Sampling Big Trajectory Data

Yanhua Li[†], Chi-Yin Chow[#], Ke Deng[*], Mingxuan Yuan[$],
Jia Zeng[♯♮$], Jia-Dong Zhang[#], Qiang Yang[**], and Zhi-Li Zhang[‡]
[†]Worcester Polytechnic Institute, Worcester, MA, [#]City University of Hong Kong, Hong Kong
[*]RMIT University, Australia, [**]Hong Kong University of Science and Technology, Hong Kong
[♯]Soochow University, China, [♮]Novel Software Technology and Industrialization, China
[$]Noah's Ark Lab, Hong Kong, [‡]University of Minnesota, Twin Cities
yli15@wpi.edu

ABSTRACT

The increasing prevalence of sensors and mobile devices has led to an explosive increase of the scale of spatio-temporal data in the form of trajectories. A *trajectory aggregate query*, as a fundamental functionality for measuring trajectory data, aims to retrieve the statistics of trajectories passing a user-specified spatio-temporal region. A large-scale spatio-temporal database with big disk-resident data takes very long time to produce exact answers to such queries. Hence, approximate query processing with a guaranteed error bound is a promising solution in many scenarios with stringent response-time requirements. In this paper, we study the problem of approximate query processing for trajectory aggregate queries. We show that it boils down to the distinct value estimation problem, which has been proven to be very hard with powerful negative results given that no index is built. By utilizing the well-established spatio-temporal index and introducing an inverted index to trajectory data, we are able to design random index sampling (RIS) algorithm to estimate the answers with a guaranteed error bound. To further improve system scalability, we extend RIS algorithm to concurrent random index sampling (CRIS) algorithm to process a number of trajectory aggregate queries arriving concurrently with overlapping spatio-temporal query regions. To demonstrate the efficacy and efficiency of our sampling and estimation methods, we applied them in a real large-scale user trajectory database collected from a cellular service provider in China. Our extensive evaluation results indicate that both RIS and CRIS outperform exhaustive search for single and concurrent trajectory aggregate queries by two orders of magnitude in terms of the query processing time, while preserving a relative error ratio lower than 10%, with only 1% search cost of the exhaustive search method.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Systems – Query processing; H.4.0 [**Information Systems and Applications**]: General

General Terms

Algorithms, Experimentation.

CIKM'15, October 19–23, 2015, Melbourne, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806422.

Keywords

Spatio-temporal databases, trajectory aggregate query, sampling, approximate query processing

1. INTRODUCTION

A trajectory stands for a sequential time-stamped geo-locations in a three dimensional spatio-temporal space. In reality, a trajectory may contain a variety of attributes, for example, a taxi GPS trajectory has the instant or average velocity, trajectory length, occupation indicator, etc. [36]; A Flickr user's trajectory may infer users' sequential activities, (e.g., what he/she has done) from the multimedia contents attached to the locations (e.g., text, images and videos). These trajectory data facilitate many emerging applications including urban planning [36, 19], spatio-temporal data mining [21, 15], and various location-based services [33, 20]. *A trajectory aggregate query*, as a fundamental functionality of spatial-temporal databases, aims to retrieve the statistics of distinct trajectories passing a user-specified spatio-temporal region [4, 8]. A typical trajectory aggregate query is "the total number of taxi trajectories with speed greater than 5 miles per hour in New York City during 2013".

The increasing prevalence of sensors and mobile devices, such as GPS set on cars or smart phones carried by people, and the fast development of location-acquisition technologies have led to an explosive increase of the scale of spatio-temporal databases in the form of trajectories, which generates new challenges, namely, how to efficiently process numerous trajectory aggregate queries in large-scale trajectory databases.

Various spatio-temporal indexing techniques, such as augmented R-tree [24], multi-version R-tree [30, 31], and grid-based index [5], have been proposed to divide the spatio-temporal space into indices and facilitate the access to spatio-temporal data. These spatio-temporal indices are of hierarchical structure, namely, multiple adjacent index nodes in a lower level are aggregated to an index node in a higher level. When processing spatio-temporal queries, the index nodes are typically browsed in a top-down manner. Many spatio-temporal aggregate queries, such as counting the total spatio-temporal records, can be answered by aggregating the information maintained in index nodes at the higher levels to avoid accessing the raw spatio-temporal data. However, for a trajectory aggregate query, maintaining the statistical trajectory information on index nodes does not work. The reason is that a trajectory aggregate query aims to find the number of distinct trajectories in a spatio-temporal query region. To determine whether two index nodes contain some trajectories in common, the trajectory IDs must be recorded in every index node. Using such an index structure makes no difference from the brute-forcing. Thus, to produce the exact answer to a trajectory aggregate query, a non-avoidable brute-force search has to be performed, which takes very long time when the spatio-temporal query region covers a large amount of disk-resident data. As a result, approximate query processing becomes a promising solution in many scenarios with stringent response-time requirements, which provides (reasonably accurate) approximate answers.

In this paper, we make the first attempt to investigate the problem of approximate query processing for trajectory aggregate queries. By introducing an inverted index to trajectory data, we design **R**andom **I**ndex **S**ampling (RIS) algorithm to estimate the answers with guaranteed error bounds. To further improve system scalability, we design **C**oncurrent **R**andom **I**ndex **S**ampling (CRIS) algorithm to process a number of trajectory aggregate queries arriving concurrently with overlapping spatio-temporal query regions. We highlight our contributions as follows.

- We show that the approximate query processing for trajectory aggregate queries boils down to the distinct value estimation problem [6], which has been proven to be very hard with powerful negative results given that no index is built. By utilizing the well-established spatio-temporal index and introducing an inverted index to the trajectory data, we develop random index sampling (RIS) algorithm and unbiased estimator to quickly answer the query with bounded estimation error.
- For a large-scale trajectory database management system, a large number of trajectory aggregate queries usually arrive concurrently with overlapping spatio-temporal query regions. Instead of performing RIS algorithm on each individual query, we further extend it to *concurrent random index sampling* (CRIS) algorithm to significantly improve the efficiency of answering concurrent queries, by employing stratified sampling technique and reusing samples collected from the overlapping regions.
- We conduct extensive experiments on a real-world large-scale user trajectory dataset (with 3TB data) collected from a cellular service provider in China to evaluate the efficiency and effectiveness of our sampling and estimation methods. Our experimental results show that both RIS and CRIS outperform an exhaustive search method for single and concurrent trajectory queries by two orders of magnitude in terms of the query processing time, while preserving a relative error ratio lower than 10%, with only 1% search cost of the exhaustive search method.

The remainder of this paper is organized as follows. Section 2 discusses related work. We define our problem in Section 3. In Sections 4 and 5, we present our proposed RIS and CRIS algorithms, respectively. Section 6 analyzes the performance of RIS and CRIS algorithms. Section 7 describes some generalizations of our algorithms. Finally, we conclude this paper in Section 8.

2. RELATED WORK

We make the first attempt to employ sampling techniques to improve the effectiveness in processing trajectory aggregate queries in large-scale spatio-temporal databases, with the aim to significantly reduce the query processing time while preserving certain estimation accuracy. In this section, we discuss three topics that are closely related to our work and highlight the differences from them, including (1) trajectory data management, (2) distinct value estimation, and (3) concurrent query processing.

2.1 Trajectory Data Management

Trajectory queries. As moving object trajectories have been collected in a massive scale due to the advancement of sensor technologies like satellite, RFID, GPS, and mobile cellular networks, the query processing of trajectory databases has been paid much attention. Trajectory queries aim to evaluate various spatio-temporal relationships among spatial data objects, such as regions, points, and trajectories [8]. The trajectory queries can be generally classified into two main categories: range (or window) queries and nearest-neighbor queries. *Range queries* retrieve trajectories passing a given spatio-temporal region [4, 8] or search spatio-temporal regions which are frequently passed by trajectories [16, 18]. *Nearest-neighbor queries* ask for the top-k nearest-neighbor trajectories to a specified point or trajectory [13, 32]. In this paper, we particularly focus on a sub-category of the trajectory range queries, namely, a trajectory aggregate query, which returns the statistics (e.g., counts, sum, or average) of all distinct trajectories in the spatio-temporal query region. Traditional methods to handle trajectory queries are through spatio-temporal indices, as discussed below.

Executing trajectory queries using database indices. It is important to utilize the appropriate indexing techniques for trajectory data to improve the efficiency and effectiveness of processing trajectory queries. The trajectory database has unique requirements to the index techniques because of its spatio-temporal data characteristics [8]. The state-of-the-art trajectory indexing techniques can be grouped into three categories, including augmented R-tree, multi-version R-tree, and grid-based index. (1) An *augmented R-tree* employs R-trees over the 3D spatio-temporal space, e.g., Spatio-Temporal R-tree (STR-tree) and TB-tree (Trajectory Bundle tree) [24]. (2) A *multi-version R-tree* builds multiple structures of R-trees. For each time stamp, an R-tree is created, and R-trees for different time stamps are indexed, e.g., historical R-tree (HR-tree) [30] and multi-version B-trees and 3D R-trees (MV3R-trees) [31]. (3) A *grid-based index* divides the spatial dimension into grids using Quad-trees [9] or KD-trees [1], and then builds a separate temporal index for each grid, e.g., multiple time-split B-tree (MTSB-trees) [37] and Scalable and Efficient Trajectory Index (SETI) [5] to divide the spatio-temporal space into indices and facilitate the spatio-temporal data access.

However, as elaborated in Section 1, to produce the exact answers to a trajectory aggregate query, a non-avoidable brute-force search has to be performed, which takes very long time when the spatio-temporal query region covers a large amount of disk-resident data. As a result, approximate query processing becomes a promising solution for trajectory aggregate queries in many scenarios with stringent response-time requirements. In a spatio-temporal database indexed by one of the index structures, we propose to estimate the answers to the trajectory aggregate queries by sampling a limited number of index leaf nodes. Our proposed random index sampling (RIS) algorithm can significantly reduce the query processing time while preserving certain estimation accuracy. Moreover, RIS algorithm can be applied (and is orthogonal) upon various trajectory indexing methods.

2.2 Distinct Value Estimation

In a column of a large table with each entry filled with a value with no index built, distinct value estimation problem aims to design a scalable approach to generate accurate estimation to the number of distinct values, with limited space or time resources. When time is a major constraint, random sampling is the only approach, which collects samples from the table, and estimates the total number of distinct values. Unfortunately, the existing works

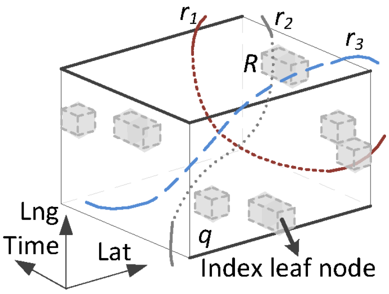


Figure 1: A trajectory aggregate query q returning the number of distinct trajectories in the spatio-temporal query region depicted in the cuboid. r_1, r_2, and r_3 are three trajectories.

show powerful *negative* results [6, 7, 14, 23] that no estimator can guarantee small error across all input distributions, unless it examines a large fraction of the input data. On the other hand, when space is a primary concern, scanning the entire table is allowed, synopses based approaches [2, 12, 17] yield reasonably accurate estimation to the number of distinct values. Our trajectory aggregate query problem boils down to the distinct value estimation problem with stringent time constraint, when sampling is performed upon spatio-temporal points of trajectories, with the trajectory IDs viewed as the "distinct values". To tackle this seemingly unsolvable problem, we in this paper employ spatio-temporal index and inverted index to allow obtaining the probability of each trajectory being collected while sampling, thus enabling the design of unbiased estimator for the number of trajectories with bounded variance and confidence interval.

2.3 Concurrent Query Processing

A database, especially a large-scale spatio-temporal database, usually has to deal with numerous concurrent queries from a large number of users. Thus, it is important and desirable to process these concurrent queries through exploiting a shared execution paradigm to avoid redundant computations and save the computational cost. In general, there are mainly three shared execution paradigms. (1) *Multi-query optimization.* This technique detects and re-uses common sub-expressions among queries by operating on batches of queries only during the optimization phase and materializing shared intermediate results at the cost of memory [27]. (2) *Simultaneous pipelining.* This paradigm extracts identical sub-plans from concurrent queries, executes only one of them and sends the results to the rest simultaneously [25]; however, it is limited to common sub-plans, that is, it is not applicable to the case that two queries have similar sub-plans but with different selection predicates. (3) *Global query plans.* This paradigm utilizes shared operators (with fine granularity) rather than sub-plans (with coarser granularity) to increase sharing opportunities, in which a single shared operator can evaluate two queries sharing a similar plan simultaneously [25]. In this paper, we apply the global query plans paradigm to utilize the sharing opportunities, the key challenge of which is to decompose the operators (i.e., sampling) into shared sub-operators and reconstruct the results for all concurrent trajectory range queries. We employ stratified sampling and sample reuse to solve the problem. (See Section 5 for more details.)

3. PROBLEM STATEMENT

In this section, we clarify key terms used in the paper and formally define the objective of approximate query processing for trajectory aggregate queries. Table 1 provides the notations and terminologies used in this paper.

Table 1: Notations and terminologies

Notation	Description
$Q = \{q_1, \cdots, q_M\}$, $Q_j \subseteq Q$, $1 \leq j \leq M'$	Q is a set of M concurrent trajectory aggregate queries. Q_j is a subset of concurrent queries in Q.
$Q' = \{q'_1, \cdots, q'_{M'}\}$, $Q'_i \subseteq Q'$, $1 \leq i \leq M$	Q' is a set of non-overlapping spatio-temporal regions. Q'_i is a subset of spatio-temporal regions in Q'.
q, q_0	q is a trajectory aggregate query. $q_0 = \vee_{q_i \in Q} q_i$ is the union of the spatio-temporal regions of all concurrent queries in Q.
$\mathcal{R}^q = \{R^q_1, \cdots, R^q_n\}$	$\mathcal{R}^q$ is the set of n index leaf nodes that q's spatio-temporal query region covers.
B	B is the sampling budget (i.e., the maximum number of sampled index leaf nodes) for answering q.
$\hat{R}^q_t$, $1 \leq t \leq B$	$\hat{R}^q_t$ is the t-th sampled index leaf node.
k^q_r	The number of index leaf nodes in $\mathcal{R}^q$ that trajectory r traverses.
N_q, $\hat{N}_q$	N_q is the number of trajectories that match the query constraints of q. $\hat{N}_q$ is the estimator of N_q.

- A **trajectory** is the sequence of spatial points that a moving object follows through space as a function of time. Each point thus consists of an trajectory ID, latitude, longitude, and a time stamp. For example, the GPS points of a vehicle and the spatio-temporal points of a cellphone user are all trajectories.

- **Trajectory attributes:** A trajectory may be associated with a few features describing the properties of the trajectory, e.g., the number of spatio-temporal points, length, average speed, etc.

- **Trajectory data structure:** Trajectory data are usually stored in a database using a spatio-temporal index, such as Quad-tree/R-tree (for spatial indexing) and B/B$^+$-tree (for temporal indexing)[1]. Then, as shown in Figure 1, each *index leaf node* represents a three dimensional subspace, containing spatio-temporal points of trajectories.

- A **trajectory aggregate query** q returns a statistical result of distinct trajectories in a spatio-temporal query region, with or without constraints on trajectories attributes, e.g., ranges on average speed, trajectory length, and etc. Figure 1 illustrates an example of a trajectory aggregate query.

As discussed in Section 1, to produce exact answers to a trajectory aggregate query q, a non-avoidable brute-force search has to be performed, which takes very long time when the spatio-temporal

[1]Note that there are many spatio-temporal index structures in the literature [3, 11], and our random index sampling algorithms can be applied (and is orthogonal) to different indexing methods.

query region covers big disk-resident data. In this paper, we employ sampling methods to perform *approximate query processing* for trajectory aggregate queries with guaranteed error bounds.

Problem definition. Given a trajectory aggregate query q in a large-scale trajectory database, the database returns the total number of distinct trajectories N_q in q's spatio-temporal query region, with certain attribute constraints. Suppose q's query region covers n spatio-temporal index leaf nodes $\mathcal{R}^q = \{R_1^q, \cdots, R_n^q\}$. We aim to sample and search B index leaf nodes (where $B \ll n$), $\{\hat{R}_1^q, \cdots, \hat{R}_B^q\}$ from $\mathcal{R}^q$, based on which we provide an accurate estimator $\hat{N}_q$ that converges to N_q, and for any $\epsilon > 0$ and $0 < \alpha \leq 1$, a certain confidence interval is guaranteed as $Pr(|\hat{N}_q - N_q| > \epsilon) \leq \alpha$.

4. RANDOM INDEX SAMPLING

In this section, we introduce our proposed random index sampling (RIS) algorithm to sample index leaf nodes in the spatio-temporal query region of a trajectory aggregate query q, and design an unbiased estimator to estimate the answer to q, with provable bounds on the confidence interval.

4.1 Model of Random Index Sampling

RIS algorithm works in three stages as follows, inverted indexing stage, sampling stage, and estimating stage.

Inverted indexing stage. An inverted index [38] is an index data structure storing a mapping from content, such as numbers or words, to its locations. We build an inverted index for the trajectory data, where each record represents a trajectory and all index leaf nodes the trajectory traverses (as shown in Figure 2). Given a trajectory aggregate query q, we denote k_r^q as the number of index leaf nodes that a trajectory r goes through in q's spatio-temporal query region. k_r^q can be quickly obtained by checking the record of r in the inverted index. It will be clear shortly that k_r^q governs the probability of a trajectory r being sampled in *sampling stage* and k_r^q is an important variable to design estimator to the answer of q.

Sampling stage. Let B denote the sampling budget, i.e., the maximum number of index leaf nodes allowed to collect. In our analysis, we assume that B is always sufficiently large. We uniformly at random pick up B index leaf nodes from the leaf node set covered by q's spatio-temporal query region, i.e., $\mathcal{R}^q$, with replacement[2]. Thus, each index leaf node in $\mathcal{R}^q$ has an equal probability of $1/n$ being sampled. Then, we scan the sampled index leaf nodes and find the trajectories of our interest, namely, those matching the trajectory attribute constraints of q. If a sampled index leaf node is chosen more than once, we only perform one exhaustive search on it. Hence, as shown in Figure 2, we obtain a list of sampled index leaf nodes, $\{\hat{R}_1^q, \cdots, \hat{R}_B^q\}$, from the population space $\mathcal{R}^q$. Note that a sample $\hat{R}_t^q \in \mathcal{R}^q$ represents both the index leaf node and the trajectory set in it, where we will interchangeably use these two meanings of $\hat{R}_t^q$ in the rest of the paper ($1 \leq t \leq B$).

Various statistics of a sampled index leaf node $\hat{R}_t^q$ can be represented as functions of $\hat{R}_t^q$ ($1 \leq t \leq B$). For instance, the number of distinct trajectories that match the query contraints of q can be viewed as a mapping $g_q : \mathcal{R}^q \to \mathbb{N}$ from a trajectory set $\hat{R}_t^q \in \mathcal{R}^q$ to a natural number, i.e., $g_q(\hat{R}_t^q)$. Moreover, given a trajectory $r \in \hat{R}_t^q$, the number of index leaf nodes that r traverses in $\mathcal{R}^q$, i.e.,

[2]Note that in general q's query region is *much larger* than the index leaf node. Hence, for simplicity, we consider all trajectories in an index leaf node R_i^q traverse q's query region, if q partially or completely covers R_i^q in the spatio-temporal space. On the other hand, when the query region of q is small, exhaustive search can be directly applied to obtain q's exact answer.

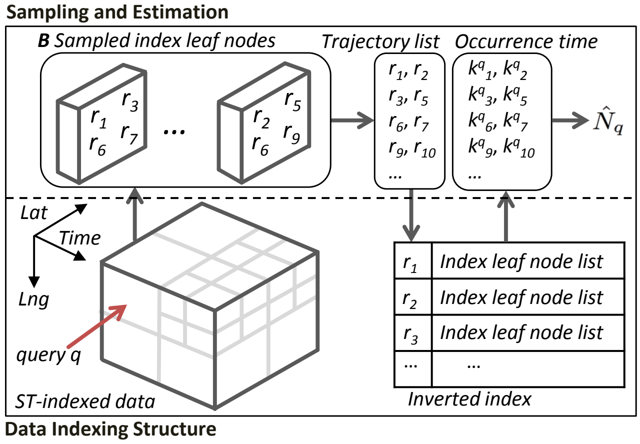


Figure 2: Random index sampling framework

k_r^q, is proportional to the probability that the trajectory r being sampled by randomly picking up an index leaf node from $\mathcal{R}^q$. Define a mapping $f_q : \mathcal{R}^q \to \mathbb{R}^+$ as $N_t^q = f_q(\hat{R}_t^q) = \sum_{r \in \hat{R}_t^q \wedge q} \{1/k_r^q\}$, where $\hat{R}_t^q \wedge q$ represents the trajectory set in $\hat{R}_t^q$ that matches the attribute constraints of q. It is clear that the ground-truth answer for the trajectory aggregate query q, namely, the total number of distinct trajectories in $\mathcal{R}^q$, is $N_q = \sum_{i=1}^n f_q(R_i^q)$. Note that (as shown in Figure 2) after a trajectory r is obtained from $\hat{R}_t^q \in \mathcal{R}^q$, the number of index leaf nodes r traverses in $\mathcal{R}^q$, i.e., k_r^q, can be found by checking the inverted index.

Estimating stage. Using RIS, Theorem 1 presents that the probability of each trajectory being sampled is proportional to the number of index leaf nodes it traverses in $\mathcal{R}^q$.

THEOREM 1 (IMPORTANCE SAMPLING). *The probability $Pr(r)$ of each trajectory $r \in \mathcal{R}^q \wedge q$ being sampled using RIS algorithm is $Pr(r) = k_r^q/n$, where $\mathcal{R}^q = \{R_1^q, \cdots, R_n^q\}$.*

PROOF. Each index leaf node R_i^q ($1 \leq i \leq n$) has an equal probability $Pr(R_i^q) = 1/n$ to be chosen. The conditional probability of each trajectory r being sampled given that R_i^q is sampled is $Pr(r|R_i^q) = 1$, if $r \in R_i^q$; and $Pr(r|R_i^q) = 0$, otherwise. Applying the law of total probability yields the probability of r being sampled as $Pr(r) = \sum_{i=1}^n Pr(R_i^q)Pr(r|R_i^q) = k_r^q/n$. □

In the next subsection, we develop an unbiased estimator to estimate the total number of distinct trajectories that match the query constraints of q, using the samples $\{\hat{R}_q^1, \cdots, \hat{R}_B^q\}$ collected by RIS algorithm.

4.2 Estimator Design

An estimator is a function of a sequence of observations that outputs an estimate of an unknown population parameter. Below, we present an asymptotically unbiased estimator of N_q.

Unbiased estimator. We aim to estimate the total number N_q of distinct trajectories that match the query constraints of q. We define an estimator $\hat{N}_q$ in Theorem 2.

THEOREM 2 (ESTIMATOR TO N_q). *With a sampling budget B, RIS algorithm collects B sampled index leaf nodes $\{\hat{R}_1^q, \cdots, \hat{R}_B^q\}$, with each $\hat{R}_t^q \in \mathcal{R}^q$ for $1 \leq t \leq B$. Then, $\hat{N}_q$ in Equation (1) is the asymptotically unbiased estimator of N_q:*

$$\hat{N}_q = \frac{n}{B} \sum_{t=1}^{B} f_q(\hat{R}_t^q). \tag{1}$$

PROOF. Since each $\hat{R}_t^q$ is chosen uniformly at random from the same population space $\mathcal{R}^q$, $f_q(\hat{R}_t^q)$'s are all independent and follow the same distribution, and thus have the same expectation as $E[f_q(\hat{R}_t^q)] = \mu = \sum_{i=1}^{n} f_q(R_i^q)/n = N_q/n$. Using the linearity of the expectation, $E[\hat{N}_q] = N_q$. $\square$

Sampling Budget Analysis. Under a certain trajectory index structure, such as Quad-tree and B-tree, all the trajectory records and indices are stored as regular files. Thus, the size of an index leaf node is bounded by the file size in the underlying file system. For example, on Hadoop Distributed File System (HDFS), the block file size is 64MB, and when in-block index is built, a smaller memory page size (e.g., 4KB) is used [34]. As a result, given that each trajecotory record takes roughly a constant amount of space, e.g., 100B, the number of distinct trajectories in an index leaf node is bounded, e.g., by $4KB/100B = 40$, and we denote this upper bound as δ. Hence, for a trajectory aggregate query q, $f_q(\hat{R}^q) \leq \delta$ holds. Theorem 3 below determines the sampling budget B needed to achieve a certan estimation accuracy.

THEOREM 3. *The estimator $\hat{N}_q$ (Equation (1)) guarantees that for any $\epsilon > 0$ and $0 < \alpha \leq 1$,*

$$Pr(|\hat{N}_q - N_q| > \epsilon) \leq \alpha, \tag{2}$$

when $B \geq \frac{\ln(2/\alpha)\delta^2 n^2}{2\epsilon^2}$, with δ as the maximum number of trajectories in an index leaf node.

PROOF. Recall that each sample $\hat{R}_t^q \in \mathcal{R}^q$ is drawn uniformly at random from $\mathcal{R}^q$. Hence, random variables $f_q(\hat{R}_t^q)$ are independent and identically distributed, and they follow the same distribution. Given that each $f_q(\hat{R}^q) \leq \delta$ holds, the following bound holds by applying Hoeffding's inequality [28],

$$Pr(|\hat{N}_q - N_q| > \epsilon) \leq 2e^{\frac{-2B\epsilon^2}{n^2\delta^2}}. \tag{3}$$

Then, by letting the right hand side (in Equation (3)) be smaller than or equal to α, we obtain $B \geq \frac{\ln(2/\alpha)\delta^2 n^2}{2\epsilon^2}$ as the minimum budget needed to guarantee the confidence interval in Equation (2). $\square$

It is clear that to achieve a certain confidence interval, governed by ϵ and α, the minimum sampling budget needed satisfies $B \propto \delta^2$ and $B \propto n^2$.

5. CONCURRENT RANDOM INDEX SAMPLING

The proposed RIS algorithm properly deals with a *single* trajectory aggregate query with a guaranteed estimation error bound. However, when it comes to queries in a large-scale trajectory database management system, e.g., user mobility data from carriers in a country with millions of users, a large number of trajectory range queries come concurrently. These queries may overlap in spatio-temporal query regions, as well as ranges of attribute constraints. As illustrated in Figure 3(a), two concurrent queries q_1 and q_2 arrive concurrently and overlap with each other in the spatio-temporal space.

A naive method for handling concurrent queries is to perform RIS algorithm individually for each query. However, by reusing the samples obtained in the overlapping space among queries, the estimation accuracy can be significantly improved. In this section, we extend RIS algorithm to concurrent random index sampling (CRIS) algorithm, that performs *stratified sampling* and *sample reuse* on concurrent trajectory aggregate queries. Our theoretical results show that CRIS algorithm can achieve higher estimation accuracy for each concurrent trajectory aggregate query, with less sampling budget than simply running RIS algorithm for each query. Below, we first formally define the problem of concurrent overlapping queries, and then present our CRIS algorithm.

5.1 Model of Concurrent Random Index Sampling

Two queries q_1 and q_2 are concurrent, if the database system processes them at the same time (i.e., they arrive at the database system or are dispatched from a query buffer at the same moment). Their spatio-temporal query regions may overlap, as well as ranges of trajectory attributes, such as average speed, trajectory length, etc. In the rest of the paper, we will use *concurrent overlapping queries* to represent concurrent queries with overlapping spatio-temporal constraints. Figure 3(a) shows an example of two concurrent overlapping queries. Now, we formally define the problem of concurrent overlapping queries as follows.

Concurrent overlapping queries. Given $M > 1$ concurrent overlapping queries, $Q = \{q_1, \cdots, q_M\}$, with required estimation accuracy guarantees $\epsilon_i > 0$ and $0 < \alpha_i \leq 1$ for $i \in \{1, \cdots, M\}$. The goal is to provide estimations to the numbers of trajectories, $N_{q_1}, \cdots, N_{q_M}$, with guaranteed confidence interval as $Pr(|\hat{N}_{q_i} - N_{q_i}| > \epsilon_i) \leq \alpha_i$. The major challenge is how to reuse the index leaf nodes sampled from overlapping regions of concurrent queries and improve the estimation accuracy over that of running RIS algorithm on each individual query.

Concurrent random index sampling. We propose CRIS algorithm to deal with concurrent overlapping queries. Given M concurrent trajectory aggregate queries, $Q = \{q_1, \cdots, q_M\}$, let $q_0 = q_1 \vee \cdots \vee q_M$ be the entire spatio-temporal region Q covers. Different from RIS algorithm, the basic idea behind CRIS algorithm is that we first divide q_0 into $M' \geq M$ non-overlapping spatio-temporal regions $Q' = \{q'_1, \cdots, q'_{M'}\}$, by the boundaries of all queries q_i's ($1 \leq i \leq M$). Each q'_j corresponds to an intersection of a subset of queries $Q_j \subseteq Q$. Then, CRIS samples index leaf nodes from the leaf node set $\mathcal{R}'_j$ of each q'_j with sampling budget B'_j ($1 \leq j \leq M'$), where B'_j is porportionally allocated with respect to the number of index leaf nodes in q'_j. Eventually, for each region q'_j, CRIS algorithm collects B'_j index leaf nodes, denoted as $\hat{\mathcal{R}}'_j = \{\hat{R}_1^{q'_j}, \cdots, \hat{R}_{B'_j}^{q'_j}\}$ for $1 \leq j \leq M'$. These sampled index leaf nodes are exhaustively searched for trajectories matching the attribute constraints of queries in Q. We prove that such design of CRIS reduces the estimation variance and the needed sampling budget over that of running RIS algorithm independently on each query q_i.

5.2 Two Concurrent Overlapping Queries

Now, we use two concurrent overlapping queries to show how CRIS algorithm works. Given two concurrent overlapping queries, q_1 and q_2 as shown in Figure 3(a). The goal is to provide estimations to the numbers of distinct trajectories, N_{q_1} and N_{q_2}, with guaranteed estimation confidence interval, $Pr(|\hat{N}_{q_1} - N_{q_1}| > \epsilon_1) \leq \alpha_1$ and $Pr(|\hat{N}_{q_2} - N_{q_2}| > \epsilon_2) \leq \alpha_2$. CRIS algorithm consists of three stages as follows.

Stratification Stage. The entire spatio-temporal space, defined as $q_0 = q_1 \vee q_2$ is divided into three non-overlapping queries, i.e., $Q' = \{q'_1, q'_2, q'_3\}$, such that $q'_3 := q_1 \wedge q_2$, $q'_1 = q_1 - q_3$, and $q'_2 = q_2 - q_3$. Moreover, we denote $\mathcal{R}'_1$, $\mathcal{R}'_2$, and $\mathcal{R}'_3$ as the corresponding non-overlapping leaf node sets, as illustrated in Figure 3(b). Hence, q_1 (resp. q_2) is divided into two strata, i.e., $q_1 = q'_1 \vee q'_3$ (resp. $q_2 = q'_2 \vee q'_3$).

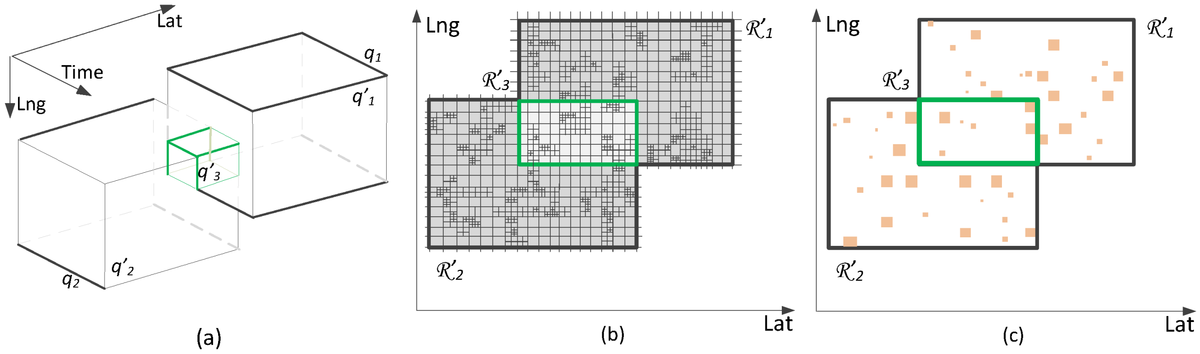


Figure 3: Illustration of CRIS algorithm. (a) Two concurrent overlapping queries q_1 and q_2 are mapped to a three dimensional space. They are divided into non-overlapping regions, q_1', q_2', and q_3'. (b) The leaf node sets $\mathcal{R}_1'$, $\mathcal{R}_2'$, and $\mathcal{R}_3'$; (c) Given sampling budgets needed for q_1 and q_2 as $B_1 = B_2 = 28$, q_1', q_2', and q_3' are assigned with budgets $B_1' = 21$, $B_2' = 21$, and $B_3' = 7$ (based on Equation (4)), respectively. Orange boxes represent the sampled index leaf nodes, based on which the numbers of distinct trajectories N_{q_1} and N_{q_2} are estimated by Equations (5) and (6), respectively.

Sampling Stage. Based on Theorem 3, there exist lower bounds B_1 and B_2 on sampling budgets for q_1 and q_2 to guarantee the estimation accuracy, defined by ϵ_1, ϵ_2, α_1, and α_2. Then, each stratum q_j' is sampled individually with a sampling budget B_j' proportional to n_j' as

$$\begin{aligned} &B_1' = [B_1 n_1'/n_1], \quad B_2' = [B_2 n_2'/n_2], \\ &B_3' = \max\{[n_3' B_1/n_1], [n_3' B_2/n_2]\}. \end{aligned} \tag{4}$$

Then, three sample sets, i.e., $\{\hat{R}_1^{q_j'}, \cdots, \hat{R}_{B_j'}^{q_j'}\}$ for $1 \leq j \leq 3$, are drawn uniformly at random from leaf node sets $\mathcal{R}_1'$, $\mathcal{R}_2'$, and $\mathcal{R}_3'$, respectively (See Figure 3(c)).

Estimating Stage. The three sample sets are observations used to estimate N_{q_1} and N_{q_2}. By similar proof for Theorem 2, $\hat{N}_{q_1}$ and $\hat{N}_{q_2}$ in Equation (5) and Equation (6) are proven to be the asymptotically unbiased estimators of N_{q_1} and N_{q_2}.

$$\hat{N}_{q_1}^{CRIS} = \frac{n_1}{B_1}\left(\sum_{t=1}^{[B_1 n_1'/n_1]} f_{q_1}(\hat{R}_t^{q_1'}) + \sum_{t'=1}^{[B_1 n_3'/n_1]} f_{q_1}(\hat{R}_{t'}^{q_3'})\right), \tag{5}$$

$$\hat{N}_{q_2}^{CRIS} = \frac{n_2}{B_2}\left(\sum_{t=1}^{[B_2 n_2'/n_2]} f_{q_2}(\hat{R}_t^{q_2'}) + \sum_{t'=1}^{[B_2 n_3'/n_2]} f_{q_2}(\hat{R}_{t'}^{q_3'})\right), \tag{6}$$

where $f_q(R) = \sum_{r \in R \vee q} \frac{1}{k_r^q}$.

In next subsection, we prove that for two concurrent overlapping queries, CRIS algorithm outperforms RIS algorithm with smaller estimation variance and less sampling budget.

5.3 Multiple Concurrent Overlapping Queries

Given concurrent queries $Q = \{q_1, \cdots, q_M\}$, the entire spatio-temporal region of those queries is denoted as $q_0 = \vee_{q \in Q} q$. CRIS partitions q_0 into non-overlapping spatio-temporal regions, so each region represents an intersection of a unique subset of queries in Q. Taking the partition as stratification for concurrent queries, we randomly sample index leaf nodes from each region, and reuse the samples to estimate the answers of queries that cover that region. Below we elaborate the details of stratifying q_0 and performing sampling and estimations. The detailed CRIS algorithm is summarized in Algorithm 1.

Stratification. We divide q_0 into $M' \geq M$ non-overlapping queries by the boundaries of q_i's, i.e., $Q' = \{q_1', \cdots, q_{M'}'\}$, such that $q_0 = \vee_{q' \in Q'} q'$ holds. Let $Q_j \subseteq Q$ be a subset of Q with the size as $1 \leq |Q_j| \leq M$, and $\bar{Q}_j = Q - Q_j$ be the complementary set of Q_j with the size as $|\bar{Q}_j| = M - |Q_j|$. Each $q_j' = \wedge_{q \in Q_j} q - \vee_{q' \in \bar{Q}_j} q'$ with $1 \leq j \leq M'$ corresponds to a region that only covers the intersection of all regions in Q_j, but not any region in $\bar{Q}_j$, as shown in Figure 3(a). Denote the index leaf node set of q_j' as $\mathcal{R}_j'$ with n_j' index leaf nodes. $\mathcal{R}_j'$'s are non-overlapping with each other. Let $Q_i' \subseteq Q'$ with $1 \leq i \leq M$ be a subset of Q' such that $q_i = \vee_{q' \in Q_i'} q'$ holds, namely, each q_i is divided (or stratified) into strata, i.e., Q_i', which enables us to perform sampling on each q_j'. Note that every index leaf node in the population is assigned to only one stratum and no index leaf node can be excluded.

Algorithm 1 Concurrent random index sampling (CRIS) algorithm

INPUT: Queries $Q = \{q_1, \cdots, q_M\}$, with $\epsilon_i > 0$, $0 < \alpha_i \leq 1$.
OUTPUT: Sample Sets $\hat{\mathcal{R}}_j' = \{\hat{R}_1^{q_j'}, \cdots, \hat{R}_{B_j'}^{q_j'}\}$, $1 \leq j \leq M'$.
Computer budgets $\{B_1, \cdots, B_M\}$ by Theorem 3.
Generate $\{Q_1, \cdots, Q_{M'}\}$; Each $Q_j \subseteq Q$ and $q_j' = \wedge_{q \in Q_j} q - \vee_{q' \in \bar{Q}_j} q' \neq \phi$.
Keep the non-overlapping region set $Q' = \{q_1', \cdots, q_{M'}'\}$;
Generate non-overlapping leaf node sets $\{\mathcal{R}_1', \cdots, \mathcal{R}_{M'}'\}$;
Generate $\{Q_1', \cdots, Q_M'\}$; Each $Q_i' \subseteq Q'$ and $q_i = \vee_{q' \in Q_i'} q'$.
for $1 \leq j \leq M'$ **do**
    Each sampling budget $B_j' = \max_{q_i \in Q_j} [B_i n_j'/n_i]$;
    Sample leaf nodes from each $\mathcal{R}_j'$ with budget B_j' to obtain $\hat{\mathcal{R}}_j'$;

Sampling on strata. From Theorem 3, the estimation accuracy requirement ϵ_i and α_i, with $1 \leq i \leq M$, determines the sampling budget B_i needed for each q_i. Then, the leaf node set $\mathcal{R}_j'$ of the stratum q_j' is sampled independently with a sampling budget $B_j' \propto n_j'$, namely, proportionate allocation is employed to ensure that the sampling budget fraction in each stratum q_i' is proportional to n_j'. To be precise, for each query q_i, stratum $q_j' \in Q_i'$ is assigned with $B_j'(q_i) = [B_i n_j'/n_i]$ sampling budget. Instead of collecting samples from q_j' repeatedly for each $q_i \in Q_j$, with a total budget of $\tilde{B}_j' = \sum_{q_i \in Q_j} B_j'(q_i)$, we collect only $B_j' = \max_{q_i \in Q_j} B_j'(q_i)$ samples from q_j' and allow sample reuse when performing estimations for different $q_i \in Q_j$. Obviously, $B_j' < \tilde{B}_j'$.

Estimator Design. After the sampling process, a total of M' sample sets are drawn randomly from leaf node sets $\mathcal{R}_1', \cdots, \mathcal{R}_{M'}'$.

They are expressed as

$$\hat{\mathcal{R}}'_1 = \{\hat{R}_1^{q'_1}, \cdots, \hat{R}_{B'_1}^{q'_1}\}, \cdots, \hat{\mathcal{R}}'_{M'} = \{\hat{R}_1^{q'_{M'}}, \cdots, \hat{R}_{B'_{M'}}^{q'_{M'}}\}.$$

These sample sets are observations for estimating $N_{q_1}, \cdots, N_{q_M}$. Below we first present the asymptotically unbiased estimator of N_{q_i} in Theorem 4. Then, we discuss the corresponding estimation accuracy.

THEOREM 4. *Using CRIS algorithm, $\hat{N}_{q_i}$ in Equation (7) is the asymptotically unbiased estimator of N_{q_i}.*

$$\hat{N}_{q_i}^{CRIS} = \frac{n_i}{B_i} \sum_{q'_j \in Q'_i} \sum_{t=1}^{\lceil B_i n'_j / n_i \rceil} f_{q_i}(\hat{R}_t^{q'_j}), \tag{7}$$

where $f_q(R) = \sum_{r \in R \vee q} \frac{1}{k_r^q}$.

PROOF. Since each $\hat{R}_t^{q'_j}$ is chosen uniformly at random from the same population space $\mathcal{R}'_j$, $f_{q_i}(\hat{R}_t^{q'_j})$'s with $1 \le t \le B'_j$ are all independent and follow the same distribution. Thus, they have the same expectation as $E[f_{q_i}(\hat{R}_t^{q'_j})] = \sum_{R \in \mathcal{R}'_j} f_{q_i}(R)/n'_j$. Using the linearity of the expectation, we prove $E[\hat{N}_{q_i}] = N_{q_i}$ for each $1 \le i \le M$. □

Performance comparison to RIS algorithm.

THEOREM 5. *For $M > 1$ concurrent overlapping queries, CRIS algorithm achieves lower estimation variance with less sampling budget than RIS algorithm.*

PROOF. Consider M concurrent overlapping queries $Q = \{q_1, \cdots, q_M\}$, with accuracy requirements $\epsilon_i > 0$ and $0 < \alpha_i \le 1$, $i \in \{1, \cdots, M\}$. Below, we prove (1) CRIS requires less number of samples than individually applying RIS; (2) for the estimation accuracy of each query $q \in Q$, CRIS always outperforms RIS.
(1)*Sampling budget.* By Theorem 3, the sampling budgets $B_1, \cdots, B_M$ of M queries in Q can be obtained. With sample reuse, the total budget spent using CRIS algorithm is $B^{CRIS} = \sum_{j=1}^{M'} B'_j = \sum_{j=1}^{M'} \max_{q_i \in Q_j} \lceil B_i n'_j / n_i \rceil \le \sum_{j=1}^{M'} \sum_{q_i \in Q_j} \lceil B_i n'_j / n_i \rceil = \sum_{i=1}^{M} B_i = B^{RIS}$.
(2)*Estimation accuracy.* Given a query $q \in Q$, CRIS divides q into $M_q \ge 1$ non-overlapping spatio-temporal regions (strata), q'_j's, with $1 \le j \le M_q$, with each q'_j covering n'_j index leaf nodes. Denote $\mathcal{R}^q$ and $\mathcal{R}'_j$ as the index leaf node sets of q and q'_j, respectively. We now prove that given the same sampling budget B, the estimation variances (when using RIS and CRIS algorithms) follow $Var[\hat{N}_q^{CRIS}] \le Var[\hat{N}_q^{RIS}]$.

Recall that each sample $\hat{R}_t^q \in \mathcal{R}^q$ is drawn uniformly at random from $\mathcal{R}^q$. Hence, for RIS, each random variable $f_q(\hat{R}_t^q)$ follows the same distribution with the expectation $\mu = E[f_q(\hat{R}_t^q)]$ and the variance as $Var[f_q(\hat{R}_t^q)] = \frac{1}{n}\sum_{i=1}^{n}(f_q(R_i^q) - \mu)^2$. Then, the variance of the estimator $\hat{N}_q^{RIS}$ (Equation (1)) is computed as

$$Var[\hat{N}_q^{RIS}] = Var\left[\frac{n}{B}\sum_{t=1}^{B} f_q(\hat{R}_t^q)\right] = \frac{n^2}{B}\left(\frac{S}{n} - (\sum_{j=1}^{M_q} \frac{n'_j}{n}\mu_{q'_j})^2\right),$$

where $S = \sum_{i=1}^{n} f_q^2(R_i^q)$ and $\mu_{q'_j} = E[f_q(\hat{R}_t^{q'_j})]$. On the other hand, the variance of $\hat{N}_q^{CRIS}$ (Equation (7)) can be written as

$$Var[\hat{N}_q^{CRIS}] = \frac{n^2}{B^2}\sum_{j=1}^{M_q} B'_j Var[f_q(\hat{R}_t^{q'_j})] = \frac{n^2}{B}\left(\frac{S}{n} - \sum_{j=1}^{M_q}\frac{n'_j}{n}\mu_{q'_j}^2\right).$$

From the above two variances, we obtain

$$\begin{aligned} Var[\hat{N}_q^{RIS}] - Var[\hat{N}_q^{CRIS}] &= \frac{n^2}{B}\left((\sum_{j=1}^{M_q}\frac{n'_j}{n}\mu_{q'_j})^2 - (\sum_{j=1}^{M_q}\frac{n'_j}{n}\mu_{q'_j}^2)\right) \\ &= \frac{n^2}{B} Var[\mu_{q'_j}] \ge 0. \end{aligned}$$

Hence, we proved that CRIS always acheives a lower estimation variance than RIS. □

6. EXPERIMENTAL RESULTS

In this section, we conduct extensive experiments to evaluate our RIS and CRIS algorithms in a large-scale user trajectory dataset from a leading cellular network service provider in China. We first introduce our dataset and configurations of the spatio-temporal database system. Then, we provide comprehensive comparison results between our RIS and CRIS algorithms and **E**xhaustive **S**earch (ES) method. The evaluation results demonstrate that RIS and CRIS algorithms can reduce the query processing time by two orders of magnitude, while preserving a relative error ratio lower than 10%, with only 1% search cost of ES method.

6.1 Data and System Descriptions

The trajectory dataset is a collection of mobile broadband (MBB) data in a large city in eastern China, with an urban area of about 400 square miles and three million people. The dataset was collected for eight days at the end of 2010, and it represents trajectories of 109,914 3G users. When a user's 3G service is enabled, the 3G device periodically reports its location by probing the nearby base stations. Moreover, once a user is using 3G data service, e.g., browsing websites or playing online games, the data exchanged with 3G network are also recorded [3]. We consider each user as a single trajectory. Each record in our database is represented as a spatio-temporal point of a user, where in total more than 400 million $(407, 040, 083)$ records were obtained. Each record has four core attributes including *trajectory ID*, *longitude*, *latitude* and *time*. For each user (trajectory), there are eighty-two attributes associated, including the number of spatio-temporal points, time duration (seconds), spatial range (in square miles), activity frequency (number of points per hour), trajectory length (in miles), average speed, total data usage (in Bytes), etc. Figure 4 shows the trajectory distributions over four attributes. In the experiments, we consider a trajectory aggregate query returning the number of distinct trajectories in a spatio-temporal query range, with certain attribute constraints. The dataset takes 3 TB storage space in the database system.

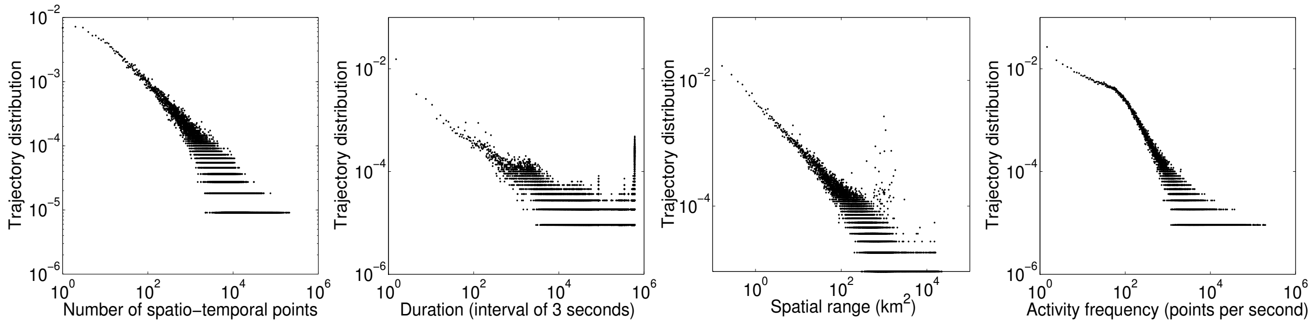
The underlying spatio-temporal data storage system is based on *Clost* [29] and the query system is built on top of Spark [35], a MapReduce-like in-memory cluster computing system. The system runs on a cluster with three machines. Each machine has 24 Intel X5670 2.93GHz processors and 94GB memory. All machines run on Suse Linux Enterprise Server 11. Hadoop 0.20.2-cdh3u6 and Spark-0.7.3 are selected as the running platform.

6.2 Evaluation Results

We perform two sets of experiments for single and concurrent trajectory aggregate queries to evaluate the efficiency of RIS and CRIS algorithms, respectively.

6.2.1 Single Trajectory Aggregate Query

[3]Note that in a 3G network, a user can be located without users' GPS service enabled, since 3G network can locate a user by the signal strength between the device and the nearby base stations.



(a) Spatio-temporal points (b) Time duration (c) Spatial range (d) Activity frequency

Figure 4: Trajectory distributions over different attributes

In the experiments, we set the side length of the trajectory aggregate queries as 18 miles, and temporal range as all eight days, and uniformly at random generate 500 trajectory aggregate queries with the same query range size. We use Quad-tree and B-tree to index the trajectory data with different granularities, i.e., dividing the data into $n = 6,889$ (7k), $n = 13,410$ (13k), $n = 23,481$ (23k) index leaf nodes, respectively. For each granularity, we perform ES and RIS to estimate the total number of distinct trajectories, where we change the sampling budget ratio $a = B/n$ from 0.1% to 25.6%, i.e., the ratio between the number of sampled index leaf nodes and the total number of leaf nodes. Given a randomly generated query, we perform RIS sampling and estimation 200 times for each sampling budget ratio.

To evaluate the estimation accuracy of random index sampling, Figure 5(a) shows the normalized mean square error (NMSE) between our estimation results using RIS and the ground truth. NMSE [26] is a normalized measure of the dispersion of the estimates, defined as

$$NMSE(\hat{N}) = \frac{\sqrt{E[(\hat{N} - N)^2]}}{N}. \tag{8}$$

From Equation (8), an estimation with NMSE larger than 1 is not acceptable. From Figure 5(a), we observe that as the sampling budget increases, the average NMSE decreases quickly. For the same sampling budget ratio, smaller index leaf nodes (i.e., with larger n) lead to smaller NMSE, because smaller index leaf nodes lead to smaller variances of the number of distinct trajectories among index leaf nodes. Define the relative error ratio of an estimation as the normalized difference from the ground-truth, i.e., $b = (\hat{N} - N)/N$. Figure 5(b) shows the box plot of the relative error ratios. For each sampling budget ratio, three boxes are drawn for different granularities. Box plots display differences between populations, where the spacing between the different parts of the box helps indicate the degree of dispersion (spread) and skewness in the data, and identify outliers. The bottom and top of the box are always the first and third quartiles, and the band inside the box is always the second quartile (the median). The lowest datum is still within 1.5 interquartile range (IQR) of the lower quartile, and the highest datum is still within 1.5 IQR of the upper quartile [10]. Any data not included between the whiskers should be plotted as an outlier with a marker "$+$". We observe that the estimation converges to the ground truth as the sampling budget increases. Note that when the sampling budget ratio reaches 1%, the estimation error ratios become lower than 10%. Moreover, the dispersion of the estimated numbers validates the unbiasedness of the designed estimator (Equation (1)).

To evaluate the time efficiency of our RIS algorithm, Figure 5(c) shows the processing time of using ES and RIS. When ES is used, the processing time is more than 110 seconds (with a slight difference for different granularities), where RIS only needs on average 2 to 17 seconds, with two orders of magnitude time reduction. ***The total time reduction ratio is defined as a ratio of the reduced processing time to ES's processing time***. Figure 5(d) shows the total time reduction ratio of RIS, which indicates the power of RIS, namely, a reduction ratio from 85% to 98%.

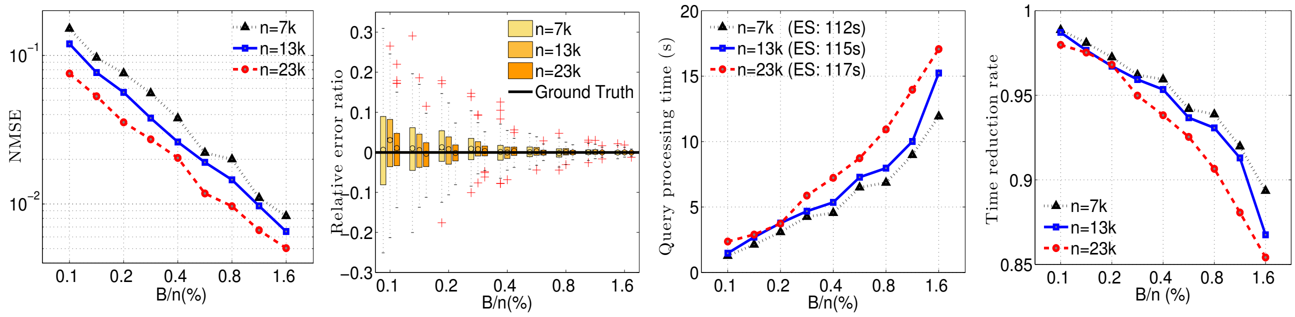
6.3 Concurrent Trajectory Aggregate Queries

To evaluate the performance of CRIS, we randomly generate concurrent queries and compare the time-efficiency and estimation accuracy with RIS and ES algorithms. Our results demonstrate that CRIS improves the performance significantly over that of ES and running RIS independently.

We generate three concurrent queries $\{q_1, q_2, q_3\}$ for 500 times with the following configuration. The spatial side lengths of q_1, q_2, and q_3 are 18 miles, 15 miles, and 12 miles, respectively. In time dimension, they all span for four days, i.e., from day 1 to day 4 for q_1, from day 3 to day 6 for q_2, and from day 5 to day 8 for q_3. We only consider a single index granularity as $n = 23,481$ (23k) index leaf nodes using Quad-tree and B-tree. To answer q_1, q_2, and q_3, we perform ES, RIS, and CRIS, with the sampling budget ratio $a = B/N$ varying from 0.1% to 25.6%, and compare their performance, in terms of the estimation accuracy, processing time, and time reduction ratio. Again, given three randomly generated concurrent queries, we run 200 times for RIS and CRIS for each sampling budget ratio.

Figure 6(a) presents the average normalized mean square error (NMSE) of our estimations using RIS and CRIS. CRIS indicates lower NMSE values for all concurrent queries q_1, q_2, and q_3. Moreover, as the sampling budget ratio increases, NMSE decreases for both RIS and CRIS. The box plots in Figure 6(b) show the distributions of the relative error ratios for q_1, q_2, and q_3. Clearly, the estimations by both RIS and CRIS algorithms converge to the ground-truth when the sampling budget ratio increases. In addition, given the same sampling budget ratio, CRIS has higher estimation accuracy than that of RIS. When the sampling budget ratio reaches 1%, the estimation error ratios are within 10%.

Moreover, Table 2 provides the numbers of sampled index leaf nodes collected by RIS and CRIS for different sampling budget ratios. We observe that CRIS requires much smaller number of samples than RIS by applying the overlapping sample reuse. Thus, the query processing time of CRIS is much smaller than that of RIS. As shown in Figure 6(c), comparing to ES (taking 226 seconds), RIS needs 5 to 26 seconds and CRIS only needs 4 to 17 seconds on average. Hence, CRIS reduces the processing time over RIS



(a) NMSE (b) Relative error (c) Processing time (d) Time reduction rate

Figure 5: Performance comparison between RIS and exhaustive search (ES) algorithm

Table 2: Numbers of sampled indices required by RIS vs CRIS.

Sample Rate (%)	0.1	0.2	0.4	0.8	1.6	3.2	6.4	12.8	25.6
Samples (RIS)	35	69	139	278	555	1,111	2,221	4,443	8,885
Samples (CRIS)	21	42	84	167	334	669	1,337	2,675	5,350

by 20% to 34.6%. When considering the time reduction rate, Figure 6(d) shows that RIS and CRIS reduce the processing time over exhaustive search by 88% to 97% and 92% to 98%, respectively.

7. DISCUSSIONS AND EXTENSIONS

Throughout this paper, we take counting distinct trajectories in a spatio-temporal region as an example of trajectory aggregate queries. Our RIS and CRIS algorithms are in fact generic to other trajectory aggregate queries, such as sum and average. We briefly discuss how our sampling and estimation algorithms can be applied to these queries by adjusting the mapping function on an index leaf node. Due to the limited space, we introduce the unbiased estimators and omit the detailed proof and convergence analysis.

Sum. A typical sum aggregation query is "the total length of all trajectories with speed greater than 5 miles per hour in New York City during 2013". Let ℓ_r denote the length (in miles) of a trajectory r. Given an index leaf node R_i^q with $1 \le i \le n$, define $h_q(R_i^q) = \sum_{r \in R_i^q} \ell_r / k_r^q$. Then, the exact total length of trajectories is $\ell_q = \sum_{i=1}^{n} h_q(R_i^q)$. By following similar proof in Theorem 2, it is easy to prove that $\hat{\ell}_q$ in Equation (9) is the asymptotically unbiased estimator of ℓ_q.

$$\hat{\ell}_q = \frac{n}{B} \sum_{t=1}^{B} h_q(\hat{R}_t^q). \tag{9}$$

Average. An average query is "the average trajectory length of all trajectories with speed greater than 5 miles per hour in New York City during 2013". $L_q = \ell_q / N_q$ is thus the exact answer, where N_q is the number of distinct trajectories with speed greater than 5 miles per hour in New York City during 2013. The asymptotically unbiased estimator $\hat{L}_q$ is presented in Equation (10), which can be proven by the ratio form of the law of large numbers (Theorem 17.2.1 on p.426 in [22]).

$$\hat{L}_q = \frac{\sum_{t=1}^{B} h_q(\hat{R}_t^q)}{\sum_{t'=1}^{B} f_q(\hat{R}_{t'}^q)}. \tag{10}$$

8. CONCLUSION

Large-scale trajectory data create challenges in processing trajectory aggregate queries with stringent response-time constraints. Exhaustively brute-forcing the query space to get an exact answer is usually too time-consuming. Given a well-indexed trajectory database, in this paper, we develop random index sampling (RIS) algorithm that randomly samples a small number of index leaf nodes and the associated trajectories to estimate the answer to the trajectory aggregate query, with guaranteed estimation error bounds. Moreover, for concurrent trajectory aggregate queries with overlapping spatio-temporal query regions, we design concurrent random index sampling (CRIS) algorithm using stratified sampling and overlapping sample reuse that achieves higher estimation accuracy with less sampling budgets than that of using RIS algorithm independently. We evaluated our RIS and CRIS algorithms using a large-scale user trajectory dataset (with 3TB data) collected from a cellular service provider in China. Our extensive evaluation results show that RIS and CRIS algorithms outperform the exhaustive search algorithm for single and concurrent trajectory aggregate queries by two orders of magnitude in terms of processing time, while preserving a relative error ratio lower than 10%, with only 1% search cost of the exhaustive search algorithm.

9. ACKNOWLEDGEMENT

Yanhua Li and Zhi-Li Zhang were supported in part by NSF grants CNS-1117536, CRI-1305237, CNS-1411636, DoD DTRA grant HDTRA1-14-1-0040 and ARO MURI Award W911NF-12-1-0385. Jia Zeng was supported in part by NSFC grants No. 61373092 and 61033013, Natural Science Foundation of the Jiangsu Higher Education Institutions of China No. 12KJA520004, and Collaborative Innovation Center of Novel Software Technology and Industrialization.

10. REFERENCES

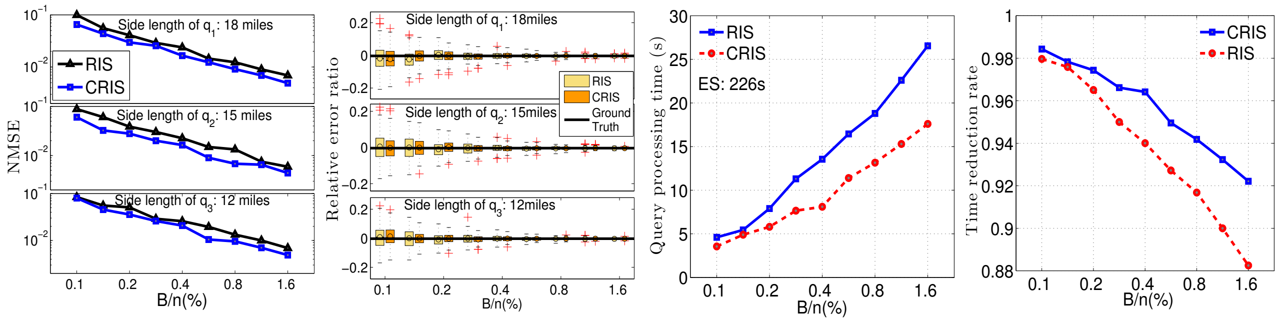
[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[2] K. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla. On synopses for distinct-value estimation under multiset operations. In *ACM SIGMOD*, 2007.

[3] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[4] Y. Cai, K. A. Hua, G. Cao, and T. Xu. Real-time processing of range-monitoring queries in heterogeneous mobile databases. *IEEE TMC*, 5(7):931–942, 2006.

[5] V. P. Chakka, A. Everspaugh, and J. M. Patel. Indexing Large Trajectory Data Sets With SETI. In *CIDR*, 2003.



(a) NMSE (b) Relative error (c) Processing time (d) Time reduction rate

Figure 6: Performance comparison between CRIS, RIS and exhaustive search (ES) algorithms for continuous queries

[6] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *ACM PODS*, 2000.

[7] S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: How much is enough? In *ACM SIGMOD*, 1998.

[8] K. Deng, K. Xie, K. Zheng, and X. Zhou. Trajectory indexing and retrieval. In *Computing with Spatial Trajectories*, pages 35–60. 2011.

[9] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[10] M. Frigge, D. C. Hoaglin, and B. Iglewicz. Some implementations of the boxplot. *The American Statistician*, 43(1):50–54, 1989.

[11] V. Gaede and O. Günther. Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[12] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB*, 2001.

[13] R. H. Güting, T. Behr, and J. Xu. Efficient k-nearest neighbor search on moving object trajectories. *The VLDB Journal*, 19(5):687–714, 2010.

[14] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *VLDB*, 1995.

[15] Y. Huang, F. Zhu, M. Yuan, K. Deng, Y. Li, B. Ni, W. Dai, Q. Yang, and J. Zeng. Telco churn prediction with big data. In *ACM SIGMOD*, 2015.

[16] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. Discovery of convoys in trajectory databases. In *VLDB*, 2008.

[17] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *ACM PODS*, 2010.

[18] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. In *ACM SIGMOD*, 2007.

[19] Y. Li, J. Luo, C.-Y. Chow, K.-L. Chan, Y. Ding, and F. Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *IEEE ICDE*, 2015.

[20] Y. Li, M. Steiner, J. Bao, L. Wang, and T. Zhu. Region sampling and estimation of geosocial data with dynamic range calibration. In *IEEE ICDE*, 2014.

[21] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. In *VLDB*, 2010.

[22] S. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

[23] F. Olken. *Random sampling from databases*. PhD thesis, University of California, 1993.

[24] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches to the indexing of moving object trajectories. In *VLDB*, 2000.

[25] I. Psaroudakis, M. Athanassoulis, and A. Ailamaki. Sharing data and work across concurrent analytical queries. In *VLDB*, 2013.

[26] B. Ribeiro and D. Towsley. Estimating and sampling graphs with multidimensional random walks. In *ACM IMC*, 2010.

[27] T. K. Sellis. Multiple-query optimization. *ACM TODS*, 13(1):23–52, 1988.

[28] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[29] H. Tan, W. Luo, and L. M. Ni. Clost: a hadoop-based storage system for big spatio-temporal data analytics. In *ACM CIKM*, 2012.

[30] Y. Tao and D. Papadias. Efficient historical r-trees. In *SSDBM*, 2001.

[31] Y. Tao and D. Papadias. MV3R-Tree: A spatio-temporal access method for timestamp and interval queries. In *VLDB*, 2001.

[32] C. Xu, Y. Gu, L. Chen, J. Qiao, and G. Yu. Interval reverse nearest neighbor queries on uncertain data with Markov correlations. In *IEEE ICDE*, 2013.

[33] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger? In *ACM UbiComp*, 2011.

[34] M. Yuan, K. Deng, J. Zeng, Y. Li, B. Ni, X. He, F. Wang, W. Dai, and Q. Yang. OceanST: A distributed analytic system for large-scale spatiotemporal mobile broadband data. In *VLDB Demo*, 2014.

[35] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. In *HotCloud*, 2010.

[36] Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *ACM UbiComp*, 2011.

[37] P. Zhou, D. Zhang, B. Salzberg, G. Cooperman, and G. Kollios. Close pair queries in moving object databases. In *ACM SIGSPATIAL GIS*, 2005.

[38] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM TODS*, 23(4):453–490, 1998.