Efficient query processing over uncertain road networks

Bamikole A. Ogundele
University of Texas-Pan American

EFFICIENT QUERY PROCESSING OVER UNCERTAIN ROAD NETWORKS

A Thesis

by

BAMIKOLE A OGUNDELE

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 2015

Major Subject: Computer Science

EFFICIENT QUERY PROCESSING OVER UNCERTAIN ROAD NETWORKS

A Thesis

by

BAMIKOLE A OGUNDELE

May 2015

ABSTRACT

Ogundele, Bamikole A., Efficient Query Processing Over Uncertain Road Networks. Master of Science (MS), May, 2015, 41 pp., 4 tables, 10 figures, references, 30 titles.

One of the fundamental problems on spatial road networks has been the *shortest traveling time query*, with applications such as *location-based services* (LBS) and trip planning. Algorithms have been made for the shortest time queries in deterministic road networks, in which vertices and edges are known with certainty. Emerging technologies are available and make it easier to acquire information about the traffic. In this paper, we consider uncertain road networks, in which speeds of vehicles are imprecise and probabilistic. We will focus on one important query type, *continuous probabilistic shortest traveling time query* (CPSTTQ), which retrieves sets of objects that have the smallest traveling time to a *moving* query point *q* from point *s* to point *e* on road networks with high confidences. We propose effective pruning methods to prune the search space of our CPSTTQ query, and design an efficient query procedure to answer CPSTTQ via an index structure.

DEDICATION

This thesis is dedicated to my sweetheart and loving wife, Adekumbi, and my children, Precious, Cherish, and Emmanuel. My mother-in-law, Yetunde and my mother Abigael. The completion of my master program would not have been possible without the unfailing love, sacrifice and support of my family. I give my deepest expression of love and appreciation for the encouragement that you gave and the sacrifices you made during this graduate program. Thank you for the support and company during late nights of typing and endurance when away from home.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Xiang Lian, chair of my thesis committee, who contributed immensely to the success of my graduate program. He stood by me from the day one of my program to the last day. He went far and beyond including being available for me during the holiday just to make sure that I completed my program. He patiently corrected my errors and provides best guidance for me. Words are not enough to appreciate his effort.

I would also like to thank my thesis committee members, Dr. Christine Reilly, Dr. Timothy Wylie, and Dr. Dong-Chul Kim, for their contributions to this work. They have given me superb scientific guidance, many insightful suggestions and demonstrated a sincere interest in my work. I am fortunate to have such a group of intelligent professor as a team.

I would never forget the help of my classmate, roommate, and a brother, Chiemezie Ukaumunna who has been a source of encouragement to me in completion of my program.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

It has been a major problem determining the shortest traveling time from one location to the other due to some variation in traffic. Finding objects like a gas station, hotel, or coffee shop with the shortest traveling time on a path from a particular source point *s* to a destination point *e* has been a challenge over the decades. One of the fundamental problems is the *shortest traveling time* query, which has many significant applications, such as Web-based mapping services and network routing. As a result, several methods have been developed for shortest time queries in deterministic graphs, in which vertices and edges (associated with edge weights) are known with certainty.

However, in some applications, the road networks may contain some uncertainties. For example, a road network can be modeled as a graph, in which each edge represents a highway, but in reality some highways are likely to have traffic jams and thus be associated with uncertain vehicle speeds.

Therefore, route planning on such road networks should consider uncertain traveling time on roads and provide reasonable paths for users in any circumstances. However, recent GPS-enabled path services ignore the uncertainty inherently contained in traffic, and this uncertainty leads to probabilistic graphs in which edges are labeled with uncertain velocity values (samples) associated with existence probabilities. Moreover, the uncertainty could be as a result of noisy measurements from sensors/experiments, the existence of unstable communication links, or unknown factors such as traffic accidents, and weather conditions (e.g. snow or thunderstorms). The existence probability of an edge can be predicted by some machine learning algorithm, or inferred from historical traffic data.

Figure 1 below shows an example of a road network (graph), which contains 14 nodes that are intersection points of roads $n_1, n_2, \ldots, n_{14}$ denoted as circles, and edges that are road segments, $e_{i,j}$ may not exist between two nodes n_i and n_j $(for\ 1 \le i,j \le 14)$. On road segments (e.g.,$e_{3,4}$), there are several facilities like the gas station and the rest area denoted with a square shape (e.g.,o_1 and o_8, respectively).

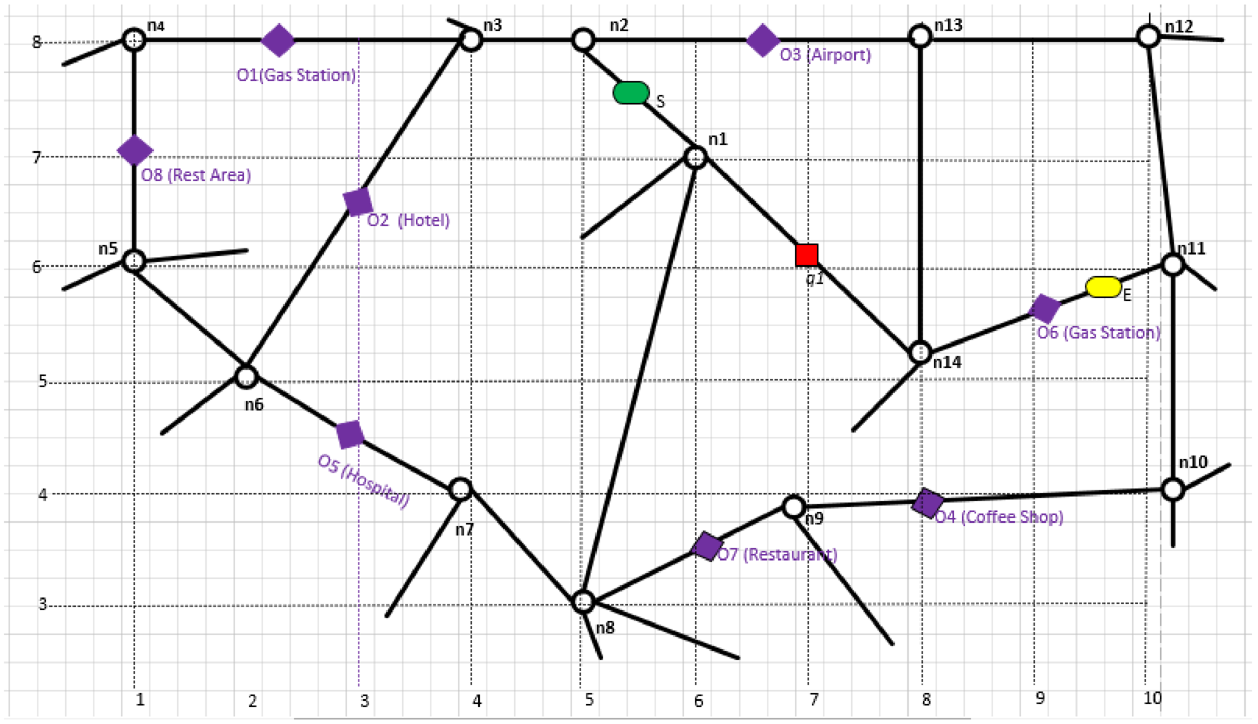

Figure 1: Uncertain Road Network Model

In many real-world applications such as route planning, some people may want to travel from one location to another with the best alternate road. But the shortest traveling time from one point (hotel) to another point (airport) is challenging. Many analysis has shown that the shortest path from one location to another does not necessarily guarantee that it will have the shortest traveling time. There are many factors (bad road, road traffic, accident, stop lights and signs) that could contribute to the fact that the shortest path may take longer traveling time.

In this paper, we will investigate an important type of query, namely *continuous probabilistic shortest traveling time queries* (CPSTTQ), over uncertain road networks, which retrieves points of interest (POIs) or facilities on an uncertain road networks with the shortest traveling time and high confidence, while a given query point q is moving from the source point s to the destination point e on road networks. In order to efficiently process CPSTTQ queries, we propose effective pruning methods, *time bound pruning* and *probabilistic bound pruning*, to reduce the CPSTTQ search space. Even though many works have been proposed for various queries on road networks, most of the prior works did not consider the uncertainty of the traveling time but rather focused on certain shortest traveling path (with the shortest traveling distance). Thus, we cannot borrow previous techniques to tackle our CPSTTQ query.

CHAPTER II

PROBLEM DEFINITION

Uncertainty Model. In this paper, we will analyze the data model of traffic network over an uncertain road network with continuous probabilistic nearest neighbor queries. Several applications have been developed over the years to track the location of an object and calculate the traveling time. For example, GPS Tracking is installed on most phones which use state-of-the-art GPS technology quickly and accurately locate the people, providing real-time location updates using GPS navigation. We will consider the shortest traveling time to reach a nearest neighbor over continuous line segment.

Figure 1 illustrates that there are intersection nodes ($n_1 \sim n_{14}$) and POI objects ($o_1 \sim o_8$) in a road network. The nodes are the intersection points of roads, which are denoted with n_1, n_2, …, n_{14} on the road network, whereas POI objects are facilities, which are denoted with o_1, o_2, …., o_8 corresponding to a gas station, hotel, airport, coffee shop, hospital, restaurant, or rest area on the road network. Previous analyses have always considered the shortest traveling distance from one location to the other based on the belief that the shorter the distance, the faster approach to the destination. However, in reality, the road network may be associated with uncertain traffic information such as uncertain vehicle speeds, which can thus be modeled as an *uncertain road network*.

In particular, each edge in uncertain road networks is associated with an uncertain speed variable, which follows some probabilistic distribution (pdf) (represented by vehicle speed samples). Table 1 shows an example of nodes in uncertain road networks in Figure 1, and Table 2 depicts the edge information of uncertain road network in Figure 1. For example, edge $e_{1,2}$ has 3 possible speed samples, capturing the uncertainty of the vehicle speed on this edge.

Node	**Location (x, y)**	**Node**	**Location (x, y)**
n_1	(6, 7)	n_8	(5, 3)
n_2	(5, 8)	n_9	(7, 4)
n_3	(4, 8)	n_{10}	(10, 4)
n_4	(1,8)	n_{11}	(10, 6)
n_5	(1, 6)	n_{12}	(10, 8)
n_6	(2, 5)	n_{13}	(8, 8)
n_7	(4, 4)	n_{14}	(8, 5)

Table 1: The Nodes of Uncertain Road Networks

Edge $e_{i,j}$	**Distance** dist. (n_i, n_j)	**Velocity** $(e_{i,j})$
$e_{1,2}$	$\sqrt{2}$	(35, 0.3) (36, 0.3) (37, 0.4)
$e_{2,3}$	$\sqrt{1}$	(35, 0.4) (45, 0.2) (50, 0.4)
$e_{3,4}$	3	(55, 0.3) (58, 0.3) (60, 0.4)
$e_{3,6}$	$\sqrt{13}$	(30, 0.6) (32, 0.3) (35, 0.1)
$e_{4,5}$	2	(40, 0.7) (42, 0.2) (45, 0.1)
$e_{5,6}$	$\sqrt{2}$	(36, 0.2) (40, 0.4) (46, 0.4)

$e_{6,7}$	$\sqrt{5}$	(55, 0.3) (58, 0.3) (60, 0.4)
$e_{7,8}$	$\sqrt{2}$	(42, 0.3) (45, 0.3) (50, 0.4)
$e_{8,9}$	$\sqrt{5}$	(30, 0.5) (35, 0.2) (45, 0.3)
$e_{9,10}$	3	(50, 0.3) (55, 0.3) (60, 0.4)
$e_{10,11}$	2	(45, 0.6) (48, 0.1) (55, 0.3)
$e_{11,12}$	2	(32, 0.1) (35, 0.4) (40, 0.5)
$e_{11,14}$	$\sqrt{5}$	(28, 0.5) (38, 0.3) (48, 0.2)
$e_{12,13}$	2	(25, 0.2) (30, 0.1) (35, 0.7)
$e_{13,14}$	3	(40, 0.6) (42, 0.2) (47, 0.2)
$e_{13,2}$	3	(50, 0.3) (55, 0.4) (60, 0.3)
$e_{14,1}$	$\sqrt{8}$	(32, 0.7) (42, 0.1) (52, 0.2)

Table 2: The Edges of Uncertain Road Networks

DEFINITION 1. **(Uncertain Road Network).** *An uncertain road network is define as a Graph $G = (V, E, T)$ where vertices V is a set $n_1, n_2, n_3, \dots, n_{|s|}$, each n_i $(1 \leq i \leq |S|)$ residing at a 2D location $\{x(n_i), y(n_i)\}$, E is a set of edges, where each $e_{i,j}$ in E associated with a velocity variable $Z(e_{i,j})$, and T is a mapping from $V \times V$ to E with edges E that shows the traveling time along the edge e.*

In Definition 1, the variable of a velocity $Z(e_{i,j})$ on each edge $e_{i,j}$ of an uncertain road network follows a probabilistic distribution, represented by discrete velocity samples. In particular, each sample $S_{i,j}$ of random variable $Z(e_{i,j})$ is associated with an appearance probability $S_{i,j}.p(\sum \forall s_{i,j}\ S_{i,j}.p = 1)$.

Example 1. (Uncertain Road Network) In an uncertain road network from location $S\ to\ E$ there are three possible path to arrive at location E from the starting point of location S. The traveling time from all the three paths are different even though they are starting and arriving at the same location, but due to some uncertain traffic network on the road the traveling time varies.

From the example of Figure 1, nodes $n_1, n_2,\ldots, n_{14}$ are the intersection points on road segments of a continuous road network. Table 1 illustrates all the locations of each node n_i $(1 \leq i \leq 14)$, which is described by 2D coordinates. Moreover, $e_{i,j}$ is an edge from node (n_i, 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 distribution function, pdf $(e_{i,j})$ of the velocity variable.

Facility	**Position (x, y)**	**Edge $(e_{i,j})$**
O_1 (Gas Station)	(2, 8)	(3, 4)
O_2 (Hotel)	(3, 6.5)	(3, 6)
O_3 (Airport)	(6.5, 8)	(2, 13)
O_4 (Coffee Shop)	(8, 4)	(9, 10)
O_5 (Hospital)	(3, 4.5)	(6, 7)
O_6 (Gas Station)	(9, 5.5)	(11, 14)
O_7 (Restaurant)	(6, 3.5)	(8, 9)
O_8 (Rest Area)	(1, 7)	(4, 5)

Table 3: Facilities on Uncertain Road Networks

There are many paths from one point, u, to another destination location, v, on the road network. Following, we define the best path to a destination location.

DEFINITION 2 (The Best Path to a Destination) *In a graph $G = (S, T, Y)$, two points $u, v \in V(G)$, and a probability threshold $\alpha \in$ (0, 1), a probabilistic path query (u, v) finds all paths $l \in path(u, v)$, such that $Pr\ (l)\ \geq \alpha$.*

There can be many paths between two vertices in a large graph. Often, a user is interested in only the "best" paths. In this paper, we will consider the best alternative path to a location that provides the shortest traveling time. Given a path, l, from vertex $u\ to\ v$ in an uncertain graph G, l is called the shortest path over G if and only if l is the shortest path in at least one instance of G. The probability that l is the shortest path from $u\ to\ v$ in G is denoted as $Pr\{l\}$.

Example 2. Table 3 illustrates all the facilities points on the $road\ network$ in Figure 1, o_1 (Gas Station), o_2 (Hotel), o_3 (Airport), o_4 (Coffee Shop), o_5 (Hospital), o_6 (Gas Station), o_7(Restaurant), and o_8 (Rest Area). Each facility o_i $(1 \leq i \leq 8)$ has its own location on the road network and its corresponding edge, e.g., the facility Gas Station o_1 is at position (2, 8) on road segment $e_{3,4}$, Hotel o_2 is at position (3, 6.5) on road segment $e_{3,6}$, and so on.

In the example, there are different paths to get to a location on the road network. For example, traveling on this road network from edge e_1 to a facility Hotel $o_{2,}$ there are different paths that lead to the facility. You may travel on the $road\ network$ (RN) from $e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow (o_2)$, or from $e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4 \rightarrow e_5 \rightarrow e_6 \rightarrow (o_2)$, or from $e_1 \rightarrow e_8 \rightarrow e_7 \rightarrow e_6 \rightarrow (o_2)$. We will discuss the best path to get to each location on the road network.

Definition 3 (Probabilistic Shortest Traveling Time Queries, PSTTQ). *Consider a road network RN with a set of facilities $o_1, o_2, \ldots, o_n$ on RN, a static query point q with a probabilistic threshold α, a PSTTQ returns a set of objects o_i that have the shortest traveling time to the static query point q with probability, $Pr_{PSTTQ}(q, o_i)$, $\geq \alpha$, that is,*

$$Pr_{PSTTQ}(q, o_i) = \sum(Pr\{t(q,\ o_i) = T\} \ \prod(1 - Pr\{t(q,\ o_j) \leq T\})\ ) \ \geq \alpha \qquad (1)$$

In Definition 3, we calculate the probability of the traveling time, $t(q, o_i)$ from q to o_i, for each possible value T of $t(q, o_i)$ which will allow us to find the query result with the shortest time to a moving query point q and with high confidence.

The condition below determines the probability that o_i is the PSTTQ, i.e., The probability that the time cost $t(q, o_i)$ equals T and the time costs $t(q, o_i)$ of any other objects are higher than T should be greater than or equal threshold α. Assuming independent velocities on connecting edges, the PSTTQ probability $Pr_{PSTTQ}(q, o_i)$, is given by summing up the probabilities that $t(q, o_i) = T$ and $t(q, o_i) > T$ for other objects o_j over all possible values of T.

Example 3. Given a query point q from location s *to* e on a road network with a probabilistic threshold ≥ α, the PSTTQ query retrieves those objects (e.g. gas stations) from S (o_1, o_6) that have the smallest traveling times with probabilities ≥ α. We compute the traveling time t and its probability p from point q to o_1 and o_6 the traveling time and its probability on each edge are shown in the Table 4.

Path	$\boldsymbol{t(q, n_1)}$	**p**	$\boldsymbol{t(n_1, n_2)}$	**p**	$\boldsymbol{t(n_2, n_3)}$	**p**	$\boldsymbol{t(n_3, o_1)}$	**p**
$Path(q, o_1)$	0.024	0.1	0.031	0.3	0.050	0.3	0.19	0.2
	0.022	0.2	0.046	0.5	0.048	0.3	0.17	0.1
	0.020	0.3	0.056	0.1	0.044	0.2	0.16	0.3
	0.019	0.4	0.053	0.1	0.041	0.2	0.14	0.4
	$\boldsymbol{t(q, n_{14})}$	**p**	**t(n$_{14}$, $\boldsymbol{o_6}$)**	**p**				
$Path(q, o_6)$	0.054	0.6	0.56	0.5				
	0.049	0.4	0.54	0.5				

Table 4: Possible traveling times of road segments on path (q, o_i) from q to gas station

In Table 4, $t(q, n_1)$ records four possible values of the traveling time in minutes on the first edge $e_{q:n_1}$ of the path from $q\ to\ n_1$, and **p** represents their probabilities. Based on Eq. (1) we have $Pr_{PSTTQ}(q, o_1) = 0$, $Pr_{PSTTQ}(q, o_6) = 89.663\%$. Object o_6 has a high probability of having the shortest time to query object q, among all traveling times of the three objects. Thus, object o_6 is one of our $PSTTQ$ query answers.

Definition 4 (Continuous Probabilistic Shortest Traveling Time Queries, CPSTTQ). *Consider a road network RN with a set of facilities $o_1, o_2, \dots, o_n$ on RN, a moving query point q (moving from source point s to destination point e), and a probabilistic threshold α, a continuous probabilistic shortest traveling time query (CPSTTQ) returns sub-paths of moving query point q between s and e on the road network, each associated with a set of objects o_i that have the shortest traveling time to query point q within this sub-path with probability, $Pr_{PSTTQ}(q, o_i) \geq \alpha$, where $Pr_{PSTTQ}(q, o_i)$ is given by Eq. (1).*

As an example, in Figure 2, we have a query point q moving along three connected sub-paths, $p_1, p_2,$ and p_3, with a starting location $s(5,7.5)$, through nodes n_1 and n_{14}, and ending at location $e(10,6)$. The CPSTTQ retrieves these three sub-paths because the query point moves from point s to e, and each sub-path is associated with an object set such that any object o_i in the set has the shortest traveling time $t(q, o_i)$ to the moving query point q in this sub-path with probability greater than α.

From Eq. (1) in Definition 3 we can define the condition of the CPSTTQ answer by*:* ∀ q in the sub-path of *path*(*s*, *e*),

$$Pr_{CPSTTQ}(q, o_i) = \sum \left(Pr\{t(q,\ o_i) = T\} \prod (1 - Pr\{t(q,\ o_j) \le T\}) t(s, o_i),\ t(e, o_i)\right) \ge \alpha \quad (2)$$

To find the query result with the shortest traveling time to point q, we calculate the traveling time $t(q,\ o_i)$ from $q\ to\ o_i$, $t(s,\ o_i)$ from $s\ to\ o_i$ and $t(e,\ o_i)$ from $e\ to\ o_i$ and its probability for each possible value of T of the traveling time. The shortest traveling time to a moving query point q is then determined within each sub-path on the road network RN and each sub-path is associated with an object set which is the answer to that query point q.

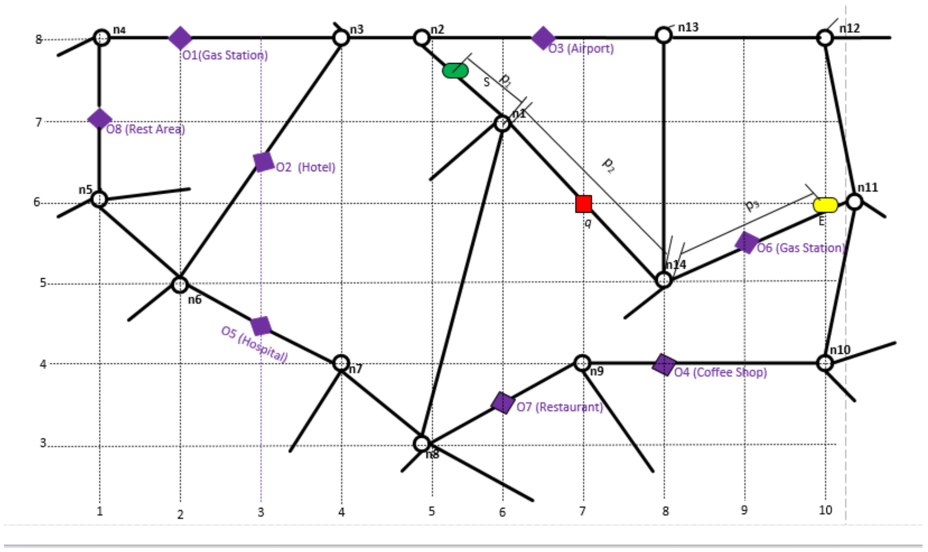

Figure 2: Uncertain Road Network Model with Query Point

Example 4. In the example of Figure 2, consider a moving query point q on a road networks with a starting point $s(5, 7.5)$ through a path n_1, n_{14} and ending point $e(10, 6)$ location. There are three sub-paths $p_1, p_2,$ and p_3 of the path, *path*(s, e), from the starting point s to the ending point e. Each sub-path p_i is associated with a set of objects such that the object set is the query answer to query point q on this sub-path. That is, no matter where the query point q is on the sub-path p_i, the object set within that sub-path is the PSTTQ answer set of q.

Figure 2 illustrates an example on a road network with a moving query point q from starting point s to the ending point e. As the query point q moves on a road network in any of the sub-paths $p_1, p_2,$ and p_3, the object set contains objects with the shortest traveling time to the query point q with probability $\geq \alpha$.

CHAPTER III

PRUNING METHODS

The straightforward method to answer the CPSTTQ is to compute the PSTTQ answers for any *static* query point q on the path from s to e, which is quite inefficient. Therefore, in this section, we propose effective pruning methods to reduce the search space of the CPSTTQ problem. In particular, our goal is to retrieve a small candidate set of CPSTTQ answers for any particular query sub-path (on path(s, e)), by filtering out bad candidates.

3.1 Time Bound Pruning

In this subsection, we propose a time bound pruning mechanism which eliminates objects that are not CPSTTQ query answers. We consider the intervals of two traveling times on road networks $t(q, o_i)$ and $t(q, o_j)$. The basic idea of the time bound pruning is as follows. This time bound pruning method utilizes the lower/upper bounds of the traveling time to enable pruning. Specifically, if the upper bound of the traveling time $t(q, o_j)$ is less than the lower bound of the traveling time $t(q, o_i)$, then we can conclude that o_i is not a query answer to the CPSTTQ due to the existence of o_j. This way, we can eliminate objects from a facility set, $M = (o_1, o_2, \ldots, o_n)$ by using this pruning method.

LEMMA 1. (**Time Bound Pruning**). *Assume that lower bound $lb\_t(x, y)$ and upper bound $ub\_t(x, y)$ are the bounds of the traveling time from location x to location y on road networks, respectively. Given a candidate object o_i, we can safely prune*

o_i *from* $M(set\ of\ facilities)$, *if* $lb\_t(q, o_i) > ub\_t(q, o_j)$ *assuming that*

$t(q, o_i) \epsilon [lb\_t(q, o_i), ub\_t(q, o_j)]$ *and* $t(q, o_j) \epsilon\ [lb\_t(q, o_i), ub\_t(q, o_j)]$ *for* $o_i\ , o_j\ \epsilon M$.

PROOF.

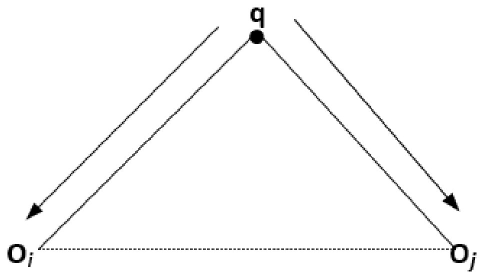

Figure 3: Illustration of Time Bound Pruning

We prove that we are able to prune object o_i from the facility set *M* as follows:

Assume that: $t(q, o_i) \epsilon [lb\_t(q, o_i), ub\_t(q, o_i)]$ and $t(q, o_j\ )\ \epsilon [lb\_t(q, o_j), ub\_t(q, o_j)]$, where $[lb\_t(q, o_i), ub\_t(q, o_i)]$ represent the traveling time interval of $t(q, o_i)$ from q to o_i and $[lb\_t(q, o_j), ub\_t(q, o_j)]$ is the traveling time interval of $t(q, o_j)$ from q to o_j. By using the transitivity of the inequality, we have: $t(q, o_i) \leq\ ub\_t(q, o_i)$ and $t(q, o_j) \leq\ ub\_t(q, o_j)$

Since the condition that $t(q, o_i) \geq\ lb\_t(q, o_i)\ >\ ub\_t(q, o_j)\ \geq t(q, o_j)$ holds, therefore, we can obtain $t(q, o_i) >\ t(q, o_j)$. This result shows that o_i has higher traveling time from q than o_j, which implies that it is not a CPSTTQ answer. Hence, object o_i can be safely pruned from the facility set *M*.

Lemma 1 allows us to use a time bound pruning method to filter out bad candidates via lower and upper bounds of the traveling time on a road network. Intuitively, we use Lemma 1 to prune object o_i from the set of objects from the facilities *M*.

The Computation of Lower/Upper Bounds for the Traveling Time. To use the time bound pruning method, we need to compute the lower and upper bounds of the traveling time. For instance, consider the road network containing a path $s \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_k$, where s is the starting point and n_k is an ending point of a segment on the road network. This segment on the road network has different sub-paths with nodes n_f, n_g which belong to path (q, o_i). Therefore, the velocity of the vehicle in this segment $e_{f,g}$ on road networks is given by $v \in [v^-(n_f, n_g), v^+(n_f, n_g)]$. Thus, we have the lower and upper bounds of the traveling time from n_f to n_g on edge $e_{f,g}$, denoted as $lb\_t(n_f, n_g)$ and $ub\_t(n_f, n_g)$, which are given by $lb\_t(n_f, n_g) = dist(n_f, n_g)/v^+(n_f, n_g)$ and $ub\_t(n_f, n_g) = dist(n_f, n_g)/v^-(n_f, n_g)$, respectively. Thus, the summation of $lb\_t(n_f, n_g)$ for all edges $e_{f,g}$ on path (q, o_i) is the lower bound of the time cost $lb\_t(n_f, n_g)$ of path from $q\ to\ o_i$, whereas the summation of $ub\_t(n_f, n_g)$ for all edges $e_{f,g}$ on path (q, o_i) is the upper bound of the time cost $ub\_t(n_f, n_g)$ of path from $q\ to\ o_i$.

LEMMA 2. (**Lower and Upper Bounds of the Traveling Time**). *The lower bound of the traveling time from a moving query point q on any subpath of a road network from a location st to location ed is given by:*

$$lb\_t(q, o_i) = \min\{lb\_t(st, o_i), lb\_t(ed, o_i)\} \qquad (3)$$

The upper bound of the traveling time of a moving query point q on the subpath of a road network from location s to location e is given by:

$$ub\_t(q, o_i) = \max\{ub\_t(st, ed) + ub\_t(st, o_i), ub\_t(st, ed) + ub\_t(ed, o_i)\} \qquad (4)$$

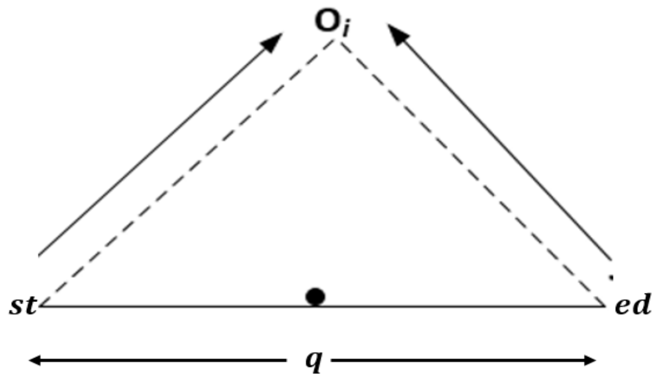

Figure 4: A Query point *q* on a sub-path of a Road Network

PROOF.

We want to prove that Eq. 3 gives the minimum possible traveling time from q to o_i. In particular, we consider two possible paths from q to o_i. Assume that query point q, is on a road segment from node st to node ed. Then, the path from q to o_i may go through either node st (i.e., $q \rightarrow st \rightarrow \ldots \rightarrow o_i$) or node ed (i.e., $q \rightarrow ed \rightarrow \ldots \rightarrow o_i$). We have the following inequality with respect to the traveling time, $t(q, o_i)$, from q to o_i.

$$t(q, o_i) \geq \min\{(lb\_t(q, st) + lb\_t(st, o_i)), (lb\_t(q, ed) + lb\_t(ed, o_i))\}.$$

Since we have:

$$lb\_t(q, st) \geq 0 \; and \; lb\_t(q, ed) \geq 0$$

by the inequality transition, we can obtain

$$t(q, o_i) \geq \min\{lb\_t(st, o_i), lb\_t(ed, o_i)\},$$

which is exactly equal to the RHS of Eq. 3 (i.e., $lb\_t(q, o_i)$). Thus, $\min\{lb\_t(st, o_i), lb\_t(ed, o_i)\}$ is the lower bound of the traveling time $t(q, o_i)$.

Similarly, we can also prove the correctness of the upper bound, $ub\_t(q, o_i)$, of the traveling time from q to o_i in Eq. 4. Specifically, similar to the proof of the lower bound, we consider two paths through st and ed, respectively, on path $path(q, o_i)$, and derive the following inequality w.r.t. time upper bounds.

$$t(q, o_i) \leq \max\{ub\_t(q, st) + ub\_t(st, o_i), ub\_t(q, ed) + ub\_t(ed, o_i)\}.$$

Next, since we have:

$$ub\_t(q, st\ ) \leq ub\_t(st, ed)\ and\ ub\_t(q, ed\ ) \leq ub\_t(est, ed),$$

by the inequality transition, we can obtain:

$$t(q, o_i) \leq \max\{ub\_t(st, ed) + ub\_t(st, o_i), ub\_t(st, ed) + ub\_t(ed, o_i)\},$$

which is exactly equal to the RHS of Eq. 4 (i.e.,$ub\_t(q, o_i)$).

Thus, $\max\{ub\_t(st, ed) + ub\_t(st, o_i), ub\_t(st, ed) + ub\_t(ed, o_i)\}$ is the upper bound of the traveling time $t(q, o_i)$.

Intuitively, to compute traveling time bounds from q to o_i, we consider two possible paths. The first path is $q \rightarrow st \rightarrow \ldots \rightarrow o_i$ and the other way is $q \rightarrow ed \rightarrow \ldots \rightarrow o_i$. The traveling time from q to o_i through the path st is greater than that from st to o_i . Similarly, the traveling time from $q\ to\ o_i$ through the path ed is greater than that from $ed\ to\ o_i$. Therefore, the traveling time from $q\ to\ o_i$ can be lower bounded by the minimum between the traveling time from ed to o_i and that from st to o_i.

Similarly, to compute the upper bound of the traveling time from $q\ to\ o_i$, we also consider two possible paths. The traveling time from q to o_i through the path st is smaller than that from ed to o_i (through *st*). The traveling time from $q\ to\ o_i$ through the path ed is greater than that from $st\ to\ o_i$ (through ed). Therefore, the traveling time from $q\ to\ o_i$ can be upper bounded by the maximum between the traveling time from st → ed → … →o_i and that from ed → st → … →o_i.

3.2 Probabilistic Bound Pruning

In this subsection, we will consider probabilistic information and propose an effective probabilistic bound pruning method to filter out false alarms. For continuous queries, we will take into account all possible locations within the sub-paths as the q moves along the road network from $s\ to\ e$. Specifically, we mainly focus on the shortest traveling time $t(q, o_i)$ on a sub-path, path (q_1, q_2), within a road network as the query point q moves from one point q_1 to another one q_2. We introduce a way to prune those false alarms o_i that have low CPSTTQ probabilities (as given in Eq. (2)).

DEFINITION 5 (*β*-Upper-Bound). *Let the traveling time* $t(q, o_i)\epsilon[lb\_t(q, o_i), ub\_t(q, o_j)]$ *for* $o_i \in M(set\ of\ facilities)$, a *β-upper-bound is denoted as* $(ub\_t(q, o_i).\beta)$, which satisfies $\Pr\{t(q, o_i) \in [lb\_t(q, o_i), ub\_t(q, o_i).\beta]\} = \beta$.

The β-upper-bound, $ub\_t(q, o_i).\beta$, is given by an upper bound of the traveling time $ub\_t(q, o_i)$ such that $t(q, o_i)$ is the shortest traveling time within the interval $[lb_{t(q, o_i)}, ub\_t(q, o_j).\beta]$ with probability β.

LEMMA 3. (**Probabilistic Bound Pruning**). Assuming that $t(q, o_i) \in [lb\_t(q, o_i),$ $ub\_t(q, o_i)]$ *and* $t(q, o_j) \in [lb\_t(q, o_j), ub\_t(q, o_j)]$ *for* o_i , o_j $\in$ M, *if it holds that* β *>1-*α *and* $lb\_t(q, o_i) >$ $ub\_t(q, o_j).\beta$, *then object* o_i *can be safely pruned from M* $(set\ of\ facilities)$.

PROOF. We want to prove the correctness of our probabilistic bound pruning, by using the diagram in Figure 5. Specifically, for $o_i, o_j \in M, t(q, o_i)$ *and* $t(q, o_j)$ are within the intervals of two traveling times, $lb\_t(q, o_i), ub\_t(q, o_i)$ and $lb\_t(q, o_j), ub\_t(q, o_j)$ respectively.

Assume that, $ub\_t(q, o_j).\beta$ *is a* β-upper-bound of $t(q, o_j)$ which satisfies $Pr\ \{t(q, o_j) \in [lb\_t(q, o_j), ub\_t(q, o_j).\beta]\}= \beta$. The $ub\_t(q, o_j).\beta$ from Definition 5 indicates that the $Pr(q, o_j)$ of an object is ≥ 1. Since the probability of o_i has the shortest traveling time to q, therefore:

$$Pr_{CPSTTQ}(q, o_i) \leq \Sigma\ (Pr\{t(q, o_i)=T\} \cdot \Pi\ (1- Pr\{t(q, o_j) \leq T\})$$

$$\leq \Sigma\ Pr\{t(q, o_i)=T\} \cdot ((1- Pr\{t(q, o_j) \leq T \mid t(q, o_j) \in [lb\_t(q, o_j), ub\_t(q, o_j).\beta]\})$$

$$\cdot\ Pr\{t(q, o_j) \in [lb\_t(q, o_j), ub\_t(q, o_j).\beta]\}$$

$$+ (1- Pr\{t(q, o_j) \leq T \mid t(q, o_j) \in [ub\_t(q, o_j).\beta, ub\_t(q, o_j)]\})$$

$$\cdot\ Pr\{t(q, o_j) \in [ub\_t(q, o_j).\beta, ub\_t(q, o_j)]\})$$

$$= \Sigma\ Pr\{t(q, o_i)=T\} \cdot ((1- Pr\{t(q, o_j) \leq T \mid t(q, o_j) \in [lb\_t(q, o_j), ub\_t(q, o_j).\beta]\}) \cdot \beta$$

$$+ (1- Pr\{t(q, o_j) \leq T \mid t(q, o_j) \in [ub\_t(q, o_j).\beta, ub\_t(q, o_j)]\}) \cdot (1-\beta))$$

Based on the lemma assumption that $lb\_t(q, o_i) > ub\_t(q, o_j).\beta$ and the definition of β-upper bound, we have:

$$T = t(q, o_i) \geq lb\_t(q, o_i) > ub\_t(q, o_j).\beta.$$

Then, in the formula above, we infer that:

$$Pr\{t(q, o_j) \leq T \mid t(q, o_j) \in [lb\_t(q, o_j), ub\_t(q, o_j).\beta]\} = 1.$$

Moreover, since probability is non-negative, we have:

$$Pr\{t(q, o_j) \leq T \mid t(q, o_j) \in [ub\_t(q, o_j).\beta, ub\_t(q, o_j)]\} \geq 0.$$

As a result, we can obtain:

$$Pr_{CPSTTQ}(q, o_i) \leq \Sigma\, Pr\{t(q, o_i)=T\} \cdot (1\text{-}\beta) = 1\text{-}\beta.$$

We can thus conclude that (1-β) is an upper bound of $Pr_{CPSTTQ}(q, o_i)$. If $\beta > 1 - \alpha$, that is, $1 - \beta < \alpha$, by the inequality transition, we can obtain $Pr_{CPSTTQ}(q, o_i) < \alpha$. In other words, the probability $Pr_{CPSTTQ}(q, o_i)$ that object o_i has the shortest traveling time to q on the sub-path must be less than α, and object o_i cannot be the CPSTTQ answer. Hence, we can safely prune object o_i from M.

How to compute the $\boldsymbol{\beta}$-Upper-Bound. To enable the probabilistic bound pruning, we need to compute the β-upper-bound. Assuming that $t(q, o_i) \in [lb\_t(q, o_i), ub\_t(q, o_j)]$ *and* $t(q, o_j) \in [lb\_t(q, o_i), ub\_t(q, o_j)]$, then $t(q, o_i)$, the traveling time on any subpath (p_1, p_2, p_3) within the road segment such that we are able to satisfy $\Pr\{t(q, o_i) \in (lb\_t(q, o_i), ub\_t(q, o_j)\ .\beta)\} = \beta$.

Let us assume that there are e edges on any subpath within the road network which is upper bounded by the interval $[lb\_t_e(q, o_i), ub\_t_e(q, o_j)]$ such that the traveling time is within this probability.

Greedy Algorithm to Compute the β-Upper-Bound. Here, we are getting a very small β-upper-bound of the traveling time on each sub-path on a road network segment by not considering all the sample combinations but very specific with few sample combinations to be considered. We vary the length and progressively extend our greedy algorithm paths, u, from 1 to l, and will then compute $M(ub\_t_\beta(l), \beta)$ with path (q, o_j).

Initial Case. P_1 is the first subpath from location $s\ to\ e$ on the road network segment with length $u = 1$ and edge of path with $path(q, o_j)$ that we first consider. Assuming arr_t_d is a sorted array of size for c_1, which contains c_1 discrete time samples in ascending order of the traveling time on P_1 . For every time sample $arr\_t_d[i]\ (1 \leq i \leq c_1)$, which is associated with probability $arr\_t_d[i].p$. If the $ub\_t_\beta(1) = ub = arr\_t_d[i]$ holds, since the corresponding β value is given as $z_1 = \sum_{j=1}^{i} arr\_t_d[j].p$ the probability that the traveling time on P_1 is not greater than $arr\_t_d[j]$. From c_1 traveling time samples from array arr_t_d, randomly select M and denote it as $t_d^{(a)},\ t_d^{(b)}, \dots, t_d^{(n)}$ therefore $t_d^{(a)} \leq\ t_d^{(b)}$ and so on.

Progressive Computation. For P_k, a subpath of $path(q, o_j)$ has length $k\ (i.e., u\ =\ k)$, we can obtain two arrays, arr_ub_k and arr_β_k, which contain M β-upper-bounds and M corresponding β values for subpath P_k (in ascending order), respectively. When we consider the subpath, $P_k + 1$, of path $path(q, o_j)\ \ (u\ =\ k\ +\ 1)$, we combine the β-upper bound on subpath P_k with that of the $(k\ +\ 1)$ −th edge on $path(q, o_j)$.

Therefore we obtain combinations of M β-upper-bound $arr\_ub_k\ [i]$, for subpath P_k, and $C_k + 1$ traveling time samples, t_{k+1}^{j} on the $(k + 1)$ −th edge of the path. Each combination results in β-upper-bounds, $arr\_ub_k\ [i] + arr\_t_{k+1}[j]$, and its β value $arr\_\beta_k[i].z_{k+1}^{(j)}$, where $z_{k+1}^{(j)} = \sum_{r=1}^{j} t_{k+1}^{(r)}.p$, for $1 \leq i \leq M$ and $1 \leq j \leq c_{k+1}$. M out of $(m.c_{k+1})$ combinations where $\geq 1 - \alpha$. The β-upper-bounds and their values in arrays arr_ub_{k+1} and arr_β_{k+1} could be store pairs, therefore we are able to derive arrays, arr_ub_i and $arr\_\beta_{i,}$ for path $path(q, o_j)$ of length l. $ub\_t_\beta(q, o_j)$ smallest value $arr\_ub_i[i]$, as the β-upper-bound, where $arr\_\beta_{i,}[i] \geq 1 - \alpha$, for $1 \leq i \leq l$.

Algorithm *Compute_Beta_UB{*

Input*: arrays arr_t_u ($1 \leqslant u \leqslant l$)*

Output*: arrays arr_{ub1} and arr_β_i*

```
(1) randomly select M samples, from $C_1$ traveling time samples from array, $arr_{ti}$
(2) let the array $arr\__{ub1}$ (i)= $t_i$ for $M \geqslant I \geqslant 1$
(3) let the array $arr_{\beta i}$ (i)=z for $M \geqslant I \geqslant 1$
(4) for k=1 to l-1
(5)     for each of the ($M.P_{(k+1)}$ ) where $i \in [1,M]$ and $j \in [1, P_{(k+1)}]$
(6)             compute the $\beta$-upper-bounds ($arr\__{ubk}(i)$+ $arr\_t_{k+1}(j)$)
(7)     randomly select M out of ($M.P_{(k+1)}$)
(8)     store M pairs to arrays $arr\__{ub1\ k+1}$ and $arr_{\beta k+1}$ )
(9) return arrys $arr\__{ubi}$ and $arr\__{\beta i}$
```

}

Figure 6-Algorithm for computing β-upper-bound

Algorithm of Computing β-Upper-Bound. Figure 6 demonstrates the pseudo-code for computing the M β-upper-bound of the traveling time on path $path(q, o_j)$ where $\beta \geq 1 - \alpha$. Since we have subpaths P_1, P_2, P_3 we begin from the first edge P_1 on path $path(q, o_j)$ and randomly selected M out of C_1 traveling time samples from array arr_t_i and updates β arr_β_1 array and β-upper-bound arr_ub_1 each time. The P_k subpath is extended by adding the $(k + 1)$-th edge on path $path(q, o_j)$ by considering $m.c_{k+1}$ combinations of m β-upper-bounds of possible traveling times on path P_k and c_{k+1} time samples on the $(k + 1) -$th edge, then calculate their corresponding pairs of β-upper-bounds and β values for the path P_{k+1}. We also randomly select M out of $m.c_{k+1}$ combinations satisfying $\beta \geq 1 - \alpha$, and update the arrays arr_ub_{k+1} and arr_β_{k+1} with $\beta -$ upper-bounds and β values for path P_{k+1}. The iterations $(l - 1)$, for path P_l therefore, we return the M β-upper-bounds in arr_ub_l and β values in arr_β_l.

CHAPTER IV

CPSTTQ PROCESSING

In this section, we consider building an index over offline pre-computed data, in terms of *Minimum Bounding Rectangle* (MBR), from each object o_i which could lead to improved efficiency of CPSTTQ processing.

4.1 Index Construction

Data Structure of the Index. Firstly, we consider constructing a spatial index R-tree over the *Minimum Bounding Rectangles* (MBRs) of objects o_i and their pre-computed data, which can be used to retrieve CPSTTQ answers. We use the example in Figure 6 to illustrate our index structure over 8 objects, o_1, o_2, ..., o_8. Specifically, we first pre-compute the neighbors (breadth-first search) of each object in the road networks, and then use an MBR to bound this object and its neighbors. Thus, we can obtain 8 MBRs for 8 objects, respectively. Figure 6 illustrates the R-tree index over these 8 MBRs, which are inserted into the R-tree with the normal insertion method of the R-tree code. The R-tree index is a tree structure, which contains both leaf and non-leaf nodes.

Leaf Nodes. Each of the leaf nodes has entry A_i that stores octuplet with objects o_1 to o_8. MBRs of objects are stored in each leaf node, each of which tightly bounds objects o_i and their corresponding neighbors in the road network. Here, the reason that we use MBRs to bound objects and their neighbors is that we can utilize these MBRs to retrieve CPSTTQ candidates.

Intuitively, if a path, *path*(s, e), intersects with MBRs (including objects and their neighbors on road networks),then these MBRs might be CPSTTQ candidates (i.e., PSTTQ answers of query point) for query point moving from s to e.

In the example of Figure 6, there are 4 leaf nodes, A_4 ~ A_7, each of which contains MBRs of two objects. For instance, leaf node A_4 contains MBRs of two objects, o_2 and o_5. In particular, the neighbors of object o_2 are o_5 and o_7. Thus, the MBR of o_2 tightly bounds objects o_2, o_5 and o_7.

Non-Leaf Nodes. For non-leaf nodes, we recursively use MBRs to bound MBR of their child nodes on a lower level. In the example of Figure 6, we have non-leaf nodes A_2 and A_3, which contain pointers, pointing to child nodes A_4, A_5 and A_6, A_7, respectively. In particular, the MBR of non-leaf node A_2 tightly bounds child entries A_4 and A_5 under A_2. The case of A_3 is similar. Furthermore, the root of the tree, A_1, recursively bounds MBRs of A_2 and A_3.

The Construction of the Tree Structure. We are interested in showing our pre-computed minimum value from object (o_i, o_j) to its nearest neighbor by inserting the Minimum Bounding Rectangles (MBR) in the R-tree. Figure 7 illustrates how we determine the minimum values to objects o_i and give more explanation on the construction of the tree index. We are able to insert Minimum Bounding Rectangles (MBR) of the object o_i into the R-tree tightly bounded to its nearest neighbor through the standard insertion method to be able to locate the nearest neighbor and we considered all the nearest neighbor and all points associated within our Minimum Bounding Rectangles (MBR). To insert an object, the tree is traversed recursively from the root node. At each step, all rectangles in the current directory node are examined, and a candidate is chosen by choosing the rectangle which requires least enlargement. The search then descends into this page, until reaching a leaf node. If the leaf node is full, it must be split before the insertion is made

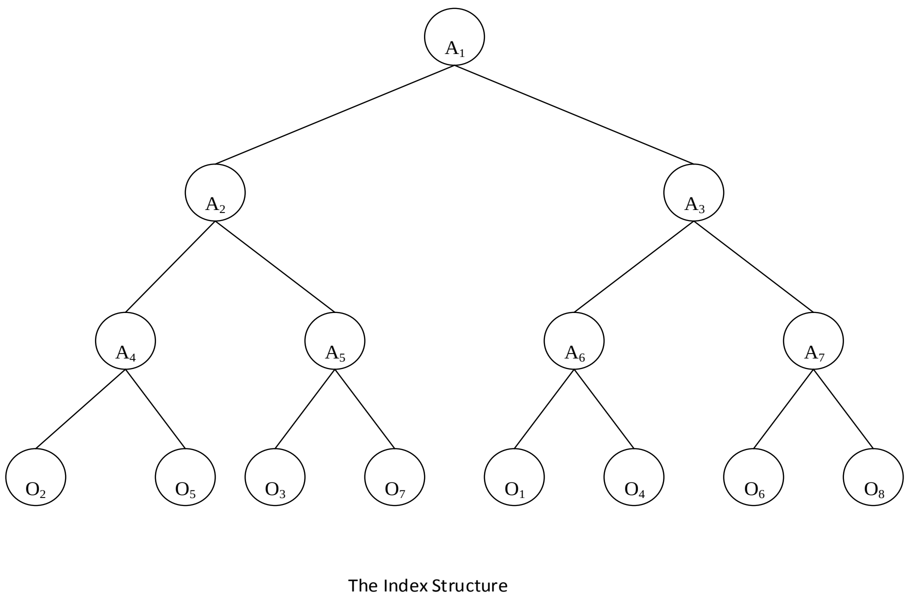

Figure 7- The Index Structure

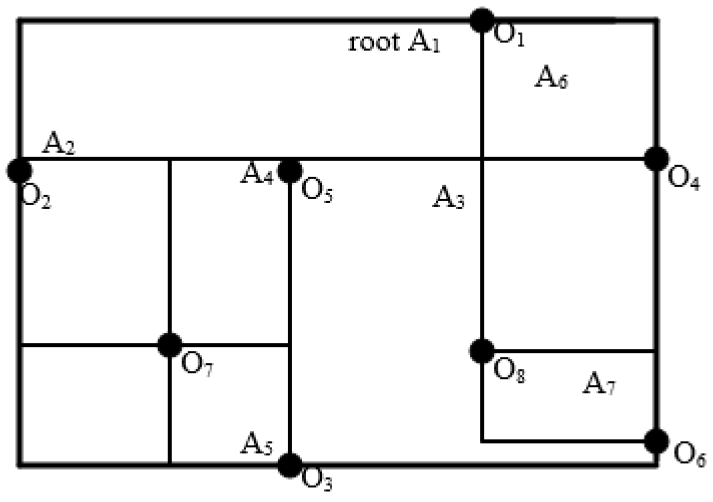

Figure 8- Minimum Bounding Rectangles (MBR)

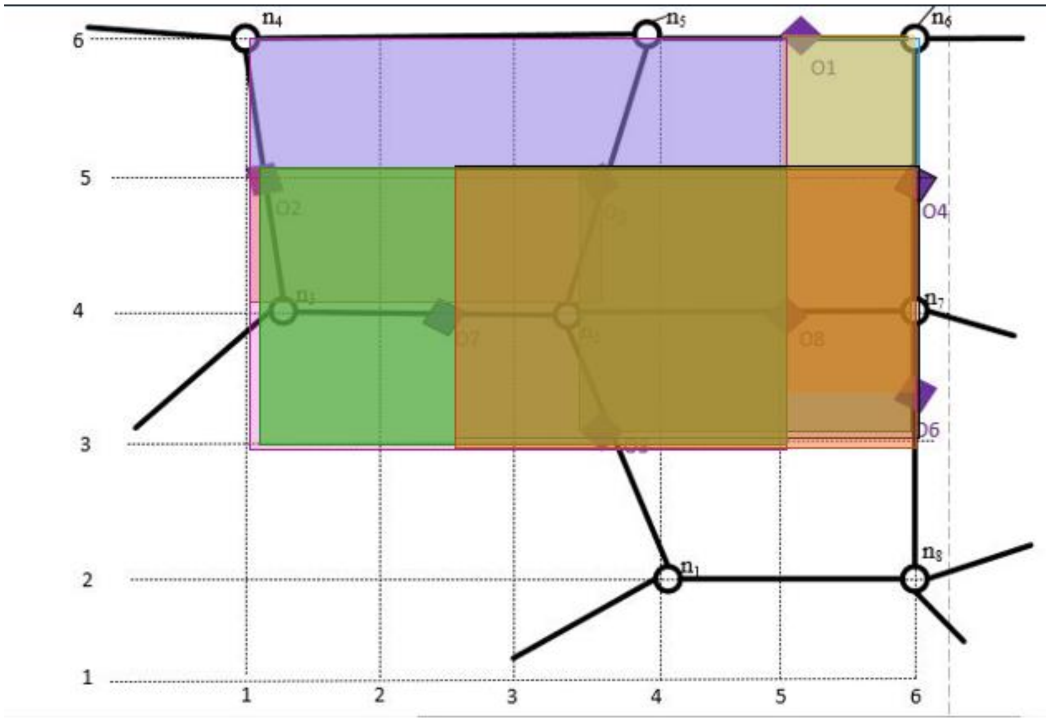

Figure 9-Offline pre-computation

4.2 Query Procedure

Pruning with the Index. We are able to use the pruning method with the R-tree index to reduce the search space, and retrieve the CPSTTQ candidates by traversing the tree index. We use the example in Figure 6 to illustrate our basic idea. From the root node A_1 in the index tree, we begin to check the MBR of each child node (i.e., $A_2 \sim A_7$) and the path, *path*(*s*, *e*), of moving query point *q* from *s* to *e*. When the query path, *path*(*s*, *e*), is not intersecting with node A_i, then we can safely prune all objects *o* under this node A_i. This is because the CPSTTQ answers can only be those neighbor objects, *NN*(o_i), of each object *o* under A_i, but not object o_j (i.e., the traveling time from *q* on path *path*(*s*, *e*) to neighbor *NN*(o_i) is always greater than or equal to that from *q* to o_j, since *q* is not in the MBR of object o_j). Thus, we are able to safely prune objects o_i under A_i. On the other hand, if an index node A_i intersects with the query path *path*(*s*, *e*), then we cannot prune this node

A_i, and we need to further check its children to retrieve CPSTTQ candidates through the index traversal.

LEMMA 4. *(Index Pruning). Given a query point q moving along a path, path(s, e), from location s to location e, and Minimum Bounding Rectangle (MBR) of a node A_i, we can safely prune A_i, if the MBR,MBR (A_i), of node A_i intersects with path(s, e).*

CPSTTQ Processing Algorithms. In the sequel, we will discuss how to use the index to answer the CPSTTQ query. Given a path, *path*(*s*, *e*), of a moving query point *q* from *s* to *e*. First, calculate the shortest path from point *s* to *e* for query point *q*, and then check if this shortest path, *path*(*s*, *e*), intersect the Minimum Bounding Rectangles (MBR), *MBR (A_i)*, and then traverse the index *I* by maintaining a queue *Q*.

For each non-leaf node, check each child node whether the shortest path, *path*(*s*, *e*), is intersecting with the corresponding MBR of A_i (lines 7-9) in algorithm CPSTTQ processing. If they intersect with each other, we will add it to the queue (line 10). But if not, we will ignore it, because it cannot contain any CPSTTQ candidates within the MBR.

For the leaf node that contains objects o_i, we will check one by one if we are able to apply our proposed pruning methods (lines 12-13). Then, we will do the pruning for each object o_i. We first use the index pruning to see whether or not *path*(*s*, *e*) intersects with *MBR*(o_i) (line 13). If the answer is yes, then o_i is a potential CPSTTQ answer. We will further check whether o_i can be pruned by the time bound pruning or probabilistic bound pruning. If o_i survives, then we will add it to the candidate set S_{cand} (lines 14-15). Finally, we will refine the candidates in the set S_{cand} by computing actual CPSTTQ answers (line 16).

Algorithm *CPSTTQ_Processing*

Input: *index I, a query path, path(s, e), from s to e, and a probability threshold α*

Output: *CPSTTQ Answer*

```
(1)  initialize a queue Q accepting the tree nodes
(2)  $S_{cand}=\emptyset$; $r=r_\beta=+\infty$
(3)  calculate the shortest path from s to e
(4)  insert root, into queue Q
(5)  while Q is not empty
(6)      node A=de-queue Q
(7)      if A is a non-leaf node
(8)          for each entry $A_i$ of A
(9)                if path(s, e) intersects with $A_i$.MBR
(10)                   insert $A_i$ into queue Q
(11)   else
(12)         for each object $o_i$ in leaf node $A_i$
(13)               if path(s, e) intersects with $o_i$.MBR
(14)                 if $o_i$ cannot be pruned by time bound pruning and probabilistic bound pruning
(15)                    add to candidate set to $S_{cand}$
(16) refine candidates in $S_{cand}$ and return the actual CPSTTQ answers.
```

Figure 10: CPSTTQ Query Processing Algorithm

CHAPTER V

RELATED WORK

The study from this paper is related to some previous works on probabilistic shortest time queries over uncertain databases, continuous nearest neighbor, path queries on probabilistic graphs and on uncertain road networks.

Uncertain Databases and Continuous Nearest Neighbor. Probabilistic databases [CP1987] can store types of information that cannot be represented using the relational model. Probabilistic databases may also be viewed as generalizations of relational databases; any relational database can be represented without loss of information by a probabilistic database. Cavallo and Pittarelli [CP1987] defined a new project-join mapping for relational databases that is based on transforming a relational to a probabilistic database. Probabilistic databases [Z1997], are formalized by means of logic theories based on a probabilistic first-order language proposed by Halpem.

Probabilistic shortest time queries considers the uncertainty of traveling times in uncertain road networks, it considers the probabilistic path problem of traveling time from query point q to a facility, with the shortest traveling time. A probabilistic database [DS2007] comprises of a list of x-tuples and each x-tuple has one or more exclusive alternatives in which each of these alternatives is associated with an existence probability. [CKP2003] studied probabilistic query evaluation based upon uncertain data. They proved that classification of queries is made based upon the nature of the result set and thereby they develop algorithms for computing probabilistic answers. They also address the important issue of measuring the quality of the answers to the queries, and provide algorithms for efficiently pulling data from relevant sensors or moving objects in order to improve the quality of the executing queries. [CKP2003] considers all the possible semantic world over the uncertain road networks, due to the fact that each possible world is a deterministic road network which could eventually appear in the real world.

Many times probabilistic database query processing mostly looks into the avenue of semantics. [DS2007] in their previous works describe an optimization algorithm that can compute efficiently most queries. They show the data complexity of some queries is #P-complete, which implies that these queries do not admit any efficient evaluation methods. There are an exponential number of possible worlds in a probabilistic database and many previous works including nearest neighbor queries [CKP2003], skyline queries [PJLY2007], and top-k queries. [LSD2009] have address the issue. Frank [F1969] simulates a system which generates a set of random branch-length vectors according to given probability distributions and thereby apply a shortest path algorithm to compute the value for this simulation.

[YPS2002] studied how a continuous nearest neighbor query retrieves the nearest neighbor (NN) of every point on a line segment. In their result they have a set of (point, interval) tuples, such that point is the nearest neighbor of all points in the corresponding interval. [RKV1995] studied the efficient branch-and-bound R-tree traversal algorithm to find the nearest neighbor object to a point, and then generalize it to finding the k nearest neighbors. They furthermore discussed the metrics for an optimistic and a pessimistic search ordering strategy as well as for pruning and propose a depth-first method that starts from the root of the tree to the leaf node in which the process is repeated recursively until the first nearest neighbor is detected.

[ZR2001], [BJKS2002] in their previous work they addressed the problem of finding *k* nearest neighbors for *moving* query point (*k*-NNMP) and in dealing with the problem they assume that the query point is not static, as in *k*-nearest neighbor problem, but varies its position over time.

Probabilistic Graphs Path Queries. Some of the works that have been previously studied under the probabilistic graph and on uncertain road networks include a study by Rasteiro and Anjo [RA2004] who examine the issue of optimal paths in directed random networks in which this path is the path that maximizes utility function expected values. Ming and Pei [MP2010] in their model consider arbitrary weight distributions and correlations between the weights of adjacent edges. Also in tackling the problem of traffic uncertainty they proposed probabilistic path queries and two new types of top-k path queries.

[HL2009], [JLDW2011], [PBGK2010] are some of the other previous works that review the RDF graph with containing edges associated with existence probabilities. Also previous works from [WMGH2008], [F2005], [LC2011] use graphical models to model the probabilistic graph, in which the labels of vertices connecting through edges are dependent and also they are represented by the conditional probability tables. [CA2012] considers a new network reachability problem where their goal is to find the most reliable path between two nodes in a network, represented as a directed acyclic graph. They computed a path between two end nodes when each edge has a failure probability.

[HP2010] studied how to take traffic uncertainty into account in answering path queries in road networks. They proposed to capture the uncertainty in traffic such as the travel time between two vertices and the weight of an edge is modeled as a random variable and is approximated by a set of samples. They also proposed three novel types of probabilistic path queries using basic probability principles and reviewed uncertain traffic information by studying the path queries on road networks of which they consider the paths between two vertices having a total weight less than a threshold with high probability.

[L1983] studied the proposed simulation method to approximate the probability distribution of the shortest path and defined the utility function that specifies the preferences among the paths. Moreover, [RA2004] studied the problem of optimal paths in directed random networks, whereby the cost of each arc is a real-valued random variable whereas the optimal path is regarded as the path that maximizes the expected values of a utility function. They consider the linear, quadratic, and exponential cases, presenting a theoretical formulation based on multi-criteria models as well as the resulting algorithms and computational tests.

CHAPTER VI

CONCLUSIONS

In this paper, we studied the uncertain road network by modelling the road network traffic and considered many factors which may cause speeds of vehicles to be imprecise and probabilistic. We therefore focus more on continuous probabilistic shortest traveling time query (CPSTTQ), which retrieves sets of objects that have the smallest traveling time to a moving query point *q* from one location to the other on road networks with high confidences. We propose effective pruning methods and time bound pruning to prune the search space of our CPSTTQ query and filter out CPSTTQ false alarms. Moreover, we designed an efficient query procedure to answer CPSTTQ via an index structure and are thereby able to retrieve the CPSTTQ answer efficiently.

REFERENCES

[BFM2006] Bast, H., Funke, S., and Matijevic, D. (2006). “TRANSIT—ultrafast shortest-path queries with linear-time preprocessing." *9th Discrete Mathematics and Theoretical Computer Science Implementation Challenge [1]* (2006).

[BFMSS2007] Bast, H., Funke, S., Matijevic, D., Sanders, P., and Schultes, D. "In Transit to Constant Time Shortest-Path Queries in Road Networks." In *Algorithm Engineering and Experiments*. 2007.

[BJKS2002] Benetis, R., Jensen, C. S., Karciauskas, G., and Saltenis, S. "Nearest neighbor and reverse nearest neighbor queries for moving objects." In *Database Engineering and Applications Symposium, 2002. Proceedings. International*, pp. 44-53. IEEE, 2002.

[CP1987] Cavallo, R., Pittarelli, M.: The theory of probabilistic databases. Very Large Data Bases 71–81 (1987)

[CA2012] Chang, A. and Amir, E. "Reachability under uncertainty." *arXiv preprint arXiv:1206.5253* (2012).

[CKP2003] Cheng, R., Kalashnikov, D. V., and Prabhakar, S. "Evaluating probabilistic queries over imprecise data." In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 551-562. ACM, 2003.

[DS2007] Dalvi, N., and Suciu, D. Efficient query evaluation on probabilistic databases. Very Large Data Bases *J.*, 16(4), 2007.

[DS1996] Dey, D., Sarkar, S. A probabilistic relational model and algebra. ACM Trans. Database Syst. 21(3), 339–369 (1996).

[F1969] Frank, H. Shortest paths in probabilistic graphs. Operations Research, 17(4):583–599, 1969.

[F2005] Fukushige, Y. "Representing probabilistic relations in RDF," in Proc. International Semantic Web Conference, 2005, pp. 106–107.

[HP2010] Hua, M., and Pei, J. "Probabilistic path queries in road networks: traffic uncertainty aware path selection." *Proceedings of the 13th International Conference on Extending Database Technology*. ACM, 2010.

[HL2009] Huang, H. and Liu, C. *Query evaluation on probabilistic RDF databases*. Springer Berlin Heidelberg, 2009.

[JLDW2011] Jin, R., Liu, L., Ding, B., and Wang, H. "Distance-constraint reachability computation in uncertain graphs," in Proc. Very Large Data Bases, Jun. 2011, pp. 551–562.

[MYPM2006] Mouratidis, K., Yiu, M. L., Papadias, D. and Mamoulis, N. "Continuous nearest neighbor monitoring in road networks." In *Proceedings of the 32nd international conference on Very large data bases*, pp. 43-54. Very Large Data Bases Endowment, 2006.

[LSD2009] Li, J., Saha, B. and Deshpande, A. A unified approach to ranking in probabilistic databases. Very Large Data Bases, 2(1), 2009.

[LC2011] Lian, X. and Chen, L. “Efficient query answering in probabilistic RDF graphs,” in Proc. Special Interest Group on Management of Data, Athens, Greece, 2011.

[LC2014] Lian, X., and Chen, L. "Trip Planner over Probabilistic Time-Dependent Road Networks." *Knowledge and Data Engineering, IEEE Transactions on* 26.8 (2014): 2058-2071.

[L1983] Loui, R. P. Optimal paths in graphs with stochastic or multidimensional weights. Commun. ACM, 26(9):670–676, 1983.

[PJLY2007] Pei, J., Jiang, B., Lin, X., and Yuan, Y. Probabilistic skylines on uncertain data. In Very Large Data Bases, 2007.

[PBGK2010] Potamias, M., Bonchi, F., Gionis, A., and Kollios, G. “K-nearest neighbors in uncertain graphs,” in Proc. Very Large Data Bases, Singapore, Sep.2010.

[RA2004] Rasteiro, D. D. M. L., and Anjo, A. J. B. "Optimal paths in probabilistic networks." *Journal of Mathematical Sciences* 120, no. 1 (2004): 974-987.

[RKV1995] Roussopoulos, N., Kelley, S., and Vincent, F. "Nearest neighbor queries." *ACM Special Interest Group on Management of Data record*. Vol. 24. No. 2. ACM, 1995.

[SS2005] Sanders, P. and Schultes, D. Highway hierarchies hasten exact shortest path queries. In Enterprise Software and Applications 2005.

[SS2006] P. Sanders and D. Schultes. Engineering highway hierarchies. In Enterprise Software and Applications 2006

[SKS2002] Shahabi, C., Kolahdouzan, M. R. and Sharifzadeh, M. A road network embedding technique for k-nearest neighbor search in moving object databases. In *Geographical Information Systems*, 2002.

[YPS2002] Tao, Y., Papadias, D., and Shen, Q. "Continuous nearest neighbor search." *Proceedings of the 28th international conference on Very Large Data Bases*. Endowment, 2002.

[WMGH2008] Wang, D. Z., Michelakis, E., Garofalakis, M., and Hellerstein, J. "Bayestore: Managing large, uncertain data repositories with probabilistic graphical models," in Proc. *Very Large Data Bases*, 2008, pp. 922–933.

[WJYQ2003] Yimin, W., Jianmin, X., Yucong, H., and Qinghong, Y. "A shortest path algorithm based on hierarchical graph model." In *Intelligent Transportation Systems, 2003.* vol. 2, pp. 1511-1514. IEEE, 2003.

[ZR2001] Song, Z., and Roussopoulos, N. "K-nearest neighbor search for moving query point." In *Advances in Spatial and Temporal Databases*, pp. 79-96. Springer Berlin Heidelberg, 2001.

[Z1997] Zimányi, E. "Query evaluation in probabilistic relational databases." *Theoretical Computer Science* 171, no. 1 (1997): 179-219.

BIOGRAPHICAL SKETCH

Bamikole Ajibola Ogundele, the author was born in a city called Ile-Ife in western part of Nigeria, West Africa in mid 1970s to his mother Abigael and his father Nathaniel Ogundele. He attended his elementary and secondary school education in Nigeria. He migrated to United States of America where he attended Century Community College, White Bear Lake, Minnesota and received his Associate Degree in Microcomputer Support Technology in May 2009. He then transferred his credit to Metropolitan State University, Saint Paul, Minnesota and completed his Bachelor of Science Degree in Management Information System in December 2012. The author finally decided to pursue his Master degree from University of Texas Pan American where he received his Master of Science in Computer Science in May 2015.

He briefly worked for State of Minnesota until December of 2013 and Chevron Corporation in 2014 as Project Manager where he supervised software application development. He also worked as Graduate Teaching Assistant at University of Texas Pan American from 2014 till 2015. The author is married with three children. His mailing address is 3079 Chisholm Court North, Maplewood MN. 55109.