Probabilistic Shortest Time Queries Over Uncertain Road Networks

Yaqing Chen
University of Texas-Pan American

PROBABILISTIC SHORTEST TIME QUERIES OVER UNCERTAIN ROAD NETWORKS

A Thesis

by

YAQING CHEN

Submitted to the Graduate School of
The University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2014

Major Subject: Computer Science

PROBABILISTIC SHORTEST TIME QUERIES

OVER UNCERTAIN ROAD NETWORKS

A Thesis
by
YAQING CHEN

COMMITTEE MEMBERS

Dr. Xiang Lian
Chair of Committee

Dr. Zhixiang Chen
Committee Member

Dr. Christine Reilly
Committee Member

May 2014

ABSTRACT

Yaqing Chen, Probabilistic Shortest Time Queries Over Uncertain Road Networks. Master of Science (MS), May, 2014, 43 pp., 6 tables, 9 figures, 26 references, 8 titles.

In many real applications such as location-based services (LBS), map utilities, trip planning, and transportation systems, it is very useful and important to provide query services over spatial road networks. Nowadays we can easily obtain rich traffic information such as the speeds of vehicles on roads. However, due to the inaccuracy of devices or integration in consistencies, the traffic data (i.e., speeds) are often imprecise and uncertain.

In this paper, we model road networks by uncertain graphs, which contain edges that are associated with probabilistic velocities. We formalize the problem of probabilistic shortest time query, and we propose time bound pruning and probabilistic bound pruning to filter out false alarms. Moreover, we design offline pre-computation to facilitate PSTQ processing.

DEDICATION

The completion of my master degree would not have been possible without the love and support of my family. My mother, my father and my brother, wholeheartedly inspired, motivated and supported me by all means to accomplish this degree. Thank you for your love and patience.

ACKNOWLEDGEMENTS

I will always be grateful to Dr. Xiang Lian, chair of my thesis committee, for all his mentoring and advice. From research design, and experiment processing, to manuscript editing, he encouraged me to complete this process through his infinite patience and guidance. My thanks go to my dissertation committee members: Dr. Zhixiang Chen, and Dr. Christine Reilly. Their advice and comments on my thesis helped me to ensure the quality of my intellectual work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

Recently, with the proliferation of geo-positioning techniques and GPS-enabled mobile devices, query processing over (spatial) road networks has become increasingly important in many real applications, such as location-based services (LBS), map utilities, trip planning, and transportation systems. Specifically, on road networks, there are many facilities such as restaurants, hotels, shopping malls, and so on, which may raise many interesting and useful queries.

Figure 1 shows an example of a road network (graph), which contains 11 nodes (intersection points of roads), $n_1 \sim n_{11}$, denoted as white circles, and edges (road segments), $e_{i,j}$, between two nodes $n_i$and n_j($1 \leq i, j \leq 11$). On road segments (e.g., $e_{5,6}$), there are some facilities, o_1~o_5, denoted as solid circles, that correspond to airport, hotel,or restaurant. In map applications, one classical yet important query might be "*retrieve a path with the shortest distance between two facilities, say airport and hotel*", or "*given the current location q of a query issuer, obtain a driving path to its nearest restaurant*".

While traditional queries over road networks only consider the traveling distances (e.g., the shortest path queries), a path with the shortest network distance may not be the one with the shortest traveling time. This is because some roads might be busy or have traffic jams, and the path that passes by these roads may take very long time, even if it has the shortest traveling distance. Thus, in some urgent scenario (e.g., catching up with the deadline), the traveling time is be one of the most desirable measurements from the query issuer's point of view.

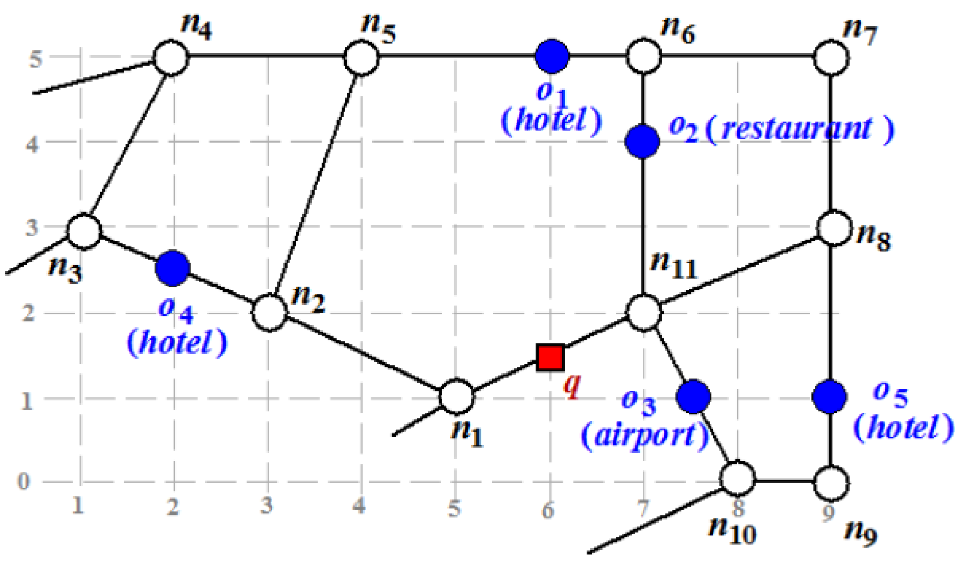


Figure 1: Illustration of uncertain road network model.

Nowadays, with the newly emerging technologies such as roadside sensors, GPS-equipped mobile devices/vehicles, and speed testing equipment, more and more traffic data (e.g., velocity samples) have been collected from road networks (e.g., via crowd sourcing). Such traffic information has greatly enriched the context of road networks by incorporating real-time traffic conditions. As a result, many road-network applications can now utilize actual traffic data to provide robust and reliable services, for example, route planning or map services by taking into account the traveling time.

Note, however, that traffic data collected from GPS or sensing devices can be imprecise and uncertain, due to the imperfect nature of devices or inherent uncertainty of vehicle velocities. For example, different vehicles may have different speeds on the same roads, and the speed of even the same vehicle may be time-varying. Thus, the collected speeds or traveling times of vehicles may not be accurate and fully reflect the actual traffic conditions on roads. Inspired by this, in this paper, we model such road networks by probabilistic graphs, over which each edge is associated with uncertain traveling time (represented by traveling time samples on the road, each with an appearance probability). This model can capture the realistic scenario of road networks.

In this paper, we will study an important query, *probabilistic shortest time query* (PSTQ), in the context of uncertain road networks which retrieves those facilities on road networks with the shortest traveling time from a given query point and with high confidence. Intuitively, the PSTQ problem can obtain probabilistic paths from the query point to facilities that have smaller traveling time than all other facilities with high probability. Different from prior works on nearest neighbor queries over certain road networks, our PSTQ problem considers the uncertainty of the traveling times (rather than certain traveling distance/time) in uncertain road networks (instead of certain road networks). Therefore, it is more challenging to answer PSTQ efficiently and effectively.

In particular, the challenges of answering PSTQs are as follows. First, due to the uncertainty of road networks, we usually consider *possible worlds* semantics following the literature of probabilistic databases [4, 24], where each possible world is a deterministic road network (with a certain traveling time on each road) that may appear in the real world. Since the number of possible worlds for uncertain road networks is exponential, it is not efficient or even feasible to materialize all possible worlds. Thus, it is challenging to efficiently retrieve PSTQ answers. Second, the PSTQ problem involves the graph structure, in which the time complexity of calculating the shortest traveling time of paths is high. Therefore, it also requires designing efficient solution to speed up the query efficiency.

In order to tackle the challenges mentioned above, in this paper, we design effective pruning strategies, *time bound pruning* and *probabilistic bound pruning*, to reduce the PSTQ search space. Moreover, to further improve the query performance, we also propose a novel pruning technique, namely *(probabilistic) time Voronoi pruning*, by utilizing an offline pre-computed *(probabilistic) time Voronoi* over uncertain road networks. Based on offline pre-computations, we construct an index, and propose efficient PSTQ processing approach to query over the index.

Specifically, we make the following contributions in this paper.

1. We model the road networks by uncertain graphs with uncertain traffic conditions, and propose the problem of *probabilistic shortest time query* (PSTQ) over uncertain road networks in Section 2.
2. We carefully design effective pruning methods, *time bound pruning* and*probabilistic bound pruning*, to filter out false alarms of PSTQ answers in Section 3.
3. We propose an offline pre-computation technique that utilizes *(probabilistic) time Voronoi* to prune PSTQ candidates in Section 4.
4. We design an index to organize the pre-computed data, and facilitate the PSTQ

pruning. We illustrate efficient PSTQ query procedure by traversing the index in Section 5.

5. We demonstrate through extensive experiments the efficiency and effectiveness of our proposed PSTQ approaches in Section 6.

In addition, Section 7 reviews previous works on query processing over certain/uncertain road networks. Finally, Section 8 concludes this paper.

CHAPTER II

PROBLEM DEFINITION

Uncertainty Model

Data Model of Uncertain Road Networks

In this subsection, we describe the data model of road networks with uncertain traveling time. Currently, there are many real-word applications that involve road networks. For example, online map services have recently become very popular and are frequently used to query the shortest distance or traveling time between the source and destination, where traffic data can be obtained from roadside sensors or cell phone signal analysis on a daily basis. However, this online service cannot always provide accurate real-time traffic data. For instance, different vehicles may have different speeds even on the same road segments; the velocities of vehicles may also change over time. Thus, the traveling time along each road segment derived from sensors or cell phone signals is often imprecise and uncertain. Therefore, we will model the road network by a probabilistic graph, in which each edge is associated with uncertain traveling time. To capture the uncertainty, we assume that the traveling time on each edge is a random variable, which follows some probabilistic distribution. Such a distribution can be represented by either discrete sample or a continuous *probability destiny function* (pdf).

Definition 1 (Uncertain Road Networks). An uncertain road network, RN, is defined as an uncertain graph $G = (V, E, \phi)$, where V contains a set of vertices(nodes) n_1, n_2, . . . , and $n_{|V|}$, each n_i ($1 \leq i \leq |V|$) residing at a 2D location ($x(n_i)$,$y(n_i)$), E is a set of edges, each $e_{i,j} \in E$ associated with a velocity variable$Z(e_{i,j})$, and ϕ is a mapping from $V \times V$ to E.

In the definition above, we assume that the velocity variable$Z(e_{i,j})$ on each edge $e_{i,j}$ has a pdf function, which can be represented by either discrete samples or a continuous pdf function. In this paper, we use discrete samples to represent the distribution of velocity variables $Z(e_{i,j})$, where each sample, $s_{i,j}$, isassociated with an appearance probability $s_{i,j}.p\left(\sum_{\forall s_{i,j}} s_{i,j}.p = 1\right)$.

Example 1. In the example of Figure 1, nodes n_1, n_2,…, and n_{11} are intersection points of road segments on the road network. Table 1 depicts the location of each node n_i (for 1 $\leq$i $\leq$ 11), which is described by 2D coordinates. Moreover, $e_{i,j}$ is an edgefrom node n_i to n_j, whose distance is denoted by $dist(n_i, n_j)$. Table 2 shows traveling distances and uncertain velocities on each edge, where each edge $e_{i,j}$ is associated with a probability density function, pdf($e_{i,j}$), of the velocity variable.

The Traveling Time of a Probabilistic Path

The traveling time on a road segment $e_{i,j}$ is often determined by the edge length and uncertain speeds of vehicles on it. In the sequel, we formally define uncertain traveling time of a path on road networks.

Definition 2 (The Traveling Time of a Path on UncertainRoad Networks). Given an uncertain road network RN, and two points x and y on RN, we denote the traveling time between two points, x and y, on uncertain road networks as $t(x, y) \in [t^{-}(x, y), t^{+}(x, y)]$, where $t^{-}(x, y)$ and $t^{+}(x, y)$ are minimum and maximum possible traveling time on the shortest path, $path(x, y)$, between points x and y on RN.

Intuitively, the traveling time $t(o_i, o_j)$ of each edge (road) $e_{i,j}$ on the path $path(x, y)$ is given by $dist(o_i, o_j)/v(o_i, o_j)$, where $dist(x, y)$ is the length of edge $e_{i,j}$ and $v(o_i, o_j)$ is the velocity variable. Since the vehicle velocity, $v(o_i, o_j)$, of edge $e_{i,j}$ is a variable (following probability density functions pdf ($e_{i,j}$)), the traveling time $t(o_i, o_j)$is therefore uncertain. Thus, in Definition 2, the traveling time of the entire path $path(\mathrm{x}, y)$ is given by summing up traveling times of edges on $path(\mathrm{x}, \mathrm{y})$, which is also a variable within an interval $[t^{-}(\mathrm{x}, \mathrm{y}), t^{+}(\mathrm{x}, \mathrm{y})]$.

In this paper, we use discrete samples of the traveling time (i.e., edge length divided by velocity samples) to represent the distribution of the traveling time variable. We assume that samples of the traveling time are collected independently on different road segments. Moreover, the case of correlated time samples on adjacent roads can be modeled by considering the joint probabilities of traveling time variables. We would like to leave this interesting topic (i.e., with correlated time samples on consecutive roads) as our future work.

Facilities on Uncertain Road Networks

Next, we give the definition of facilities (a.k.a. *points of interest*, or POI) on uncertain road networks below.

Definition 3 (Facility). Given an uncertain road network RN, facilities on RN are denoted by objects o_1, o_2, . . . , and o_n,which have attributes, including facility types and 2-dimension allocations $(\mathrm{x}(\mathrm{o}_i), \mathrm{y}(\mathrm{o}_i))$.

In a road network, there are different types of facilities such as hotels, restaurants and airports. Thus, each facility, o_i, as definedin Definition 3, has its type attribute. Moreover, facilities usually reside on edges, which also have their 2D coordinates.

Example 2. Table 3 shows 5 facility points, o_1 (hotel), o_2 (restaurant), o_3 (airport),o_4 (hotel), and o_5 (hotel) in the road network of Figure 1. Each facility o_i ($1 \leq \mathrm{i} \leq 5$) has its own location on the road network and its associated edge. For example, hotel o_1 is at position (6, 5) on road segment $\mathrm{e}_{5,6}$, restaurant o_2 is at position (7, 4) on road segment $\mathrm{e}_{6,11}$, and so on.

For the ease of illustration, in the following discussions, we will focus on uncertain road networks with one type of facilities (e.g., hotel). The case of multiple types of facilities can be easily extended by considering different facility types separately.

Probabilistic Shortest Time Queries

In this subsection, we formally define the probabilistic shortest time query (PSTQ) on uncertain road networks.

Definition 4 *(Probabilistic Shortest Time Query, PSTQ). Given an uncertain road network RN, a set of facilities* o_1, o_2,. . . , *and* o_n *on RN, a query point* q, *and a probabilistic threshold* α, *a PSTQ returns a set of objects* o_i *that have the shortest time to* q *with probability,* $\Pr_{PSTQ}(q, o_i)$, *greater than* α, *that is,*

$$\Pr_{PSTQ}(q, o_j) = \sum_{\forall T}\left(Pr\{t(q, o_i) = T\} \cdot \prod_{\forall o_j \neq o_i}\left(1 - Pr\{t(q, o_j) \leq T\}\right)\right) > \alpha \quad (1)$$

In order to find the query result with the shortest time to query point q, we calculate the traveling time, $t(q, o_i)$, from q to o_i, and its probability for each possible value T of the traveling time. The probability that o_i is the PSTQ answer is determined by the following condition: the probability that the time cost $t(q, o_i)$ equals T and the time costs $t(q, o_j)$ of any other objects are higher than T should be greater than threshold α. Assuming independent velocities on connecting edges, the PSTQ probability, $\Pr_{PSTQ}(q, o_i)$, is given by summing up probabilities that $t(q, o_i)$= T and $t(q, o_i) > T$ over all possible values of T.

Example 3. In the example of Figure 1, given a query point *q* at location (6, 1.5) on uncertain road networks and a probabilistic threshold α= 80%, the PSTQ query retrieves those objects (hotels) from $S = \{o_1, o_4, o_5\}$that have the smallest traveling times with probability greater than α. We compute the traveling time t and its probability p from point *q* to o_1, o_4, and o_5, the traveling time and its probability on each edge are shown in Table 4. For example, $t(q, n_{11})$ records 3 possible values of the traveling time on the first edge $e_{q,n_{11}}$ of the path from *q* to o_1, and p represents their probabilities.

As shown in Table 5, the traveling time interval of $t(q, o_1)$ from*q* to $o_1$is [0.134, 0.152], the time interval of $\mathrm{t}(q, o_4)$ from q to o_4 is [0.112, 0.131], and the traveling time $\mathrm{t}(q, o_5)$ from q to o_5 is within interval [0.122, 0.137]. Based on the Eq. (1) in Definition4, we have $\mathrm{Pr}_{\mathrm{PSTQ}}(q, o_1) = 0$, $\mathrm{Pr}_{\mathrm{PSTQ}}(q, o_4) = 87.557\%$, and $\mathrm{Pr}_{\mathrm{PSTQ}}(q, o_5) = 7.706\%$. Object o_4 has a high probability (greater than 80%) of having the shortest time to query object q, among all traveling times of the three objects. Thus, object o_4 is one of our PSTQ query answers.

Challenges

In a large road network, one straightforward method of answering the PSTQ problem is to compute the PSTQ probabilities above α as query answers. This method, however, is rather time consuming. Specifically, it is rather costly to compute the PSTQ probability, $\mathrm{Pr}_{\mathrm{PSTQ}}(\mathrm{q}, \mathrm{o}_\mathrm{i})$, in Eq. (1). This is because, it involves all objects in road networks (and has to consider exponential number of possible worlds [4]), which is very inefficient. Moreover, the cost of computing probabilities by traversing the graph is also high. Thus, in order to efficiently answer PSTQs, we should design effective pruning methods to reduce the search space, and propose efficient query processing approaches to retrieve PSTQ results. Table 6 summarizes the commonly used symbols and their descriptions in this paper.

Node	Location	Node	Location
n_1	(5, 1)	n_7	(9, 5)
n_2	(3, 2)	n_8	(9, 3)
n_3	(1, 3)	n_9	(9, 0)
n_4	(2, 5)	n_{10}	(8, 0)
n_5	(4, 5)	n_{11}	(7, 2)
n_6	(7, 5)		

Table 1: Nodes of uncertain road networks in Figure 1.

Edge $e_{i,j}$	**Distance** $dist(n_i, n_j)$	**Velocity** $pdf(e_{ij})$
$e_{1,2}$	$\sqrt{5}$	(30, 0.5) (35, 0.3) (36, 0.2)
$e_{2,3}$	$\sqrt{5}$	(45, 0.6) (48, 0.2) (50, 0.2)
$e_{3,4}$	$\sqrt{5}$	(50, 0.4) (52, 0.3) (55, 0.3)
$e_{4,5}$	2	(45, 0.3) (46, 0.3) (50, 0.4)
$e_{2,5}$	$\sqrt{10}$	(55, 0.6) (58, 0.3) (60, 0.1)
$e_{5,6}$	3	(47, 0.2) (48, 0.3) (50, 0.5)
$e_{6,7}$	2	(48, 0.3) (49, 0.3) (50, 0.4)
$e_{6,11}$	3	(30, 0.7) (32, 0.2) (35, 0.1)
$e_{7,8}$	2	(50, 0.3) (51, 0.3) (52, 0.4)
$e_{1,11}$	$\sqrt{5}$	(36, 0.1) (38, 0.4) (40, 0.5)
$e_{8,9}$	3	(40, 0.6) (42, 0.3) (45, 0.1)
$e_{9,10}$	1	(40, 0.8) (42, 0.1) (45, 0.1)
$e_{10,11}$	$\sqrt{5}$	(55, 0.3) (58, 0.3) (60, 0.4)
$e_{8,11}$	$\sqrt{5}$	(40, 0.4) (42, 0.3) (45, 0.3)

Table 2: Edges of uncertain road networks in Figure 1.

Facility	**Position**	**Edge**
o_1(hotel)	$(6, 5)$	$e_{5,6}$
o_2(restaurant)	$(7, 4)$	$e_{6,11}$
o_3(airport)	$(7.5, 1)$	$e_{10,11}$
o_4(hotel)	$(2, 2.5)$	$e_{2,3}$
o_5(hotel)	$(9, 1)$	$e_{8,9}$

Table 3: Facilities on uncertain road networks of Figure 1.

Path	$t(q, n_{11})$	p	$t(n_{11}, n_6)$	p	$t(n_6, o_1)$	p
$path(q, o_1)$	0.031	0.1	0.100	0.7	0.021	0.2
	0.029	0.4	0.094	0.2	0.021	0.3
	0.028	0.5	0.086	0.1	0.020	0.5
	$t(q, n_1)$	p	$t(n_1, n_2)$	p	$t(n_2, o_4)$	p
$path(q, o_4)$	0.031	0.1	0.075	0.5	0.025	0.6
	0.029	0.4	0.064	0.3	0.023	0.2
	0.028	0.5	0.062	0.2	0.022	0.2
	$t(q, n_{11})$	p	$t(n_{11}, n_8)$	p	$t(n_8, o_5)$	p
$path(q, o_5)$	0.031	0.1	0.056	0.4	0.050	0.6
	0.029	0.4	0.053	0.3	0.048	0.3
	0.028	0.5	0.050	0.3	0.044	0.1

Table 4: Possible traveling times of road segments on path $path(q, o_i)$ from q to hotels.

$t(q,o_1)$	$t(q,o_1).p$	$t(q,o_4)$	$t(q,o_4).p$	$t(q,o_5)$	$t(q,o_5).p$
0.134	0.025	0.112	0.020	0.122	0.015
0.135	0.045	0.113	0.036	0.123	0.012
0.136	0.020	0.114	0.046	0.125	0.018
0.137	0.005	0.115	0.118	0.126	0.057
0.138	0.005	0.116	0.076	0.127	0.036
0.142	0.050	0.117	0.096	0.128	0.113
0.143	0.090	0.118	0.090	0.129	0.142
0.144	0.040	0.120	0.018	0.130	0.036
0.145	0.010	0.125	0.050	0.131	0.112
0.146	0.010	0.126	0.090	0.132	0.141
0.148	0.175	0.127	0.040	0.133	0.048
0.149	0.315	0.128	0.160	0.134	0.138
0.150	0.140	0.129	0.130	0.135	0.108
0.151	0.035	0.131	0.030	0.137	0.024
0.152	0.035				

Table 5: Possible traveling times, t(q, o_i), of the path from q to $o_i$and their appearance probabilities, t(q, o_i). p.

Symbols	Descriptions
RN (or G)	An uncertain road network
V	A set of nodes, $n_1 \sim n_{\|V\|}$, in RN
E	A set of edges in RN
S	A set of facility objects, $o_1 \sim o_n$, in RN
q	A query point of PTSQ
$dist(x, y)$	The shortest traveling distance from point x to y
$t(x, y)$	The traveling time from x to y
α	The user-specified probability threshold
$[T_i^-, T_i^+]$	Traveling time interval of $t(q, o_i)$
$[v^-(n_j, n_k), v^+(n_j, n_k)]$	The minimum and maximum speeds on e_{jk}
$T^+(.,.).\beta$	A β-upper-bound of $t(.,.)$
$dist^i(o_j, o_k)$	The distance of the ith road segment e^i_{jk}
$[v^{i-}(o_j, o_k), v^{i+}(o_j, o_k)]$	The minimum and maximum speeds on e^i_{jk}
$[T_j^{(w)-}, T_j^{(w)+}]$	The traveling time interval of the w-th edge on path $path(q, o_j)$
$dist^i(o_j, m_1)$	The distance between o_j and m_1
$dist^i(o_j, m_1^\beta)$	The distance between o_j and m_1^β
$m_i(o_i, o_j)$	Time Voronoi point of o_i between o_i and o_j
$m_i^\beta(o_i, o_j)$	Probabilistic time Voronoi point of o_i w.r.t. o_j

Table 6: Notations and their descriptions.

CHAPTER III

PRUNING STRATEGIES

Time Bound Pruning

In this section, we will introduce a pruning method to quickly eliminate objects that are not PSTQ query answers. Specifically, consider intervals of two traveling time $\mathrm{t}(q, o_i)$ and $\mathrm{t}(q, o_j)$. If the lower bound of the traveling time $\mathrm{t}(q, o_i)$ is greater than the upper bound of the traveling time $\mathrm{t}(q, o_j)$, then we can conclude that o_i is not a query answer to the PSTQ due to the existence of o_j. More generally, we can eliminate objects from a facility set, S = $\{\mathrm{o}_1$, o_2,… , $\mathrm{o}_n\}$, by using this pruning method. Let $[\mathrm{T}_i^-, \mathrm{T}_i^+]$ represent the traveling time interval of $\mathrm{t}(q, o_i)$= $dist(q, o_i)$/ $v(q, o_i)$ from object q to o_i, where T_i^- is the lower bound of the traveling time, and T_i^+ is the upper bound of the traveling time. We have the following lemma about the pruning method with the time bounds.

Lemma 1 (Time Bound Pruning). Assuming that $\mathrm{t}(q, o_i) \in [\mathrm{T}_i^-, \mathrm{T}_i^+]$ and $\mathrm{t}(q, o_j) \in [\mathrm{T}_j^-, \mathrm{T}_j^+]$ for $o_i,\ o_j \in$ S, if $\mathrm{T}_i^- > \mathrm{T}_j^+$, then o_i can be safely pruned from S.

Next, we discuss how to compute the lower and upper bounds of the traveling time in our time bound pruning method. Assume that there exists a set of points n_{st}, n_{st+1}, n_{st+2}, . . . , n_{ed}, where n_{st} is a starting point and n_{ed} is an ending point. Nodes n_j(or n_k) are points on road segments, which belong to $path(\mathrm{q}, \mathrm{o_i})$. The velocity of traveling on road $\mathrm{e}_{j,k}$, is given by v $\in$[$v^-(\mathrm{n}_j, \mathrm{n}_k)$, $v^+(\mathrm{n}_j, \mathrm{n}_k)$]. Thus, we have T_{jk}^- = $dist(\mathrm{n}_j, \mathrm{n}_k)$ / $v^+(\mathrm{n}_j, \mathrm{n}_k)$, and T_{jk}^+ = $dist(\mathrm{n}_j, \mathrm{n}_k)/v^-(\mathrm{n}_j, \mathrm{n}_k)$, where T_{jk}^- and T_{jk}^+ are lower and upper bounds of the traveling time on edge $\mathrm{e}_{j,k}$, respectively. The lower bound of the time cost, T_i^-, of path from q to o_i is the summation of T_{jk}^{-} for all edges $e_{j,k}$ on $path(q, o_i)$. Similarly, the upper bound of the time cost, T_i, is the summation of T_{jk}^{+} for all edges $e_{j,k}$ on $path(q, o_i)$.

Lemma 2 (Lower and Upper Bounds of the Traveling Time). For two points n_j and n_k on $path(q, o_i)$, if the velocity of traveling on the road is $v \in [v^{-}(n_j, n_k), v^{+}(n_j, n_k)]$, we have $T_{jk}^{-} = dist(n_j, n_k)/v^{+}(n_j, n_k)$, $T_{jk}^{+} = dist(n_j, n_k)/v^{-}(n_j, n_k)$, $T_i^{-} = \sum_{\forall ejk \in path(q,o_i)} T_{jk}^{-}$ and $T_i^{+} = \sum_{\forall ejk \in path(q,o_i)} T_{jk}^{+}$.

Below, we show our time bound pruning method that filters out false alarms, using the following example.

Example 4. As shown in Table 5, the traveling time, $t(q, o_1)$, from point q to o_1 is within interval $[T_1^{-}, T_1^{+}]$ = [0.134, 0.152], $t(q, o_4)$ from point q to o_4 is within interval $[T_4^{-}, T_4^{+}]$ = [0.112, 0.131], and $t(q, o_5)$ from point q to o_5 is within interval $[T_5^{-}, T_5^{+}]$ = [0.122, 0.137]. Since $T_1^{-} > T_4^{+}$, we can thus immediately prune object o_1.

In Example 4, we use Lemma 1 to prune object o_1 from S. Note, however, that it is infeasible to prune object o_5 from S by using time bound pruning, since the traveling time interval of $t(q, o_5)$ overlaps with that of $t(q, o_4)$. Thus, in the next section, we will explore probabilistic information to enhance the pruning power.

Probabilistic Bound Pruning

In this section, we introduce a pruning method that uses probabilistic bound to filter out false alarms. Specifically, consider intervals of two traveling time $t(q, o_i)$ and $t(q, o_j)$, that have overlaps. In order to prune object o_j, we introduce a way to prune a portion of $t(q, o_i)$, and get probabilistic upper bound of $t(q, o_i)$. The definition of probabilistic upper bound can be given as follows:

Definition 5 (β – upper – bound). Assuming that the traveling time $t(q, o_i) \in [T_i^{-}, T_i^{+}]$ for $o_i \in S$, a β – upper – bound of $t(q, o_i)$ is denoted as $T_i^{+}.\beta$ which satisfies $\Pr\{t(q, o_i) \in [T_i^{-}, T_i^{+}.\beta]\} = \beta$.

In Definition 5, the β – upper – bound, $T_i^{+}.\beta$, is given by an upper bound of the traveling time t(q,o_i), such that t(q,o_i) is in the interval $[T_i^-, T_i^+.\beta]$ with probability β.

Discussion on how to choose β in discrete sample

Given a list of discrete samples $\tau=\{\tau_1, \tau_2, \ldots, \tau_n\}$and its corresponding probability $\tau.p=\{p_1, p_2, \ldots, p_n\}$, where τ_i ($i \in [1,n]$) is in ascending order, we add probability of the first m samples. If it satisfies $\sum_{i=1}^{m-1} p_i<\beta$ and $\sum_{i=1}^{m} p_i \geq \beta$, then we choose $\sum_{i=1}^{m} p_i$ as the value of β.

Lemma 3 (Probabilistic Bound Pruning). Assuming that t$(q,o_i) \in [T_i^-, T_i^+]$ and t$(q,o_j) \in [T_j^-, T_j^+.\beta]$ for $o_i, o_j \in$ S, if it holds that $\beta \geq 1-\alpha$ and $T_i^- > T_j^+.\beta$, then object o_i can be safely pruned fromS.

Proof: We prove the correctness of our probabilistic bound pruning, by using the example in Figure 2. Specifically, for $o_i, o_j \in$ S, t(q,o_i) and t(q,o_j) are with intervals of two traveling time $[T_i^-, T_i^+]$ and $[T_j^-, T_j^+]$ respectively. Assume that, $T_j^+.\beta$ is a β – upper – bound of t(q,o_j) which satisfies Pr{ t$(q,o_j) \in [T_j^-, T_j^+.\beta]$}= β. From Eq. (1) in Definition 4, we overestimate Pr{ t$(q,o_j) > T$ } of any other objects o_j in S. Since we have Pr{ t$(q,o_j) > T$ } ≤ 1 for any other objects, we have the probability that o_i has the shortest time to q below:

$$Pr_{PSTQ}(q,o_j) \qquad (2)$$

$$\leq \sum_{\forall T} \left(\Pr\{t(q,o_i)=T\} \cdot (\Pr\{t(q,o_j)>T\}) \times 1\right)$$

$$\leq \sum_{\forall T} \Pr\{t(q,o_i)=T\} \cdot (\Pr\{t(q,o_j)>T \mid t(q,o_j) \epsilon [T_j^+, T_j^+.\beta]\} +$$

$$\Pr\{t(q,o_j)>T \mid t(q,o_j) \epsilon [T_j^+.\beta, T_j^+]\}))$$

From the lemma assumption, since $T_j^+.\beta < T_i^- \leq t(q,o_i)=T$, we have $\Pr\{t(q,o_i) > T \mid t(q,o_j) \epsilon [T_i^-, T_i^+.\beta]\} = 0$. Moreover, from the definition of β – upper – bound in Definition 5, it hold that : $\Pr\{t(q,o_j) > T \mid t(q,o_j) \in [T_j^+.\beta, T_j^+]\} < 1-\beta$. Thus, we can derive that:

$$Pr_{PSTQ}(q,o_j) < \sum_{\forall T} \Pr\{t(q,o_j)=T\} \times (0+1-\beta)$$

$$< 1 \times (1-\beta) = 1-\beta$$

Therefore, we can conclude that the upper bound of $Pr_{PSTQ}(q, o_j)$ is $1-\beta$. If $\beta \geq 1-\alpha$, that is $1-\beta \leq \alpha$, the probability $Pr_{PSTQ}(q, o_j)$ that o_i has the shortest time to q must be less than α. Hence, we can safely prune o_i from S.

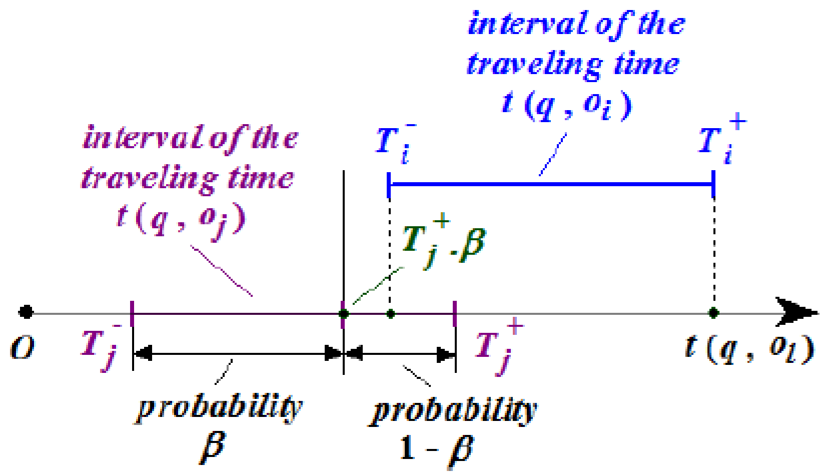


Figure 2: An example of probabilistic bound pruning.

An example of using probabilistic bound pruning can be shown as follows:

Example 5. As shown in Table 5, the traveling time, $t(q, o_4)$ from point q to o_4 is within interval $[T_4^-, T_4^+] = [0.112, 0.131]$. And that $t(q, o_5)$ from point q to o_5 is within interval $[T_5^-, T_5^+] = [0.122, 0.137]$. T_5^- is the lower bound of $t(q, o_5)$. Next, we compute a $\beta-\text{upper}-\text{bound}$ for o_4. According to Lemma 3, we have the following upper bound for $t(q, o_5)$. Given a threshold $\alpha = 80\%$, a β bound of $t(q, o_5)$ must satisfy $\beta \geq 1-\alpha = 20\%$, we choose a $0.22-\text{upper}-\text{bound}$ of $t(q, o_5)$ which is within [0.122,0.137], a probabilistic upper bound of $t(q, o_5)$ is $T_4^+.0.22 = 0.115$, since $0.115 < 0.122$, we can immediately prune object o_5 from S.

Generalization of Probability Upper Bound

In Lemma 3, we use the $\beta-\text{upper}-\text{bound}$, $T_j^+.\beta$, of only one object o_j to prune object o_i. Below, we generalize the pruning method to the one that uses multiple (e.g., s) candidate objects o_{j1}, o_{j2},…, o_{js}, where $s \geq 1$. Specifically, for each object $o_{jr}(1 \leq r \leq s)$ assume that its traveling time $t(q, o_{jr})$ has a $\beta_r-\text{upper}-\text{bound}$, denoted as $T_{jr}^+.\beta r$, such that $T_i^- > T_{jr}^+.\beta r$, where T_i^- is the lower bound of the traveling time $t(q, o_i)$ for an object o_i.

Then, we can define a generalized (tighter) upper bound of the PSTQ probability $\mathrm{Pr_{PSTQ}}(\mathrm{q}, \mathrm{o_i})$ which is given by $\prod_{r=1}^{s}(1-\beta_r)$. From Lemma 3, we can immediately derive the following corollary.

Corollary 1 Given objects $\mathrm{o}_i, o_{jr} \in S (1 \leq r \leq s)$, assume that $t(q, o_i) \in [T_i^-, T_i^+]$ and $t(q, o_{jr}) \in [T_j^-, T_{jr}^+ \cdot \beta_r]$, where $T_i^- > T_{jr}^+ . \beta_r$ holds. If it satisfies the condition that $\prod_{r=1}^{s}(1-\beta_r) \leq \alpha$, then object o_i can be safely pruned.

Discussions on How to Obtain Probabilistic Upper Bound

In Lemma 3, object o_i can be filtered out from S, if a $\beta - \mathrm{upper} - \mathrm{bound}$ of $t(q, o_j)$ holds $\beta \geq 1 - \alpha$ and $T_i^- > T_{jr}^+ . \beta$. Now one remaining issue to be addressed is as follows: given a threshold α, how to obtain probabilistic upper bound (i.e., $\beta - \mathrm{upper} - \mathrm{bound}$) for a path between two objects q and o_i.

We next discuss how to obtain such a $\beta - \mathrm{upper} - \mathrm{bound}$ of traveling time $t(q, o_j)$ (as given in Lemma 3) for a path, $path(\mathrm{q}, \mathrm{o_j})$. Our basic idea is to consider summing up $\beta_w - \mathrm{upper} - \mathrm{bounds}$ of traveling times for all the l edges (road segments) on $path(\mathrm{q}, \mathrm{o_j})$.

Without loss of generality, we assume that the $\mathrm{w-th}$ edge on $path(\mathrm{q}, \mathrm{o_j})$ has the traveling time intervals $\left[T_j^{(w)-}, T_j^{(w)+}\right]$, where $1 \leq w \leq l$. Moreover, we denote $ub_w (1 \leq w \leq l)$ as the $\beta_w - \mathrm{upper} - \mathrm{bounds}$ of the traveling time on the $\mathrm{w-th}$ edge of $path(\mathrm{q}, \mathrm{o_j})$. In other words, the traveling time on the $\mathrm{w-th}$ edge is interval $[T_j{}^{(w)}{}^-, ub_w]$ with probability β_w. Then, for $\beta = \prod_{w=1}^{l} \beta_w$, we can obtain a $\beta - \mathrm{upper} - \mathrm{bound}$, $T_j^+ . \beta$, of $t(q, o_j)$ by $\sum_{w=1}^{l} ub_w$. For simplicity, below, we denote the $\beta - \mathrm{upper} - \mathrm{bound}$, $T_j^+ . \mathrm{\beta}$ as ub.

Note that, we can select different pairs of β_w and ub_w on the $\mathrm{w-th}$ edge $(1 \leq w \leq l)$. In order to enhance the pruning power, we want to choose $(\beta_w, ubw$)-pairs on edges of $path(\mathrm{q}, \mathrm{o_j})$ such that the $\beta - \mathrm{upper} - \mathrm{bound}$, ub(i.e.,T_j^+, β)， is minimized, on condition that

$\beta \geq 1-\alpha$.

Straightforward Method: A straightforward method to minimize ub is to enumerate all possible combinations of traveling time samples on edges $path(\mathrm{q}, \mathrm{o_j})$ and identify one combination that has the smallest ub and satisfies the β constraint (i.e., $\beta \geq 1-\alpha$).

Specifically, assume that there are m samples of traveling times on each edge. Then, the $\mathrm{w-th}$ edge on $path(\mathrm{q}, \mathrm{o_j})$ has m possible $\beta_w - \mathrm{upper} - \mathrm{bounds}$, ub_w, which indicate that the traveling time falls into interval $[T_j{}^{(w)\,-}, ub_w]$ with probability β_w. By materializing all possible combinations of (β_w, ub_w)-pairs on edges, we can find a minimum $\beta - \mathrm{upper} - \mathrm{bound}$, $ub(= \sum_{\mathrm{w}=1}^{l} ub_w)$, such that β=$\prod_{w=1}^{l} \beta_w \geq 1-\alpha$.

However, the straightforward method mentioned above is rather inefficient. In particular, the time complexity is exponential, that is, $\mathrm{O}(m^l)$, where m is the (average) number of samples on each edge, and l is the path length from q to o_j. Thus, it is impractical, or even infeasible, to enumerate all possible combinations for long paths. Inspired by this, we alternatively seek for a greedy algorithm to approximately compute a tight $\beta - \mathrm{upper} - \mathrm{bound}$.

Greedy Algorithm to Compute the β-upper-bound: Next, we provide a greedy algorithm to obtain a tight (small) $\beta - \mathrm{upper} - \mathrm{bound}$ of the traveling time on a path, without enumerating all sample combinations. Specifically, our greedy algorithm recursively extends the path by varying its length from 1 to l, and maintains M (β, ub)-pairs for the traveling time on $path(\mathrm{q}, \mathrm{o_j})$ of length l.

1. *Base Case*. On the first edge of path $path(\mathrm{q}, \mathrm{o_j})$ (i.e., the subpath, P_1, of length w = 1), let arr_t_1 be a sorted array of size n, which contains $\mathrm{n_1}$ samples of the traveling time in ascending order. We first select M out of $\mathrm{n_1}$ samples $t_1^{(1)}, t_1^{(2)}$ and $t_1^{(M)}$ as $\beta - \mathrm{upper} - \mathrm{bounds}$ of the first edge (note: including the minimum and maximum time samples). Each sample, $t_1^{(w)} (1 \leq w \leq M)$, is associated with a β_w value, denoted as $t_1^{(w)}.p$,

which is given by the probability that the traveling time on the first edge is not greater than $t_1^{(w)}$. Denote arr_u_1 and arr_β_1 as arrays $\{t_1^{(w)}/1 \leq w \leq M\}$ and $\{t_1^{(w)}.p/1 \leq w \leq M\}$, respectively, where arr_u_1 is sorted in ascending order of $\beta_w -$ upper $-$ bounds, and arr_β_1 $[w] \geq 1 - \alpha$ for all w.

2. *Recursive Computation.* When the prefix, P_K,of $path(\mathrm{q}, \mathrm{o_j})$ has length k (i.e., w = k), assume that we can obtain two arrays, arr_u_k = $\{t_k^{(w)}/1 \leq w \leq M\}$and arr_β_k=$\{t_k^{(w)}.p/1 \leq w \leq M\}$, which stores different $\beta_w -$ upperbounds of the traveling time on subpath P_K, and their corresponding $\beta(\geq 1 - \alpha)$ values, respectively. Note that, $\beta_w -$ upper bounds in array arr_u_w are sorted in ascending order.

When we consider the subpath (prefix), P_{k+1}, of $path(\mathrm{q}, \mathrm{o_j})$ (for $w = k + 1$), we combine the $\beta_w -$ upper bounds on subpath P_k with that of the $(k + 1) -$ th edge on $path(\mathrm{q}, \mathrm{o_j})$ Specifically, we obtain combinations of M $\beta_w -$ upper bounds from arr_u_k and n_{k+1} samples of traveling times on the $(\mathrm{k} + 1) -$ th edge of the path (stored in a sorted array arr_t_{k+1}). Each combination is corresponding to a $\beta -$ upper $-$ bounds, $(arr\_u_k[i] + arr\_t_{k+1}[j])$, as well as the β value $arr\_\beta_k[i] \sum_{r=1}^{j} arr_{t_{k+1}[r]}.p$ for $1 \leq i \leq M$ and $1 \leq j \leq \mathrm{n}_{k+1}$. Then, we choose M out of $(M \cdot \mathrm{n}_{k+1})$ combinations (uniformly w.r.t.β values) such that $\beta \geq 1 - \alpha$, and store pairs of $\beta -$ upper $-$ bounds and their β values s in arrays arr_u_{k+1} and arr_β_{k+1}, respectively.

```
Algorithm PUB {
  Input: arrays $arr\_t_w$ that store the traveling time samples on the $w$-th edge of
         path $path(q, o_j)$, and each sample $arr\_t_w[i]$ has an existence
         probability $arr\_t_w[i].p$ ($1 \leq w \leq l$)
  Output: arrays $arr\_u_l$ and $arr\_\beta_l$
  (1)  let $arr\_u_w$ and $arr\_\beta_w$ be arrays of size $M$ ($1 \leq w \leq l$)
  (2)  $arr\_u_1[1] = min_{\forall w} arr\_t_1[w]$
  (3)  $arr\_u_1[M] = max_{\forall w} arr\_t_1[w]$
  (4)  randomly select $(M - 2)$ samples from array $arr\_t_1$ (excluding $arr\_u_1[1]$
         and $arr\_u_1[M]$) and store them in array $arr\_u_1$ in ascending order
  (5)  update array $arr\_\beta_1$, w.r.t. $arr\_u_1$
  (6)  for $w = 2$ to $l$        // with the length of the path prefix = $w$
  (7)    for $i = 1$ to $M$
  (8)      for $j = 1$ to $n_w$
  (9)        compute a $\beta$-upper bound, $(arr\_u_{w-1}[i] + arr\_t_w[j])$, and its
               $\beta$ value, $(arr\_\beta_{w-1}[i] \cdot \sum_{r=1}^{j} arr\_t_w[r].p)$
  (10)   randomly select $M$ out of $(M \cdot n_w)$ pairs of $\beta$-upper bounds and $\beta$ ($\geq 1 - \alpha$)
  (11)   store $M$ pairs into arrays $arr\_u_w$ and $arr\_\beta_w$
  (12) return arrays $arr\_u_l$ and $arr\_\beta_l$
}
```

Figure 3: Algorithm for finding probabilistic upper bound.

Figure 3 illustrates the pseudo code, Algorithm PUB, of finding $\beta - \text{upper} - \text{bounds}$ of $path(\mathrm{q}, \mathrm{o_j})$, as well as different β values. In particular, we maintain two arrays of size M, arr_u_w and arr_β_w, which are used for storing $\beta - \text{upper} - \text{bounds}$ and β values for a subpath(prefix) of path $path(\mathrm{q}, \mathrm{o_j})$ with length w (line 1). Then, starting from the first edge on $path(\mathrm{q}, \mathrm{o_j})$, we compute arrays arr_u_1, by randomly selecting samples from array arr_t_1 (which contains traveling time samples on the first edge) (lines 2-4). We also update the β values in array arr_β_1, with respect to that in arr_u_1 (line 5). Next, we extend the subpath for length w from 2 to l, and compute the corresponding arrays arr_u_w and arr_β_w (lines 6-11). That is, each time we consider a combination of traveling time samples of subpath P_{w-1} and that of the $\mathrm{w-th}$ edge, and obtain the $\beta - \text{upper bound}$ given by $(arr\_u_{w-1}[i] + arr\_t_w[j])$, where $\beta = arr\_\beta_{w-1}[i] \cdot \sum_{r=1}^{j} arr\_t_w[r] \cdot \mathrm{p}$(line 9). Then, we randomly retain M sample combinations with β values $\geq 1 - \alpha$, and store them in arrays arr_u_w and arr_β_w (lines 10-11). Finally, we return the arrays for the entire $path(\mathrm{q}, \mathrm{o_j})$ of length l, that is, arr_u_l

and arr_β_l (line 12).

CHAPTER IV

PRUNING WITH PRE-COMPUTATIONS

Time Voronoi Pruning

In this section, we introduce a time Voronoi pruning method to enable PSTQs that return objects with low probabilities, of being the PSTQ answers. Consider two facility points o_1 and o_2 on road networks, we propose an approach to locate two time Voronoi points m_1 and m_2, which divided the path from o_1 to o_2 into three parts $path(\mathrm{o}_1, \mathrm{m}_1)$, $path(\mathrm{m}_1, \mathrm{m}_2)$, and $path(\mathrm{m}_2, \mathrm{o}_2)$. The time Voronoi points m_1 and m_2 are defined as follows:

Definition 6 (Time Voronoi Points). Given two facility points o_1 and o_2 on the road network RN, m_1 and m_2 are located on the path from o_1 to o_2. Let $T^+(\mathrm{o}_1, \mathrm{m}_1) = \sum_{\forall e_{jk} \in path(\mathrm{o}_1, \mathrm{m}_1)} dist(\mathrm{o}_j, \mathrm{o}_k)/v^-(\mathrm{o}_j, \mathrm{o}_k)$ represent the maximum traveling time between o_1 and m_1, and $T^-(\mathrm{m}_1, \mathrm{o}_2) = \sum_{\forall e_{jk} \in path(\mathrm{m}_1, \mathrm{o}_2)} dist(\mathrm{o}_j, \mathrm{o}_k)/v^+(\mathrm{o}_j, \mathrm{o}_k)$ represent the minimum traveling time between m_1 and o_2. The time $Voronoi$ point, m_1 satisfies the condition that $T^+(\mathrm{o}_1, \mathrm{m}_1) = T^-(\mathrm{m}_1, \mathrm{o}_2)$.

Similarly, $T^+(\mathrm{o}_1, \mathrm{m}_2) = \sum_{\forall e_{jk} \in path(\mathrm{o}_1, \mathrm{m}_2)} dist(\mathrm{o}_j, \mathrm{o}_k)/v^-(\mathrm{o}_j, \mathrm{o}_k)$ represent the maximum traveling time between o_1 and m_2, and $T^-(\mathrm{m}_2, \mathrm{o}_2) = \sum_{\forall e_{jk} \in path(\mathrm{m}_2, \mathrm{o}_2)} dist(\mathrm{o}_j, \mathrm{o}_k)/v^+(\mathrm{o}_j, \mathrm{o}_k)$ represent the minimum traveling time between m_2 and o_2. The time Voronoi point, m_2, satisfies the conditionthat $T^+(\mathrm{o}_1, \mathrm{m}_2) = T^-(\mathrm{m}_2, \mathrm{o}_2)$.

Lemma 4 (Time Voronoi Pruning). Assuming that m_1 and m_2 are two time Voronoi points on $path(\mathrm{o}_1, \mathrm{o}_2)$, if q is located on $path(\mathrm{o}_1, \mathrm{m}_1)$, we can immediately prune o_2; if q is located on $path(\mathrm{m}_2, \mathrm{o}_2)$, we can safely prune o_1.

Proof: We consider the following two cases:

Case 1: Time Voronoi point m_1 satisfies the condition that:

$\sum_{\forall e_{jk} \in path(o_1, m_1)} dist(o_j, o_k)/v^-(o_j, o_k) = \sum_{\forall e_{jk} \in path(m_1, o_2)} dist(o_j, o_k)/v^+(o_j, o_k)$ in Definition 6. If q is located on $path(o_1, m_1)$, we have $dist(o_1, q) < dist(o_1, m_1), \forall ejk \in path(o_1, q), v(o_j, o_k) \geq v^-(o_j, o_k)$. Moreover, we have$dist\ (o_2, q) > dist(o_2, m_2), \forall ejk \in path(o_2, q), v(o_j, o_k) \leq v^+(o_j, o_k)$. Thus, we can conclude that $T(o_1, q) = \sum_{\forall ejk \in path(o_1, q)} dist(o_j, o_k)\ /v(o_j, o_k)$ must be less than $T(o_2, q) = \sum_{\forall ejk \in path(o_2, q)} dist(o_j, o_k)\ /v(o_j, o_k)$, that is, $T(o_1, q) < T(q, o_2)$. Thus, if q is located on $path(o_1, m_1)$, o_2 can be pruned.

Case 2: The time Voronoi point m_2 satisfies the condition that

$\sum_{\forall ejk \in path(o_2, m_2)} dist(o_j, o_k)/v^-(o_j, o_k) = \sum_{\forall ejk \in path(m_2, o_1)} dist(o_j, o_k)/v^+(o_j, o_k)$ in Definition 6. If q is located on $path(m_2, o_2)$, we have $dist(q, o_2) < dist(m_2, o_2)$, $\forall ejk \in path(q, o_2), v(o_j, o_k) \geq v^-(o_j, o_k)$. Moreover, we have$(o_1, q) > dist(o_1, m_2)$, $\forall ejk \in path(o_2, q), v(o_j, o_k) \leq v^+(o_j, o_k)$.

Then, $T(o_2, q) = \sum_{\forall ejk \in path(o_2, q)} dist(o_j, o_k)/v(o_j, o_k)$ must be less than $T(o_1, q) = \sum_{\forall ejk \in path(o_1, q)} dist(o_j, o_k)/v(o_j, o_k)$, that is, $T(o_2, q) < T(o_1, q)$. Therefore, if q is located on $path(m_2, o_2)$, o_1 can be pruned.

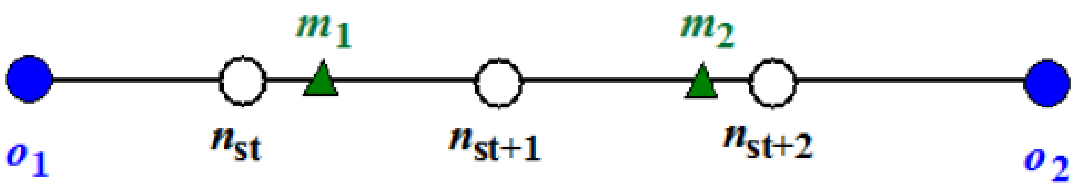


Figure 4: Time Voronoi pointsm_1 and m_2.

Example 6. As shown in Figure 5, o_4 and o_5 are two facility points on road network RN, n_2, n_1, n_{11}, n_8 are intersection points on the path between o_4 and o_5, m_4 and m_5 are two time Voronoi points. m_4 holds $T^+(o_4, m_4) = T^-(m_4, o_5)$ and m_5 holds $T^+(o_5, m_5) = T^-(m_5, o_4)$. Based on Lemma 4, we know that when q is located on $path(o_4, m_4)$, o_5 can be pruned safely;

when q is located on $path(o_5,m_5)$, o_4 can be pruned safely.

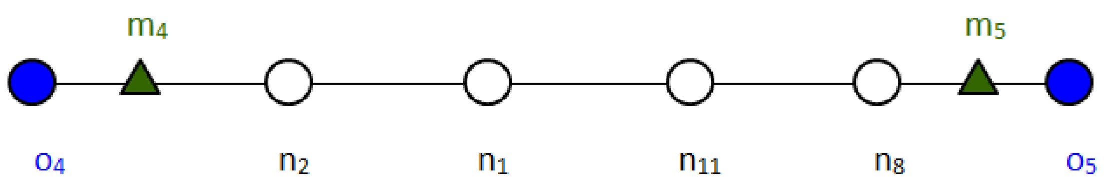


Figure 5: Time Voronoi Pointsm_4 and $m_5$on path(o_4, o_5)

Evaluation of m_1 and m_2

We now discuss how to obtain offline pre-computations, m_1 and m_2. Assuming that o_1 and o_2 are two facility points on road network RN, there exist a set of nodesn_{st}, n_{st+1}, n_{st+2}, . . . , and n_{ed}, and l edges between o_1 and o_2 on the path.For edges e_{ij} on path(o_1, o_2), we calculate lower and upper bounds of the traveling times on e_{ij}, that is $[T_{ij}^{(1)-}, T_{ij}^{(1)+}]$, $[T_{ij}^{(2)-}, T_{ij}^{(2)+}]$,…, and $[T_{ij}^{(l)-}, T_{ij}^{(l)+}]$. If $T_{ij}^{(1)+} \geq \sum_{w=2}^{l} T_{ij}^{(w)-}$, we can determine that m_1 is located on the first edge; otherwise, we further consider the second edge on the path. If $T_{ij}^{(1)+} \leq \sum_{w=2}^{l} T_{ij}^{(w)-}$and $T_{ij}^{(1)+} + T_{ij}^{(2)+} \geq \sum_{w=3}^{l} T_{ij}^{(w)-}$, then m_1 must be located on the second edge. The rest can be deduced by analogy: If $\sum_{w=1}^{i-1} T_{ij}^{(w)+} < \sum_{w=i}^{l} T_{ij}^{(w)-}$and $\sum_{w=1}^{i} T_{ij}^{(w)+} \geq \sum_{w=i+1}^{l} T_{ij}^{(w)-}$, we can determine that m_1 located on the w – th edge of path(o_1,o_2).

Next, we explain how the location of m_1 is computed on an edge e_{ij} by the pervious step. Moreover, the offset of m_1 on edge e_{ij} is denoted by $dist(n_i, m_1)$. Assume that the minimumand maximum speeds on edge e_{ij} are denoted by $v^-(n_i, n_j)$ and $v^+(n_i, n_j)$, respectively. Then, from Definition 6, we have $T^+(o_1, m_1)= T^-(m_1, o_2)$. The formula can be rewritten as follows:

$$T^+(o_1, n_i) + \frac{dist(n_i,m_1)}{v^-(n_i,n_j)} = \frac{dist(n_i,n_j)-dist(n_i,m_1)}{v^+(n_i,n_j)} + T^-(n_j, o_2) \qquad (3)$$

which can be rewritten as:

$$dist(n_i, m_1) = \frac{T^-(n_i,o_2)-T^+(o_1,n_i)}{\frac{1}{v^-(n_i,n_j)}+\frac{1}{v^+(n_i,n_j)}} \qquad (4)$$

Thus, we can find the offset, $dist(n_i, m_1)$, of m_1 on edge e_{ij}. The case of finding the position of m_2 is similar and thus omitted.

Probabilistic Time Voronoi Pruning

In order to enhance the pruning power of time Voronoi pruning, we will propose an improved approach that considers probabilistic information. Similar to time Voronoi points, we introduce two probabilistic time Voronoi points, m_1^{β} and m_2^{β} as follows:

Definition 7 (Probabilistic Time Voronoi Points). Given two facility points o_1 and o_2 on the road network RN, denote probabilistic time Voronoi points with respect to o_1 and o_2 as m_1^{β} and m_2^{β} respectively, where $\beta \geq 1-\alpha$ holds. The probabilistic time Voronoi point, m_1^{β}, is defined as a point on the path from o_1 to o_2 such that $T^{+}\left(\mathrm{o}_1, \mathrm{m}_1^{\beta}\right).\beta = T^{-}\left(\mathrm{m}_1^{\beta}, \mathrm{o}_2\right)$, where $T^{+}\left(\mathrm{o}_1, \mathrm{m}_1^{\beta}\right).\beta$ is a $\beta-\text{upper}-\text{bound of the traveling time }\mathrm{t}\left(\mathrm{o}_1, \mathrm{m}_1^{\beta}\right)$.

Similarly, a probabilistic time Voronoi point, m_2^{β}, is defined as a point on the path, $\mathrm{path}(\mathrm{o}_2, \mathrm{o}_1)$, from o_2 to o_1, that satisfies the condition $T^{+}\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right).\beta = T^{-}\left(\mathrm{m}_2^{\beta}, \mathrm{o}_1\right)$, where $T^{+}\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right).\beta$ is a $\beta-\text{upper}-\text{bound ofthe traveling timet}\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right)$, and $\beta \geq 1-\alpha$.

Lemma 5 (Probabilistic Time Voronoi Pruning). Assuming that m_1^{β} and m_2^{β} are two probabilistic time Voronoi points on path, $path(\mathrm{o}_1, \mathrm{o}_2)$, if q is located on $path\left(\mathrm{o}_1, \mathrm{m}_1^{\beta}\right)$, then o_2 can be safely pruned; if q is located on $path\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right)$, then o_1 can be pruned.

Proof: We consider the following 2 cases:

Case 1: If q is located on $path\left(\mathrm{o}_1, \mathrm{m}_1^{\beta}\right)$, we have $dist(\mathrm{o}_1, \mathrm{q}) < dist\left(\mathrm{o}_1, \mathrm{m}_1^{\beta}\right)$ and $dist(\mathrm{o}_2, q) > dist\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right)$. From Definition7, we know that m_1^{β} holds that $T^{-}\left(\mathrm{o}_1, \mathrm{m}_1^{\beta}\right) = T^{+}\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right).\beta$ and $\beta \geq 1-\alpha$, thus we have $\beta \geq 1-\alpha$ and $T^{-}(\mathrm{o}_2, q) > T^{+}(\mathrm{o}_1, \mathrm{q}).\beta$. Based on Lemma3, we have $Pr_{PSTQ}\{t(q, \mathrm{o}_2)\} < \alpha$, thus, o_2 can be pruned.

Case 2: If q is located on $path\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right)$, we have $dist(\mathrm{o}_2, q) < dist\left(\mathrm{o}_2, \mathrm{m}_2^{\beta}\right)$ and

$dist(\mathrm{o}_1, \mathrm{q}) > dist\left(\mathrm{o}_2, \mathrm{m}_2^\beta\right)$. From Definition7, we know that m_2^β holds that $T^-\left(\mathrm{o}_2, \mathrm{m}_2^\beta\right) = T^+\left(\mathrm{o}_1, \mathrm{m}_2^\beta\right).\beta$ and $\beta \geq 1-\alpha$, thus we have $\beta \geq 1-\alpha$ and $T^-(\mathrm{o}_1, q) > T^+(\mathrm{o}_2, \mathrm{q}).\beta$. Based on Lemma3, we have $Pr_{PSTQ}\{t(q, o_1)\} < \alpha$, thus, o_1 can be pruned.

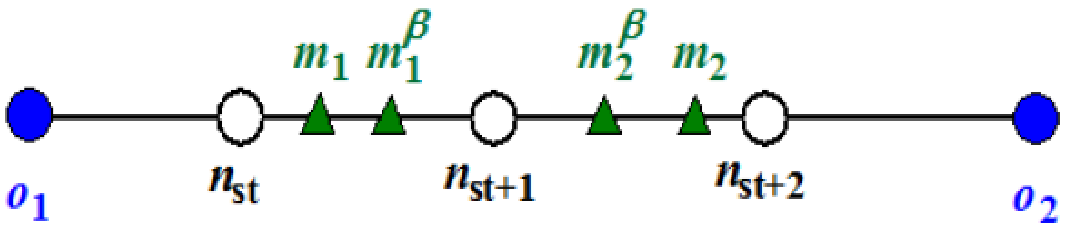


Figure 6: Probabilistic time Voronoi points $\mathrm{m}_1^\beta$and m_2^β

Example 7. As shown in Figure 7, o_4 and o_5 are two facility points on road network RN, n_2, n_1, n_{11} and n_8 are intersection points located between o_4 and o_5, m_4 and m_5 are two time Voronoi points, and m_4^β and m_5^β are two probabilistic time Voronoi points, and m_4^β satisfies $\beta \geq 1-\alpha$ and $\mathrm{T}^-(\mathrm{o}_5, \mathrm{m}_4^\beta) = \mathrm{T}^+(\mathrm{m}_4^\beta, \mathrm{o}_4).\beta$, and m_5^β satisfies $\beta \geq 1-\alpha$ and $\mathrm{T}^-(\mathrm{o}_4, \mathrm{m}_5^\beta) = \mathrm{T}^+(\mathrm{m}_5^\beta, \mathrm{o}_5).\beta$. Based on Lemma 5, we know that when q is located on $path(\mathrm{o}_4, \mathrm{m}_4^\beta)$, o_5 can be pruned safely; when q is located on $path(\mathrm{o}_5, \mathrm{m}_5^\beta)$, o_4 can be pruned safely.

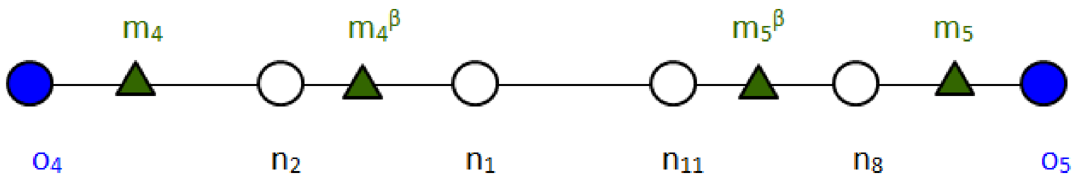


Figure 7: Probabilistic time Voronoi points $\mathrm{m}_4^\beta$and m_5^β on $path(\mathrm{o}_4, \mathrm{o}_5)$

Evaluation of m_1^β and m_2^β

Next, we will discuss how to compute the probabilistic time Voronoi points. We first consider how to find m_1^β. As introduced in Definition 7, m_1^β must satisfy the following conditions: $\beta \geq 1-\alpha$ and $T^-\left(\mathrm{o}_2, \mathrm{m}_1^\beta\right) = T^+\left(\mathrm{o}_1, \mathrm{m}_1^\beta\right).\beta$. Thus, our goal is to find probabilistic time Voronoi point m_1^{β} which satisfies $\mathrm{T}^-\left(\mathrm{o}_2,\mathrm{m}_1^{\beta}\right) = \mathrm{T}^+\left(\mathrm{o}_1,\mathrm{m}_1^{\beta}\right).\beta$, and $\beta = 1-\alpha$. The case of point m_2^{β} is similar, and thus omitted.

In order to obtain the location of m_1^{β}, we first need to identify the edge on which m_1^{β} resides. Our observation is as follows. Given an edge e_{ij}(with ending nodes n_i and n_j) on a path from o_1 to o_2, m_1^{β} is located on e_{ij}, if and only if $T^+(\mathrm{o}_1,\mathrm{n}_i).\beta \leq T^-(n_i,\mathrm{o}_2)$ and $T^+\left(\mathrm{o}_1,\mathrm{n}_j\right).\beta \geq T^-\left(n_j,\mathrm{o}_2\right)$ hold, where $T^+(x,\mathrm{y})\cdot\beta$ is the $\beta-\mathrm{upper}-\mathrm{bound}$ of the traveling time between points x and y on road networks, and $T^-(x,\mathrm{y})$ is the lower bound of the traveling time between x and y.

The Computation of Offset $dist(\mathrm{n}_i,\mathrm{m}_1^{\beta})$

Once we know m_1^{β} is on edgee_{ij}, we can compute the offset of point m_1^{β} on edge e_{ij}, that is $dist(\mathrm{n}_i,\mathrm{m}_1^{\beta})$. Denote the minimum and maximum speeds on edge e_{ij} as $\mathrm{v}^-(n_i,n_j)$ and $\mathrm{v}^+(n_i,n_j)$, respectively. The speed variable on edge e_{ij} has n samples v_1,v_2, ..., andv_n (in ascending order), where each sample $\mathrm{v}_r (1 \leq r \leq n)$ is associated with an appearance probability $\mathrm{v}_r.p$. We have the following lemma to obtain $dist(\mathrm{n}_i,\mathrm{m}_1^{\beta})$.

Lemma 6 Assume that m_1^{β} resides on edgee_{ij}. Given $\mathrm{v}^-(n_i,n_j)$ and $\mathrm{v}^+(n_i,n_j)$, if the speed sample v_r satisfies the condition that $\sum_{\mathrm{s=r}}^{\mathrm{n}} \mathrm{v}_s.p$ is the smallest probability that is greater than or equal to β_r, then we have:

$$dist\left(\mathrm{n}_i,\mathrm{m}_1^{\beta}\right) = \frac{\mathrm{T}^-(\mathrm{n}_i,\mathrm{o}_2)-\mathrm{T}^+(\mathrm{o}_1,\mathrm{n}_i).\beta_s}{\frac{1}{\mathrm{v}_r}+\frac{1}{\mathrm{v}^+(n_i,n_j)}} \qquad (5)$$

Proof: From Definition 7, we consider $T^+\left(\mathrm{o}_1,\mathrm{m}_1^{\beta}\right).\beta = T^-\left(\mathrm{o}_2,\mathrm{m}_1^{\beta}\right)$.

The formula can be rewritten as follows:

$$\mathrm{T}^+(\mathrm{o}_1,\mathrm{n}_i).\beta_s + \mathrm{T}^+\left(\mathrm{n}_i,\mathrm{m}_1^{\beta}\right).\beta_r = T^-\left(\mathrm{m}_1^{\beta},\mathrm{n}_j\right) + \mathrm{T}^-\left(\mathrm{n}_j,\mathrm{o}_2\right) \qquad (6)$$

Where $\beta_s \cdot \beta_r = \beta$

Since it holds that:

$\mathrm{T}^{+}\left(\mathrm{n}_i,\mathrm{m}_1^{\beta}\right).\beta_r = dist\left(\mathrm{n}_i,\mathrm{m}_1^{\beta}\right)/\mathrm{v}_r$ and

$$T^{-}\left(\mathrm{m}_1^{\beta},\mathrm{n}_j\right)=\frac{dist\left(\mathrm{n}_i,\mathrm{n}_j\right)-dist\left(\mathrm{n}_i,\mathrm{m}_1^{\beta}\right)}{\mathrm{v}^{+}(n_i,n_j)}$$

we substitute Eq. (5) into Eq. (6), and obtain:

$$\mathrm{T}^{+}(\mathrm{o}_1,\mathrm{n}_i)\cdot\beta_s+\frac{dist\left(\mathrm{n}_i,\mathrm{m}_1^{\beta}\right)}{\mathrm{v}_r}=\frac{dist\left(\mathrm{n}_i,\mathrm{n}_j\right)-dist\left(\mathrm{n}_i,\mathrm{m}_1^{\beta}\right)}{\mathrm{v}^{+}(n_i,n_j)}+\mathrm{T}^{-}\left(\mathrm{n}_j,\mathrm{o}_2\right)$$

which can be rewritten as Eq. (5). Hence, the lemma holds.

This way, we can obtain the location of m_1^{β} on edgee_{ij}.The case of finding the position of m_2^{β} is similar and thus omitted.

CHAPTER V

PSTQ PROCESSING

In this section, we build an index over offline pre-computed data which can facilitate efficient PSTQ processing. Section 5.1 presents the index construction. Section 5.2 illustrates the PSTQ query procedure over the constructed index.

Index Construction

Data structure of the Index

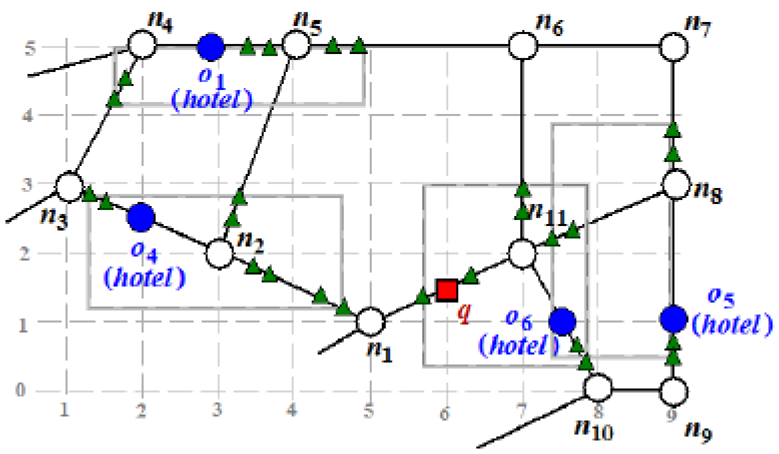
We construct a spatial index, R*-tree [2], over the pre-computed data, including probabilistic time Voronoi points, which can enable efficient PSTQ answering. An R*-tree index is a hierarchical tree structure. Each non-leaf node of the tree contains entries that refer to a set of child nodes. Each leaf node of the index contains a list of index entries that correspond to information of objects. Figure 6 shows an example of such an index.

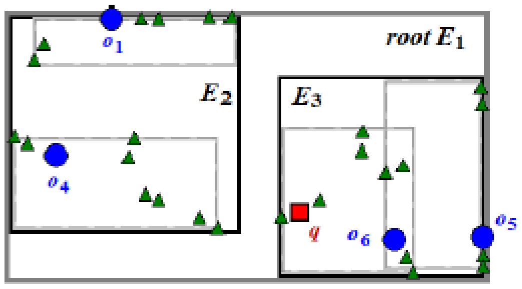
Leaf Nodes

Each entry, E_i, in leaf node, E, stores entries of the form $(oid, \mathrm{MBR}^{\beta}, \mathrm{path}^{\beta}, pointer)$. Here, oid refers to the identity of a facility object o_i in the database. For each pre-computed β value (i.e.,β_1, β_2,…, or β_l), MBR^{β} in entry E_i is the minimum bounding rectangle that tightly encloses object, o_i, and its associated probabilistic time Voronoi points m_i^{β}(i.e., between o_i and any adjacent objects o_j, where o_i is connected to o_j directly). Note that, when $\beta = 1$, the probabilistic time Voronoi point, m_i^{β}, is exactly the time Voronoi point m_i. Moreover, path^{β} in entry E_i records the path between o_i and m_i^{β} for the pre-computed $\beta \in [0, 1]$. The $pointer$, $pointer$, in E stores the address of objecto_i.

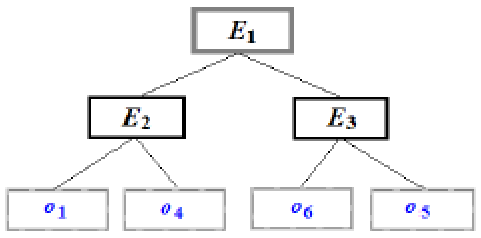
Non-Leaf Nodes

Each entry, E_i, of non-leaf node E contains records in the form $(MBR^{\beta_1}, MBR^{\beta_2}, \ldots, MBR^{\beta_l}, pointer)$, where MBR^{β_k} ($1 \leq k \leq l$) is the minimum bounding box of all MBR^{β} (for $\beta = \beta_k$) of the child node E_i in E, and the pointer, pointer, points to the child node E_i.





(b) Minimum bounding rectangle



(c) Index structure

Figure 8: An example of the index construction

The Construction of Tree Structure

We now illustrate how to construct the index. Figure 6(a) shows how to obtain $MBRs$ of objects. For each object o_i, we record other objects o_j, which are connected to o_i directly (i.e., on the path from o_i to o_j, there are no other objects of the same facility type). According to Lemma 6, we can obtain probabilistic time Voronoi points m_i^{β} of o_i, with respect to other adjacent objects o_j. Then, we generate a minimum bounding rectangle (MBR)MBR, which covers object o_i and points m_i^{β} for different β values. This way, we can compute the $MBRs$ for all objects by the procedure above. Then, the index can be constructed by inserting $MBRs$ of objects into the R*-tree, by using the standard insertion method [2].

Discussions on Selection of β Values

Next, we discuss how to select the value of β. As introduced before, each entry E_i of nodes in the index contains $\mathrm{E}_i.\mathrm{MBR}^{\beta}$, for $\beta \in$ [0, 1]. In a special case where $\beta = 1$, probabilistic time Voronoi point $\mathrm{m}_i^{\beta}(\mathrm{o}_i, \mathrm{o}_j)$ is exactly the time Voronoi point $\mathrm{m}_i(\mathrm{o}_i, \mathrm{o}_j)$. To select possible valuesof β for offline pre-computations, we uniformly generate β values within [0, 1]. In the case that we know the statistics of α specified by historical PSTQ queries, we can pre-select β valuesthat follow the distribution of $(1 - \alpha)$.

Query Procedure

Pruning with the Index

In the sequel, we discuss how to use our constructed index to support execution of queries. Specifically, by traversing the constructed index, we can find the location of a given query point q, and meanwhile retrieve those PSTQ candidates. That is, starting from the root, for each $\mathrm{E}_i.\mathrm{MBR}^{\beta}$ of entry E_i in node E we encounter (for a smallest pre-computed β value greater than or equal to $1 - \alpha$), we check whether or not $\mathrm{E}_i.\mathrm{MBR}^{\beta}$ contains q. If the answer is yes, we will explore the children of node E; otherwise, entry E_i in the index can be safely pruned. After the index traversal, we can obtain a set of PSTQ candidates, which are then refined to return true PSTQ answers. We give the pruning with the index in the following lemma.

Lemma 7 (Index Pruning). Given a query point q, if q is not contained in an MBR^{β} of entry E_i in the index, then E_i can be safely pruned, where $\beta \geq 1 - \alpha$.

Proof: As mentioned before, MBR^{β} of entry E_i is a minimum bounding rectangle enclosed by o_i, time $Voronoi$ point m_i of o_i, and probabilistic time $Voronoi$ point m_i^{β} of o_i for $\beta \in [1 - \alpha, 1]$. Object o_i is a PSTQ answer of q, if and only if q is contained in the MBR^{β} of E_i. Conversely, if q is not contained in an $\mathrm{MBR}$of entry E_i, then E_i cannot be a PSTQ answer of q. Therefore, E_i can be safely pruned.

PSTQ Processing Algorithm

We now discuss how to use the index to perform the PSTQ query. Given a query point q, we traverse the index I by maintaining a queue Q which contains entries E in the form of MBR. In addition, we keep a candidate set S_{cand}, in which candidates can be used to prune other objects.

Figure 9 illustrates the pseudo code of PSTQ query procedure, namely PSTQ_ Processing, which retrieves PSTQ candidates by traversing the index and refines the candidates. Given a PSTQ query with query object q, we first initialize a queue Q and a candidate set S_{cand}. Then, we insert the root, root(I), of the index into queue Q. Every time we pop out a node N from queue Q (lines 4-5), if N is a non-leaf node, for each entry E_i in N, we check if q is contained in E_i.MBR^{β} of entry E_i, where β is set to the smallest pre-computed value that satisfies $\beta \geq 1 - \alpha$. If the answer is yes, we insert E_i into the queue Q for further filtering; otherwise, entry E_i can be safely pruned by probabilistic time $Voronoi$ pruning (lines 6-9).

When N is a leaf node, for each object o_i in N, we check if q is contained in o_i.MBR^{β} of object o_i. If the answer is yes, we add object o_i to S_{cand}, indicating that o_i is a PSTQ candidate; otherwise, o_i can be pruned by probabilistic time *Voronoi* pruning(lines 11-13). Then, we update two (probabilistic) thresholds τ^+ and $\tau^+.\beta$, which are defined as the minimum upper bound and minimum $\beta - \mathrm{upper} - \mathrm{bound}$ of traveling time $\mathrm{t}(q, o_j)$ for all candidates o_j in S_{cand} respectively (line 14). Next, we aim to prune false alarms in S_{cand} by using time bound and probabilistic bound pruning methods (lines 15-21). In particular, for each candidate o_j in S_{cand}, if it holds that $T_j^- > \tau^+$, candidate o_j can be safely pruned (i.e., time bound pruning), where T_j^- is the lower bound of traveling time $\mathrm{t}(q, o_j)$ (lines 15-18). When o_j cannot be pruned, we further check the pruning condition $\mathrm{T}_j^- > \tau^+.\beta$. If this condition holds, o_j can be safely pruned via probabilistic bound pruning(lines 19-21). Finally, we refine candidates in S_{cand} by computing PSTQ probabilities (given in Eq. (1)), and return actual PSTQ answers (line 22).

```
Algorithm PSTQ_Processing {
  Input: index I, query object q, and a probability threshold $\alpha$
  Output: PSTQ answers
  (1)  initialized a queue $Q$ accepting tree nodes $N$
  (2)  $S_{cand} = \emptyset$
  (3)  insert root, $root(I)$, into queue $Q$
  (4)  while $Q$ is not empty
  (5)     node $N$=de-queue $Q$
  (6)     if $N$ is a non-leaf node
  (7)        for each entry $E_i$
  (8)           if $q \subseteq E_i.MBR^{\beta}$          // probabilistic time Voronoi pruning
  (9)              insert $E_i$ into $Q$
  (10)    else       // leaf nodes
  (11)       for each object $o_i$ in $N$
  (12)          if $q \subseteq o_i.MBR^{\beta}$          // probabilistic time Voronoi pruning
  (13)             $S_{cand} = S_{cand} \cup \{o_i\}$
  (14)             update (probabilistic) thresholds $\tau^+$ and $\tau^+.\beta$ in $S_{cand}$
  (15)             for each object $o_j$ in $S_{cand}$
  (16)                let $T_j^-$ be the lower bound of $t(q, o_j)$
  (17)                if $T_j^- > \tau^+$          // time bound pruning
  (18)                   $S_{cand} = S_{cand} - \{o_j\}$
  (19)                else
  (20)                   if $T_j^- > \tau^+.\beta$    // probabilistic bound pruning
  (21)                      $S_{cand} = S_{cand} - \{o_j\}$
  (22) refine candidates in $S_{cand}$ and return actual PSTQ answers
}
```

Figure 9: PSTQ query processing

CHAPTER VI

EXPERIMENTAL EVALUATION

In this section, we empirically evaluate the efficiency and effectiveness of our proposed pruning methods to answer PSTQ queries through extensive experiments over both real and synthetic data sets. Specifically, we use a 2D real data sets, which contains MBRs of 175812 intersection points and 179178 edges of road network of north America. In particular, we randomly generate 1000-10000 facility nodes o_i on the road network, for each object o_i, we first record the shortest path between $o_i$and any other object o_j, where o_i connects to $o_j$directly. Then we obtain probabilistic time $Voronoi$ points m_i^{β} for $\beta \epsilon [0.1, 0.2, \ldots, 1]$ on each shortest path on $path(o_i, o_j)$. We obtain probabilistic time Voronoi point m_i^{β} for all object o_i by this procedure. After that, we generate minimum bounding rectangle MBRs for all objects. And insert these MBRs into R*- tree to enable our PSTQ processing.

We evaluate the performance of our pruning methods in answering PSTQ queries, in terms of CPU time and I/O cost. Specifically, we count the CPU time during the process of PSTQ queries and I/O cost of construction of R*-tree by inserting MBRs into R*-tree. Moreover, we count the number of objects which are pruned by PSTQ query processing. The reported results are the average of 1000 queries.

CHAPTER VII

RELATED WORK

In this section, we overview previous works on query processing over probabilistic/uncertain databases, probabilistic graphs, and certain/uncertain road networks.

Probabilistic and Uncertain Databases

The data uncertainty can be classified into two categories [24], *attribute uncertainty* and *tuple uncertainty*. The attribute uncertainty is usually captured by uncertain databases, whereas the tuple uncertainty is often modeled by probabilistic databases.

A probabilistic database [4] is composed of *x-tuples,* and each *x*-tuple has one or multiple mutually exclusive *alternatives*. one or multiple mutually exclusive alternatives. Each alternative is associated with an *existence probability*, which indicates the probability that the alternative can appear in reality. Query processing in probabilistic databases usually considers the *possible worlds* [4] semantics, where each possible world is materialized instance of the database that may appear in the real world. Existing works on queries over probabilistic databases include range queries or nearest neighbor queries [3], top-k queries [18], and skyline queries [21]

Certain/Uncertain Road Network

There are many existing works on the data model of certain spatial road networks (or graphs), whose road segments (edges) are associated with deterministic weights. One classical query over certain road networks is the shortest path query, which retrieves the shortest path between source and destination points on road networks. Dijkstra [6] proposed the well-known Dijkstra’s algorithm to solve such a problem. In order to improve the query efficiency, several variants have been proposed to heuristically prune the search space [16] or materialize some paths [1, 12]. Huang et al. [11] studied the shortest path search with constraints. Li et al. [17] explored the shortest path queries between source and destination that pass through some types of interesting data points. Apart from the shortest path query, many other query types have been studied in spatial road networks, for example, range queries [20, 13], k-nearest neighbor (kNN) queries [23, 20], reverse nearest neighbor queries [26],multi-source skyline queries [5], and so on. In these works, road networks are usually modeled by certain graphs, and the traveling distances are used as the metric to measure the distance between two points on road networks. In contrast, our PSTQ problem utilizes the traveling time (rather than the traveling distance) as the measurement, and considers uncertain model of road networks (instead of certain ones), which reflects real traffic conditions on road networks. Thus, previous works on certain road networks cannot be directly applied to our PSTQ problem over uncertain road networks.

Ding et al. [7] investigated the shortest path query over time-dependent road networks, which return paths with the smallest total traveling times, but allowing staying time at some facilities, where edges are associated with delay time functions that are certain and varying over time. However, the delay time functions on edges are assumed to be predicted, which, in practice, may not be very accurate. In contrast, our PSTQ problem considers a different data model of road networks with uncertain traffic conditions (time variables) on road segments (rather than certain graphs). Moreover, our problem tackles a different query type, probabilistic shortest time query, instead of the path query.

There are some studies on uncertain road networks that are relevant to our work. Hua and Pei [9] proposed probabilistic path queries on uncertain road networks, and presented three semantics of probabilistic path queries, with respect to the traveling time and probability. Different from the path query type, our PSTQ problem considers the probability shortest time query. Potamias et al. [22] used different metrics such as median distance and majority distance to re-define traditional k-NN queries in uncertain graphs, which assume existence probabilities on edges. Our work considers a different graph model with uncertain time variables on edges (rather than edge existence probabilities), and formalize the query that uses the traveling time (instead of the distance) as the measure.

Probabilistic Graphs

For probabilistic graphs, the existing works [10, 14] modeled the (RDF) graph that contains edges with existence probabilities. That is, edges in the probabilistic graph either exist or do not exist. Other works [8, 25, 19] represented a probabilistic graph by the graphical model, Bayesian network [15], which has uncertain labels. In contrast, our work considers spatial road networks, whose edge weights correspond to uncertain time variables. Moreover, we consider a different query type, that is, PSTQs rather than graph matching queries. Thus, we cannot borrow prior techniques to solve our PSTQ problem.

CHAPTER VIII

CONCLUSIONS

In this paper, we model the road network with actual traffic conditions by uncertain road network (uncertain graph) whose edges are associated with traveling time variables, and formalize the problem of the probabilistic shortest time query (PSTQ). In order to efficiently answer PSTQs, we propose effective pruning methods, time bound pruning and probabilistic bound pruning, to filter out PSTQ false alarms. Moreover, we design offline pre-computation techniques to enable the probabilistic time$Voronoi$ pruning, over which an index can be constructed to facilitate an efficient PSTQ query procedure. Extensive experiments have been conducted to demonstrate the efficiency and effectiveness of our proposed PSTQ approach under various experimental settings.

REFERENCES

[1] R.Agrawal, S. Dar, and H. V. Jagadish. Direct transitive closure algorithms: design and performance evaluation. *TODS*, 15, 1990.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD*, 1990.

[3] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 2003.

[4] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4), 2007.

[5] K. Deng, X. Zhou, and H. T. Shen. Multi-source skyline query processing in road networks. In *ICDE*, 2007.

[6] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, 1959.

[7] B. Ding, J. X. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. In *EDBT*, 2008.

[8] Y. Fukushige. Representing probabilistic relations in RDF, 2005.

[9] M. Hua and J. Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *EDBT*, 2010.

[10] H. Huang and C. Liu. Query evaluation on probabilistic RDF databases. In *WISE*, 2009.

[11] Y.-W. Huang, N. Jing, and E. A. Rundensteiner. Integrated query processing strategies for spatial path queries. In *ICDE*, 1997.

[12] Y. Ioannidis, R. Ramakrishnan, and L. Winger. Transitive closure algorithms based on graph traversal. *TODS*, 18, 1993.

[13] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Path prediction and predictive range querying in road network databases. *The VLDBJournal*, 19, 2010.

[14] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-constraint reachability computation in uncertain graphs. *PVLDB*, 4, 2011.

[15] M. I. Jordan. Graphical models. In *Statistical Science (Special Issueon Bayesian*

Statistics), 2004.

[16] R.-M. Kung, E. Hanson, Y. Ioannidis, T. Sellis, L. Shapiro, and M. Stonebraker. Heuristic search in data base systems. In *Exper tDatabase Systems*, 1986.

[17] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In *SSTD*, 2005.

[18] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1), 2009.

[19] X. Lian and L. Chen. Efficient query answering in probabilistic RDF graphs. In *SIGMOD*, 2011.

[20] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, 2003.

[21] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *VLDB*, 2007.

[22] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios.*k*-nearest neighbors in uncertain graphs. *PVLDB*, 3, 2010.

[23] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh.A road network embedding technique for k-nearest neighbor search in moving object databases.In *GIS*, 2002.

[24] S. Singh, C. Mayfield, R. Shah, S. Prabhakar, S. Hambrusch,J. Neville, and R. Cheng. Database support for probabilistic attributes and tuples. In *ICDE*, 2008.

[25] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. Hellerstein. Bayestore: Managing large, uncertain data repositories with probabilistic graphical models. In *VLDB*, 2008.

[26] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao. Reverse nearest neighbors in large graphs. In *ICDE*, 2005.

BIOGRAPHICAL SKETCH

In 2014, Yaqing Chen received her Master of Science Degree from University of Texas Pan-American, majored in Computer Science. In 2012, she received her Bachelor of Science Degree in Computer Science from Wuhan University of Science and Technology. Her mailing address is 1809 W Schunior St Apt 707, Tx, 78541.