Mining Spatio-Temporal Reachable Regions With Multiple Sources over Massive Trajectory Data

Yichen Ding, Xun Zhou, Guojun Wu, Yanhua Li, Jie Bao, Yu Zheng, and Jun Luo

Abstract—Given a set of user-specified locations and a massive trajectory dataset, the task of mining spatio-temporal reachable regions aims at finding which road segments are reachable from these locations within a given temporal period based on the historical trajectories. Determining such spatio-temporal reachable regions with high accuracy is vital for many urban applications, such as location-based recommendations and advertising. Traditional approaches to answering such queries essentially perform a distance-based range query over the given road network, which does not consider dynamic travel time at different time of day. By contrast, we propose a data-driven approach to formulate the problem as mining actual reachable regions based on a real historical trajectory dataset. Efficient algorithms for the Single-location spatio-temporal reachability Query (S-Query) and the Union-of-multi-location spatio-temporal reachability Query (U-Query) were presented in our recent work. In this paper, we extend the previous ideas by introducing a new type of reachability query with multiple sources, namely, the Intersection-of-multi-location spatio-temporal reachability Query (I-Query). As we demonstrate, answering I-Queries efficiently is generally more computationally challenging than answering either S-Queries or U-Queries because I-Queries involve complicated intersect conditions. We propose two new algorithms called the Intersection-of-Multi-location Query Maximum Bounding region search (I-MQMB) algorithm and the I-Query Trace Back Search (I-TBS) algorithm to efficiently answer I-Queries, which utilize an indexing schema composed of a spatio-temporal index and a connection index. We evaluate our system extensively by using a large-scale real taxi trajectory dataset that records taxi rides in Shenzhen, China. Our results demonstrate that the proposed approach reduces the running time of I-Queries by 50 percent on average compared to the baseline method.

Index Terms—Spatio-temporal databases, trajectory query processing, reachability queries

1 Introduction

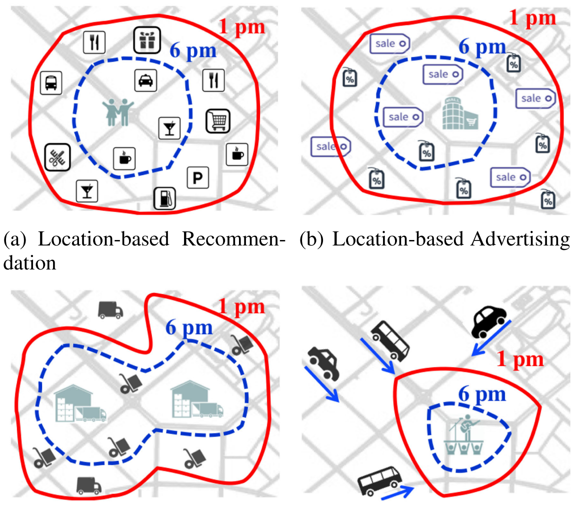
A spatio-temporal reachability query aims to find the reachable area in a spatial network from a location or a set of locations in a given time period. As illustrated in Fig. 1, finding the spatio-temporal reachable region is very useful in many urban applications, including: (1) location-based recommendation, where a user wants to find a nearby restaurant based on her current location and time; (2) location-based advertising, where a business owner identifies potential spatial regions to arrange special activities, such as distributing coupons and sales discounts; (3) business coverage analysis, where chain stores such as UPS and FedEx can find the overall business spatial coverage of their branches; and (4) business location selection, where some stores use this technique to identify convenient locations reachable from major transportation hubs to open new brunches. In Fig. 1, the first two application examples illustrate the Single-location spatio-temporal reachability Query (S-Query, with only one query location as input). The third and fourth applications are used to illustrate the two extended multi-location queries, namely, the Union-of-multi-location spatio-temporal reachability Query (U-Query) and the Intersection-of-multi-location spatio-temporal reachability Query (I-Query). In this paper, we mainly focus on the I-Query.

The traditional reachability query [1], [2], [3] on the road network has several drawbacks to fulfill the aforementioned urban applications. Most of the existing work, for instance, focuses on a reachable range based on spatial network distance rather than the time period. However, in real application scenarios, users care more about the actual travel time rather than distance. Also, most of the existing works do not account for the impact of time of query on the results. In reality, due to different traffic conditions in rush hours versus in normal hours, the reachable area may vary significantly over different time of the day. The traditional spatial-network-based approaches cannot capture such differences.

To improve the usability of the reachability query in real application scenarios, we propose a data-driven approach to find the spatio-temporal reachable regions based on massive

- Y. Ding and X. Zhou are with the Department of Business Analytics, University of Iowa, Iowa City, IA 52242 USA. E-mail: {yichen-ding, xun-zhou}@uiowa.edu.
- G. Wu and Y. Li are with the Worcester Polytechnic Institute, Worcester, MA 01609 USA. E-mail: {gwu, yli15}@wpi.edu.
- J. Bao is with Urban Computing Business Unit, JD Finance, Beijing 101111, China. E-mail: baojie@jd.com.
- Y. Zheng is with JD Intelligent Cities Research, JD Intelligent Cities Business Unit, JD Digits, Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu, Sichuan 611756, China, and also with Xidian University, Xi'an, Shaanxi 710126, China. E-mail: zheng.yu@jd.com.
- J. Luo is with Lenovo Machine Intelligence Center, Hong Kong. E-mail: jluo1@lenovo.com.

Manuscript received 7 Oct. 2018; revised 4 Nov. 2019; accepted 24 Nov. 2019. Date of publication 18 Dec. 2019; date of current version 3 June 2021. (Corresponding author: Xun Zhou.) Recommended for acceptance by B. Glavic. Digital Object Identifier no. 10.1109/TKDE.2019.2959531



(a) Location-based Recommendation (b) Location-based Advertising

(c) Business Coverage Analysis (d) Business Location Selection

Fig. 1. *Application Examples.* These four simple examples offer an intuitive understanding of how spatio-temporal reachability queries can be applied in daily life.

real trajectory data collected over the road network. The impetus behind our approach is to formulate the spatio-temporal reachability query as a data mining process, which finds all the trajectories that passed the query locations and aggregates all their destinations within the given time period. This way, the reachable area is more realistic, as it is essentially a summary from real-life data.

In our prior work [4], we proposed an efficient computational solution framework to address the reachability query problem. We first developed a Spatio-Temporal Index (ST-Index) and a Connection Index (Con-Index) to support the reachability queries. Then we proposed a Single-location Query Maximum Bounding region search (SQMB) algorithm to answer a simple spatio-temporal reachability query with only one query location (S-Query). Using the solutions to the S-Query as a building block, we further proposed an Multi-location Query Maximum Bounding region search (MQMB) algorithm to process the multi-location reachability query, which is the union of multiple S-Queries' results (referred to as the U-Query in this paper). The proposed algorithms significantly reduced the running time of straightforward S-Query and U-Query processing algorithms on real taxi trajectory data.

In this paper, we further extend our proposed framework to answer the I-Query. Given a set of n query locations, the I-Query finds a subset of the road network, where each output road segment is reachable from *all* n query locations. Answering the I-Query is important to a number of urban applications such as business location selection and hotspot detection.

Efficiently answering I-Queries is more challenging than answering S-Queries and U-Queries. First of all, the I-Query is a conjunctive query, where the intersection of input locations could be an empty set. Therefore, in many scenarios, it is impossible to apply the expansion idea we previously proposed for the S-Query and the U-Query. Second, unlike in a U-Query, where all the input location segments are treated equally, in an I-Query the order in which the query locations are handled will make a significant difference in the processing time, since some of them will intersect early and some will never intersect.

To solve this problem, we propose a novel Intersection-of-Multi-location Query Maximum Bounding region search (I-MQMB) algorithm. The algorithm evaluates each individual query location and efficiently keeps track of the intersection of subgroups of these input locations. In particular, the algorithm can handle complex cases when subsets of the reachable regions of the input query locations intersect at different time steps. We also design an I-Query Trace Back Search (I-TBS) algorithm to refine the results.

The *new* contributions of this paper are summarized as follows.

- We formulate the Intersection-of-multi-location spatio-temporal reachability query (I-Query) and propose an Intersection-of-Multi-location Query Maximum Bounding region search algorithm to efficiently answer the I-Query.
- We conduct extensive experiments on a real-world road network with a large-scale moving-object trajectory dataset (with 194 GB size) collected from a metropolis in China to evaluate the efficiency and effectiveness of our proposed query processing algorithms. Experimental results show that our I-Query solution algorithms outperform the straightforward query processing method by reducing 50-70 percent of the query processing time.

The remainder of the paper is organized as follows. Section 2 defines our problem and outlines our system framework. Section 3 presents our data preprocessing and index construction steps. Section 4 summarizes our S-query and U-query processing algorithms *from a previous work*. Section 5 introduces the new I-MQMB and the I-TBS algorithms to efficiently answer I-Queries. Section 6 analyzes the time complexity for all 4 presented algorithms. Section 7 presents the evaluation results based on large scale real trajectory data. Section 8 discusses the related work. Section 9 discusses the individual privacy of trajectory data and concludes the paper with future work.

2 Overview

In this section, we first clarify key terms used in this paper and provide a formal definition of *spatio-temporal reachability queries* with single and multiple query locations. Finally, we give an overview of the system framework.

2.1 Basic Concepts

- *Road Network.* A road network can be viewed as a directed graph $G(V, E)$, where E is a set of edges, and V is a set of vertices representing the intersections on the road network. Each road segment has a unique ID, a list of connected road segments in the network, a list of intermediate points (2 terminal points at the beginning and the end) that describe its shape, a value of its length, an indicator of direction (i.e., one-way or two-way), a type value that identifies its level (primary or secondary) and a MBR (Minimum Bounding Rectangles) that describes its spatial coverage.
- *Trajectory.* A trajectory is a sequence of spatio-temporal points. Each point consists of a trajectory ID, spatial information (e.g., latitude, longitude), a timestamp, and a collection of properties (e.g., travel speed,

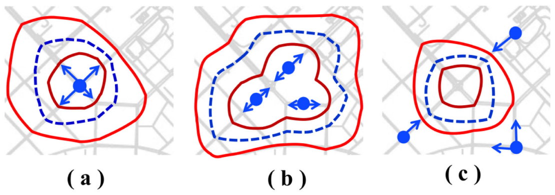


Fig. 2. *Spatio-Temporal (ST) Reachability Query.* An ST Reachability Query q returns road segments within dash line area. The point(s) is (are) the start location(s) specified by user and the solid circles indicates the bounding region of the query q. (a) a Single-location ST reachability Query with only one start location. (b) a Union-of-multi-location ST reachability Query with 3 start locations. (c) an Intersection-of-multi-location ST reachability Query with 3 start locations.

direction, or occupancy indicating if the taxi has a passenger or not).

- *Trajectory Reachability.* Given a start location S, a road segment r_i in the road network, a start time T, and a travel duration L, the trajectory reachability refers to whether any historical trajectories has traversed the road segment r_i from the start location S within the given duration (i.e., from T to T+L). If the road segment r_i is reachable from S, the trajectory reachability is 1, otherwise the trajectory reachability is 0. For example, given the above constrains, if there was a trajectory passing S at t_1 and then passing r_i at t_2 and $t_2 - t_1 < L$ holds, the trajectory reachability for road segment r_i is 1.
- *Reachable Area.* Given a start location S, a start time T and a travel duration L, a reachable area is the area covered by a set of reachable road segments from S.
- *Prob-Reachable Area.* The *Prob*-reachable area is a more general description of the reachable area, where we introduce a reachable probability that describes the percentage of days in the historical trajectory dataset that support the fact that a road segment r_i is reachable from S within the given duration. For example, if there are 20 out of 100 days in the dataset with moving objects starting from location S and traversing the reachable area within $[T, T + L]$, the probability of this reachable area is 20 percent.

2.2 Problem Definition

Single-Location Spatio-Temporal Reachability Query. Given a road network graph $G(V, E)$, where E is a set of road segments and V is a set of intersections, a query location S, a start time T, a travel duration L, a probability ratio *Prob* and a trajectory database *TR*, we want to find a set of road segments as the *Prob*-reachable area in the road network G, where the road segments in the set all have at least *Prob* chance in the trajectory database to be reached from the start location S in a given duration. The objective of our system is to minimize the overall system overhead in finding the *Prob*-reachable region based on the user's query parameters. We call this type of query "S-Query".

Multi-Location Spatio-Temporal Reachability Queries.

(1) Union-of-multi-location ST reachability Query (U-Query): Given a set of n start locations, and all the other input parameters for the S-Query, the U-Query finds a set of roads, that are reachable with respect to the query parameters from *at least one of the* n *query locations*.

(2) Intersection-of-multi-location ST reachability Query (I-Query): Given a set of n start locations, and all the other input parameters for the S-Query, the I-Query finds a set of roads, that are reachable with respect to the query parameters from *all of the* n *query locations*.

Fig. 2 illustrates examples of the three types of queries: (a) S-query, (b) U-Query, and (c) I-Query.

Table 1 provides a summary of the notations and terminologies that are frequently used in this paper.

TABLE 1
Notations and Terminologies

Notation	Description
S	S is the spatial information of a location including longitude and latitude from query q.
T	T is a time value indicating the temporal information from query q.
L	L is the prediction time length of travel duration from query q.
Prob	*Prob* is the probability of a reachable area for answering query q.
B	B is a set of road segments indicating the maximum/minimum bounding region of a query q.
r_i	r_i is the ith road segment in the road network.
n	n is the total number of road segments.
$N(r_i, t)$	$N(r_i, t)$ is the Near ID list of road segment r_i in a connection table in time slot t.
$F(r_i, t)$	$F(r_i, t)$ is the Far ID list of road segment r_i in a connection table in time slot t.
Tr_i	Tr_i is the ith trajectory in the trajectory history.
m	m is the total number of trajectory days.

2.3 System Overview

Fig. 3 gives an overview of our proposed system, which consists of three main components: *Pre-processing*, *Index Construction*, and *Query Processing*.

- *Data Pre-processing.* This component performs two main tasks: (1) road re-segmentation and (2) trajectory map-matching. This step breaks the original road segments in the road network into smaller sections (e.g., 500 m) to improve the precision of the reachability queries, and maps the raw trajectories onto the new road network.
- *Index Construction.* This component builds two indexing structures to speed up the later query processing: (1) spatio-temporal index and (2) connection index. The spatio-temporal index partitions trajectories based on space and time. On the other hand, the connection index links road segments based on the historical trajectory information, which records a lower bound range as *Near Table* and upper bound range as *Far Table*, i.e., noted as N and F in Fig. 3. The connection index is used to refine the spatio-temporal reachability query process.
- *Query Processing.* This component processes queries from the user. It employs two main techniques: (1) S-Query maximum bounding region search using our spatio-temporal index and connection index to

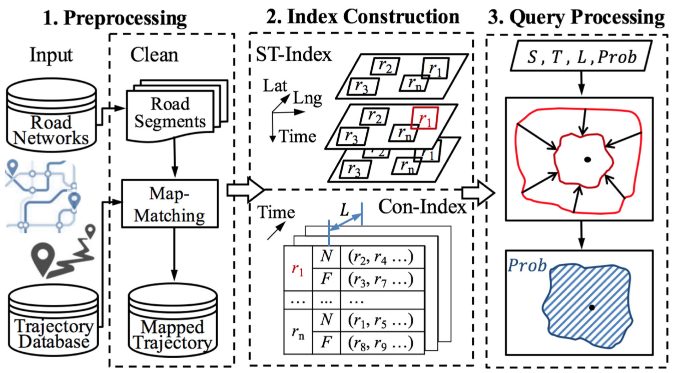


Fig. 3. *An overview of framework.* Take a Single-location ST reachability Query q with S={r_1} as an example, we first find road segment r_1 at start timestamp T by ST-Index and then jump to other road segments according to Con-Index within travel duration L. Finally, we trace back search from maximum bounding region to minimum bounding region until road segments satisfy *Prob* requirement.

generate a rough estimation of the upper bound of *Prob*-reachable region based on the query parameters; and (2) trace back search, which uses the connection index and the original road network to refine the region from the first step. We use these two algorithms to answer the U-Query and the I-Query as well.

3 Pre-Processing and Index Construction

In this section, we introduce the pre-processing steps and the details of our two indexing structures: (1) a Spatio-Temporal Index (ST-Index) and (2) a Connection Index (Con-Index).

3.1 Data Pre-Processing

Road Re-Segmentation. To generate results with finer granularity, we process the road network data to break long road segments into shorter units based on a given length limit (e.g., 500 m). New intersections are added as needed between new segments.

Map-Matching. We then map the raw trajectory data onto the newly segmented road network. We employ an existing method [5] to perform the task. Note that we connect all the trips of the same moving object in one day into a single trajectory. That is to say, a moving object only has one trajectory per day in our dataset.

3.2 Index Construction

3.2.1 Spatio-Temporal Index

The ST-Index is used to speed up the process to find out the corresponding start road segment based on the query location. The main variation in our spatio-temporal index is having two levels of temporal information embedded (i.e., time of the day and date) in order to calculate the *Prob*-reachable area more efficiently. Therefore, the ST-Index consists of 3 components: *Temporal Index*, *Spatial Index* and *Time List*. We can use a B-tree as the temporal index to find out the start time slot and use an R-tree as the spatial index to match the start road segment for a query. For the illustration example and detailed information, please refer to the supplementary materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2019.2959531.

We compare our indexing structures with state-of-the-art ones. A wide variety of indexing structures have been developed but not all of them are applicable to our spatio-temporal reachability queries (e.g., TRIFL [6] designed for flash storage and U-tree [7] considering uncertain objects obeying a probability density function over multidimensional points). Other well-known ST index methods such as the Scalable and Efficient Trajectory Index (SETI) [8], the Practical Index for Spatio-Temporal (PIST) [9], and the PARtitionned Index for in-NEtwork Trajectories (PARINET) [10] also separate the spatial and temporal components of data and index the moving objects' information. With these ST index methods, users can index the exact position in a specific road segment at a certain time for a given trajectory. It is unnecessary to use these advanced indexing structures under our current problem settings. However, users can adopt any applicable indexing structure to meet their own needs.

3.2.2 Connection Index

With the spatio-temporal index built as above, a naive solution to answer the spatio-temporal reachability query can be proposed as follows. We use the traditional network expansion algorithm, e.g., [3] to expand the road network from the query location and verify each expanded road segments to see if it fulfills the reachability probability by reading the trajectory IDs from the disk. However, this query process can be prohibitively inefficient, as it has to access the disk very frequently to retrieve the trajectory information.

To improve system efficiency and avoid unnecessary disk accesses, we propose a connection index to skip some network expansion steps. The basic idea is to use the historical trajectory data to build a connection table for each road segment and record the lower and upper bound of its reachable road segments based on our temporal granularity. In particular, each road segment with different temporal granularity is associated with: (1) a Near ID list (lower bound range) and (2) a Far ID list (upper bound range) indicating the nearest (farthest) road segments that could be arrived at within the given time slot. Note that our Con-Index can reflect multi-lane speed pattern [11] by connecting road segments across different time slots with Near ID list and Far ID list, since these two ID lists indicate traffic speed variations among the corresponding road segments.

To build the connection table, we modify the conventional network expansion algorithm [3]. We generate the Near ID list of each road segment by considering the minimum speed (removing the zero speed) in all directions, after which we expand the road network using the networking expansion algorithm [3] with the temporal granularity. After that, all the reachable road segments in this process are added to the table as the Near ID list of the start road segment. The Far ID list is constructed in a similar way by using the maximum traveling speed calculated from the historical trajectories. Fig. 4 illustrates a connection table in an arbitrary time slot of Con-Index. Take road segment r_1 as an example. Road segments (r_2, r_5, r_7, r_9) belong to its Near ID list while road segments $(r_4, r_6, r_8, r_{10}, r_{12}, r_{14}, r_{15})$ as Far ID list. As can be seen, the range of Far ID list is larger and extends to more intersections over the road network.

Connection Table ($\forall t \in T$)

r_1	*Near*	(r_2, r_5, r_7, r_9)
	Far	$(r_4, r_6, r_8, r_{10}, r_{12}, r_{14}, r_{15})$
r_2	*Near*	*ID List*
	Far	*ID List*
…	…	…
r_n	*Near*	*ID List*
	Far	*ID List*

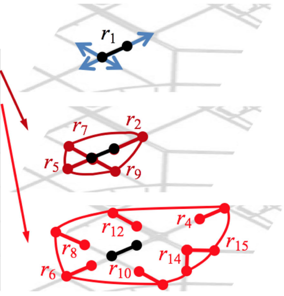


Fig. 4. *Con-Index.* The left table indicates a connection table in time slot t. The right figure depicts the road segments in the Near ID list and Far ID list of road segment r_1 on a real road network.

4 Processing S-Query and U-Query

With the ST-index and the Con-Index, we summarize algorithms to solve the S-Query and the U-Query problems from our prior conference paper [4] to make our paper self-contained.

4.1 Single-Location ST Reachability Query (S-Query)

A single-location ST reachability query, i.e., S-Query $q = (S, T, L, Prob)$, includes one query location specified as $S = \{s\}$, start time T, a travel duration L, and a probability $0 < Prob \leq 1$. The S-Query was solved in our prior conference paper [4]. Since it is not directly related to our main contributions of this paper, please refer to the supplementary materials, available online, and our prior conference paper for details.

4.2 Union-of-Multi-Location ST Reachability Query (U-Query)

A U-Query (previously called the m-query[4]) is formally defined as $q = (S, T, L, Prob)$, with a set of querying n locations $S = \{s_1, \ldots, s_n\}$, start time T, travel duration L, and a confidence probability $Prob$. The U-Query q asks for the $Prob$-reachable region from any of the location $s \in S$ during time interval $[T, T+L]$. We present the solutions to U-Query to make this paper self-contained as we use the solution as part of our new solution for the I-Query.

In theory, if we consider each query location $s_i \in S$ as an S-Query, namely, $q = (s_i, T, L, Prob)$, with a result of $Prob$-reachable region as B_i, the answer of a U-Query is then the outer-most bounding regions of the union among all B_i's. Fig. 5a shows an example of U-Query with two start road segments, r_1 and r_2. The solid lines outline the $Prob$-reachable region of the U-Query, which is the outer-most bounding region of two single $Prob$-reachable regions of r_1 and r_2, where the overlapping parts (in dashed lines) are removed.

The basic idea behind the query processing algorithm for U-Query is a two-step approach: (i) finding a unifying maximum and minimum bounding region of the U-Query by checking ST-Index and Con-Index; (ii) trace back searching the road segments from the maximum to minimum bounding regions to identify the $Prob$-reachable region of U-Query q. As shown in Fig. 5b, the maximum and minimum bounding regions are the outer-most boundary of the merged bounding regions across all single S-Queries. We develop the U-Query maximum bounding region search algorithm, which works as follows.

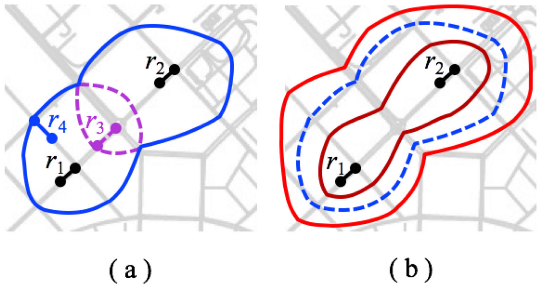


Fig. 5. *U-Query Bounding Regions.* Solid line indicates the outer bounding region which is the real maximum reachable boundary of both r_1 and r_2 while dashed line indicating inner bounding regions. Road segments r_1 and r_2 are the start locations from a U-Query. Road segment r_3 is on the boundary of r_2 while r_4 on r_1. However, r_3 is not on the outer-most bounding regions while r_4 is.

(1) Maximum Bounding Region Search Algorithm. The Multi-location Query Maximum Bounding region search algorithm for U-Query is presented in Algorithm 1. Given the start locations $S = \{s_1, \ldots, s_n\}$ in U-Query q, the algorithm identifies the set of start road segments $R = \{r_1, \ldots, r_n\}$ and starts to expand the maximum bounding regions until the user-specified travel time $T + L$ (Lines 1-2). Then, it simply constructs the union set B of the maximum bounding regions of all road segments in R (Lines 3-4). As shown in Fig. 5a, the overlapping parts (in dashed line) are removed and a unifying maximum bounding region (in solid line) are formed (Lines 5-8).

Algorithm 1. Multi-Location Query Maximum Bounding Region Search (MQMB) Algorithm

```
Input: U-Query $q = \{S = \{s_1, \ldots, s_n\}, T, L, Prob\}$.
Output: Unified Maximum bounding region set $Result$.
 1: Find start road segment list $R$ in ST-Index, Compute $k$ with
    $k\Delta t \leq L < (k+1)\Delta t$
 2: for $k' = 0$ to $k - 1$ do
 3:   for $\forall r$ in $R$ do
 4:     Bounding set $B \leftarrow B \cup F(r, T + k'\Delta t)$.
 5:   for $\forall b$ in $B$ do
 6:     $r_s = \mathbf{argmin}_{r' \in R}\{dis(r', b)\}$
 7:     if $b \in F(r_s)$ then
 8:       $Result = Result \cup \{b\}$
 9:   $R = Result$
10: return $Result$
```

(2) The U-Query Trace Back Search Algorithm (U-TBS). The maximum and minimum bounding regions provide a refined and smaller geographic region to further identify the exact $Prob$-reachable region of a U-Query q. It guarantees that all road segments $Prob$-reachable are between the maximum and minimum bounding regions. Utilizing such bounded information, we develop a trace back search algorithm to search road segments from the maximum bounding region back to the minimum bounding region to find the $Prob$-reachable region, which works as follows. For a *single query road segment* r_0, we take the following steps to refine the results. First, by checking ST-Index, we extract the list of trajectory IDs from the start road segment r_0 in time interval $T_0 = [T, T + \Delta t]$ during each day d, represented as $Tr(r_0, T_0, d)$, with $1 \leq d \leq m$ and m as the total number of

days in the trajectory dataset. The maximum bounding region B includes a list of road segments. For each road segment $r \in B$, we check the ST-Index to extract the list of trajectory IDs starting from the road segment r in time interval $T_B = [T, T+L]$ of each day d, represented as $Tr(r, T_B, d)$. Then, for each day $1 \le d \le m$, we check if r is reachable from r_0 on day d, by checking if there are some common trajectories in both $Tr(r_0, T_0, d)$ and $Tr(r, T_B, d)$. Suppose $Tr(r_0, T_0, d) \cap Tr(r, T_B, d) \neq \emptyset$ holds for m^* out of m days, then the reachable $probability(r, r_0)$ from r_0 to r during the period of $[T, T+L]$ is as follows, which represents from historical statistics, the probability that road segment r is reachable from r_0 during time interval $[T, T+L]$.

$$probability(r, r_0) = \frac{m^*}{m} \times 100\%. \quad (1)$$

For a given $r \in B$, if $probability(r, r_0) \ge Prob$, the road segment r is close enough to the start road segment r_0 that is reachable with a higher probability than $Prob$. r will be included in the $Prob$-reachable region set. Otherwise, if $probability(r, r_0) < Prob$, it means that r does not have large enough probability to be reached from r_0, therefore, we add r's neighboring road segment set $neighbor(r)$ to the search space B for further investigation. Since we search from the maximum bounding region to the minimum bounding region, the neighboring road segments of r added are always closer than r to the start road segment r_0. The process terminates when $B = \emptyset$ or all the road segments between maximum and minimum bounding regions are searched.

For a set of *multiple query road segments*, we modified the above algorithm when calculating the probability of each candidate road segment between the two bounding regions. For the U-Query, if a candidate road segment r can be reached from *any* query road segment r_s, the candidate is added to the final result.

Algorithm 2. U-Query Trace Back Search (U-TBS) Algorithm

Input: Bounding set B_{max} and B_{min}, probability *Prob*, start road segments R.
Output: Bounding set B' with respect to *Prob*

```
1: $B \leftarrow B_{max}$, $B' = \emptyset$
2: **while** $B \neq \emptyset$ **do**
3:   $r \leftarrow dequeue(B)$
4:   *found* = false
5:   **for each** $r_s$ in $R$ **do**
6:     **if** $probability(r, r_s) \ge Prob$ **then**
7:       $B' \leftarrow \{r\} \cup B'$
8:       *found* = true
9:       Break
10:  **if** *found* == false **then**
11:    $B \leftarrow (neighbor(r) - B_{min}) \cup B$
12: **return** $B'$
```

Note that we take the probability calculation in Equation (1) as an example. Since our trajectory data was collected from one month, if there are some sparse areas with no trajectories, then the probability is considered 0 since nobody travels to this area based on historical data. This makes sense in our motivating examples. In fact, our U-TBS algorithm is flexible to other probability definitions. Users can define their own probabilities according to their own needs.

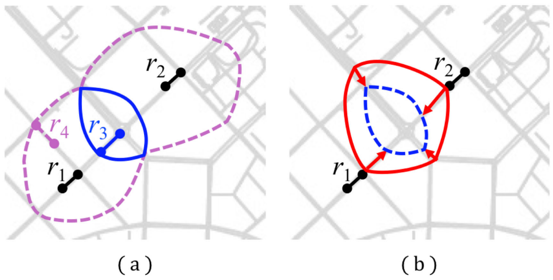


Fig. 6. *I-Query Bounding Regions.* In (a), the solid line indicates the inner bounding region which is the real intersecting reachable boundary of both r_1 and r_2 while dashed line indicating non-intersecting outer bounding regions. Road segments r_1 and r_2 are the start locations from an I-Query. Road segment r_3 is on the boundary of r_2 while r_4 on r_1. However, r_4 is not on the inner bounding regions while r_3 is. In (b), the solid lines pointing to the dashed lines indicate the trace back search from maximum to minimum bounding regions of the I-Query.

The detailed U-Query Trace Back Search (U-TBS) algorithm is summarized in Algorithm 2. First, bounding set B' is initialized and the searching road segment is set as the maximum bounding region B_{max} (Line 1). Then, the algorithm checks if there are still road segments to be searched and examines if r is *Prob*-reachable from any road segment in R. If yes, r is added to *Prob*-reachable set, and it moves forward to search the next road segment in B (if any) (Lines 2-9); otherwise, we add r's neighboring road segments to B (if not yet overlapping with B_{min}) for further investigation (Line 10-11). Finally, the union of the *Prob*-reachable regions are reported (Line 12).

5 Processing the Intersection-of-Multi-Location ST Reachability Query (I-Query)

This section presents our new contributions in this paper, namely, the processing algorithms to answer the Intersection-of-Multi-Location ST Reachability Query, in short, I-Query. An I-Query is formally defined as $q = (S, T, L, Prob)$, with the same query settings as U-Query. However, the I-Query q asks for the *Prob*-reachable region from *all* of the location $s \in S$ during time interval $[T, T+L]$. In theory, if we consider each query location $s_i \in S$ as an S-Query, namely, $q = (s_i, T, L, Prob)$, with a result of *Prob*-reachable region as B_i, the answer of an I-Query is then the outer-most bounding regions of the *intersection* among all B_i's. Fig. 6a shows an example of I-Query with two start road segments, r_1 and r_2. The solid lines outline *Prob*-reachable region of the I-Query, which is the inner bounding region of the two single *Prob*-reachable intersecting regions of r_1 and r_2, where the non-intersecting parts (in dashed lines) are removed.

Naive solution. To solve an I-Query, a naive (but always working) solution is treating an I-Query as multiple S-Queries [4], answering them one by one, and intersecting the *Prob*-reachable regions of each S-Query to obtain the *Prob*-reachable region for the I-Query. However, this approach is not efficient because in any case, we have to run the S-Query n times even if there are no intersections in the results. Therefore, we are motivated to develop an I-Query processing algorithm that

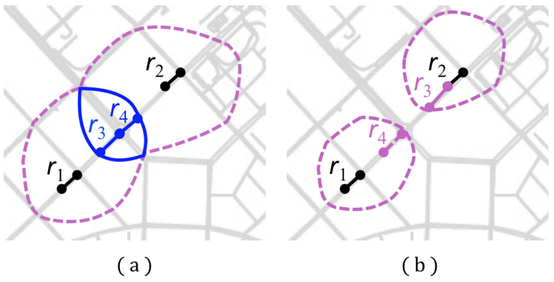


Fig. 7. *Intersect Conditions.* Solid line indicates the inner bounding region which is the real intersecting reachable boundary of both r_1 and r_2 while dashed line indicating non-intersecting outer bounding regions. Road segments r_1 and r_2 are the start locations from an I-Query. In (a), Road segment r_3 and r_4 are on the inner bounding regions while in (b) they are on the boundary of r_2 and r_1 correspondingly.

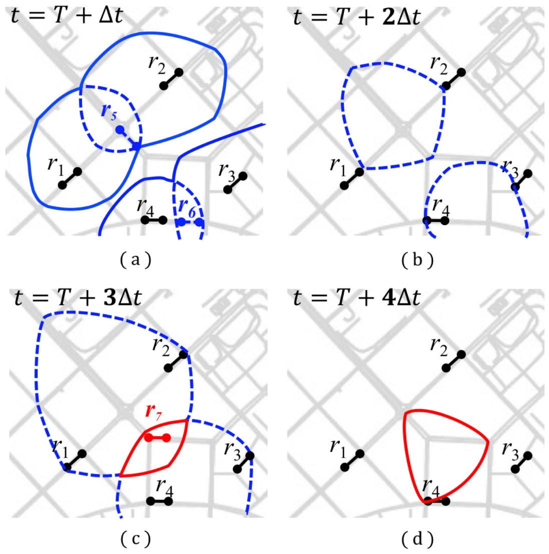


Fig. 8. *I-Query Example.* In (a) and (b), solid lines indicate the outer bounding regions of both r_1 and r_2, as well as r_3 and r_4, while dashed lines indicating inner bounding regions. Road segments r_1, r_2, r_3 and r_4 are the start locations from an I-Query. Road segment r_5 is on the intersecting boundary of r_1 and r_2 while r_6 on r_3 and r_4. However, these two dashed lines do not intersect with each other, indicating no intersecting road segments of all start road segments $r_1, \ldots, r_4$. In (c) and (d), solid lines indicate the inner bounding regions of all start locations, while dashed lines indicate partial intersecting boundary. Road segment r_7 is on the intersection of bounding regions.

can automatically take advantage of the intersection information, to avoid useless expansion over the road network.

Query processing algorithm for I-Query. The basic idea behind the query processing algorithm for I-Queries is similar to that for U-Queries, which is also a two-step approach. As shown in Fig. 6b, it employs *(1) an Intersection-of-Multi-location Query Maximum Bounding Region Search (I-MQMB)* algorithm to identify the maximum bounding region of candidate roads for the I-Query, and *(2) an I-Query Trace Back Search (I-TBS) algorithm* to refine the results. These two algorithms are different from the previous U-Query solutions. Because an I-Query requires that the target road segments are reachable from all query locations while a U-Query requires that the reachability from any of query locations.

5.1 Step 1: The I-MQMB Algorithm

I-Queries are more challenging compared to U-Queries since in addition to road expansions, we also need to check the intersection of expanded roads from different sources. Therefore, the I-MQMB algorithm now has two Stages: (1) Intersect bounding region and (2) Intersect-Unified bounding region. At STAGE 1, we keep track of all the intersecting bounding regions of the start road segments. If all of them intersect at some time, we get this intersecting bounding region and enter STAGE 2. At STAGE 2, we treat the intersecting bounding region as a single candidate bounding region and use MQMB algorithm to figure out the final maximum bounding region.

Specifically, we similarly form a start road segment set R_s by matching each start location $s_i \in S$ to a start road segment r_i using the R-tree in the ST-Index at the beginning. Then, for each road segment, we track the expansion of its reachable region individually using the Con-Index. Next, we compare each intersected pair to figure out whether a new intersection will occur or not. Those road segments can be intersected by the following rule: Given an intersected pair R_a and R_b, for any road segment $r \in R_a$, if the distance between the nearest road segment $r_s \in Inter[R_b]$ (current reachable region of R_b) to r, i.e., $r_s = \mathbf{argmin}_{r' \in Inter[R_b]}\{dis(r', r)\}$ and the nearest road segment $r_{ss} \in R_b$ to r, i.e., $r_{ss} = \mathbf{argmin}_{r' \in R_b}\{dis(r', r)\}$ is greater than the distance between r_{ss} and r, r should be included into the new intersection R_{ab}. Once acquiring an intersection R_{ab} equal to the start road segment list R_s, we find the intersection for all start locations. In Fig. 7a, road segment r_3 and r_4 are reachable for both $R_a = \{r_1\}$ and $R_b = \{r_2\}$. Therefore, we get a new intersection $R_{ab} = \{r_1, r_2\}$ by checking $dist(r_2, r_3) \geq dist(r_2, r_4)$. Otherwise, Fig. 7b illustrates no new intersection.

The detailed Intersection-of-Multi-location Query Maximum Bounding region search algorithm is summarized in Algorithm 3. Fig. 8 shows a concrete example on how I-MQMB algorithm works to answer a simple I-Query with four start locations by efficiently identifying the maximum bounding region at the first stage. Given the query $q = (s_i, T, L, Prob)$, where $i = 4, L = 4\Delta t$, we first match each start location s_i to a start road segment r_i (Line 1) and then find the intersecting parts of these road segments as many as possible within the given time duration $[T, T + L]$ (Lines 2-8). According to Fig. 8a, the maximum bounding region of start road segments r_1 and r_2 have already intersected during time interval $[T, T + \Delta t]$, as well as start road segments r_3 and r_4 (Lines 9-20). Note that once the bounding region of a start road segment has intersected with another one, the intersection of these two road segments will be marked as "checked", so that the non-intersecting bounding regions will not be expanded during next time interval (Lines 21-26). To be precise, taking time interval $[T + \Delta t, T + 2\Delta t]$ as an example in Fig. 8b, there are only two intersect-unified bounding regions $\{r_1, r_2\}$ and $\{r_3, r_4\}$ instead of four bounding regions of $r_1, \ldots, r_4$ respectively. In detail, the upper left dashed line indicates the expansion of intersecting parts of start road segments r_1 and r_2, while the lower right one indicates r_3 and r_4. Next, it keeps on intersecting alternative road segments until the complete intersection has been checked. As shown in Fig. 8c, solid line indicates the intersecting of $\{r_1, r_2\}$ and $\{r_3, r_4\}$. In other words, during time interval $[T + 2\Delta t, T + 3\Delta t]$, it has already checked the complete intersection $\{r_1, \ldots, r_4\}$. Finally, it continues expanding the complete intersect-unified bounding regions (red line in Fig. 8d) until $t = T + 4\Delta t$ (Lines 27-29). Otherwise, the algorithm reports an empty set, indicating no overall intersection of all the start locations (Line 30). Such mechanism ensures the efficiency of I-MQMB algorithm and avoids useless expansions.

Algorithm 3. Intersection-of-Multi-Location Query Maximum Bounding Region Search (I-MQMB) Algorithm

```
Input: I-Query $q = \{S = \{s_1, \ldots, s_n\}, T, L, Prob\}$.
Output: Intersect-Unified Maximum bounding region set $Result$.
1: Find start road segment list $R_s$ in ST-Index, Compute $k$ with $k\Delta t \leq L < (k+1)\Delta t$
2: Initial checked intersection $Inter[R_s] = R_s$, flag list $Flag = \emptyset$
3: for $k' = 0$ to $k-1$ do
4:   STAGE 1 Intersect bounding region
5:   for $\forall R_a$ in $Inter.keys$ do
6:     for $\forall r$ in $Inter[R_a]$ do
7:       Bounding set $B \leftarrow B \cup F(r, T + k'\Delta t)$.
8:     $Inter[R_a] \leftarrow B$; $B \leftarrow \emptyset$
9:   $interKeys \leftarrow Inter.keys$; $R_{ab} \leftarrow \emptyset$
10:  while $interKeys \neq \emptyset$ do
11:    $R_a \leftarrow dequeue(interKeys)$
12:    for $\forall R_b$ in $Inter.keys$ do
13:      $interT$ = false
14:      if $R_a \neq R_b$ and $R_a \cup R_b \notin Flag$ then
15:        $Flag.add(R_a \cup R_b)$
16:        for $\forall r$ in $Inter[R_a]$ do
17:          $r_s = \mathbf{argmin}_{r' \in Inter[R_b]}\{dis(r', r)\}$
18:          $r_{ss} = \mathbf{argmin}_{r' \in R_b}\{dis(r', r)\}$
19:          if $dist(r_{ss}, r_s) \geq dist(r_{ss}, r)$ then
20:            Intersection set $R_{ab} \leftarrow R_{ab} \cup \{r\}$
21:        if $R_{ab} \neq \emptyset$ then
22:          $interT$ = true; continue
23:    if $interT$ then
24:      $interKeys.delete(R_b)$; $interKeys.add(R_{ab})$
25:      $Inter[R_a \cup R_b] \leftarrow R_{ab}$;$Inter.delete(R_a, R_b)$;$R_{ab} \leftarrow \emptyset$
26:  $Flag = \emptyset$
27:  if $Inter.keys = R_s$ then
28:    STAGE 2 Intersect-Unified bounding region
29:    return MQMB($Inter[R_s], T + k'\Delta t, L - k'\Delta t, Prob$)
30: return Null
```

5.2 Step 2: The I-TBS Algorithm

Once we obtain the maximum bounding region of the query, we can use the Trace Back Search algorithm presented in the U-Query section to find the final result. However, we have to make a modification on the algorithm. When we evaluate each candidate road segment r in the bounded region, we need to evaluate its reachability from all the query road segments in R. Only when the road segment is reachable from *all* (as opposed to only one in U-Query) the query road segments can it be added to the final output. We name this algorithm as the I-Query Trace Back Search algorithm. The pseudo code is presented in Algorithm 4. Note Lines 4-11 are different from the U-TBS algorithm. If a candidate road cannot be reached from any of the start road segments, it will be dropped out of the results.

Note that our approach to solving I-Queries is a feasible, heuristic solution. The algorithm might not always provide the theoretically optimal performance but is guaranteed to outperform the naive solution. We show in the experimental results that our algorithms are efficient enough to solve I-Query compared with the baseline method.

Algorithm 4. I-Query Trace Back Search (I-TBS) Algorithm

```
Input: Bounding set $B_{max}$ and $B_{min}$, probability $Prob$, start road segments $R$.
Output: Bounding set $B'$ with respect to $Prob$
1: $B \leftarrow B_{max}$, $B' = \emptyset$
2: while $B \neq \emptyset$ do
3:   $r \leftarrow dequeue(B)$
4:   $violate$ = false
5:   for each $r_s$ in $R$ do
6:     if $probability(r, r_s) < Prob$ then
7:       $B \leftarrow (neighbor(r) - B_{min}) \cup B$
8:       $violate$ = true
9:       Break
10:  if $violate$ == false then
11:    $B' \leftarrow \{r\} \cup B'$
12: return $B'$
```

6 Complexity Analysis

In this section, we analyze the time complexities for all the algorithms we propose in this paper. Since the time complexity depends on the query parameters and actual data distributions, we analyze the best-case and worst-case complexities.

For Algorithm 1 MQMB, we search the Con-Index for k steps until time duration L, that is, $k = \frac{L}{\Delta t}$. In each time slot $[T + k'\Delta t, T + (k'+1)\Delta t]$, $(0 \leq k' \leq k-1)$, for each road segment r in current bounding set $B_{k'}$, we union their maximum reachable road segments $F(r, T + k'\Delta t)$. Iteratively, we check the Con-Index for (1 + 1 + B_1 + 1 + B_1 + B_2 + ... + 1 + $\sum_1^k B_{k'}$) times. According to historical data, we obtain the maximum value B_M of all maximum reachable road segment set $B_{k'}$, which is the size of the largest Far ID list in Con-Index (large constant varying with city size). Therefore, 1 + 1 + B_1 + 1 + B_1 + B_2 + ... + 1 + $\sum_1^k B_{k'}$ is bounded by 1 + 1 + B_M + 1 + 2 × B_M + ... + 1 + k × B_M (=$k + 1 + \frac{(B_M + k \times B_M) \times k}{2}$). The best case of complexity is $O(k^2)$ with B_M=1 indicating only one trajectory in each Far ID list. In the worst case, for each step we have to check all road segments in current maximum bounding region. Therefore, the worse-case time complexity is $O(2 \times k^2 \times B_M) = O(k^2 \times B_M)$.

For Algorithm 2 U-TBS, we need to find the bounding set satisfying probability *Prob* requirement. With respect to *Prob*, the best case is that all road segments in the maximum bounding region satisfy the requirement. In contrast, the worst case requires to search all road segments between maximum and minimum bounding regions. Hence, the time complexity depends on the maximum bounding region B_{max} acquired by Algorithm 1 MQMB. Therefore, the best case is $O(B_{max})$ and the worst case is bounded by $O(B_{max}^2 - B_{min}) = O(B_{max}^2)$ where B_{min} indicates the minimum bounding region.

For Algorithm 3 I-MQMB, at STAGE 1, we need to double check whether the road segments of two bounding regions intersect or not, which takes quadratic time to compare each road segment pair. If we can find the intersection bounding region at k_1 step ($k_1 < k$), then we use MQMB to find the final intersect-unified maximum bounding region set for the rest steps at STAGE 2. Hence, I-MQMB takes $O((k_1^2 \times B_M)^2) = O(k_1^4 \times B_M^2)$ time at STAGE 1 while takes $O((k - k_1)^2 \times B_M)$ at STAGE 2. Therefore, the best case is to

TABLE 2
Time Complexity

Algorithm	Best Case	Worst Case
MQMB	$O(k^2)$	$O(k^2 \times B_M)$
U-TBS	$O(B_{max})$	$O(B_{max}^2)$
I-MQMB	$O(k^2)$	$O(k^4 \times B_M^2)$
I-TBS	$O(B_{max})$	$O(B_{max}^2)$

TABLE 3
Evaluation Configurations

Parameter	Settings
Travel Duration L	$\{5, 10, \ldots, 35\}$ minutes. Default: 15 min
Number of Query Locations $\lvert S\rvert$	$\{1, 2, \ldots, 9\}$. Default: 3
Probability *Prob*	$\{20\%, 40\%, \ldots, 100\%\}$. Default: 20%
Time Interval Δt	$\{5, 10, 15, 20\}$ minutes. Default: 5 min
Baseline Solutions	Multiple S-Queries and Intersect Results
I-Query Algorithms	I-MQMB follwed by I-TBS

find the intersection of bounding regions at the first step and then take only $O((k-1)^2) = O(k^2)$ time with B_M=1. The total complexity is $O(1+k^2) = O(k^2)$. In contrast, the worst case fails to find the intersection bounding region until the end, indicating the total complexity of $O(k^4 \times B_M^2)$.

For Algorithm 4 I-TBS, the complexity is the same as that of the U-TBS algorithm, which is $O(B_{max})$ in the best case and $O(B_{max}^2 - B_{min}) = O(B_{max}^2)$ in the worst case, where B_{min} indicates the minimum bounding region.

7 Evaluation

In this section, we conduct extensive experiments to evaluate our new query processing algorithms for I-Queries using the a taxi trajectory dataset from Shenzhen, China. For detailed evaluations of the S-Query and the U-Query, please refer to our conference paper [4]. To make this paper self-contained, we also include those results in the supplementary materials, available online, of this paper. We mainly focus on the evaluation of the I-Query solution (I-MQMB + I-TBS). Our baseline method is a naive solution, where we run the single-location query (S-Query) algorithms in our prior conference paper [4] for each query location independently and get the final query results from the intersection of all results of each S-Query. Below, we elaborate on the dataset we used, experiment configurations, and experimental results.

7.1 Data Descriptions and Experiment Configurations

We use a large-scale trajectory dataset collected from taxis in Shenzhen, with an urban area of about 400 square miles and ten million people. The dataset was collected during the month of November, 2014. These trajectories represent 21,385 unique taxis in Shenzhen. They are equipped with GPS devices, which periodically (i.e., roughly every 30 seconds) generate GPS records.

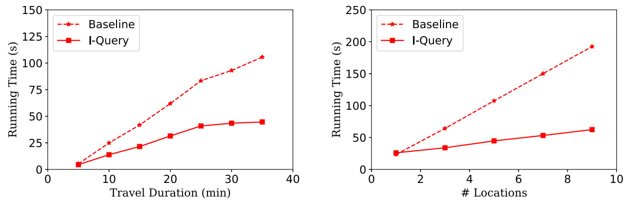
Hence, each GPS record in our database is represented as a spatio-temporal point of a taxi, where in total 407,040,083 GPS records were obtained. Each record has five core attributes including *trajectory ID*, *longitude*, *latitude*, *speed* and *time*. To calculate the probability of reachable areas, we consider the same taxi at different dates as different trajectories, e.g., with different trajectory IDs. The detailed experiment configurations are listed in Table 3. All the experiments 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.

7.2 I-Query Evaluation Results on Real Data

Varying Time Duration L. First, we execute I-Queries with three query locations (Fubin Neighborhood, Futian Business District and Huanggang Port) and we set the probability threshold as 20 percent, start time as 10am and vary the length of travel duration L from 5 to 35 minutes. Fig. 9a shows the running time of the baseline versus our I-Query algorithms. The result shows our I-MQMB+I-TBS algorithm save around 50 percent processing time over running the baseline solution. Specifically, our I-MQMB+I-TBS algorithm only takes 21 seconds to process an I-Query with travel duration L=15 mins, while the baseline method takes 42 seconds. Particularly, when the travel duration is larger than 25 minutes, running time of our I-MQMB+I-TBS algorithms becomes flat. The reason is that in this case, the reachable regions of these 3 locations have already intersected to a small region at 25 minutes. After that, we only continue to expand the new intersection region which is much smaller compared to the outer bounding region. At last, it takes few more seconds to compute the final result satisfying the query requirement using the I-TBS algorithm.

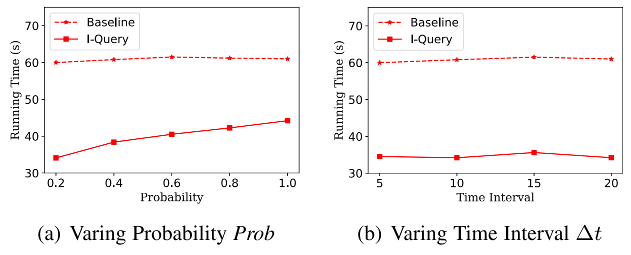
Varying Number of Query Locations $|S|$. We also change the number of query locations in the I-Query from 1 to 9 randomly from the city of Shenzhen. We set start time as 10 am, travel duration as 20 minutes and probability as 20 percent. Fig. 9b shows the running time of the baseline versus our solution. It indicates that our I-MQMB+I-TBS algorithm does significantly outperform the baseline. Specifically, as the number of start locations are increasing, our solution's advantage becomes more and more apparent, saving up to 70 percent running time. Due to the extra step to check the intersection components in the I-MQMB algorithm, the overall solution may be slightly slower than the baseline when the query has only one location (i.e., the S-Query). However, this procedure does improve processing time with more than one location.

Varying Probability Thresold Prob. Then, we fix start locations, travel duration as 20 minutes and start time as 10 am to study how different query probabilities influence the performance of our I-MQMB+I-TBS algorithm. Fig. 11a shows that as we change query probability *Prob* in $\{20\%, 40\%, 60\%, 80\%, 100\%\}$, the number of accessed road segments during query processing also increases while the number of road segments within bounding regions decreases, which indicates that we have



(a) Varing Travel Duration L (b) Varing Number of Locations $|S|$

Fig. 9. Comparison of Baseline and I-Query over travel duration and number of locations.



(a) Varing Probability *Prob* (b) Varing Time Interval Δt

Fig. 10. Comparison of Baseline and I-Query over probability and time interval.

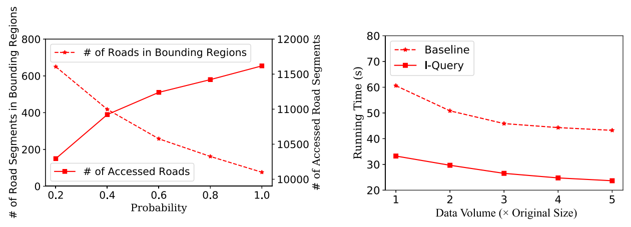
to access further road segments in order to satisfy higher probability requirement since we search from the maximum bounding region to the minimum bounding region in I-TBS algorithm. Fig. 10a demonstrates the test results of running time. With an increasing probability threshold, the running time increases roughly linearly due to accessing more road segments. The overall improvement of running time compared to the baseline solution is between 30-50 percent.

Varying Time Granularity Δt. In addition, we also evaluate how the time interval Δt affects the running time of our I-MQMB+I-TBS algorithm. Fig. 10b shows the running time with varying Δt between 5 minutes and 20 minutes with a step of 5 minutes. We observe that the running time is almost unchanged when varying the time intervals. This indicates that our I-MQMB+I-TBS algorithm is stable on the system parameter, Δt, that governs the time granularity in the indexing structure. The reason behind this is that increasing the time interval length with L fixed will reduce the number of time steps but increase the total number of trajectories in each time slot, therefore making more roads qualified for the query. The two factors lead to a stable time cost.

Memory Cost. We conduct all experiments using a real taxi trajectory dataset with 14 million trajectories, which was collected for 30 days in November, 2014 from 21,385 unique taxis in Shenzhen, China. According to our indexing structure, it saves the whole road network and its traversing trajectory IDs instead of original trajectories. With setting time granularity as 5 mins, the memory cost of Con-Index is 31 MB while the disk size of storing R-trees is 36 MB. Therefore, our methods are practical on real big data.

7.3 I-Query Evaluation on Synthetic Data

In order to evaluate the performance of our algorithms on large datasets, we generate a synthetic data of 30 days by sampling trajectories from the real dataset we have. We adjust the number of sampling, which means the ratio between the synthetic data volume and the real data volume. For each day in the synthetic data, we randomly sample true trajectories from the real data (any day and time) and re-arrange these trajectories at a different start time (day and hour) in the synthetic dataset. When the number of sampling equals to one, the synthetic dataset is of the same size as the real dataset but with a shuffled travel date of each trajectories. Then, in order to increase the traffic volume, we sample multiple times with replacement to increase the number of unique trajectories traversing on road segments, which leads to an increase of the reachability of road segments according to our probability definition. We sample up to five times the volume of the original data.



(a) Comparison of Road Segments in Bounding Regions and Accessed Road Segments during Query Processing (b) Comparison of Baseline and I-Query on Synthetic Data

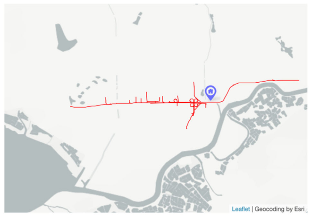
Fig. 11. Comparison of Road Segments over probability on real data and Baseline and I-Query on synthetic data.

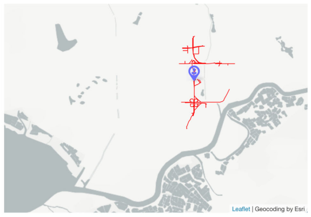
We fix all the parameters to the default values. Fig. 11b shows the running time of the algorithms. The running time decreases more and more slowly but the I-Query algorithms (I-MQMB + I-TBS) still outperform the baseline by 50-66 percent. The main reason is that the dominant factor of running time during query processing is the I-TBS algorithm. When we double the traffic volume, it significantly increases the number of unique trajectories traversing over the whole road network which results in expanding the *Prob*-reachable region of each query roads. Because I-TBS algorithm trace back searches from the maximum to the minimum bounding regions, it is easier to meet the threshold *Prob* as early as possible with a larger number of trajectories. Once the probability requirement is met, I-TBS algorithm will terminate. The cost of this algorithm dominates the total time cost during query processing, which explains the reasons why the running time decreases even when we increase the data volume. However, when the data volume increases to four times the volume of the original data, the impact on the reachability becomes limited because trajectories traversing each roads on different dates are saturated.

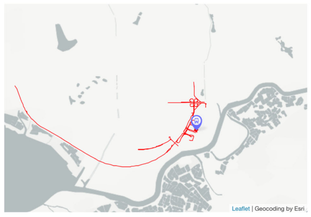
In summary, the evaluation results show that our I-MQMB+I-TBS can reduce on average 50 percent running time over baseline solution with different parameter settings such as varying travel duration L, number of start locations $|S|$, probability *Prob* and time interval Δt. In addition, it is stable with different time interval Δt.

7.4 Case Study

We visualize the results of I-Queries under different parameters. Fig. 12 shows the results of the first case. We observe that the reachable region of all three locations is the intersection of reachable areas of three individual S-Queries. According to our analysis, people from the transportation hub, business district and living neighborhood could gather at the Union Plaza and meet for lunch. In our case, people from Fubin Neighborhood and Huanggang Port are able to reach quite larger region than others from Futian Business District because of the Binhe Avenue and Jinggang'ao Expressway (horizontal crossing the map). However, the final intersecting region mostly depends on the traffic condition on normal roads which is deceptively changing with the time of day. Fig. 13 visualizes the results when we reduce the duration time L. The I-Query returns Null result for this case, suggesting that the query results are sensitive to local traffic condition.


(a) S-Query at Fubin Neighborhood


(b) S-Query at Futian Business District


(c) S-Query at Huanggang Port

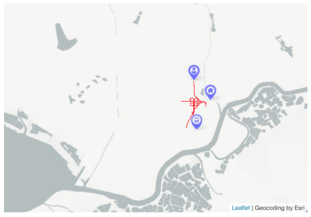

(d) I-Query at 3 locations

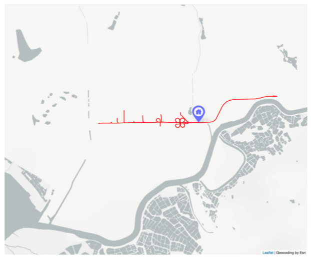
Fig. 12. Results of ST reachability queries at three locations.

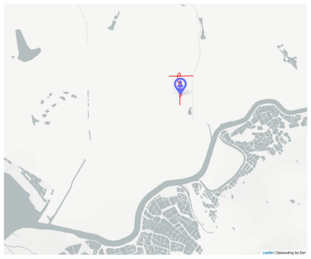
8 Related Work

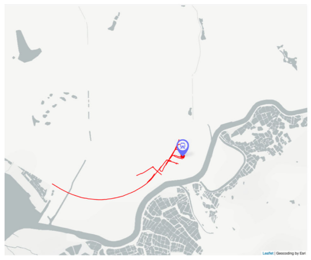
In this paper, we make the first attempt to study a problem of mining the spatio-temporal reachable region from a location and within a time interval. In this section, we discuss three topics that are closely related to our work and highlight the differences from them, including (1) Spatio-tempereal data management, (2) trajectory query processing, and (3) reachability query processing.

Spatio-Tempereal Data Management. Researchers have proposed a number of decent models to store and index those data. For spatial information, R-tree structure has been widely used to index data [12], [13]. Then, B-tree and R-tree are combined together to store both spatial and temporal information [14]. In these structures, an R-Tree is maintained for each leaf node of B-tree, which could take a huge space to store in either an internal or external storage. With the observation that most of R-trees share a similar or even same structure, a set of methods have been developed to compress the index structure into a reasonable size [15], [16]. Another approach to store and index spatio-tempereal data is to utilize its network structure. The connectivity of road segment could be maintained using adjacency matrix or adjacent list [3]. Predictive Tree has been proposed to maintain the reachability of road segments using additional information such as road length [2]. Moreover, grid-based indexing structures are used to store data: first, spatial dimensions are split into different grids, indexed by Quad-tree or KD-trees, etc; then, based on each spatial grid, different temporal indices could be built such as scalable and efficient trajectory index [8]. All models above share a common drawback, namely, they all use separate structure to represent spatial and temporal aspect of data. However, in real-world applications, we need an indexing structure that describes both spatial and temporal aspects of spatio-temporal data. For example, if two roads are connected to each other, they should be connected in both spatial dimension and temporal dimension. If it takes 5 minutes to travel from A to B, the node which represents road A should be connected to the node of road segment B in 5 minutes. Our proposed Con-Index structure fill this gap that record connections between connected road segments across different time slots using speed information.

Trajectory Query Processing. With a large amount of trajectory data generated over time, it is increasingly challenging to


(a) S-Query at Fubin Neighborhood


(b) S-Query at Futian Business District


(c) S-Query at Huanggang Port

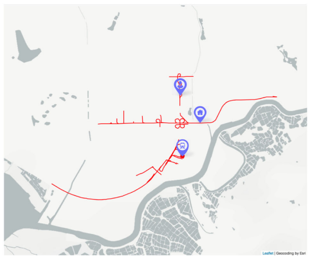

(d) I-Query at 3 locations

Fig. 13. Results of Null for ST reachability queries at three locations.

answer various trajectory queries in different application scenarios in urban computing [17], [18], [19]. One typical trajectory query is range query, where historical trajectories are used to predict the possibility that a moving object will go towards a next location. Such query can be answered based on predicted route for every object in the trajectory database [20]. Moreover, the transformed Minkowski Sum has been used to answer such range queries, if the query input is a circular region [21]. In [22], sampling based approach is proposed to efficiently answer trajectory aggregate queries, that ask for the total number of trajectories in a user-specified spatio-temporal region. Another type of popular trajectory query is route query, that answers how to get to a location from another location. For simple shortest path queries, Dijkstra Algorithm is the optimal method when any additional information is available [23]. However, information such as transit nodes, which can be viewed as the connection between local road network and general road network, can be used to accelerate shortest path query [24]. Meanwhile, predicting traffic speed in road network can also be used to find shortest path [25] and solve traveling salesman problem [26]. Moreover, researchers have extensively studied trip planning problems, that seek for a best path passing through a set of distinct objects [27]. Others also take advantage of time-dependent travel time to solve routing problems [28], [29]. Overall, our proposed spatio-temporal reachability query problem is different from the trajectory queries described above, which aims to find the region that are reachable from a given location within a time interval. Note that instead of predicting speed information, traffic patterns reveal the traffic conditions in different times of day. Combined with appropriate probability calculation, traffic patterns can identify special activity at some specific days that can be applied to events prediction. However, speed patterns couldn't have the same effect.

Reachability Queries. Reachability queries are usually conducted to test whether there is a path from a node u to another node v in a (directed) graph setting, which have been widely studied in the literature, and are treated as a very basic type of graph queries for many applications. There are three types of techniques commonly used to process graph reachability query. The transitive closure (TC) of vertex v is the set of vertices that v can reach in a graph, normally this TC structure is large and different methods are proposed to compress TC [30], [31]. Moreover, 2-hop labeling schema are introduced to solve the query and some heuristic methods are proposed to reduce the size of labels [32]. The methods [33], [34] construct a small index with a small construction cost to solve such reachability queries. In urban setting, the road networks can be naturally viewed as a (directed) graph. Therefore, these approaches can be applied to find the reachabilities on road networks. However, all these methods are designed over static graphs [35]. Our prior work [4] was among the first to explore data-driven reachability queries. However, it only considered single-location and union-of-multi-location query. In this paper we consider the intersection-of-multi-location query (I-Query), which is more challenging.

9 Discussion and Future Work

With the widespread development of location-based services, trajectory data has been generated and collected in a variety of applications. Data privacy has become a more important problem. Therefore, improper mining and publishing trajectory data may jeopardize individual privacy. In our paper, all the taxi trajectories are identified by the taxi ID number. However, we do not use any personal information of the drivers and do not identify their home locations. The proposed solutions can be applied on any anonymized trajectory dataset. A copy of the de-identified dataset and the code for this work is available online.[1] In addition to anonymized data, we would like to consider privacy preserving query technique on top of the framework to improve privacy protection, such as the proposed Location Anonymizer [36].

This paper formulated and solved the problem of multi-location reachability query based on real trajectory data. This problem is useful to a number of applications such as location-based advertising and store location choice. However it is also challenging due to high computational cost and large volume of real data. In this paper we extend the prior work by introducing a new type of reachability query, namely, the Intersection-of-multi-location ST reachability Query (I-Query), where the reachable region from all the query locations is found. We propose a new I-MQMB algorithm to find the maximum bounding region of the solution effectively and use an I-TBS algorithm to refine the results and find the final solution. Evaluation results on real and synthetic data show that our solution can save 50-70 percent running time compared to naive solution using our prior results.

In the future, we plan to take advantage of existing methods to protect the user privacy and apply our indexing structures and algorithms to a big data processing framework in order to tackle large-scale trajectory data more efficiently, such as paralleling our proposed methods using MapReduce [37].

Acknowledgments

Yichen Ding and Xun Zhou were partially supported by the NSF under Grant No. IIS-1566386. Guojun Wu and Yanhua Li were supported in part by NSF grants CNS-1657350 and CMMI-1831140, and a research grant from DiDi Chuxing Inc. Jie Bao and Yu Zheng were supported by the National Key R&D Program of China (2019YFB2101805). Yu Zheng was also supported by NSFC (Grant No. 61672399 and No. U1609217).

1. https://www.biz.uiowa.edu/faculty/xzhou/paper/Reachability_TKDE/

References

[1] J. Bao, C.-Y. Chow, M. F. Mokbel, and W.-S. Ku, "Efficient evaluation of k-range nearest neighbor queries in road networks," in *Proc. 11th Int. Conf. Mobile Data Manage.*, 2010, pp. 115–124.

[2] A. M. Hendawi, J. Bao, M. F. Mokbel, and M. Ali, "Predictive tree: An efficient index for predictive queries on road networks," in *Proc. IEEE 31st Int. Conf. Data Eng.*, 2015, pp. 1215–1226.

[3] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proc. 29th Int. Conf. Very Large Data Bases*, 2003, pp. 802–813.

[4] G. Wu, Y. Ding, Y. Li, J. Bao, Y. Zheng, and J. Luo, "Mining spatio-temporal reachable regions over massive trajectory data," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 1283–1294.

[5] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *Proc. 11th Int. Conf. Mobile Data Manage.*, 2010, pp. 43–52.

[6] D. H. T. That, I. S. Popa, and K. Zeitouni, "TRIFL: A generic trajectory index for flash storage," *ACM Trans. Spatial Algorithm Syst.*, vol. 1, no. 2, 2015, Art. no. 6.

[7] Y. Tao, X. Xiao, and R. Cheng, "Range search on multidimensional uncertain data," *ACM Trans. Database Syst.*, vol. 32, no. 3, 2007, Art. no. 15.

[8] V. P. Chakka, A. C. Everspaugh, and J. M. Patel, "Indexing large trajectory data sets with seti," in *Proc. Conf. Innovative Data Syst. Res.*, 2003, vol. 75, p. 76.

[9] V. Botea, D. Mallett, M. A. Nascimento, and J. Sander, "PIST: An efficient and practical indexing technique for historical spatio-temporal point data," *GeoInformatica*, vol. 12, no. 2, pp. 143–168, 2008.

[10] I. Sandu Popa, K. Zeitouni, V. Oria, D. Barth, and S. Vial, "Indexing in-network trajectory flows," *VLDB J.*, vol. 20, no. 5, pp. 643–669, 2011.

[11] B. Zhang and G. Trajcevski, "Probabilistic speed profiling for multi-lane road networks," in *Proc. 18th IEEE Int. Conf. Mobile Data Manage.*, 2017, pp. 164–173.

[12] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *ACM SIGMOD Record*, vol. 14, no. 2, pp. 47–57, 1984.

[13] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1990, pp. 322–331.

[14] R. H. Güting, T. de Almeida, and Z. Ding, "Modeling and querying moving objects in networks," *VLDB J.*, vol. 15, no. 2, pp. 165–190, 2006.

[15] V. T. De Almeida and R. H. Güting, "Indexing the trajectories of moving objects in networks," *GeoInformatica*, vol. 9, no. 1, pp. 33–60, 2005.

[16] Y. Tao, D. Papadias, and J. Sun, "The TPR*-tree: An optimized spatio-temporal access method for predictive queries," in *Proc. 29th Int. Conf. Very Large Data Bases*, 2003, pp. 790–801.

[17] A. V. Khezerlou, X. Zhou, L. Tong, Y. Li, and J. Luo, "Forecasting gathering events through trajectory destination prediction: A dynamic hybrid model," *IEEE Trans. Knowl. Data Eng.*, 2019, doi: 10.1109/TKDE.2019.2937082.

[18] Y. Zheng, "Trajectory data mining: An overview," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, 2015, Art. no. 29.

[19] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, 2014, Art. no. 38.

[20] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen, "Path prediction and predictive range querying in road network databases," *VLDB J.*, vol. 19, no. 4, pp. 585–602, 2010.

[21] R. Zhang, H. Jagadish, B. T. Dai, and K. Ramamohanarao, "Optimized algorithms for predictive range and KNN queries on moving objects," *Inf. Syst.*, vol. 35, no. 8, pp. 911–932, 2010.

[22] Y. Li *et al.*, "Sampling big trajectory data," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, 2015, pp. 941–950.

[23] M. Thorup and U. Zwick, "Approximate distance oracles," *J. ACM*, vol. 52, no. 1, pp. 1–24, 2005.

[24] H. Bast, S. Funke, and D. Matijević, "TRANSIT: Ultrafast shortest-path queries with linear-time preprocessing," in *Proc. 9th DIMACS Implementation Challenge—Shortest Path*, 2006. [Online]. Available: http://www.dis.uniroma1.it/~challenge9/

[25] E. Kanoulas, Y. Du, T. Xia, and D. Zhang, "Finding fastest paths on a road network with speed patterns," in *Proc. IEEE 22nd Int. Conf. Data Eng.*, 2006, pp. 10–10.

[26] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "LC-RNN: A deep learning model for traffic speed prediction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 3470–3476.

[27] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *Proc. Int. Symp. Spatial Temporal Databases*, 2005, pp. 273–290.

[28] B. Fleischmann, M. Gietz, and S. Gnutzmann, "Time-varying travel times in vehicle routing," *Transp. Sci.*, vol. 38, no. 2, pp. 160–173, 2004.

[29] G. V. Batz, R. Geisberger, P. Sanders, and C. Vetter, "Minimum time-dependent travel times with contraction hierarchies," *J. Exp. Algorithmics*, vol. 18, pp. 1–4, 2013.

[30] Y. Chen and Y. Chen, "An efficient algorithm for answering graph reachability queries," in *Proc. IEEE 24th Int. Conf. Data Eng.*, 2008, pp. 893–902.

[31] S. J. van Schaik and O. de Moor, "A memory efficient reachability data structure through bit vector compression," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 913–924.

[32] J. Cai and C. K. Poon, "Path-hop: Efficiently indexing large graphs for reachability queries," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, 2010, pp. 119–128.

[33] S. Seufert, A. Anand, S. Bedathur, and G. Weikum, "FERRARI: Flexible and efficient reachability range assignment for graph indexing," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 1009–1020.

[34] R. R. Veloso, L. Cerf, W. Meira Jr, and M. J. Zaki, "Reachability queries in very large graphs: A fast refined online search approach," in *Proc. Int. Conf. Extending Database Technol.*, 2014, pp. 511–522.

[35] H. Wu, Y. Huang, J. Cheng, J. Li, and Y. Ke, "Reachability and time-based path queries in temporal graphs," in *Proc. IEEE 32nd Int. Conf. Data Eng.*, 2016, pp. 145–156.

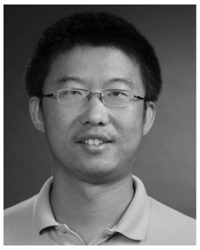
[36] M. F. Mokbel, "Towards privacy-aware location-based database servers," in *Proc. IEEE 22nd Int. Conf. Data Eng. Workshops*, 2006, pp. 93–93.

[37] L. Alarabi, M. F. Mokbel, and M. Musleh, "ST-Hadoop: A MapReduce framework for spatio-temporal data," *GeoInformatica*, vol. 22, no. 4, pp. 785–813, 2018.

Yichen Ding received the bachelor's degree in statistics and the MS degree in data science. She is currently working toward the PhD degree in the Department of Business Analytics, Tippie College of Business, The University of Iowa. Her current research interests include big data analytics, spatial temporal data mining and urban computing.

Xun Zhou received the PhD degree in computer science from the University of Minnesota, Twin Cities, in 2014. He is currently an assistant professor with the Department of Business Analytics, The University of Iowa. His research interests include big data management and analytics, spatial and spatio-temporal data mining, and Geographic Information Systems (GIS). He has published more than 50 papers in these areas and has received four best paper awards. He also served as a co-editor-in-chief of Springer's Encyclopedia of GIS, 2nd Edition.

Guojun Wu received the bachelor's degree in management information system from Beihang University. He is currently working toward the PhD degree in the Data Science Program, Worcester Polytechnic Institute. His general research interests include Urban data analysis and User choice modeling.



Yanhua Li received two PhD degrees in electrical engineering from the Beijing University of Posts and Telecommunications, Beijing in China, in 2009, and in computer science from the University of Minnesota at Twin Cities, in 2013, respectively. He has worked as a researcher in Noahs Ark LAB at Hong Kong from August 2013 to December 2014, and has interned in Bell Labs in New Jersey, Microsoft Research Asia, and FutureWei Research Labs of America from 2011 to 2013. He is currently an assistant professor with the Department of Computer Science, Worcester Polytechnic Institute (WPI) in Worcester, MA. His broad research interests are in smart cities, data-driven cyber-physical systems (CPS), urban intelligence. He is a recipient of NSF CISE Research Initiation Initiative (CRII) Award.

Jie Bao received the PhD degree from the Department of Computer Science & Engineering, University of Minnesota at Twin Cities, under the advisory of Prof. Mohamed Mokbel, in 2014. He leads the Data Management Department, JD Intelligent City Business Unit, where he is in charge of the design and development of JD Urban Spatio-Temporal data engine (aka JUST), as well as all the data-centric products in the BU. His main research interests include urban computing, spatio-temporal data management/mining, and distributed computing platforms. He has published more than 40 research papers in refereed journals and conferences (e.g., SIGKDD, ICDE, AAAI, VLDB, SIGMOD, and SIGSPATIAL). Before joining JD, he was a researcher in MSRA from 2014- 2017.

Yu Zheng is currently the vice president of JD.COM, leading the Intelligent Cities Business Unit and JD Intelligent Cities Research. He also serves as the chief data scientist at JD Digits, passionate about using big data and AI technology to tackle urban challenges. Before Joining JD.COM, he was a senior research manager at Microsoft Research. He is also a chair professor at Shanghai Jiao Tong University and an adjunct professor at the Hong Kong University of Science and Technology. He currently serves as the editor-in-chief of the *ACM Transactions on Intelligent Systems and Technology* and has served as chair on more than 10 prestigious international conferences, e.g., as the program co-chair of ICDE 2014 (Industrial Track), CIKM 2017 (Industrial Track), and IJCAI 2019 (industrial track). He is also a keynote speaker of AAAI 2019, KDD 2019 Plenary Keynote Panel and IJCAI 2019 Industrial Days. His monograph, entitled Urban Computing, has been used as the first text book in this field. In 2013, he was named one of the Top Innovators under 35 by MIT Technology Review (TR35) and featured by Time Magazine for his research on urban computing. In 2014, he was named one of the Top 40 Business Elites under 40 in China by Fortune Magazine. In 2017, he is honored as an ACM Distinguished scientist.

Jun Luo received the PhD degree in computer science from the University of Texas at Dallas, in 2006. He is a principal researcher and director at Lenovo Machine Intelligence Center in Hong Kong. Then he spent two years as postdoc in Utrecht University, the Netherlands. He has worked as an associate professor in Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences from 2008 to 2013 and as a researcher at Noahs Ark Lab in Hong Kong from 2013 to 2015. His research interests include big data, machine learning, spatial temporal data mining and computational geometry. He has published more than 100 journal and conference papers in these areas.