Real-Time Large-Scale Map Matching Using Mobile Phone Data

ESSAM ALGIZAWY, Department of Computer Science and Engineering,
Egypt-Japan University of Science and Technology
TETSUJI OGAWA, Department of Computer Science, Waseda University
AHMED EL-MAHDY, Department of Computer Science and Engineering, Egypt-Japan University of Science and Technology and on-Leave from Department of Computer and Systems Engineering, Alexandria University

With the wide spread use of mobile phones, cellular mobile big data is becoming an important resource that provides a wealth of information with almost no cost. However, the data generally suffers from relatively high spatial granularity, limiting the scope of its application. In this article, we consider, for the first time, the utility of actual mobile big data for map matching allowing for "microscopic" level traffic analysis. The state-of-the-art in map matching generally targets GPS data, which provides far denser sampling and higher location resolution than the mobile data. Our approach extends the typical Hidden-Markov model used in map matching to accommodate for highly sparse location trajectories, exploit the large mobile data volume to learn the model parameters, and exploit the sparsity of the data to provide for real-time Viterbi processing. We study an actual, anonymised mobile trajectories data set of the city of Dakar, Senegal, spanning a year, and generate a corresponding road-level traffic density, at an hourly granularity, for each mobile trajectory. We observed a relatively high correlation between the generated traffic intensities and corresponding values obtained by the gravity and equilibrium models typically used in mobility analysis, indicating the utility of the approach as an alternative means for traffic analysis.

CCS Concepts: • **Human-centered computing** → **Empirical studies in ubiquitous and mobile computing**; *Mobile phones*; • **Applied computing** → **Transportation**; Forecasting; • **Mathematics of computing** → *Kalman filters and hidden Markov models*; Matchings and factors; • **Information systems** → *Spatial-temporal systems*; Data mining; • **Theory of computation** → *Streaming, sublinear and near linear time algorithms*; • **Computing methodologies** → *Unsupervised learning*; *Maximum likelihood modeling*; • **Software and its engineering** → *Software performance*

Additional Key Words and Phrases: Mobile big data, cellular duration records, fine-grained spatial tracking, adaptive HMM, low cost

ACM Reference Format:
Essam Algizawy, Tetsuji Ogawa, and Ahmed El-Mahdy. 2017. Real-time large-scale map matching using mobile phone data. ACM Trans. Knowl. Discov. Data 11, 4, Article 52 (July 2017), 38 pages.
DOI: http://dx.doi.org/10.1145/3046945

Authors' addresses: E. Algizawy, Parallel Computing Lab (PCL), Computer Science and Engineering Department, Egypt-Japan University for Science and Technology(E-JUST) - P.O. Box 179, New Borg El-Arab City Postal Code 21934, Alexandria, Egypt; email: essam.algizawy@ejust.edu.eg; T. Ogawa, Waseda University - 6-1, Nishiwaseda 1-Chōme - Shinjuku, Tokyo - Japan 169-0051; email: ogawa@pcl.cs.waseda.ac.jp; A. El-Mahdy, Parallel Computing Lab (PCL), Computer Science and Engineering Department, Egypt-Japan University for Science and Technology(E-JUST) - P.O. Box 179, New Borg El-Arab City Postal Code 21934, Alexandria, Egypt and Computers and Systems Engineering, Faculty of Engineering, Alexandria University, Postcode 21544, Alexandria, Egypt; email: ahmed.elmahdy@ejust.edu.eg.

DOI: http://dx.doi.org/10.1145/3046945

1. INTRODUCTION

According to the International Telecommunication Union, mobile phones have reached an unprecedented global penetration level of 95.5%, and 90.2% for the developing world [itu 2014]. Being mobile, and personal, the phones provide a wealth of big data information, enabling urban mobility sensing at virtually no cost at the global level [Calabrese et al. 2014; Naboulsi et al. 2015].

Map-matching is an important problem for intelligent transportation systems; its applications include traffic routing, network analysis, control, and general network improvements [Zheng 2015]. Performing map-matching on mobile big data has five main issues:

(1) Non-uniform, sparse time, and space sampling: The non-uniformity and sparsity happen as a sample is determined when a user receives/initiates a call/SMS, which is highly irregular and infrequent (e.g., less than 20% of the users generate these samples in an hour, for the Senegal D4D dataset [de Montjoye et al. 2014]).
(2) Large location error: This is mainly due to only recording the cell tower ID with each sample. The location within the tower coverage zone cannot be determined, and is usually in kilometer scale.
(3) Noisy data: The data suffers from oscillations among cell towers, as many factors affect the choice of cell tower including cell network load and environment.
(4) Large-data volume: The data requires high computational power. For example, for the Senegal D4D dataset, there is in the order of 100 thousands trips per day for 300 thousands users, for one year; presenting a total of 300 million trips for the whole year.
(5) No trip meta-data: The data does not include information about which samples identify trips start/stop points.

Existing map-matching methods generally rely on continuity for modelling the corresponding road network transitions among samples. This is generally exploited by the Hidden-Markov model (HMM) [Newson and Krumm 2009]. Such continuity is lost with increasing the sampling duration; therefore, low sampling techniques usually rely on some extra information, such as signal strength [Thiagarajan et al. 2011; Becker et al. 2011], map hints [Mohamed et al. 2014, 2016], and history [Zheng et al. 2012], while assuming uniform sampling (raw or interpolated samples). Schulze et al. [2015] consider both low sampling and non-uniform sampling for a mobile data set of 2,249 traces. However, the samples are more frequent than a typical mobile dataset, as it includes handover events. Also, that initial work does not consider time optimizing the map-matching process, resulting in more than 5s processing for a single trajectory.

Map-matching mobile big data is, therefore, a highly unexplored area in the literature; existing methods are generally trajectory oriented, where map-matching is done on a single trajectory level, and do not consider the data across trajectories to adapt the mapping. In this article, we attempt to consider mining big data to improve map-matching. In particular, we extend the HMM, so it adapts to the aggregate flow patterns in the mobile big data. Moreover, we consider only "exit" road segments (roads at cell boundaries) to represent states, therefore, providing for the sought continuity for the HMM and spatial regularity, while decreasing the state space. Additionally, we limit transitions among exit segments that span a fixed number of zones; thereby accommodating for a practical number of missed observations, while decreasing the number of transitions among the states.

Finally, we optimize the Viterbi decoding algorithm exploiting the sparsity in terms of emissions and transitions per state. The optimization delays writing a corresponding zero score until it is read in a later Viterbi stage, which is generally infrequent due to

the limited connectivity, thereby avoiding most of the zero write operations. Therefore, the complexity of the Viterbi decoding algorithm reduces to linear time and space complexities instead of the typical quadratic complexities. This provides for better scalability, which is especially important for processing big mobile data.

In summary, the article has the following contributions:

—Propose an adaptive probabilistic HMM-based method that utilizes trip antenna coverage zones to learn the road-level trajectories patterns (model parameters) and infer corresponding roads for a given trip antenna sequence (states).
—Exploit the sparsity in model parameters for a typical mobile call data record (CDR) data set and propose a fast, real-time Viterbi decoder with linear time and space complexities.
—Implement our proposed approach and evaluate its accuracy using real challenging positioning (GPS and Cellular-based) data traces to assess the accuracy of individual trajectories.
—Validate the method utilizing both gravity and equilibrium transportation models, to check the quality of extracted origin/destination (OD) trips and routes, respectively. The validation is done over a real CDR data set provided by Orange through the Data for Development (D4D) challenge, for the city of Dakar, Senegal [de Montjoye et al. 2014].

The rest of the article is organized as follows: Related work is presented in Section 2; Section 3 analyses the set of considered mobile phone CDRs; Section 4 presents our map-matching/traffic monitoring system based on HMM; Section 5 discusses our performance optimization that exploits sparsity in HMM's transition probabilities to achieve linear time and space complexities; Section 6 provides experimental validation, including both GPS subsampled trajectories and gravity/equilibrium models; Finally, Section 7 concludes the article and discusses future work.

2. RELATED WORK

2.1. Map-Matching Approaches

Mobile phone data is showing a great potential in large-scale sensing of urban areas, including sensing human mobility [Calabrese et al. 2014; Caceres et al. 2012]. Notable work includes sensing home, work, commercial, and business locations [Becker et al. 2013; Ahas et al. 2010; Ratti et al. 2006; Xiao-Yong et al. 2011; Calabrese et al. 2011; Ahas et al. 2010; Gonzalez et al. 2008; Girardin et al. 2008]. As reported in the literature, CDRs have been used in several research fields, and their use can be categorised into three main approaches [Calabrese et al. 2014; Pucci et al. 2015]:

—Mobility behaviour study of a specific group of people using sampling and mining individual trajectories: This use requires substantial processing power and have concerns about the individual privacy [de Montjoye et al. 2013; De Jonge et al. 2012; Isaacman et al. 2011; Ahas et al. 2010].
However, the CDRs still cannot precisely associate identified locations for a particular cellphone user [Isaacman et al. 2012]. Nevertheless, worries may arise, in general, from using cell phone data if cellphone user's movement is made available for a beneficiaries third parties [Calabrese et al. 2014].
—Study the mobility patterns from a previously anonymised data provided by a phone carrier: This approach is antithesis of the previous one; it shifts the focus from individual level tracking and directing it to urban dynamics relating frequently visited places or the geometrical patterns of urban mobility [Wang et al. 2012; Calabrese et al. 2011; Thiagarajan et al. 2011; Becker et al. 2011; Gonzales et al. 2008; Gonzalez et al. 2008].

—Mapping of a geo-referenced and aggregated mobile phone data for land use studies: It focuses on real-time geographical referencing of the traffic between cell phone towers, which reflects the urban spacing (density mapping) by people as time and space dependent relationships [Traag et al. 2011].

In this article, we focus on studying dynamics of the traffic by estimating road usage patterns that originate in the second category; Table I summarises related approaches as explained below.

Becker et al. [2011] have attempted to identify the route and estimate the traffic intensity from cellular handoff patterns and signal strength; 15 routes were identified using nearest-neighbour classifier based on earth mover distance (EMD) [Rubner et al. 2000] or logarithmic signal-to-noise ratio (SNR)-based classifier. The former system is not feasible to apply to large-scale data as both nearest-neighbour-based classification and EMD calculation are significantly time-consuming, and the latter does not assume to efficiently conduct time evolution of the systems, since it is memoryless (i.e., does not use any models).

Newson and Krumm [2009] introduce an HMM-based map matching algorithm for finding the most probable routes represented by a sequence of GPS time stamps and Lat/Long pairs. Their approach is close to ours; however, the considered GPS data has different characteristics from mobile phone data in terms of spatial accuracy and regularity and density of sampling. Both of which are low for mobile data and thus affects the formulation; our HMM model differs in state definition (boundary exist points), transitions (allowing for further transitions), and learning the model parameters.

Krumm et al. [2007] applied HMMs to the map-matching problem. Thiagarajan et al. [2009] developed *VTrack* to address the same problem, which carried out mobile phone localisation using WiFi and GPS, followed by mapping the location estimates onto the road segments using HMMs. They have also developed *CTrack* [Thiagarajan et al. 2011], which is a well-organised system for GSM-based map-matching and designed on the basis of a comparable concept to our work. *CTrack* can match a set of GSM fingerprints to road segments using HMMs with an accuracy of about 75%. It should be noted that the main focus of *CTrack* is accurate trajectory mapping of each trip, while our focus is not only trajectory mapping but also efficient system adaptation, that is, development of the framework for efficient time evolution of the system. In *CTrack*, the emission and state transition scores from HMM are heuristic and non-parametric (i.e., explicitly using data to calculate the scores). This system, therefore, could need troublesome work such as data selection in the big-data tasks, making system adaptation difficult. In contrast, our system calculates the emission and state transition scores from probabilistic density functions trained using accumulated statistics to achieve efficient time evolution of the system. Especially, in our method, intensity map generation and model parameter estimation are fully incorporated into the time evolution of the road and traffic network model based on segmental K-means clustering algorithm.

In addition, the systems provided by Becker et al. [2011] and Thiagarajan et al. [2011] (*CTrack*) assume rich information such as high-resolution observation sequences (e.g., one sample per second and one sample per 10m) that consist of the base transceiver station (BTS) fingerprints and signal strengths with time stamps. *CTrack* can utilise further information such as an accelerometer and a compass to improve the accuracy. In contrast, our system is applicable even under more restricted and challenging condition assumed in the D4D Orange Challenge task, that is, only very sparse BTS fingerprints with non-constant intervals of time stamps are provided. Our HMM-based system developed in the present work is designed to overcome this severe requirements.

Berlingerio et al. [2013] have developed *AllABoard*, which estimates origin/destination flows and traffic volumes under the constraints in the D4D Orange Challenge, to optimize public transport. *AllABoard* consists of: (1) stop extraction conducted

Table I. Urban Roads Map-matching Approaches Comparison

Method	Problem	Dataset	Feature parameter	Sampling rate	Exact trajectory	Power	model	model parameter	trajectory mapping/ classification	model update	ground truth	validation
Newson and Krumm [2009]	GPS map matching	small scale (1 second sampling for 80 mile (7531 sample/2 hours) + larger period simulated data)	GPS fingerprints, time stamps	sparse	yes	high	HMM	zero-mean Gaussian distribution	Viterbi search	N/A	yes	precision and recall
Thiagarajan et al. [2011]	trip trajectory mapping	small scale (126 hours of 15 G1 mobile and for 4 months from 1 Nexus mobile)	BTS fingerprints, time stamps, signal strength (up to 7 BTSs per location); sensor hints (accel, compass)	high resolution (sample per second)	yes	low	HMM	heuristic (non-parametric)	Viterbi search	non-parametric	yes	precision and recall
Becker et al. [2011]	route identification and traffic intensity estimation	small scale (8 months; 15 routes; only 8 drives for each route)	BTS fingerprints, duration, signal strength is required	high resolution (sample per 10m)	yes	high	Nearest neighbour (NN)	NN (non parametric)	NN-based classifier (earth mover distance)+log SNR-based classifier	N/A (non parametric)	yes	classification accuracy
Berlingerio et al. [2013]	trip trajectory mapping (coarse grained) and traffic intensity estimation	D4D Orange Challenge 2013	BTS fingerprints, time stamps	very sparse (with missing observations)	no, only Sequence of visited places	low	N/A	N/A	Linear optimization	N/A	no	correlation with gravity model
Chen et al. [2014]	map-mapping low-frequency GPS floating car data (FCD) onto the road network	large-scale (10,000 taxis in Wuhan, China)	time stamped long/lat records	low sampling rate (sample per minute)	yes	high (continually computed shortest paths)	N/A	N/A	multi-criteria dynamic programming map-matching (Dijkstra-based search)	no	yes	accuracy comparison with GPS tracks using accuracy ratio of points matched (ARP)

(Continued)

Table I. Continued

Method	Problem	Dataset	Feature parameter	Sampling rate	Exact trajectory	Power	model	model parameter	trajectory mapping/ classification	model update	ground truth	validation
Mohamed et al. [2014, 2016]	cellular-based accurate real-time map matching	small scale(400km car driving - two cities Egypt using 4 mobile devices)	BTS fingerprints, an estimate of localization error point by point	sparse (location update on cell-tower change occur (2minutes on average))	yes	high (0.58msec per observation)	HMM	Gaussian distribution	online Viterbi search	no	yes	accuracy comparison with GPS (1sec sampling rate) tracks using matched sequences precision and recall
Aly and Youssef [2015]	HMM-based map matching challenging position trajectories	small scale(150km car driving - 2minutes sampling rate using 3 android mobiles)	inertial sensors, sparse noisy and coarse-grained position data (Cellular and GPS traces)	high (2 minutes sampling rate)	yes	low	HMM	Gaussian distribution	Viterbi decoder	no	yes	accuracy comparison with GPS tracks using matched sequences precision, recall and F-measures
Schulze et al. [2015]	trip trajectory mapping (fine-grained)	BTS fingerprints, time stamps for 2294 trajectories with sampling rate 260 sec	BTS fingerprints, time stamps	low sparse (high resolution relative to the actual CDRs)	yes	high (transition subgraph per each two consecutive observa-tions+dijkstra search)	N/A	N/A	shortest path between consecutive observa-tions(start/stop road segments selected from a closer edges the cellular tower)	no	yes	divergence, alignment ratio and matching ratio from the recorded GPS tracks
Proposed	trip trajectory mapping; traffic intensity estimation; and model adaptation	D4D Orange Challenge 2014	BTS fingerprints, time stamps	very sparse (with missing observations)	yes	low	HMM	multinomial distribution for state transition and emission	Viterbi search	segmental K-means clustering + MAP-like adaptation	yes	traveling distances, divergence and alignment ratio from the recorded GPS tracks + correlation with gravity model, user equilibrium traffic model, and Google earth airborne maps for the aggregated flows

by each user; (2) aggregated origin/destination flows between those stops; and (3) shared route patterns extraction from the sequences of stops visited by each user. This system does not assume any parametric models to estimate the origin/destination flows and do not consider efficient time evolution of the system using observed data. This work focuses on the flows based on the stops. The yielded routes, therefore, are coarse-grained. In contrast, an attempt is made in our work to estimate the fine-grained routes.

Chen et al. [2014] have considered a low-sampling rate (1min) of GPS data for a group of vehicle (floating vehicle); the purpose of which is to map-match the GPS data into the traffic network, thereby estimating traffic flow. They used a multi-criteria dynamic programming map-matching (MDP-MM) algorithm. Although the volume of data collected from the vehicles is large, the considered problem differs in that the spatial precision of the GPS data is high, and the trajectory is sampled regularly. The MDP-MM algorithm relies on computing the velocity in its processing, which is not possible for the mobile CDR data due to the high error in location accuracy/precision.

Mohamed et al. [2014, 2016] and Aly and Youssef [2015] have used HMM to provide real-time map-matching for challenging position data traces that are characterized by noise and sparsity, with and without input location estimation errors in SnapNet and semMatch systems, respectively. SnapNet uses coarse-grained cellular-based time stamp locations, that are sampled at high rates; where each site is associated with an estimate of the localization error. Map-matching is performed in an incremental manner that combines favouring main roads (through associating weights to roads according to their classes), and a number of heuristics to reduce noise. The incremental matching allows for on-line, real-time processing.

More specifically, SnapNet starts with two main preprocessing phases: filtering and interpolation, before applying an online Viterbi decoding operation to produce the desired routes transitions. Transitions that are unlikely to be in the actual route are removed through the first preprocessing phase through a series of consecutive filters (speed filter, α-trimmed filter, and direction filter). These filters depend mainly on the in-transient sampled traces. Transitions from a road segment to the next is a function of the geodesic distances between consecutive location observation (trajectory dependent HMM). However, in our case data traces comprise both stationary and in-transient history, thus affecting the applicability of the method.

SemMatch has almost the same system architecture as SnapNet coupled with the commonly available and energy-efficient inertial sensors. Inertial sensors allow sem-Match to detect the road semantic and use it as a hint to the map-matching process to infer a vehicle's current road segment.

Evaluation of SnapNet covers more than 400km with location update on cell-towers change (every 2min, on average) in different cities achieving more than 90% accuracy with noisy coarse-grained location estimation; while semMatch covers about 150km, including coarse-grained cellular-based positioning, low-sampling GPS, and noisy traces with back-and-force movement transitions.

Schulze et al. [2015] propose a geometric map-matching method relying on BTS locations and time stamps of each signaling event. The map-matching process is carried through two main phases; preprocessing geometric maps (OpenStreetMaps) by filtering in desired roads, followed by the construction of a search graph using corridors; a corridor defines an area between two consecutive observations. The geometry of the corridor is controlled using two parameters: the radius of cellular towers emission and the deviation of the corridor shape, varying from narrow, straight into broad round shapes. They assumed that individuals tend to follow shortest paths, thus the real path approximately follow geodesic lines between the origin and destination of a trip. The search graph thus uses a modified Dijkstra algorithm to compute the shortest path

for each corridor, speeding up the computation through caching common sub-paths in a corridor.

Schulze et al. perform a detailed evaluation of 2,249 trajectories with an average sampling time of 260s. They achieved 44% matching percentage in urban areas and 55% in rural and mixed areas. We further analyze this algorithm in Section 6, as it is the closest to our considered problem with both large scale and sparse BTS data.

2.2. HMM-Based Algorithms Optimization

HMMs are statistical models for sequential data with hidden underlying structure. HMMs have important applications in language processing [Rabiner 1989; Martin and Jurafsky 2000], speech recognition [Bahl et al. 1986; Rabiner 1989; Huang et al. 1990; Rabiner and Juang 1993], biological sequence analysis, which include modeling protein structure, gene finding, and sequence alignment [Krogh et al. 1994; Hughey and Krogh 1996; Eddy 1998, 1996a].

More recently HMM techniques are utilized in traffic monitoring and localization [Gavrila 1999; Kamijo et al. 2000; Li and Porikli 2004; Aly and Youssef 2015; Thiagarajan et al. 2009]. Thus, many modern computational techniques are employed to accelerate HMM. Vector intrinsics and multi-core architectures are used in HMMlib systems; Graphical Processing Units (GPUs) is used in cuHMM system; zipHMM uses common expressions in multi-core systems. In this subsection, we explore the most recent approaches that achieve a considerable speedup with different HMM approaches. Table II presents a comparison of the most recently HMM optimization methods compared to our proposed data sparsity optimization approach.

Soiman et al. [2014] introduced an efficient parallel HMM optimization based on IBM Cell Broadband Engine (Cell/B.E.) architecture, which offers two levels of parallelization: Message Passing Interface (MPI) and MPI combined with Synergistic Processing Element (SPEs) have been used in parallelizing Forward algorithm. The proposed optimization has been stress-tested with a very long observation sequences, on IBM Roadrunner architecture, leads to an encouraging decrease in the execution time with a total speed up $\sim 20\times$.

ZipHMM is an HMM library based on exploiting the observations repetitions to expedite the log likelihood computations for input sequences in a similar manner to string compression algorithms [Sand et al. 2013]. The library relies on a preprocessing step to identify common substrings in the input sequences and build a computational structure to be reused. This optimization leads to an enhancement of $78\times$ speedup.

HMMlib [Sand et al. 2010] is an HMM-based algorithms implementation that takes advantage of modern computational techniques, especially the SIMD instruction and multi-threading, targeting large-scale state-spaces. Double-precision SSE and OpenMP implementation lead to a speedup of $10\times$ for the Viterbi algorithm with single-precision.

Moreover, Nielsen and Sand [2011] presented an alternative formulation of the Forward and Viterbi algorithms for small-scale state space. They parallelize the workloads across observation sequences giving each processor a greater chunk of work; reducing the total communication overhead to the minimum. This optimization leads to a speedup of roughly $5\times$ and more than $6\times$ for Forward and Viterbi algorithm, respectively.

Parallelizing an input observation stream using GPUs has also been used for optimizing the HMM-based algorithms performance, leading to higher speedup factors compared to traditional optimization methods. Liu [2009] parallelized a stream of observation sequences of the same length over NVIDIA GPU platform threads and accumulated the sought probabilities using per state threads. However, the input data has limitations over the number of sequences, states, and the output. Liu et al. achieved a significant speedup of $200\times$. HMMERsearch [Eddy et al. 2007] package used GPU

Table II. MM Parallelization and Optimization Approaches Comparison

Method	Optimization Approach	Usage	Space size	Speedup
IBM Roadrunner clusters based HMM [Soiman et al. 2014]	Partition observation sequences equally over Cell BE processors using two levels of MPI combined with Synergistic Processing Elements (SPEs) accelerators	Require a specific hardware features	Large sequence length	20x
HMMlib [Sand et al. 2010]	Parallel HMM taking the advantages of SIMD support and multi-cores using SSE and OpenMP	SIMD and multicore support	All	2×, 4×, 4×, 10× for Posterior decoding, Baum-Welch, forward and Viterbi algorithms, respectively
ParredHMMlib [Nielsen and Sand 2011]	Workload is parallelized over observed sequences using distributed memory	Distributed memory	Small space size $\leq$ 512 element in both emissions and transitions tables	6×
zipHMMlib [Sand et al. 2013]	Exploiting commonly occurring substrings in the input to reuse computations as a preprocessing step failed to be used with Viterbi algorithm	Multi-core systems	All	4× -78×
cuHMM [Liu 2009]	Parallelize a stream of observation sequences of the same length over GPU threads and accumulated the interested probabilities using per state threads. The input data has some limitation over the number of sequences, states and the output.	NVIDIA GPU platform	The number of states and the number of sequences must be a multiple of the block size	200× over CPU Peak performance at no. sequences= no. states
GPU speculative HMMERsearch [Li et al. 2012]	Unroll the outer loop of the HMMERsearch [Eddy et al. 2007], but unrolling the outer loop has no benefit because of the data dependencies between iteration. Thus, they allow speculative execution of the next loop. Moreover storing intermediate results into registers, SSE intrinsic, full use of texture memory and pipelining are used in the optimization	NVIDIA GPU platform	All	6.5× over single-threaded SSE implementation
Proposed-Sparsity-based HMM	Exploit the sparsity of the HMM data by eliminating the zero transitions from the calculations and allow only emitable states to be explored. The optimized algorithm does not sacrifice accuracy, it provides the exact optimal solutions.	No specific hardware - serial execution	All	depends on the sparsity level but almost linear time complexity at high sparsity data

capabilities to enhance its performance. However, they obtained only a 6.5× speedup factor using single threaded SSE implementation.

None of the literature work in HMM-based algorithms optimization exploits HMM data sparsity. In Section 5, we exploit the sparsity of the HMM data by eliminating the zero transitions from calculations and allowing only emitted states to be explored. This provides a linear Viterbi performance. This optimization does not sacrifice accuracy, and it provides the exact optimal solutions.

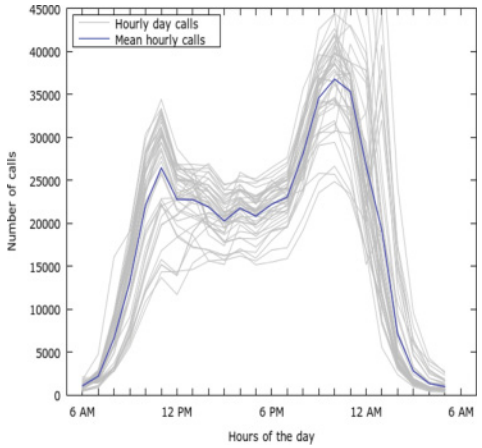


Fig. 1. Hourly cumulative frequencies of mobile calls. Gray line represents distribution for each day and blue line represents average distribution over six weeks.

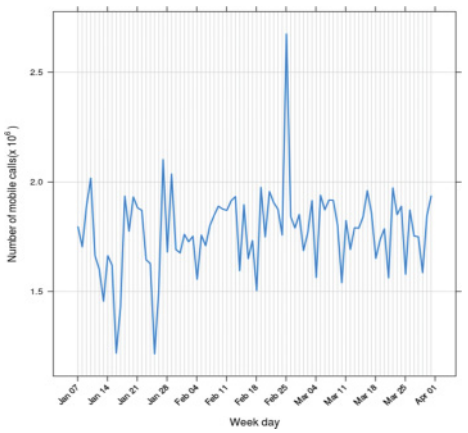


Fig. 2. Daily cumulative number of mobile calls distribution for six weeks.

3. MOBILE PHONE DATA ANALYSIS

Orange Telecom Co. and SONATEL Telcom Senegal Co. have held a competition, Data for Development (D4D), in 2014, where it provided three sets of CDRs for participating teams [de Montjoye et al. 2014]. The data is provided for the Country of Senegal, covering the year 2013. The first set provides antenna-to-antenna traffic, with hourly time granularity; the second provides antenna zone-level (fine-grained) user mobility on a rolling two weeks for about 300,000 randomly sampled users, at 10min time granularity; the third provides arrondissement-level user mobility for about 150,000 randomly sampled users.

For the purpose of this article, we consider the second set, as it provides for individualised user tracking. In this set, each CDR consists of an index to an anonymised user's mobile phone number and an index to the corresponding antenna, referred to as base transceiver station (BTS). A CDR is recorded when a user initiates or receives a phone call or an SMS; the duration of calls are not provided.

The set is structured into two weeks buckets, in which the users are not changed. The CDR data are collected from about 300,000, randomly chosen users. The original CDRs are filtered such that only users who interact more than 75% days in two weeks are kept. Also, the users with more than 1000 interactions per one week are filtered out, as those are more likely to be machines or shared phones. The BTSs are located in various places and are co-located with multiple mobile operators at the same site, each serve a different location or range of frequencies, in Senegal. The meta data provided includes distorted BTS locations to preserve privacy and for other commercial reasons; the BTS locations are noised so they belong in a random location inside its Voronoi cell. We restrict our analysis to the Dakar area, considering up to 492 BTS. Dakar is the capital and is the largest city of Senegal (total area of 82.34km^2, has about 25% of the country's population and 80% of the country's industrial and commercial enterprises).

Figure 1 shows daily mobile call distributions for six weeks, providing hourly call frequencies. Each grey line represents the hourly mobile call frequencies accumulated for all users over a day. The data has a local maximum frequency within the time interval from 10 AM to 1 PM, and a global maximum frequency during the interval from 9 PM to 2 AM per day. The blue line represents the average distribution; it has the same behaviour as the individual daily distributions. Figure 2 shows daily cumulative

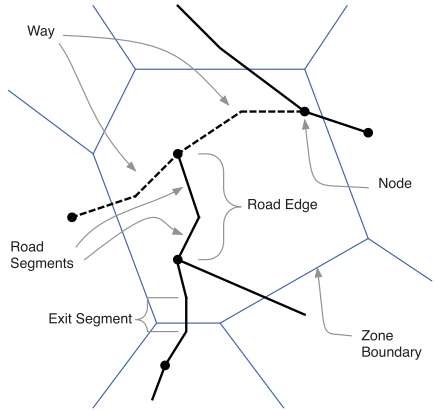


Fig. 3. Road network model notation.

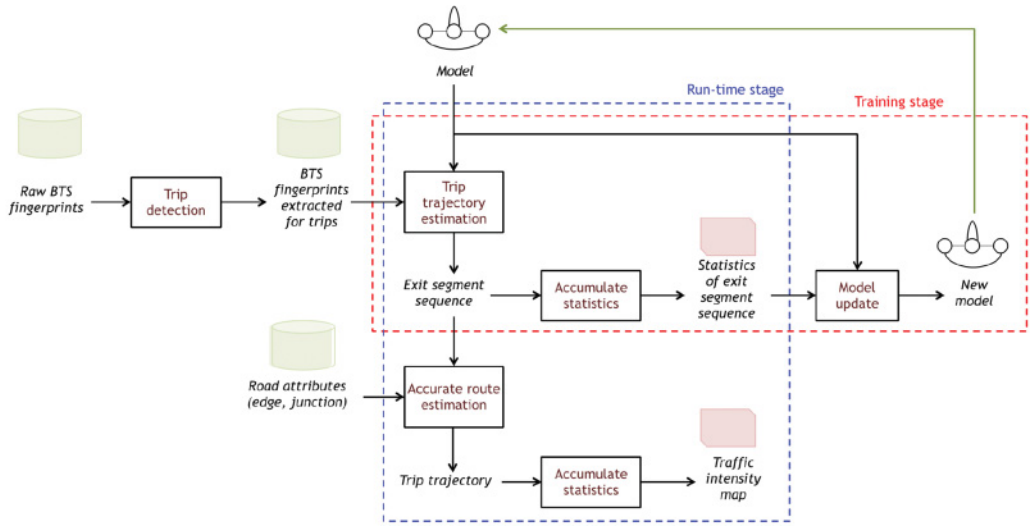


Fig. 4. Adaptive traffic monitoring system.

frequencies of mobile calls, which indicates a large diversity from day to day within the same week and the same day over weeks.

4. TRAFFIC MONITORING USING HIDDEN MARKOV MODEL

Figure 3 illustrates notations of a road network model used. A typical urban road network is modeled as a graph where each node represents a junction intersecting two or more roads. An edge represents a part of a road, we refer to as "road-edge." A road-edge is composed of a sequence of short straight-line road sub-edges, we refer to them as "road segments." An "exit-segment" is a road segment that crosses a zone's boundary.

A "road" is referred to as a "way," and it is defined as a sequence of connected nodes. To provide road geometry, extra information is associated with each road segment.

HMMs are utilized to represent the traffic phenomena on that road network. Each state in this model is defined to be an exit-segment. Figure 4 illustrates the diagram of the system for sensing the traffic after generating the map.

The system mainly consists of four processing stages as follows:

(1) **Trip detection:** BTS fingerprints for each trip (only the mobility) are extracted from all the BTS fingerprints, including stops.
(2) **Trip trajectory mapping using Viterbi algorithm:** A trajectory for each trip is estimated by mapping BTS fingerprints onto road exit-segments. This procedure is achieved by the following two steps:
   —**Viterbi alignment:** An optimal exit-segment sequence is estimated from BTS fingerprints for each trip using the Viterbi algorithm.
   —**Accurate route estimation (post-processing of Viterbi decoding):** More accurate route for each trip is estimated using not only the Viterbi outputs (i.e., exit-segment sequences) but also other road network attributes such as the corresponding road-edge and junction.
(3) **Statistics accumulation of trip trajectories and traffic intensity map generation:** The statistics of trip trajectories are accumulated. The traffic intensity map can be yielded using those statistics in terms of state transitions. The statistics of the exit-segment sequences are also accumulated for model adaptation.
(4) **Model update using accumulated statistics:** Efficient time evolution of the system is conducted by adaptively updating the model parameters using the previous model and aforementioned statistics.

4.1. Road Network Generation

The open street map (OSM) is used to generate the map for Dakar.

According to Britain transportation statistics, users are more likely to take main roads rather than non-major highways, for example, residential, unclassified, or unpaved roads, with 65.7% of vehicles on main roads out of 317.1 billion mile traffic coverage [Mohamed et al. 2016]. So, we consider five main highway road types, such as motorway, trunk, primary, secondary, and tertiary, and their links. Eliminating other road types produces road disconnections, affecting the traffic dynamics, as shown in Figure 5; we therefore manually inspected such cases and included lower ranking roads to remove the disconnections. With such breaks in roads, some of the existing routings cannot be obtained, which affect the transitions from one point to another.

4.2. BTS Zone Representation

The road network space is partitioned with the Voronoi algorithm to yield a coverage area for each BTS. Each BTS, therefore, is associated with a Voronoi cell, which is defined to be a convex set of vertices. We used the zone indices with the BTS indices determined in the CDR. Figure 6 shows a typical partition for 492 BTSs, covering Dakar city. The centers of the Voronoi partitions are determined by a systematic projection of the latitudes and longitudes of WGS84 BTS locations within the Dakar Geo-zone.

In this case, a mobile communication can be associated with the particular BTS. However, as will be explained in the next section, we model the fact that a mobile connection is not necessarily associated with the closest BTS because of the communication situation. This phenomenon can be modeled by using the probability distribution for observations (i.e., BTS indices).

4.3. Model Representation

HMMs are commonly described as a five-tuple model, $\Omega = (\{\phi\}, \{\pi\}, \{a\}, \{b\}, \{\lambda\})$ [Eddy 1996b; Rabiner 1989]. The states, ϕ, here correspond to the road exit segments. Since each exit segment is associated with only one junction, we can easily identify those exit junctions, thereby providing for more information about traffic in the roads with substantially smaller state space than taking all roads, and thus be appropriate for

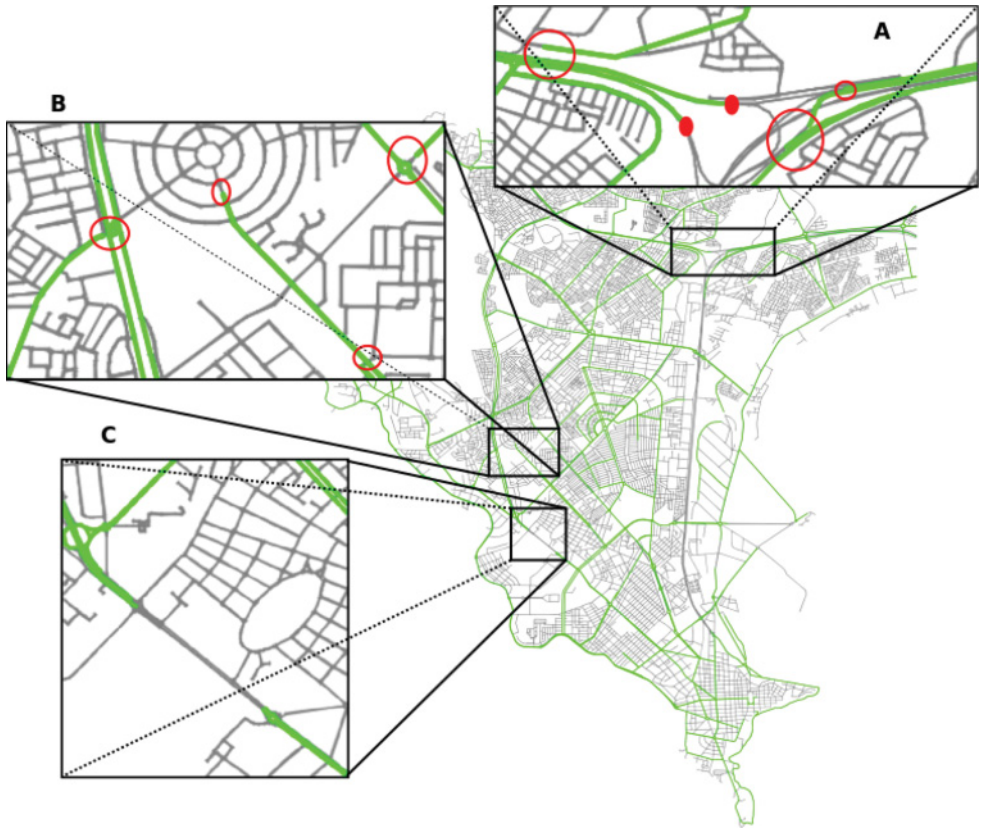


Fig. 5. Examples of routes discontinuity with plotting motorway, trunk, primary, secondary, and tertiary roads types in green and the remaining roads types of Dakar, Senegal in gray. Circles, in red color, are breaks in the original roads network result from the inaccurate definition of road segments related to the same route.

the considered city-scale problem. The state transition probability, a, is the probability of an individual moving from a particular state to the same/another state. The initial probability, π, is the probability of an individual starting from a particular state. A discrete symbol (index), λ, is observed according to the emission probability, b, for each state. CDRs contain records of individuals receiving or initiating a call or an SMS message. A CDR consists of a mobile tower's Voronoi partition (BTS index), a time stamp, and anonymized individual id. In this case, the observations in our model, $\{\lambda\}$, are only the BTS indices.

Both state transition and emission probabilities in HMMs are represented by multinomial (discrete) distributions. a_{ij} denotes the probability of the state transition from ϕ_i to ϕ_j; $b_i(c)$ is the probability of the signal from the BTS c being received at the state ϕ_i; N_S is the number of states; and Z_i is the zone where ϕ_i is included.

Figures 7 and 8 give an example illustrating how HMM is applied to a road network. Figure 7(A) shows a road network and four BTSs observations λ labeled: A, B, C, and D. Single direction roads are indicated by arrows. Nodes represent junctions, with and without exit-segments, colored grey and green, respectively. Exit-segments are labeled with small orange squares. Figure 8 shows the corresponding HMM for the not dimmed roads and junctions; the considered exit-segments, states ϕ, are labeled

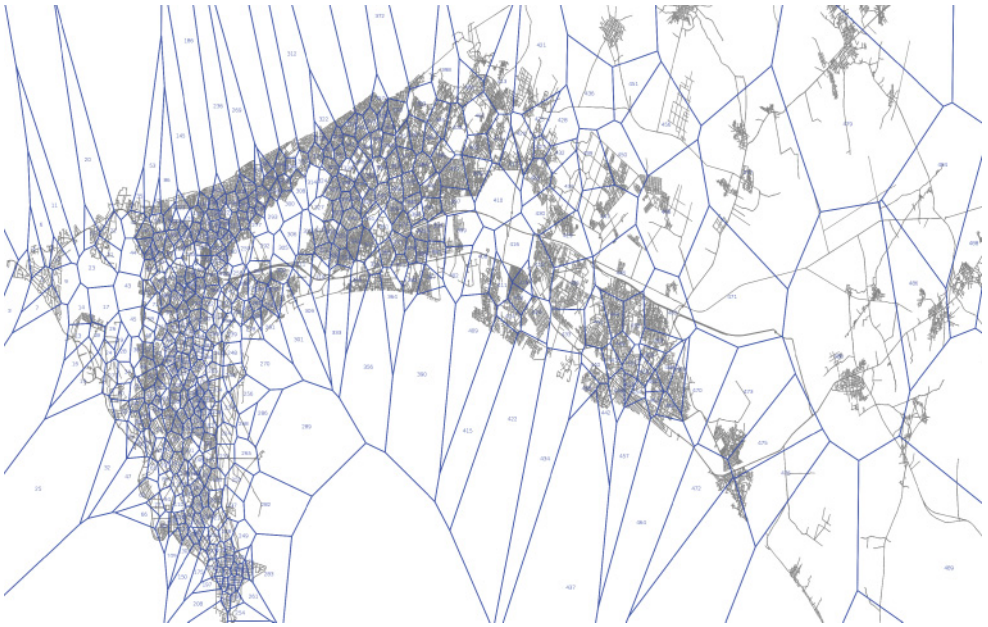


Fig. 6. Typical Voronoi partitioning for 492 BTSs with boundaries plotted in blue that covers complete area of Dakar, Senegal.

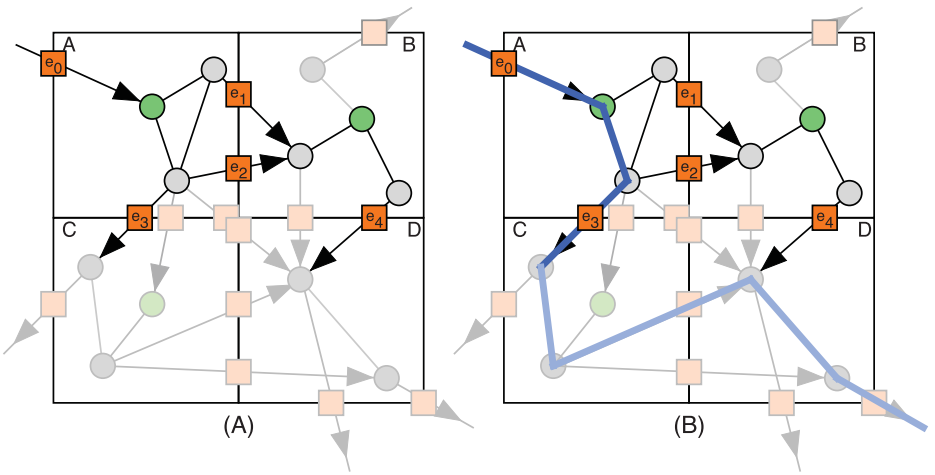


Fig. 7. (A) Road network representation for regular coverage areas of four BTSs; e_i denotes exit-segments; colors grey and green express nodes with and without exit-segments, respectively. (B) Trajectory decoding of the BTS sequence $< A, C, D >$.

with e_i. Notice that the edge between e_0 and e_4 models the case of missed observation. Figure 7(B) shows a corresponding decoded trajectory (blue) for a given BTS sequence of $< A, C, D >$.

Self state transitions, a_{ii} $(1 \leq i \leq N_S)$, and adjacent state transitions, a_{ij} $(i \neq j;$ $1 \leq j \leq N_S$; and ϕ_i is adjacent to $\phi_j)$, are dominant among all state transitions in the road and traffic network model. However, it is much likely to have missed observations as it is not guaranteed that users receive very frequent calls for each region. Small

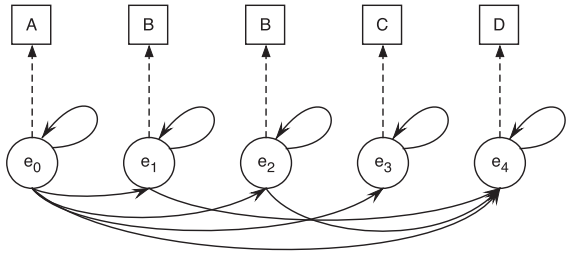


Fig. 8. State representation of HMM for traffic modeling.

probabilities, therefore, are assigned to non-adjacent state transitions to cope with the missed observations. As for the emission probability, observations from neighboring zones are considered. This accounts for possible errors in the communication model's representation, where a call is not necessarily associated with the nearest BTS. To handle this case, the multinomial (discrete) distributions for all the BTSs being observed are assigned to the states.

4.4. Parameter Initialization

The state transition probabilities are initialized on the basis of the driving distance on roads and direction between the states; we set the probability to be inversely proportional to distance. The traffic road network is reformulated as a directed weighted graph, to take traveling direction into consideration. The state transition probabilities are denoted by a_{ij}, for the transition from state ϕ_i to ϕ_j. They are initialized as follows:

$$a_{ij} = \begin{cases} \frac{(\min_{1\le k\le n_S} \mathcal{D}(\phi_i,\phi_k)-1)^{-1}}{\sum_{1\le k\le n_S} \mathcal{D}(\phi_i,\phi_k)^{-1}+(\min_{1\le k\le n_S} \mathcal{D}(\phi_i,\phi_k)-1)^{-1}} & i=j \\ \frac{\mathcal{D}(\phi_i,\phi_j)^{-1}}{\sum_{1\le k\le n_S} \mathcal{D}(\phi_i,\phi_k)^{-1}+(\min_{1\le k\le n_S} \mathcal{D}(\phi_i,\phi_k)-1)^{-1}} & \mathcal{D}(\phi_i,\phi_j)\le \ell_t\,, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\mathcal{D}(\phi_i, \phi_j)$ denotes the shortest driving distance on real roads between the exit-segments (states) ϕ_i and ϕ_j. Self transitions are set as the highest probability, which is smaller than the minimum distance between any considered exits; and it is the normalization of $1/(\min_k \mathcal{D}(\phi_i, \phi_k) - 1)$.

Cellular CDRs are generally sparse, since it is not guaranteed that subscribers initiate or receive mobile activities per visited zones in his/her way. This results in missing trips information, which makes it more complex to tracking individual routes or even may lead to inaccurate directions.

Thus, exit states do not necessarily only belong to adjacent zones, but for non-adjacent zones as well to allow for modeling missed observations. Allowing non-adjacent zone's exits transitions gives HMM the opportunity to interpolate linearly missed observation in a range controlled by ℓ_t, where ℓ_t is the spatial threshold for defining reachable states. Generally, missing observations in urban areas are higher than rural area as the coverage area per BTS in urban areas is smaller, based on the population density on urban to rural or peri-urban areas. Accordingly, we set ℓ_t to the largest distance between any two adjacent BTS (6 km), as it is experimentally matched with the BTS network of Dakar. Such considerable distances between towers can be located in rural areas. This large distance between adjacent zones guarantees that the transition probability can cope with any missed observation in either rural or urban area. We define n_S to be the number of reachable states from state i with distance less than ℓ_t.

For computing the emission probabilities, we consider the fact that the communication power is generally proportional to inverse square of the distances. Thus, we set

the initial emission probability of the BTS as the follows:

$$b_i(k) = \begin{cases} \frac{\mathcal{D}(\phi_i, \mathbf{p}_k)^{-2}}{\sum_{1 \le u \le N_z} \mathcal{D}(\phi_i, \mathbf{p}_u)^{-2}} & \mathcal{D}(\phi_i, p_u) \ge \ell_{\mathbf{p}_u} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where $b_i(k)$ denotes the emission probability of observing BTS k while being at state ϕ_i; $\mathbf{p}_k$ denotes the location of the BTS in a Voronoi zone (which is set to be the center of the zone) and, $\ell_{\mathbf{p}_u}$ denotes the maximum distance threshold between the state ϕ_i and the specific BTS within adjacent zones of $\mathbf{p}_k$.

Finally, the initial probabilities are initialized and stress-tested using uniform distribution and randomly generated probabilities. In all test cases, different π probabilities tend to converge to the same probable paths without any differences, which makes cellular-based routes largely insensitive to these particular initial values.

4.5. Trip Trajectory Mapping

Moving from the origin to destination is referred to as a "trip." An attempt is made to estimate the trajectories of individual trips from the BTS fingerprints by using the Viterbi algorithm.

4.5.1. Trip Detection. Only the BTS fingerprints are used as the observations for the HMMs in the present study. It is, therefore, needed to extract the trips by filtering out the observations that are mainly due to the individual staying at his/her origin and destination and not in transit. Our approach for detecting trips is based on that proposed by Calabrese et al. [2011] and Berlingerio et al. [2013]; this method computes a set of stop points for each user, where every two consecutive (in time) stops constitutes a trip. We, therefore, extend this approach to allow for keeping the intervening observations.

Let x and t be the BTS index and associated time stamp, respectively, and $H =< o_1, o_2, \ldots, o_n >$ be the history of an individual's activities, where o_i denotes the ith CDR observation given by (x, t). We modified the define stop described in Berlingerio et al. [2013], as shown in Algorithm 1,[1] where a stop is the maximal sub-sequence $s =< o_m, \ldots, o_k >$ for $0 < m \le k \le n$ such that $\text{Distance}(o_i, o_j) < \text{th}_s$ and $\text{Duration}(o_m, o_k) \ge \text{th}_t$,

$$\begin{aligned} \mathbf{s} =& < o_m, \ldots, o_k > \\ & \text{s. t. } 0 < m \le k \le n, \\ & \max_{\forall m \le i \le j \le k} (\text{Distance}(o_i, o_j)) < \text{th}_s, \\ & \text{Duration}(o_m, O_{k+1}) > \text{th}_t, \end{aligned} \tag{3}$$

where $\text{Distance}(o_i, o_j)$ is the distance between any two observations belong to the buffer, $\text{Duration}(o_m, O_{k+1}) > \text{th}_t$ is the duration between the start of the trip to the current observation being examined, th_s is a spatial threshold (set to 1km), and th_t is a temporal threshold (set to 1h), such values have been demonstrated to be realistic in Calabrese et al. [2011].

In our formulation, we differ in that we reduce the stop sequence into the first element for a destination of a trip and the last element o_k as the origin of the next trip. The rationale behind choosing these elements is to provide for continuity of observations within a trip.

[1]The algorithm rejects stop sequences overlapping with earlier non-stop sequences.

ALGORITHM 1: Modified stop detection

```
Input: Historical observation sequence H of the user u
Spatial threshold th_s
Temporal threshold th_t
Output: Trips sequence separated by a seperator
begin
    Trips ← {};
    Buffer ← {};
    Max_dist ← 0;
    for all a in H do
        for all b in Buffer do
            if (distance(a, b) > Max_dist) then
                Max_dist ← distance(a, b);
            end
        end
        if (Max_dist > th_s) then
            Duration ← a.time − first(Buffer).time;
            if (Duration ≥ th_t) then
                if (Trips.empty) then
                    Trips ← Trips ∪ {last(Buffer)};
                else
                    Trips ← Trips ∪ {first(Buffer)};
                    Trips ← Trips ∪ {separator};
                    Trips ← Trips ∪ {last(Buffer)};
                end
                Buffer ← {};
                Max_dist ← 0;
            else
                Trips ← Trips ∪ {Buffer};
                Buffer ← {};
                Max_dist ← 0;
            end
        end
        Buffer ← Buffer ∪ {a};
    end
end
```

Thus, we define a trip by two consecutive stops (the end point of one stop as the origin and the first element of the next stop as the destination of the current trip), and we also include all intervening (nonstop) observations.

We have special consideration for the first observation; if it does not belong to a stop, we define it alone as a stop, thereby defining the first trip. Moreover, to alleviate the effect of event-driven location measurement, we remove successive observations with the same time stamp (handover events) [Caceres et al. 2012; Ahas et al. 2010].

4.5.2. Trajectory Decoding. Forced alignment using Viterbi algorithm estimates an optimal state sequence (exit-segments sequence) $\phi_{n,1}, \ldots, \phi_{n,T_n}$ for the observation sequence of the nth trip $\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,T_n}$.

Let $v_n(t,i)$ be the emission probability of $\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,t}$ being generated along the optimal state sequence and $\mathbf{x}_{n,t}$ being emitted at ϕ_i; $\psi_n(t,i)$, the pointer to the previous state; T_n, the number of samples in the nth trip; and N_S, the number of states. In this case, the optimal state sequence can be obtained by the following procedure:

(1) Initialization ($1 \leq i \leq N_S$):

$$v_n(1,i) = \pi_i \cdot b_i(\mathbf{x}_{n,1}), \tag{4}$$

$$\psi_n(1,i) = 0. \tag{5}$$

(2) Iteration ($2 \leq t \leq T_n, 1 \leq i \leq N_S$):

$$v_n(t,i) = \max_{1 \leq k \leq N_S} [v_n(t-1,k) \cdot a_{ki}] \cdot b_i(\mathbf{x}_{n,t}), \tag{6}$$

$$\psi_n(t,i) = \arg\max_{1 \leq k \leq N_S} [v_n(t-1,k) \cdot a_{ki}] \cdot b_i(\mathbf{x}_{n,t}). \tag{7}$$

(3) Determination:

After iterative calculation of $v_n(t,i)$ for $t = 1, \ldots, T_n$, the emission probability of $\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,T_n}$ being generated from the optimal state sequence can be determined as follows:

$$P(\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,T_n}) = \max_{1 \leq k \leq N_S} [v_n(T_n,k)], \tag{8}$$

$$\phi_{T_n} = \arg\max_{1 \leq k \leq N_S} [v_n(T_n,k)]. \tag{9}$$

(4) Backtracking ($t = T_n - 1, T_n - 2, \ldots, 1$):

The optimal state sequence of the nth trip can be determined by tracking back the nodes giving the maximum likelihood as

$$\phi_{n,t} = \psi_n(\phi_{n,t+1}, t+1). \tag{10}$$

Figure 11(b) shows three Typical example of individual users, fine-grained, trip trajectories (green). These trajectories are obtained through mapping of mobile calls sequences to produce most probable paths taken by individuals.

4.5.3. Sufficient Statistics Accumulation for Trip Trajectories. Two sorts of statistics are accumulated in the developed system: the statistics for road-level, fine-grained trip trajectories and those of the BTS fingerprints for exit-segment (state) sequences. The former yields the traffic intensity map and the latter is utilized for model adaptation.

The traffic intensity, which represents the number of vehicles passing through the specific road segments during a given period with N trips, can be computed as statistics of trip trajectories. In this case, the number of trips trajectories are accumulated for N trips for each road segment; for repeated segments result from repeated observations, only the count is incremented once, as this situation potentially indicates slow traffic and no loop through the zone. Figure 9 shows a typical example of the traffic intensity during day working hours. Light traffic at the early hours (6 AM to 9 AM) increases gradually as time proceeds then high traffic flow becomes limited to only a few road segments at the work break hours. Next the traffic flow backs to a light state at 3 PM to 4 PM then increases at the end of the working hours. These figures show that the adaptation algorithm captures only cellphones in a motion state and eliminates all observations within stops.

Forced alignment using the Viterbi algorithm can assign an observation $\mathbf{x}_{n,t}$ to a state in the HMM. As aforementioned, this assignment yields an optimal state (exit-segment) sequence for a trip. Let $\xi_n(t,i,j)$ be an assignment of the observations $\mathbf{x}_{n,t-1}$ and $\mathbf{x}_{n,t}$ to the state ϕ_i and ϕ_j, respectively; and $\gamma_n(t,i)$ be an assignment of $\mathbf{x}_{n,t}$ to ϕ_i. In the case of the Viterbi alignment, $\xi_n(t,i,j)$ takes one if the state transition $s_{t-1} = \phi_i$ to $s_t = \phi_j$ is on the optimal state sequence for the nth trip and otherwise takes zero.

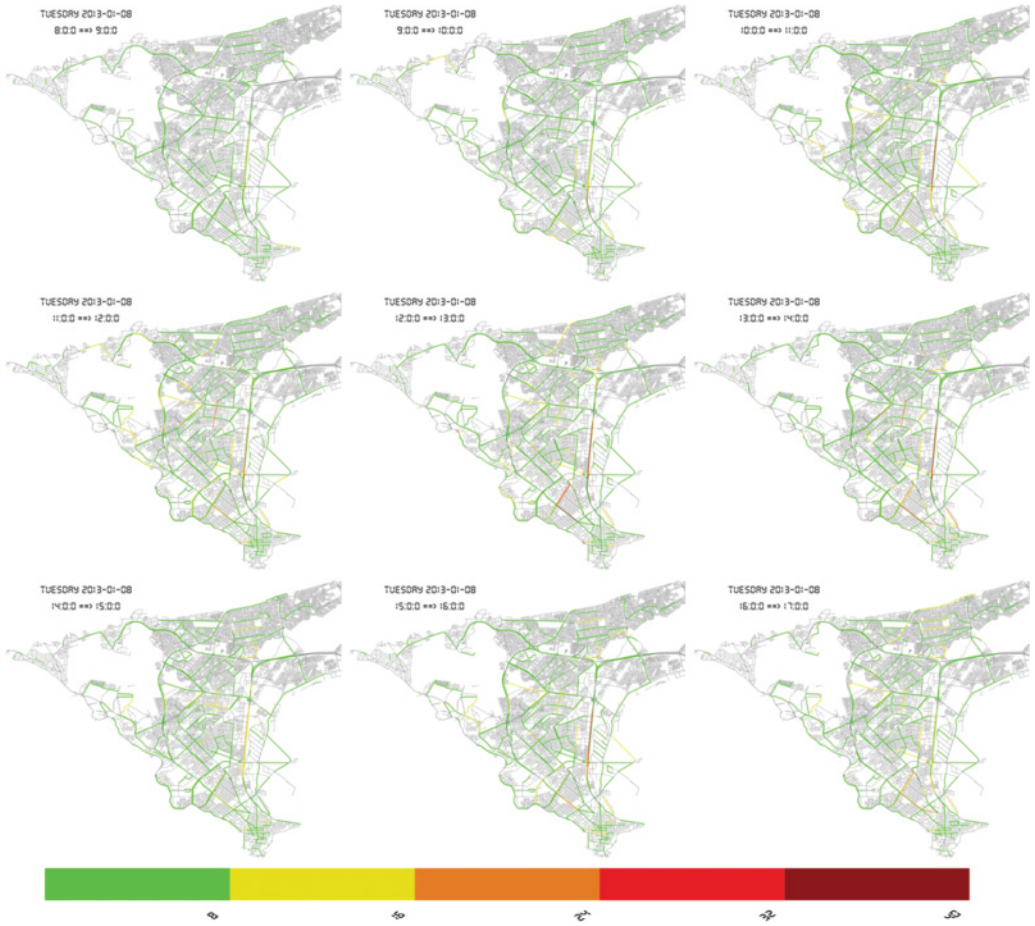


Fig. 9. Typical example of a working day hourly traffic intensity; starts with a light traffic at early hours, then gradually increases during the 9 AM to 11 AM period. At noon hours, only few roads remain active with high traffic intensity. As time proceeds to work break time (1 PM to early 3 PM), more roads become active, then the traffic state drops to light state and increases at end of working hours.

In addition, $\gamma_n(t, i)$ takes one if the state $s_t = \phi_i$ is on the optimal state sequence and otherwise takes zero. Noted that the statistics of $\xi_n(t, i, j)$ and $\gamma_n(t, i)$ accumulated along the optimal state sequences are used also for updating the model parameters under the maximum likelihood criterion.

4.6. Adaptive Traffic Model Estimation

Time evolution of the system can be achieved by adaptively updating the model parameters in HMMs. The maximum likelihood (Viterbi training) [Allahverdyan and Galstyan 2011] estimates of the state transition and emission probability distribution can be calculated using accumulated statistics about the BTS fingerprints for the state sequences, $\sum_{n=1}^{N} \sum_{t=1}^{T_n} \xi_n(t, i, j)$ and $\sum_{n=1}^{N} \sum_{t=1}^{T_n} \gamma_n(t, i)$, obtained as the results of Viterbi alignment, as follows:

$$a_{ij}^{\mathrm{ML}} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T_n} \xi_n(t, i, j)}{\sum_{n=1}^{N} \sum_{j=1}^{N_S} \sum_{t=1}^{T_n} \xi_n(t, i, j)}, \tag{11}$$

$$b_i^{\mathrm{ML}}(c) \;=\; \frac{\sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_n(t,i)\cdot\delta(\mathbf{x}_{n,t},c)}{\sum_{n=1}^{N}\sum_{t=1}^{T_n}\gamma_n(t,i)}, \tag{12}$$

where a_{ij}^{ML} denotes the ML estimate of the probability of the state transition ϕ_i to ϕ_j; $b_i^{\mathrm{ML}}(c)$, the ML estimate of the probability of the signal from BTS c being received at the state ϕ_i; N, the number of training data sequences (i.e., trips); T_n, the number of samples in nth trip; and

$$\delta(\mathbf{x}_{n,t},c) = \begin{cases} 1, & \text{if } \mathbf{x}_{n,t} = c \\ 0, & \text{otherwise} \end{cases}. \tag{13}$$

For example, Equation (11) represents the rate of the vehicles receiving the signal while passing through the road ϕ_i to ϕ_j among the vehicles passing through ϕ_i for N trips.

The maximum likelihood estimates described in Equations (11) and (12) are possible to be unreliable for the states and state transitions with only small amount of data assigned. To solve this problem, an attempt has been made to linearly interpolate the ML estimates of the model a_{ij}^{ML} and $b_i^{\mathrm{ML}}(c)$ with the previously estimated parameters $a_{ij}^{(r)}$ and $b_i^{(r)}(c)$, yielding the new estimates $a_{ij}^{(r+1)}$ and $b_i^{(r+1)}(c)$ as follows:

$$a_{ij}^{(r+1)} \;=\; \alpha\cdot a_{ij}^{(r)} + (1-\alpha)\cdot a_{ij}^{\mathrm{ML}}, \tag{14}$$

$$b_i^{(r+1)}(c) \;=\; \alpha\cdot b_i^{(r)}(c) + (1-\alpha)\cdot b_i^{\mathrm{ML}}(c), \tag{15}$$

where r denotes the number of updates and α, the interpolation coefficient. The effect of α on the performance of the proposed system is investigated in Section 6 (in Figure 18(b)).

Our HMM formulation is based on the underlying traffic network and cellular towers locations semantics. This makes it suitable to infer traffic flow with different data patterns including work days and weekend days. However, training our model using a hybrid collection of weekend and labor days significantly alter model convergence, as work days and weekend days may have different transition probabilities. Therefore, training the proposed model with a day-to-day slicing periods may significantly enhance its accuracy, for example, peak hours transitions probabilities. In our results, we focus on working days eliminating training with weekend days.

5. VITERBI DECODER OPTIMIZATION

The execution time of the Viterbi algorithm is critical, owing to a large number of states and observation sequences. A standard implementation of the algorithm results in around 80 seconds to reconstruct a single observation sequence (running on a recent iDataPlex, Intel Xeon CPU E5-2660, 2.20GHz based processing node). For the 300,000 users provided, the number of observation sequences per day is in the order of 100,000; thus, this will require 92 days to process a single day.

In this section, we propose an optimized Viterbi algorithm that exploits the sparsity of the data, allowing for real-time processing; the optimized algorithm does not sacrifice accuracy and provides the exact optimal solution as the standard Viterbi algorithm. The algorithm is easily parallelizable over parallel cluster systems, allowing for further scalability.

5.1. Viterbi Algorithm

To illustrate our optimisation, we first review the standard Viterbi decodor [Forney 1973; Viterbi 1967] Algorithm 2. The algorithm takes the following as input:

ALGORITHM 2: Viterbi Algorithm

Input: Emissions, $b_m = \{b_0, b_1, \ldots, b_{m-1}\}$ where m is the number of all possible emissions.
Observation sequence, $\lambda =< o_0, o_1, o_2, \ldots, o_{l-1} >$ where $o_i \in b_m$ and l islength of the sequence.
Transition Probabilities, $a_{i,j}$ such that $0 \leq i, j \leq n-1$.
States, $\phi = \{\phi_0, \phi_1, \ldots, \phi_{n-1}\}$.
Emission Probabilities, $b_{i,j}$ such that $i \in \phi$ and $j \in \lambda$.
Start Probabilities $\pi_i \in \phi$
Output: Max. Probability, OutputProb
Edge sequence, OutputPath $=< \phi_{o_0}, \phi_{o_1}, \ldots, \phi_{o_{l-1}} >$

```
1  begin
2      V ← {};
3      Path ← {};
4      /* Initialize the case for the observation 0                    */
5      for all y ∈ φ do
6          V[0][y] ← π_y × b_{y,O_0};
7          Path[y] ← y;
8      end
9      /* Proceed with dynamic programming for the rest of observations */
10     for i ← 1 to l − 1 do
11         for all y ∈ φ do
12             (prob, state) ← (max_{y_0∈φ}(V[t − 1][y_0]a_{y_0,y}b_{y,O_t}), y_0);
13             V[t][y] ← prob;
14             NewPath[y] ← Path[state] + y;
15         end
16         Path ← NewPath;
17     end
18     (prob, state) ← (max_{y∈φ}(V[n][y]), y);
19     OutputProb ← prob;
20     OutputPath ← Path[state];
21 end
```

—Set of HMM states, ϕ.
—The transition probabilities among states, $a_{i,j}$ (where $i, j \in S$).
—A set of all possible emissions, b_m, that can be emitted by all the states.
—The emission probabilities for each state, $b_{i,j}$ (where $i \in b_m$, $j \in \phi$).
—Initial starting probabilities for each state, π_i (where $i \in \phi$).
—A given observation sequence, λ, which the Viterbi algorithm decodes, finding the corresponding hidden states that emits such sequence with maximal probability.

The algorithm starts by the initialization of the Viterbi array, V, to null (line 2); V is a two-dimensional array, where the first index is the time and the second is the state. The array holds the best (maximum) Viterbi probability for a given state. The algorithm also initializes the Path array to null (line 3). The path is a one-dimensional array indexed by the state and contains the sequence of states constituting the best current path to that particular state.

The algorithm then initializes the V array to initial probabilities and Path array to the corresponding state (lines 5–8).

The algorithm's main loop (lines 9–17) iterates over each time step (corresponding to an observation), and for each state it checks the best predecessor state, that has a maximum probability (line 12). As we have n states, and l observations (or time steps), the complexity of such operation is $O(n^2 l)$.

The last part of the algorithm selects the V entry with the highest probability and emits it with the corresponding path (lines 18–20).

5.2. Eliminating Multiplication by Zero Computations

One obvious optimization is eliminating zero transitions from the calculation. This can be easily done by providing a linked list representation of all predecessor states to a particular state, thus potentially improving the asymptotic complexity by n for sparse transitions.

A more precise optimization is to eliminate multiplication by zero emission probabilities, that is, for states that have zero emission probabilities. The problem here is that the zero has to be written into the corresponding entry in the V array, requiring the same order of computation. However, we propose a simple data structure that allows for avoiding the zero writes.

A significant observation with the Viterbi algorithm is that at a given time step, t, all states values (in the V array) are updated, based on reading values from the previous time step $t-1$. We thus associate a time stamp, T, with each value indicating the time of the last update, as well as keeping the last two written values, at T and $T-1$; when reading and writing, the current time step value, t, is given, and the value is either one of the stored values or zero if $T < t-1$.

Algorithms 3 and 4 below provide the corresponding algorithms for writing and reading. The V array dimensionality is reduced to a one, such that each entry includes:

—T: time stamp of the last write operation;
—v[0..1]: an array of two Viterbi values, organized as a stack;
—h: the index of the head of the stack, indicating the most recent written value.

The write operation (Algorithm 3) writes the "value" onto the oldest entry, updating the head index, h, to point to the other entry. If the "WriteTimestamp" is not the next $T+1$ (i.e., not the next, expected, time stamp), then the oldest entry is nulled; in this case, we maintain the condition that the last entries are recorded, and since no write is done for $T-1$, it is set to zero.

ALGORITHM 3: Write Algorithm

```
   Input: WriteTimestamp,Value
   Output: T, v[2], h
1  begin
2  |  /* Invert the index of the head of the stacks (h) between 0 and 1   */
3  |  h ← h̄;
4  |  v[h] ← Value;
5  |  if (T+1) ≠ WriteTimestamp then
6  |  |  v[h] ← 0;
7  |  end
8  |  T ← WriteTimestamp;
9  end
```

The Read operation (Algorithm 4) takes the read time stamp, "ReadTimestamp," and the corresponding entry values, and returns the correct value; as the entry includes values stored into T and $T-1$, a read to an earlier entry returns zero.

The full algorithm after optimization is given in Algorithm 5; the modified lines are shown in "bold."

ALGORITHM 4: Read Algorithm

```
   Input: ReadTimestamp
   T, v[0..1], h
   Output: Value
 1 begin
 2 |  ΔT ← T −ReadTimeStamp;
 3 |  if (ΔT = 1) then
 4 |  |  Value ← v[h̄];
 5 |  else
 6 |  |  if (ΔT = 0) then
 7 |  |  |  Value ← v[h];
 8 |  |  else
 9 |  |  |  Value ← 0;
10 |  |  end
11 |  end
12 end
```

5.3. Performance Analysis

It is worth noting that when considered the total number of exit-edges considers (order of 12,000) and a sequence of 16 observation, the execution time of the base algorithm is approximately 13s. Our new optimization reduces the time into 2.5ms, giving a speed up of 5,200×. Moreover, with increasing the number of roads (e.g., when considering a larger area), the number of transitions and observations per state is likely to be constant as it depends on the local traffic network topology and antenna. Therefore the computation complexity becomes $O(n)$ (in both time and space). Our experiments have been carried on iDataPlex, Intel Xeon CPU E5-2660 with two NUMA nodes of 8 cores per socket (256KB L2 cache, 20MB L3 cache) with single thread per core, running at 2.20GHz and 125GB main memory.

Furthermore, we have compared our implementation with the highly optimised HMMlib [Sand et al. 2010] for new CPUs. HMMlib takes advantages of CPU's new features, such as SIMD and multiple processing cores. Features of HMMlib can be controlled by the user and it has the only advantage over the large states space. Comparing with such highly optimized package is interesting, because it shows the effectiveness of eliminating zeros multiplications. Comparing the proposed sparsity optimization with HMMlib, we got a speedup of 1,234× with states space 3,072 states and 512 observables. As shown in Figure 10, reducing the state space to 1,024 state, the running time gap between the highly parallelized HMMlib and the proposed optimized sequential Viterbi decreases as the total number of states multiplication is decreased, giving 580× speedup to the optimized Viterbi for sparse HMM over HMMlib.

ParredhmmLib [Nielsen and Sand 2011] is another parallelized HMM package developed by the same team of HMMlib. ParredhmmLib has been optimized for small state space. Unfortunately, all trials to use this package for state space greater than 1,024 failed. So, we examine it against our implementation with a small state space, as shown in Table III; for 256 states the proposed optimization achieves a speedup of 2,075× when compared to ParredhmmLib. When duplicating the number of states to 512, the speed up dramatically increases to 9,286×l with further increase in the state space to 1,024 states, the speedup increases to 13,445×.

Finally, we have compared our implementation with GHMM [Schliep et al. 2006]. GHMM implements the standard HMM computing algorithms [Rabiner 1989] with some extensions to the model and the implemented algorithms. GHMM is not optimized for neither large state space not small space and it doesn't provide any parallelization

ALGORITHM 5: Optimized Viterbi Algorithm for Sparse Transitions and Emissions

Input: Emissions, $b_m = \{b_0, b_1, \ldots, b_{m-1}\}$ where m is the number of all possible emissions.
Observation sequence, $\lambda =< o_0, o_1, o_2, \ldots, o_{l-1} >$ where $o_i \in b_m$ and l is length of the sequence.
Transition Probabilities, $a_{i,j}$ such that $0 \leq i, j \leq n-1$.
States, $\phi = \{\phi_0, \phi_1, \ldots, \phi_{n-1}\}$.
Emission Probabilities, $b_{i,j}$ such that $i \in \phi$ and $j \in \lambda$.
Start Probabilities $\pi_i \in \phi$
Output: Max. Probability, OutputProb
Edge sequence, OutputPath $=< \phi_{o_0}, \phi_{o_1}, \ldots, \phi_{o_{l-1}} >$

```
1  begin
2      V ← {};
3      Path ← {};
4      /* Initialize the case for the observation 0                          */
5      for all y ∈ φ do
6          /* Initialize the Viterbi entry to have a time stamp T with zero, v[0]
              with StratProb, v[1] with zero, and head index h with zero     */
7          V[y] ← (0, π_y × b_{y,O_0}, 0, 0);
8          Path[y] ← y;
9      end
10     /* Proceed with dynamic programming for the rest of observations      */
11     for i ← 1 to l − 1 do
12         for all y ∈ φ|b_{y,O_t} ≠ 0 do
13             (prob, state) ← ( max_{y_0∈φ|a_{y_0,y}≠0} (Read(V[y_0], t − 1)a_{y_0,y} b_{y,O_t}), y_0 );
14             Write(V[y], t, prob);
15             newpath[y] ← path[state] + [y];
16         end
17         path ← newpath;
18     end
19     OutputProb ← prob;
20     OutputPath ← path[state];
21 end
```

like the proposed optimized algorithm. Comparison with such a general implementation gives a speedup of 1902× with states space 3,072 states and 1,024 observables.

So our proposed optimization can be useful for high sparsity HMM model, as it eliminates all multiplications zero and reduces the search space by keeping only with emit-table states from each state observable.

6. VALIDATION AND ANALYSIS

In this section, we present our evaluation of the proposed HMM-based map-matching approach. With the absence of ground truth data for the CDR, we resorted for two main validation approaches; the first relies on oversampling individual ground-truth traces to emulate cellular traces and using the GPS as ground truth. The second is to consider the aggregate cellular-based traffic flow and correlate against theoretical traffic flow models (e.g., Gravity and Equilibrium traffic flow models), and instantaneous traffic volume snapshot using Google maps as explained in the following subsections.

6.1. Emulated Cellular Traces Map-matching

We stress-tested our approach using real challenging positioning (GPS) data traces to assess the accuracy of individual trajectories. We consider two trajectories types: relatively accurate GPS-based trajectories (considered as ground truth) and cellular-based

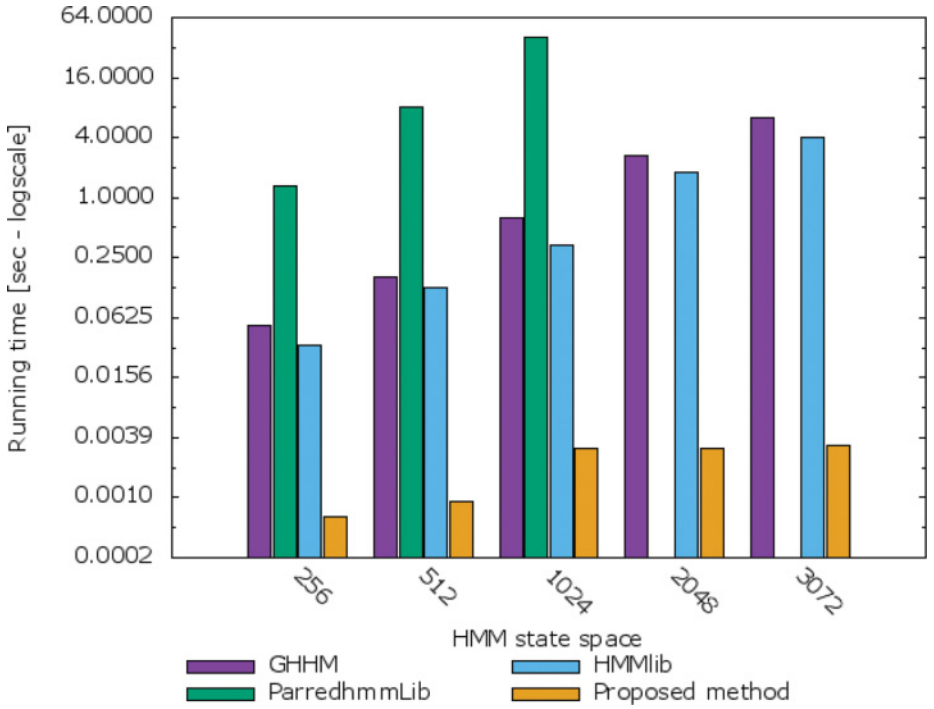


Fig. 10. Sparsity optimized Viterbi decoder running time compare to highly parallelized HMMs implementations.

Table III. Sparsity Optimisation Speedup Versus HMMs

State space	GHHM	ParredhmmLib	HMMlib
256	80	2,075	52
512	181	9,286	143
1,024	201	13,445	108
2,048	844	—	582
3,072	1,902	—	1,234

trajectories evaluated using our proposed map-matching and the method of Schulze et al. [2015]; the closest work in the literature work using only cellular fingerprint without any additional information, and considering large-scale data. The cellular-based trajectory is subsampled from the ground-truth traces, with a fixed sampling rate.

We used six metrics (similar to Schulze et al. [2015] and Aly and Youssef [2015]) to assess the proposed method including: divergence, precision, recall, F-score, commuting distances, and, finally, the execution time. To evaluate the metrics, we define the trajectory as pair-wise connected exit segments (connected with straight lines). We measure the commonly matched trajectories subsequences lengths between the ground truth and cellular mapped trajectories by, sampling each path with a fixed rate (1m samples), computing the distance from each point to the nearest ground truth.

We define the evaluation metrics as follows:

(1) *Divergence*: is the average distance between the map-matched trajectory and the nearest GPS point. A zero distance from a cellular-based trajectory segment to the ground-truth traces means a perfectly matched route.

(2) *Precision, recall, and F-score*: are defined in terms of segments in the map-matched path that have a distance less than a certain threshold from the nearest GPS point. We refer to these segments as "correctly matched trajectory segments."

For evaluation, we set the distance threshold to 150 meter, which is half the average distance from any road segment to the nearest targeted roads categories, for example, primary, secondary, ..., and so on. The three metrics are defined as follows:

—*Precision*: is defined as the ratio of the total length of the correctly matched trajectory segments to the total length of the whole map-matched trajectory.

—*Recall*: is defined as the ratio of the total length of the correctly matched trajectory segments the total length of the ground-truth trajectory. The ground-truth trajectory a sequence of straight lines connecting each consecutive pair or GPS points.

—*F-Measure*: is the harmonic means the precision and recall, defined as

$$F\text{-}Measure = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right). \tag{16}$$

(3) *Commuting distance*: represents the total length of the map-matched trajectory.

(4) *Execution time*: is the time required to map a cellular observations sequence into trajectory segments without taking into consideration the preprocessing time (e.g., in Schulze et al. execution time is the time consumed to construct a search graph and find the shortest path).

We test the accuracy of our map-matching approach and Schulze et al.'s map-matching algorithm on real data traces collected from Dakar, Senegal. Our data consists of $\sim$ 300km (3,700 observation) of driving cover almost all Dakar, Senegal. As shown in Figure 11(a), data traces intersect with both peri-urban and urban areas (mixed areas).

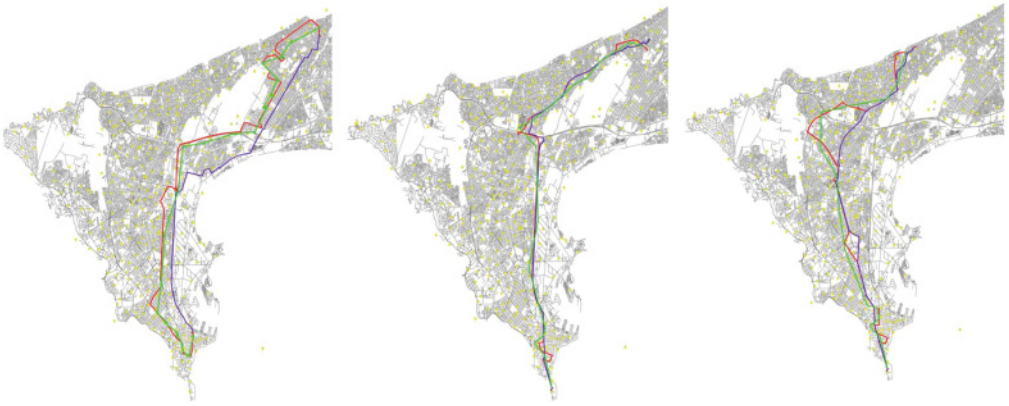
We stress-tested Schulze et al.'s proposed algorithm against varying transmission radius. We set the minimum cellular tower coverage of transmission radius to be 0.5km; we also consider half the average distance between two adjacent towers, which is 1km; and we considered the maximum distance between adjacent zones, which is 5km; moreover, we take into consideration Schulze et al.'s recommendation of 3km in combination with deviation equals to 1. These values yield the best results, according to Schulze et al., and produce good approximation for rural area's trajectories. Moreover, we measure the accuracy of these settings at a varying sampling intervals from 1min to 30min. Figure 11(b) presents real data traces mapped to road segments using our probabilistic HMM-based map-matching approach and Schulze et al.'s algorithm, in green and blue colors, respectively, at 1km radius and low-sampling interval equals 1min. Base stations are denoted by a small yellow circle, indicating their physical locations.

In the three randomly presented routes, cellular-based decoded trajectories are very close to the ground-truth, but there are two primary notices:

—For any concave or convex subroutes with arc length less than the assumed radius of transmission, Schulze et al. take the shortest path instead of the actual routes. According to the findings of Hoteit et al. [2013], in estimating real human trajectories using mobile data, human trajectories can be approximated using a number of models, including linear, nearest and polynomial interpolation, indexed based on individual's radius of gyration; thus, shortest route can result in inaccuracies.

—Cellular-based trajectories commuting distances are less than the actual length in both ours and Schulze et al. However, it is shorter in Schulze et al.'s case. This might be attributed to that length and divergence are highly related to the radius of

(a) Real data traces; representing both urban and rural areas of Dakar, Senegal. Data consists of 15 trajectories, $\sim$ 300 km total commuting distance, each plotted in a unique color.

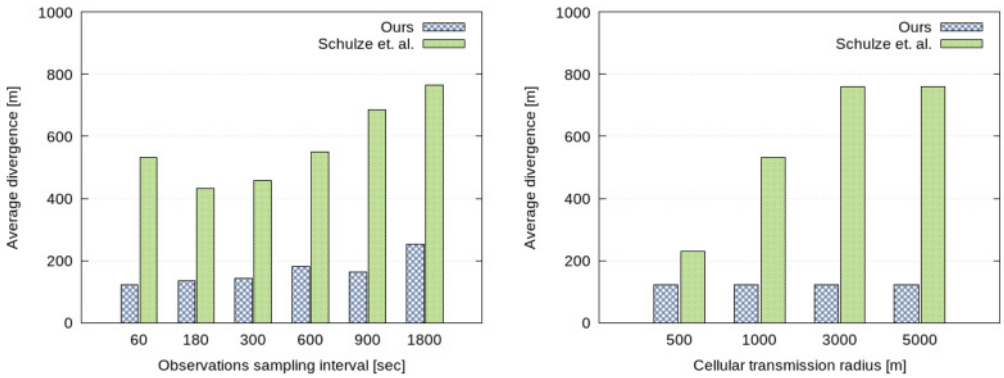


(b) Real data challenging decoded routes using our probabilistic map-matching method and Schulze et al. algorithm, in green and blue colors respectively, at 1 km radius and short sampling intervals equals to 1 min. Base stations are denoted by a small yellow circle indicating their physical locations. GPS data traces are plotted in solid red color. Trajectories presented in this example are selected randomly.

Fig. 11. Real data challenging map-matching positions example.

transmission, and thus a uniform setting introduces inaccuracies; this is especially important when considering urban, peri-urban, and rural areas.

We used different scenarios of challenging map-matching using our probabilistic HMM-based proposed method as well as Schulze et al.'s algorithm to estimate the evaluation metrics. The results are as follows:



(a) Sampling rate comparison at a fixed transmission radius of $1km$

(b) Transmission radius effects at constant sampling rate of $1minute$

Fig. 12. Average divergence comparison between GPS ground-truth trajectories and cellular-based matched routes decoded using our proposed HMM-based map-matching approach and Schulze et al.'s algorithm.

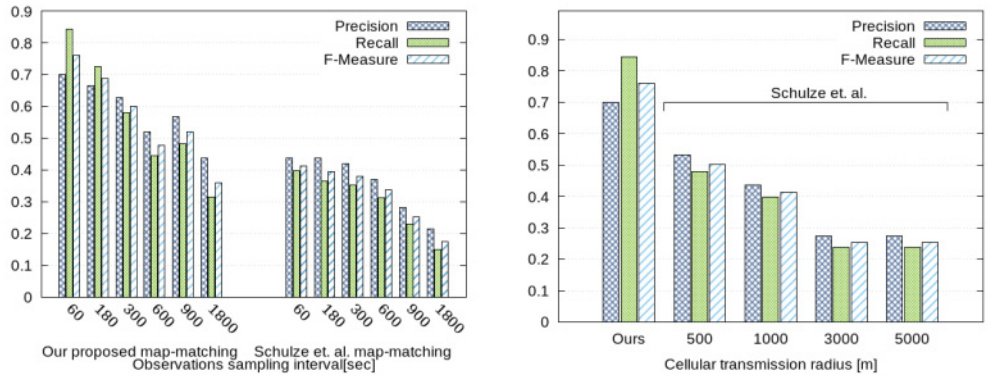
—*Divergence*: Figure 12 shows the overall average divergence of the cellular-based mapped trajectories from the ground-truth paths with an alternating observation sampling interval and transmission radius. Our probabilistic map-matching approach tends to have the least divergence as compared to Schulze et al.'s algorithm, at a distinct sampling intervals as Figure 12(a) presents. Schulze et al.'s algorithm deviation starts high with 1min sampling then reaches the best deviation at time sampling of 3min and back to increase. Increasing the sampling rate negatively affects the deviation from the true routes but with higher deviation in Schulze et al.'s algorithm.

While the radius of transmission affects the route of Schulze et al.'s divergence, it has no effect on our method, as shown in Figure 12(b). Increasing the transmission radius negatively affects the deviation as enlarging the transitions search graph, which allows Dijkstra to find much more shorter paths than the actual route even if the chosen route is far from the true path.

—*Precision, recall, and F-measure*: Figure 13 shows that our proposed HMM-based map-matching has better accuracy as compared to Schulze et al.'s algorithm. For the best sampling rate 1 min and the smallest recommended transmission radius 500m, our map-matching approach leads to enhancement in the precision, recall, and F-Measures of 131%, 176%, and 152%, respectively, over Schulze et al.'s algorithm.

Moreover, increasing the sampling rate reduces the accuracy of the cellular-based route in general. But the accuracy reduction in Schulze et al.'s routes is much higher than ours as shown in Figure 13(a). As mentioned before the presumed radius of transmission also negatively affects the accuracy, as presented in Figure 13(b). This highlights the advantages of our probabilistic model over Schulze et al.'s algorithm in mapping cellular observations into road segments.

—*Commuting distance*: At the short sampling interval (1min), our HMM-based map-matching approach and Schulze et al.'s algorithm tend to produce routes at almost the same distance from the ground-truth path length. Increasing the sampling rate closes the gap between our map-matching routes and the ground truth. Table IV shows a randomly selected routes commuting distances using different map-matching methods as well as the corresponding true paths lengths at varying time sampling rate and 1km transmission radius. Commuting numbers prove that the length of the



(a) Varying observations sampling rate (b) Diverging cellular transmission radius

Fig. 13. Accuracy of the proposed HMM-based map-matching as compared to Schulze et al.

Table IV. Randomly Selected Commuting Distances using Different Mapping Methods and the Ground-truth with Diverging Observations Sampling Rate and Fixed Transmission Radius Equals 1km

	3min sampling		5min sampling		10min sampling		15min sampling	
GPS	Ours	Schulze et al.	Ours	Schulze et al.	Ours	Schulze et al.	Ours	Schulze et al.
20.437	21.813	18.653	22.574	18.653	18.962	17.907	19.577	17.438
20.965	19.412	21.397	19.362	17.445	18.054	19.050	16.365	19.496
12.627	11.108	9.643	10.484	9.643	7.921	9.643	8.862	9.873
20.721	20.846	16.550	21.650	16.550	18.387	15.133	16.988	15.880
20.639	22.525	18.879	21.668	18.100	19.156	17.598	18.249	16.983

map-matching route using our approach below the true length in the worst case with ∼1.5km, while Schulze et al. produce routes less in length in a range of 2 to 5km. These values can be confirmed using the ratio of the decoded routes to the ground-truth presented in Figure 14. At a sampling rate of 3min, our approach decodes on average route of the same length as the true paths with a ratio of more than 103%, but Schulze et al. results in shorter routes. Increasing the sampling rate reduces the gap in the decoded routes length to be almost equal at 30min sampling rate.

—*Execution time*: Map-matching approaches use only time stamped cellular locations, without any further information from the mobile sensors or intermediate locations error estimation, targeting cellular providers CDRs big-data. CDRs datasets are in order of millions of records as compared to the individual's mobile data probed, which require intensive computational resources. This makes the execution time of a proposed method a crucial issue.

As explained in detail in Section 5, the execution time of the kernel algorithm in our proposed method is critical especially with a large number of states and observations. Nevertheless, exploiting the data sparsity gives us the opportunity to optimize Viterbi algorithm without losing its accuracy, and provides the exact optimal solution in linear time complexity. Schulze et al. perform an expensive search graph construction and start/end road segments selection. Search transition graph is used to limit the choice of road segments to these segments belonging to the cellular observations coverage areas.

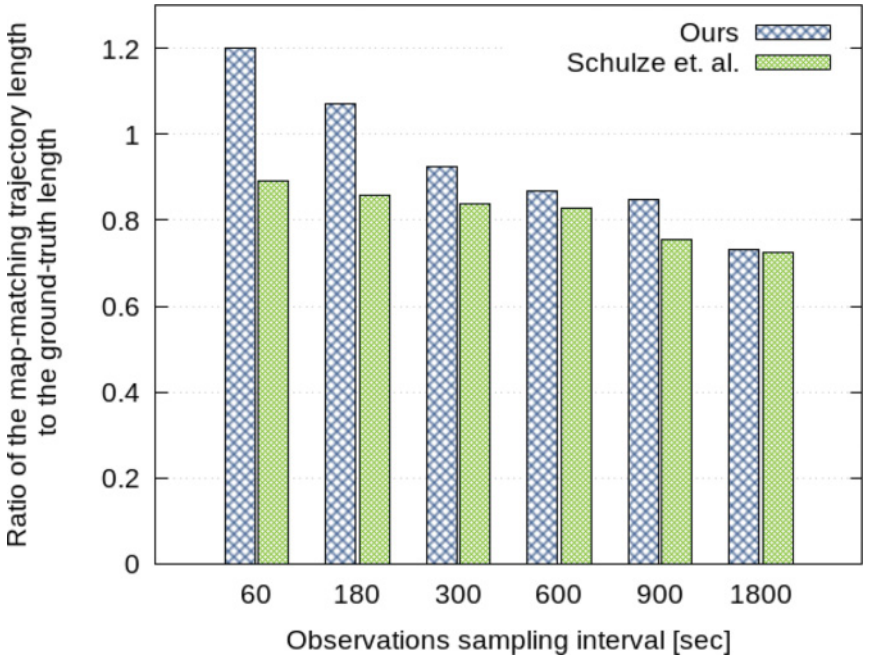


Fig. 14. Commuting distances ratios comparison, of the cellular-based trajectories to the ground-truth GPS traces, between our HMM-based map-matching probabilistic approach and Schulze et al.'s algorithm.

The graph construction is based on adding road segments belonging to a corridor covering the area of each two consecutive observations in a time frame, for example, for observation length L, and z cellular base stations, and n road segments belong to a map portion, the transition graph construction is $O(Lnz)$. The newly constructed transition graph has k road segment where $k < n$.

Graph cut is accompanied by selecting the start and end segments of an individual's trips, for example, m chosen road segments close to the start/end observation base station and each search takes $O(klg(k))$; the search part is $O(m^2 \times klg(k))$. All of these operations per single trajectory make it slower than ours as indicated in Figure 15. At the smallest radius of transmission, the best slow down of Schulze et al. to ours is 107× increase with increasing the sampling rate from 1 to 30min to be 685×. Extending the radius of transmission widens the gap between us to be 1296× at 5km radius of transmission.

Finally, training influences individual's routes. The adapted training procedure tends to put high probabilistic weights on the most frequently used road segments. Consequently, some routes may be slightly shifted from routes decoded using the original HMM-based map-matching. This behavior enhances a number of the decoded routes precision and may negatively cause a shift to a path away from actual segments.

As expected, as shown in Figure 16(a), trajectory segments precision of some routes are enhanced such as routes 1, 9, 11, and 12; while others are deviated from the actual routes. Results show 52%, 45%, and 48% in precision, recall, and F-measure, on average, respectively (in comparison with 70%, 84%, 76%, respectively, for the not trained case). The overall precision and hence the accuracy metric has been reduced by 8%, as demonstrated by Figure 16(b).

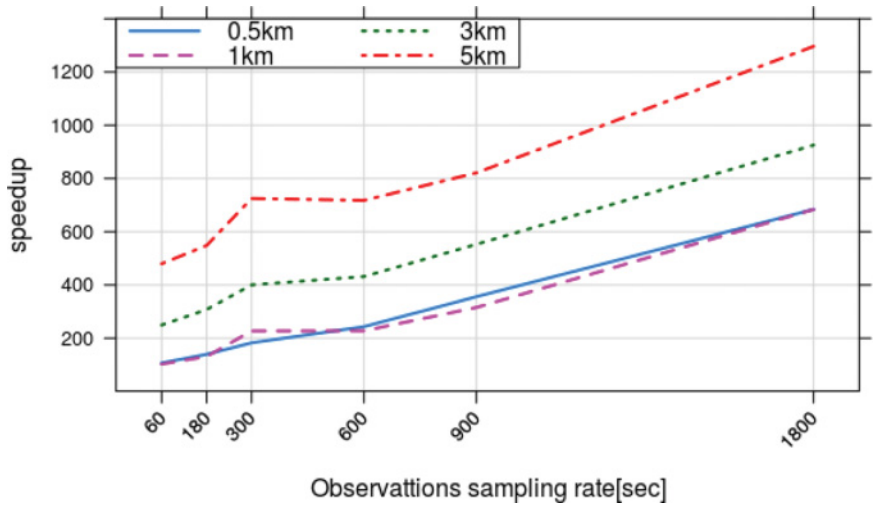
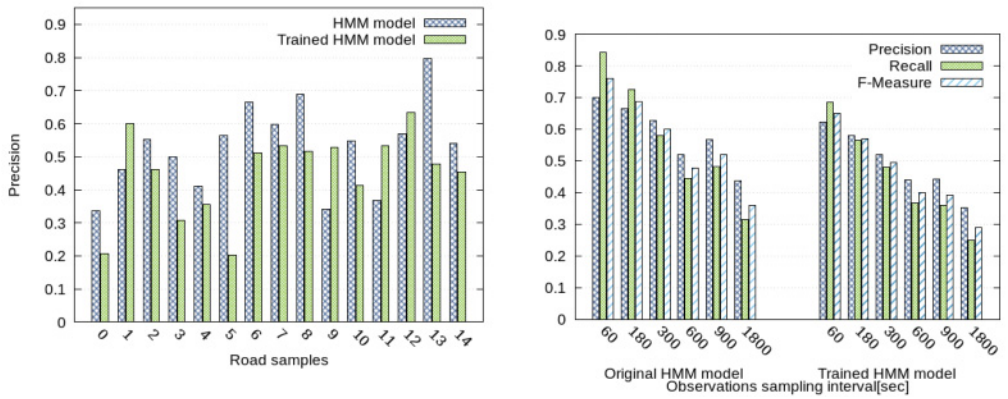


Fig. 15. Performance of the proposed HMM-based map-matching decoding speedup as compared to Schulze et al.'s algorithm.



(a) Small-scale map-matching samples' alignment ratio comparison using the original HMM model and a trained model at 10*min* sampling rate

(b) Sampling rates impacts on both trained HMM model and the original model in terms of the accuracy metrics

Fig. 16. Accuracy of the proposed HMM-based map-matching vs. its trained model at varying sampling rates and routes.

One reason behind this mostly negative training result might be attributed to the fact that training is done with no ground truth (not labelled data); training maximized a system metric (which as aggregate traffic flows), which results in a better system metric result (as described in the next subsection), rather than individual trajectories.

This leaves us to train the proposed model with inaccurate routes, in the best case of guaranteed 10min observations, regarding the CDRs dataset; results show 52%, 45%, and 48% in precision, recall, and F-measure, on average, respectively (in comparison with 70%, 84%, 76%, respectively, for the not trained case).

6.2. Aggregated Traffic Flow

Validating the obtained aggregated traffic flow is especially difficult with an absence of the ground truth (i.e., correct labels), which is typical in mobile big data problems. We, therefore, conducted an aggregate analysis using the following:

—the correlation between the widely-used gravity model and the CDRs ODs extracted using the modified stop detection method was investigated [Calabrese et al. 2011; Berlingerio et al. 2013];
—the correlation between road-segments flow estimated using HMM and the widely-used user equilibrium traffic model was investigated for validating the aggregate trip trajectory mapping using identical ODs [Jahn et al. 2005]; and
—the road-segment flows obtained using the proposed HMM was compared with Google Earth history images for a randomly selected set of road segments from four different regions within Dakar, Senegal.

Extracting the OD flows can provide the capacity to gain an accurate deep intuitive understanding of the mobility within a city. We, therefore, validate the extracted OD flows by comparing it with the basic traffic model used in transportation (gravity model).

The gravity model is defined as Berlingerio et al. [2013]:

$$\text{Gravity}(O, D) = \frac{O_{\text{out}} * D_{\text{in}}}{\text{Distance}(O, D)^2}, \tag{17}$$

where O and D denote two zones corresponding to the origin as the starting point of an individual trip and the destination of the trip with $O \neq D$; O_{out} is the cumulative number of vehicles going out of the zone O; D_{in} is the cumulative number of vehicles going in D; and Distance(O, D) is the shortest path distance from the designated origin O to the destination D.

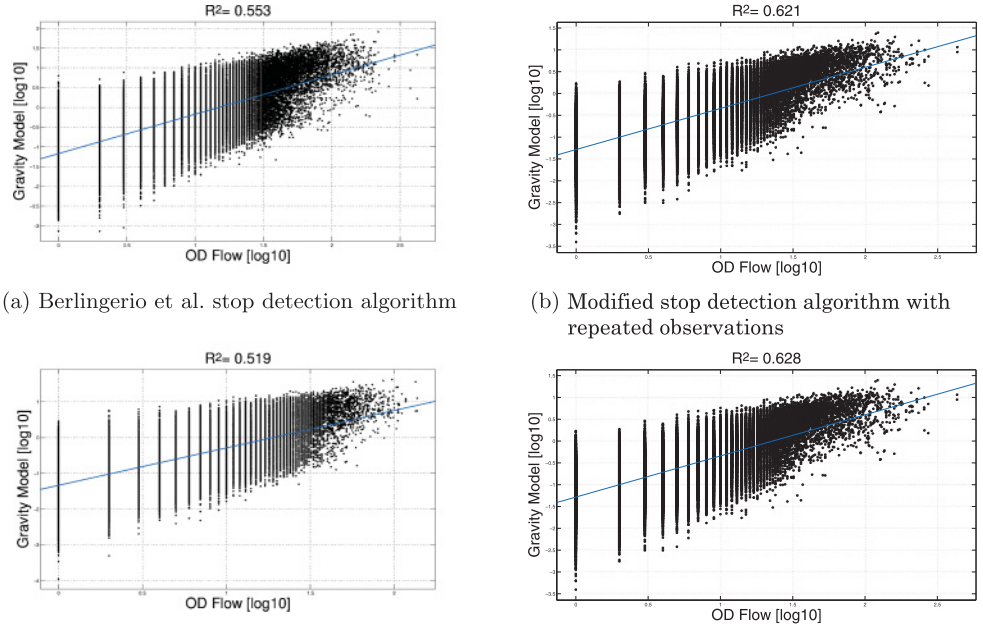
Figure 17 explains correlation comparison of the estimated OD flows versus gravity model using our modified stops detection algorithm as compared to Berlingerio et al.'s stops detection algorithm. It compares between these algorithms at with/out the presence of repeated observations. As shown in Figure 17(a), Berlingerio et al.'s ODs correlation with the Gravity model is $r^2 = 0.55$, which is very close to that reported values in their work [Berlingerio et al. 2013]. But eliminate the repeated observation negatively affects its correlation values; it reduces it to be 0.52.

In the other hand, our algorithm overweight Berlingerio et al.'s algorithm with and without repeated observations, as shown in Figures 17(b) and 17(d), with correlation ratios 0.621 and 0.628, respectively. Moreover, our algorithm shows a high consistency aginst the presence of repeated observation. Our algorithm's detected ODs correlation with the widely used Gravity model increased with a small fraction $+0.7\%$, while Berlingerio et al.'s correlation reduced by -3.4%.

The proposed model has been validated on the basis of the user equilibrium standard traffic-flow model. User equilibrium is one of the most popular route assignment models, typically used to predict traffic routes, given ODs [Jahn et al. 2005]. In this model, a commuter follows the route with the least possible commute time, taking into account earlier assignments.

We have applied our HMM to the extracted trips; around $\sim$40% of the trips are decoded using Viterbi algorithm; the others apparently require more relaxation for model parameters, which is subject for future work.

Figure 18(a) shows the cumulative probabilities of vehicles traveling per road edge obtained by the user equilibrium traffic model (obtained from a standard module of the SUMO traffic simulator) and the individual trajectories calculated by the proposed



(a) Berlingerio et al. stop detection algorithm

(b) Modified stop detection algorithm with repeated observations

(c) Berlingerio et al. stop detection algorithm

(d) Modified stop detection algorithm without repeated observations

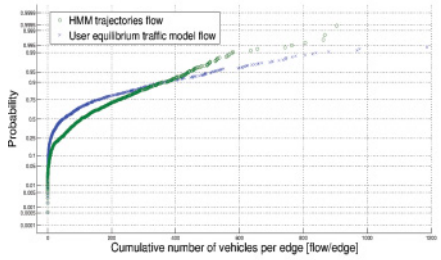
Fig. 17. Correlation comparison of the estimated OD flows vs. gravity model using our modified stops detection algorithm as compared to Berlingerio et al.'s stops detection algorithm. Panels (a) and (b) show the correlation at the presence of repeated observations, while panels (c) and (d) show the estimated correlation with eliminated repeated observation.

HMM model. This figure shows almost similar distribution of vehicles per road edge, which implies a high correlation between these methods.

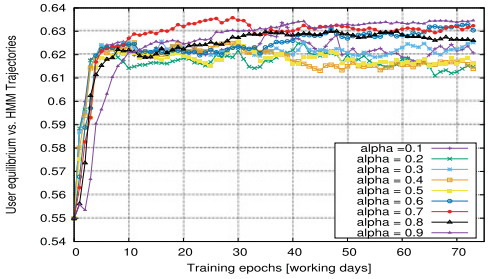
As shown in Figure 18(c), the proposed model provides a correlation of $r^2 = 0.562$ with the standard user equilibrium traffic-flow model at the confident interval of 0.5245 to 0.5745. This correlation's values gradually increase with the training epochs till saturation is around 35 epochs with the interpolation coefficent of $\alpha = 0.8$ and $\alpha = 0.9$, as shown in Figure 18(b). The correlation between the trained HMM-based traffic model and user equilibrium, shown in Figure 18(d), becomes $r^2 = 0.639$ at the saturation point of the training at confident interval estimation between 0.618 to 0.6417.

The average daily HMM flow calculated with the proposed system is visually compared with user equilibrium with ODs for the same period in Figure 19. This figure shows a strong correlation between the flows of estimated trajectories and those of user equilibrium model. Furthermore, the average number of vehicles per edges in both are very close, for the HMM-based vehicle density, average flow is 135.48, and using the same OD flow with user equilibrium traffic-flow assignment, the average number is 120.56 over one-month workdays.

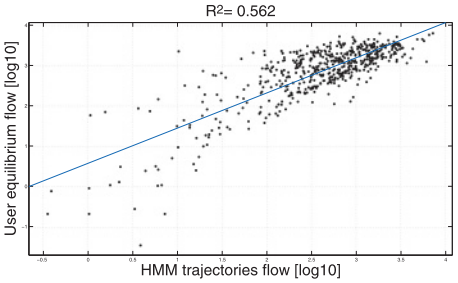
Finally, we compare the number of vehicles obtained using the proposed system and Google Earth history images for a randomly selected set of road segments from five different regions (Biscuterie, Grand Dakar, Castor, Cite Millionaire Hlm Patte doie, and Sicap dieupeul) within Dakar, Senegal. The number of vehicles is calculated at the imaging time of Google Earth images, which is 9 AM for all Google images history available for this region during 2013. Table V lists vehicle densities using Google's airborne images history and HMM vehicle densities. The vehicles count shows a high



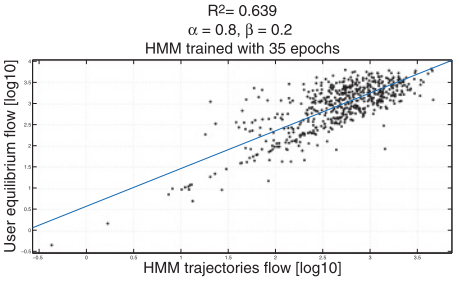
(a) Cumulative probabilities of vehicles passing the edge as a function of a number of vehicles per edge.



(b) Effect of varying interpolation coefficients as a function of training epochs.



(c) Correlation between user equilibrium traffic model flow and traveling individuals trajectories calculated by proposed system.



(d) Correlation between user equilibrium traffic model flow and traveling individuals trajectories calculated by proposed system.

Fig. 18. Correlation comparison of the estimated routes using our probabilistic map-matching approach and the user equilibrium traffic flow model.

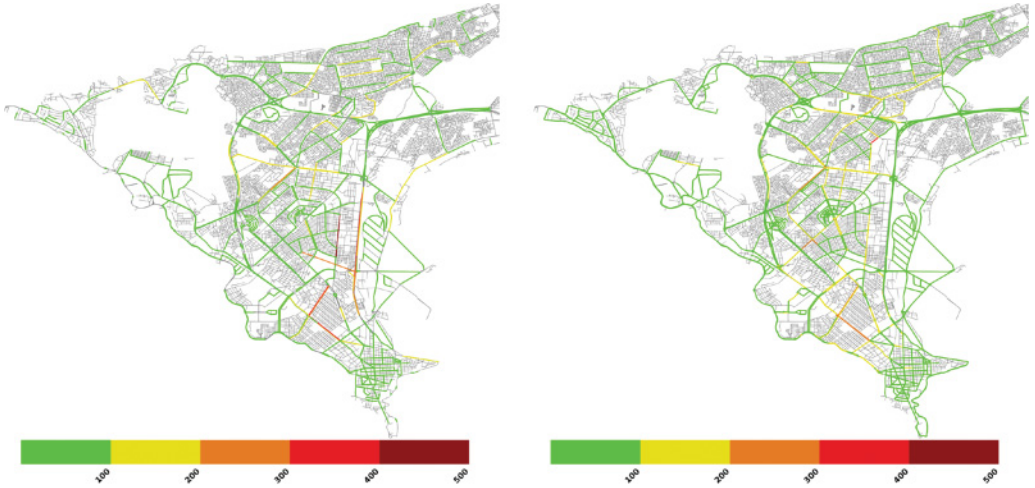


Fig. 19. Average daily vehicle densities for trained HMM-based traffic model with 35 epochs and average daily user equilibrium traffic model for one month workdays. Green color indicates light traffic flow; as the color goes darker, the traffic flow per road-edge becomes higher.

Table V. Vehicle Density Using Google Airborne Images History and HMM Vehicle Densities. The HMM Vehicle Are Counted at Same Imaging Time of Google Earth Maps, Which Is 9 AM

Street/Counting method	8/2/2013		20/10/2013		7/11/2013		22/12/2013	
	Google	HMM	Google	HMM	Google	HMM	Google	HMM
Avenu Cheikh Ahmudou Mbacke	54	44	24	10	59	24	20	32
Rue Serigne Fallou Mbacke	37	23	27	9	24	7	29	16
Allees Cheikh Sidaty Aidara	57	23	59	17	54	16	43	9
Boulevard Dial Diop	24	16	11	9	27	15	18	5
Allees Seydou Nourou Tall	16	24	5	8	13	32	7	8
Route du front de terre	63	16	49	32	174	40	50	24
Av. El-hadj Monsour	44	15	17	8	64	30	36	5
Route des Niays	31	24	13	15	27	24	28	8
Rue GY-476	34	31	19	16	53	40	22	23

correlation between values obtained by the proposed system with CDRs and Google maps with $r^2 = 0.4989$ for all streets with a 95% confidence interval of 0.2052 to 0.7116 as shown in Table V.

7. CONCLUSION

This article proposes an adaptive HMM-based model for map matching individual mobile phone trajectories to road segments, crossing the BTS zones. The model allows for fine-grain map matching using by solely using mobile big data for the first time. The model shows that stationary state transitions probabilities can be used to model the traffic phenomena taking into consideration the sparsity in time and space intrinsic to the mobile trajectories. Moreover, a simple maximum likelihood estimator further adapts probabilities and improves overall accuracy. The article also proposes a fast Viterbi decoding algorithm that exploits sparsity in transition probabilities, allowing for linear-time complexity and real-time performance for large-scale mobile data. The method studies a real mobile trajectories data set provided by the D4D challenge, for the city of Dakar, Senegal. Results indicate a high correlation with various traffic models and manual car counts from Google Maps for a sample of roads captured during the dataset time frame. Moreover, faster than real-time processing time is achieved with three orders of magnitude speedup over existing Viterbi implementations.

Future work will consider time-variant state transition probabilities, as well as traffic speeds and road types in model generation and analysis; and, also, more complex state representation and learning methods (especially for coping with seasonality in mobility patterns). Another potential future work is to consider scaling in terms of the mobile data size, roads (including smaller road types), and the required processing power; the use of hardware accelerators (such as graphics processing units) is thus highly sought for real-time processing at such a scale. Moreover, future work can include conducting experiments with many users participating to providing GPS location information and CDRs for more accurate validation at the (single) trip level.

ACKNOWLEDGMENTS

The authors thank Dr. Hisham El-Shishiny, from IBM Center for Advanced Studies in Cairo, for fruitful discussions. This research is partially supported by a PhD scholarship from the Egyptian Ministry of Higher Education (MoHE). The CDR data used in this work was made available by ORANGE/SONATEL within the framework of the D4D Challenge 2015.

REFERENCES

World Telecommunication/ICT Indicators Database. 2014. Retrieved from http://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx (2014). Accessed April 2, 2015.

Rein Ahas, Siiri Silm, Olle Järv, Erki Saluveer, and Margus Tiru. 2010. Using mobile positioning data to model locations meaningful to users of mobile phones. *J. Urban Technol.* 17, 1 (2010), 3–27.

Armen Allahverdyan and Aram Galstyan. 2011. Comparative analysis of viterbi training and maximum likelihood estimation for HMMS. In *Proceedings of the 2011 Conference on Advances in Neural Information Processing Systems*. 1674–1682.

Heba Aly and Moustafa Youssef. 2015. semMatch: Road semantics-based accurate map matching for challenging positioning data. arXiv:1510.03533 (2015).

L. R. Bahl, Peter F. Brown, Peter V. De Souza, and Robert L. Mercer. 1986. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP'86)*, Vol. 86. 49–52.

Richard Becker, Ramn Cceres, Karrie Hanson, Sibren Isaacman, Ji Meng Loh, Margaret Martonosi, James Rowland, Simon Urbanek, Alexander Varshavsky, and Chris Volinsky. 2013. Human mobility characterization from cellular network data. *Communications of the ACM* 56, 1 (Jan 2013), 74–82. DOI:http://dx.doi.org/10.1145/2398356.2398375

R. A. Becker, R. Caceres, K. Hanson, J. M. Loh, and S. Urbanek. 2011. Route classification using cellular handoff patterns. In *Proceedings of the 2011 Conference on Ubiquitous Computing (UbiComp'11)*. 123–132.

M. Berlingerio, F. Calabrese, G. Di Lorenzo, R. Nair, F. Pinelli, and M. L. Sbodio. 2013. AllAboard: A system for exploring urban mobility and optimizing public transport using cellphone data. In *Proceedings of the 2013 Conference on Mobile Phone Data of Development Analysis of Mobile Datasets for Development for Ivory Coast, 1–3 May 2013*.

Noelia Caceres, Luis M. Romero, Francisco G. Benitez, and Jose M. del Castillo. 2012. Traffic flow estimation models using cellular phone data. *IEEE Trans. Intell. Transport. Syst.* 13, 3 (2012), 1430–1441.

Francesco Calabrese, Giusy Di Lorenzo, Liang Liu, and Carlo Ratti. 2011. Estimating origin-destination flows using mobile phone location data. *IEEE Pervas. Comput.* 10, 4 (2011), 0036–44.

F. Calabrese, L. Ferrari, and V. D. Blonde. 2014. Urban sensing using mobile phone network data: A survey of research. *ACM Comput. Surveys* 47, 2 (2014).

Bi Yu Chen, Hui Yuan, Qingquan Li, William H. K. Lam, Shih-Lung Shaw, and Ke Yan. 2014. Map-matching algorithm for large-scale low-frequency floating car data. *Int. J. Geogr. Info. Sci.* 28, 1 (2014), 22–38.

Edwin De Jonge, Merijn van Pelt, and Marko Roos. 2012. Time patterns, geospatial clustering and mobility statistics based on mobile phone network data. In *Paper for the Federal Committee on Statistical Methodology Research Conference, Washington*. Retrieved from https://fcsm.sites.usa.gov/files/2014/05/vanPelt_2012FCSM_VII-C.pdf.

Yves-Alexandre de Montjoye, Csar A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports* 3 (Mar 2013), 1376. DOI:http://dx.doi.org/10.1038/srep01376

Yves-Alexandre de Montjoye, Zbigniew Smoreda, Romain Trinquart, Cezary Ziemlicki, and Vincent D. Blondel. 2014. D4D-Senegal: The second mobile phone data for development challenge. arXiv:1407.4885 (2014).

S. R. Eddy and others. 2007. HMMER-biosequence analysis using profile hidden Markov models. Retrieved from http://hmmer.janelia.org (2007).

Sean R. Eddy. 1996a. Hidden Markov models. *Curr. Opin. Struct. Biol.* 6, 3 (1996), 361–365.

S. R. Eddy. 1996b. Hidden Markov models. *Curr. Opin. Struct. Biol.* 6, 3 (June 1996), 361–365.

Sean R. Eddy. 1998. Profile hidden Markov models. *Bioinformatics* 14, 9 (1998), 755–763.

G. David Forney. 1973. The viterbi algorithm. *Proc. IEEE* 61, 3 (1973), 268–278.

Dariu M. Gavrila. 1999. The visual analysis of human movement: A survey. *Comput. Vision Image Understand.* 73, 1 (1999), 82–98.

Fabien Girardin, Francesco Calabrese, Filippo Dal Fiore, Carlo Ratti, and Josep Blat. 2008. Digital footprinting: Uncovering tourists with user-generated content. *IEEE Pervas. Comput.* 7, 4 (2008), 36–43.

M. C. Gonzales, C. A. Hidalgo, and A.-L. Barabasi. 2008. Understanding individual human mobility patterns. *Nature* 453 (2008), 779–782.

Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *Nature* 453, 7196 (2008), 779–782.

Sahar Hoteit, Stefano Secci, Stanislav Sobolevsky, Guy Pujolle, and Carlo Ratti. 2013. Estimating real human trajectories through mobile phone data. In *Proccedings of the 2013 IEEE 14th International Conference on Mobile Data Management (MDM'13)*, Vol. 2. IEEE, 148–153.

Xuedong D. Huang, Yasuo Ariki, and Mervyn A. Jack. 1990. *Hidden Markov Models for Speech Recognition*. Vol. 2004. Edinburgh University Press, Edinburgh.

Richard Hughey and Anders Krogh. 1996. Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *Comput. Appl. Biosci. (CABIOS)* 12, 2 (1996), 95–107.

Sibren Isaacman, Richard Becker, Ramón Cáceres, Stephen Kobourov, Margaret Martonosi, James Rowland, and Alexander Varshavsky. 2011. Identifying important places in peoples lives from cellular network data. In *Pervasive Computing*. Springer, 133–151.

Sibren Isaacman, Richard Becker, Ramón Cáceres, Margaret Martonosi, James Rowland, Alexander Varshavsky, and Walter Willinger. 2012. Human mobility modeling at metropolitan scales. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*. ACM, 239–252.

O. Jahn, R. H. Mohring, A. S. Schulz, and N. E. Stier-Moses. 2005. System-optimal routing of traffic flows with user constraints in networks with congestion. *Operat. Res.* 53, 4 (Aug. 2005), 600–616.

Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. 2000. Traffic monitoring and accident detection at intersections. *IEEE Trans. Intell. Transport. Syst.* 1, 2 (2000), 108–118.

Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjölander, and David Haussler. 1994. Hidden Markov models in computational biology: Applications to protein modeling. *J. Mol. Biol.* 235, 5 (1994), 1501–1531.

J. Krumm, J. Letchner, and E. Horvitz. 2007. Map matching with travel time constraints. In *Proceedings of the 2007 SAE Conference*.

Xiaoqiang Li, Wenting Han, Gu Liu, Hong An, Mu Xu, Wei Zhou, and Qi Li. 2012. A speculative HMMER search implementation on GPU. In *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW'12)*. IEEE, 735–741.

Xiaokun Li and Fatih M. Porikli. 2004. A hidden Markov model framework for traffic event detection using video features. In *Proceedings of the 2004 International Conference on Image Processing (ICIP'04)*, Vol. 5. IEEE, 2901–2904.

Chuan Liu. 2009. cuHMM: A CUDA implementation of hidden Markov model training and classification. *The Chronicle of Higher Education* (2009).

James H. Martin and Daniel Jurafsky. 2000. Speech and language processing. *International Edition* (2000).

Reham Mohamed, Heba Aly, and Moustafa Youssef. 2014. Accurate and efficient map matching for challenging environments. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 401–404.

Reham Mohamed, Heba Aly, and Moustafa Youssef. 2016. Accurate real-time map matching for challenging environments. *IEEE Trans. Intell. Transport. Syst.* PP, 99 (2016), 1–11. DOI:http://dx.doi.org/10.1109/TITS.2016.2591958

Diala Naboulsi, Marco Fiore, Stephane Ribot, and Razvan Stanica. 2015. Large-scale mobile traffic analysis: A survey. *IEEE Commun. Surv. Tutor.* 18, 1 (2015), 124–161.

Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 336–343.

J. Nielsen and A. Sand. 2011. Algorithms for a parallel implementation of hidden Markov models with a small state space. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW'11)*. 452–459. DOI:http://dx.doi.org/10.1109/IPDPS.2011.181

Paola Pucci, Fabio Manfredini, and Paolo Tagliolato. 2015. *Mapping Urban Practices Through Mobile Phone Data*. Springer.

Lawrence Rabiner and Biing-Hwang Juang. 1993. Fundamentals of speech recognition (1993). PTR Prentice Hall.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.

Carlo Ratti, Riccardo Maria Pulselli, Sarah Williams, and Dennis Frenchman. 2006. Mobile landscapes: Using location data from cell phones for urban analysis. *Environment and Planning B: Urban Analytics and City* 33, 5 (2006), 727–748. DOI:http://dx.doi.org/10.1068/b32047

Y. Rubner, C. Tomasi, and L. J. Guibas. 2000. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision* 40 (Nov. 2000), 99–121.

Andreas Sand, Martin Kristiansen, Christian N. S. Pedersen, and Thomas Mailund. 2013. zipHMMlib: A highly optimised HMM library exploiting repetitions in the input to speed up the forward algorithm. *BMC Bioinformat.* 14, 1 (2013), 339.

A. Sand, C. N. S. Pedersen, T. Mailund, and A. T. Brask. 2010. HMMlib: A C++ library for general hidden Markov models exploiting modern CPUs. In *Proceedings of the 2nd International Workshop on Parallel and Distributed Methods in Verification, 2010 9th International Workshop on High Performance Computational Systems Biology*. 126–134. DOI:http://dx.doi.org/10.1109/PDMC-HiBi.2010.24

Alexander Schliep, Wasinee Rungsarityotin, Alexander Schnhuth, and Benjamin Georgi. 2006. *The General Hidden Markov Model Library: Analyzing Systems with Unobservable States. Proceedings of the Heinzbilling-price.*

Gunnar Schulze, Christopher Horn, and Roman Kern. 2015. Map-matching cell phone trajectories of low spatial and temporal accuracy. In *Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2707–2714.

Stefania-Iuliana Soiman, Irena Rusu, and Stefan-Gheorghe Pentiuc. 2014. A parallel accelerated approach of HMM forward algorithm for IBM roadrunner clusters. In *Proceedings of the 2014 International Conference on Development and Application Systems (DAS'14)*. IEEE, 184–188.

A. Thiagarajan, L. S. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod. 2011. Accurate, low-energy trajectory mapping for mobile devices. In *Proceedings of the 2011 NSDI*.

A. Thiagarajan, L. Sivalingam, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, and H. Balakrishnan. 2009. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 2009 SenSys*.

V. A. Traag, A. Browet, F. Calabrese, and F. Morlot. 2011. Social event detection in massive mobile phone data using probabilistic location inference. In *Proceedings of the 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust (PASSAT'11) and 2011 IEEE Third International Conference on Social Computing (SocialCom'11)*. 625–628. DOI:http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.133

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Info. Theory* 13, 2 (1967), 260–269.

Pu Wang, Timothy Hunter, Alexandre M. Bayen, Katja Schechtner, and Marta C. Gonzlez. 2012. Understanding road usage patterns in urban areas. *Scientific Reports* 2 (Dec 2012), Article No. 1001. DOI:http://dx.doi.org/10.1038/srep01001

Yan Xiao-Yong, Han Xiao-Pu, Zhou Tao, and Wang Bing-Hong. 2011. Exact solution of the gyration radius of an individual's trajectory for a simplified human regular mobility model. *Chinese Phys. Lett.* 28, 12 (2011), 120506.

Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. 2012. Reducing uncertainty of low-sampling-rate trajectories. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering (ICDE'12)*. IEEE Computer Society, Washington, DC, 1144–1155.

Yu Zheng. 2015. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol. (TIST)* 6, 3 (2015), 29.

Received October 2015; revised September 2016; accepted January 2017