2017 IEEE 33rd International Conference on Data Engineering

Mining Spatio-Temporal Reachable Regions over Massive Trajectory Data

Guojun Wu[#,‡], Yichen Ding[#,‡], Yanhua Li[#], Jie Bao[†], Yu Zheng[†♯], Jun Luo[*§]
[#]Data Science Program, Worcester Polytechnic Institute (WPI), [†]Microsoft Research
[♯]School of Computer Science and Technology, Xidian University, China
[*]Shenzhen Institutes of Advanced Technology, CAS, [§]Lenovo Group Limited
Email: {gwu, yding3, yli15}@wpi.edu, {jiebao, yuzheng}@microsoft.com, jun.luo@siat.ac.cn

***Abstract*—Mining spatio-temporal reachable regions aims to find a set of road segments from massive trajectory data, that are reachable from a user-specified location and within a given temporal period. Accurately extracting such spatio-temporal reachable area is vital in many urban applications, e.g., (i) location-based recommendation, (ii) location-based advertising, and (iii) business coverage analysis. The traditional approach of answering such queries essentially performs a distance-based range query over the given road network, which have two main drawbacks: (i) it only works with the physical travel distances, where the users usually care more about dynamic traveling time, and (ii) it gives the same result regardless of the querying time, where the reachable area could vary significantly with different traffic conditions. Motivated by these observations, we propose a data-driven approach to formulate the problem as mining actual reachable region based on real historical trajectory dataset. The main challenge in our approach is the system efficiency, as verifying the reachability over the massive trajectories involves huge amount of disk I/Os. In this paper, we develop two indexing structures: 1) spatio-temporal index (ST-Index) and 2) connection index (Con-Index) to reduce redundant trajectory data access operations. We also propose a novel query processing algorithm with: 1) maximum bounding region search, which directly extracts a small searching region from the index structure and 2) trace back search, which refines the search results from the previous step to find the final query result. Moreover, our system can also efficiently answer the spatio-temporal reachability query with multiple query locations by skipping the overlapped area search. We evaluate our system extensively using a large-scale real taxi trajectory data in Shenzhen, China, where results demonstrate that the proposed algorithms can reduce 50%-90% running time over baseline algorithms.**

I. Introduction

A spatio-temporal reachability query aims to find the reachable area in a spatial network from a location in a given time period. As demonstrated in Figure 1, the spatio-temporal reachable region is very useful in many urban applications: 1) Location-based recommendation, when a user wants to find a nearby restaurant based on her current location and time, the spatio-temporal reachable region provides a candidate list for location recommendations; 2) location-based advertising, where some business owner finds out the potential spatial regions to arrange special activities, such as distributing coupons and sales discount; and 3) business coverage analysis, for example, a chained company, such as UPS and MacDonald's, can find their overall business spatial coverage of their branches. Those information can help them to make the right decisions, when planning for some new branch locations. In Figure 1, the first two application examples are illustrated using single-location reachability query (with only one querying location as input), and the third one is illustrated using a multi-location reachability query (with more than one querying locations).

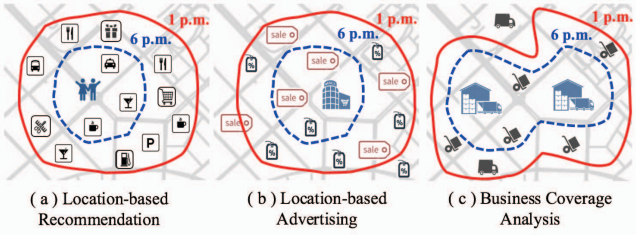


Fig. 1. **Application Examples.** These three simple examples offer an intuitive understanding of applying our methods to daily life with spatio-temporal reachability query.

The traditional reachability query, e.g., [1], [13], [17], on the road network has several drawbacks to fulfill the aforementioned urban applications: 1) most of the existing work focuses on reachable range based on spatial network distance rather time period. However, in the real application scenarios, users care more about the actual traveling time period rather than distance. 2) most of the existing work does not support queries at different time stamps. However, in reality, due to the different traffic conditions in the rush hours, the reachable area may vary significantly. The traditional spatial network based approach cannot capture such differences.

To improve the usability of the reachability query in the real application scenarios, we propose a data-driven approach to find the spatio-temporal reachable regions based on the massive real trajectory data collected over the road network. The main intuition behind our approach is that, we want to formulate the spatio-temporal reachability query as a data mining process, which finds out all the trajectories that passed the query location 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 dynamic data.

The main challenge to answer the reachability query with massive trajectory dataset is the system efficiency, because the trajectory data usually cannot fit into the memory, and

‡ The first two authors contributed equally in this work.

2375-026X/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDE.2017.171

analyzing them involves heavy I/O access from the disk, which results in long responding time. To improve the efficiency of the spatio-temporal reachability queries, we proposes a set of novel indexing structures and an efficient query processing algorithm to minimize the redundant disk accesses.

By introducing a connection index to represent the connections across road segments in two adjacent time slots, we first develop our **S**patio-**T**emporal Index (ST-Index) and **Con**nection Index (Con-Index) and propose **S**ingle-location Reachability **Q**uery **M**aximum/**M**inimum **B**ounding Region Search (SQMB) algorithm to determine the bounding region of a range query q. To further extend system functionality, we devise **M**ulti-location Reachability **Q**uery **M**aximum/**M**inimum **B**ounding Region Search (MQMB) algorithm to process multi-location Spatio-Temporal Reachability queries which has a number of starting locations with overlapping bounding regions. Within each bounding region, we also devise **T**race **B**ack **S**earch (TBS) algorithm to search the $Prob$-reachable region from the maximum bounding region. The main contributions of this paper can be summarized as follows.

- We propose a novel spatio-temporal reachability query, which introduces the temporal awareness to the traditional reachability query processing on the road networks. The proposed query provides a more realistic and data-driven result by mining the information within the massive historical trajectory dataset.
- We develop a spatio-temporal index and a connection index to facilitate the query processing with the dynamic connection information across road segments, which enable us the significantly reduce the searching space in answering the spatio-temporal reachability queries.
- We develop SQMB algorithm to address the spatio-temporal reachability query with single query location. The proposed algorithm first determines the bounding region of query q for further trace back search by utilizing the well-established spatio-temporal index and connection index to mining in large-scale trajectory database. Moreover, we also propose a trace back search algorithm to trace back from maximum to minimum bounding region until the spatio-temporal reachable area with the corresponding probability is obtained.
- We also develop an efficient algorithm to answer spatio-temporal reachability query with multiple query locations. Instead of performing SQMB algorithm on these single queries multiple times, our proposed algorithm MQMB significantly improves the efficiency, by employing shortest path techniques and eliminating duplicated influence of road segments in the overlapping regions.
- We conduct extensive experiments on a real-world road network with large-scale moving-object trajectory dataset (with 194 GB size) collected from a metropolis in China to evaluate the efficiency and effectiveness of our indexing structure and query processing algorithms. Our experimental results show that our SQMB and MQMB algorithms outperform the exhaustive search method for single- and multi-location query with 50%-90% reduction on the query processing time.

TABLE I
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 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 i^{th} 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 i^{th} trajectory in the trajectory history.
m	m is the total number of trajectory days.

The remainder of the paper is organized as follows. In Section II, we define our problem and outline our system framework. Section III presents our approach for data preprocessing. We elaborate on the indexing structure in Section IV. Section V presents our novel query processing algorithms. Section VI presents the evaluation results based on large scale real trajectory data. Section VII discusses the related work. Section VIII concludes the paper.

II. OVERVIEW

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

A. Basic Concepts

- **Road Network.** A road network can be viewed a directed graph $G(V, E)$, where E is as set of edges, and V is a set of vertices representing the intersections on the road network. Each road segment has a unique ID, an adjacent list of the connected road segments in the network, a list of intermediate points (2 terminal points at the beginning and the end) describing its shape, a value of its length, an indicator of direction (i.e., one-way or two-way), a type value describing its level (primary or secondary) and a MBR (Minimum Bounding Rectangles) describing its spatial range.
- **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 set of properties (e.g., travel speed, direction, or occupancy).

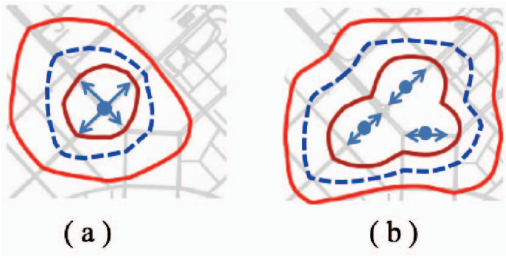


Fig. 2. **Spatio-Temporal (ST) Reachability Query.** A ST Reachability Query q returns road segments within dash line area. The inner point(s) is(are) the start location(s) specified by user and the solid circles indicates the bounding region of the query q. (a) a Single Location ST Reachability Query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

- **Trajectory Reachability.** Given a start location S, a road segment r_i in the road network, a start time T, and a duration L, the trajectory reachability reflects the fact if any of historical trajectories has traversed the given road segment from the start location within a given duration (i.e., from T to T + L). If the road segment r_i is reachable from L, the trajectory reachability is 1, otherwise the trajectory reachability for these locations is 0. For example, given the above constrains, if there was a trajectory passing S at t_1 and passing r_i at t_2 and $t_2 - t_1 < L$ holds, the trajectory reachability is 1.
- **Reachable Area.** Given a start location S, a start time T and a duration L, a reachable area is a set of road segments which contain all the road segments that trajectory reachability from S for each of them is 1.
- $Prob$**-Reachable Area.** The $Prob$-Reachable area is a more general description of the reachable area, where we introduce a reachable probability, which 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 were 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%.

B. Problem Definition

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 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 segment 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.

Extensions. We consider the aforementioned spatio-temporal reachability query as a building block upon which our framework can be extended to support more complex spatio-temporal reachability queries with multiple query locations, illustrated in Figure 2b, where we want to find the union area of the $prob$-reachable area of all the query locations.

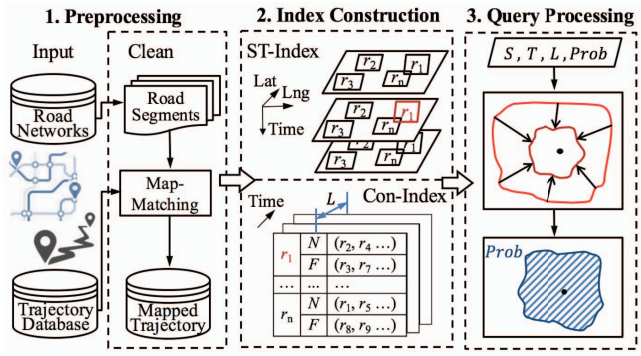


Fig. 3. **An overview of framework.** Take a Single Location ST Reachability Query q with S=$\{r_1\}$ as an example, we first find road segment r_1 at start timestamp T by ST-Index and then jump to other road segments according to Con-Index within duration L. Finally, we trace back search from maximum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

C. System Overview

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

- **Pre-processing.** This component performs two main tasks: 1) road re-segmentation and 2) trajectory map-matching. The objective of the road re-segmentation step is to improve the granularity of our reachability range. The pre-processing component re-segments the original road network based on the given spatial granularity (e.g., 500 meters). After that, the system reads the massive trajectory data from a database and maps the trajectory to the newly partitioned 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 $NearTable$ and upper bound range as $FarTable$, i.e., noted as N and F in the Figure. The connection index is used to prune the spatio-temporal reachability query process.
- **Query Processing.** This component processes queries from the user. This component employs two main techniques: 1) s-query maximum/minimum bounding region search, which uses our spatio-temporal index and connection index to 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. This component also has the ability to efficiently process the complex spatio-temporal

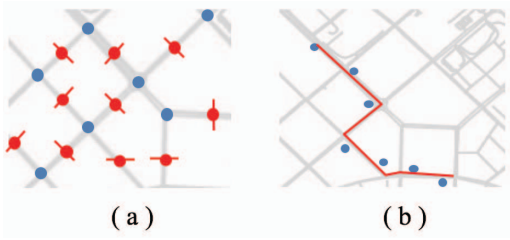


Fig. 4. **Pre-Processing.** (a) a glimpse of new road network after re-segments. New segment points are marked with ticks. (b) an example of map-matching. Red line respects a trajectory mapped to a route on the road network which connecting GPS points.

reachability query by minimizing the redundant searching space with multiple query locations.

III. Pre-Processing

In this section, we present the details in the *pre-processing* module. The objective of this module is to convert the raw trajectory data in to a set of map-matched trajectory data. There are two main steps as follows:

Road Re-segmentation. The road re-segmentation step partitions the original road segments based on a given spatial granularity (e.g., 500 meters). The main intuition behind this step is that, in the real road network data, there are many road segments with very large length value (e.g., some highways), and we want to avoid having such long road in our result set to improve the system effectiveness. After importing the whole road network, we re-segment original roads by combining all roads with junction information at first and then chopping roads into new segments which shows in Figure 4 (a). According to the given length, we add some new intersection points to create more road segments in the original long road segment.

Map-Matching. In this step, we map the raw trajectory data onto the newly segmented road network. We employed an existing method [24] to perform the task. Figure 4 (b) provides an example of map-matching part. At first, we map GPS points to corresponding road segments and then connect all road segments to make up the mapped trajectory. At the same time, we add the value of instant speed, car ID (considered as trajectory ID which connecting points into a trajectory) and timestamp into the corresponding road segment as its attributes. As a result, we acquire our cleaned trajectory database which includes both road network and trajectory information by mapping trajectories to road network. Note that one moving object only has one trajectory per day which is consisted of GPS points recorded at different timestamps.

IV. Index Construction

In this section, we introduce the details of our two index structures: 1) **S**patio-**T**emporal Index (ST-Index) and 2) **Con**nection Index (Con-Index).

A. Spatio-Temporal Index

ST-Index is used to speed up the process to find out the corresponding start road segment based on the query location.

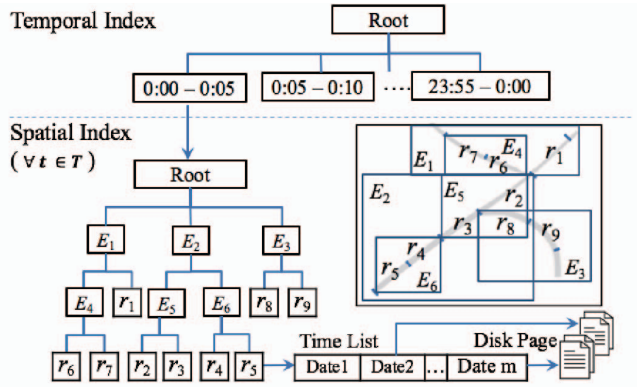


Fig. 5. **ST-Index.** The upper component is a temporal partition indicating the time line per day with the time interval of 5 minutes. Each time slot corresponds to a spatial partition illustrated in the bottom component. Each leaf node of the spatial index has a time list to identify the date of trajectories traversing its road segment.

The main difference in our spatio-temporal index is having two levels of temporal information embedded (i.e., time of the day and date) in order the calculate the $prob$-reachable area more efficiently. Therefore, ST-Index consists of 3 components: $Temporal\ index$, $Spatial\ index$ and $Time\ List$. Figure 5 illustrates the indexing structure of ST-Index.

Temporal index. To support finer granularity of the spatio-temporal reachability query, we split one day into several time slots. For example, if we want to support the query with 5 mins granularity in the Figure, we will divide the time with many 5-mins intervals. After that, we build a B-tree upon all the small temporal intervals to speed up the temporal range selection. In the each leaf node of the index, a spatial index is associated with it.

Spatial index. A spatial index (e.g., R-tree) is built based on the re-segmented road network. As the road network is static, essentially all the leaf nodes in the temporal index have the same spatial index structure. As a result, during query processing, we only need to access the same spatial index to find out the candidate road segments.

Time List. For each leaf node in the index, we maintain a time list. Each entry of the time list is identified based on the date. And all the trajectory IDs that passed this road segment during the corresponding date and time is stored as the content of this entry in the disk, as shown in the Figure. The main reason to keep this time list with trajectory date information is to speed up the $prob$-reachable area computation, as the system needs to identify trajectories to verify the reachability probability.

B. Connection Index

With the spatio-temporal index built as above, a naive solution to answer the spatio-temporal reachability query can be proposed as: we use the traditional network expansion algorithm, e.g., [17] to expand the road network from the query location and verifies each expanded road segments to see if it fulfills the reachability probability by reading the trajectory IDs from the disks. However, this query process can be prohibitively inefficient, as it has to access very frequently

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*

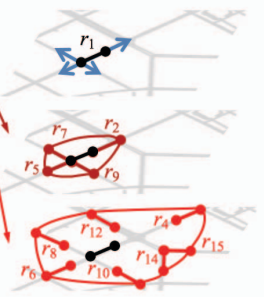


Fig. 6. **Con-Index.** The left table indicates a connection table in time slot t. The right figure depicts the road segments in the Near ID list and Far ID list of road segment r_1 on a real road network.

to the disk to retrieve the trajectory information.

To improve the system efficiency and avoid the 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) Near ID list (lower bound range) and 2) Far ID list (upper bound range) indicating the nearest (farthest) road segments that could be arrived at within the given time slot.

To build the connection table, we modified the conventional network expansion algorithm [17]. We generate Near ID list of each road segment by considering the minimum speed (removing the 0 speed) in all directions, after that we expand the road network using the networking expansion algorithm [17] with the temporal granularity. After that, all the reachable road segments in this process are added in the table as the Near ID list of the start road segment. The Far ID list is constructed in the similar way by using the maximum traveling speed calculated from the historical trajectories. Figure 6 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 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 you can see, it is obvious that the range of Far ID list is larger and extends to more intersections over road network.

V. Spatio-Temporal Reachability Query Processing

With the ST-Index and Con-Index, now we are in a position to introduce query processing algorithms to answer single- and multi-location ST reachability queries. Below, we refer the single-location (resp. multiple-location) ST reachability query as to s-query (resp. m-query) for simplicity.

A. Single-location ST Reachability Query (s-query)

For a single-location ST reachability query, i.e., s-query $q = (S, T, L, Prob)$,includes one query location specified as $S = \{s\}$, starting time T, a query duration L, and a probability $0 < Prob \leq 1$. We answer an s-query in two steps: (i) by checking the Con-Index, maximum bounding region is first extracted, that provides an upper bound of $Prob$-reachable region from (S, T) over a duration L; (ii) a trace back search algorithm is conducted to search the $Prob$-reachable regions from the maximum bounding regions. Below, we elaborate on the maximum bounding region search and trace back search algorithms for an s-query.

1) S-query Maximum Bounding Region Search: To answer an s-query $q = (S, T, L, Prob)$, the first step is to find a maximum bounding region, that the result of the s-query can possibly reach. As an upper bound, the maximum bounding region allows the process to quickly approach the query result, without exhaustively searching from the starting location $S = \{s\}$ of the query q. This can be done by checking ST-index and Con-Index as follows. First, with the start location $S = \{s\}$ and time stamp T from q, we identify the start road segment r_0 in the R-tree from ST-Index. Then, by checking the start road segment r_0 at time T, we can find the list of r_0's maximum reachable road segments from T, denoted as $F(r_0, T)$, in the next Δt time interval. Likewise, by checking each $r \in F(r_0, T)$ in Con-Index for their maximum reachable road segments $F(r, T+\Delta t)$ from a start time $T+\Delta t$ in a next Δt time interval, we can obtain a maximum reachable road segment set $F^2(r_0, T) = \cup_{r \in F(r_0,T)} F(r, T + \Delta t)$. We keep searching the Con-Index for k steps, until the time duration L is met, namely, $k\Delta t \leq L < (k+1)\Delta t$. The maximum reachable region is thus $F^k(r_0, T) = \cup_{r \in F^{k-1}(r_0,T)} F(r, T + (k-1)\Delta t)$. The detailed **S**-**Q**uery **M**aximum **B**ounding Region Search (SQMB) algorithm is summarized in Algorithm 1.

Algorithm 1 s-query maximum bounding region search (SQMB) algorithm

```
1: INPUT: s-query q = {S = {s}, T, L, Prob}.
2: OUTPUT: Maximum bounding region set B = {b_1, ..., b_m}.
3: Find road segment r_0 in ST-Index, with s ∈ r_0
4: Segment list R = {r_0}
5: for 0 ≤ ℓ ≤ L do
6:    for ∀r in R do
7:       Bounding set B = B ∪ F(r, T + ℓ).
8:    R = B
9:    ℓ = ℓ + Δt
10: return B
```

Line 3 identifies the starting road segment r_0 that the query location s resides on. Line 4 initiates the segment list as r_0. Starting from r_0, Line 5–9 search the maximum bounding region through Con-Index, and Line 10 returns the maximum bounding region B. Note that SQMB algorithm can also be naturally applied to find the minimum bounding region, by using the records for the nearest reachable region, in each Δt.

Illustration example. We show how SQMB algorithm works in a concrete example shown in Figure 7, which employs both ST-Index and Con-Index to determine the maximum bounding region of an s-query q. In Figure 7, the upper component shows all paths to locate the bounding regions through Con-Index, while the bottom component illustrates the same search paths across index nodes in ST-Index. From the starting road segment r_1, the query finds the first hop

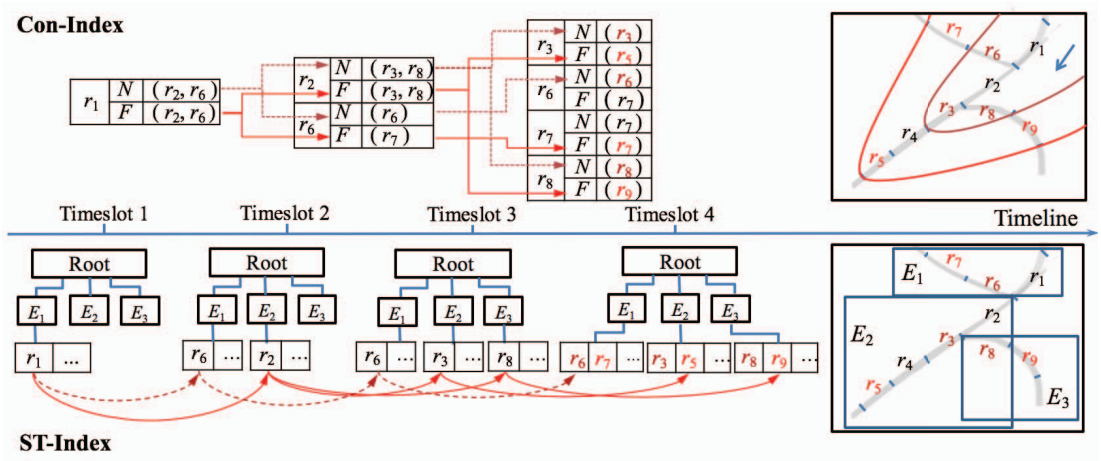


Fig. 7. **Maximum/Minimum bounding regions.** An illustrating example on s-query of how ST-Index and Con-Index are employed to determine the maximum and minimum bounding region of a query starting from road segment r_1. The upper component shows the searching paths in Con-Index across time slots. The subfigure on the right illustrate the bounding region from the start road segment r_1 on a real map, where the two solid lines represent the minimum/maximum bounding regions, respectively). The bottom component indicates the searching paths for the query bounding region across R-trees in different time slots.

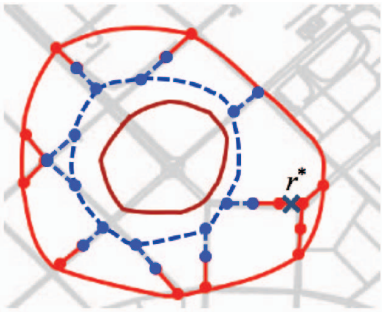


Fig. 8. **Trace Back Search.** Two solid circles indicate the maximum and minimum bounding regions, respectively. The dashed circle indicates the $Prob$-reachable region with respect to$Prob$. Trace back search starts from the (outer) maximum bounding region to inner minimum one.

(in one time slot) maximum bounding region of $\{r_2, r_6\}$, by checking the Con-Index in time slot 1. Then, by checking and merging the maximum bounding region of r_2 and r_6, a final maximum bounding region is obtained as $\{r_5, r_7, r_9\}$. Similarly, a minimum bounding region from r_1 can be found as $\{r_3, r_6, r_8\}$. The corresponding geographical location of each road segment is presented in the subfigures in Figure 7.

2) Trace Back Search: The maximum and minimum bounding regions provide a refined and smaller geographic region to further identify the exact $Prob$-reachable region of an s-query q. It guarantees that all road segments on $Prob$-reachable region 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. Firstly, by checking ST-Index, we extract the list of trajectory IDs from the starting 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 the trajectory dataset spans. The maximum bounding region B include a list of road segments. For each road segment $r \in B$, we check ST-Index to extract the list of trajectory IDs from the road segment r in time interval $T_B = [T, T + L]$ of each day d, represented as $Tr(r_0, T_B, d)$. Then, for each day $1 \leq d \leq m$, we check if r is reachable from r_0 on day d, by checking if there is some common trajectories in both $Tr(r_0, T_0, d)$ and $Tr(r_0, T_B, d)$ or not. Suppose that there are m^* out of m days where $Tr(r_0, T_0, d) \cap Tr(r_0, T_B, d) \neq \emptyset$ holds, then the reachable probability $probability(r, r_0)$ from r_0 to r during the period of $[T, T + L]$ is as follows, which represents from the historical statistics, the probability that road segment r is reachable from r_0 during the time interval $[T, T + L]$.

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

For a given $r \in B$, if $probability(r, r_0) \geq Prob$, 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, thus we add r's neighboring road segment set $neighbor(r)$ to the search space B for further investigation. Note that since we search from the maximum bounding region to the minimum bounding region, the neighboring road segments of r being 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.

The detailed **T**race **B**ack **S**earch (TBS) algorithm is summarized in Algorithm 2. Line 3 initializes the searching road segment set as the maximum bounding region B_{max}. Line 4–5 check if there is still road segments to be searched or not: the searching process terminates if B is empty, otherwise, the next road segment $r \in B$ is popped out. Line 6–9 examine if r is $Prob$-reachable from r_0 or not. If yes, r is added to $Prob$-reachable set, and it moves forward to search next road

Algorithm 2 Trace Back Search (TBS) algorithm

1: **INPUT:** Bounding set B_{max} and B_{min}, Probability $Prob$, stat road segment r_0.
2: **OUTPUT:** Bounding set B' with respect to $Prob$
3: $B \leftarrow B_{max}$
4: **while** $B \neq \emptyset$ **do**
5: $r \leftarrow dequeue(B)$
6: **if** $probability(r, r_0) \geq prob$ **then**
7: $Result \leftarrow \{r\} \cup Result$
8: **else**
9: $B \leftarrow (neighbor(r) - B_{min}) \cup B$
10: return $Result$

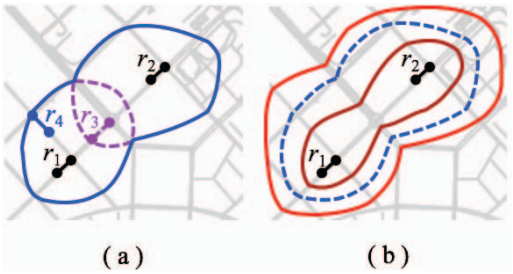


Fig. 9. **Multiple Query Bounding Regions.** Solid line indicates the outer bounding region which is the real maximum reachable boundary of both r_1 and r_2 while dashed line indicating inner bounding regions. Road segments r_1 and r_2 are the start locations from a m-query. Road segment r_3 is on the 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.

segment in B (if any); otherwise, we add r's neighboring road segments to B (if not yet overlapping with B_{min}) for further investigation. Line 10 terminates the TBS search and returns $Prob$-reachable region of r_0.

Illustration example. Figure 8 shows a concrete example on how TBS algorithm works to answer query q by searching from the maximum to minimum bounding regions. The solid road segments are with lower reachable probability than $Prob$ from q, and dashed road segments are with higher or equal reachable probability than $Prob$. Note that once a road segment has been searched, it will be marked as "visited", so that it will not be searched when being expanded from other road segments in B. To be precise, taking road segment r^* as an example in Figure 8, there are two paths traversing it from the maximum bounding region. However, once one of them has visited r^*, it will be marked as "visited". When the other path expands to r^*, trace back search algorithm will not add it again to B. Such mechanism ensures the efficiency of TBS algorithm and avoid duplicated searches.

B. Multi-location ST Reachability Query (m-query)

Going beyond s-query, which allows one single query location $S = \{s\}$, we now consider a ST reachability query with multiple starting locations, i.e., $S = \{s_1, \cdots, s_n\}$, referred to as multi-location ST reachability query, in short, m-query. A m-query is formally defined as $q = (S, T, L, Prob)$, with a set of querying n locations $S = \{s_1, \cdots, s_n\}$, starting time T, duration L, and a confidence probability $Prob$. The m-query q asks for the $Prob$-reachable region from any of the location $s \in S$ during the 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 m-query is thus the outer-most bounding regions of the union among all B_i's. Figure 9(a) shows an example of m-query with two starting road segments, r_1 and r_2. The solid lines outline $Prob$-reachable region of the m-query, which is the outer-most bounding region of the two single $Prob$-reachable regions of r_1 and r_2, where the overlapping parts (in dashed lines) are removed.

Naive solution. To solve an m-query, a naive (but always working) solution is treating an m-query as multiple s-queries, answer them one by one, and merge the $Prob$-reachable region of each s-query to obtain the $Prob$-reachable region for the m-query. However, the potential inefficiency of this approach is that when answering multiple s-queries, the road segments lying between the maximum and minimum bounding regions of different s-queries may be searched multiple times, due to the lack of communication among individual s-queries. When the number of locations in an m-query is large, say, tens to hundreds and the query duration L is long, i.e., 4 hours or more, the issue may lead to huge processing time. As a result, we are motivated to develop an m-query processing algorithm that can automatically take advantage of the overlapping information, to avoid duplicate search for road segments.

Query processing algorithm for m-query. The basic idea behind the query processing algorithm for m-query is still a two-step approach: (i) finding a unifying maximum and minimum bounding region of the m-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 m-query q. As shown in Figure 9(b), the maximum and minimum bounding regions are the outer-most boundary of the merged bounding regions across all single s-queries. We develop the m-query maximum/minimum bounding region search algorithm, which works as follows. First, we match each start location $s_i \in S$ to a start road segment $r_{0,i}$ from R-tree in ST-Index, forming a starting road segment set $R_0 = \{r_{0,1}, \cdots, r_{0,n}\}$. Then, we check each $r_{0,i} \in R_0$ in Con-Index and obtain a list of $r_{0,i}$'s maximum and minimum reachable road segments from T, denoted as $F(r_{0,i}, T)$, in the next Δt time interval. We denote the simple union set of all $F(r_{0,i}, T)$'s as $F(R_0, T) = \cup_{r \in R_0} F(r_{0,i}, T)$, which would include road segments in the overlapping regions of $F(r_{0,i}, T)$'s. Those road segments can be eliminated by the following rule: Given a road segment $r \in F(R_0, T)$, if the nearest road segment $r_s \in R_0$ to r, i.e., $r_s = argmin_{r' \in R_0}\{dis(r', r)\} \in R_0$ is the same as the one whose bounding region contains r, i.e., $r \in F(r_s, T)$, r should be included into the bounding region of R_0. Otherwise, r should be eliminated. To better understand the logic behind this, we look at r_3 Figure 9(a). r_3 has the shorter distance to the starting road segment r_1 than r_2, where r_3 is in the bounding region of r_2, thus r_3 is in the overlapping region, and should be eliminated. After this filtering processing, a unifying maximum bounding region of m-query q is obtained as $R(R_0, T)$, from start road segment set R_0, during time

interval $[T, T + \Delta t]$. Next, taking $R(R_0, T)$ as the starting road segment set, we can obtain $R^2(R_0, T)$, a maximum bounding region of m-query q, with starting road segment set R_0 during time interval $[T, T+2\Delta t]$. Keeping searching Con-Index for k steps, until the time duration L is met, namely, $k\Delta t \leq L < (k+1)\Delta t$. The maximum bounding region of R_0 is thus $R^k(R_0, T)$ with starting road segment set R_0 during time interval $[T, T + k\Delta t]$.

The detailed **M**-**Q**uery **M**aximum **B**ounding Region Search (MQMB) algorithm is summarized in Algorithm 3. Line 3 identifies the set of starting road segments $R = \{r_1, \cdots, r_n\}$ of the locations $S = \{s_1, \cdots, s_n\}$ in m-query q. Line 4 starts the loop of increasing the targeted time interval $[T, T + \ell]$ until it reaches the user-specified duration L. Line 5–6 simply construct the union set B of maximum bounding regions of all road segments in R. Line 7–10 remove road segments in overlapping regions, and construct a unifying maximum bounding region. Line 11–12 update the target road segments set R and time interval ℓ for next iteration. Line 13 returns the maximum bounding region $Result$.

Algorithm 3 m-query maximum bounding region search (MQMB) algorithm

1: **INPUT:** m-query $q = \{S = \{s_1, \cdots, s_n\}, T, L, Prob\}$.
2: **OUTPUT:** Maximum bounding region set $Result$.
3: Find starting road segment list R in *ST-Index*
4: **for** $0 \leq \ell \leq L$ **do**
5: **for** $\forall r$ in R **do**
6: Bounding set $B \leftarrow B \cup F(r, T + \ell)$.
7: **for** $\forall b$ in B **do**
8: $r_s = \textbf{argmin}_{r' \in R}\{dis(r', b)\}$
9: **if** $b \in F(r_s)$ **then**
10: $Result = Result \cup \{b\}$
11: $R = Result$
12: $\ell = \ell + \Delta t$
13: return $Result$

VI. EVALUATION

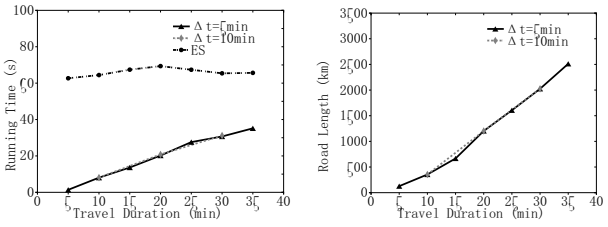
In this section, we conduct extensive experiments to evaluate our indexing structure and query processing algorithms for both s-query and m-query using a one-month taxi trajectory dataset from Shenzhen, China. For s-query, we compare our SQMB+TBS algorithm with exhaustive search method; for m-query, we compare our MQMB+TBS algorithm with SQMB+TBS algorithm. The extensive evaluation results demonstrate that our SQMB+TBS can on average reduce 50% running time than exhaustive search method, and our MQMB+TBS algorithm can reduce on average 30% running time over SQMB+TBS algorithm. Below, we elaborate on the dataset we used, experiment configurations, and experimental results.

A. 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 three million people. The dataset was collected for 30 days in November, 2014. These trajectories represent 21,385 unique taxis in Shenzhen. They are equipped with GPS sets, 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.

TABLE II
EVALUATION CONFIGURATION

duration L	$\{5, 10, \cdots, 35\}min$
probability $Prob$	$\{20\%, \cdots, 100\%\}$
start time T	$\{[00:00-00:05], \cdots, [23:55, 00:00]\}$
interval Δt	$\{1, 5, 10, 20\}min$
s-query algorithms	ES, SQMB+TBS
m-query algorithms	SQMB+TBS, MQMB+TBS



(a) Processing time (b) Reachable segment length

Fig. 10. Effect on duration L

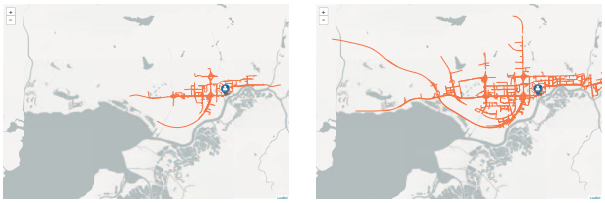
B. Single-Location ST Reachability Query

In the experiments, we evaluate our query processing method for s-query by changing different parameters, including, duration L (in minutes), probability of reachable areas $Prob$, starting time T, and time interval Δt (in minutes). The detailed experiment configurations are listed in Table II.

Baseline method. For s-query, we choose baseline algorithm as exhaustive search (ES) method, which starts from the querying location s and time T, to search the neighboring road segments through the road network. The searching process terminates until $Prob$-reachable road segments at all possible branches on the road network.

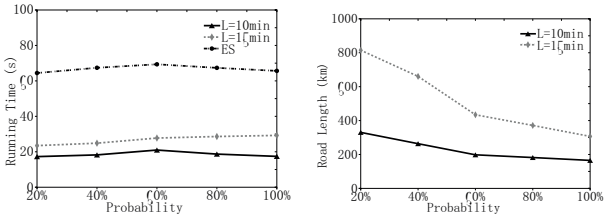
Evaluation metrics. We use two different metrics to measure the efficiency and effectiveness of our method, i.e., running time, and total length of covered road segments from the query result. Query processing running time is used to evaluate algorithm efficiency, which captures how long different algorithms take to process an s-query. On the other hand, we use the total length of all reachable road segments as the measurement of the effectiveness of our model. Moreover, we also employ visualizations of query results on road map to better understand and illustrate the performances of our proposed query processing algorithms.

1) Effect on Duration L: To evaluate the processing time of baseline method and our algorithm, Figure 10(a) shows query processing time as we increase the duration L. We set travel duration L from 5 to 35 mins with different time



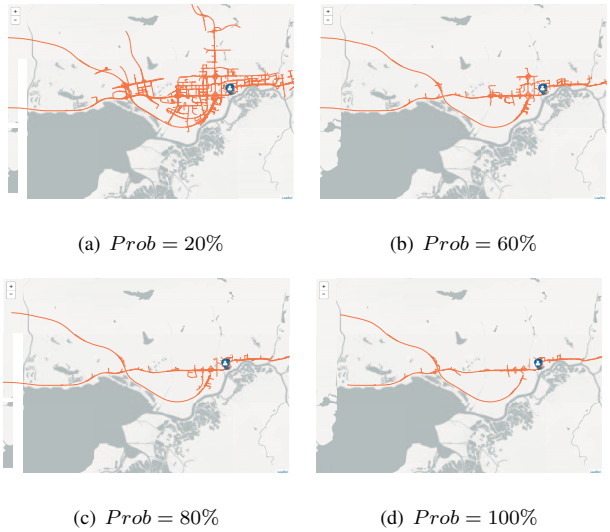
(a) $L = 5$ min, $Prob = 20\%$ (b) $L = 10$ min, $Prob = 20\%$

Fig. 11. Examples of $Prob$-reachable region



(a) Processing time (b) Reachable segment length

Fig. 12. Effect on query probability



(a) $Prob = 20\%$ (b) $Prob = 60\%$

(c) $Prob = 80\%$ (d) $Prob = 100\%$

Fig. 13. Results of $Prob = 20\%$, 60%, 80% and 100%

interval $\Delta t = 5$ mins and 10 mins, respectively. Moreover, for all queries, they share the same start time $T = 11$am, location $s = (22.5311, 114.0550)$ in latitude and longitude, and probability $Prob = 20\%$. ES method always traverses road segments from the starting location s in the road network, where our SQMB+TBS algorithm can skip the nearby region of the starting location, and start search from the road segments on the maximum bounding region. As shown in Figure 10(a), our SQMB+TBS algorithm always achieves much less running time than ES method. Moreover, the processing time of SQMB+TBS algorithm, increases as duration L increasing, because of the expansion on the maximum bounding region. Figure 10(a) shows that our algorithm can reduce up to 90% of processing time when users tend to query a small duration. Even with larger duration like 30 minutes, our algorithm can still reduce nearly 50% of processing time. In Figure 10(b), it shows the increase of road length with the increase of duration L when applying our algorithm. This is simply because longer duration allows to travel longer distance. Moreover, we observe that with different Δt, i.e., the two curves in the figures, the $Prob$-reachable road segment length does not change much. This is because the total length of $Prob$-reachable road segments is fixed given an s-query. Note that Δt as a granularity in the indexing structure does not have impact of the query result.

The Figure 11(a) and Figure 11(b) visualize our results on a road map, with all road segments that can be reached on at least 20% days in the historical data with $L = 5$ and 10 mins, respectively. Both figures show that the entire $Prob$-reachable region from the starting location s to the boundary road segments. Clearly, on the high-speed road segments, the region is further away from the starting location, while on the local low-speed roads, the query result region is smaller. This occurs because vehicles on highways in general travel faster than on those low speed roads. Moreover, long trips tend to take highway, while shorter trips tend to take local low-speed roads.

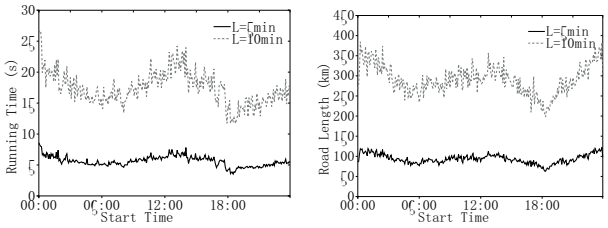
2) Effect on the probability Prob: Now, we fix the start time T, start location s, time granularity Δt and duration L to study how different query probabilities influence the performance of our query processing algorithm.

We use Figure 12(a) and Figure 12(b) to demonstrate the effect of query probability $Prob$ on both efficiency and effectiveness. Figure 12(a) shows that as we change $Prob \in \{20\%, 40\%, 60\%, 80\%, 100\%\}$, the running time is almost unchanged, which indicates that our SQMB+TBS algorithm employs Con-Index to directly find the maximum and minimum bounding region of the s-query, thus the query processing time does not hinge much over the querying probability. Moreover, the running time of SQMB+TBS is much smaller than ES method, which is reasonable, since our SQMB+TBS algorithm skips the searching process near the start location s, which significantly reduces the running time.

The Figure 12(b) shows the reachable road segment length with respect to different probabilities. We observe that as the query probability increases, the length of reachable road segment decreases, which is reasonable, since there will be less road segments qualified for the query with higher query probability.

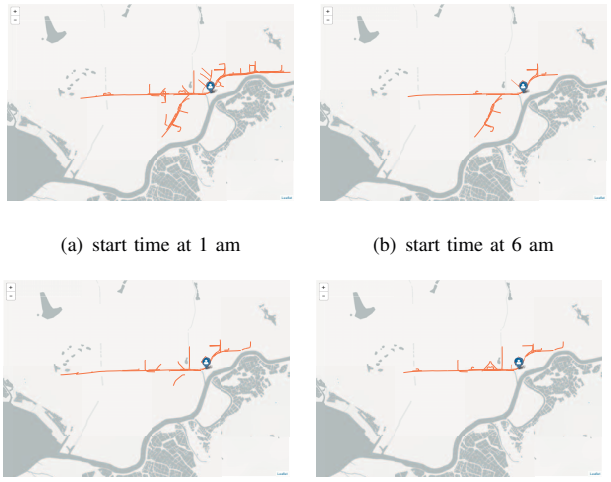
Figure 13 shows the visualization results on a road map, which gives us an idea of how reachable region look like. Figure 13(a), Figure 13(b), Figure 13(c) and Figure 13(d) illustrate the query results of probability = 20%,60%,80%,100%, respectively. As we increase the probability, the visualization of reachable region starts to shrink, especially, on those low-speed local roads. However, the overall reachable road network structure, formed by the highways remains the same.

3) Effect on start time T: In this section, we evaluate the performance of our SQMB+TBS algorithm over travel starting time T in a day, and we fix all other parameters. In Figure 14(a), we observe that the running time changes as



(a) Processing time (b) Reachable segments length

Fig. 14. Effect on start time



(a) start time at 1 am (b) start time at 6 am

(c) start at time 12 pm (d) start time at 6 pm

Fig. 15. Results of T = 1am, 6am, 12pm and 6pm

starting time varies. For example, at around 7am and 6pm, the running time drops significantly, which primarily because of the effect of rush hours. The traffic condition goes down during these rush hours, which leads to smaller reachable regions at these starting time, thus with less running time to process the query. To be precise, in Con-Index, slower maximum speed leads to smaller maximum bounding region, which covers less road segments candidates, which in turn takes less running time. This phenomenon can also be verified using reachable road segment length in Figure 14(b), where the total length of reachable road segments exhibits similar pattern with running time.

To better illustrate the effect of starting time, we visualize the reachable road segments with 80% probability, $L = 5$ mins and start time as 1 am, 6 am, 12 pm and 7 pm, respectively. Clearly, Figure 15(d) shows the results of start time at 6 pm has the smallest reachable region. Moreover, the changes of reachable regions are more on local low-speed roads, and the highway reachable regions are relatively stable over time.

4) Effect on time interval Δt: In this section, we evaluate how the time interval Δt affects the running time of our SQMB+TBS algorithm. From Figure 16, we observe that the running time does not change much as the time interval varies, when other parameters are fixed, such as start time, location and probability. This indicates that our SQMB+TBS algorithm is stable on the system parameter, Δt, that governs the time granularity in the indexing structure.

C. Multi-location ST Reachability Query Processing

For multi-location ST reachability query, in short m-query, we compare our MQMB+TBS algorithm with running single-location query algorithm SQMB+TBS multiple times. To use SQMB+TBS algorithm to process m-query, we treat the m-query with n query locations as n s-queries, where each s-query has a unique query location. The SQMB+TBS algorithm is then applied to process each s-query, and the final m-query results comes from the union of all results of each s-query.

In Figure 17, we execute an m-query with three locations and we set probability as 20%. The result shows that it is roughly three times slower than processing the m-query using SQMB+TBS algorithm on each individual s-queries. Moreover, we also examine the performance of MQMB+TBS vs SQMB+TBS, with changing the number of query locations in the m-query. We set start time as 10 am, duration as 20 minutes and probability as 20%. Figure 18 indicates that when we use MQMB+TBS algorithm to process an s-query, with only one query location, it is slightly slower than normal SQMB+TBS algorithm, which is reasonable because that MQMB+TBS algorithm has an extra filtering stage trying to eliminate the road segments in the overlapping region among different query locations. However, when the number of locations increases (say, more than one), the running time MQMB+TBS algorithm takes to process m-query is almost constant but processing time of SQMB+TBS algorithm incearses linearly to number of locations. When visualizing our results on road maps in Figure 19, we observe that the reachable region of all three locations is the union of reachable areas of three individual s-queries.

VII. 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-temperal data management, (2) trajectory query processing, and (3) reachability query processing.

A. Spatio-temperal data management

With the rapid development of sensor technologies like satellites, GPS, 4G networks and Internet of Thing, we can collect a massive scale of spatio-temporal data like real-time rode condition, trajectories and point-of-interest status. 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], [4], [17], [28]. Then, B-tree and R-tree are combined together to store both spatial and temporal information [10], [11]. In these structures, an R-Tree is maintained for each leaf node of B-tree, which could take a huge space to store them 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

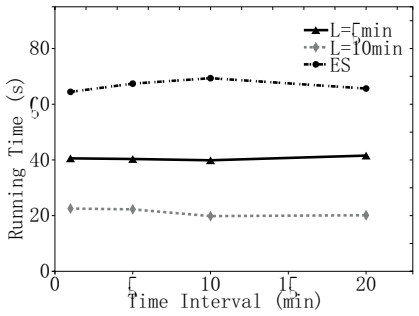


Fig. 16. Processing time over different time intervals

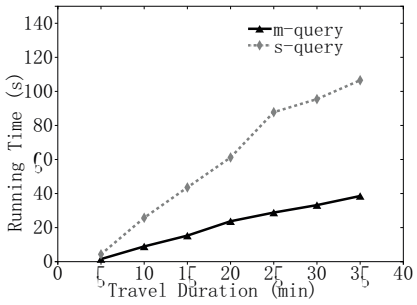


Fig. 17. Comparision of s-query and m-Query over duration

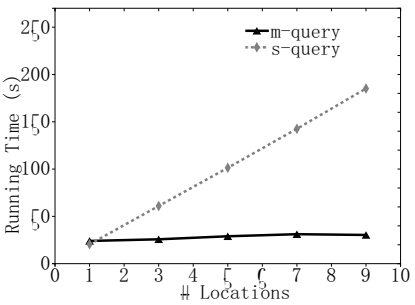
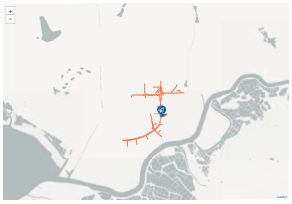


Fig. 18. Comparision of s-query and m-query over nubmer of locations

(a) Result of all 3 locations

(b) Result of location A

(c) Result of location B



(d) Result of location C

Fig. 19. Results of ST Reachability of three locations

have been developed to compress the index structure into a reasonable size [3], [10], [19]. Another approach to store and index spatio-temperal data is to utilize its network structure. The connectivity of road segment could be maintained using adjacency matrix or adjacent list [17]. Predictive Tree has been proposed to maintain the reachability of road segments using additional information such as road length [13]. 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 (SETI) [7]. 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.

B. Trajectory query processing

With large amount of trajectory data generated over time, it is increasingly challenging to answer various trajectory queries in different application scenarios in urban computing [26], [27]. 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 [14]. Moreover, the transformed Minkowski Sum has been used to answer such range queries, if the query input is a circular region [25]. In [16], 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 [20]. 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 [2]. Moreover, researchers have extensively studied trip planning problems, that seek for a best path passing through a set of distinct objects [15]. 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.

C. 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 [8], [9], [21]. Moreover, 2-hop labeling schema are introduced to solve the query and some heuristic methods are proposed to reduce the size of labels [5], [6]. The methods [18], [22] 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, thus these approaches apply to find the reachabilities on road networks. However, all these methods are designed over static graphs [23], [29]. However, the traditional graph reachability query mainly focuses on the static graph structure rather than dynamic data associated with the graph structure like trajectories. Differing from these queries, our proposed spatio-temporal reachability query results hinge on querying time.

VIII. Conclusion

In this paper, we investigate a problem of mining spatio-temporal reachable regions from a user-specified location within a given time period, which has a wide range of applications in reality, including location based recommendation, advertising, etc. To solve such a trajectory mining problem efficiently over massive trajectory data, we develop a novel indexing and query processing framework. First of all, to capture the temporal connection information between road segments over time, we introduce both spatio-temporal index and connection index to index the trajectory data. Utilizing these two indexing structures, we further develop algorithms to quickly locate the maximum and minimum bounding regions of the query results, and introduce a trace back search algorithm to find the exact reachable region of a query. Finally, we evaluate our indexing structure and query processing algorithms using a large-scale taxi trajectory dataset (with 194 GB size) collected from Shenzhen, China. Extensive experiments demonstrate that our query processing framework can reduce 50%–90% running time to answer spatio-temporal reachability query over baseline algorithms.

IX. Acknowledgement

Guojun Wu and Yanhua Li were supported in part by NSF CRII grant CNS-1657350 and a research grant from Pitney Bowes Inc. Yu Zheng was supported by the National Natural Science Foundation of China (Grant No. 61672399). Jun Luo was supported by NSFC Grant 11271351.

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 *2010 Eleventh International Conference on Mobile Data Management*, pages 115–124. IEEE, 2010.

[2] H. Bast, S. Funke, and D. Matijević. Transit: ultrafast shortest-path queries with linear-time preprocessing. In *9th DIMACS Implementation Challenge—Shortest Path*, 2006.

[3] B. Becker, S. Gschwind, T. Ohler, B. Seeger, and P. Widmayer. An asymptotically optimal multiversion b-tree. *The VLDB JournalThe International Journal on Very Large Data Bases*, 5(4):264–275, 1996.

[4] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The r*-tree: an efficient and robust access method for points and rectangles. In *ACM SIGMOD Record*, volume 19, pages 322–331. Acm, 1990.

[5] R. Bramandia, B. Choi, and W. K. Ng. On incremental maintenance of 2-hop labeling of graphs. In *Proceedings of the 17th international conference on World Wide Web*, pages 845–854. ACM, 2008.

[6] J. Cai and C. K. Poon. Path-hop: efficiently indexing large graphs for reachability queries. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 119–128. ACM, 2010.

[7] V. P. Chakka, A. C. Everspaugh, and J. M. Patel. Indexing large trajectory data sets with seti. *Ann Arbor*, 1001(48109-2122):12, 2003.

[8] Y. Chen and Y. Chen. An efficient algorithm for answering graph reachability queries. In *2008 IEEE 24th International Conference on Data Engineering*, pages 893–902. IEEE, 2008.

[9] Y. Chen and Y. Chen. Decomposing dags into spanning trees: A new way to compress transitive closures. In *2011 IEEE 27th International Conference on Data Engineering*, pages 1007–1018. IEEE, 2011.

[10] V. T. De Almeida and R. H. Güting. Indexing the trajectories of moving objects in networks. *GeoInformatica*, 9(1):33–60, 2005.

[11] R. H. Güting, T. de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *The VLDB JournalThe International Journal on Very Large Data Bases*, 15(2):165–190, 2006.

[12] A. Guttman. *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[13] A. M. Hendawi, J. Bao, M. F. Mokbel, and M. Ali. Predictive tree: An efficient index for predictive queries on road networks. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1215–1226. IEEE, 2015.

[14] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Path prediction and predictive range querying in road network databases. *The VLDB Journal*, 19(4):585–602, 2010.

[15] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In *International Symposium on Spatial and Temporal Databases*, pages 273–290. Springer, 2005.

[16] Y. Li, C.-Y. Chow, K. Deng, M. Yuan, J. Zeng, J.-D. Zhang, Q. Yang, and Z.-L. Zhang. Sampling big trajectory data. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 941–950. ACM, 2015.

[17] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 802–813. VLDB Endowment, 2003.

[18] S. Seufert, A. Anand, S. Bedathur, and G. Weikum. Ferrari: Flexible and efficient reachability range assignment for graph indexing. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1009–1020. IEEE, 2013.

[19] Y. Tao, D. Papadias, and J. Sun. The tpr*-tree: an optimized spatio-temporal access method for predictive queries. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 790–801. VLDB Endowment, 2003.

[20] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

[21] S. J. van Schaik and O. de Moor. A memory efficient reachability data structure through bit vector compression. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 913–924. ACM, 2011.

[22] 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 *EDBT*, pages 511–522. Citeseer, 2014.

[23] H. Wu, Y. Huang, J. Cheng, J. Li, and Y. Ke. Reachability and time-based path queries in temporal graphs. In *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*, pages 145–156. IEEE, 2016.

[24] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. An interactive-voting based map matching algorithm. In *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, pages 43–52. IEEE Computer Society, 2010.

[25] R. Zhang, H. Jagadish, B. T. Dai, and K. Ramamohanarao. Optimized algorithms for predictive range and knn queries on moving objects. *Information Systems*, 35(8):911–932, 2010.

[26] Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[27] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[28] R. Zhong, G. Li, K.-L. Tan, L. Zhou, and Z. Gong. G-tree: An efficient and scalable index for spatial search on road networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2175–2189, 2015.

[29] A. D. Zhu, W. Lin, S. Wang, and X. Xiao. Reachability queries on large dynamic graphs: a total order approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1323–1334. ACM, 2014.