Probabilistic Path Queries in Road Networks: Traffic Uncertainty Aware Path Selection[*]

Ming Hua
Facebook Inc.
Palo Alto, CA, USA
arceehua@facebook.com

Jian Pei
Simon Fraser University, Canada
Burnaby, BC, Canada
jpei@cs.sfu.ca

ABSTRACT

Path queries such as "finding the shortest path in travel time from my hotel to the airport" are heavily used in many applications of road networks. Currently, simple statistic aggregates such as the average travel time between two vertices are often used to answer path queries. However, such simple aggregates often cannot capture the uncertainty inherent in traffic. In this paper, we study how to take traffic uncertainty into account in answering path queries in road networks. To capture the uncertainty in traffic such as the travel time between two vertices, the weight of an edge is modeled as a random variable and is approximated by a set of samples. We propose three novel types of probabilistic path queries using basic probability principles: (1) a probabilistic path query like "what are the paths from my hotel to the airport whose travel time is at most 30 minutes with a probability of at least 90%?"; (2) a weight-threshold top-k path query like "what are the top-3 paths from my hotel to the airport with the highest probabilities to take at most 30 minutes?"; and (3) a probability-threshold top-k path query like "what are the top-3 shortest paths from my hotel to the airport whose travel time is guaranteed by a probability of at least 90%?" To evaluate probabilistic path queries efficiently, we develop three efficient probability calculation methods: an exact algorithm, a constant factor approximation method and a sampling based approach. Moreover, we devise the P* algorithm, a best-first search method based on a novel hierarchical partition tree index and three effective heuristic evaluation functions. An extensive empirical study using real road networks and synthetic data sets shows the effectiveness of the proposed path queries and the efficiency of the query evaluation methods.

[*]This research is supported in part by an NSERC Discovery Grant and an NSERC Discovery Accelerator Supplement Grant. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

1. INTRODUCTION

How often do you search for driving directions online by asking path queries on road networks such as "finding the shortest path in travel time from my hotel to the airport"? Thanks to a few emerging techniques for traffic monitoring such as roadside sensors and cell phone signal analysis, more and more traffic data about road networks becomes available. Currently, using online services such as Google maps and Yahoo! maps, simple statistic aggregates such as the average travel time between two vertices are often used to answer path queries. However, such simple aggregates often cannot sufficiently capture the uncertainty inherent in traffic.

At the data collection level, roadside sensors are often used to count traffic volumes, measure vehicle speeds, or classify vehicles. The actual travel time along a road segment can be derived from such sensor readings. However, data collected from sensors as such cannot be accurate all the time due to the limitations of equipment and delay or loss in data transfer. Therefore, confidence values are often assigned to those data, based on the specific sensor characteristics, the predicted value, and the physical limitations of the system [12]. Moreover, different vehicles have different speeds. Therefore, the travel time along each road segment derived from sensor readings is inherently uncertain and probabilistic.

At the model level, while a road network is certain, uncertainty is inherent in travel time on the road network. Travel time is often represented as weights of edges in a road network. To capture the uncertainty, the weight on an edge can be modeled as a random variable. In practice, a weight random variable can be approximated by a set of samples where each sample takes a membership probability to be the representative. We call such a network incorporating uncertain traffic information an *uncertain traffic network*.

Does uncertainty affect path queries in a road network?

EXAMPLE 1 (PROBABILITY AND PATHS). Suppose in an uncertain traffic network, from point A to point B there are two paths, P_1 and P_2. The set of travel time samples (in minutes) of P_1 is $\{35, 35, 38, 40\}$ and the set of samples of P_2 is $\{25, 25, 48, 50\}$. Should each sample take a membership probability of 25%, the average travel time on both P_1 and P_2 is 37 minutes. Which path is better?

If a user wants to make sure that the travel time is no more than 40 minutes, path P_1 is better since according to the samples, it has a probability of 100% to meet the travel time constraint, while path P_2 has a probability of only 50% to meet the constraint. On the other hand, if a user wants to go from A to B in 30 minutes, path P_2 should be recommended since the path has a probability of 50% to make it while P_1 has no chance to make it. ∎

Example 1 clearly shows that answers to path queries in an uncertain traffic network depend on the distributions of weight variables and users' constraints. We need to extend the traditional path queries in classical road networks to fully utilize the rich traffic information available in an uncertain traffic network.

Example 2 (Probabilistic path queries). Suppose multiple paths exist from point A to point B in an uncertain traffic network. Taking the uncertainty of travel time into consideration, path queries may be extended to **probabilistic path queries** such as "what are the paths from my hotel to the airport whose travel time is at most 30 minutes with a probability of at least 90%?"

In a large road network, there can be many paths between two points. A user is often not interested in all answers to a path query. Instead, the top-k paths in a ranked list are highly desirable. Since we have two constraints in a probabilistic path query, the travel time and the probability, there are two types of top-k probabilistic path queries.

First, a user can constrain the travel time and rank paths in probability descending order in a **weight-threshold top-k path query**, such as "what are the top-3 paths from my hotel to the airport with the highest probabilities to take at most 30 minutes?"

Second, a user can constrain the probability and rank paths in travel time ascending order in a **probability - threshold top-k path query**, such as "what are the top-3 shortest paths from my hotel to the airport whose travel time is guaranteed by a probability of at least 90%?" ∎

Although answering traditional path queries in road networks has been studied extensively, answering probabilistic path queries on uncertain traffic network is challenging.

Challenge 1: How can path queries be asked in an intuitive and useful way in uncertain traffic networks? As illustrated in Example 1, path queries in an uncertain traffic network are more complicated when traffic uncertainty is considered. Therefore, we need to extend path queries properly in uncertain traffic networks.

Our contribution. We model an uncertain traffic network as a simple graph with probabilistic weights on edges. Correlations among probabilistic weights are considered. Moreover, we propose three types of probabilistic path queries as elaborated in Example 2. The queries are based on simple probability principles and the answers can be understood easily.

Challenge 2: How to answer probabilistic path queries efficiently in uncertain traffic networks? A straightforward method is to compute the probability distribution of the travel time on every path from the source point to the destination point. However, in a large road network, the number of paths can be huge. Moreover, computing the distribution of travel time itself on a long path consisting of many edges is costly.

There has been much existing work on path queries in probabilistic graphs, such as stochastic shortest paths [9, 20], routing with probabilistic graphs [11], and stochastic on-time arrival problem (SOTA) [6]. However, the above methods cannot be extended straightforwardly to answer probabilistic path queries on real road networks due to the following two reasons. First, they often adopt a simple probabilistic graph model, where all probabilistic weights of edges are independent from each other, or only some particular types of correlations are considered. Second, path query evaluation in real road networks is computationally challenging. Pre-computation is often used to reduce the evaluation cost. Without considering pre-computation, the above methods may not scale well in real road networks.

Path queries on certain (deterministic) traffic networks have been studied extensively [26, 27, 13, 25]. However, uncertainty is not considered in those studies. Therefore, those query answering methods and indices cannot be easily extended to the probabilistic path queries discussed in this paper.

Our contribution. We address the problem of probabilistic path query answering in large scale probabilistic road networks. First, we propose three algorithms to compute the distribution of travel time given a set of correlated probabilistic edge weights. The exact algorithm explores the conditional independency among edge weights. A constant factor approximation algorithm and a sampling based method approximate the probability distribution of path weights efficiently.

Second, we develop the **P* algorithm**, a best-first search method for efficient path search in probabilistic graphs. Three heuristic evaluation functions are proposed to help select the best path during the search. Moreover, to support efficient P* search in large scale uncertain traffic networks, a hierarchical partition tree is devised to index the graph structure and probability distribution information in probabilistic road networks.

The rest of the paper is organized as follows. In Section 2, we define probabilistic path queries formally. We review the related work in Section 3. We discuss an exact probability calculation algorithm and two approximation methods in Section 4. We devise P*, a best-first search algorithm in Section 5. To handle large scale uncertain traffic networks, a hierarchical index is developed in Section 6. A systematic empirical evaluation is reported in Section 7. The paper is concluded in Section 8.

Limited by space, we omit all mathematical proofs in the paper. The proofs are available at `http://www.cs.sfu.ca/~jpei/publications/edbt10proof.pdf`.

2. PROBABILISTIC PATH QUERIES

In this section, we model uncertain traffic networks as probabilistic graphs and define probabilistic path queries. To keep our discussion simple, we consider undirected simple graphs only in this paper. However, our discussion can be extended to directed graphs straightforwardly.

2.1 Probabilistic Graphs and Paths

A **probabilistic graph** $G(V, E, W)$ is a simple graph with probabilistic weights on edges, which consists of a set of vertices V, a set of edges $E \subseteq V \times V$, and a set of weights W defined on edges in E. For each edge $e \in E$, $w_e \in W$ is a real-valued random variable in $(0, +\infty)$, denoting the travel time along edge e.

Theoretically, the distribution of w_e can be captured by a **probability density function** (**PDF** for short) $w_e(x)$. In practice, $w_e(x)$ is often unavailable. Instead, a set of **samples** $\widehat{w}_e = \{x_1, \ldots, x_m\}$ is used to approximate the distribution of w_e, where each sample $x_i > 0$ $(1 \leq i \leq m)$

takes a **membership probability** $Pr[\widehat{w}_e = x_i] \in (0,1]$ to appear. Moreover, $\sum_{i=1}^{m} Pr[\widehat{w}_e = x_i] = 1$. In the rest of the paper, to keep our presentation simple, we write $\widehat{w}_e$ as w_e as well if no ambiguity, and call a sample $x_i \in \widehat{w}_e$ as the sample of the edge.

w_e is a discrete random variable whose **probability mass function** (**pmf** for short) is $f_e(x_i) = Pr[w_e = x_i]$. The **cumulative distribution function** (**cdf** for short) of w_e is $F_e(x_i) = Pr[w_e \leq x_i]$.

A **simple path** P is a sequence of non-repeated vertices $\langle v_1, \ldots, v_{n+1}\rangle$, where $e_i = (v_i, v_{i+1})$ is an edge in E $(1 \leq i \leq n)$. v_1 and v_{n+1} are called the **start vertex** and the **end vertex** of P, respectively. For the sake of simplicity, we call a simple path a **path** in the rest of the paper. Given two vertices u and v, the complete set of paths between u and v is denoted by $\mathcal{P}_{u,v}$.

For two paths $P = \langle v_1, \ldots, v_{n+1}\rangle$ and $P' = \langle v_{i_0}, v_{i_0+1}, \ldots, v_{i_0+k}\rangle$ such that $1 \leq i_0 \leq n+1-k$, P is called a **super path** of P' and P' a **subpath** of P. Moreover, $P = \langle P_1, P_2, \ldots, P_m\rangle$ if $P_1 = \langle v_1, \ldots, v_{i_1}\rangle$, $P_2 = \langle v_{i_1+1}, \ldots, v_{i_2}\rangle$, $\ldots$, $P_m = \langle v_{i_{m-1}+1}, \ldots, v_{n+1}\rangle$, $1 < i_1 < i_2 < \cdots < i_{m-1} \leq n$. P_j $(1 \leq j \leq m)$ is called a **segment** of P.

The **weight** of path $P = \langle v_1, \ldots, v_{n+1}\rangle$ is the sum of the weights of all edges in P, that is $w_P = \sum_{i=1}^{n} w_{e_i}$, where w_{e_i} is the weight of edge $e_i = (v_i, v_{i+1})$ with probability mass function $f_{e_i}(x)$. Since each w_{e_i} is a discrete random variable, w_P is also a discrete random variable. A **sample** of P is $x_P = \sum_{i=1}^{n} x_i$, where x_i $(1 \leq i \leq n)$ is a sample of edge $e_i = (v_i, v_{i+1})$. We also write $x_P = \langle x_1, \ldots, x_n\rangle$ where $x_1, \ldots, x_n$ are called the **components** of x_P.

The **probability mass function** of w_P is

$$\begin{aligned} f_P(x) &= Pr[w_P = x] \\ &= \sum_{x_1+\ldots+x_n=x} Pr[w_{e_1} = x_1, \ldots, w_{e_n} = x_n] \end{aligned} \tag{1}$$

In road networks, the travel time on a road segment e may be affected by the travel time on other roads connecting with e. Therefore, the weights of adjacent edges in E may be correlated. Among all edges in path P, the correlation between the weights w_{e_i} and $w_{e_{i+1}}$ of two adjacent edges e_i and e_{i+1} $(1 \leq i \leq n)$ can be represented using different methods, depending on the types of correlations. To keep our discussion general, in this paper we represent the correlation between w_{e_i} and $w_{e_{i+1}}$ using the joint distribution over the sample pairs $(x_i, x_{i+1}) \in w_{e_i} \times w_{e_{i+1}}$. The **joint probability mass function** of w_{e_i} and $w_{e_{i+1}}$ is $f_{e_i,e_{i+1}}(x_i, x_{i+1}) = f_{e_{i+1},e_i}(x_{i+1}, x_i) = Pr[w_{e_i} = x_i, w_{e_{i+1}} = x_{i+1}]$. Correspondingly, the **conditional probability** of w_{e_i} given $w_{e_{i+1}}$ is $f_{e_i|e_{i+1}}(x_i|x_{i+1}) = \frac{f_{e_i,e_{i+1}}(x_i,x_{i+1})}{f_{e_{i+1}}(x_{i+1})}$.

THEOREM 1 (PATH WEIGHT MASS FUNCTION). *The probability mass function of a simple path $P = \langle v_1, \ldots, v_{n+1}\rangle$ $(e_i = (v_i, v_{i+1})$ for $1 \leq i \leq n)$ is*

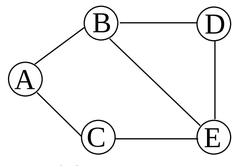
$$f_P(x) = \sum_{x_1+\ldots+x_n=x} \frac{\prod_{i=1}^{n-1} f_{e_i,e_{i+1}}(x_i, x_{i+1})}{\prod_{j=2}^{n-1} f_{e_j}(x_j)} \tag{2}$$

■

In sequel, the **cumulative distribution function** of w_P is

$$F_P(x) = Pr[w_P \leq x] = \sum_{0<x_i\leq x} f_P(x_i) \tag{3}$$

We call $F_P(x)$ the x-**weight probability** of path P.



(a) A graph.

Edge e_i	Weight: value (probability) x_{i1}	x_{i2}	x_{i3}
e_1:AB	10(0.3)	15(0.3)	20(0.4)
e_2:AC	5(0.2)	10(0.3)	15(0.5)
e_3:BD	20(0.4)	25(0.4)	30(0.2)
e_4:BE	5(0.2)	25(0.6)	40(0.2)
e_5:CE	10(0.5)	20(0.1)	45(0.4)
e_6:DE	10(0.3)	20(0.6)	50(0.1)

(b) Probabilistic weights of edges.

	20	25	30
10	0.15	0.15	0
15	0.15	0.15	0
20	0.1	0.1	0.2

(c) $f_{e_1,e_3}(x_{1i}, x_{3j})$.

	10	20	50
20	0.1	0.2	0.1
25	0.1	0.3	0
30	0.1	0.1	0

(d) $f_{e_3,e_6}(x_{3i}, x_{6j})$.

Figure 1: A probabilistic graph.

EXAMPLE 3 (PROBABILISTIC GRAPH AND PATHS). A probabilistic graph is shown in Figure 1, where the weight of each edge is represented by a set of samples and their membership probabilities.

Path $P = \langle A, B, D, E\rangle$ consists of edges AB, BD and DE. The joint probabilities of (w_{AB}, w_{BD}) and (w_{BD}, w_{DE}) are shown in Figures 1(c) and 1(d), respectively. The probability that $w_P = 45$ is

$$\begin{aligned} & Pr[w_P = 45] \\ = \; & Pr[w_{e_1} = 15, w_{e_3} = 20, w_{e_6} = 10] \\ & + Pr[w_{e_1} = 10, w_{e_3} = 25, w_{e_6} = 10] \\ = \; & \frac{f_{e_1,e_3}(15,20) \times f_{e_3,e_6}(20,10)}{f_{e_3}(20)} + \frac{f_{e_1,e_3}(10,25) \times f_{e_3,e_6}(25,10)}{f_{e_3}(25)} \\ = \; & 0.075 \end{aligned}$$

■

2.2 Path Queries

Let us formulate the probabilistic path queries elaborated in Example 2.

DEFINITION 1 (PROBABILISTIC PATH QUERIES). *Given probabilistic graph $G(V, E, W)$, two vertices $u, v \in V$, a weight threshold $l > 0$, and a probability threshold $\tau \in (0, 1]$, a **probabilistic path query** $Q_l^\tau(u, v)$ finds all paths $P \in \mathcal{P}_{u,v}$ such that $F_P(l) \geq \tau$.* ■

There can be many paths between two vertices in a large graph. Often, a user is interested in only the "best" paths. Thus, we define weight- and probability-threshold top-k path queries.

DEFINITION 2 (TOP-k PROBABILISTIC PATH QUERIES). *Given probabilistic graph $G(V, E, W)$, two vertices $u, v \in V$, an integer $k > 0$, and a weight threshold $l > 0$, a **weight-threshold top-k path query** $WTQ_l^k(u, v)$ finds the k paths $P \in \mathcal{P}_{u,v}$ with the largest $F_P(l)$ values.*

*For a path P, given a **probability threshold** $\tau \in (0, 1]$, we can find the smallest weight x such that $F_P(x) \geq \tau$, which is called the τ**-confident weight**, denoted by*

$$F_P^{-1}(\tau) = \min\{x | x \in w_P \wedge Pr[w_P \leq x] \geq \tau\} \tag{4}$$

*A **probability-threshold top-k path query** $PTQ_\tau^k(u, v)$ finds the k paths $P \in \mathcal{P}_{u,v}$ with the smallest $F_P^{-1}(\tau)$ values.* ■

Example 4 (Path Queries). In the probabilistic graph in Figure 1, there are 4 paths between A and D, namely $P_1 = \langle A, B, D\rangle$, $P_2 = \langle A, B, E, D\rangle$, $P_3 = \langle A, C, E, B, D\rangle$, and $P_4 = \langle A, C, E, D\rangle$. Suppose the weights of all edges are independent in this example.

Given a weight threshold $l = 48$ and a probability threshold $\tau = 0.8$, a probabilistic path query Q_l^τ finds the paths whose weights are at most 48 of probability at least 0.8. According to the cumulative distribution functions of the paths, we have $F_{P_1}(48) = 0.92$, $F_{P_2}(48) = 0.14$, $F_{P_3}(48) = 0.028$, and $F_{P_4}(48) = 0.492$. Thus, the answer is $\{P_1\}$.

The weight-threshold top-3 path query $WTQ_l^3(A, D)$ finds the top-3 paths P having the largest 48-weight probability values $F_P(48)$. The answer to $WTQ_l^2(A, D)$ is $\{P_1, P_4, P_2\}$.

The probability-threshold top-3 path query $PTQ_\tau^3(A, D)$ finds the top-3 paths P having the smallest 0.8-confidence weights $F_P^{-1}(0.8)$. Since $F_{P_1}(40) = 0.7$ and $F_{P_1}(45) = 0.92$, the smallest weight that satisfies $F_P(x) \geq 0.8$ is 45. Thus, $F_{P_1}^{-1}(0.8) = 45$. Similarly, we have $F_{P_2}^{-1}(0.8) = 75$, $F_{P_3}^{-1}(0.8) = 105$, and $F_{P_4}^{-1}(0.8) = 75$. Therefore, the answer to $PTQ_\tau^2(A, D)$ is $\{P_1, P_2, P_4\}$. ■

To keep our presentation simple, hereafter we call probabilistic path queries, weight- and probability-threshold top-k queries as **path queries**, **WT top-k queries**, and **PT top-k queries**, respectively.

3. RELATED WORK

Our study is related to the previous work about path queries on probabilistic graphs and on traffic networks. Both problems have been studied extensively. However, to the best of our knowledge, there is no work on extending probabilistic graphs to traffic networks.

3.1 Path Queries on Probabilistic Graphs

The shortest path problem on probabilistic graphs has been studied under various constraints. For example, Frank [9] studied the shortest path queries on probabilistic graphs, where the weight of each edge is a random variable following a certain probability distribution. A Monte Carlo simulation method is proposed to approximate the probability distribution of the shortest path. Loui [20] extended [9] by defining a utility function which specifies the preference among paths. More recently, Rasteiro and Anjo [23] studied the problem of optimal paths in directed random networks, where the cost of each arc is a real-valued random variable following Gaussian distribution, and the optimal path is a path that maximizes the expected value of a utility function.

Stochastic shortest path search [8, 21, 6, 7] is to find a path between two end nodes and maximize the probability that the path length does not exceed a given threshold. It is also referred to as the "stochastic on-time arrival problem (SOTA)". SOTA has the same semantics as the WT top-1 queries (a special case of WT top-k queries discussed in this paper). However, Nikolova *et al.* [21] only considered some particular parametric weight distribution (such as the Normal distribution and the Poisson distribution). In addition, there have been other formulations of the optimal routing problem with probabilistic graphs. Ghosh *et al.* [11] developed an optimal routing algorithm that generates an optimal delivery subgraph so that the connectivity between two end nodes is maximized. Chang and Amir [5] computed the most reliable path between two end nodes when each edge has a failure probability.

Our work here is different from the above studies in the following three aspects.

First, the probabilistic graph models are different. Many existing studies focus on simple probabilistic graphs, where probabilistic weights are independent from each other, such as [9, 6]. Moreover, some methods only work for certain probability distributions, such as the Normal distribution [21]. Last, the existing studies only consider certain types of correlations, such as the dependence with a global hidden variable [5]. Our model considers arbitrary weight distributions and correlations between the weights of adjacent edges. Our model is more capable and flexible for real road networks.

Second, the path queries are different. Most of the above studies focus on the optimal path query, where a utility function is adopted to evaluate the optimality of paths. However, as discussed in Section 1, using a single simple aggregate as the utility score may not capture traffic uncertainty sufficiently, since the probability of optimality is often very small. To tackle the problem, we propose probabilistic path queries and two new types of top-k path queries.

Last, the query answering techniques are different. We propose a novel best-first search algorithm for probabilistic path queries. Moreover, we develop a hierarchical partition tree to index the road network structure and weight distribution information. Our query answering methods are efficient and scalable thanks to the two techniques.

3.2 Path Queries on Certain Traffic Networks

The shortest path queries on traffic networks have been studied extensively before. Please see [10] for a nice survey.

The well known A* algorithm [15] uses a heuristic evaluation function $f(x) = g(x) + h(x)$ to measure the optimality of the path currently under exploration, where $g(x)$ is the cost from the start vertex to the current vertex, and $h(x)$ is the heuristic estimation of the distance to the goal. The paths with a smaller $f(x)$ score are explored earlier.

Sanders and Schultes [26, 27] proposed a "highway hierarchy" for large scale traffic networks, which captures the key edges that are on the shortest paths between two far away vertices. Then, the shortest path search is restricted to those key edges.

Bast *et al.* [2, 3] introduced a concept of "transit" node to preprocess traffic networks. A transit node is on a set of non-local shortest paths. The distance from every vertex in the network to its closest transit node is computed to help the shortest path search.

Gonzalez *et al.* [13] mined important driving and speed patterns from historical data to help to compute the fastest paths on traffic networks. A road hierarchy is built based on different classes of roads. Frequently traversed road segments are preferred in the path search.

In addition, Kurzhanski and Varaiya [19] considered a model that allows correlations between links for the reachability problem. More studies on the hierarchical approach for searching shortest path include [29, 4, 25].

The above studies tackle the path queries in large scale *certain* traffic networks. Therefore, both the graph models and the query types are different from our work. Thus, those techniques cannot be extended straightforwardly to probabilistic path queries on uncertain road networks.

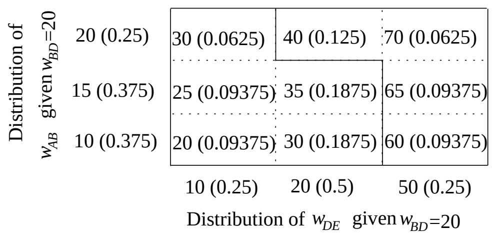


Figure 2: The weight constrained region of $P_1 = \langle A, B, D, E\rangle$ when w_{BD} takes sample 20.

Moreover, although hierarchical indices have been extensively used in path queries on certain traffic networks, the existing index techniques only work for certain path queries. In this paper, we develop a hierarchical partition tree to index the weight probability distributions on graphs.

4. PROBABILITY CALCULATION

There are two orthogonal issues in answering a path query $Q_l^\tau(u, v)$: the l-weight probability calculation and the path search. In this section, we first discuss how to compute the exact l-weight probabilities for paths. Then, two approximation algorithms are developed. We also present a straightforward depth-first path search method. An efficient best-first path search algorithm will be introduced in Section 5.

4.1 Exact l-Weight Probability Calculation

EXAMPLE 5 (l-WEIGHT CONSTRAINED REGION). Let $l = 55$ and $\tau = 0.5$. Consider path query $Q_l^\tau(A, E)$ and path $P = \langle A, B, D, E\rangle$ in the probabilistic graph in Figure 1.

P contains three edges, $e_1 = (A, B)$, $e_3 = (B, D)$ and $e_6 = (D, E)$. The joint probabilities f_{e_1,e_3} and f_{e_3,e_6} are given in Figures 1(c) and 1(d), respectively, which specify the correlation between edges. Weights w_{e_1} and w_{e_6} are conditionally independent given w_{e_3}.

The conditional probability of w_{e_1} given $w_{e_3} = 20$ is

$$f_{e_1|e_3}(x|20) = \frac{f_{e_1,e_3}(x,20)}{f_{e_3}(20)} = \begin{cases} 0.375, & x = 10 \text{ or } x = 15; \\ 0.25, & x = 20. \end{cases}$$

The conditional probability of w_{e_6} given $w_{e_3} = 20$ is

$$f_{e_6|e_3}(x|20) = \frac{f_{e_6,e_3}(x,20)}{f_{e_3}(20)} = \begin{cases} 0.25, & x = 10 \text{ or } x = 50; \\ 0.5, & x = 20. \end{cases}$$

The probability that w_P is at most l when $w_{e_3} = 20$ is

$$\begin{aligned} &Pr[w_P \le 55 | w_{e_3} = 20] \\ &= \sum_{x_1+20+x_3 \le 55} Pr[w_{e_1} = x_1, w_{e_3} = 20, w_{e_6} = x_3] \\ &= f_{e_3}(20) \sum_{x_1+x_3 \le 35} f_{e_1|e_3}(x_1|20) \cdot f_{e_6|e_3}(x_3|20) \end{aligned}$$

The sets of samples and their membership probabilities of $f_{e_1|e_3}(x_1|20)$ and $f_{e_6|e_3}(x_3|20)$ are {10(0.375), 15(0.375), 20(0.25)} and {20(0.25), 20(0.5), 50(0.25)}, respectively. We sort the samples in the weight ascending order.

There are in total $3 \times 3 = 9$ samples on $w_{e_1} \times w_{e_6}$ when $w_{e_3} = 20$. To enumerate all samples, we can build a 3×3 sample array M as shown in Figure 2. Cell $M[i, j]$ stores two pieces of information: (1) $M[i, j].ws$ is the sum of the i-th sample of w_{e_1} and the j-th sample of w_{e_6}; and (2) $M[i, j].pr$ is the membership probability of sample $M[i, j].ws$ which equals the product of the corresponding membership probabilities. For example, the lowest left-most cell $M[1, 1]$ corresponds to a sample where w_{e_1} is 10 and w_{e_6} is 10. Thus, $M[1, 1].ws = 10 + 10 = 20$ and $M[1, 1].pr = 0.375 \times 0.25 = 0.09375$.

When w_{e_3} takes sample 20, in order to satisfy $w_{P_1} \le 55$, the samples of $w_{e_1} + w_{e_6}$ should take values of at most 35. Those samples are at the left-lower part of the array. We call the region of those cells the *l-weight constrained region* when $w_{e_3} = 20$. The sum of the membership probabilities of the cells in the region is 0.375.

The l-weight constrained region when w_{e_3} takes other samples can be calculated similarly. When $w_{e_3} = 25$ and $w_{e_3} = 30$, the sum of the membership probabilities of the cells in the l-weight constraint regions are 0.09375 and 0, respectively. $F_{P_1}(55) = f_{e_3}(20) \times 0.375 + f_{e_3}(25) \times 0.09375 + f_{e_3}(30) \times 0 = 0.1875 < \tau$. P_1 is not an answer to Q_l^τ. ■

The idea in Example 5 can be generalized to paths of arbitrary length.

THEOREM 2 (PATH WEIGHT DISTRIBUTION). *Let $P_m = \langle v_1, \ldots, v_{m+1}\rangle$, $P_{m-1} = \langle v_1, \ldots, v_m\rangle$ and $e_m = (v_m, v_{m+1})$ $(m > 2)$, the conditional probability of w_{P_m} given w_{e_m} is*

$$f_{P_m|e_m}(x|y) = \sum_{z \le x-y} f_{e_{m-1}|e_m}(z|y) \times f_{P_{m-1}|e_m-1}(x-y|z) \tag{5}$$

Moreover, the probability mass function of w_{P_m} is

$$f_{P_m}(x) = \textstyle\sum_{y \le x} f_{e_m}(y) \times f_{P_m|e_m}(x|y) \tag{6}$$

■

For a path $P_m = \langle v_1, \ldots, v_{m+1}\rangle$, the probability function of w_{P_m} can be calculated from $w_{P_{m-1}}$ and w_{e_m} using Theorem 2. Calculating the probability mass function of P_m requires $O(|w_{P_{m-1}}| \cdot |w_{e_m}|) = O(\prod_{e \in P_m} |w_e|)$ time.

Interestingly, if only $F_{P_m}(l)$ is required, we do not need to use all samples in an edge weight. Apparently, the largest sample we can take from w_{e_i} $(1 \le i \le m)$ is bounded by the following rule.

LEMMA 1 (SAMPLE COMPONENTS). *For weight threshold l and path $P = \langle v_1, \ldots, v_{m+1}\rangle$, a sample x_{i_j} of edge $e_i = (v_i, v_{i+1})$ $(1 \le i \le m)$ can be a component of a sample of P in the l-weight constrained region only if $x_{i_j} \le l - \sum_{i' \ne i} x_{i'_1}$, where $x_{i'_1}$ is the smallest sample of edge $e_{i'} = (v_{i'}, v_{i'+1})$.* ■

4.2 Approximating l-Weight Probabilities

The probability mass function of $F_{P_{m+1}}(l)$ can be calculated from the distributions on $f_{P_m|e_m}$ and $f_{e_m}|f_{e_{m+1}}$ according to Theorem 2. To accelerate probability calculation, we introduce an approximation method that keeps a constant number of samples in the weight distribution of any subpath during the search.

If w_{P_m} contains $n > 2t$ samples $x_1, \cdots, x_n$ (t is a user defined parameter), then, we divide those values into b exclusive buckets $\phi_i = [x_{z_i}, x_{z'_i}]$, where for $1 < k \le b$ and $1 \le i \le b$,

$$\begin{aligned} &z_1 = 1 \\ &z_k = z'_{k-1} + 1; \\ &z'_i = \max_{j \ge z_i}\{j | F_{P_m|e_m}(x_j|y) - F_{P_m|e_m}(x_{z_i}|y) \le \frac{1}{t}\} \end{aligned} \tag{7}$$

The number of buckets is at most $2t$, as shown in the following lemma.

LEMMA 2 (NUMBER OF BUCKETS). *Given w_{P_m} with n samples ($n > 2t > 0$), let $\phi_i = [x_{z_i}, x_{z'_i}]$ ($1 \leq i \leq b$) be b exclusive buckets satisfying Equation 7. Then, $b \leq 2t$.* ■

Constructing the buckets only requires one scan of all values in w_{P_m}. The minimal value in bucket $\phi_i = [x_{z_i}, x_{z'_i}]$ is $\min(\phi_i) = x_{z_i}$, and the maximal value in ϕ_i is $\max(\phi_i) = x_{z'_i}$. When computing the probability distribution of $w_{P_{m+1}}$ using w_{P_m}, we only select one value in each bucket $\phi_i \subset w_{P_m}$ as the representative, and assign $Pr(\phi_i)$ to the representative. If $\min(\phi_i)$ is selected as the representative, then the so computed $F'_{P_{m+1}}(l)$ is greater than $F_{P_{m+1}}(l)$; if $\max(\phi_i)$ is used as the representative, then the so computed $F''_{P_{m+1}}(l)$ is smaller than $F_{P_{m+1}}(l)$.

EXAMPLE 6 (BUCKET APPROXIMATION). Consider path P_m and edge e_m. Let $f_{P_m|e_m}(x_j|y) = 0.2$ for $1 \leq j \leq 5$. If $t = 2$, then all values in w_{P_m} are divided into three buckets: $[x_1, x_2], [x_3, x_4], [x_5, x_5]$, with probabilities 0.4, 0.4 and 0.2, respectively.

If the minimal value of each bucket is used as a representative, then x_1, x_3 and x_5 are selected. As a result, the calculated $F'_{P_{m+1}}(l)$ is greater than the actual $F_{P_{m+1}}(l)$.

If the maximal value of each bucket is used as the representative, then x_2, x_4 and x_5 are selected. The l-weight constrained region of P_{m+1} is decreased. So the calculated $F''_{P_{m+1}}(l)$ is smaller than the exact value of $F_{P_{m+1}}(l)$. ■

Therefore, the average value of $F'_{P_{m+1}}(l)$ and $F''_{P_{m+1}}(l)$ can be used to approximate $F_{P_{m+1}}(l)$. The following lemma guarantees the approximation quality.

LEMMA 3 (APPROXIMATION QUALITY). *Given a real value $l > 0$ and an integer $t > 0$, let $\{\phi_i = [x_{z_i}, x_{z'_i}]\}$ be a set of buckets of w_{P_m}. Let $F'_{P_{m+1}}(l)$ and $F''_{P_{m+1}}(l)$ be the l-weight probabilities computed using $\{\min(\phi_i)\}$ and $\{\max(\phi_i)\}$, respectively. Then, $F'_{P_{m+1}}(l) - F''_{P_{m+1}}(l) \leq \frac{1}{t}$. Moreover, Let $\widehat{F}_{P_{m+1}}(l) = \frac{F'_{P_{m+1}}(l)+F''_{P_{m+1}}(l)}{2}$. Then, $|\widehat{F}_{P_{m+1}}(l) - F_{P_{m+1}}(l)| \leq \frac{1}{2t}$.* ■

After obtaining the approximate probability distribution of $w_{P_{m+1}}$, we can further divide the approximate $w_{P_{m+1}}$ into buckets, and compute the approximate probability distribution of $w_{P_{m+2}}$. By applying the bucket approximation iteratively, we finally obtain an approximate l-weight probability of path $P$with quality guarantee as follows.

THEOREM 3 (OVERALL APPROXIMATION QUALITY). *Given a real value $l > 0$, an integer $t > 0$, and a path $P = \langle v_1, \ldots, v_{m+1} \rangle$ containing m edges, let $\widehat{F}_P(l)$ be the approximate l-weight probability computed using the iterative bucket approximation. Then $|\widehat{F}_P(l) - F_P(l)| \leq \frac{m-1}{2t}$.* ■

The complexity is analyzed as follows. Lemma 2 shows that there are at most $2t$ buckets constructed in the approximation. Therefore, approximating $f_{P_{i+1}}(x)$ from f_{P_i} and $f_{e_i}(x)$ ($i \geq 2$) takes $O(t \times |w_{e_i}|)$ time. The overall complexity of approximating $f_{m+1}(x)$ is $O(t \sum_{2 \leq i \leq m} |w_{e_i}|)$.

4.3 Estimating l-Weight Probabilities

The l-weight probability of a path can also be estimated using sampling. For a path $P = \langle v_1, \ldots, v_{m+1} \rangle$, let X_P be a random variable as an indicator to the event that $w_P \leq l$. $X_P = 1$ if $w_P \leq l$; $X_P = 0$ otherwise. Then, the expectation of X_P is $E[X_P] = F_P(l)$.

To estimate $E[X_P]$, we draw a set of samples uniformly with replacement. Each sample unit s is an observation of the path weight, generated as follows. At first, s is set to 0. Then, for edge $e_1 \in P$, we choose a value x_1 in w_{e_1} following the probability distribution $f_{e_1}(x)$. Then, for each edge $e_i \in P$ ($2 \leq i \leq m$), we choose a value x_i in w_{e_i} following the probability distribution $f_{e_i|e_{i-1}}(x|x_{i-1})$. The chosen value is added to s. Once the weight values of all edges are chosen, we compare s with l. If $s \leq l$, then the indicator X_P for s is set to 1, otherwise, it is set to 0.

We repeat the above procedure until a set of samples S are obtained. The mean of X_P in S is $E_S[X_P]$, which can be used to estimate $E[X_P]$. If the sample size is sufficiently large, the approximation quality is guaranteed using with the well known Chernoff-Hoeffding bound [1].

THEOREM 4 (SAMPLE SIZE). *Given a path P, for any δ ($0 < \delta < 1$), ϵ ($\epsilon > 0$), and a set of samples S of P, if $|S| \geq \frac{3 \ln \frac{2}{\delta}}{\epsilon^2}$, then $\Pr\{|E_S[X_P] - E[X_P]| > \epsilon\} \leq \delta$.* ■

The complexity of estimating $E[X_P]$ is $O(|S| \cdot |P|)$, where $|S|$ is the number of samples drawn and $|P|$ is the number of edges in P.

4.4 A Depth First Search Method

Straightforwardly, the **depth-first path search method** can be used to answer a path query $Q_l^\tau(u, v)$. The search starts at u. Each time when a new vertex v_i is visited, the weight probability mass function of the path $P = \langle u, \ldots, v_i \rangle$ is calculated using one of the three methods discussed in this section. If $v_i = v$ and $F_P(l) \leq \tau$, then P is added to the answer set.

Since weights are positive, as more edges are added into a path, the l-weight probability of the path decreases. Therefore, during the search, if the current path does not satisfy the query, then all its super paths should not be searched, since they cannot be in the answer set.

LEMMA 4 (MONOTONICITY). *Given a path P and its subpath P', and a weight threshold $l > 0$, $F_P(l) \leq F_{P'}(l)$.* ■

The overall complexity of the depth-first path search algorithm is $O(\sum_{P \in \mathcal{P}} C(P))$, where $\mathcal{P}$ is the set of visited paths and $C(P)$ is the cost of the l-weight probability calculation. If the exact method is used, then $C(P) = \prod_{e \in P} |w_e|$. If the bucket approximation method is used, then $C(P) = 4t \times \sum_{e \in P} |w_e|$, where t is the bucket parameter. If the sampling method is used, then $C(P) = |S| \times |P|$, where $|S|$ is the number of samples and $|P|$ is the number of edges in P.

The method can be extended to answer top-k path queries as following. To answer a WT top-k path query, the probability threshold τ is set to 0 at the beginning. Once a path between u and v is found, we compute its l-weight probability, and add the path to a buffer. If the buffer contains k paths, we set τ to the smallest l-weight probability of the paths in the buffer. During the search, when a new path is found between u and v and satisfying the threshold τ, it is added into the buffer. At the same time, the path in the buffer with the smallest l-weight probability is removed from the buffer. τ is updated accordingly. Therefore, during the search, the buffer always contains at most k paths. At the end of the search, the k paths in the buffer are returned.

A PT top-k path query can be answered following the similar procedure. We set the weight threshold $l = +\infty$ at the beginning. During the search, a buffer always stores at most k found paths between u and v with the smallest τ-confident weights. l is set to the value of the smallest τ-confident weight of the paths in the buffer. The k paths in the buffer at the end of the search are returned as the answers.

Limited by space, hereafter, we focus on answering probabilistic path queries and omit the details for top-k path query evaluation.

5. P*: A BEST FIRST SEARCH METHOD

Although the approximation algorithms developed in Section 4 can accelerate the probability calculation, it is still computationally challenging to search all paths in real road networks. In this section, we present the P* algorithm, a best-first search algorithm for efficient probabilistic path query evaluation.

P* carries the similar spirit as the A* algorithm [15]. It iteratively visits the next vertex that is most likely to be an answer path using a heuristic evaluation function, and stops when the rest unexplored paths have no potential to satisfy the query. However, the two algorithms are critically different due to the different types of graphs and queries.

A* is used to find the shortest path between two vertices u and v in a certain graph. Therefore, the heuristic evaluation function for each vertex v_i is simply the sum of the actual distance between u and v_i and the estimated distance between v_i and v.

P* aims to find the paths between two vertices u and v that satisfy the weight threshold l and probability threshold p in a probabilistic graph with complex correlations among edge weights. Therefore, the heuristic evaluation function for each vertex v_i is the joint distribution on a set of correlated random variables. This posts serious challenges in designing heuristic evaluation functions and calculation.

In this section, we first explain the intuition of the P* algorithm. Then, we design three heuristic evaluation functions that can be used in the P* algorithm. In Section 6, a hierarchical index is developed to support efficient query answering using the P* algorithm.

5.1 The P* Algorithm

To answer a probabilistic path query $Q_l^{\tau}(u,v)$, we search the paths from vertex u. The situation during the search is illustrated in Figure 3. Generally, before we decide to visit a vertex v_i, we want to evaluate how likely v_i would be included in an answer path. Suppose P_1 is the explored path from u to v_i, and P_2 is an unexplored path from v_i to v. Then, the l-weight probability of the path P from u to v that contains P_1 and P_2 can be determined as follows.

THEOREM 5 (SUPER PATH PROBABILITY). *Let* $P_1 = \langle u,\ldots,v_i\rangle$, $P_2 = \langle v_i,\ldots,v\rangle$ *such that* $(P_1 \cap P_2 = \{v_i\})$ *and* $e = (v_{i-1}, v_i)$. *The l-weight probability of super path*

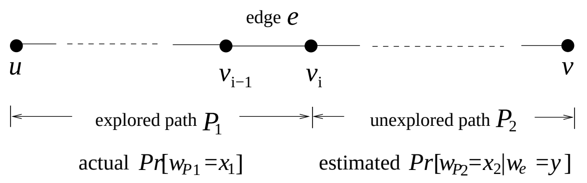


Figure 3: P* search process.

$P = \langle u,\ldots,v_i,\ldots,v\rangle$ *is*

$$F_P(l) = \sum_{x_1+y+x_2 \le l} f_{P_1,e}(x_1,y) \times f_{P_2|e}(x_2|y) \qquad (8)$$

■

In Equation 8, $f_{P_1,e}(x_1,y)$ can be easily calculated according to Theorem 2. It also can be computed using the approximation methods discussed in Section 4. However, $f_{P_2|e}(x_2|y)$, the probability distribution of P_2 given edge e, is unknown.

The objective of P* is to find a good **heuristic estimate** $h_{P_2|e}(x_2|y)$ for $f_{P_2|e}(x_2|y)$, such that $F_P(l)$ can be estimated before visiting v_i. Then, the vertex with the highest estimated $F_P(l)$ is visited next. The estimated $F_P(l)$ is used as a **heuristic evaluation function** of vertex v_i:

$$\Delta(v_i,l) = \hat{F}_P(l) = \sum_{x_1+y+x_2 \le l} f_{P_1,e}(x_1,y) \times h_{P_2|e}(x_2|y) \quad (9)$$

In order to answer a query $Q_l^{\tau}(u,v)$, P* starts from u. For any vertex v_i adjacent to u, P* calculates $\Delta(v_i,l)$ and puts v_i into a priority queue. Each time, P* removes the vertex v_i with the highest $\Delta(v_i,l)$ from the priority queue. After v_i is visited, the Δ scores of other vertices are updated accordingly. A path P is output if v is reached and $F_P(l) \ge \tau$. The algorithm stops if the priority queue is empty, or the Δ scores of the vertices in the priority queue are all smaller than τ.

In order to guarantee that P* finds all answer paths, the heuristic evaluation function should satisfy the following requirement.

THEOREM 6 (HEURISTIC EVALUATION FUNCTION). *P* outputs all answer paths if and only if for any path* $P = \langle u,\ldots,v_i,\ldots,v\rangle$, $\Delta(v_i,l) \ge F_P(l)$. ■

$\Delta(v_i,l)$ is called a **valid heuristic evaluation function** if $\Delta(v_i,l) \ge F_P(l)$.

5.2 Heuristic Estimates

It is important to find a good heuristic estimate $h_{P_2|e}(x_2|y)$ so that the evaluation function $\Delta(v_i,l)$ is valid and close to $F_P(l)$. In this subsection, we first present two simple heuristic estimates that satisfy Theorem 6. Then, we derive a sufficient condition for valid heuristic estimates, and present a more sophisticated heuristic estimate.

5.2.1 The Constant Estimate

Trivially, we can set

$$h_{P_2|e}(x_2|y) = \begin{cases} 1, & x_2 \ge 0; \\ 0, & \text{otherwise.} \end{cases}$$

Then, the evaluation function becomes

$$\Delta(v_i, l) = \sum_{x_1+y \le l} f_{P_1,e}(x_1, y) = F_{P_1}(l)$$

In this case, at each step, P* always visits the vertex v_i that maximizes the l-weight probability of $P_1 = \langle u, \dots, v_i \rangle$. The subpaths $\langle u, \dots, v_j \rangle$ whose current l-weight probabilities are smaller than τ are pruned.

Example 7 (The constant estimate). Consider the probabilistic graph in Figure 1. To answer a query $Q_{15}^{0.3}(A, E)$, the search starts from A. Using the constant estimate, the evaluation functions for B and C are $\Delta(B, 15) = \sum_{x_1 \le 15} f_{AB}(x_1) = 0.6$, and $\Delta(C, 15) = \sum_{x_1 \le 15} f_{AC}(x_1) = 1.0$. Since $\Delta(C, 15)$ is larger, C should be explored first. ■

The constant estimate is easy to construct. But clearly, it does not consider the relationship between the current vertex and the unvisited end vertex v.

5.2.2 The Min-Value Estimate

To incorporate more information about the unexplored paths into decision making, we consider the minimal possible weights from the current vertex v_i to the end vertex v. We construct a certain graph G' with the same set of vertices and edges as the uncertain graph G. For each edge e, the weight of e is the minimal value in w_e of G. Let l_i^{min} be the weight of the shortest path between v_i and v, excluding the visited edges. We set the estimation function

$$h_{P_2|e}(x_2|y) = \begin{cases} 1, & x_2 = l_i^{min}; \\ 0, & \text{otherwise.} \end{cases}$$

The evaluation function becomes

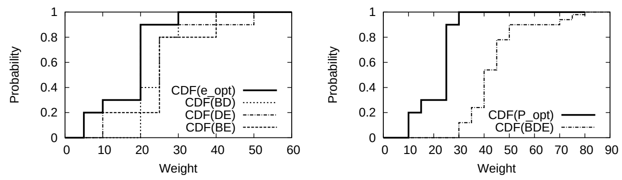
$$\begin{aligned} \Delta(v_i, l) &= \textstyle\sum_{x_1+y+l_i^{min} \le l} f_{P_1,e}(x_1, y) \times h_{P_2|e}(l_i^{min}|y) \\ &= \textstyle\sum_{x_1+y \le l-l_i^{min}} f_{P_1,e}(x_1, y) \\ &= F_{P_1}(l - l_i^{min}) \end{aligned}$$

The search algorithm always visits the vertex v_i that maximizes the $(l - l_i^{min})$-weight probability of $P_1 = \langle u, \dots, v_i \rangle$. The subpaths $\langle u, \dots, v_j \rangle$ whose $(l - l_j^{min})$-weight probabilities are smaller than τ are pruned.

Example 8 (The min-value estimate). Consider the probabilistic graph in Figure 1 and query $Q_{15}^{0.3}(A, E)$ again. Using the min-value estimate, we construct a certain graph G' with weights $w_{AB} = 5$, $w_{AC} = 5$, $w_{BD} = 20$, $w_{BE} = 5$, $w_{CE} = 10$, and $w_{DE} = 10$.

The shortest distance from B to E among the unvisited edges in G' is 5. The evaluation function for B is $\Delta(B, 15) = \sum_{x_1 \le 15-5} f_{AB}(x_1) = 0.3$. The shortest distance from C to E in G' is 10. The evaluation function for C is $\Delta(C, 15) = \sum_{x_1 \le 15-10} f_{AC}(x_1) = 0.2$. Since $\Delta(B, 15)$ is larger, B should be explored first. Moreover, there is no need to visit C further because $\Delta(C, 15) < \tau$. ■

Compared to the constant estimate, the min-value estimate considers more information about the unexplored paths, and gives a priority to the vertices that are closer to the end vertex v and more likely to satisfy the query. The drawback of the min-value estimate is that it does not consider the probabilistic distribution of the paths not explored yet. Next, we tackle this weakness.



(a) CDF of $w_{e_{opt}}$. (b) CDF of $w_{P_{opt}}$.

Figure 4: CDFs of "virtual optimal" edges and paths.

5.2.3 The Stochastic Estimate

How can we incorporate into the heuristic estimates the probability distribution information of the paths not explored yet? Let $\mathcal{P}_2 = \{P_{2_1}, \dots, P_{2_m}\}$ be all paths from v_i to v that do not contain any visited vertices except for v_i. We can construct a "virtual path" $P_{opt} = \langle v_i, \dots, v \rangle$ such that for any real number x and $P_{2_i} \in \mathcal{P}_2$, $F_{P_{opt}}(x) \ge F_{P_{2_i}}(x)$.

Definition 3 (Stochastic order [24]). *For two random variables r_1 and r_2 with distribution functions $F_{r_1}(x)$ and $F_{r_2}(x)$, respectively, r_1 is smaller than r_2 in stochastic order if for any real number x, $F_{r_1}(x) \ge F_{r_2}(x)$.* ■

P_{opt} **stochastically dominates** all paths in $\mathcal{P}_2$, if $w_{P_{opt}}$ is smaller than any w_{P_j} $(1 \le j \le m)$ in stochastic order. An example is shown in Figure 4(b). P_{opt} stochastically dominates $\langle B, D, E \rangle$. The following theorem shows that P_{opt} can be used to define a valid heuristic evaluation function.

Theorem 7 (Sufficient condition). *Let $\mathcal{P}_2 = \{P_{2_1}, \dots, P_{2_m}\}$ be all paths between v_i and v in graph G. For a path $P_{opt} = \langle v_i, \dots, v \rangle$, if P_{opt} stochastically dominates all paths in $\mathcal{P}_2$, then*

$$\Delta(v_i, l) = \sum_{x_1+y+x_2 \le l} f_{P_1,e}(x_1, y) \times f_{P_{opt}}(x_2)$$

is a valid heuristic evaluation function for P.* ■

More than one P_{opt} can be constructed. Here we present a simple three-step construction method that ensures the resulting path is a stochastically dominating path for $\mathcal{P}_2$.

Step 1: Constructing the path. We find the path P_{2_i} in $\mathcal{P}_2$ with the least number of edges. Let n be the number of edges in P_{2_i}. We construct $n-1$ virtual vertices $\hat{v}_1, \dots, \hat{v}_{n-1}$. Let $P_{opt} = \langle v_i, \hat{v}_1, \dots, \hat{v}_{n-1}, v \rangle$. Thus, P_{opt} has the least number of edges among all edges in $\mathcal{P}_2$.

For example, in the probabilistic graph in Figure 1, there are two paths between B and E: $P_1 = \langle B, E \rangle$ and $P_2 = \langle B, D, E \rangle$. Since P_1 only contains one edge, the path P_{opt} should also contain only one edge.

Step 2: Assigning edge weights. Let $\mathcal{E}$ be the set of edges in $\mathcal{P}_2$. We want to construct the weight of an edge e_{opt} in P_{opt} such that e_{opt} stochastically dominates all edges in $\mathcal{E}$.

We construct $w_{e_{opt}}$ as follows. At the beginning, we set $w_{e_{opt}} = \emptyset$. Then, we represent each sample $x \in w_e$ $(e \in \mathcal{E})$ as a pair $(x, F_e(x))$. We sort all samples in value ascending order. If there are two samples having the same value, we only keep the sample with the larger cumulative probability.

We scan the samples in the sorted list. If $w_{e_{opt}} = \emptyset$, then we add the current sample $(x, F_e(x))$ into $w_{e_{opt}}$. Otherwise, let $(x', F_{e_{opt}}(x'))$ be the sample with the largest value x' in $w_{e_{opt}}$. We add the current sample $(x, F_e(x))$ into $w_{e_{opt}}$ if $x > x'$ and $F_e(x) > F_{e_{opt}}(x')$. Last, we assign weight $w_{e_{opt}}$ to each edge $e_i \in P_{opt}$ $(1 \leq i \leq n)$.

Example 9 (Assigning edge weights). Consider the probabilistic graph in Figure 1. There are two paths between B and E: $P_1 = \langle B, E\rangle$ and $P_2 = \langle B, D, E\rangle$. $\mathcal{E} = \{BD, DE, BE\}$. The probability mass function of the constructed weight $w_{e_{opt}}$ is $\{5(0.2), 10(0.1), 20(0.6), 30(0.4)\}$. $w_{e_{opt}}$ is smaller than w_{BD}, w_{DE} and w_{BE} in stochastic order. The weight cumulative distribution functions of those edges are shown in Figure 4(a). ■

Step 3: Assigning the path weight. Since $w_{P_{opt}} = \sum_{1\leq i\leq n} w_{e_i}$, the distribution of $w_{P_{opt}}$ depends on the marginal distribution of w_{e_i} and the correlations among w_{e_i}'s. If the correlations among edges are explicitly represented, then we just construct $w_{P_{opt}}$ according to the correlation function.

In most cases, the correlation functions among weights are not available. In such a case, we construct a path weight $w_{P_{opt}}$ that stochastically dominates all possible weights given the same $w_{e_1}, \dots, w_{e_n}$, as defined in the following rule, which follows with Lemma 1 immediately.

Theorem 8 (Upper bound). *For path P_{opt} containing edges $e_1, \dots, e_n$, let x_i^{min} be the smallest sample of weight w_{e_i}, and $l_{min} = \sum_{1\leq i\leq n} x_i^{min}$. Then,*

$$F_{P_{opt}}(l) \leq \min_{1\leq i\leq n}\{F_{e_i}(l - l_{min} + x_i^{min})\}.$$

■

Therefore, for any value l, we assign

$$F_{P_{opt}}(l) = \min_{1\leq i\leq n}\{F_{e_i}(l - l_{min} + x_i^{min})\}.$$

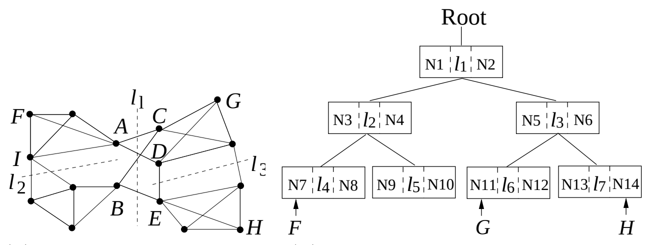
The heuristic evaluation function is

$$\Delta(v_i, l) = \sum_{x_1+y+x_2\leq l} f_{P_1,e}(x_1, y) \times f_{P_{opt}}(x_2).$$

Example 10 (Assigning path weights). Continuing Example 9, since P_{opt} only contains one edge, the probability distribution of P_{opt} is the same as that of e_{opt}.

As another example, suppose P_{opt} contains two edges e_1 and e_2 with the same weight $w_{e_{opt}}$. How can we calculate the probability distribution of P_{opt}? The sum of the minimal samples of w_{e_1} and w_{e_2} is 10. Then, $F_{P_{opt}}(20)$ is $\min\{F_{e_1}(20 - 10) = 0.3, F_{e_2}(20 - 10) = 0.3\} = 0.3$. The cumulative distribution function of P_{opt} is shown in Figure 4(b). ■

Using the stochastic estimate, in each step, the search algorithm visits the vertex v_i whose heuristic evaluation function using optimal virtual path P_{opt} is the largest. However, constructing P_{opt} requires enumerating all possible paths from v_i to v, which is computationally prohibitive. Therefore, in the next section, we discuss how to approximate P_{opt} using a hierarchical partition tree index in large road networks.



(a) A graph partitioning. (b) A hierarchical partition tree.

Figure 5: Graph partitioning and HP-tree.

6. A HIERARCHICAL INDEX FOR P*

In this section, we introduce a hierarchical partition tree to index the vertices of a graph and maintain the information of weight probability distribution for efficient query answering using the P* algorithm.

6.1 HP-Tree Index

A **graph partitioning** divides the vertices of a graph into subsets of about the same size, such that there are few edges spanning between subsets [18]. Each subset can be partitioned recursively. In general, an m-**partitioning** recursively partitions a graph such that at each step a subset is partitioned into m smaller subsets (unless there are less than m vertices in a subset), until each subset has at most d vertices, where $d \geq 1$ is a user specified parameter.

Figure 5(a) illustrates a 2-partitioning on a graph, where all vertices are divided into two subsets at the first level, separated by line l_1. Among the vertices on the left of l_1, only A and B are connected to the vertices on the right subset. A and B are called the **border vertices**. Similarly, C, D and E are the border vertices in the right subset. The left subset is further partitioned into two smaller subsets by line l_2, and the right subset is further partitioned into two smaller subsets by l_3.

Given a graph G and an m-partition H on G, a **hierarchical partition tree** (**HP-tree** for short) is an m-nary tree that indexes the vertices in G according to H. Each leaf node in an HP-tree represents a vertex, and each non-leaf node represents a set of vertices. Figure 5(b) shows the HP-tree corresponding to the 2-partitioning in Figure 5(a). Please note that not all leaf nodes are drawn.

An HP-tree can be constructed top-down starting from the root which represents the complete set of vertices in graph G. Then, the graph is partitioned recursively. If we apply a linear time heuristic graph partitioning method [14], then constructing an HP-tree using m-partitioning takes $O(n \log_m \frac{n}{d})$ time.

6.2 Approximating the Min-Value Estimate

To approximate a min-value estimate, we store auxiliary information for each node in an HP-tree as follows. For each leaf node N_L representing a vertex v and its parent node N associated with a set of vertices V_N, we compute the weight of the shortest path between v and each border vertex of V_N. The smallest weight is stored in N_L. Then, for node N and each of its ancestor nodes N_A, let V_A be the set of vertices associated with N_A. We compute the shortest paths between each border vertex of V_N and each border vertex of V_A. The smallest weight is stored in N.

For example, in the HP-tree shown in Figure 5(b), F is a leaf node, and we store the smallest weight of shortest paths from F to the border vertices of $N7$, which is w_1 in Figure 5(a). Then, for node $N7$, we compute the smallest weight of the shortest paths between any border vertex in $N7$ and any border vertex in $N3$. Since $N7$ and $N3$ share a common border vertex I, the smallest weight is 0. The smallest weight of the shortest paths between $N7$ and $N1$ is w_2 in Figure 5(a).

The min-value estimate for a vertex u can be approximated as follows. Let N_u and N_v be the parent node of u and v, respectively. Let N be the lowest common ancestor node of N_u and N_v. Then, the weight of any path between u and v is at least $w(u, N_u)+w(N_u, N)+w(v, N_v)+w(N_v, N)$, where $w(u, N_u)$ and $w(v, N_v)$ are the smallest weight from u and v to N_u and N_v, respectively, and $w(N_u, N)$ and $w(N_v, N)$ are the smallest weight from N_u and N_v to N, respectively. Searching the lowest common ancestor node of two vertices takes $O(\log_m \frac{n}{d})$ time.

For example, in Figure 5(b), the lowest common ancestor node of F and G is the root node, which is partitioned by l_1. Thus, the weight of the shortest path between F and G is at least $w_1 + w_2 + w_3 + w_4$. It can be used as the min-value estimate of F with respect to the end vertex G.

6.3 Approximating the Stochastic Estimate

To approximate the stochastic estimate, we store two pieces of information for each node in the HP-tree.

For each leaf node N_L representing a vertex v and its parent node N associated with a set of vertices V_N, we first compute the shortest path between v and each border vertex in V_N. The number of edges is stored in leaf node N_L. Then, we compute a weight stochastically dominating all weights in V_N, and store it as the "optimal weight" of the node.

For an intermediate node N and each of its ancestor nodes N_A, let V_A be the set of vertices associated with N_A. We first compute the shortest edge paths between each border vertex of V_N and each border vertex of V_A. The number of edges in the path is stored in N. Second, we compute a weight that stochastically dominates all weights of the edges between the border vertices in V_N and V_A. It is stored as the "optimal weight" of the node.

Therefore, the stochastic estimate can be approximated by slightly changing the three steps in Section 5.2.3 as follows. In the first step, in order to find the smallest number of edges from a vertex v_i to v, we compute the lower bound of the least number of edges. Second, instead of finding an edge weight that stochastically dominates all edges in $\mathcal{P}_2$, we use the optimal weights stored in nodes. The third step remains the same. In this way, we can compute a path that has a smaller number of edges and weights that dominate all weights in P_{opt}.

6.4 Maintaining HP-Trees

An HP-tree can be maintained incrementally as edge weights change. For each node in an HP-tree that contains a set of edges, an optimal weight stochastically dominating all edge weights is stored. The optimal weight can be constructed using a set of (value, probability) pairs that are the skyline points among the (value, probability) pairs of all edges. Therefore, the optimal weight in a sliding window can be maintained using any efficient skyline maintenance algorithms for streams, such as [28].

7. EXPERIMENTAL RESULTS

In this section, we report a systematic empirical study. All experiments were conducted on a PC computer with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk, running the Microsoft Windows XP Professional Edition operating system. Our algorithms were implemented in Microsoft Visual Studio 2005. The graph partitioning algorithm in METIS 4.0.1 (`http://glaros.dtc.umn.edu/gkhome/views/metis`) and Dijkstra's Shortest Path Algorithm in the Boost C++ Libraries (`http://www.boost.org/`) were used in the implementation of HP-trees.

7.1 Experiment Setup

We tested the efficiency, the memory usage, the approximation quality, and the scalability of the algorithms on the following five real road network data sets (available at `http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm`): City of Oldenburg Road Network (*OL*) ($6,104$ nodes and $7,034$ edges), City of San Joaquin County Road Network (*TG*) ($18,262$ nodes and $23,873$ edges), California Road Network (*CAL*) ($21,047$ nodes and $21,692$ edges), San Francisco Road Network (*SF*) ($174,955$ nodes and $223,000$ edges), and North America Road Network (*NA*) ($175,812$ nodes and $179,178$ edges).

The available weight of each edge in the above real road networks is certain. To simulate an uncertain road network, we generated a set of uncertain samples for each edge following the Normal distribution and the Gamma distribution.

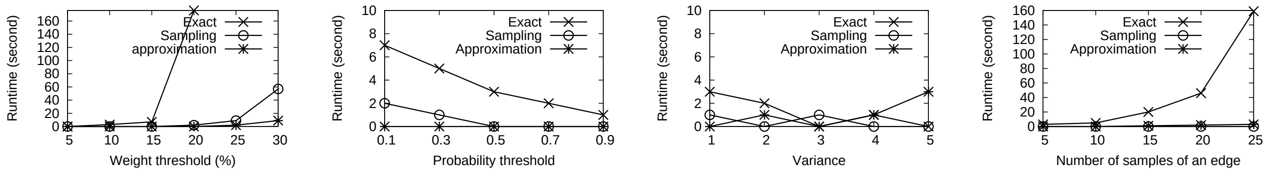
In the Normal distribution $N(\mu, \sigma)$, μ is the original edge weight in the certain graphs and σ is the variance. σ was generated for different edges following the Normal distribution $N(\mu_\sigma, \sigma_\sigma)$, where $\mu_\sigma = xR$, x was varied from 1% to 5%, and R is the range of weights in the data sets. By default, μ_σ was set to $1\% \cdot R$. This simulation method follows the findings in the existing studies on traffic simulations [16], which indicate that the travel time on paths in road networks follows the Normal distribution in short time intervals.

To simulate the travel time in real road networks using the Gamma distribution $\Gamma(k, \theta)$, as suggested in [22], we set $\theta = 0.16$ and $k = \frac{\mu}{\theta}$, where μ is the original edge weight in the certain graphs. Since the experimental results on the weights under the Gamma distribution and the Normal distribution are highly similar, limited by space, we only reported here the results on the data sets with the Normal distribution.

After generating the edge weights $w_{e_1} = \{x_1, \ldots, x_m\}$ and $w_{e_2} = \{y_1, \ldots, y_n\}$, the joint distribution $f_{e_1,e_2}(x_i, y_j)$ was randomly generated from the interval $[0, p_{ij})$, where $p_{ij} = \min\{f_{e_1}(x_i), f_{e_2}(y_j)\}$ ($i = 0$, $j = 0$), and $p_{ij} = \min\{f_{e_1}(x_i) - \sum_{1 \le s \le j-1} f_{e_1,e_2}(x_i, y_s), f_{e_2}(y_j) - \sum_{1 \le t \le i-1} f_{e_1,e_2}(x_t, y_j)\}$, ($i > 0$ or $j > 0$).

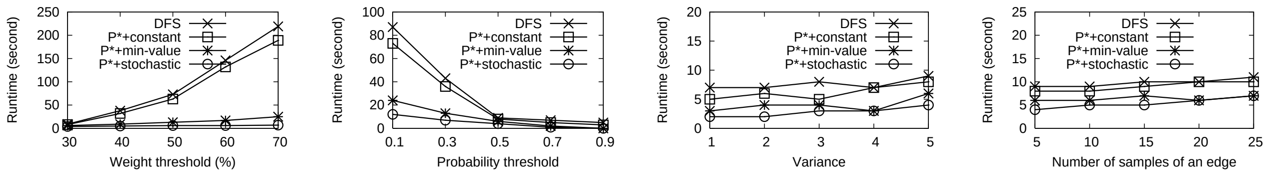
The path queries were generated as follows. The weight threshold was set to $x\%$ of the diameter d of the network, where d is the maximal weight of the shortest paths among pairs of nodes. The probability threshold was varied from 0.1 to 0.9. The start and the end vertices were randomly selected.

By default, the number of samples of each edge was set to 5, the weight threshold l was set to $10\% \cdot d$, and the probability threshold τ was set to 0.5. 500 samples were used in the sampling algorithm to estimate the weight probability distribution of each path. In the hierarchical approxima-



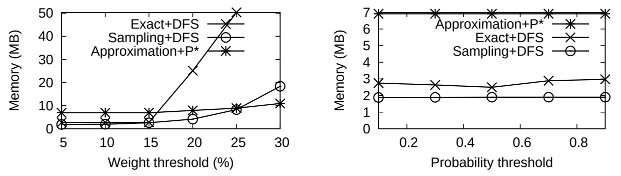
(a) Runtime vs. l. (b) Runtime vs. prob threshold. (c) Runtime vs. variance. (d) Runtime vs. sample size.

Figure 6: Efficiency of weight probability calculation methods.



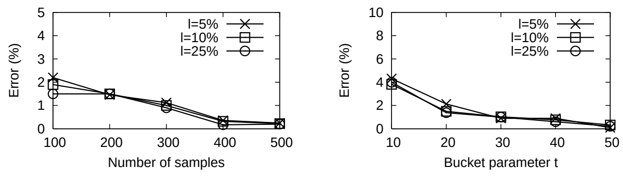
(a) Runtime vs. l. (b) Runtime vs. prob threshold. (c) Runtime vs. variance. (d) Runtime vs. sample size.

Figure 7: Efficiency of path search methods.



(a) Memory vs. l. (b) Memory vs. prob threshold.

Figure 8: Memory usage.



(a) The sampling algorithm. (b) The bucket approx method.

Figure 9: Approximation quality.

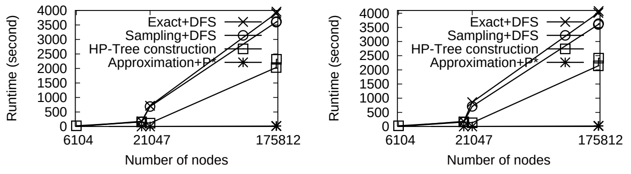
tion algorithm, the bucket parameter t was set to 50, and 2-partitionings were used to construct HP-trees. For each different parameter setting, we ran 20 path queries and reported the average results.

7.2 Efficiency and Memory Usage

Using data set *OL*, we evaluated the efficiency of the three probability calculation methods, the exact method (*Exact*), the bucket approximation method (*Approximation*), and the sampling based method (*Sampling*). The depth-first path search was used. The results are shown in Figure 6. Clearly, the bucket approximation method and the sampling based method are more efficient than the exact algorithm.

Particularly, the runtime increases as the weight threshold increases (Figure 6(a)), since a larger weight threshold qualifies more paths in the answer set. The runtime of all three algorithms decreases when the probability threshold increases (Figure 6(b)), because fewer paths can pass the threshold when the probability threshold becomes higher. The runtime of the algorithms decreases slightly as the variance of the weight samples increases (Figure 6(c)). The runtime of the exact algorithm increases significantly when the number of samples of each edge increases, but the runtime of the sampling algorithm and the bucket approximation algorithm remains stable (Figure 6(d)) thanks to the efficient probability approximation techniques.

We also tested the efficiency of the P* algorithm against the depth-first search (*DFS*) (Figure 7). Three heuristic estimates were used: the constant estimate (*P*+constant*), the min-value estimate (*P*+min-value*), and the stochastic estimate (*P*+stochastic*). The bucket approximation method was used to compute the path probability distribution. To show the difference in efficiency of the different methods



(a) Normal distribution. (b) Normal distribution with noise.

Figure 10: Scalability.

more clearly, we increased the weight threshold to $30\% \cdot d$.

Clearly, the P* method is more efficient than the depth-first search method thanks to the heuristic path evaluation during the search. Among the three types of heuristic estimates, the stochastic estimate is the most effective, because it uses the weight probability distribution of the paths not explored yet to guide the search. The HP-tree construction takes 25 seconds on this data set.

The memory usage for different algorithms is shown in Figure 8. The memory requirement in the exact probability calculation method with DFS increases rapidly when the weight threshold increases, since longer paths are explored with a larger weight threshold. However, the memory usage in P* with the bucket approximation probability calculation is stable, because the space used for an HP-tree does not depend on the weight threshold.

7.3 Approximation Quality and Scalability

Using data set *OL*, we tested the approximation quality of the sampling algorithm and the bucket approximation algorithm. In the same parameter setting as in Figure 6, the precision and the recall of all queries are computed. Since they are all 100%, we omit the figures here.

The average approximation error of the l-weight probability computed in the two algorithms is shown in Figure 9. For any path P, the approximation error is defined as $\frac{|\hat{F}_P(l)-F_P(l)|}{F_P(l)}$, where $\hat{F}_P(l)$ and $F_P(l)$ are the approximate and exact l-weight probabilities of P, respectively.

The error rate of the sampling method is always lower than 3%, and is very close to 0 when a set of 500 samples are used (Figure 9(a)). The average error rate of the bucket approximation method decreases from 4.31% to 0.1% when the bucket parameter t increases from 10 to 50 (Figure 9(b)).

Figure 10 shows the scalability of the three algorithms on the five real road network data sets. Figure 10(a) shows the results for weights following the Normal distribution. Figure 10(b) is the results on the data with the Normal distribution and 10% noise. That is, 10% of the edges have a sample drawn from the uniform distribution in $[x_{min}, x_{max}]$, where x_{min} and x_{max} are the minimal and maximal weights in the original road network, respectively. The results in Figures 10(a) and (b) are similar. All three algorithms are scalable, and the hierarchical approximation algorithm has very short runtime (20 seconds on the largest data set, the North America Road Network (*NA*) with $175,812$ nodes). Although constructing the HP-tree takes around $2,000$ seconds in this case, it is constructed only once offline.

8. CONCLUSIONS

In this paper, we studied the novel probabilistic path queries in road networks with uncertain weights. Three methods for efficient probability calculation and a best-first search algorithm were proposed. The experimental results on real road networks verified the effectiveness and efficiency of the probabilistic path queries and our methods.

As future work, we will explore temporal correlations and other types of uncertainty in road networks.

9. REFERENCES

[1] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *STOC'77*.

[2] H. Bast *et al.* Transit: Ultrafast shortest-path queries with linear-time preprocessing. In C. Demetrescu *et al.* (editors). *9th DIMACS Implementation Challenge — Shortest Path*, 2006.

[3] H. Bast *et al.* In transit to constant time shortest-path queries in road networks. In *ALENEX'07*.

[4] E. P. F. Chan and J. Zhang. A fast unified optimal route query evaluation algorithm. In *CIKM'07*.

[5] A. Chang and E. Amir. Reachability under uncertainty. In *UAI'07*.

[6] Y. Fan *et al.* Arriving on Time. *CJ. Optimization Theory and Applications*, 127(3):497–513, 2005.

[7] Y. Fan *et al.*, Shortest paths in stochastic networks with correlated link costs, *Computers and Mathematics with Applications*, vol. 49, no. 9-10, pp. 1549–1564, 2005.

[8] Y. Fan and Y. Nie. Optimal Routing for Maximizing the Travel Time Reliability. *Networks and Spatial Economics*, 6(3-4):333–344, 2006.

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

[10] L. Fu *et al.* Heuristic shortest path algorithms for transportation applications: State of the art. *Computers and Operations Research*, 33(11):3324–3343, 2006.

[11] J. Ghosh *et al.* On a routing problem within probabilistic graphs and its application to intermittently connected networks. In *INFOCOM'07*.

[12] K. Goebel and A. M. Agogino. Sensor validation and fusion for automated vehicle control using fuzzy techniques. *Journal of Dynamic Systems, Measurement, and Control*, 123(1):145–146, 2001.

[13] H. Gonzalez *et al.* Adaptive fastest path computation on a road network: a traffic mining approach. In *VLDB'07*.

[14] M. T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comput. Syst. Sci.*, 51(3):374–389, 1995.

[15] P. Hart *et al.* A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, July 1968.

[16] R. R. He *et al.* Temporal and Spatial Variability of Travel Time. *Center for Traffic Simulation Studies. Paper UCI-ITS-TS-02-14.*

[17] N. Jing *et al.* Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *IEEE Trans. on Knowl. and Data Eng.*, 10(3):409–432, 1998.

[18] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, Febuary 1970.

[19] A. B. Kurzhanski and P. Varaiya. On reachability under uncertainty. *SIAM J. Control Optim.*, 41(1):181–216, 2002.

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

[21] E. Nikolova *et al.* *Stochastic Shortest Paths Via Quasi-convex Maximization.* In *ESA'06*.

[22] A. Polus. A study of travel time and reliability on arterial routes . *Transportation*, 8(2):141–151, 1979.

[23] D. D. M. L. Rasteiro and A. J. B. Anjo. Optimal paths in probabilistic networks. *Journal of Mathematical Sciences*, 120(1):974–987, 2004.

[24] R. Righter. Scheduling. In *Stochastic Orders and Their Applications*, 1994.

[25] H. Samet *et al.* Scalable network distance browsing in spatial databases. In *SIGMOD'08*.

[26] P. Sanders and D. Schultes. Highway hierarchies hasten exact shortest path queries. In *ESA'05*.

[27] P. Sanders and D. Schultes. Engineering highway hierarchies. In *ESA'06*.

[28] Y. Tao and D. Papadias. Maintaining Sliding Window Skylines on Data Streams. In *TKDE*, vol. 18, no. 3, pp. 377 – 391, 2006.

[29] Y. Wu *et al.* A shortest path algorithm based on hierarchical graph model. In *Intelligent Transportation Systems*, pages 1511 – 1514, 2003.