $S^{|N_l|}$ of the neighbors N_l, while the subsequent product is over terms on each of its individual neighbor state given the neighbor states $(r_{i1}, \ldots, r_{i|N_l|})$.

The probability of probe speed observations Y_t given the congestion states Z_t can be represented as

$$lnP(Y_t|Z_t) = \sum_{l=1}^{L}\sum_{s=1}^{S} z_{t,l}^{s}\Big(\sum_{y_{t,l,i}\in y_{t,l}} ln(g_l^s(y_{t,l,i}))\Big) \tag{4}$$

The probability of traffic event observations E_t given the congestion states Z_t can be represented as

$$lnP(E_t|Z_t) = \sum_{l=1}^{L}\sum_{s=1}^{S} z_{t,l}^{s}\Big(\sum_{e_{t,l,i}\in e_{t,l}} ln(f_l^s(e_{t,l,i}))\Big) \tag{5}$$

A. Solution: EM Algorithm

Given the distribution function parameters P_l^s of the observations and the state transition matrix A_l, it is possible to estimate the congestion states of the links. Similarly, given the congestion states of the road links, we can estimate the parameters in the model. Motivated by this idea, EM algorithm can be applied to solve the model.

In the E-step, for each road link l we compute the expected state probabilities $z_{t,l}^s$ and the transition probabilities $q_{t,l}^{R_i,s}$ given the observations $(y_{t,l},\ e_{t,l})$, the distribution parameters P_l^s, and the state transition probability matrix A_l.

$$z_{t,l}^s \leftarrow E(z_{t,l}^s|y_{t,l}, e_{t,l}, P_l^s, A_l) \tag{6}$$

$$q_{t,l}^{R_i,s} \leftarrow E(q_{t,l}^{R_i,s}|y_{t,l}, e_{t,l}, P_l^s, A_l) \tag{7}$$

Note that the traffic state $z_{t,l}^s$ is inferred based on both the GPS probe observation $y_{t,l}$ and the tweet observation $e_{t,l}$. To distinguish the importance of the two types of observations, we rewrite formula (7) as follows.

$$z_{t,l}^s \leftarrow \begin{cases} E(z_{t,l}^s|e_{t,l}, P_l^s, A_l) & \text{if } Cardinality(y_{t,l}) = 0 \\ \alpha_{t,l} E(z_{t,l}^s|y_{t,l}, P_l^s, A_l) & \\ \quad + (1-\alpha_{t,l}) E(z_{t,l}^s|e_{t,l}, P_l^s, A_l) & \text{otherwise} \end{cases} \tag{8}$$

If only the tweet observation $e_{t,l}$ is available on road link l in time interval t, the congestion state $z_{t,l}^s$ is estimated only based on $e_{t,l}$. Otherwise, $z_{t,l}^s$ is estimated by using both types of observations. $\alpha_{t,l}$ is the confidence of the probe observations. The idea is that if sufficient probe observations are available, we trust more on the traffic state $z_{t,l}^s$ estimated by probe observations. If the probe data are very spare, we trust more on the estimation results with the tweet observations. Here we use a sigmoid function to estimate the importance of the coefficient $\alpha_{t,l} = \frac{1}{1+e^{\theta - Cardinality(y_{t,l})}}$, where θ is a threshold of the probe observation size. More probe observations result in a larger $\alpha_{t,l}$. In this paper we set $\theta = 3$.

In the M-step, we maximize the expected complete log-likelihood, given the probabilities $z_{t,l}^s$ and the transition probabilities $q_{t,l}^{R_i,s}$.

$$\begin{aligned} &(P_l^s, A_l, \pi_l^s) \\ &\leftarrow \mathop{\mathbf{argmax}}_{\mathbf{P},\mathbf{A},\Pi} lnP(\mathbf{Y}, \mathbf{Z}, \mathbf{E}, \mathbf{P}, \mathbf{A}, \Pi) \\ &\qquad subject\ to \sum_{s=1}^{S} A_l(R_i, s) = 1,\ A_l(R_i, s) \in [0, 1],\ \forall l, R_i, s; \\ &\qquad\qquad \sum_{s=1}^{S} \pi_l^s = 1, \quad \pi_l^s \in [0, 1],\ \forall l, s. \end{aligned} \tag{9}$$

VI. Parameter Inference With Parallel Particle Filtering

On small networks, it is possible to do exact inference in the CHMM by converting it to a HMM. However, it is intractable to do exact inference for any reasonable traffic network with the naive solution due to the following reasons. 1) Computation of the forward variable involves S^L additions and N multiplications at each of T time steps; 2) each forward variable requires $8S^L$ bytes of memory to store, and all T of them must be stored; and 3) the transition matrix itself is $S^L \times S^L$. To address this computational challenge, previous work proposed to apply importance sampling based method to approximately estimate the large number of variables [1], [15]. However, the time cost of importance sampling based variable estimation methods is still high since a large number samples are needed to obtain an accurate estimation. In this section, we will first introduce to apply sequential particle filtering to estimate the variables in the E-step. Based on it, next we will propose a parallel particle filtering algorithm which can evenly partition the samples into multiple processors and conduct particle filtering in parallel.

A. Parameter Inference With Parallel Particle Filtering

1) Sequential Particle Filtering: As a popular sequential importance sampling method, particle filtering is widely adopted to approximately estimate the internal states in dynamical systems such as signal processing and Bayesian statistical inference [24], [25]. In our setting, each particle or sample represents an instantiation of the traffic state evolution on the traffic network. Given the observed probe readings and the traffic event tweets, each particle or sample is assigned a weight proportional to the probability of the observations. With a large number of sampled particles, particle filtering can estimate the traffic state probabilities for all the road links and the traffic state transition probabilities.

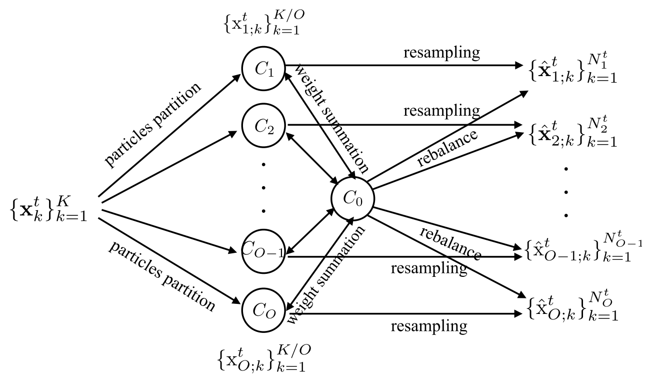
The details of the sequential particle filtering algorithm to estimate traffic states in the E-step is given in Algorithm 1. One can see that the Sequential Particle Filtering algorithm (SPE for short) contains three major steps: *particles generation*, *weight computation*, and *resampling*. In the *particle generation* step, the algorithm generates new particles based on the sampled particles in the last time step and the state transition matrix. In the *weight computation* step, the algorithm computes the weight of each particle based on the probability of generating the traffic event observations E_t and the probe observation Y_t given the traffic state x_k^t. With the computed particle weight, the *resampling* step first normalizes the weight and then resamples the particles with replacement to avoid the particle degeneracy problem. The degeneracy occurs when one particle has an accumulated weight close to one while the weights of all the other particles are close to zero.

Although SPF is an efficient method to approximately estimate the state probability distribution matrix $\mathbf{Z}$ and the transition probability $\mathbf{Q}$ based on a large number of generated particles, it is still very time consuming and hard to scale up to a large arterial network in practice. Its major limitation is that to obtain a relatively accurate estimation, usually a very large number of particles are needed, and more particles mean a larger computational cost. In our case, as the number L of road links can be large, the computational complexity becomes even higher as each particle is a L-dimensional vector. The overall computational complexity of SPF is $O(KTL)$, where K is the number of particles, T is the number of time intervals, and L is the number of road links. If we also consider the iteration times N in the EM algorithm, the computational complexity is $O(NKTL)$, which is very time consuming.

Algorithm 1 Sequential Particle Filtering for Traffic States Estimation

```
Input: Number of particles $K$ and time intervals $T$,
       the state transition matrix $\mathbf{A}$, the parameters of
       the observation probability function $P_l^s$ for each
       road link $l$.
Output: The state probability distribution matrix $\mathbf{Z}$, and
        the transition probability $\mathbf{Q}$.
1  Initialization: randomly sample $K$ particles $\{\mathbf{x_k^0}\}_{\mathbf{k=1}}^{\mathbf{K}}$;
2  for $t = 1 : T$ do
3      // particles generation;
4      Generate $K$ traffic state particles $\{\mathbf{x}_k^t\}_{k=1}^K$ based on
       previous particles $\{\mathbf{x}_k^{t-1}\}_{k=1}^K$ and state transition matrix
       $\mathbf{A}$: $\mathbf{x}_k^t \sim q(\mathbf{x}_k|\mathbf{x}_k^{t-1})$;
5      // weight computation;
6      Compute the weights: $w_k^t = p(Y_t, E_t|\mathbf{x}_k^t)$;
7      // resampling;
8      Normalize the weights: $\hat{w}_k^t = \frac{w_k^t}{\sum_{j=1}^K w_j^t}$;
9      Resample $K$ random particles $\{\hat{\mathbf{x}}_k^t\}_{k=1}^K$ from $\{\mathbf{x}_k^t\}_{k=1}^K$
       with replacement in proportion to the weights:
       $\{\hat{w}_k^t\}_{k=1}^K$;
10     Replace the particle set with these new particles:
       $\{\mathbf{x}_k^t\}_{k=1}^K \leftarrow \{\hat{\mathbf{x}}_k^t\}_{k=1}^K$ ;
11     Set the weights to be equal: $\hat{w}_k^t = \frac{1}{N}, k = 1, \ldots, N$
12 Estimate the state probability matrix $\mathbf{Z}$ based on equation
   (8) and transition probability $\mathbf{Q}$ based on equation (7)
   with the $K$ samples;
13 return $\mathbf{Z}$, $\mathbf{Q}$
```

2) Parallel Particle Filtering: To reduce the computational complexity, we propose a parallel particle filtering schema. The *particle generation* and *weight computation* steps are readily parallelizable, being independent operations on each particle x_k^t and its weight w_t^i. The *resampling* step, however, is a collective operation across all the particles and weights, making the parallelism difficult. The *resampling* step contains two steps. The first step is to normalize the weights such that the weights of the particles is considered as the probability of resampling them in the next iteration. The normalization operation requires accessing the weights of all the particles. The second step resamples the samples with replacement based on their weights, and thus the algorithm needs to interact with all the particles. We follow the work [30] and exploit a data parallel approach to conduct the parallelism. As a straightforward approach, it first partitions all the particles into several subsets of the same size. Each subset is then assigned to



$\{x^t_{1;k}\}_{k=1}^{K/O}$
C_1
C_2
C_{O-1}
C_O
C_0
$\{x^t_k\}_{k=1}^K$
particles partition
particles partition
weight summation
weight summation
resampling
resampling
resampling
resampling
rebalance
rebalance
$\{\hat{x}^t_{1;k}\}_{k=1}^{N^t_1}$
$\{\hat{x}^t_{2;k}\}_{k=1}^{N^t_2}$
$\{\hat{x}^t_{O-1;k}\}_{k=1}^{N^t_{O-1}}$
$\{\hat{x}^t_{O;k}\}_{k=1}^{N^t_O}$
$\{x^t_{O;k}\}_{k=1}^{K/O}$

Fig. 2. Framework of the parallel particle filtering.

one processor, and multiple processors process corresponding particle subsets separately in parallel. To address the issue that the resampling step of SPF is hard to parallel, we propose a local resampling strategy. In addition, a particles rebalancing algorithm is also proposed to avoid the need of operating on all the particles and weights. Assume there are $O+1$ processors denoted as $\{C_o\}_{o=1}^{O}$, and one is the master processor and the others are subtask processors. The framework of the proposed parallel approach is depicted in Fig. 2. We first describe the notations and then briefly introduce the workflow of the algorithm. $\{\mathbf{x}^t_k\}_{k=1}^K$ is the initially generated particles, C_0 is the master processor and $\{C_o\}_{o=1}^{O}$ are the subtask processors, where $\{\hat{\mathbf{x}}^t_k\}_{k=1}^{N^t_o}$ is the locally resampled N^t_o particles in processor o. The workflow of the algorithm in each iteration is as follows:

- Partition the particles into O subset of the same size, and assign each subset to a processor C_o;
- Generate particles $\{\mathbf{x}^t_{o;k}\}_{k=1}^{K/O}$ in parallel for all the subtask processors;
- Compute the particle weights in parallel for all the subtask processors;
- The master processor sums the particle weights and calculates the number of the resampled particles for each subtask processor;
- Resample the particles $\{\hat{\mathbf{x}}^t_{o;k}\}_{k=1}^{N^t_o}$ in parallel for all the subtask processors;
- The master processor rebalances the subset particles to make their size equal;

Instead of conducting the resampling operation on the entire particle set, we conduct a local resampling on each particle subset separately and independently, making the resampling step readily to run in parallel. However, a problem is that we need to determine the size of the resampled particles for each subset. As the the particles are randomly sampled to assign to each processor, the particle weights are not evenly distributed. For example, assuming the sums of the particle weights of two particle subsets are 0.2 and 0.3, respectively, we should resample more particles from the second subset than the first one since its total particle weight is larger. To address this issue, we first calculate the weight distribution of all the particle subsets, and then resample new particles based on their weight distribution from each subset separately and independently. Specifically, as show in the lines 7-15 of Algorithm 2, we first compute the weights in parallel for each processor (Line 7). Then each subtask processor sums the local weights and upload the sum $w_{local;o}$ to the master processor (Line 8-10). When all the subtask processors have submitted corresponding weight sums, the master processor computes the global weight sum w_{global} by summating all the local sums and return the value to each subtask processor (Line 11). Next each subtask processor calculates the number of the resampled particles based on its local weight sum and the global weight sum by $N^t_o = \lceil K\frac{w_{local;o}}{w_{global}}\rceil$, where $\lceil\rceil$ denotes rounding up the number (Line 12-13). With the resample size N^t_o and locally normalized weight $\hat{w}^t_{o;k}$, each subtask processor locally resamples the particles with replacement (Line 14-15).

The rebalance algorithm used in line 16 of Algorithm 2 is given in Algorithm 3. Given the unevenly distributed particles in the O processors, the goal of this algorithm is to rebalance

Algorithm 2 Parallel Particle Filtering for Traffic States Estimation

```
   Input: The same as in Algorithm 1
   Output: The same as in Algorithm 1
1  Initialization: randomly sample $K$ particles $\{\mathbf{x}^0_k\}_{k=1}^K$;
2  for $t = 1 : T$ do
3      for processor $o = 1 : O$ do
4          // particle generation in parallel;
5          Generate $K/O$ particles $\mathbf{x}^t_{o;k}$: $\mathbf{x}^t_{o;k} \sim q(\mathbf{x}_{o;k}|\mathbf{x}^{t-1}_{o;k})$;
6          // weight computation in parallel;
7          Compute the weights: $w^t_{o;k} = p(Y_t, E_t|\mathbf{x}^t_{o;k})$;
8          Sum the local weights: $w_{local;o} = \sum_{i=1}^{K/O} w^t_{o;i}$;
9          // global weight summation;
10         Upload the sum of local weights $w_{local;o}$ to the
           master processor;
11         Wait for the master processor to return the global
           sum of the weight: $w_{global} = \sum_{o=1}^{O} w_{local;o}$;
12         // resampling in parallel;
13         Calculate the number of resampled particles for
           processor $m$ and upload to the master processor:
           $N^t_o = \lceil K\frac{w_{local;o}}{w_{global}}\rceil$;
14         Locally normalize the weights: $\hat{w}^t_{o;k} = \frac{w^t_{o;k}}{\sum_{k=1}^{K/O} w^t_{o;k}}$;
15         Resample $N^t_o$ particles $\{\hat{\mathbf{x}}^t_{o;k}\}_{k=1}^{N^t_o}$ from $\{\mathbf{x}^t_{o;k}\}_{k=1}^{K/O}$
           with replacement in proportion to the weights
           $\{\hat{w}^t_{o;k}\}_{k=1}^K$;
16         Replace the sample set with these new samples,
           i.e. $\{\mathbf{x}^t_k\}_{k=1}^K \leftarrow \{\hat{\mathbf{x}}^t_k\}_{k=1}^{K/O}$;
17         Set the weights to be equal:
           $\hat{w}^t_k = \frac{1}{N}, k = 1, \ldots, N$;
18         // particles rebalance;
19         Rebalance the number of particles for all the
           processors with Algorithm 3.
20 Estimate the state probability matrix $\mathbf{Z}$ and transition
   probability $\mathbf{Q}$ with the $K$ samples;
21 return $\mathbf{Z}$, $\mathbf{Q}$;
```

Algorithm 3 Particles Rebalance Algorithm

```
Input: Number of particles distributed in the $O$
       processors $\{N_o^t\}_{o=1}^O$.
Output: A balanced particle distributions in the $O$
       processors with each one having $N/O$ particles
1 for processor o: 1 to $O$ do
2     if $N_o^t > N/O$ then
3         Randomly sample $N_o^t - N/O$ particles without
          replacement and put them to the particle pool;
4     else if $N_o^t > N/O$ then
5         Randomly sample $N/O - N_o^t$ particles without
          replacement from the particle pool and assign them
          to processor $o$;
```

the particles such that the particles are evenly distributed in the processors. The algorithm first scans all the processors to check their numbers of particles. If there are more than N/O particles in a processor, the algorithm randomly samples $N_o^t - N/O$ particles without replacement and puts them to a particle pool; otherwise, the algorithm randomly samples $N/O - N_o^t$ particles without replacement from the particle pool and assigns them to the processor. Note that the particle rebalance algorithm also runs in parallel, and each processor can interact with the particle pool independently.

B. M-Step: Road Network Decomposition

In the M-step, we update three groups of parameters: the initial congestion state probability π_l^s, the observation distribution function parameters P_l^s, and the transition probability matrix A_l. To update these parameters, the expected complete log-likelihood is maximized given the probability $z_{t,l}^s$ that each link l is in state s at time t and probability $q_{t,l}^{R_i,s}$ of link l to be in state s given that neighbors of link l are in states R_i at time $t-1$.

$$\max_{\mathbf{P},\mathbf{A}} \Lambda(\mathbf{Y},\mathbf{E}|\mathbf{Z},\mathbf{Q},\mathbf{P},\mathbf{A},\Pi): \begin{cases} \sum_{s=1}^{S} A_l(r,s)=1, & \forall l,r \\ A_l(r,s)\in[0,1], & \forall l,r,s \\ \sum_{s=1}^{S}\pi_{l,s}=1, & \forall l \\ \pi_{l,s}\in[0,1], & \forall l,s \end{cases} \quad (10)$$

Based on formulas (1)-(5), the expected complete log likelihood is as follows.

$$\begin{aligned} &lnP(\mathbf{Y},\mathbf{E}|\mathbf{Z},\mathbf{Q},\mathbf{P},\mathbf{A},\Pi) \\ &= \sum_{l=1}^{L}\sum_{s=1}^{S}\sum_{t=1}^{T} z_{t,l}^s\Big(\sum_{y_{t,l,i}\in y_{t,l}} ln(g_l^s(y_{t,l,i})) \\ &\quad + \sum_{e_{t,l,i}\in e_{t,l}} ln(f_l^s(e_{t,l,i}))\Big) \\ &\quad + \sum_{l=1}^{L}\sum_{t=2}^{T}\sum_{s=1}^{S}\sum_{i=1}^{S^{|N_l|}} q_{t,l}^{R_i,s} ln(A_l(R_i,s)) + \sum_{l=1}^{L}\sum_{s=1}^{S} z_{1,l}^s ln(\pi_{l,s}) \end{aligned} \quad (11)$$

We can simplify the computation of formula (11) in the following two ways. 1) One can see that formula (11) is comprised of three parts. Different parameters appear in different parts, and thus the three parts can be solved separately. 2) The optimization problem on the entire road network can be further decomposed into $S \times L$ smaller optimization problems with each one associated to a particular congestion state and road link of the network. Thus the optimization on formula (11) can be decomposed as optimizing the following three subtasks independently.

$$\max_{P_l^s} \sum_{t=1}^{T} z_{t,l}^s\Big(\sum_{y_{t,l,i}\in y_{t,l}} ln(g_l^s(y_{t,l,i})) + \sum_{e_{t,l,i}\in e_{t,l}} ln(f_l^s(e_{t,l,i}))\Big) \quad (12)$$

$$\max_{q_{l,s}^{R_i,s}} \sum_{t=2}^{T}\sum_{i=1}^{S^{|N_l|}} q_{t,l}^{R_i,s} ln(A_l(R_i,s)) \quad (13)$$

$$\max_{z_{1,l}^s} z_{1,l}^s ln(\pi_{l,s}) \quad (14)$$

Based on formula (12)-(14), it is not hard to derive that the parameters can be inferred by the following equations based on Lagrangian method.

$$\pi_l^s = \frac{\sum_{t=1}^{T} z_{t,l}^s}{\sum_{s=1}^{S}\sum_{t=1}^{T} z_{t,l}^s} \quad (15)$$

$$A_l(R_i,s) = \frac{\sum_{t=1}^{T} q_{t,l}^{R_i,s}}{\sum_{s=1}^{S}\sum_{t=1}^{T} q_{t,l}^{R_i,s}} \quad (16)$$

$$\mu_{l,s} = \frac{\sum_{t=1}^{T} z_{t,l}^s \sum_{i=1}^{I_{t,l}} y_{t,l,i}}{\sum_{t=1}^{T} I_{t,l}} \quad (17)$$

$$\sigma_{l,s}^2 = \frac{\sum_{t=1}^{T} z_{t,l}^s \sum_{i=1}^{I_{t,l}} (y_{t,l,i}-\mu_{l,s})^2}{\sum_{t=1}^{T} I_{t,l}} \quad (18)$$

$$f_l^s(e_{t,l,i}) = \frac{\sum_{t=1}^{T} z_{t,l}^s n_{t,l,i}}{\sum_{s=1}^{S}\sum_{t=1}^{T} z_{t,l}^s n_{t,l,i}} \quad (19)$$

$I_{t,l}$ is the number of the probe readings on the road link l in the time interval t. $\mu_{l,s}$ is the mean probe speed on link l associated with traffic state s. $\sigma_{l,s}^2$ is the variance of the probe speed. $n_{t,l,i}$ is the total number of traffic states for traffic event tweet m_i on road link l in time interval t. $f_l^s(e_{t,l,i})$ is the parameter for the multinomial distribution denoting the probability of observing traffic event $e_{t,l,i}$ on road link l associated with traffic state s.

VII. Evaluation

A. Experiment Setup

1) Real Dataset: The real dataset is introduced in Section IV.

2) Synthetic Dataset: To evaluate the speedup performance of the proposed algorithm, we also generate a group of synthetic datasets with much larger number of road links. We first set the number of the road links L and the number of time intervals T of the synthetic dataset. Then we generate a $L \times L$ road link matrix with each entry denoting the connection between two road links. For each road link, we randomly select several other road links as its neighbors and set the value of corresponding entires as 1, and the other entries are set to 0. Then for each time interval, we generate some GPS probe readings based on a Gaussian distribution and traffic event observations based on a multinomial distribution on each road link. As the synthetic datasets are only used to evaluate the efficiency of our model, the ground truth traffic states of the road links are not required in the experiments.

3) Ground Truth: Chicago Transit Authority (CTA) defines a 3-state traffic conditions in downtown Chicago by fully considering the traffic situations in urban area[2]. The 3 states are *heavy congestion*, *medium*, and *flow*. In this paper, we add two traffic states and use a 5-state traffic conditions: *heavy congestion*, *medium-heavy congestion*, *medium*, *light*, and *flow*. The corresponding traffic speeds of the 5 traffic states are 0-10, 10-15, 15-20, 20-25, and over 25 mph, respectively. Note that except for a vary few road links, speed on arterial roads of downtown Chicago is limited to 30 mph by ordinance. To quantitively distinguish the 5 states, we assign different values to different states. We assign larger values to worse traffic conditions and smaller values to better conditions. We first assign 1 to the traffic state *heavy congestion*, and set the value difference between two successive traffic states as 0.2. Thus the values for the remaining four states *medium-heavy*, *medium*, *light* and *flow* are 0.8, 0.6, 0.4, and 0.2, respectively. In the following part of this paper, we use the 5 traffic states defined by CTA as a measure of traffic conditions.

The traffic conditions are estimated based on more than 5 million GPS traces generated by more than 2,000 CTA public passenger buses from 11/25/2014 to 12/30/2014. As the real time GPS traces for some links are sparse, we also consider the historical average traffic speed for each road link in the last 3 years. Given a time interval t and a road link l, the traffic speed can be estimated as $speed_{t,l} = w\sum_{i=1}^{n}\frac{speed_{t,l,i}}{n} + (1-w)speed^{h}_{t,l}$, where $speed_{t,l,i}$ is the ith real time probe speed record, $speed^{h}_{t,l}$ is the historical speed, and w is a weight. In this paper we set w to 0.8. For simplicity, we can consider there are only two traffic states: congestion and normal. In the 2-state traffic condition system, we consider a road link is in congestion if the average speed is lower than 15 mph, and it is in normal state otherwise.

4) Competitive Methods: We compare PE_CHMM with the following baselines.

- **E_CHMM with probe and tweets observations [14].** E_ CHMM is our previously proposed model. The major difference is that E_CHMM uses sequential particle filtering to estimate parameters, and thus is less efficient.
- **CHMM with probe observations (P_CHMM) [1].** To study whether and to what extend the performance can be improved by incorporating the Twitter data, we compare the proposed mode with P_CHMM which only utilizes sparse GPS probe data to estimate traffic congestions on an arterial network.
- **CTCE model [6].** CTCE is a recently proposed traffic congestion estimation model with social media as the primary data source. Instead of utilizing CHMM, CTCE models the traffic information on the road links as matrices and tensors, and applies matrix factorization technique to address the estimation task.
- **CHMM with tweet observations (T_CHMM).** In this model, only the tweet observations are available. We use this baseline to evaluate the performance of the CHMM model with the tweet observations only.
- **Linear combination of two CHMMs (LC_CHMM).** We use two CHMMs with each one associated with one type of observation to estimate the traffic conditions separately. Assuming the estimation results of the two models are $\mathbf{Z}_1$ and $\mathbf{Z}_2$, the final estimation is the linear combination of the two results, $\mathbf{Z} = \alpha\mathbf{Z_1} + (1-\alpha)\mathbf{Z_2}$.

[2]https://data.cityofchicago.org/api/assets/3F039704-BD76-4E6E-8E42-5F2BB01F0AF8

B. Evaluation Results

1) Evaluation Metrics: We use the following metrics to evaluate the performance of the proposed model: *accuracy*, *precision@k*, and *Root Mean Square Error (RMSE)*. We use *accuracy* to evaluate the estimation performance on all the road links in all the time intervals. Normally, in a particular time interval only a small number of road links are in congestion. Thus to better evaluate whether the proposed model can give good estimations on the road links that are very likely to occur congestion, we also use *precision@k* as a metric. We first rank the congestion probabilities $z^{s}_{t,l}$ for all the road links in all the time intervals. Then we only consider the road links with the *top-k* congestion probabilities are in congestion. To further evaluate the performance of the model on the above mentioned 5-state traffic conditions, we use the *Root Mean Square Error (RMSE)* as the evaluation metric: $RMSE = \sqrt{\frac{\sum_{t,l}(z_{t,l}-\hat{z}_{t,l})^2}{L*T}}$, where $z_{t,l}$ is the estimated traffic state of link l in time interval t, and $\hat{z}_{t,l}$ is the ground truth.

2) Evaluation With precision@k: Table IV shows the average *precision@k* of different methods over various k with the 2-state traffic condition. As the traffic conditions on weekdays and weekends are different, we present the results by weekday and weekend separately. We run the algorithm and calculate the *precision@k* on each day, and then average the results. The best results are highlighted in bold font. One can see that E_CHMM performs best among all the methods. The performance difference between E_CHMM and the parallel algorithm PE_CHMM is not significant. *precision@k* of PE_CHMM is lower than E_CHMM by only around 0.015 on weekday and 0.01 on weekend. LC_CHMM model is inferior to E_CHMM and PE_CHMM, but better than the other methods. It is no surprise that T_CHMM presents the worst performance among all the methods. One can infer that the traffic event tweets are too sparse for the T_CHMM model to get an accurate estimation. P_CHMM can achieve comparable performance with CTCE, but both methods are inferior to the three methods LC_CHMM, E_CHMM, and PE_CHMM. Compared to P_CHMM, E_CHMM and PE_CHMM improve *precision@k* by around 6%, which shows that incorporating Twitter data does help to improve the estimation accuracy.

3) Performance Evaluation in Rush Hours: People concern more on the traffic conditions in rush hours of a day. Thus we

TABLE IV
AVERAGE PRECISION @k OF DIFFERENT METHODS

Average Precision @k on Weekday									
	top-10	top-20	top-30	top-50	top-100	top-150	top-200	top-250	top-300
P_CHMM	0.870	0.850	0.845	0.832	0.812	0.792	0.773	0.744	0.732
T_CHMM	0.690	0.665	0.624	0.613	0.585	0.532	0.473	0.464	0.452
LC_CHMM	0.890	0.850	0.852	0.842	0.832	0.817	0.792	0.784	0.775
CTCE	0.870	0.860	0.853	0.840	0.824	0.816	0.718	0.705	0.712
E_CHMM	**0.920**	**0.900**	**0.894**	**0.887**	**0.864**	**0.826**	**0.810**	**0.795**	**0.786**
PE_CHMM	0.910	0.885	0.876	0.885	0.860	0.812	0.789	0.774	0.757
Average Precision @k on Weekend									
	top-10	top-20	top-30	top-50	top-100	top-150	top-200	top-250	top-300
P_CHMM	0.860	0.850	0.843	0.822	0.816	0.766	0.752	0.745	0.722
T_CHMM	0.660	0.650	0.612	0.625	0.570	0.464	0.453	0.415	0.425
LC_CHMM	0.870	0.850	0.845	0.825	0.820	0.812	0.805	0.785	0.768
CTCE	0.850	0.834	0.820	0.820	0.754	0.715	0.678	0.654	0.644
E_CHMM	**0.910**	**0.900**	**0.868**	**0.852**	**0.844**	**0.820**	**0.812**	**0.794**	**0.783**
PE_CHMM	0.900	0.894	0.854	0.835	0.836	0.816	0.810	0.786	0.765

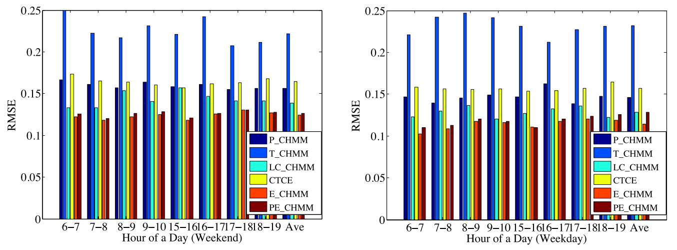
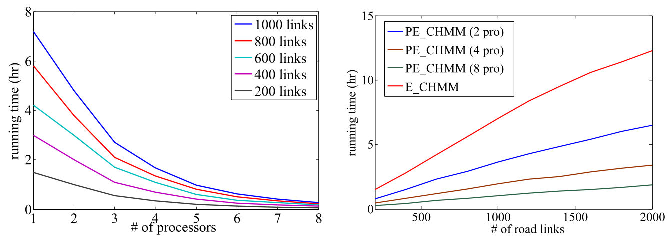


Fig. 3. RMSE comparison of various methods in rush hours.



(a) # of processors *vs* running time (b) # of road links *vs* running time

Fig. 4. Efficiency evaluation on running time. (a) Running time of PE_CHMM with difference number of subtask processors. (b) Running time of PE_CHMM on different size of road links.

also evaluate the performance of the different models in rush hours. Fig. 3 shows the experiment results in the rush hours of 6:00-10:00 and 15:00-17:00 on weekday and weekend, respectively. One can see that the RMSE of PE_CHMM and E_CHMM are mostly lower than the other four methods. The performance of PE_CHMM is comparable to E_CHMM with its RMSE higher than E_CHMM by 0.05 on weekday and 0.03 on weekend on average. The result shows that PE_CHMM is a good approximation of E_CHMM and can achieve comparable estimation performance. The performance of T_CHMM is the worst among all the methods, which is consistent with the previous experiment results. LC_CHMM is consistently better than P_CHMM and CTCE, which means incorporating traffic event information from tweets does help us better estimate traffic conditions. However, LC_CHMM is inferior to E_CHMM and PE_CHMM. Thus we can conclude that although LC_CHMM is a straightforward method combining the two types of data, the performance it achieves is less desirable. By comparing the results on weekday and weekend, one can see that on average the RMSE of various methods on weekday is larger than that on weekend. This finding also verifies that traffic conditions on weekend is harder to estimate than on weekday due to the irregular travel plans of people.

C. Efficiency Evaluation

To evaluate the efficiency of the proposed algorithm, we compare the computational time of PE_CHMM with different numbers of processors and different numbers of road links. Fig. 4(a) shows the running time of PE_CHMM with different number of subtask processors on different numbers of road links. One can see that, first the E_CHMM algorithm (PE_CHMM with only one subtask processor) is time consuming. It takes more than 7 hours for the E_CHMM to give an estimation result for a road network with 1000 road links. This is unacceptable for a real application scenario where a timely estimation result is essential. Second, the running time of the algorithm increases with the increase of the road link numbers. More road links mean more variables to be estimated and thus need more computational time. With the increase of the subtask processors, the running time decreases significantly showing the efficiency of the proposed parallel algorithm. It takes 7 hours for only 1 processor to estimate traffic conditions on a road network with 1000 road links, but less than one hour for 8 processors to conduct the same task. It implies that the proposed PE_CHMM can effectively and evenly distribute the entire computation to different processors. However, when there are too many subtask processors, the efficiency improvement becomes less significant. As show in Fig. 4(a), the decrease trend of the curves become smooth when more than 4 processors are used. This is mainly because that more processors mean more data interactions among the processors, which may consume more additional time for particles rebalance.

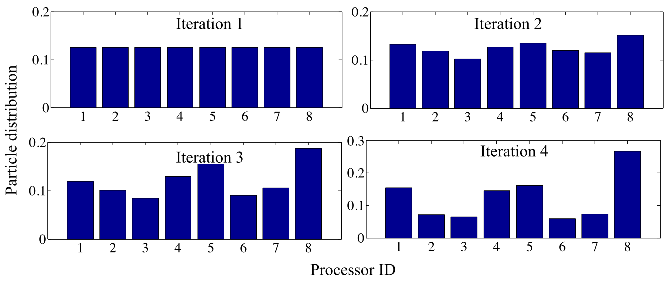


Fig. 5. Particle distribution changes over iteration without applying the particle rebalancing strategy.

To further evaluate the efficiency of PE_CHMM on larger road networks. We run the algorithms on a synthetic road network with 2,000 road links. Fig. 4(b) shows the increase trend of the running time of the algorithms with the increase of the road links. From this figure, one can see that the running time curves all show a linear increase trend. PE_CHMM with 8 subtask processors achieves about 6 times speedup compared to E_CHMM. On the road network with 2,000 road links, E_CHMM needs more than 12 hours to complete the estimation while PE_CHMM needs only about 2 hours. This result verifies that our proposed paralleled algorithm PE_CHMM can achieve nearly a linear speedup associated with the processor number.

D. Effect of the Particles Rebalancing Strategy

A key step of the proposed parallel solution of particle filtering is that we apply a rebalancing strategy after each iteration to make the new sampled particles evenly distributed on the subtask processors. In this subsection we conduct experiments to investigate the effect of the rebalancing strategy on the speedup performance of the algorithm.

We first study how the particle distribution on multiple processes changes over iterations if the particle rebalancing strategy is not applied. Fig. 5 illustrates the particle distribution changes over 4 iterations on the real dataset by using 8 subtask processors. One can see that at first the particles are evenly distributed, and each processor is assigned the same size of particles. From the second iteration on, some processors are assigned more particles while some are assigned less, leading to an unbalanced distribution of the particles on the 8 subtask processors. In Algorithm 2, in each iteration the master processor needs all the subtask processors to submit their local weights and then return the global weight. Thus if the particles are not evenly distributed, the local weight computation time varies with the particle size of the processor, and the master processor has to wait for all the subtask processors to submit their local weights. The subtask processor with the larger number of particles will hurt the speedup of the algorithm as it need longer time to compute its local weight and resample the particles. Fig. 5 shows that without the particle rebalancing strategy, the particle distribution tends to become skewed.

Table V shows the speedup comparison between PE_CHMM with and without the particle rebalancing strategy on both real and synthetic datasets. One can see that the speedup of the algorithm decreases with the increase of iteration on both datasets. This is because the distribution of particles becomes more skewed as the iteration increases, and more skewed particle distribution leads to worse speedup performance. By using 8 subtask processors, the overall speedup of PE_CHMM with rebalancing strategy on the two datasets is 7.54 and 7.44, while the numbers are 6.06 and 5.7 for the algorithm without rebalancing strategy. It demonstrates that the proposed particle rebalancing strategy can effective improve the speedup performance.

TABLE V
SPEEDUP COMPARISON WITH AND WITHOUT PARTICLE REBALANCE (8 SUBTASK PROCESSORS)

Road Network of Chicago						
	Ite_1	Ite_2	Ite_3	Ite_4	Ite_5	Ave
Speedup(w.)	7.6	7.5	7.6	7.6	7.4	7.54
Time(min)	47.6	45.2	44.3	45.7	42.1	45
Speedup(w.o.)	7.6	7.1	6.4	5	4.2	6.06
Time(min)	47.6	50.9	56.5	72.4	86.1	62.4
Synthetic Data with 2,000 Road Links						
	Ite_1	Ite_2	Ite_3	Ite_4	Ite_5	Ave
Speedup(w.)	7.7	7.4	7.5	7.4	7.2	7.44
Time(min)	72.6	75.7	74.3	74.7	71.1	88.7
Speedup(w.o.)	7.7	6.8	6.2	4.7	3.1	5.7
Time(min)	72.6	82.2	90.2	118.9	180.3	108.8

VIII. CONCLUSION AND FUTURE WORK

In this paper, we proposed an effective and efficient framework PE_CHMM to integrate two types of data, traffic event tweets and GPS probe readings to enhance citywide traffic congestion estimation. As GPS probe data are sparse and relying them only can hardly achieve a promising estimation result, we first extensively collect traffic related tweets and extract various traffic events including traffic jam, accident, and road construction. To utilize the extracted traffic events and model their potential impact on traffic congestions, we extended the traditional Coupled Hidden Markov Model by considering the traffic event tweets as a new type of observations following multinomial distribution. To address the high computational challenge of solving CHMM, we further proposed a parallel algorithm which can effectively decompose the entire computation task into several subtasks and solve the subtasks in parallel on multiple processors. Experimental results on both real dataset and synthetic dataset showed the efficiency and effectiveness of the proposed PE_CHMM.

One potential research direction in the future is that we will study on the impact of various traffic events on traffic conditions. For example, how long and to what extent a car accident will affect the traffic congestion? Based on this study, we will further investigate how to use real time traffic event information to perform traffic condition prediction continuously. Another research issuer we are interested in is detecting traffic event based on GPS probe data.

REFERENCES

[1] R. Herring, A. Hofleitner, P. Abbeel, and A. Bayen, "Estimating arterial traffic conditions using sparse probe data," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Sep. 2010, pp. 929–936.

[2] J. Shang, Y. Zheng, W. Tong, E. Chang, and Y. Yu, "Inferring gas consumption and pollution emission of vehicles throughout a city," in *Proc. ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2014, pp. 1027–1036.

[3] Y. Wang, Y. Zhu, Z. He, Y. Yue, and Q. Li, "Challenges and opportunities in exploiting large-scale GPS probe data," HP Laboratories, Palo Alto, CA, USA, Tech. Rep. HPL-2011-109, 2011.

[4] S. K. Endarnoto, S. Pradipta, A. S. Nugroho, and J. Purnama, "Traffic condition information extraction & visualization from social media twitter for Android mobile application," in *Proc. Int. Conf. Electr. Eng. Inform.*, Jul. 2011, pp. 1–4.

[5] S. Bregman, *Uses of Social Media in Public Transportation*. Washington, DC, USA: Transportation Research Board, 2012.

[6] S. Wang, L. He, L. Stenneth, P. S. Yu, and Z. Li, "Citywide traffic congestion estimation with social media," in *Proc. ACM SIGSPATIAL GIS*, 2015, Art. no. 34.

[7] S. S. Ribeiro, Jr., C. A. Davis, Jr., D. D. R. Oliveira, W. Meira, Jr., T. S. Gonçalves, and G. L. Pappa, "Traffic observatory: A system to detect and locate traffic events and conditions using twitter," in *Proc. ACM SIGSPATIAL GIS LBSN*, 2012, pp. 5–11.

[8] E. M. Daly, F. Lecue, and V. Bicer, "Westland row why so slow?: Fusing social media and linked data sources for understanding real-time traffic conditions," in *Proc. Int. Conf. Intell. User Interfaces*, 2013, pp. 203–212.

[9] M. Liu, K. Fu, C.-T. Lu, G. Chen, and H. Wang, "A search and summary application for traffic events detection based on twitter data," in *Proc. ACM SIGSPATIAL GIS*, 2014, pp. 549–552.

[10] H. Sayyadi, M. Hurst, and A. Maykov, "Event detection and tracking in social streams," in *Proc. Int. AAAI Conf. Web Social Media*, 2009, pp. 311–314.

[11] S. Wang *et al.*, "Computing urban traffic congestions by incorporating sparse GPS probe data and social media data," *ACM Trans. Inf. Syst.*, vol. 35, no. 4, 2017, Art. no. 40.

[12] S. Wang, L. He, L. Stenneth, P. S. Yu, Z. Li, and Z. Huang, "Estimating urban traffic congestions with multi-sourced data," in *Proc. IEEE Int. Conf. Mobie Data Manage.*, Jun. 2016, pp. 82–91.

[13] P.-T. Chen, F. Chen, and Z. Qiang, "Road traffic congestion monitoring in social media with hinge-loss Markov random fields," in *Proc. Int. Conf. Data Mining*, Dec. 2014, pp. 80–89.

[14] S. Wang, F. Li, L. Stenneth, and P. S. Yu, "Enhancing traffic congestion estimation with social media by coupled hidden Markov model," in *Proc. ECMLPKDD*, 2016, pp. 247–264.

[15] A. Hofleitner, R. Herring, P. Abbeel, and A. Bayen, "Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1679–1693, Dec. 2012.

[16] L. Muñoz, X. Sun, R. Horowitz, and L. Alvarez, "Traffic density estimation with the cell transmission model," in *Proc. Amer. Control Conf.*, Jun. 2003, pp. 3750–3755.

[17] D. Helbing, "Traffic and related self-driven many-particle systems," *Rev. Modern Phys.*, vol. 73, no. 4, pp. 1067–1141, 2001.

[18] F. Porikli and X. Li, "Traffic congestion estimation using HMM models without vehicle tracking," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2004, pp. 188–193.

[19] W. Pattara-Atikom, P. Pongpaibool, and S. Thajchayapong, "Estimating road traffic congestion using vehicle velocity," in *Proc. ITS Telecommun.*, Jun. 2006, pp. 1001–1004.

[20] J. Yuan *et al.*, "T-drive: Driving directions based on taxi trajectories," in *Proc. ACM SIGSPATIAL GIS*, 2010, pp. 99–108.

[21] M. R. Gahrooei and D. B. Work, "Inferring traffic signal phases from turning movement counters using hidden Markov models," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 91–101, Feb. 2015.

[22] C. de Fabritiis, R. Ragona, and G. Valenti, "Traffic estimation and prediction based on real time floating car data," in *Proc. IEEE Conf. Intell. Transp. Syst.*, Oct. 2008, pp. 197–203.

[23] R. Wang, D. B. Work, and R. Sowers, "Multiple model particle filter for traffic estimation and incident detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3461–3470, Dec. 2016.

[24] N. Polson and V. Sokolov, "Bayesian particle tracking of traffic flows," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 345–356, Feb. 2018.

[25] N. Polson and V. Sokolov, "Bayesian analysis of traffic flow on interstate I-55: The LWR model," *Ann. Appl. Statist.*, vol. 9, no. 4, pp. 1864–1888, 2015.

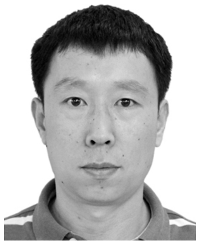
[26] L. Mihaylova, A. Hegyi, A. Gning, and R. K. Boel, "Parallelized particle and Gaussian sum particle filters for large-scale freeway traffic systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 36–48, Mar. 2012.

[27] S. Wang, Z. Li, W. Chao, and Q. Cao, "Applying adaptive over-sampling technique based on data density and cost-sensitive SVM to imbalanced learning," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2012, pp. 1–8.

[28] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura, "Traffic state estimation on highway: A comprehensive survey," *Annu. Rev. Control*, vol. 43, pp. 128–151, 2017.

[29] D. Schrank, B. Eisele, and T. Lomax, "TTI's 2012 urban mobility report," Texas A&M Transp. Inst., Texas A&M Univ. Syst., College Station, TX, USA, Tech. Rep., Dec. 2012.

[30] O. Brun, V. Teuliere, and J.-M. Garcia, "Parallel particle filtering," *J. Parallel Distrib. Comput.*, vol. 62, no. 7, pp. 1186–1202, 2002.



Senzhang Wang received the M.Sc. degree from Southeast University, Nanjing, China, in 2009, and the Ph.D. degree from Beihang University, Beijing, China, in 2016. He is currently an Associate Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. He has authored over 50 conference and journal papers in computer science, including ACM SIGKDD, AAAI, ACM SIGSPATIAL, ECML-PKDD, ACM TIST, KAIS, ITS, and the IEEE TMM. His main research focus is on data mining, social computing, and urban computing.

Xiaoming Zhang received the B.Sc. and M.Sc. degrees in computer science and technology from the National University of Defense Technology, China, in 2003 and 2007, respectively, and the Ph.D. degree in computer science from Beihang University in 2012. Since 2012, he has been a Lecturer. He is currently with the School of Computer, Beihang University. He has authored over 30 papers, in journals and conferences such as the IEEE TRANSACTION ON MULTIMEDIA, the IEEE TRANSACTION ON CYBERNETICS, WWWJ, *Neurocomputing*, JIIS, *Signal Processing*, IJCAI 2015, ICMR 2015, SDM 2014, WAIM 2014, AAAI 2013, and Coling 2012. His major interests are social media analysis, image tagging, and text mining.

Fengxiang Li received the bachelor's degree from Peking University, Beijing, China, in 2016. He is currently pursuing the master's degree with the College of Computer and Information Science, Northeastern University. His research focus is on data mining, social computing, and geographical big data.

Philip S. Yu (M'78–F'93) received the B.S. degree in electrical engineering from National Taiwan University, the M.S. and Ph.D. degrees in electrical engineering from Stanford University, and the M.B.A. degree from New York University. He was with IBM, where he was the Manager of the Software Tools and Techniques Department, Watson Research Center. He is currently a Distinguished Professor in computer science with The University of Illinois at Chicago, where he is also the Wexler Chair in Information Technology. He has authored over 1100 papers in refereed journals and conferences. He holds or has applied for over 300 U.S. patents. His research interest is on big data, including data mining, data stream, database, and privacy. He is a fellow of the ACM. He was a recipient of the ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion, and anonymization of big data, the IEEE Computer Society's 2013 Technical Achievement Award for "pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining and anonymization of big data," the IEEE International Conference on Data Mining (ICDM) 2013 10-Year Highest-Impact Paper Award, and the EDBT Test of Time Award in 2014. He received the Research Contributions Award from the IEEE ICDM in 2003 for his pioneering contributions to the field of data mining. He was the Editor-in-Chief of the *ACM Transactions on Knowledge Discovery from Data* from 2011 to 2017 and the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING from 2001 to 2004.

Zhiqiu Huang received the B.S. and M.S. degrees from the National University of Defense Technology, China, and the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics, China. He is currently a Professor and a Tutor of Ph.D. candidates with the Nanjing University of Aeronautics and Astronautics. His major research interests include formal method, software engineering, and cloud computing.