Inferring Uncertain Trajectories from Partial Observations

Prithu Banerjee
IBM Research
Manyata Tech Park
Bangalore, India
prithuba@in.ibm.com

Sayan Ranu
Dept. of CSE
IIT Madras
Chennai, India
sayan@cse.iitm.ac.in

Sriram Raghavan
IBM Research
Manyata Tech Park
Bangalore, India
sriramraghavan@in.ibm.com

***Abstract*—The explosion in the availability of GPS-enabled devices has resulted in an abundance of trajectory data. In reality, however, majority of these trajectories are collected at a low sampling rate and only provide partial observations on their actually traversed routes. Consequently, they are mired with uncertainty. In this paper, we develop a technique called *InferTra* to infer *uncertain trajectories* from network-constrained partial observations. Rather than predicting the most likely route, the inferred uncertain trajectory takes the form of an edge-weighted graph and summarizes all probable routes in a holistic manner. For trajectory inference, InferTra employs Gibbs sampling by learning a *Network Mobility Model (NMM)* from a database of historical trajectories. Extensive experiments on real trajectory databases show that the graph-based approach of InferTra is up to 50% more accurate, 20 times faster, and immensely more versatile than state-of-the-art techniques.**

I. Introduction

The last decade has witnessed an unprecedented growth in the availability of location-tracking technologies, which can be deployed at large scales to collect trajectory data. However, these trajectories are often recorded at a *low sampling rate* wherein the time interval between two consecutive recorded locations is large. As a result, these trajectories only provide partial observations of the actual traversed route and the intermediate portions remain hidden.

Trajectories can be tracked most accurately through gps-enabled devices such as cell-phones or in-car navigations systems. However, a recent work has shown that to reduce power consumption, majority of the taxis in big cities have a sampling interval exceeding two minutes [1]. The high power consumption of GPS also limits its usage on cell phones over large continuous durations. In the absence of GPS, location of a cell-phone can also be tracked through call detail records (*CDR*) [2], which stores the sequence of base-stations through which a call or data-usage session is routed. However, the problem of low sampling remains since CDRs are generated only when either a call or a data session is in progress.

High power consumption of GPS is not the sole reason behind low sampling rates. Most social networks today provide "check-in" services to announce and share location of a user (e.g., Facebook). Trajectories can be generated from these check-ins by ordering them temporally [3]. Similar trajectories also arise from geo-tagged photos in photo-sharing sites (e.g., Flickr), credit card transactions and snapshots of vehicles captured through surveillance cameras. Due to the inherent properties of the underlying applications, all of these trajectories are generated at low sampling frequencies.

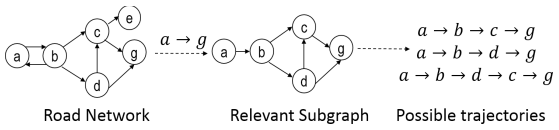


Fig. 1. Demonstrates the possible acyclic trajectories arising out of the partial observation $a \rightarrow g$.

To understand the uncertainty surrounding low sampling rates, consider the example shown in Fig. 1. Hereon, we use the term *observations* to indicate a low sampled trajectory, and the term *trajectory* is used to denote the complete sequence of nodes that is actually traversed. The first directed graph shown in Fig. 1 depicts a road network where each node represents a region and an edge corresponds to the road segment connecting two regions. Now, consider the partial observation $a \rightarrow g$. As it can be seen, there is no direct edge from a to g. Assuming that trajectories are acyclic, the relevant subgraph in Fig. 1 shows the region within which the mobility is constrained and any of the three paths connecting a and g is a possible trajectory that was actually traversed.

Managing the uncertainty highlighted in Fig. 1 is critical towards designing accurate higher-order systems that are driven by trajectory data. For example, trajectories from geo-tagged photos can be inferred for trip-mining [4] and used for recommending tourist itineraries. Beyond inferring the most likely trajectory, it is also essential to infer regions which have a high likelihood of being traversed. For example, investigative agencies are often interested in retrospective analysis of movements of suspected criminals based on their spatio-temporal footprints generated from CDR data, credit card transactions, surveillance camera snapshots, etc. [5]. Consider a bomb blast that occurred at 9 PM. Investigative agencies are interested in identifying suspected terrorists that were present in the vicinity of the blast site around 9 PM with a high likelihood. For suspects who are mobile, this query cannot be answered using existing mechanisms. Note that the most likely region may not necessarily be part of the most likely trajectory. In essence,

1550-4786/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDM.2014.41

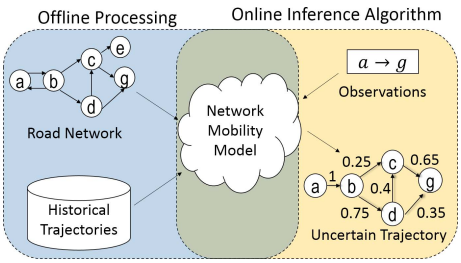


Fig. 2. Pipeline of the *InferTra* algorithm.

we want to capture the entire uncertainty surrounding partial observations, which would allow us to answer the following interesting questions.

- Which is the most likely trajectory?
- What are the top-k most likely road segments?
- What are the top-k most likely locations at time t?

In this paper, we design a technique called *InferTra (INFERring TRAjectories)* to answer all of the above questions in a principled manner. Fig. 2 outlines the pipeline of InferTra. Against the backdrop of a road network and a database of historical trajectories, InferTra learns a Network Mobility Model (*NMM*), which is then used to predict an "uncertain" trajectory. In contrast to existing techniques [6], an uncertain trajectory is an edge-weighted graph. An edge weight denotes the probability of the corresponding road segment being traversed. Overall, the graph summarizes the uncertainty around each possible trajectory arising from the partial observations. Using a graph to model the uncertainties is a significant deviation from the current state of the art [6]. While one could operate in the world of maximum likelihoods and predict the most likely trajectory, as the uncertainty grows, the information content in maximum likelihood estimations deteriorates. InferTra recognizes this issue and focuses on a more holistic analysis of the uncertainty that imparts both higher accuracy in predictions and the ability to answer a wider range of queries. To summarize, the paper makes the following contributions:

- We develop an algorithm called *InferTra* that infers an "uncertain" trajectory from a set of partial observations. Our graph-based approach provides a more holistic representation of uncertainty and thus enabling us to answer deeper questions than state-of-the-art techniques.
- To accomplish inference, we compute a generative model, called Network Mobility Model (NMM), by learning the mobility patterns in a road network from a database of historical trajectories. NMM not only captures the spatial patterns, but it is also the first technique to leverage the temporal signals.
- Empirical evaluations establish the graph-based approach in InferTra as immensely more versatile in answering a wider range of queries. In addition, InferTra is up to 50% more accurate and 20 times faster than HRIS [6].

II. Related Work

HRIS [6] is the first and the only work to infer network-constrained trajectories from partial observations. In this section, we outline how InferTra is different.

- **1. Inference Goals:** Given a set of input observations, HRIS predicts the most likely trajectory. Thus, it operates in the maximum likelihood world, where the likelihood of a trajectory is modeled using a "popularity" score. While the most likely trajectory is a good predictor under low uncertainties, as the uncertainty grows, the information content in the prediction deteriorates. For example, in the uncertain trajectory in Fig. 2, none of the possible trajectories have a high likelihood; rather, they are distributed across the entire relevant subgraph. It is therefore critical to understand how the possibilities that are not captured in the most likely trajectory are spread across the road network. InferTra recognizes this issue. Accordingly, the goal is to infer a single uncertain trajectory that summarizes all of the possibilities in a coherent manner. The uncertain trajectory not only allows us to identify the most likely trajectory, but also facilitates answering deeper questions such as the most likely location at time t, which may not necessarily be part of the most likely trajectory, and identification of all regions with a likelihood above a user-provided threshold. Due to this versatility of uncertain trajectories, a decent body of work already exists on querying uncertain trajectories [7], [8], which is complementary to our problem. Instead of inferring the uncertain trajectory, they assume a database of uncertain trajectories on which further processing is performed.
- **2. Capturing Historical Patterns:** HRIS proposes two techniques to connect a pair of consecutive observations. In the first technique, historical trajectories are used to extract a sub-network of the road network within which the mobility is assumed to be bounded. Next, to connect the partial observations, the shortest paths in the sub-network are computed. In the second technique, starting from the first observation, greedy choices are made to iteratively hop to a neighboring node and reach the destination observation. In InferTra, no assumptions, such as preference toward shortest paths, are made based on the network properties. Rather, spatio-temporal patterns displayed by the historical trajectories are learned and utilized in prediction. If indeed shortest paths are favored in certain regions, then this property is automatically learned from the historical trajectories itself.
- **3. Temporal Signals:** InferTra not only learns the spatial signals embedded in the historical trajectories, but also unearths the temporal signals, which are not utilized in HRIS.

More recently, a technique [1] was designed to study the trajectory inference problem in a setting where trajectories are not constrained by a network. Due to the focus on network-free trajectories, [1] is not applicable to our problem.

III. Problem Formulation

First, we define the concepts central to our paper.

***Definition** 1:* Road Network. *A road network is a directed graph $G(V, E)$. V is the set of nodes representing*

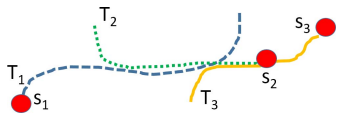


Fig. 3. A hypothetical inference scenario.

intersections and terminal points, and E is the set of edges $e = (v_i, v_j)$, connecting $v_i, v_j \in V$, depicting road segments. The position of a node is characterized by its latitude and longitude.

We use the notation $e.p_1$ and $e.p_2$ to denote edge e's two endpoints, where e is directed from p_1 to p_2 where p_1, $p_2 \in V$. Generally, a trajectory $T = \{s_1, \cdots, s_n\}$ is a temporally ordered sequence of spatio-temporal points. A spatio-temporal point $s = (v, t)$ is a tuple containing a spatial location v and a timestamp t encoding the time of the day at which v is traversed. We use the notation $T.s_i$ to denote the i_{th} spatio-temporal point in T, and $s.v$ and $s.t$ to denote the location and timestamp in s respectively. Indeed, $T.s_i.t < T.s_{i+1}.t$. In this work, we only consider *network-constrained* trajectories.

***Definition** 2:* NETWORK-CONSTRAINED TRAJECTORY. *T is constrained in a road network $G(V, E)$, if $\forall T.s_i$, $T.s_i.v \in V$, and $\forall T.s_i$, $T.s_{i+1}$, $(T.s_i.v, T.s_{i+1}.v) \in E$.*

In simple words, a network-constrained trajectory is a connected path in the road network. We assume that each edge (or road segment) in a trajectory takes it progressively closer to its destination, and thus a trajectory is acyclic. As illustrated in Fig. 1, a set of *partial observations*, is a sequence of network regions where a trajectory T, which is unknown, has been spotted.

***Definition** 3:* PARTIAL OBSERVATIONS. *A partial observation $O = \{s_1, \cdots, s_n\}$, is a temporally ordered sequence of spatio-temporal points where $\forall O.s_i$, $O.s_i.v \in V$.*

Given a set of partial observations O, our goal is to infer an uncertain trajectory that captures all possible trajectories arising from O, quantify the uncertainty associated with each possible trajectory, and capture how this uncertainty is distributed across the road network.

***Definition** 4:* UNCERTAIN TRAJECTORY INFERENCE. *Given a road network $G(V, E)$, a database of historical trajectories $\mathbb{H}$, and a set of partial observations O, construct an edge-weighted graph $U(V', E') \subseteq G(V, E)$, such that $E' = \{p(e|O) > 0 \mid e \in E\}$, where $p(e|O)$ denotes the probability of e being traversed given O. V' is defined analogously from the endpoints of edges in E'.*

We assume that the historical trajectories are clean and entirely observed.

To summarize our formulation, U contains all edges that have a non-zero traversal likelihood, and consequently, encapsulates all possible trajectories. The edge weights capture how the uncertainty is distributed across the road network. Mathematically, U is modeled as a multivariate distribution of its constituent edges, i.e., $p(U|O) = p(e_1, \cdots, e_m|O)$, where $\{e_1, \cdots, e_m\} \in U$. The edge weights are therefore the marginal distributions, which we need to learn from the observations in conjunction with the evidence provided by the historical data. More precisely, for any given edge $e \in U$,

$$P(e|O) = \sum_{\forall T \in \mathbb{T}} P(T|O) \tag{1}$$

where $\mathbb{T}$ is set of all trajectories (or paths) connecting the sequence of observed nodes in O and also containing, edge e. So, if we can compute $p(T|O)$ for all trajectories in U, then we can identify all edges e with $p(e|O) > 0$, and therefore, compute the uncertain trajectory corresponding to O.

IV. CHALLENGES

Let $O = \{s_1, s_2, s_3\}$ be the observation set and $T = \{e_1, \cdots, e_n\}$ be one path connecting all nodes in O. In its simplest form, $p(T|O)$ can be estimated directly from the historical trajectories. More specifically,

$$p(T|O) = \frac{\left\|\{T \subseteq T' | T' \in \mathbb{H}\}\right\|}{\left\|\{O \subseteq T' | T' \in \mathbb{H}\}\right\|} \tag{2}$$

The formulation in Eq. 2 works perfectly when the historical database is infinitely large, which, however, is not a realistic assumption. Consider the scenario in Fig. 3 to understand the limitations of Eq. 2 better. For $O = \{s_1, s_2, s_3\}$, no trajectory exists that covers all three regions, and thus, Eq. 2 cannot be employed for inference. However, it is easy to see that there are local signals embedded in the historical trajectories that can be consolidated to infer a possible route. More specifically, by stitching together overlapping sub-trajectories from T_1, T_2, and T_3 the likelihood of a path connecting s_1, s_2 and s_3 can be computed. Constructing such arbitrary trajectories from known ones is in fact how a human would draw inference when asked to connect a set of observations that have not been traveled in a single journey. Consider a car that has just crossed s_2 in Fig. 3 and still needs to find its way to s_3. To select the next edge, the driver of the car would ask the following question: *Given my recent past, and based on historical evidence, which road should I take next to maximize my chances of reaching s_3?* Thus, the fact that the car started from s_1 and there is no path in $\mathbb{H}$ connecting s_1 to s_3 has no impact on the decision choice between s_2 and s_3. What matters are the recent past and the current target node to reach, which is s_3.

This natural human tendency of taking locally optimal decisions to construct the globally optimal route can be mathematically expressed as the following. Let trajectory T be the sequence of edges $\{e_1, \cdots, e_n\}$ traveled till now. Then,

$$\begin{aligned} &p(e_n|e_1, \cdots, e_{n-1}, O) = \\ &\quad p(e_n|e_{n-m}, \cdots, e_{n-1}, s_i) \\ &\quad \approx \frac{\left\|\{(e_{n-m}, \cdots, e_n) \subseteq T | T \in \mathbb{H},\ s_i \in T\}\right\|}{\left\|\{(e_{n-m}, \cdots, e_{n-1}) \subseteq T | T \in \mathbb{H},\ s_i \in T\}\right\|} \end{aligned} \tag{3}$$

where m quantifies "recency" and $s_i \in O$ is an observed node, such that it is not present in T, but all its preceding observed nodes, have already been traversed by T, i.e.,

$\forall j,\ s_j \in O,\ 1 \leq j < i;\ s_j \in T$. Thus, Eq. 3 is simply the proportion of historical trajectories that share the same recent history as of e_n and have traveled through s_i. Now, notice that Eq. 3 allows us to compute the conditional distribution in an n-dimensional space if the m-dimensional joint distributions of the numerator and the denominator in Eq. 3 are known. Since $m \ll n$, estimating these m-dimensional joint distributions directly from a finite $\mathbb{H}$ is likely to be more accurate.

To formalize this intuition, let us define the notion of a *density* Δ, which is the ratio of representative samples to the volume of the space. If the density is above a certain threshold θ, then we can assume that the joint distribution sampled directly from the representative samples is accurate. Thus, the density for an n-dimensional joint distribution of a trajectory T computed directly from $\mathbb{H}$ is $\Delta_n = \frac{\|\mathbb{H}\|}{2^n}$. The volume is 2^n since there are n random variables and each variable can take two values: traversed or not traversed. So, to satisfy a given threshold θ, the number of representative samples needs to grow exponentially with the dimension of the space. Since $m \ll n$, satisfying this accuracy criteria is significantly easier in an m-dimensional space.

The above analysis suggests that, while we may not have the information to compute joint distribution $p(T|O)$ directly, the historical database may be enough to compute conditionals $p(e|T, O)$. Thus, well defined statistical techniques, such as *Gibbs Sampling* [9], can be used to approximate the joint distribution by sampling from the conditionals.

V. Gibbs Sampling for Trajectory Inference

Gibbs sampling (*GS*) is a generalized probabilistic inference algorithm that is used to generate a sequence of samples from a joint distribution of two or more random variables. The first requirement for GS is some observable data. Let us denote this observed data as Y. Next, GS requires a vector of random variables that are unknown to start with. Let us denote this n-dimensional vector as $\boldsymbol{\phi} = (\phi_1, \cdots, \phi_n)$. The goal of GS is to learn $\boldsymbol{\phi}$ to model the observable data. Towards that goal, GS follows the following iterative procedure.

1) Initialize each $\phi_i \in \boldsymbol{\phi}$ to some arbitrary value.
2) for $\tau = 1, \cdots, T$
3) for $i = 1, \cdots, n$
4) Sample $\phi_i^{\tau+1} \approx p(\phi_i | \phi_1^{\tau+1}, \cdots, \phi_{i-1}^{\tau+1}, \phi_{i+1}^{\tau}, \cdots, \phi_n^{\tau}, Y)$
5) Iterate over i and τ

In this process, the nested iteration assigns a value to each random variable by conditioning it on the current values of the remaining random variables and the observation set. The full execution of this inner loop computes a point in the n-dimensional space of the joint distribution $p(\boldsymbol{\phi})$. The outer loop repeats this same process T times to sample from the n-dimensional space repeatedly till the joint distribution converges. It has been shown [9] that the joint distribution $(\phi_1^{\tau}, \cdots, \phi_n^{\tau})$ converges geometrically to $p(\phi_1, \cdots, \phi_n|Y)$ as $T \to \infty$, and therein lies the power of GS.

A. Trajectory inference through Gibbs sampling

In our problem, Y corresponds to the observations O, and, $\boldsymbol{\phi}$ corresponds to the uncertain trajectory U. The edges in U correspond to the random variables. For a given O, the random variables can be identified by taking the union of edges in all paths that connect the nodes in O. Now, to employ GS in our problem, we first need to initialize the random variables. This can be achieved by setting each edge in U as either traversed or non-traversed. The conditionals can be computed as outlined in Eq. 3. However, one key difference from normal GS is that a trajectory is an ordered sequence of random variables. More precisely, the ordering at which each edge is set to be traversed is important and this ordering is used by the recency factor in Eq. 3. Thus, we need to maintain an additional variable to track the timestamp. Timestamp is set to 0 at the start of each iteration of the outer loop over τ. For each edge set as traversed in the inner loop, timestamp is incremented by 1.

The above steps complete the adaptation of GS for the trajectory inference problem. However, the following two aspects of the algorithm affect its scalability.

- **Identifying random variables:** The process to identify the random variables (or edges) by computing all paths connecting the observations is expensive.
- **Computing conditionals:** Computing the conditionals is expensive since we need to scan the entire historical database each time Eq. 3 is computed. Furthermore, this operation happens repeatedly till convergence of the GS.

The proposed algorithm, *InferTra*, removes the scalability bottlenecks while maintaining high accuracy.

VI. InferTra

InferTra operates in two phases: an offline learning phase to build a Network Mobility Model (NMM), and an online inferencing algorithm using the NMM.

A. Learning a Network Mobility Model

NMM is a generative model for trajectories and its task is to connect a set of observations (or nodes) without compromising on the mobility patterns of the historical trajectories. The set of all possible paths between a source node and a destination in a road network can be huge. In reality, however, vehicles show affinity towards a limited set of roads that form majority of the trajectories. These spatial and temporal patterns provide a rich characterization of trajectory movements, which InferTra learns through the NMM using a *higher-order Markov Model*.

The NMM is learned from two sources of input data: a road network $G(V, E)$, and a database of historical trajectories $\mathbb{H}$. In a Markov process, elements of the system make transitions from one state to another based on the preceding history. The length of the history, which dictates the state transitions, is known as the "order" of the model. For example, in the most commonly used 1st-order Markov process, the state transition of an element is influenced only by its current state. In the NMM, the state space is defined by nodes V of G, and transitions take place only through edges. The transition probabilities are learned from the database of historical trajectories.

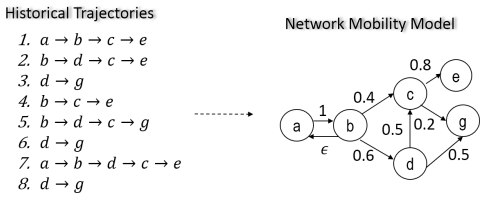


Fig. 4. The NMM for the shown trajectories constrained within the road network in Fig. 1.

The NMM's goal is to better understand the likelihood of a road segment being traversed based on the recent history of a vehicle and the time of the day. These segment traversal likelihoods are modeled as state transitions. Towards that end, we define the concept of an m-history sequence of a node $v \in V$, where m is the order of the Markovian process in the NMM and corresponds to the "recency" factor in Eq. 3.

***Definition** 5: The m-history sequence of a node v is a path $H = \{v_1, \cdots, v_p\}$ through p connected nodes of G such that $p \leq m$ and $v_p = v$.*

***Example** 1: For the road network in Fig. 1, node 'c' has three 2-history sequences: $b \rightarrow c$, $d \rightarrow c$, and c.*

***Definition** 6:* SPATIO-TEMPORAL CONTAINMENT. *H is said to be contained in trajectory T at time t, if $\forall i,\ 1 \leq i \leq |H|,\ H.v_i = T.s_{i+a}.v$, where $\exists a, 0 \leq a \leq (|T| - |H|)$. Additionally, $|t - T.s_{(|H|+a)}.t| \leq \delta$. This relationship is denoted using $H \in_t T$.*

More simply, $H \in_t T$ if H is a sub-sequence of T and the final destination in H is reached within δ time units from t. Incorporating the time of the day in the model allows us to capture the inherent periodicities in mobility patterns. δ is a user provided parameter. We discuss the semantics of δ below. Prior to that, we introduce the concept of *affinity* toward an edge e based on an m-history sequence H of $e.p_1$.

***Definition** 7:* EDGE AFFINITY. *The affinity $\alpha(e, H, t)$ of an edge e at time t with respect to an m-history H of $e.p_1$ is the probability of e being traversed by a trajectory T, given that $H \in_t T$. Formally,*

$$\alpha(e, H, t) = max\{P(H \cup \{e.p_2\} \in_t T | H \in_t T), \epsilon\}$$
$$= max\left\{\frac{|\{H \cup \{e.p_2\} \in_t T \mid T \in \mathbb{H}\}|}{|\{\forall e' \in E,\ e'.p_1 = e.p_1,\ H \cup \{e'\} \in T \mid T \in \mathbb{H}\}|}, \epsilon\right\}$$

where $\epsilon \approx 0$.

In essence, $\alpha(e, H, t)$ quantifies the transition probability from $e.p_1$ to $e.p_2$ based on the m-history H and timestamp t. The NMM is constructed by following this procedure. More specifically, a time window of δ is slid across all edges, and the corresponding affinities are computed. The distribution of affinities across timestamps is then stored at each edge. Generally, a window size in the range of $\delta = [30, 45]$ minutes produces consistent results.

***Example** 2: Fig. 4 demonstrates the NMM for the shown historical trajectories constrained within the network in Fig. 1. We assume $m = 1$ and all segments are traversed at the same timestamp for simplicity, . We thus ignore the temporal aspect.*

While the time of the day certainly influences the mobility pattern, storing the entire affinity distribution for each edge and m-history pair incurs a large storage cost. For instance, sliding a time window of 30 minutes would produce $24 * 60 - 30 = 1410$ affinity values for each pair. For a vast majority of the segments, the affinities remain constant with time, and storing the entire distribution promotes redundancy. To remove this redundancy, we partition these distributions into the optimum number of bins using the *Freedman-Diaconis* rule [10]. The Freedman-Diaconis rule states that the width w of each bin in the distribution $\alpha(e, t)$ should be:

$$w = 2\frac{IQR(\alpha(e, H, t))}{n^{\frac{1}{3}}} \tag{4}$$

where n is the number of time windows, and $IQR(\alpha(e, H, t))$ denotes the *interquartile range* of the affinity distribution at e for m-history H. The interquartile range is a measure of the statistical dispersion of a distribution and is equal to the difference between the third and the first quartile, i.e., the 75th and 25th percentile of $\alpha(e, H, t)$. Generally, the Freedman-Diaconis rule is based on minimizing the sum of the squared errors between the bin height and the underlying actual distribution. Based on this rule, the number of bins at each edge is automatically learned and set to $\frac{n}{w}$.

Selecting m: While a longer history takes a global view, it creates an explosion in storage cost. We optimize m by analyzing the storage vs. accuracy trade off in training data.

B. Inferencing from the NMM

The NMM models the likelihood of a segment being traversed based on its m-history and the current time of a day. By performing *semi-supervised random walks with restarts* on the NMM, we compute the conditional of an edge being traversed without performing any trajectory scans. Using the principle of GS, by repeatedly sampling from these conditionals, we compute the conditional joint distribution $p(U|O)$.

1) Semi-supervised Random Walk with Restarts: Random walk with Restarts (RWR) simulate the trajectory of a random walker who starts from a source node and iteratively jumps from one node to a neighbor. The probability of jumping to a neighbor is proportional to the weight of the corresponding edge. At the same time, with a restart probability r, the walker jumps back to the source node. Upon returning to the source node, a new walk is initiated.

Trajectory Generation: We generate trajectories from the NMM through semi-supervised RWR. Each generated trajectory represents a sample from the joint distribution space of $p(U|O)$. By repeatedly generating these trajectories, the uncertain trajectory U is inferred.

Given a set of observations, each pair of consecutive observations, s_i and s_{i+1}, is picked and a RWR is initiated from $s_i.v$ (Alg. 1). As in a normal RWR, for each jump, the new destination is selected based on the affinities of all outgoing edges from $s_i.v$. However, to model the mobility pattern in a trajectory, we only consider edges that do not create a cycle (line 11 in Alg. 1). This follows from the

Algorithm 1 SampleTrajectory(s_1, s_2)

1: $curr \leftarrow s_1.v$
2: $t \leftarrow s_1.t$
3: $r \leftarrow$ restart probability selected based on Eq. 6
4: $\mathcal{S} \leftarrow \emptyset$
5: **while** $curr \neq s_2.v$ **do**
6: $p \leftarrow$ sample uniformly from $[0, 1]$
7: **if** $p \leq r$ **then**
8: $curr \leftarrow s_1.v$
9: $\mathcal{S} \leftarrow \emptyset$
10: **else**
11: $\bar{E} \leftarrow \{e.p_1 = curr,\ e \in E$ does not create a cycle in current walk$\}$
12: $H \leftarrow$ extract m-history of $curr$ from $\mathcal{S}$
13: $e \leftarrow$ select edge from $\bar{E}$ proportional to $\frac{\alpha(e,H,t)}{\sum_{\forall e' \in \bar{E}} \alpha(e',H,t)}$
14: $t \leftarrow t + speed(e,t) * length(e)$
15: $curr \leftarrow e.p_2$
16: **if** $curr = s_2.v$ **and** $p \leq \tau$ **then**
17: $curr \leftarrow s_1.v$
18: $\mathcal{S} \leftarrow \emptyset$
19: **else**
20: $\mathcal{S} \leftarrow \mathcal{S} \cup \{e\}$
21: **return** $\mathcal{S}$

underlying assumption that each transition in a trajectory takes us closer to the destination. To avoid cycles, we maintain a set $\mathcal{S}$ that stores each visited edge while performing the walk. Now, given a current node v and its m-history at time t, the chances of selecting an outgoing edge e is proportional to its affinity (line 13 in Alg. 1). Mathematically, the transition probability through an edge e is expressed as:

$$p(e) = \begin{cases} (1-r)\frac{\alpha(e,H,t)}{\sum_{\forall e' \in \bar{E}} \alpha(e',H,t)} & \text{if } e \in \bar{E} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$\bar{E} = \{e.p_1 = v, e.p_2 \notin \{e'.p_1 \cup e'.p_2 | \forall e' \in \mathcal{S}\} \mid e \in E\}$ is the set of outgoing edges that do not induce a cycle. v is the current node, and t is the current time (line 14 in Alg. 1).

Capturing destination bias: An edge transition to e in the random walk does not compute e's conditional probability. As formulated in Eq. 3, the conditional probability is equivalent to computing the transition probability on the subset of historical trajectories that pass through the destination s_2. Computing Eq. 3 from $\mathbb{H}$ requires us to perform trajectory scans, which, as we have shown in Sec. V-A is extremely expensive. Thus, to approximate Eq. 3 in a scalable manner, we enforce the destination bias on RWR itself using the restart probability r. In Eq. 5, only $(1-r)$ of the probability mass is distributed among edge transitions. Like in a normal RWR, there is a chance of jumping back to the source node with probability r. If a restart occurs, semantically, the walk is considered to have ventured towards a wrong direction and thus discarded by reinitializing $\mathcal{S}$ to an empty set (line 9 in Alg. 1). A new walk is then initiated with the goal of reaching destination $s_{i+1}.v$, and finally, only those walks that successfully reach $s_{i+1}.v$ are recorded. Consequently, the destinations bias is strictly enforced on all edge transitions.

In a traditional RWR, the restart probability is static since the goal is to compute network proximity to the source node. In our problem, the goal is to reach the destination node $s_{i+1}.v$ within the expected duration $X_t = s_{i+1}.t - s_i.t$. Otherwise, it is likely that the walker is in the wrong path. Therefore, sufficient time must be allowed to the walker to successfully reach the target. However, if the walker is unsuccessful in reaching the destination within X_t, then the chances of restarting the walk should be explored. To model these requirements, we define the restart probability r as the following.

$$r = 1 - \frac{1}{e^{\frac{max\{0, t - X_t\}}{X_t}}} \quad (6)$$

where t is the time spent on the "current" walk (line 14 of Alg. 1). Thus, till the time spent on a walk is less than the expected time X_t, the restart probability is 0. As t exceeds X_t, the restart probability begins to increase exponentially.

To summarize, we generate trajectories by repeatedly sampling from the conditionals expressed through edge transition probabilities and the destination bias. Each generated trajectory is a point in the joint distribution space of $p(U|O)$. Thus, U is defined over all edges that are sampled at least once. As outlined in Eq. 1,

$$edgeWeight(e) = \frac{\|\{T \in \mathbb{T} | e \in T\|}{\|\mathbb{T}\|} \quad (7)$$

where T is the trajectories generated through RWR. RWR terminates once the joint distribution of U converges

***Example** 3: Fig. 5 shows a probable result with respect to the road network in Fig. 1 and its corresponding NMM in Fig. 4, under observations $\{a \rightarrow g\}$. From 8 different walks, 4 are successful in reaching destination g. For simplicity we ignore the temporal aspect and assume that the estimated time of all successful paths match the observed time. In other words, $\tau \approx 0$. From these successful walks, the uncertain trajectory is constructed. Note that although 80% of the vehicles go to e from c in the NMM, due to the destination bias enforced by g, the $c \rightarrow g$ edge has a probability of 50% of being traversed.*

C. Properties of Uncertain Trajectories

We next highlight the properties of an uncertain trajectory $U(V', E')$.

Node Visit Likelihood: The probability of visiting a node $v \in V'$ is the sum of the incoming edge-weights $E_{in} = \{e.p_2 = v | e \in E'\}$.

$$nodeWeight(v) = \sum_{\forall e \in E_{in}} edgeWeight(e) \quad (8)$$

The sum of incoming edge weights in a node v is equal to the sum of its outgoing edge weights.

Trajectory Likelihood: Let $\mathbb{P}$ be the set of paths from the source to the destination of U. The probability of a trajectory $T \in \mathbb{P}$ is expressed as the following.

$$p(T) = \prod_{i=2}^{|T|} p(v_i \in T | v_{i-1} \in T) \quad (9)$$

where,

$$p(v_i \in T | v_{i-1} \in T) = \frac{edgeWeight((v_{i-1}, v_i))}{\sum_{\forall e \in E', e.p_1 = v_{i-1}} edgeWeight(e)}$$

Taking products of the individual node likelihoods that constitute a trajectory $T \in \mathbb{P}$ is not enough since they

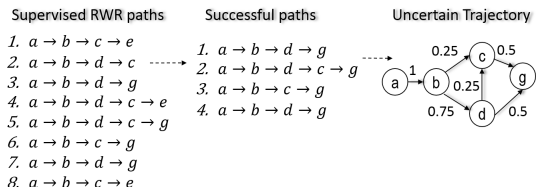


Fig. 5. The uncertain trajectory constructed from the NMM in Fig. 4 on the observations $\{a \rightarrow g\}$.

are not independent. The *maximum likelihood trajectory* is, $T^* = \arg\max_{\forall T \in \mathbb{P}} p(T)$.

D. Discussion

This section answers two important questions. *Why RWR on the NMM is equivalent to Gibbs sampling?* And *Why InferTra is more efficient than the straight-forward implementation of Gibbs sampling outlined in Sec. V?*

The first step in GS is to identify the vector of random variables and initialize them. The random variables can be initialized to arbitrary values since it is independent from the final convergence to the joint distribution. In InferTra, the first edge transition is conditioned only on the observed data, which translates to initializing all edges as "not-traversed".

After initialization, GS moves into the iterative phase. When compared to the outline in Sec. V, in RWR, each new walk corresponds to the outer loop over τ, and the edge transitions correspond to the inner loop. Let us first focus on the inner loop, which samples each edge conditioned on the observations and the current values of remaining edges. In GS the inner loop is iterated over all edges. In RWR, the sampling stops as soon as the walk reaches the destination. This early termination, however, does not cause any loss of information, as stated in Theorem 1.

***Theorem** 1: Let $i = m$, when the random walker reaches the destination. For all subsequent iterations of i*

$$p(e_i|e_1^{\tau+1}, \cdots, e_{i-1}^{\tau+1}, e_{i+1}^{\tau}, \cdots, e_n^{\tau}, O) = 0 \qquad (10)$$

PROOF: Recall, that all trajectories are cycle-free. Thus, for all subsequent iterations or i, the probability of transitioning to any other edge and returning to the destination is 0. □

The second key difference from GS is that each new iteration over τ does not forget the values of the random variables assigned in the previous iteration. More specifically, the conditional at $i = 1$ at any iteration of τ is expressed as following:

$$e_1^{\tau+1} \approx p(e_1|e_2^{\tau}, \cdots, e_n^{\tau}, O) \qquad (11)$$

In a new RWR walk, the previous walk is completely forgotten and the first transition to edge e_1 is conditioned only on the observations. This deviation, however, does not cause any information loss as stated in the following theorem.

***Theorem** 2:*

$$p(e_i|e_1^{\tau+1}, \cdots, e_{i-1}^{\tau+1}, e_{i+1}^{\tau}, \cdots, e_n^{\tau}, O) = p(e_i|e_1^{\tau+1}, \cdots, e_{i-1}^{\tau+1}, O) \qquad (12)$$

PROOF: From Eq. 3, we know that the conditional depends only on the past m traversed edges and the observation set. Now, as discussed in Sec. V-A, since trajectory is a sequence of random variables, for each edge that is marked as traversed, a current timestamp value is also assigned to it. Furthermore, the timestamp is reset to 0 at the start of each iteration of the outer loop. Let us denote the set of traversed edges at the current iteration of τ as $\mathbb{E}$. It is easy to see that $\mathbb{E} \subseteq \{e_1, \cdots, e_{i-1}\}$, where i is the latest edge being sampled in the inner loop. Thus, edges sampled in previous iterations of τ have no impact in the current iteration, which is how the RWR operates. □

Above analysis establishes how RWR on NMM conforms to GS framework. Now, we focus on the efficiency of InferTra. The naive pipeline of Sec. V-A suffers from two scalability issues. First issue is of identifying the random variables, which finds all paths connecting each pair of consecutive observed nodes. RWR completely skips this step. Since in the initialization step, we initialize all edges as not traversed, we do not need to identify these edges explicitly. We identify these edges on-the-fly during the random walk.

The second aspect affecting scalability of GS is computing the conditionals. This step is expensive since computing the conditionals requires scanning the entire trajectory database. InferTra tackles this problem by completely negating the need to perform database scans in the online phase. The NMM pre-computes the m-history for each transition in its offline learning. To condition the transition probabilities with the destination bias as required by Eq. 3, RWR only records those walks that successfully reach the destination. Furthermore, using Theorem 1, the number of conditionals computed is restricted to only those that have non-zero likelihoods.

VII. EXPERIMENTS

In this section, we show that:

- **Inference:** InferTra is more accurate and scalable than the state-of-the-art trajectory inferencing technique.
- **Versatility:** InferTra supports a wider range of queries.

A. Datasets

GPS traces: We use gps-traces of cabs in the city of Beijing [11], [12]. Each cab is tracked for a week-long duration. Prediction on this dataset is particularly difficult, since cabs do not have any common or frequent routes that are typically observed in trajectories of buses or personal vehicles.

Road network: The road network of Beijing is extracted from OpenStreetMap [13]. The Beijing road network contains 623,975 nodes and 672,284 edges.

Network-constrained trajectories: The trajectories are map-matched [14], [15], [16] to the Beijing network generating $136,759$ network-constrained trajectories.

B. Experimental Setup

Our algorithms are implemented in Java JDK 1.6.0 and evaluated on a PC with 12GB memory and Intel i5 2.60GHz quad core processor running Ubuntu 13.04. We benchmark InferTra against HRIS [6], the shortest path (SP), and the

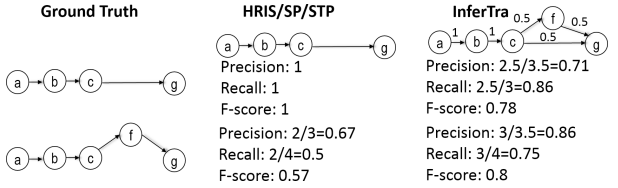
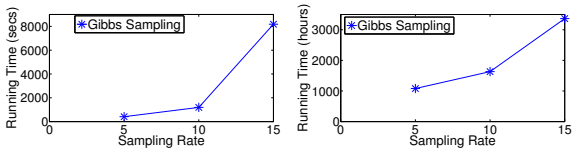


Fig. 6. Precision, Recall and F-score values for HRIS/SP and InferTra against the two shown ground truth trajectories.



(a) Identifying random variables (b) Computing Conditionals

Fig. 7. Growth rate of running time with sampling interval (SI) while (a) Identifying random variables, (b) Computing conditionals.

shortest time path (STP) in the road network. To compute STP, we use average traversal times in the historical data as edge weights.

Parameters: HRIS contains 8 different parameters, which are set as suggested by the authors in [6]. InferTra has only two parameters: the sliding time window size δ, and the model order. δ is set to 30. The order of the Markov model for NMM is learned from the training dataset and is set to 3. The learning procedure is discussed in Sec. VII-D. The default sampling interval (SI) is assumed to be 15 minutes per sampled point.

Benchmarking Setup To evaluate prediction accuracies, we perform 10-fold cross validation. The training dataset is used to learn the NMM and inference is performed on the test set. To model a desired sampling rate, nodes from trajectories in the test set are deleted accordingly. Next, a prediction $U_p = (V_p, E_p)$ is generated on the under-sampled trajectory and compared with the original ground truth trajectory $T = (V, E)$. For InferTra, U_p is an edge-weighted graph, whereas for HRIS/SP/STP, U_p is a path in the road network. The accuracy of the prediction is quantified using *F-score*. F-score can be visualized as a weighted average of the precision and recall, where the best performance corresponds to a value of 1, and the worst corresponds to 0. For HRIS (or SP/STP), computing precision and recall is straightforward. In an uncertain trajectory however, a constituent edge exists with a probability. Thus, the formulations of precision and recall are modified to handle both certain and uncertain trajectories. Let $E_c = E \cap E_p$ be the set of common edges in T and U_p. Now,

$$\text{recall} = \frac{\sum_{\forall e \in E_c} edgeWeight(e)}{|E|} \tag{13}$$

$$\text{precision} = \frac{\sum_{\forall e \in E_c} edgeWeight(e)}{\sum_{\forall e \in E_p} edgeWeight(e)} \tag{14}$$

Eqs. 13 and 14 degenerate to their standard formulations for certain trajectories. For uncertain trajectories, rather than operating in a binary world, their likelihoods are considered.

***Example** 4: Fig. 6 demonstrates the F-score computations.*

C. Performance of naive Gibbs sampling

Before benchmarking the performance of InferTra, we revisit the performance issues of GS highlighted in Sec. V-A.

- **Identifying random variables:** Fig. 7(a) shows the growth rate of time taken to identify the random variables with SI. To keep the path identification practical, we restrict ourselves to only those paths that are at most 2.5 times the length of the shortest path. Even then, at $SI = 15$, it takes 8183 seconds to identify all paths.
- **Computing conditionals:** Computing the conditionals presents an even larger scalability challenge. At $m = 3$, computing Eq. 3 takes 1.2 seconds on average. Consequently, even at an SI of 5, it is practically infeasible to achieve convergence. Assuming it takes at least 10000 iterations of τ given the high dimension of the space, we compute the average number of random variables, d, at SI= 5, 10, 15, and compute the projected convergence time using the formula $1.2 * 10000 * d$. Figure 7(b) demonstrates this projected time. Clearly, 1000 hours even at SI= 5 is prohibitively large.

D. Performance of InferTra

Among the various queries outlined in Sec. I, we first benchmark the performance on trajectory inference.

1) What is the actual trajectory?: First, we study the impact of sampling interval (SI) on trajectory inference. The first plot in Fig. 8(a) demonstrates the results as the SI is varied from 5 minutes per point to 25 minutes. As expected, the F-score decreases with increase in the SI. Across all SIs, InferTra outperforms HRIS. STP does not display a good performance since drivers do not have a global knowledge of the shortest routes. Furthermore, a separate work has shown that people prefer more pleasant routes than the quickest [17]. In addition to the uncertain trajectory predicted by InferTra, we also evaluate its maximum likelihood trajectory (MLT) computed as outlined in Sec. VI-C. The accuracy of the MLT is comparable to the uncertain trajectory till an SI of 10. Beyond 10, there is a sharp deterioration and it resembles the accuracy of HRIS. This result highlights why it is important to go beyond maximum likelihood estimations and capture the entire uncertainty surrounding partial observations. Our more holistic approach to inference not only ensures a higher accuracy, but also enables slower deterioration rate with SI. Consequently, even at an SI of 25 minutes, InferTra achieves an F-score of 0.51, which is 50% higher than the F-score of 0.32 by HRIS.

Next, we benchmark the inferencing time of InferTra. Plot two in Fig. 8(a) demonstrates the results. At a higher SI, there is more uncertainty between two intermediate points, and consequently, a higher inference time is required. While SP is marginally faster than InferTra, InferTra is up to 20 times faster than HRIS. This amazing speed-up is achieved since InferTra is only reliant on local state transitions. In contrast, HRIS performs a search across the entire dataset to identify trajectories that overlap with the given observations. Based

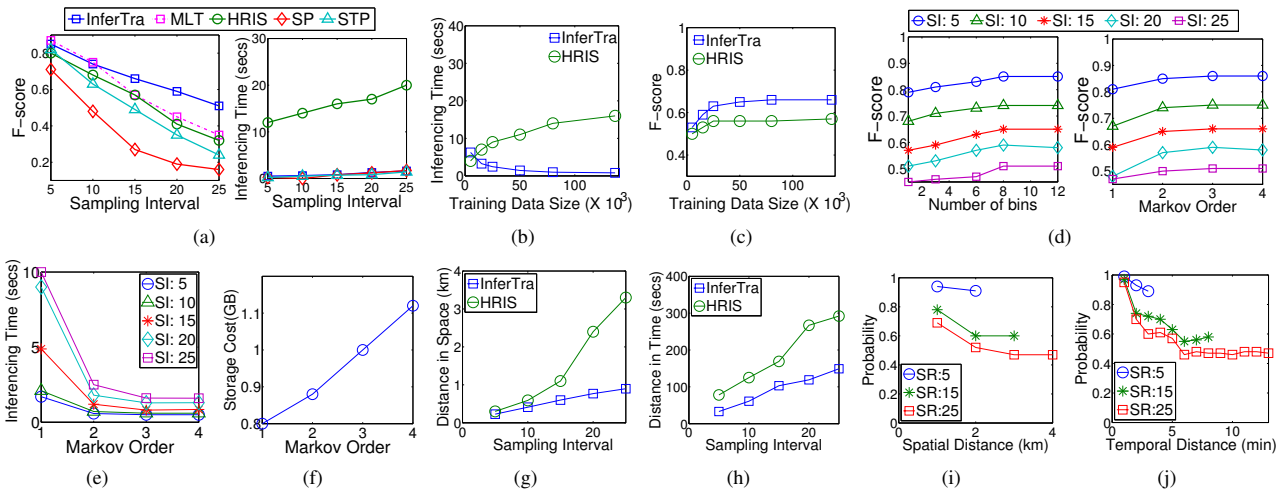


Fig. 8. Growth rate of accuracy and inferencing time with (a) sampling interval (SI) and (b-c) training dataset size. (d) Growth rate of accuracy with number of bins in the affinity vector, and order of the Markov model. Growth rate of (e) inferencing time, and (f) storage with the order of the Markov model. (g) Spatial and (h) temporal distances between actual times and actual locations, respectively, against sampling interval. Node probability against distance in (i) space and (j) time from the nearest observation.

on the results, a relevant subgraph is extracted on which the inference is made. Due to the online searching of the trajectory database, a high inferencing time is incurred in HRIS.

Continuing on the topic of inference time, we next study how it varies with the training dataset size. As shown in Fig. 8(b), InferTra has the attractive property of a decreasing inference time with increase in training data. As more training data is available, the conditional probabilities get more accurate and consequently, converges quicker to the joint distribution. On the other hand, the inferencing time of HRIS grows with the training data size since the online search on the trajectory database imparts a larger cost. In addition to the inferencing time, we also study the impact of training data on the accuracy. As expected, the performances of both techniques improve with additional training data.

InferTra not only learns spatial signals, but is also sensitive to any temporal periodicity in the mobility patterns. We thus analyze if identifying such temporal periodicities improves the inference performance. Towards that goal, instead of automatically learning the optimal number of bins in the affinity vector at each edge, we manually set the number of bins across all edges. Lower the number of bins, the more is the reliance on using only the spatial signals. In the extreme case where only 1 bin is used, no temporal information is incorporated in the NMM. Fig. 8(d) demonstrates the results. With increase in the number of bins, there is up to 10% increase in the F-score. Across all SIs, the improvement saturates at 8 bins. This result clearly highlights the importance of capturing temporal patterns in addition to the spatial signals. Prior to InferTra, this dimension of the inference problem has not been explored.

Do vehicles take locally optimal decisions in route selections? The order of the Markov model in the NMM controls how well this preference is learned. We next study this issue. First, we evaluate the inference accuracy as a longer history is retained to determine the transition probabilities in the NMM. As it can be seen in Fig. 8(d), the F-Score saturates at an order of 3 across all SIs. This result, combined with the weak performance of STP, show that vehicles typically make locally optimum decisions and thus, looking too far back in the history is not useful. A trend similar to the accuracy is also visible in the inferencing time analysis in Fig. 8(e). At an order of 1, where only the current state is examined probabilities, the mobility patterns are not captured as accurately. Due to the resultant ambiguity, the convergence to the joint distribution is slower. Although, a higher Markov order incurs more storage cost, as shown in Fig. 8(f), the growth rate is linear. This result is expected given the fact that the average outgoing degree in the Beijing road network is close to 1. Overall, to summarize the attractive features of InferTra:

- By not relying on shortest paths, which is at the core of HRIS, InferTra is up to 50% more accurate and 20 times faster.
- Both the efficiency and the accuracy of InferTra improve with increase in the size of training data.
- InferTra is the first technique to capture the temporal signals embedded in historical trajectories.
- InferTra has only 2 parameters, of which only 1 has any noticeable impact on the performance.

2) Where should I search at time t?: Given a set of observations, what was the most likely location of a vehicle at time t? Or, if a vehicle was at node n, when did the vehicle reach n? Such queries routinely find applications in retrospective analysis of crime investigations, and existing techniques cannot answer them adequately. We analyze performance of InferTra in these scenarios. Recall, Alg. 1 also estimates the time at any node n in the uncertain trajectory (line 14).

First, we study the accuracy of the predicted location at an input time t. To setup the experiment, the input time t is set

to the timestamp of a randomly picked node that is part of the ground truth trajectory, but not included in the observation set. We quantify the prediction accuracy, by computing the spatial distance between the actual and the predicted locations at time t. In InferTra, there can be multiple routes to the destination and therefore, a distribution of nodes is produced as possible locations at time t. We compute the overall spatial distance sd, by taking their weighted sum. More formally,

$$sd = \sum_{\forall d_n \in \mathbb{SD}} p(d_n) \times d_n \tag{15}$$

where $\mathbb{SD}$ is the set of distances from the actual location corresponding to each predicted node n at time t, and $p(d_n)$ is the associated probability.

In its original form, HRIS cannot answer this query. First, HRIS cannot predict time. In addition, to predict location at an input time, one needs to analyze node-level likelihoods instead of whole trajectory likelihoods. Nonetheless, for benchmarking purposes, we extract an answer from HRIS based on the assumption that the most likely node at time t is part of the most likely trajectory. We estimate the time at each node based on the average speeds in its constituent edges. Fig. 8(g) shows the results as the SI is varied. While InferTra estimates the location within a radius of 1 KM even at an SI of 25 minutes, the error range in HRIS is as high as 3.5 KM. This result stems from the fact that the most likely node may not necessarily be part of the most likely trajectory. While an uncertain trajectory captures both node and trajectory level likelihoods, HRIS relies on a close correspondence between the most likely trajectory and most likely node. Consequently, the error range is higher.

Fig. 8(h) performs the dual of the previous study. Instead of predicting the location at a given time, we predict the time at a given node n from the ground truth. If the input node n is not part of the uncertain trajectory or the HRIS-inferred trajectory, we output the time at the node that is spatially closest to n. Similar to the previous query, HRIS is not built for answering this query. This inability to make node-level predictions coupled with the tight integration of both spatial and temporal patterns in the inference procedure of InferTra, result in a significant performance disparity. As visible in Fig. 8(h), InferTra is 2 times more accurate than HRIS.

Finally, we look at InferTra's performance in predicting node likelihoods based on its distance from the closest observation. To assess the performance, we randomly pick nodes from ground truth trajectories and compute its visit likelihood in the InferTra prediction. Generally, closer a node is to an observation, higher is its probability; as the distance from the observation grows, the more difficult the prediction task becomes. Figs. 8(i) and 8(j) demonstrate the results against spatial and temporal distances respectively. The trends are similar and in line with the general intuition. It is interesting to note that the SI is an important factor. To give an example, when the destination is 10 minutes away, the number of possible routes between the source and the destination is much larger. Consequently a higher number of candidate nodes are present that are 1 KM away. On the other hand, if the destination is only 2 minutes away, only a few routes exist that can connect the source to destination within the observed time. As a result, the candidate space is smaller, and hence, less is the uncertainty.

VIII. Conclusion

In this paper, we studied the problem of trajectory inference from partial observations. We developed a technique called *InferTra* that summarizes all of the possibilities through an "uncertain" trajectory. By taking the shape of an edge-weighted graph, an uncertain trajectory captures a richer representation of the uncertainty surrounding partial observations than maximum likelihood estimations. InferTra is built on the foundation of Gibbs sampling and is powered by a *Network Mobility Model (NMM)*, which not only utilizes the spatial patterns embedded in historical data, but also unearths how these patterns vary with time. Extensive experiments on real network-constrained trajectories showed InferTra to be up to 50% more accurate and 20 times faster than the state-of-the-art inferencing technique. In addition, an uncertain trajectory can handle a wider range of important queries.

Acknowledgements: We thank Google for travel grant PO 55023527.

References

[1] L.-Y. Wei, Y. Zheng, and W.-C. Peng, "Constructing popular routes from uncertain trajectories," in *KDD*, 2012, pp. 195–203.
[2] V. Kolar, S. Ranu, A. P. Subramainan, Y. Shrinivasan, A. Telang, R. Kokku, and S. Raghavan, "People in motion: Spatio-temporal analytics on call detail records," in *COMSNETS*, 2014, pp. 1–4.
[3] J. Shi, N. Mamoulis, D. Wu, and D. W. Cheung, "Density-based place clustering in geo-social networks," in *SIGMOD*, 2014, pp. 99–110.
[4] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *WWW*, 2009.
[5] J. Stanley and B. Steinhardt, *Bigger monster, weaker chains: the growth of an American surveillance society.* American civil Liberties Union Technology and Liberty Program, 2003.
[6] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories." in *ICDE*, 2012, pp. 1144–1155.
[7] J. Niedermayer, A. Züfle, T. Emrich, M. Renz, N. Mamoulis, L. Chen, and H.-P. Kriegel, "Probabilistic nearest neighbor queries on uncertain moving object trajectories," *PVLDB*, vol. 7, no. 3, pp. 205–216, 2013.
[8] T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle, "Querying uncertain spatio-temporal data," in *ICDE*, 2012.
[9] G. Casella and E. I. George, "Explaining the gibbs sampler," *The American Statistician*, vol. 46, no. 3, pp. 167–174, 1992.
[10] D. Freedman and P. Diaconis, "On the histogram as a density estimator:L2 theory," *Probability Theory and Related Fields*, vol. 57, no. 4, pp. 453–476, 1981.
[11] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-drive: driving directions based on taxi trajectories," in *SIGSPATIAL GIS*, 2010.
[12] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *KDD*, 2011, pp. 316–324.
[13] "http://openstreetmap.org/."
[14] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in *GIS*, 2009.
[15] J. S. Greenfeld, "Matching GPS observations to locations on a digital map," in *81st Annual Meeting of the Transportation Research Board*, 2002.
[16] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *MDM*, 2010, pp. 43–52.
[17] D. Quercia, L. M. Aiello, and R. Schifanella, "The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city," in *Proceedings of Hypertext*, 2014.