Sampling Big Trajectory Data for Traversal Trajectory Aggregate Query

Yichen Ding, Yanhua Li, *Senior Member, IEEE*, Xun Zhou, Zhuojie Huang, Simin You, Jun Luo

Abstract—This paper defines and investigates a novel trajectory query, namely, Traversal Trajectory Aggregate (TTA) Query: Given a trajectory database and a pair of upstream and downstream spatio-temporal (ST) regions (i.e., spatial area coupled with a time interval), a TTA query aims to retrieve the total number of unique trajectories that traverse through these two ST regions. Such TTA queries play an important role in various urban applications, such as route planning, taxi dispatching, and location-based advertising. Two baselines can answer such TTA queries: *(a)* exact search (over the entire ST query regions) can obtain the exact answer, but it leads to extremely long running time when the ST query regions are huge; *(b)* uniform-sampling-based approaches estimate the query answer with sampled trajectories. However, the uniform sampling distribution may lead to significant estimation variance for TTA query, because traversal trajectories are relatively few and unevenly distributed in the query regions. To tackle these challenges, this paper proposes a novel Targeted Index Sampling (TIS) framework to answer TTA queries with high estimation accuracy. TIS employs a two-stage framework, with a Pilot Sampling Estimation (PSE) stage to estimate the distribution of trajectories in ST query region, and an Integrated Importance Sampling (IIS) stage, which collects trajectories with the importance sampling distribution obtained in PSE, and estimates the query result with an asymptotically unbiased estimator. Extensive *experiments and case studies* using a large-scale real taxi trajectory dataset from Shenzhen, China demonstrate that our TIS framework achieves $\leq$10% estimation error with $\geq$ 90% computational time reduction over exact search, and 50% reduction on estimation error (with similar running time) over uniform-distribution-based sampling approaches.

Index Terms—Traversal Trajectory, Aggregate Query, Importance Sampling.

1 Introduction

With the advanced development of location-acquisition technologies, an increasing amount of trajectories are generated over time from various moving objects (such as vehicles and mobile phone users). Integrating such large-scale trajectory data with road network structures, human mobility, points of interest makes it possible to facilitate and enable various emerging applications including transportation system [1], [2], [3], urban computing [4], [5], [6], [7] and other location-based pattern discoveries [8], [9], [10], [11], [12], [13].

Statistical aggregate queries over large-scale trajectory data, such as total number, average length, speeds of trajectories are essential building blocks toward complex urban computing tasks. In this paper, we define and investigate a novel trajectory aggregate query, namely, the *traversal trajectory aggregate (***TTA***) query*. Given a trajectory database and a pair of upstream and downstream spatio-temporal (ST) regions (i.e., spatial area coupled with a time window), the traversal trajectory aggregate query returns the total number of unique trajectories that go through these two ST regions sequentially. For example, "finding the number of unique trajectories that traversed the airport and the hotel zone sequentially between 8pm and 10pm" is a TTA query. Answering such traversal trajectory aggregate queries have great potentials to facilitate many urban applications including but not limited to: (1) Location-based advertising and recommendations, such as recommending trip plans with multiple stops for tourists based on trip popularity in candidate locations, and (2) Event and congestion detection and prediction based on traversal trajectories across different locations [14].

It is computationally challenging to answer traversal trajectory aggregate queries. First, due to the large number of trajectories and the varying sizes of the ST query regions, checking all the trajectories in the query regions is very time-consuming. Alternatively, sampling [15], [16] becomes a promising approach to TTA query. Instead of checking every trajectory in the query regions, a subset of trajectories is obtained through sampling and the query result is estimated based on sampled trajectories. However, traditional sampling methods for trajectory aggregate queries are designed for single-region aggregate queries and typically employ a uniform distribution [17] to conduct sampling in query regions. For TTA queries, with relatively few and unevenly distributed traversal trajectories in the query regions, such uniform-sampling-based approaches, e.g., Random Index Sampling algorithm (RIS) [17], would yield a low estimation accuracy.

In this paper, we propose a novel Targeted Index Sampling (TIS) framework to answer TTA queries with high estimation accuracy. TIS employs a two-stage framework, with a **P**ilot **S**ampling **E**stimation (PSE) stage to estimate the distribution of trajectories in ST query regions, and an

- *Y. Ding and X. Zhou are with the Department of Management Sciences, University of Iowa, Iowa City, IA, 52242. E-mail: {yichen-ding, xun-zhou}@uiowa.edu*
- *Y. Li is with Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA 01609. E-mail: yli15@uiowa.edu*
- *Z. Huang and S. You are with Pitney Bowes Inc. E-mail: {Zhuojie.Huang, Simin.You}@pb.com*
- *J. Luo is with Machine Intelligence Lab, Lenovo Group Limited, Hong Kong. E-mail: jluo1@lenovo.com*

Manuscript received January xx, 2018; revised xxxx xx, 2018.

Integrated **I**mportance **S**ampling (IIS) stage, which collects trajectory samples with the importance sampling distribution [18] obtained from PSE, and estimate the query result with an asymptotically unbiased estimator. Our contributions are summarized as follows.

- We introduce a novel trajectory query type, namely, the traversal trajectory aggregate (TTA) query. TTA query is challenging to answer, since the traditional (uniform) sampling approaches fail to provide an accurate estimation result, due to the small number of qualified trajectories and their uneven distributions in ST query regions.
- We develop a novel Targeted Index Sampling (TIS) framework, with Pilot Sampling Estimation (PSE) stage to obtain an importance sampling distribution of traversal trajectories in a query region; and an integrated importance sampling (IIS) stage that sample and estimate the query results with a provably asymptotically unbiased estimator, that guarantees a high accuracy estimation to answer the TTA queries.
- We conduct extensive experiments and case studies using a large scale taxi trajectory dataset from Shenzhen, China, to evaluate the efficiency and effectiveness of the proposed approach. Results demonstrate that the proposed TIS framework achieves $\leq 10\%$ estimation error with $\geq 90\%$ computational time reduction over exact search, and 50% reduction on estimation error (with similar running time) over uniform-distribution-based sampling approaches.

The rest of this paper is organized as follows. Section 2 discusses related work. We formally define our TTA query in Section 3. In Section 4–5, we present our proposed solution framework. Section 6 presents comprehensive evaluation results. Section 7 discusses various generalizations and extensions of our TIS framework. Finally, we conclude the paper in Section 8.

2 Related Work

Prior work related to this paper can be generally classified into three categories: (1) spatio-temporal data management, (2) trajectory query processing, and (3) sampling and estimation methods.

2.1 Spatio-temporal Data Management

With the rapid increase in the popularization of location-aware sensors in a variety of new applications like GPS, 4G network, a large amount of trajectory data are generated over time, which requires appropriate indexing structures to enable efficient query processing over such big trajectory datasets. In the literature, various spatio-temporal indexing structures have been proposed. R-tree is widely used to index two-dimension data [19], [20]. G-tree takes road network structure into consideration for effective KNN search over road network [21]. A 3D-Rtree includes the time as the third dimension, e.g., the Spatio-Temporal R-tree (STR-tree) and the Trajectory-Bundle tree (TB-tree) [22]. By dividing the time dimension into multiple time intervals and linked to corresponding spatial index, multiple version R-trees are developed, such as Historical R-tree (HR-tree), HR+-tree [23], and mv3r-tree [24]. Moreover, a grid-based index partitions a geographical space into grids, e.g., the Compressed Start-End Tree (CSE-tree) [25] and the Scalable and Efficient Trajectory Index (SETI) [26]. These grids are built by Quad-tree [27] or multidimensional binary search tree (k-d tree) [28], while temporal range indexed by the hybrid B+-tree [29] or B-tree [30] even time list. The inverted index [31] is also an efficient index for trajectory query.

With a well-built spatio-temporal index, we are able to access trajectory data efficiently. However, when the query range is large, e.g., containing a large amount of spatio-temporal data, it is still very time-consuming to examine the entire query range to get an exact answer of the query. As a result, approximate query processing becomes a promising solution under stringent query requirements. In this work, we propose a sampling based approach to provide estimated answer of traversal trajectory query (TTA), with guaranteed accuracy.

2.2 Trajectory Query Processing

Various spatio-temporal data queries have been extensively studied in the literature, which are primarily in two main categories, including Range query and K-Nearest Neighbor (KNN) query [32], [33]. A range query retrieves the trajectories falling into or intersecting with a given spatio-temporal region [34], [35], [36]. KNN query can be classified into KNN point query [37], [38] and KNN trajectory query [39] for aggregating minimum distance of a few points or a specific trajectory. Differing from these works, we define and investigate a novel traversal trajectory aggregate (TTA) query, that extracts the counts of unique trajectories traversing two spatio-temporal regions in order.

2.3 Importance Sampling and Estimation

In the big data era, the ability to approximately answer aggregate queries accurately and efficiently is of great importance for decision support and data mining tools [40], [41]. Random sampling via uniform distribution [15], [16] is a simple approach for statistic estimation. For example, [42] proposes a random prefix sampling algorithm to sample and estimate statistics of YouTube videos. [43], [44], [45] introduce various random region sampling approaches to sample and estimate statistics of venues (i.e., locations) in large-scale geographic regions. Moreover, the RIS algorithm [17] employed a uniform distribution to sample and estimate the total number of unique trajectories that go through a given spatio-temporal region. However, in Section 3, we demonstrate that when the underlying distribution of traversal trajectories in a query region greatly differs from uniform distribution. Therefore the estimation accuracy of methods using uniform sampling will be low. Our proposed framework introduces a pilot sampling stage to estimate a trajectory data distribution, with which we employ importance sampling to estimate TTA query answer, namely, the total number of distinct traversal trajectories, with high estimation accuracy.

3 Overview

In this section, we first define key terms used in the paper and provide a formal definition of *traversal trajectory aggregate queries*. Then we describe our key insights which leads

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

to the proposed algorithms. Finally, we give an overview of our system framework.

3.1 Basic Concepts

- **Spatio-Temporal (ST) Cube and Region.** A spatio-temporal (ST) partitioning divides the three-dimensional space into uniform ST cubes. For example, each ST cube in a ST partitioning could be 500m by 500m by 5 minutes. A spatio-temporal (ST) region $R = (A, Tp)$ is a subset of cubes in the underlying ST partitioning, where A is a rectangular spatial area and Tp is a time interval.
- **Trajectory.** A trajectory r is a sequence of spatio-temporal points. Each point consists of a trajectory ID, latitude, longitude, a time stamp. Each trajectory is formed by points with the same trajectory ID in the order of time stamp.
- **Traversal Trajectory.** Given a pair of upstream and downstream ST regions, a traversal trajectory is one that sequentially traverses the two ST regions. For example, a traversal trajectory r traversing two ST regions R_{up} = $\{A_{up}, Tp_{up}\}$ and R_{down}=$\{A_{down}, Tp_{down}\}$ indicates itself traversing upstream area A_{up} at time period Tp_{up} and then passing destination area A_{down} at time period Tp_{down}.
- **Valid Cube and Invalid Cube.** Given a pair of upstream and downstream ST regions, a Valid Cube is a cube in one of the ST regions, which is traversed by at least one traversal trajectory. Cubes in the given ST regions without any traversal trajectories are Invalid Cubes.

3.2 Problem Definition

Traversal Trajectory Aggregate (TTA) Query. Given a trajectory database and a pair of upstream and downstream ST regions $R_{up} = \{A_{up}, Tp_{up}\}$, $R_{down} = \{A_{down}, Tp_{down}\}$, a Traversal Trajectory Aggregate (TTA) query q returns the total number of unique trajectories that first traverse area A_{up} during time Tp_{up} and then area A_{down} during time Tp_{down}. We call a traversal trajectory satisfying the above query conditions a "qualified trajectory".

Figure 1 shows an example of TTA Query, where blue lines represent trajectories traversing only the upstream or the downstream query region while the red lines represent qualified trajectories traversing both the upstream and downstream query regions sequentially. Left and right cuboid represent upstream and downstream ST query regions, respectively.

The objective of our work in this paper is to maximize the query result accuracy, while keeping the query cost within a user-specified range. For each TTA Query, a user may give a *Query Budget* B, which is the total number of different cubes accessed while processing a TTA query. [1] In database implementation, each ST cube is stored in a disk page, with roughly same running time to access and processing the trajectory data in it. So the query budget B is equivalent to query processing time. We also define *Budget Ratio* B_r, as the ratio between B and the number of cubes in one of the ST query regions $|\mathcal{R}^{\{A,Tp\}}|$ selected (e.g., upstream), $0 < B_r \leq 1$.

1. Let $\mathcal{R}^{\{A,Tp\}}$ be the set of ST cubes of a ST region $\{A, Tp\}$. When $B \geq \min(|\mathcal{R}^{\{A_{up},Tp_{up}\}}|, |\mathcal{R}^{\{A_{down},Tp_{down}\}}|)$, all the cubes in the one of the ST regions can be exhaustively searched and an exact result is guaranteed. Otherwise, an estimation of the result is returned.

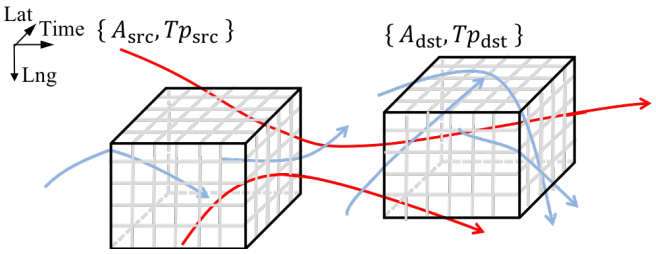


Fig. 1. **An illustration of a traversal trajectory traversing from upstream query region to downstream query region.**

3.3 Challenges and Motivation of Solution

Answering TTA queries through exact search is computationally expensive and not scalable. First, the number of queries could be huge in practical systems. Second, due to query diversity, the potentially huge number of different ST query regions makes it infeasible to pre-compute and save the query results in a static database. Moreover, the database may not be static in reality and new trajectories might be added over time so that query results need to be updated accordingly. Hence, it is a better choice to answer the query using sampling based method to improve flexibility and efficiency, as well as saving costs.

One direct way along this path is to estimate the result via uniform sampling. For example, the random index sampling (RIS) algorithm [17] is a uniform sampling method with guaranteed efficacy and efficiency. However, RIS is designed to query the number of trajectories passing only one ST query region rather than a sequence of upstream and downstream query regions. We illustrate the limitation of uniform sampling using the example below.

We randomly generate 600 TTA queries over a trajectory dataset. and test the cube **rareness** of the upstream and downstream regions, which represents the number of valid cubes as a percentage of the total number of cubes in the upstream (downstream) query region. Figure 2 shows that the cube rareness is a non-uniform distribution, with 70% queries covering no more than 60% cubes and 30% queries covering less than 40% cubes for both of the query regions. This shows that *the distribution of qualified traversal trajectories in the query regions are very different from uniform*. Details query examples could be referred to Section 6.

Clearly, sampling and estimation of the query answer with a uniform distribution may lead to high estimation variance, thus an inaccurate result. Hence, we propose a targeted index sampling (TIS) framework to tackle this issue. TIS includes a pilot sampling estimation (PSE) algorithm to first estimate an importance sampling distribution of the qualified trajectories among cubes in the query regions, and an **I**ntegrated **I**mportance **S**ampling **(IIS)** algorithm to estimate the query result with high estimation accuracy guarantee. Our proposed system overcomes issues in prior work such as high search costs and unable to handle rareness of qualified trajectories by identifying valid cubes efficiently and effectively. Besides, the proposed TIS framework could also be applied to answer other traversal

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

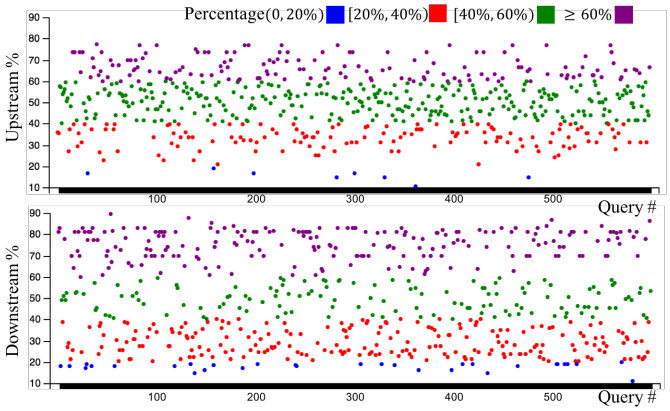


Fig. 2. **Cube rareness of randomly generated TTA queries.**

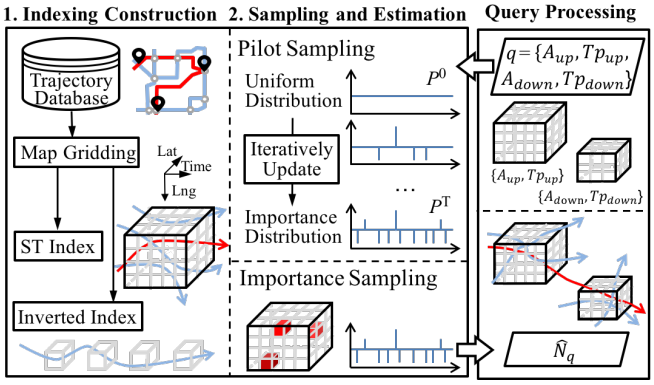


Fig. 3. **An overview of our proposed framework.**

trajectory aggregate queries like average trajectory length of trajectories traversing particular query regions with speed constraints.

3.4 System Overview

Figure 3 gives an overview of our proposed Targeted Index Sampling (TIS) system, which consists of two main components: *Indexing Structures* and *Sampling and Estimation*.

- **Indexing Structures.** This component is the data access building block of the system, which contains a spatio-temporal index and an inverted index. It provides efficient access to the trajectories and cubes. First we partition entire spatio-temporal region of study into equally-sized cubes based on a given spatio-temporal granularity (e.g., 500 meters-500 meters-5min). Then, we build two indices. (1) Spatio-temporal Index: Each cube is used to index all the trajectories traversing it. For simplicity we use a 3-D grid to index the trajectories. However, other indexing structures such as Quad-tree for spatial indexing and B tree for temporal indexing could also be used. Besides, other spatio-temporal indices are also appropriate like R-tree or B+-tree. (2) Inverted Index: We also build an inverted index [17], [31] to efficiently access all the ST cubes traversed by a given trajectory.
- **Sampling and Estimation.** This component contains two stages to answer a traversal trajectory aggregate query: 1) the Pilot Sampling Algorithm and 2) the importance sampling and estimation stage. Stage 1 estimates the distribution of qualified trajectories among cubes in one of the query regions. Stage 2 samples cubes in the same region according to importance sampling distribution acquired from stage 1 and devise an unbiased estimator $\hat{N}_q$. The details are discussed in Section 5. Note for each TAA query, ***we only sample one of the ST query regions***, e.g., the upstream region. For every sampled cube in this region, we check the number of unique trajectories that also traverse the other region by utilizing the Inverted Index structure. In this paper, we always sample the upstream ST region. However, one can sample the downstream region instead and get the same result. A smart strategy to decide which ST region to sample for a TTA query might be of interest but due to page limits we leave it for future work.

4 Indexing Construction

In this section, we present the details in the *indexing construction* component, which builds two types of indexing structures on the trajectory data.

Spatio-Temporal Index. To build the ST index, we partition spatio-temporal regions into equally-sized ST cubes based on a given spatio-temporal granularity (e.g., 500 meters-500 meters-5min). For each cube, we link it with all the trajectories that go through it. We also maintain a list of cubes that are traversed by at least one trajectory to avoid examining empty cubes. This 3-D grid index is employed due to its simplicity. However, other state-of-art indexing structures such as Quad-tree [27] and R-tree [19] for spatial indexing, and B-tree/B+-tree [29], [30] for temporal indexing could also be used.

Inverted Index. In this step, we build an inverted index [17], [31] to efficiently access all the ST cubes traversed by a given trajectory. As shown in Figure 4, each record of inverted index represents a trajectory and all the valid cubes the trajectory traverses. By checking these traversed cubes with $\{\mathcal{R}^{\{A_{down}, Tp_{down}\}}\}$, cubes in destination query region, we are able to effectively identify whether this trajectory is a qualified traversal one or not and accordingly identify its source cube(s) as invalid or valid cube(s).

Using these indices built, one can obtain the exact result of a TTA query by searching each cube (with at least one trajectory) in the upstream (downstream) region to get a list of trajectories and verify if they goes through the downstream (upstream) region using the inverted index. We call this method Exact Search (**ES**). ES is efficient given the index structures. However, when the total number of trajectories increases and the query regions are large, the query time could be extremely long. In the next section we introduce our TIS framework to provide fast response to such queries.

5 Sampling and Estimation

This section presents the Sampling and Estimation component of our system. We first propose a Pilot Sampling Estimation (PSE) algorithm to obtain importance sampling distribution of cubes in one (e.g., upstream) ST query region. Then, we design two asymptotically unbiased traversal trajectory estimators to answer the traversal trajectory aggregate query. To keep the cost below the given budget,

TABLE 1
Notations and terminologies

Notation	Description
$\mathcal{R}^{\{A_{up},Tp_{up}\}} = \{R_1^q, \cdots, R_n^q\}$	The set of n cubes in the upstream ST query region.
$\mathcal{R}^{\{A_{down},Tp_{down}\}}$	The set of cubes in the downstream ST query region.
B, B_r	B is the sampling budget for answering query q, B_r is the budget ratio.
$\mathcal{R}_r$	The set of all cubes traversed by trajectory r.
T	The total number of iterations in the Pilot Sampling Estimation algorithm.
$\hat{R}_{b,t}^q$	The b-th sampled cubes from $\mathcal{R}^{\{A_{up},Tp_{up}\}}$ at iteration t.
k_r^q	The number of valid cubes in $\mathcal{R}^{\{A_{up},Tp_{up}\}}$ that traversal trajectory r traverses.
$N_q, \hat{N}_q$	N_q is the number of distinct trajectories satisfying the query condition of q. $\hat{N}_q$ is the estimator of N_q.

we define the budget for the two stages as B_1 and B_2 where $B_1 + B_2 = B$. Let n be the number of cubes in the upstream ST region, $B_{r1} = \frac{B_1}{n}$, $B_{r2} = \frac{B_2}{n}$. Note that budget ratios at state 1 and stage 2 are separately defined while conducting experiments. They are only subject to the total budget. Specifically, we increase B_{r1} to improve estimation accuracy of importance distribution in order to improve final estimation as well. Meanwhile, increasing B_{r2} could directly improve final estimation. Note the total number of distinct cubes used in stage 2 might exceed $B_{r2} \times n$ since cubes sampled in stage 1 could be saved and reused in stage 2. Table 1 provides a summary of the notations frequently used in this paper.

5.1 Stage 1: Pilot Sampling

The PSE algorithm aims to provide an importance distribution superior to uniform distribution by sampling more valid cubes under the same budget constraint. Figure 4 offers the framework of PSE algorithm for better understanding the iterative update process. First we set an initial distribution P_0, which is a uniform distribution. Then we update the distribution through sampling cubes with replacement in each iteration. If enough valid cubes have been sampled, we update the probability of each cube. Otherwise, no update is performed. The iteration stops when the query budget is reached.

The pseudo code of the PSE algorithm is presented in Algorithm 1. First we initialize current budget $cb = 0$, iteration time $t = 0$ and all of distinct sampled cube set $Sample$, valid cube set $valid_c$ and invalid cube set $invalid_c$ as empty set (line 3). Then we set the stopping criteria as the ratio of cubes accessed reaching the sampling budget ratio B_{r1} (line 4). At each iteration t, with sample size b^t, we obtain independent and identically distributed samples (i.e., cubes) $\hat{R}_{i,t}^q, i \in [1, \cdots, b^t]$ according to the current probability distribution P^t (line 5-6) and compute the counts $S(\hat{R}_{i,t}^q), i \in [1, \cdots, b^t]$, which are the number of distinct qualified trajectories of $\hat{R}_{i,t}^q$ (line 7-10). Specifically, for each sampled cube, we examine all the trajectories passing this cube, and use the Inverted Index to check how many of these trajectories also passed the downstream region.

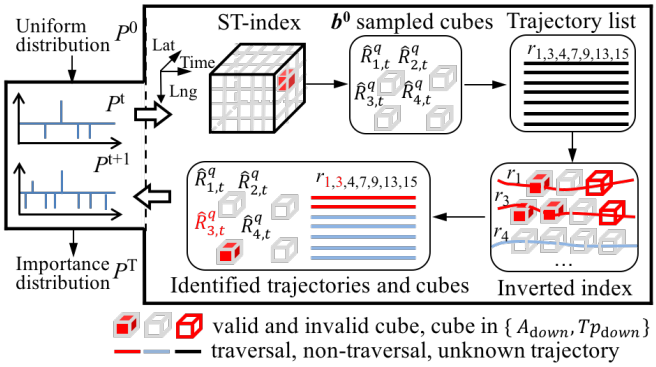


Fig. 4. **Pilot Sampling Estimation algorithm framework.**

Before updating the probability distribution, we need to make sure that sufficient valid cubes have been sampled. This is necessary since in an update we will lower the probability of invalid cubes and increase the probability of valid cubes to make the total probability 1. If only a very small portion of the samples are valid cubes, then each of them will receive a much higher probability in the next iteration, which may lead to significant overestimation. Therefore, we calculate the (1-ρ) quantile $S_{(1-\rho)}$ of all the count values from sampled cubes: $S(\hat{R}_{i,t}^q), i \in [1, \cdots, b^t]$. A zero value of $S_{(1-\rho)}$ indicates that insufficient valid cubes have been sampled so we skip the probability update in this iteration (line 11-13) and increase the sample size by a factor of $\alpha > 1$, but not exceeding an upper bound of sample size b_{upper}(line 12). Alternatively, if $S_{(1-\rho)} > 0$ then the probability distribution can be updated (lines 14-22). The value choice of parameters ρ, α and b_{upper} will be discussed in Section 6.

Then we add the newly-sampled valid and invalid cubes into the corresponding sample lists (line 17-20). Finally, we update the probability distribution P^T according to Equation 1 and 2 (line 21-24). In Equations 1, $Aprob$ is the total probability allocated for all the valid cubes, where n is the total number of cubes in the upstream ST query regions, $|valid_c|$ and $|invalid_c|$ are the number of valid and invalid cubes, respectively, and λ is a smoothing parameter. In order to fully utilize all sampled valid cubes, we increase the probability of all the valid cubes sampled, and reduce the probability of invalid cubes sampled to ensure the total probability equals 1. Instead of reducing the probability of invalid cubes to 0, we introduce a smoothing parameter $\lambda \in [0, 1)$ to smooth the probability change between iterations to avoid overestimating the probability of valid cubes. Then each sampled valid cube gets the same new probability as shown in Equation 2. The probability of each invalid cube is reduced to $\frac{\lambda}{n}$. All the other non-sampled cubes still have a probability of $\frac{1}{n}$, i.e., the initial probability. We also discuss λ value in Section 6.

$$Aprob = \frac{1}{n}|valid_c| + \frac{1-\lambda}{n}|invalid_c|. \quad (1)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

$$prob_{t,j} = \begin{cases} \frac{S(\hat{R}^q_{j,t})}{\sum_{j=1}^{m} S(\hat{R}^q_{j,t})} Aprob & \hat{R}^q_{j,t} \in valid_c \\ \frac{\lambda}{n} & \hat{R}^q_{j,t} \in invalid_c \\ \frac{1}{n} & others \end{cases} \quad (2)$$

As a result, the PSE algorithm outputs terminal importance sampling distribution P^T (line 25).

Algorithm 1 Pilot Sampling Estimation (PSE) algorithm

1: **INPUT:** Query q=$\{A_{up}, Tp_{up}, A_{down}, Tp_{down}\}$, B_{r1}, P^0, ρ, λ, b^0,α, b_{upper}.
2: **OUTPUT:** Importance sampling distribution P^T
3: Initialize $cb = 0$, $t = 0$, valid cube set $valid_c = \emptyset$, invalid cube set $invalid_c = \emptyset$,distinct sampled cube set $Sample = \emptyset$
4: **while** $cb \leq n \times B_{r1}$ **do**
5: Sample $\hat{R}^q_{1,t}, \cdots, \hat{R}^q_{b^t,t} \in \mathcal{R}^{\{A_{up}, Tp_{up}\}}$ according to P^t.
6: $Sample \leftarrow Sample \bigcup \{\hat{R}^q_{1,t}, \cdots, \hat{R}^q_{b^t,t}\}$.
7: **for** $1 \leq i \leq b^t$ **do**
8: **for** $\forall r$ in $\hat{R}^q_{i,t}$ **do**
9: **if** $|\mathcal{R}_r \bigcap \mathcal{R}^{\{A_{down}, Tp_{down}\}}| > 0$ **then**
10: $S(\hat{R}^q_{i,t})$ + = 1
11: **if** $S_{(1-\rho)b^t} \equiv 0$ **then**
12: $b^{t+1} \leftarrow min\{b_{upper}, \alpha b^t\}$
13: $P^{t+1} \leftarrow P^t$
14: **else**
15: $b^{t+1} \leftarrow b^t$
16: **for** $1 \leq i \leq b^t$ **do**
17: **if** $S(\hat{R}^q_{i,t}) > 0$ **then**
18: $valid_c \leftarrow valid_c \bigcup \{\hat{R}^q_{i,t}\}$
19: **else**
20: $invalid_c \leftarrow invalid_c \bigcup \{\hat{R}^q_{i,t}\}$
21: Compute $prob_{t,j}$ in P with Equation 1 and 2.
22: $P^{t+1} \leftarrow P$
23: t + = 1
24: $cb = |Sample|$
25: return terminal importance sampling distribution P^T

5.2 Stage 2: Importance Sampling and Estimation

In this subsection, we introduce our proposed sampling and estimation stage. First we sample b^T ST cubes, $\hat{R}^q_{i,T}$ with $i \in [1, \cdots b^T]$, based on the final importance sampling distribution P^T obtained in stage 1. These are the only samples we draw in Stage 2 thus $b^T = B_2$. Then we design two estimators to calculate the answer to the TTA query. The first one, *Primary Estimator*, only uses those b^T ST cubes sampled with P^T in stage 2. The second one *Integrated Estimator*, utilizes both the b^T cubes sampled in this stage as well as the list of samples drawn during Stage 1 to further improve the result accuracy. Finally, we present the entire algorithm in stage 2 as an Integrated Importance Sampling (IIS) algorithm.

5.2.1 Primary Estimator

The Primary Estimator uses the samples collected with importance sampling distribution P^T (in stage 2) to estimate the query answer. According to P^T, b^T cubes $\hat{R}^q_{i,T}, i \in [1, \cdots, b^T]$ were sampled from q's upstream query region, with replacement. All the sampled cubes are examined and all qualified traversal trajectories from the sampled cubes are identified. If a sampled cube is sampled more than once, we only exhaustively search it at the first time of being sampled, with the count cached. If that cube is sampled again, we will obtain the qualified traversal trajectory count from the cache without costing additional query budget. Given a trajectory r traversing cube $\hat{R}^q_{i,T}$, the number of distinct valid cubes that r traverses in the upstream ST region (denoted as k^q_r) is proportional to the probability that trajectory r is found in any sampled cube under the final importance sampling distribution P^T. Define a mapping f_q: $\mathcal{R}^q \rightarrow \mathbb{R}^+$ as $f_q(\hat{R}^q_{i,T}) = \sum_{r \in \hat{R}^q_{i,T}} \{\frac{1}{k^q_r P^T(\hat{R}^q_i)}\}$. Obviously, the ground truth answer of the TTA query $q = \{A_{up}, Tp_{up}, A_{down}, Tp_{down}\}$ is $N_q = \sum_{i=1}^{n} f_q(R^q_i)$.

Note that after obtaining a qualified trajectory r in Stage 1, the value of k^q_r can be computed by checking the inverted index of r and comparing with cube set in the upstream query region, $k^q_r = |\mathcal{R}_r \bigcap \mathcal{R}^{\{A_{up}, Tp_{up}\}}|$. Next, we develop an unbiased estimator of N_q using sampled cubes $\hat{R}^q_{i,T}, i \in [1, \cdots, b^T]$ collected via the final importance probability distribution P^T. We define primary estimator in Theorem 1 and denote $\hat{R}^q_{i,T}$ and $P^T(\hat{R}^q_i)$ as $\hat{R}^q_i$ and $P(\hat{R}^q_i)$ for short, respectively.

Theorem 1 (Primary Estimator)**.** *With a sampling budget b^T, we collect b^T sampled cubes $\{\hat{R}^q_1, \cdots, \hat{R}^q_{b^T}\}$, with each $\hat{R}^q_i \in \mathcal{R}^q$ for $1 \leq i \leq b^T$. Then, $\hat{N}_q$ in Equation 3 is the asymptotically unbiased estimator of N_q:*

$$\hat{N}_q = \frac{1}{b^T} \sum_{i=1}^{b^T} f(\hat{R}^q_i) = \frac{1}{b^T} \sum_{i=1}^{b^T} \sum_{r \in \hat{R}^q_i} \frac{1}{k^q_r P(\hat{R}^q_i)}. \quad (3)$$

Proof. Since each cube $\hat{R}^q_i, i = 1, \cdots, b^T$ is independently sampled from the same population space $\mathcal{R}^q$ using the law of large number, we have

$$\lim_{b^T \rightarrow \infty} \frac{1}{b^T} \sum_{i=1}^{b^T} f(\hat{R}^q_i) \xrightarrow{a.s.} \sum_{j=1}^{n} f(R^q_j) P(R^q_j)$$

$$\sum_{j=1}^{n} f(R^q_j) P(R^q_j) = \sum_{j=1}^{n} \sum_{r \in R^q_j} \frac{1}{k^q_r P(R^q_j)} P(R^q_j)$$

$$= \sum_{r \in \mathcal{R}^q} \sum_{j=1}^{n} \frac{1}{k^q_r} = N_q.$$

□

Estimator variance analysis. Our PSE stage is designed to yield an importance sampling distribution closer to the ground-truth distribution of qualified trajectories in query ST regions than the uniform distribution, which will be verified in Sec 6 (system evaluation and case studies). With this nice property, the primary estimator is not only unbiased, but also with a lower variance, than that of uniform distribution (i.e., $P(R^q_i) = 1/n$ for all i's) (We omit the proof here for brevity and please refer to [46] for more details).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

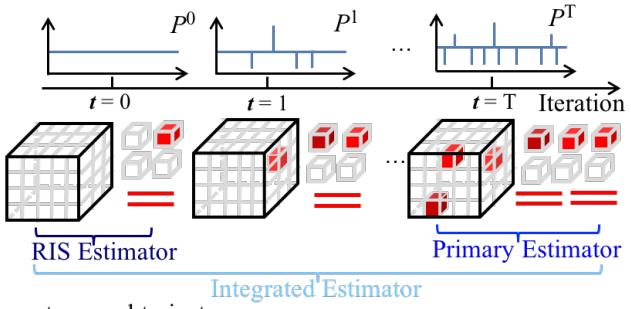


Fig. 5. **An illustration of importance sampling distribution evolutionary process and estimator comparison.**

5.2.2 *Integrated Estimator*

Clearly, the *Primary Estimator* only utilizes the sampled cubes from stage 2, where the samples collected during stage 1 are ignored, and not utilized. To further improve the estimation accuracy, we develop the *Integrated Estimator*, by employing sampled cubes at Stage 1 *Pilot Sampling* as well as sampled cubes from Stage 2. Figure 5 illustrates how the integrated estimator works, with comparison to primary estimator and RIS estimator (proposed in [17]). Theorem 2 below introduce the integrated estimator $\hat{N}'_q$, with proof of its unbiasness.

Theorem 2 (Integrated Estimator). *During Pilot Sampling Estimation (PSE) algorithm in stage 1 and importance sampling in stage 2, a list of sampling distributions $\{P^0, P^1, \cdots, P^T\}$ are obtained with the corresponding samples collected. With a sampling budget b^t ($t \in [0, 1, \cdots, T]$), b^t sampled cubes were collected, i.e., $\{\hat{R}^q_{1,t}, \cdots, \hat{R}^q_{b^t,t}\}$ according to sampling distribution P^t, with each $\hat{R}^q_{i,t} \in \mathcal{R}^q$ for $1 \le i \le b^t$. Denote $B = \sum_{t=0}^{T} b^t$, namely, the total budget (used during stage 1 and stage 2), we define $P(t) = \frac{b^t}{B} = \frac{b^t}{\sum_{j=0}^{T} b^j}$ as the budget proportion at iteration t. Then, the following integrated estimator is an unbiased estimator to the query answer.*

$$\hat{N}'_q = \frac{1}{BT}\sum_{i=1}^{B}\sum_{t=0}^{T} f'(\hat{R}^q_{i,t}) = \frac{1}{BT}\sum_{i=1}^{B}\sum_{t=0}^{T}\sum_{r\in\hat{R}^q_{i,t}} \frac{1}{k^q_r P^t(\hat{R}^q_i)P(t)}. \tag{4}$$

Proof. Since each cube $\hat{R}^q_i, i = 1, \cdots, B$ is independently sampled from the same population space $\mathcal{R}^q$ using the law of large number, we still have

$$\lim_{B\to\infty} \frac{1}{BT}\sum_{i=1}^{B}\sum_{t=0}^{T} f'(\hat{R}^q_{i,t}) \xrightarrow{a.s.} \frac{1}{T}\sum_{j=1}^{n}\sum_{t=0}^{T} f'(R^q_{j,t})P^t(R^q_j)P(t)$$

$$= \frac{1}{T}\sum_{j=1}^{n}\sum_{t=0}^{T}\sum_{r\in R^q_{j,t}} \frac{1}{k^q_r P^t(R^q_j)P(t)} P^t(R^q_j)P(t)$$

$$= \frac{1}{T}\sum_{r\in\mathcal{R}^q}\sum_{j=1}^{n}\sum_{t=0}^{T}\frac{1}{k^q_r} = \frac{1}{T}N_q T = N_q.$$

□

Algorithm 2 Integrated Importance Sampling (IIS) algorithm

1: **INPUT:** Query q=$\{A_{up}, Tp_{up}, A_{down}, Tp_{down}\}$, B_{r1}, B_{r2}, P^0, ρ, λ, b^0,α, b_{upper}, b^T, Estimator.
2: **OUTPUT:** Integrated estimator $\hat{N}'_q$.
3: Initialize $cb = 0$, $t = 0$, $f = 0$, $K^q = \emptyset$, $Sample = \emptyset$, $valid_c = \emptyset$, $invalid_c = \emptyset$.
4: **Stage 1: Pilot Sampling**
5: **while** $cb \le n \times B_{r1}$ **do**
6: Same as line 5−6 in Algorithm 1
7: **for** $1 \le i \le b^t$ **do**
8: **for** $\forall r$ in $\hat{R}^q_{i,t}$ **do**
9: **if** $|\mathcal{R}_r \bigcap \mathcal{R}^{\{A_{down}, Tp_{down}\}}| > 0$ **then**
10: $S(\hat{R}^q_{i,t})$ + = 1
11: $k^q_r = |\mathcal{R}_r \bigcap \mathcal{R}^{\{A_{up}, Tp_{up}\}}|$
12: K^q = $K^q \bigcup k^q_r$
13: Same as line 11−25 in Algorithm 1
14: **Stage 2: Importance Sampling**
15: Sample $\hat{R}^q_{1,T}, \cdots, \hat{R}^q_{b^T,T}$ according to P^T, where $b^T = B_{r2} \times n$.
16: **if** Estimator= *Primary* **then**
17: $\hat{N}_q = \frac{1}{b^T}\sum_{i=1}^{b^T}\sum_{r\in\hat{R}^q_{i,T}} \frac{1}{k^q_r P^T(\hat{R}^q_{i,T})}$
18: **else if** Estimator= *Integrated* **then**
19: **for** $0 \le t \le T$ **do**
20: **for** $1 \le i \le b^t$ **do**
21: **for** $\forall r$ in $\hat{R}^q_{i,t}$ **do**
22: **if** k^q_r not in K^q **then**
23: **if** $|\mathcal{R}_r \bigcap \mathcal{R}^{\{A_{down}, Tp_{down}\}}| > 0$ **then**
24: $k^q_r = |\mathcal{R}_r \bigcap \mathcal{R}^{\{A_{up}, Tp_{up}\}}|$
25: K^q = $K^q \bigcup k^q_r$
26: **else**
27: Find k^q_r in K^q
28: **if** $k^q_r > 0$ **then**
29: f + = $\frac{1}{k^q_r P^t(\hat{R}^q_i)P(t)}$, where $P(t) = \frac{b^t}{\sum_{t=0}^{T} b^t}$
30: $\hat{N}'_q = \frac{f}{BT}$
31: return $\hat{N}'_q$

We summarize the detailed process of sampling and estimating the query answer with integrated estimator as Integrated Importance Sampling (**IIS**) algorithm in Algorithm 2. Besides budget b^T at stage 2 *Importance Sampling*, the input of the IIS algorithm also includes all input parameters of PSE. In addition to initialization of PSE, we also initial the mapping value $f = 0$ and a list K^q recording k^q_r value of each traversal trajectory. For each iteration, IIS also computes k^q_r, the total number of distinct valid cubes that traversal trajectory r traverses for further estimation of N_q (line 11) and records it in the list K^q (line 12). At stage 2 *Importance Sampling*, we generate b^T i.i.d samples $\hat{R}^q_{1,T}, \cdots, \hat{R}^q_{b^T,T}$ according to P^T. The Primary Estimator is computed by using these samples. For the Integrated Estimator, after sampling all given budget B, we finally estimate N_q by checking all sampled cubes throughout the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

TABLE 2
Time Complexity

Algorithm	Time Complexity
Exact Search	$O(n)$
PSE	$O(n \times B_{r1})$
IIS	$O(n \times (B_{r1} + B_{r2}))$

whole sampling process and compute k_r^q for each traversal trajectory (line 19-27). Then we count the mapping value f according to Equation 4 (line 28-29). Finally, IIS returns value of integrated estimator $\hat{N}'_q$.

5.3 Complexity Analysis

In this subsection, we analyze time complexity for both Algorithm 1 *PSE* and Algorithm 2 *IIS*. For PSE, we can assume that fetching each cube and checking traversal trajectories inside takes on average the same time. Then what matters is the number of cubes fetched. Thus, in terms of sampled cubes, the complexity of PSE is $O(n \times B_{r1})$, where n is the number of cubes in the upstream query region.

The time complexity of IIS consists that of stage 1 and stage 2. In terms of sampled cubes, the complexity of IIS with primary estimator is $O(n \times (B_{r1} + B_{r2}))$. For integrated estimator, it takes extra time to collect results from sampled cubes at stage 1 and costs $O(n \times (2 \times B_{r1} + B_{r2})) = O(n \times (B_{r1} + B_{r2}))$ in total. In addition, compared with exact search ($O(n)$), the proposed TIS framework saves running time for sampling process ($B_{r1} + B_{r2} \ll 1$). Table 2 provides a summary of time complexity.

6 Evaluations and Case Studies

In this section, we conduct extensive experiments to evaluate our proposed TIS framework using a taxi trajectory dataset collected from Shenzhen, China in November, 2014. Specifically, we evaluate under varying query budgets (1) the quality of the importance distribution obtained by the PSE algorithm, (2) the final query result accuracy of TIS, (3) the total running time of TIS, and (4) the impact of cube size choice on result accuracy. For (1), (2) and (4) we use the Random Index Sampling (RIS) [17], which employs a uniform distribution sampling approach, as the baseline for comparison. For (3) we compare the run-time of TIS with that of RIS and an exact search (ES) on the upstream query region. The ES method utilizes the 3-D grid index introduced in Section 4 to retrieve all the trajectories in the upstream region and verifies if each trajectory also traverses the downstream query region using the inverted index. The ES algorithm always returns the accurate result.

6.1 Dataset and Experiment Settings

We use a large-scale trajectory dataset collected from taxis in Shenzhen, China, with an urban area of about 400 square miles and three million people. The dataset was collected for 30 days in November, 2014. These trajectories represent 21,490 unique taxis. They are equipped with GPS sets, which periodically (i.e., roughly every 30 seconds) generate GPS records. For each taxi, we combine all its GPS points in a day to form one single trajectory. Thus we obtained 21,490 distinct trajectories for each day.

TABLE 3
Dataset Descriptions

Statistics	Value
City Size	400 square miles
City Population	three million people
Duration	30 days in November, 2014
Number of taxis	21,490 unique taxis
Number of trajectories	2.3 billion (2,335,874,676)

TABLE 4
Evaluation Configurations

Item	Settings
Budget ratio at stage 1	[10%,20%]
Budget ratio at stage 2	[1%,3%,$\cdots$, 15%]
Candidate methods	TIS vs. RIS vs. ES
Grid Granularity	[250m, 500m, 750m, 1000m]
Time Interval	[5 minutes]

The TIS parameter values we use for all experiments are sample size $b^0 = 20$, percentile $\rho = 0.95$, smooth parameter $\lambda = \frac{1}{3}$, augmented parameter $\alpha = 1.1$ and upper bound of sample size $b_{upper} = 100$, which are obtained via performance tuning. Intuitively, larger sample size and smaller percentile could reduce the number of iteration for sampling but also decrease the chance to better estimate the importance distribution. Besides, parameter λ controls the degree of change in the sampling probability. A value that is too small leads to trivial update while a value that is too large leads to overfitting of the sampled cubes. The current value setting is the best for datasets similar to ours. The settings could be tuned to fit other data in future.

Each experiment is run 200 times and the average results are reported. All the experiment are conducted on a DELL PowerEdge R370 rack server, with 2x12-core Intel Xeon E5-2690 processors(2.6 GHz/30M Cache) and 192 GB memory. Data statistics and detailed experiment settings are summarized in Table 3 and Table 4, respectively.

6.2 Benchmark Query Descriptions

In order to evaluate our proposed TIS framework, we generate three different types of queries with different trajectory distribution in the upstream region as the benchmark. Figure 6 compares the percentages of cubes with each distinct number of qualified trajectories. Query Type 1 has over 75% cubes with no qualified trajectory and 15% with only 1. Query Type 2 has around 40% cubes with no qualified trajectory and most valid cubes contain no more than 5 qualified trajectories. Query Type 3 has more diverse distribution of counts among the cubes, ranging from 0 to 19. Table 5 describes three types of queries we generated in our evaluations. As discussed in Section 3.3, rareness is the percentage of cubes in the upstream ST region that contain qualified trajectories. Note that one traversal trajectory could traverse several valid cubes so the sum of all the cube counts might be larger than the total number of distinct traversal trajectories of a particular query.

Case Studies: In order to better illustrate the different query types, *we randomly generate one example query for each query type introduced above, respectively*. The details of the example queries are presented in Table 5. Figure 7, 8,and 9

TABLE 5
Query Descriptions

Type #	R Range (%) [a]	$\{A_{up}, Tp_{up}\}$	$\{A_{down}, Tp_{down}\}$	Tra Traj (#) [b]	Rareness (%)	Valid Cu(#)[c]	Total Cu(#)[c]
1	(0,40%)	$\{Shekou, 6-9am\}$	$\{Jixia, 6-9am\}$	21	14.29	252	1764
2	[40%, 60%)	$\{Buji, 5-8am\}$	$\{Bantian, 6-9am\}$	382	54.17	936	1728
3	[60%, 100%)	$\{Futian, 6-9am\}$	$\{Center, 6-9am\}$	3207	85.19	184	216

[a] Rareness range of a particular query type. [b] The total number of distinct traversal trajectories satisfying query demands. [c] The number of valid cube and total cubes in $\{A_{up}, Tp_{up}\}$.

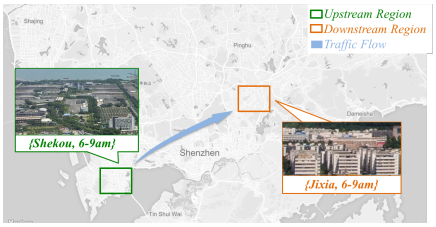


Fig. 7. **Example of Query Type 1**

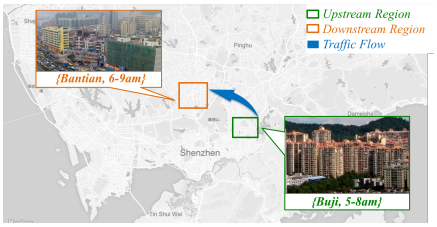


Fig. 8. **Example of Query Type 2**

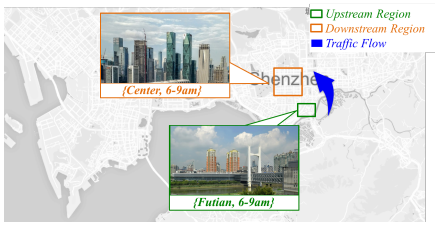
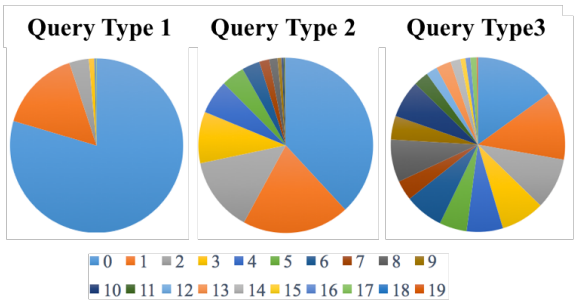


Fig. 9. **Example of Query Type 3**



* count value of distinct traversal trajectories traversing a given cube

Fig. 6. **Cube trajectory count distribution of the three types of queries.**

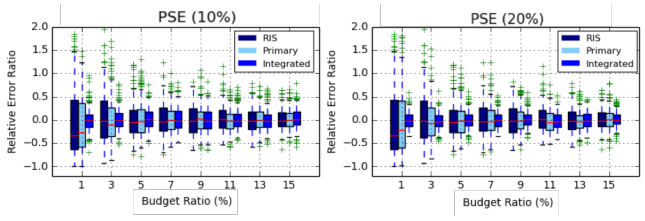


Fig. 13. **Estimation results of example Query Type 1.**

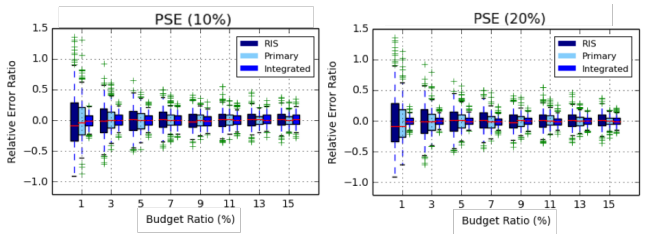


Fig. 14. **Estimation results of example Query Type 2.**

visualize the query regions and traffic flow of the three queries on our real dataset. Example Query Type 1 has the smallest traffic flow as well as rareness. Because $Shekou$ is a manufacturing district and it rarely occurs that people transit from here to $Jixia$, a residential district in morning rush hours. In contrast, it is common that lots of people transit from $Futian$, a port area to downtown $Center$, a commercial district during the same rush hour time period as shown in Figure 9. Meanwhile, Figure 8 illustrates that many people live in $Buji$ village and work at nearby $Bantian$, a high-tech industrial zone. Overall, in morning rush hours, people always transit from residential districts or transportation hubs to work area. But it is very rare to observe the opposite.

6.3 Evaluation of the PSE Algorithm

In order to evaluate Pilot Sampling Estimation (PSE) algorithm, we sample cubes according to the terminal importance sampling distribution P^T and uniform distribution P^0, respectively, with varying budget ratios and compare how many valid cubes can be sampled using each method.

Figure 10, Figure 11 and Figure 12 show the comparison of total identified valid cubes in the three benchmark queries. We increase the budget ratio B_{r1} from 10% to 20%. Results show that with the terminal importance sampling distribution P^T obtained by the PSE algorithm, we are able to identify more valid cubes with qualified traversal trajectories than using a uniform distribution. Besides, the gaps between PSE and the baseline (uniform distribution) grow bigger as we increase the budget ratio. Particularly, the gap reduces as we move from Query Type 1 to Query Type 2 and then Query Type 3, suggesting that the PSE algorithm works always better when cube rareness is high, i.e., the distribution is far from non-uniform.

6.4 Evaluation of the IIS Algorithm

Besides identified valid cubes in upstream query region, we also use the IIS algorithm to estimate the total number of distinct traversal trajectories for each example query of Query Type 1, 2 and 3. To evaluate the accuracy of the estimation result, we define the *relative error ratio of* an estimation as

$$\epsilon(\hat{N}) = \frac{\hat{N} - N}{N}$$

which is the normalized difference from the ground truth value N. All of Figure 13, Figure 14 and Figure 15 show the box plot of the relative error ratios of each benchmark query. For each one, we draw two box plots with different sampling budget ratio B_{r1}: 10% and 20% in Stage 1. We vary the budget ratio B_{r2} in Stage 2 from 1% to 15% with a step of 2%. The box plots display differences between populations, where the spacings between the different parts

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

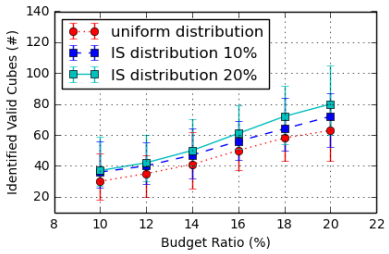


Fig. 10. **Comparison of Sampling results of Query Type 1**

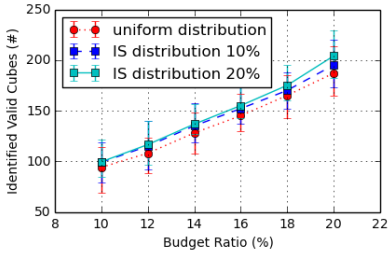


Fig. 11. **Comparison of sampling results of Query Type 2**

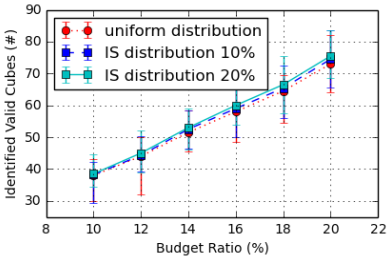


Fig. 12. **Comparison of sampling results of Query Type 3**

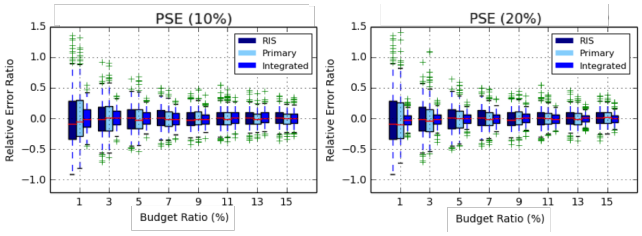


Fig. 15. **Estimation results of example Query Type 3.**

of the box indicate the degree of dispersion (spread) and skewness in the data, and show outliers. The bottom and top of the box are always the first and third quartiles filled in different degree of blue color, and the band inside the box is always the second quartile (the median) in red line. The lowest datum still within 1.5 interquartile (IQR) of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile [47]. The whiskers are represented by dashed blue line. Any data not included between the whiskers are plotted as an outlier with the marker '+' in green.

From left to right, bars in each group with different degree of blue represent the RIS algorithm, the primary estimator via terminal importance sampling distribution, and the integrated estimator via Integrated Importance Sampling (IIS) algorithm, respectively. Results show that all estimators are asymptotically unbiased in the box plot of each query. With budget ratios 10% and 20% in Stage 1 pilot sampling, it can be concluded that the baseline method (RIS) fluctuates while the integrated estimator via utilizing IIS algorithm always performance best and converges at budget ratio B_{r2} around 3%.

6.5 Evaluation on Runtime

In addition to accuracy, we also evaluate the query processing time of the entire IIS algorithm with both Stages 1 and 2 for the three queries with varying Stage 2 budget ratio B_{r2} values. For each query we choose different budget ratios B_{r1} for Stage 1 (10% and 20%). We also compare the run time with RIS, where the runtime is calculated by running RIS for upstream ST query region.

Figure 16-18 show the runtime in seconds. The runtime of IIS grows linearly with Stage 2 budget ratio from 1% to 15% but always less than 1 second. By contrast, the ES method takes 55, 53, and 39 seconds, respectively. Also lower budget ratio in Stage 1 leads to less runtime in Stage 2. Although IIS has an extra time cost compared to RIS due to the pilot sampling and estimation steps, the total query processing time is still within 1 second for all the queries.

Figure 19-21 show the runtime reduction rate against Exact Search (ES) over the same index. The runtime reduction is the percentage of computational time saved compared to the cost of the Exact Search (ES) method. Note that the Exact Search (ES) method also uses the same grid index and the inverted index described in Section 4. Therefore, the performance of ES has been optimized. So the comparison is fair. However, ES is still much slower than the proposed IIS algorithm.Each figure represents a benchmark query and the three bars in each color represent one baseline (RIS) and the IIS with two different choices of budget ratios in Stage 1 (10% and 20%). For all the parameter settings, IIS saves at least 98% query time of the ES, while savings for Query Type 3 is the highest.

6.6 Evaluation on Granularity

Finally, we evaluate our algorithms with different grid granularity settings to understand how the grid partitioning affects our algorithms' performance. We conduct experiments with four different spatio-temporal granularities varying from (250 meters -250 meters -5 min) to (1000 meters -1000 meters -5 min). Figure 22 shows that our IIS algorithm is more accurate and more stable than RIS algorithm with varying grid granularity for all the three types of queries. The relative error ratio of IIS is always within 10% while that of RIS fluctuates between 20% and 50%.

Note when using a finer granularity, the trajectory distribution will become sparser among cubes and the count in each cube will decrease. Therefore choosing very small grid size might not always give the best result. Taking this factor into consideration, we set (500 meters -500 meters -5 min) as our default granularity for our dataset. However, this parameter should be adjusted accordingly on different datasets.

Summary: The proposed PSE algorithm can better estimate the distribution of traversal trajectories in the query region than the baseline method RIS. The entire IIS algorithm achieves better accuracy and stability compared to RIS under varying budget ratio and grid size. The runtime of IIS is still orders of magnitudes faster than simple exact search using spatial index (ES) and comparable with RIS.

7 Discussion

In this section, we extend our traversal trajectory aggregate query from a pair of upstream and downstream ST regions to a sequence of different query regions. Throughout this paper, we take all example queries with upstream and downstream spatio-temporal query regions as a simplified

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

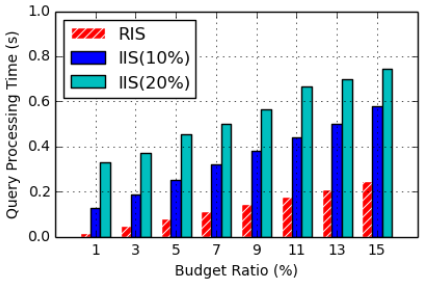


Fig. 16. **Processing Time of Query Type 1 (ES=53s)**

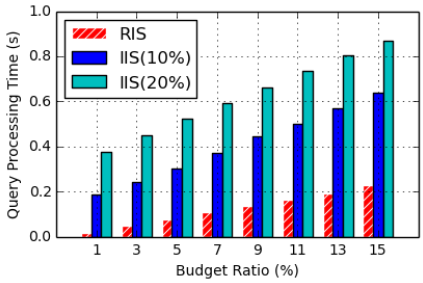


Fig. 17. **Processing Time of Query Type 2 (ES=55s)**

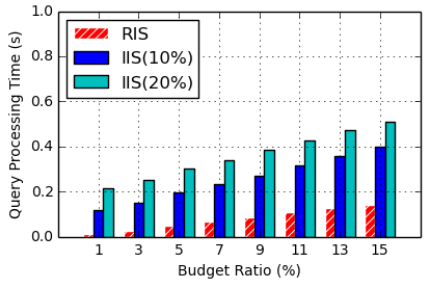


Fig. 18. **Processing Time Query Type 3 (ES=39s)**

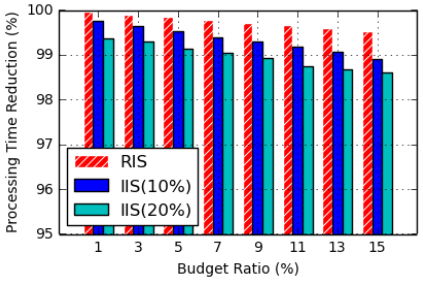


Fig. 19. **Processing Time Reduction for Query Type 1**

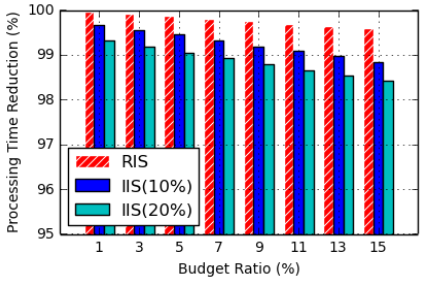


Fig. 20. **Processing Time Reduction for Query Type 2**

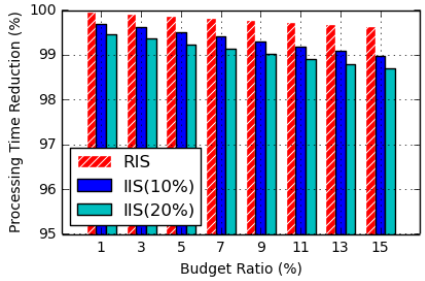
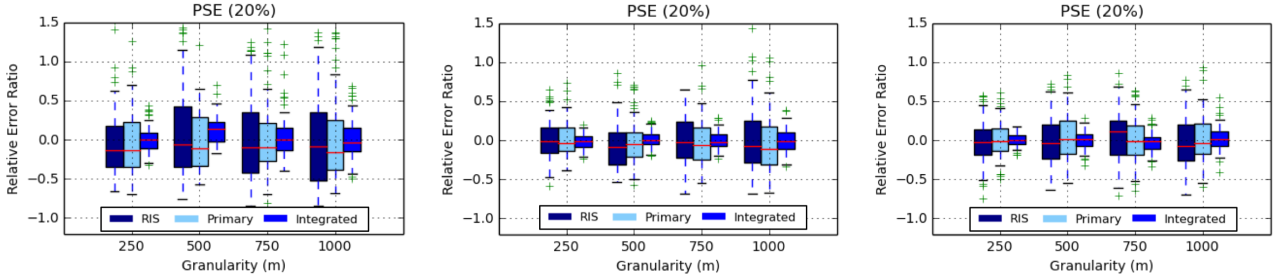


Fig. 21. **Processing Time Reduction for Query Type 3**



(a) Query Type 1 (b) Query Type 2 (c) Query Type 3

Fig. 22. **Result error with varying granularity.**

example of traversal trajectory aggregate query. In fact, our proposed TIS framework with the PSE and IIS algorithms are generic to complex traversal trajectory aggregate querye.g., with more than two spatio-temporal query regions. As illustrated in Figure 23, where blue lines represent trajectories traversing subset of given query regions while the red line represents one qualified trajectory sequentially traverse all m (> 2) query regions. Each cuboid identifies one spatio-temporal query region including three dimensions which are latitude, longitude and time. Taking an example query, we briefly discuss how our sampling and estimation algorithms can be applied to it by checking more than one query regions. We also demonstrate evaluation results of this example query.

Extension Query. Given an example of extension query $q = \{Jixia,\ 6-9am, Center, 11am-2pm, Tourist, 3-7pm\}$ with $m = 3$, the rareness of the first spatio-temporal query region $\{Jixia, 6-9am\}$ is 58.06%, i.e., 418 valid cubes out of 720 total cubes. According to query type classification listed in Table 5, it belongs to Query Type 2. Intuitively, it should have similar evaluation result compared to examples of Query Type 2. Figure 24 and 25 show comparisons between TIS and RIS on the relative error for the extension query and the number of valid cubes identified, respectively. The trends are similar as in Figure 11 and Figure 14: our

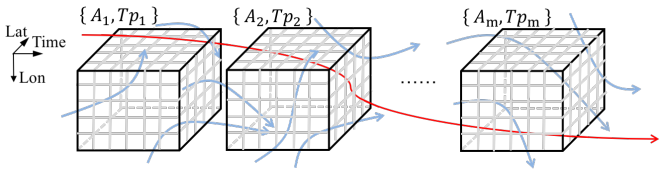


Fig. 23. **An illustration of a traversal trajectory aggregate query extension to traverse a sequence of different $m(\geq 3)$ spatio-temporal query regions.**

proposed TIS framework outperforms RIS for the extension query with lower error rate.

Figure 26 shows that the TIS consumes more time for the extension query than simple TTA queries as shown in Figure 17. This is intuitive as for the extension query we need to check two other query regions to verify each sampled trajectory, compared to only one other query region in the simple TTA queries. However there is no additional time cost introduced in the sampling/estimation phases. Figure 27 shows that, similar to Figure 20, the IIS algorithm of the proposed TIS framework still saves up to 98.8% running time of ES.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2018.2830780, IEEE Transactions on Big Data

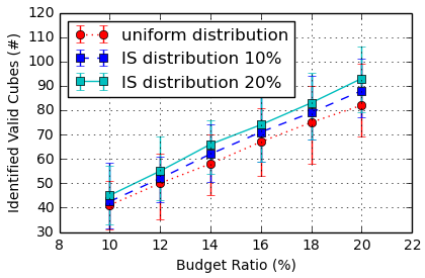


Fig. 25. **Sampling results of example extension query**

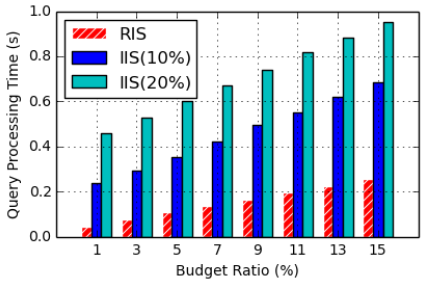


Fig. 26. **Processing time of example extension query (ES=60s)**

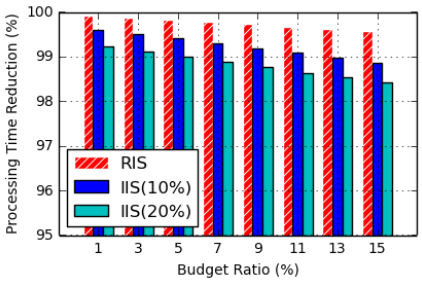


Fig. 27. **Time Reduction Rate of example extension query**

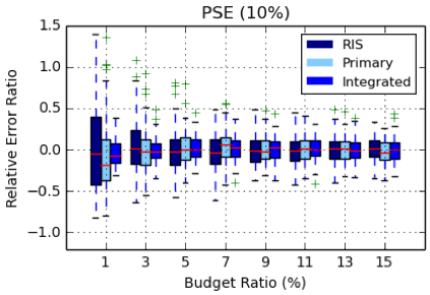


Fig. 24. **Estimation results of example extension query.**

8 Conclusion

This paper defined and investigated a novel trajectory query, the Traversal Trajectory Aggregate (TTA) Query. TTA queries play an important role in supporting various urban intelligent applications, such as vehicle route planning, taxi dispatching, and location-based advertising. Exact solutions such as exhaustive searching may lead to extremely long running time on large query regions, while traditional sampling-based approaches assume uniform distribution of trajectories in the ST query region and may lead to significant estimation variance. This paper proposed a novel Targeted Index Sampling (TIS) framework with a Pilot Sampling Estimation (PSE) stage to estimate the importance distribution of trajectories in ST query region, an Integrated Importance Sampling (IIS) stage to collect trajectory samples with obtained importance sampling distribution to estimate the query result with an asymptotically unbiased estimator. Extensive experiments obtained using a large-scale real taxi trajectory dataset demonstrated that our TIS framework achieves $\leq 10\%$ estimation error with $\geq 90\%$ computational time reduction over exact search, and 50% reduction on estimation error (with similar running time) over uniform distribution based sampling approaches.

Acknowledgments

This work is partially supported by the NSF under Grant Number IIS-1566386. Yanhua Li was partially supported by the NSF CRII grant CNS-1657350 and a research grant from Pitney Bowes Inc.

References

[1] N. J. Yuan, Y. Zheng, and X. Xie, "Segmentation of urban areas using road networks," *MSR-TR-2012–65, Tech. Rep.*, 2012.

[2] J. Bao, T. He, S. Ruan, Y. Li, and Y. Zheng, "Planning bike lanes based on sharing-bikes' trajectories," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1377–1386.

[3] Y. Zheng and X. Zhou, *Computing with spatial trajectories*. Springer Science & Business Media, 2011.

[4] S. Liu, Y. Yue, and R. Krishnan, "Non-myopic adaptive route planning in uncertain congestion environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2438–2451, 2015.

[5] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 2015, pp. 543–554.

[6] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 254–265.

[7] A. V. Khezerlou, X. Zhou, L. Li, Z. Shafiq, A. X. Liu, and F. Zhang, "A traffic flow approach to early detection of gathering events: Comprehensive results," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 6, p. 74, 2017.

[8] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang, "On discovery of gathering patterns from trajectories," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 242–253.

[9] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, "Towards mobile intelligence: Learning from gps history data for collaborative recommendation," *Artificial Intelligence*, vol. 184, pp. 17–37, 2012.

[10] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008, pp. 140–149.

[11] T. V. Le, S. Liu, H. C. Lau, and R. Krishnan, "Predicting bundles of spatial locations from learning revealed preference data," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1121–1129.

[12] Z. Chen, H. T. Shen, and X. Zhou, "Discovering popular routes from trajectories," in *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 2011, pp. 900–911.

[13] G. Wu, Y. Ding, Y. Li, J. Bao, Y. Zheng, and J. Luo, "Mining spatio-temporal reachable regions over massive trajectory data," in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017, pp. 1283–1294.

[14] A. Vahedian, X. Zhou, L. Tong, Y. Li, and J. Luo, "Forecasting gathering events through continuous destination prediction on big trajectory data," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2017, Redondo Beach, CA, USA, November 7-10, 2017*, 2017, pp. 34:1–34:10.

[15] B. Babcock, S. Chaudhuri, and G. Das, "Dynamic sample selection for approximate query processing," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003, pp. 539–550.

[16] J. M. Hellerstein, M. J. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman, and M. A. Shah, "Adaptive query processing: Technology in evolution," *IEEE Data Eng. Bull.*, vol. 23, no. 2, pp. 7–18, 2000.

[17] Y. Li, C.-Y. Chow, K. Deng, M. Yuan, J. Zeng, J.-D. Zhang, Q. Yang, and Z.-L. Zhang, "Sampling big trajectory data," in *Proceedings*

of the 24th ACM International on Conference on Information and Knowledge Management. ACM, 2015, pp. 941–950.

[18] A. B. Owen, *Monte Carlo theory, methods and examples*, 2013.

[19] I. Kamel and C. Faloutsos, "Hilbert r-tree: An improved r-tree using fractals," Tech. Rep., 1993.

[20] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: an efficient and robust access method for points and rectangles," in *ACM Sigmod Record*, vol. 19, no. 2. Acm, 1990, pp. 322–331.

[21] R. Zhong, G. Li, K.-L. Tan, and L. Zhou, "G-tree: An efficient index for knn search on road networks," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 39–48.

[22] D. Pfoser, C. S. Jensen, Y. Theodoridis *et al.*, "Novel approaches to the indexing of moving object trajectories." in *VLDB*, 2000, pp. 395–406.

[23] Y. Tao and D. Papadias, "Efficient historical r-trees," in *Scientific and Statistical Database Management, 2001. SSDBM 2001. Proceedings. Thirteenth International Conference on*. IEEE, 2001, pp. 223–232.

[24] ——, "The mv3r-tree: A spatio-temporal access method for timestamp and interval queries," in *Proceedings of Very Large Data Bases Conference (VLDB), 11-14 September, Rome*, 2001.

[25] L. Wang, Y. Zheng, X. Xie, and W.-Y. Ma, "A flexible spatio-temporal indexing scheme for large-scale gps track retrieval," in *Mobile Data Management, 2008. MDM'08. 9th International Conference on*. IEEE, 2008, pp. 1–8.

[26] V. P. Chakka, A. C. Everspaugh, and J. M. Patel, "Indexing large trajectory data sets with seti," *Ann Arbor*, vol. 1001, no. 48109-2122, p. 12, 2003.

[27] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[28] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[29] C. S. Jensen, D. Lin, and B. C. Ooi, "Query and update efficient b+-tree based indexing of moving objects," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 768–779.

[30] D. Comer, "Ubiquitous b-tree," *ACM Computing Surveys (CSUR)*, vol. 11, no. 2, pp. 121–137, 1979.

[31] J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted files versus signature files for text indexing," *ACM Transactions on Database Systems (TODS)*, vol. 23, no. 4, pp. 453–490, 1998.

[32] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, p. 29, 2015.

[33] P. Yuan, Q. Zhao, W. Rao, M. Yuan, and J. Zeng, "Searching k-nearest neighbor trajectories on road networks," in *Australasian Database Conference*. Springer, 2017, pp. 85–97.

[34] H. Wang, R. Zimmermann, and W.-S. Ku, "Distributed continuous range query processing on moving objects," in *International Conference on Database and Expert Systems Applications*. Springer, 2006, pp. 655–665.

[35] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*. IEEE, 2007, pp. 350–364.

[36] S. Shang, L. Chen, C. S. Jensen, J.-R. Wen, and P. Kalnis, "Searching trajectories by regions of interest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 7, pp. 1549–1562, 2017.

[37] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie, "Searching trajectories by locations: an efficiency study," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 255–266.

[38] L.-A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, and J. Han, "Retrieving k-nearest neighboring trajectories by a set of point locations," in *International Symposium on Spatial and Temporal Databases*. Springer, 2011, pp. 223–241.

[39] R. H. Güting, T. Behr, and J. Xu, "Efficient k-nearest neighbor search on moving object trajectories," *The VLDB JournalThe International Journal on Very Large Data Bases*, vol. 19, no. 5, pp. 687–714, 2010.

[40] P. B. Gibbons and Y. Matias, "New sampling-based summary statistics for improving approximate query answers," in *ACM SIGMOD Record*, vol. 27, no. 2. ACM, 1998, pp. 331–342.

[41] S. Chaudhuri, G. Das, and V. Narasayya, "Optimized stratified sampling for approximate query processing," *ACM Transactions on Database Systems (TODS)*, vol. 32, no. 2, p. 9, 2007.

[42] J. Zhou, Y. Li, V. K. Adhikari, and Z.-L. Zhang, "Counting youtube videos via random prefix sampling," in *IMC'11: ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 371–380.

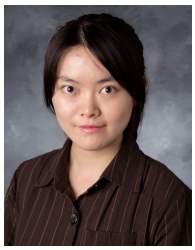
[43] Y. Li, M. Steiner, J. Bao, L. Wang, and T. Zhu, "Region sampling and estimation of geosocial data with dynamic range calibration," in *ICDE'14: The 30th International Conference on Data Engineering*, 2014, pp. 1–12.

[44] P. Wang, W. He, and X. Liu, "An efficient sampling method for characterizing points of interests on maps," in *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*. IEEE, 2014, pp. 1012–1023.

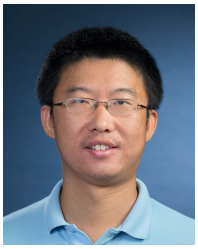
[45] ——, "Efficiently estimating statistics of points of interests on maps," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 2, pp. 425–438, 2016.

[46] S. P. Meyn and R. L. Tweedie, *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

[47] P. J. Rousseeuw, I. Ruts, and J. W. Tukey, "The bagplot: a bivariate boxplot," *The American Statistician*, vol. 53, no. 4, pp. 382–387, 1999.



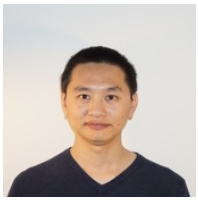
Yichen Ding received a Bachelor's degree in Statistics and a MS in Data Science. She is currently a Ph.D. student in the Department of Management Sciences at Tippie College of Business, the University of Iowa. Her current research interests include big data analytics, spatial temporal data mining and urban computing.



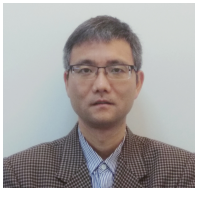
Yanhua Li (S09-M13-SM16) received two Ph.D. degrees in electrical engineering from Beijing University of Posts and Telecommunications, Beijing in China in 2009 and in computer science from University of Minnesota at Twin Cities in 2013, respectively. He has worked as a researcher in HUAWEI Noah's Ark LAB at Hong Kong from Aug 2013 to Dec 2014, and has interned in Bell Labs in New Jersey, Microsoft Research Asia, and HUAWEI research labs of America from 2011 to 2013. He is currently an Assistant Professor in the Department of Computer Science at Worcester Polytechnic Institute (WPI) in Worcester, MA. His research interests are urban network data analytics, smart cities, data-driven cyber-physical systems (CPS).

Xun Zhou is currently an Assistant Professor in the Department of Management Sciences at the University of Iowa. He received a PhD degree in Computer Science from the University of Minnesota, Twin Cities in 2014. His research interests include big data management and analytics, spatial and spatio-temporal data mining, and Geographic Information Systems (GIS). He has published over 30 papers in these areas and has received three best paper awards. He also served as a co-editor-in-chief of the Encyclopedia of GIS, 2nd Edition.

Zhuojie Hunag is a Principal Data Scientist in Pitney Bowes since 2014. He received his Phd degree in geography from the University of Florida in 2013. His current research interests are spatial data analytics, spatial temporal data mining, spatial logistics model and GIS. He has published papers in the areas of population mapping, remote sensing, GIS, location based social network analysis, public health and epidemiology.



Simin You is a Research Scientist in Pitney Bowes since 2015. He received the B.E. degree in Computer Science from University of Science and Technology of China in 2009, and Ph.D. degree in Computer Science from the Graduate Center of City University of New York in 2016. His current research interests include big spatial data management, parallel computing on GPUs and high performance Geographical Information Systems (GIS).



Jun Luo is a principal researcher at Lenovo Machine Intelligence Center in Hong Kong. He received his PhD degree in computer science from the University of Texas at Dallas, USA, in 2006. His research interests include big data, machine learning, spatial temporal data mining and computational geometry. He has published over 90 journal and conference papers in these areas.