# NRP: An Efficient Index for Stochastic Routing in Road Networks

Libin Wang[†], Raymond Chi-Wing Wong[‡]
[†]School of Computer Science, Northwestern Polytechnical University, Xi'an, Shaanxi, China
[‡]CSE Department, The Hong Kong University of Science and Technology, Hong Kong SAR, China
{lbwang95@163.com, raywong@cse.ust.hk}

***Abstract*—The pervasiveness of shortest path queries is evident in real life, particularly in online mapping applications. However, in practice, the travel times of road segments can be uncertain due to various reasons, such as traffic congestion, which leads to the shortest path not to be the fastest, resulting in an unreliable path. The *Reliable Shortest Path* (RSP) query has been developed to fulfill individuals' reliability requirements by considering travel times as random variables. Extensive solutions have been proposed to efficiently find RSPs in stochastic road networks. However, they are either unscalable to large networks or incapable of handling rapid streams of routing queries. In this paper, we propose an efficient index-based solution for RSP queries, called *Non-dominated Reliable Path* (NRP). It stores partial path answers to support fast query processing and utilizes several tailored pruning techniques that can significantly reduce the query time. Experiments conducted on large city road networks verified the superiority of our solution, which can answer each query in around 100 microseconds and beat competitors by orders of magnitude.**

## I. Introduction

The point-to-point shortest path query is the cornerstone of many spatial applications, such as navigation systems, ride-hailing platforms, and food delivery services. Online mapping platforms (e.g., Google Maps) receive numerous users' shortest path queries daily. Users essentially seek traveling guidance from these platforms to reach their destinations. The query is answered by a recommended path with the "shortest" travel time based on the current estimated travel times of all road segments. However, the estimation can be inaccurate due to various reasons, such as traffic congestion, roadway geometry, and weather conditions, which makes the path's total travel time *unreliable* [1]–[3]. Simply ignoring the *uncertainty* of travel times can lead to many unfavorable consequences, including unpredictable delays, increased fuel consumption, and risky driving behaviors. For example, the shortest or fastest path that uses deterministic values of roads' travel times as edge weights is *unreliable* when it traverses roads with large variances during rush hour, as shown in our case study (Section VI-B3). On the other hand, recent studies revealed that travel time reliability plays a critical role in travelers' decision-making of final path choices [4], [5]. Therefore, it is more realistic to take the uncertainty into account to find the *reliable shortest path* (RSP).

We focus on the prevailing definition of RSP first proposed by [6]. In a stochastic road network, where road segments and their junctions are modeled as edges and vertices, respectively, edges' travel times are characterized by random variables. It further implies that the travel time of each path (by summing up its edges' travel times) is also a random variable. For the reliability of the travel time, individuals could have varying requirements in different scenarios. For example, during rush hour, a person leaving for the airport would avoid the risk and care much about the *variance* of the travel time. However, one commuter going home wants to minimize the *expected travel time* even if the variances of some roads' travel times are high. Thus, it is more flexible to introduce a user-defined confidence level $\alpha \in (0, 1)$ to represent the degree of *reliability* requirement. To evaluate each path, we would find an upper bound of the path's travel time (as the path's metric value) such that the probability that its travel time is at most the upper bound should be at least the confidence level $\alpha$.

The main challenge of finding RSPs efficiently lies in handling the stochastic travel times, particularly with correlations. There have been extensive routing solutions in stochastic networks [1], [7]–[17]. Most of them searched the path from the source to the destination incrementally by using some path dominance conditions to prune the search space [7]–[16]. For example, [1] applied matrix operations on large correlation matrices. The state-of-the-art TBS [16] uses the travel times of the reversed paths from the destination to prune the search space. However, their routing algorithms are inefficient in large networks since they need to search the network edges one by one. They cannot fulfill the demand of answering RSP queries efficiently in large stochastic networks.

Motivated by the above limitation, we propose by far the fastest index-based solution for RSP queries. Specifically, we propose the *Non-dominated Reliable Path* (NRP) index that mainly stores some partial path answers (i.e., NRP) that are superior under some conditions. They could be quickly retrieved by label lookups and concatenated to form the final RSP. By storing only NRP, we avoid the network search during query processing and prune many useless paths that cannot be a part of the RSP under different conditions. We also explore how to maintain the index since travel times could change. The experimental results show that our solution has an average query processing time of 100 microseconds on New York's road network of standard cities and runs faster than state-of-the-art competitors by orders of magnitude.

We summarize our contributions as follows.

- We propose the NRP index that utilizes several pruning

techniques to support fast RSP query processing.
- We introduce the mechanism of maintaining the index when the travel time distribution changes.
- We empirically demonstrate the superiority of NRP on large real-world city road networks. It is orders of magnitude faster than state-of-the-art baselines.

The remainder of the paper is organized as follows. Section II defines the problem. Section III illustrates our query processing. Section IV and Section V explain the details of index construction and maintenance, respectively. Section VI shows experimental results. Section VII reviews the related work, and Section VIII concludes this paper.

## II. Preliminaries

### A. Problem Definitions

*Definition 1 (Stochastic Road Network):* A stochastic road network is a connected undirected graph, denoted by $G(V, E)$, where $V$ and $E$ are the vertex and edge sets, respectively. At any time, each edge $e \in E$ is associated with a continuous random variable, denoted by $W_e$, representing its travel time. Its cumulative distribution function (CDF) is defined by $F_e(w) = \Pr(W_e \leq w)$ (where $w \in \mathbb{R}^+$), which is the probability of the event that $e$'s travel time is at most $w$.

Following [2], [7], [18]–[20], we assume that edges' travel times are normal variables, which can be obtained by modern machine learning techniques [21]–[23] and is out of the scope of this paper. Note that this hypothesis about normal variables has been verified on historical traffic data [24], [25]. The travel times could be either independent or correlated variables. We will separately discuss the two cases for clarity, but all the proposed techniques can be applied to a network where both cases exist. For correlated variables, we define the covariance between $W_{e_i}$ and $W_{e_j}$ as $\sigma_{e_i,e_j} = \mathbb{E}[(W_{e_i} - \mathbb{E}[W_{e_i}])(W_{e_j} - \mathbb{E}[W_{e_j}])]$. In addition, the distributions of travel times can change and be arbitrarily different normal variables. For example, the mean and variance are usually higher during rush hour than in light traffic. We omit the notation of time associated with travel times at different times since we answer queries based on current travel times. We will discuss how to handle the changes in Section V.

*Example 1:* An example road network is shown in Figure 1. Beside each edge $e$, we give its travel time $W_e \sim \mathcal{N}(\mu_e, \sigma_e^2)$ with its mean $\mu_e$ and standard deviation $\sigma_e$. For example, both the mean $\mu$ and standard deviation $\sigma$ of the edge $(v_6, v_8)$ are 2 and 2, respectively, where its variance $\sigma^2$ is 4. For correlated variables, we additionally assume that $\sigma_{(v_6,v_4),(v_4,v_7)} = -2$, $\sigma_{(v_4,v_7),(v_7,v_5)} = 1$, and other covariances are 0.

*Definition 2 (Path):* A path $p$ is a finite sequence of vertices $p = (v_0, v_1, \ldots, v_k)$ such that for $i = 1, 2, \ldots, k$, each $(v_{i-1}, v_i)$ is an edge. A path is called an *s-t path* if $v_0 = s$ and $v_k = t$. Let $W_p$ denote the travel time of path $p$, defined by $W_p = \sum_{i=1}^{k} W_{(v_{i-1},v_i)}$, with its CDF $F_p(w) = \Pr(W_p \leq w)$ where $w \in \mathbb{R}^+$. The concatenation of two paths $p_1$ and $p_2$ is denoted by $p_1 \oplus p_2$ when $p_1$ and $p_2$ share a common end vertex. Given a set $P_1$ of paths and a set $P_2$ of paths, $P_1 \oplus P_2$ is defined to be $\{p_1 \oplus p_2 : p_1 \in P_1, p_2 \in P_2\}$.

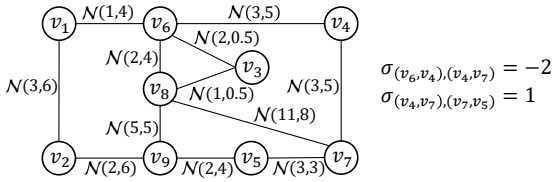



Fig. 1: A road network with random travel times

*Definition 3 (RSP Query):* Given a network $G$, a source $s \in V$, a destination $t \in V$, and a confidence level $\alpha \in (0, 1)$, the reliable shortest path (RSP) query is answered by the $s$-$t$ path $p^*$ that minimizes an upper bound $w$ of the path's travel time $W_{p^*}$ such that $F_{p^*}(w) = \Pr(W_{p^*} \leq w) \geq \alpha$. In other words, $p^*$ gives the minimum $w$ such that the probability that $W_{p^*}$ is at most $w$ is at least $\alpha$.

For a fixed path $p$, we can minimize $w$ w.r.t. path $p$ by setting $F_p(w) = \alpha$ since $F_p(w)$ monotonically increases as $w$ increases. If the inverse function $F_p^{-1}(\cdot)$ exists, then $w = F_p^{-1}(\alpha)$, and the objective of minimizing $w$ is equivalent to minimizing $F_p^{-1}(\alpha)$, no matter for the independent or correlated case. Since $p$'s travel time $W_p$ is a normal variable, $F_p^{-1}(\alpha) = \mu_p + Z_\alpha \sigma_p$, where $\mu_p$ and $\sigma_p$ are the mean and standard deviation of $W_p$, respectively, and $Z_\alpha \in \mathbb{R}$ is the inverse CDF of the standard normal distribution at $\alpha$ ($Z_\alpha = 0$ when $\alpha = 0.5$). We can find the value of $Z_\alpha$ by looking up the standard normal table (also called the Z table) or using numerical function approximation.

*Example 2:* Consider an RSP query with $s = v_6$, $t = v_5$, and $\alpha = 0.95$. For the independent case, $p^*$ can be $(v_6, v_8, v_9, v_5)$ or $(v_6, v_4, v_7, v_5)$ with the same $W_{p^*} \sim \mathcal{N}(9, 13)$ and the minimum $F_{p^*}^{-1}(\alpha) = 9 + \sqrt{13} Z_\alpha = 14.93$ where $Z_\alpha \approx 1.645$. For the correlated case, $p^* = (v_6, v_4, v_7, v_5)$ has the minimum $F_{p^*}^{-1}(\alpha) = 9 + \sqrt{11} Z_\alpha = 14.46$. Note that the variance $\sigma^2_{W_{p^*}} = \sigma^2_{W_{(v_6,v_4)}} + \sigma^2_{W_{(v_4,v_7)}} + \sigma^2_{W_{(v_7,v_5)}} + 2\sigma_{(v_6,v_4),(v_4,v_7)} + 2\sigma_{(v_4,v_7),(v_7,v_5)} = 5 + 5 + 3 + (-2) \times 2 + 1 \times 2 = 11$.

### B. Tree Decomposition

Tree decomposition is widely used for path queries [26]–[30]. It allows us to focus on a small set of vertices, called a *separator*, which makes the source $s$ and the destination $t$ disconnected after the vertices in the separator are removed. In other words, any $s$-$t$ paths must traverse at least one vertex in the separator. Though it cannot handle the stochastic travel times, we can still use it to improve query efficiency.

*Definition 4 (Tree Decomposition [26], [31]):* The tree decomposition maps the network $G$ to a rooted tree, denoted by $T$. There is a bijection, denoted by $X$, from $V$ to the tree nodes. For each tree node $X(v)$, it is associated with a subset of $V$ that includes $v$. Abusing notations slightly, we also use $X(v)$ to denote this subset. Thus, for each tree node $X(v)$, $X(v) \subseteq V$ and $v \in X(v)$. The tree decomposition satisfies three conditions: 1) $\bigcup_{v \in V} X(v) = V$. 2) For each $e = (u, v)$, there exists one $X(v')$ such that $u, v \in X(v')$. 3) For each $v$, the set $\{X(v') : v \in X(v')\}$ forms a connected subtree.

A tree decomposition can be generated by Algorithm 6 in [26]. We only care about its useful property after building it.

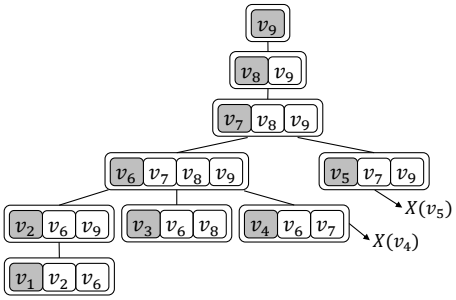



Fig. 2: An example tree decomposition for the network $G$

*Example 3:* In Figure 2, we depict a tree decomposition generated by Algorithm 6 in [26]. For each tree node $X(v)$, where $v$ is in the gray block, there is a subset of $V$. For example, the tree node $X(v_7) = \{v_7, v_8, v_9\}$.

It has a useful property about the separator defined below.

*Definition 5 (Separator):* The separator of two vertices $s$ and $t$, denoted by $H$, is a set of vertices such that $s$ and $t$ are disconnected after the vertices in the separator are removed.

*Lemma 1 (Lemma 4 in [32]):* Given any $s$ and $t$, suppose that there is no ancestor-descendant relationship between the two tree nodes $X(s)$ and $X(t)$. Let $X(l)$ denote the least common ancestor (LCA) of $X(s)$ and $X(t)$. $X(l)$ is a separator. Let $X(c_s)$ and $X(c_t)$ be the two children of the LCA $X(l)$ in the two branches containing $X(s)$ and $X(t)$, respectively. $X(c_s)\backslash\{c_s\}$ and $X(c_t)\backslash\{c_t\}$ are two separators. We only consider the one with a smaller size for efficiency.

*Example 4:* Back to Example 2, $X(v_7)$ is the LCA of $X(v_6)$ and $X(v_5)$, and $c_s = v_6$ and $c_t = v_5$. Hence, $X(v_6)\backslash\{v_6\} = \{v_7, v_8, v_9\}$ and $X(v_5)\backslash\{v_5\} = \{v_7, v_9\}$ are two separators. We focus on the latter one since it has a smaller size.

Though the NRP index uses the tree decomposition, its novelty lies in building the labels for independent and correlated variables and precomputing statistics for our proposed dominance conditions.

## III. Query Processing

### A. Overview

Given a separator $H$, one natural idea is to divide the problem into two subproblems that consider two subpaths, one from $s$ to a vertex $h \in H$ and the other from $h$ to $t$. This vertex $h$ is also called a "hoplink" since it links two "hops" (i.e., the two subpaths). Suppose that we have obtained the two "locally" optimal subpaths w.r.t. the two sub-problems. We may expect to find the optimal path $p^*$ by concatenating the two locally optimal subpaths for each $h \in H$ and comparing the $F_p^{-1}(\alpha)$ values of the $|H|$ concatenated $s$-$t$ paths. However, this intuition is wrong as shown in the following counterexample, which is essentially due to RSP's nonlinear objective.

*Example 5:* We still use Example 2. Suppose that we only consider the subgraph induced by $v_3, v_6, v_8, v_9$ for simplicity. We want to find the optimal $v_6$-$v_9$ path and use $v_8$ as a hoplink. The locally optimal $v_6$-$v_8$ path is $(v_6, v_3, v_8)$ since its $F^{-1}_{(v_6,v_3,v_8)}(\alpha) = 3 + \sqrt{1} \times 1.645 = 4.65 \leq F^{-1}_{(v_6,v_8)}(\alpha) = 2 + \sqrt{4} \times 1.645 = 5.30$ of the other path $(v_6, v_8)$. The locally optimal $v_8$-$v_9$ path is certainly $(v_8, v_9)$. However, after concatenating $(v_6, v_3, v_8)$ with $(v_8, v_9)$, we find that it is not optimal since $F^{-1}_{(v_6,v_3,v_8,v_9)}(\alpha) = 8 + \sqrt{6} \times 1.645 = 12.03 \geq F^{-1}_{(v_6,v_8,v_9)}(\alpha) = 7 + \sqrt{9} \times 1.645 = 11.93$.

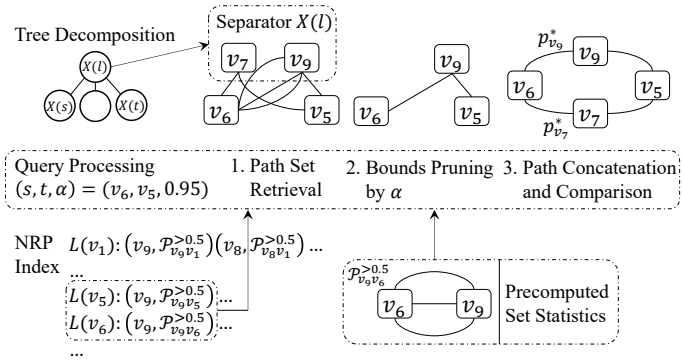



Fig. 3: Query processing overview

The main reason is that RSP's nonlinear objective makes the property of optimal substructure unavailable; that is, the locally optimal subpaths are not necessarily parts of the optimal path $p^*$. Instead of storing only one $s$-$h$ (or $h$-$t$) path, we store a set of *non-dominated* $s$-$h$ (or $h$-$t$) paths.

*Definition 6 (Path Dominance):* For any two paths $p_1$ and $p_2$, $p_1$ dominates $p_2$, denoted by $p_1 \prec p_2$, iff for any path $p_3$ and any $\alpha \in (0, 1)$, $F^{-1}_{p_1 \oplus p_3}(\alpha) < F^{-1}_{p_2 \oplus p_3}(\alpha)$.

Intuitively, no matter what $p_3$ is, $p_1 \oplus p_3$ always has a smaller $F_p^{-1}(\alpha)$ value than $p_2 \oplus p_3$, which suggests that $p_1$ is better than $p_2$. Let $\mu_i$ and $\sigma_i$ be $W_{p_i}$'s mean and standard deviation.

*Example 6:* For the independent case, $p_1 \prec p_2$ when $\mu_1 < \mu_2$ and $\sigma_1 = \sigma_2$. This is because $F^{-1}_{p_1 \oplus p_3}(\alpha) = \mu_1 + \mu_3 + Z_\alpha \sqrt{\sigma_1^2 + \sigma_3^2} < \mu_2 + \mu_3 + Z_\alpha \sqrt{\sigma_2^2 + \sigma_3^2} = F^{-1}_{p_2 \oplus p_3}(\alpha)$.

A *non-dominated* $u$-$v$ path is not dominated by any other $u$-$v$ paths. Let $\mathcal{P}_{uv}$ be the set of all non-dominated $u$-$v$ paths. Given a source $s$ and a destination $t$, the optimal path $p^*$ is in $\mathcal{P}_{st}$ since otherwise we can obtain a contradiction by setting $p_3 = (t)$ in Definition 6. We can also show that any $s$-$v$ (or $v$-$t$) subpath of $p^*$ is a non-dominated $s$-$v$ (or $v$-$t$) path by contradiction. Simply using $\mathcal{P}_{uv}$ is infeasible and prohibited since this relationship (Definition 6) rarely exists, which makes its size $|\mathcal{P}_{uv}|$ very large. The formal explanation is given in Section III-B. To make it feasible and efficient, we propose to use a *pruned path set* denoted by $\mathcal{P}^{>0.5}_{uv} \subseteq \mathcal{P}_{uv}$. Its details are given in Section III-B.

Our Non-dominated Reliable Path (NRP) index mainly stores a label $L(v)$ for each tree node $X(v)$ and $v \in V$. Each label $L(v)$ is $\{(u, \mathcal{P}^{>0.5}_{uv}) : X(u)$ is an ancestor of $X(v)\}$ whenever $X(u)$ is an ancestor of $X(v)$. Its construction is introduced in Section IV.

Algorithm 1 gives a query processing overview. In Line 1, we first find the least common ancestor (LCA) of $X(s)$ and $X(t)$. In Lines 2–4, if $X(s)$ and $X(t)$ have the ancestor-descendant relationship, we can directly compare all the paths in $\mathcal{P}^{>0.5}_{st}$ (since $\mathcal{P}^{>0.5}_{st}$ must exist in $L(s)$ or $L(t)$) and return

**Algorithm 1:** Query Processing Overview

**input :** The index and an RSP query with $s$, $t$, and $\alpha$
**output:** The optimal path $p^*$

1 $X(l) \leftarrow$ the LCA node of $X(s)$ and $X(t)$
2 **if** $l = s$ **then**
3  return the optimal path in $\mathcal{P}_{st}^{>0.5}$ stored in $L(t)$
4 **else if** $l = t$ **then**
5  return the optimal path in $\mathcal{P}_{st}^{>0.5}$ stored in $L(s)$
6 **else**
7  $Hoplinks \leftarrow \arg\min_{H \in \{H(s), H(t)\}} |H|$
8  **foreach** $h \in Hoplinks$ **do**
9  use $\alpha$ to prune $\mathcal{P}_{sh}^{>0.5}$ and $\mathcal{P}_{ht}^{>0.5}$ to get $\widehat{\mathcal{P}_{sh}}$ and $\widehat{\mathcal{P}_{ht}}$ (Algorithm 2)
10  find $p_h^* \in \{p_1 \oplus p_2 : p_1 \in \widehat{\mathcal{P}_{sh}}, p_2 \in \widehat{\mathcal{P}_{ht}}\}$ for hoplink $h$ with the minimum $F_p^{-1}(\alpha)$ value
11  return $p^* \leftarrow \arg\min_{p \in \{p_h^* : h \in Hoplinks\}} F_p^{-1}(\alpha)$

the optimal one with the minimum $F_p^{-1}(\alpha)$. Otherwise, we need to find a separator to be the set of hoplinks, denoted by $Hoplinks$. Let $H(s) = X(c_s)\backslash\{c_s\}$ and $H(t) = X(c_t)\backslash\{c_t\}$. Since $H(s)$ and $H(t)$ are two separators by Lemma 1, we set $Hoplinks$ to be the one with a smaller size in Line 7. Now consider each $h \in Hoplinks$ in Line 8. In Line 9, We first retrieve $\mathcal{P}_{sh}^{>0.5}$ and $\mathcal{P}_{ht}^{>0.5}$ from the NRP index and then use the confidence level $\alpha$ to prune them, which is introduced in Section III-B. In Line 10, we concatenate the paths from the two sets $\widehat{\mathcal{P}_{sh}}$ and $\widehat{\mathcal{P}_{ht}}$ and find the suboptimal path $p_h^*$ with the minimum $F_p^{-1}(\alpha)$. Finally, we return the optimal one $p^*$ among all $p_h^*$. These main steps are also shown in Figure 3.

*Example 7:* In Example 2, when $s = v_6$ and $t = v_5$, $X(l) = X(v_7)$, and $Hoplinks = H(v_5) = X(v_5)\backslash\{v_5\} = \{v_7, v_9\}$ since its size is smaller than $|H(v_6)| = 3$. For $h = v_9$, we find $p_{v_9}^*$ by concatenating the paths in $\mathcal{P}_{v_6 v_9}^{>0.5}$ and $\mathcal{P}_{v_9 v_5}^{>0.5}$ and similarly obtain $p_{v_7}^*$ for $h = v_7$. We obtain the final $p^*$ by comparing the $F_p^{-1}(\alpha)$ values of $p_{v_9}^*$ and $p_{v_7}^*$.

### B. Pruning Techniques

Directly applying Definition 6 may make the sets $\mathcal{P}_{sh}$ and $\mathcal{P}_{ht}$ very large. We first give two definitions by extending Definition 6 and then present pruning techniques based on the two definitions by separately discussing independent and correlated cases for clarity.

*1) Extended Path Dominance:*

To see that the case in Definition 6 rarely exists, we first show the following lemma.

*Lemma 2:* For two paths $p_1$ and $p_2$, suppose that either $\mu_1 \neq \mu_2$ or $\sigma_1 \neq \sigma_2$. For $\alpha \in (0,1)$, $F_{p_1}^{-1}(\alpha)$ and $F_{p_2}^{-1}(\alpha)$ have one intersection when $\sigma_1 \neq \sigma_2$ and none when $\sigma_1 = \sigma_2$.

*Proof:* Consider the function $g(\alpha) = F_{p_1}^{-1}(\alpha) - F_{p_2}^{-1}(\alpha)$ and its roots. Let $g(\alpha) = \mu_1 - \mu_2 + Z_\alpha(\sigma_1 - \sigma_2) = 0$. If $\sigma_1 = \sigma_2$, it has no root because $\mu_1 \neq \mu_2$. Otherwise, we get $Z_\alpha = \frac{\mu_2 - \mu_1}{\sigma_1 - \sigma_2}$ and only one root $\alpha' = \Phi(\frac{\mu_2 - \mu_1}{\sigma_1 - \sigma_2})$ (because $Z_\alpha$'s inverse function exists), where $\Phi(x)$ is the CDF of the standard normal distribution $\mathcal{N}(0,1)$. ■

Now consider the two paths $p_1 \oplus p_3$ and $p_2 \oplus p_3$. The fact that $F_{p_1 \oplus p_3}^{-1}(\alpha) < F_{p_2 \oplus p_3}^{-1}(\alpha)$ holds for any $\alpha \in (0,1)$ indicates that the two functions should have no intersection and the standard deviations of $W_{p_1 \oplus p_3}$ and $W_{p_2 \oplus p_3}$ are equal, which rarely happens since $p_1$ and $p_2$ can be different. To relax this strict condition, we consider the following two definitions.

*Definition 7 (Path Dominance on an Interval of $\alpha$):* For any two paths $p_1$ and $p_2$, $p_1 \prec p_2$ on an interval of $\alpha$, iff for any path $p_3$, $F_{p_1 \oplus p_3}^{-1}(\alpha) < F_{p_2 \oplus p_3}^{-1}(\alpha)$ for any $\alpha$ in this interval.

Based on Definition 7, we separately use the sets of all non-dominated paths w.r.t. different intervals of $\alpha$ (i.e., $\mathcal{P}_{uv}^{>0.5}$ for $\alpha > 0.5$), which are expected to be smaller.

Furthermore, in Line 9 of Algorithm 1, when we prune the paths in $\mathcal{P}_{sh}^{>0.5}$, we actually concatenate them with those in $\mathcal{P}_{ht}^{>0.5}$ (instead of any path $p_3$ as in Definition 7), and vice versa. Thus, we propose the following looser condition.

*Definition 8 (Path Dominance w.r.t. an Interval of $\alpha$ and a Path Set $P$):* For any two paths $p_1$ and $p_2$, $p_1 \prec p_2$ w.r.t. an interval of $\alpha$ and a path set $P$, iff for any path $p_3 \in P$, $F_{p_1 \oplus p_3}^{-1}(\alpha) < F_{p_2 \oplus p_3}^{-1}(\alpha)$ for any $\alpha$ in the interval.

*Theorem 1:* Algorithm 1 correctly answers the RSP.

*Proof:* We only need to show that $p^*$ is one of the concatenated paths in $\{p_1 \oplus p_2 : p_1 \in \widehat{\mathcal{P}_{sh}}, p_2 \in \widehat{\mathcal{P}_{ht}}\}$ for some $h$. By the definition of the separator, $p^*$ must visit one $h \in H$ and can be rewritten as $p^* = p_1 \oplus p_2$ where $p_1$ and $p_2$ are the $s$-$h$ and $h$-$t$ subpaths, respectively. We know that $p_1$ and $p_2$ must exist in $\widehat{\mathcal{P}_{sh}}$ and $\widehat{\mathcal{P}_{ht}}$, respectively. ■

Algorithm 1's time complexity is $\mathcal{O}(|Hoplink||\widehat{\mathcal{P}_{sh}}||\widehat{\mathcal{P}_{ht}}|)$. The main time-consuming part lies in Line 10 of finding the path $p_h^*$, which takes $\mathcal{O}(|\widehat{\mathcal{P}_{sh}}||\widehat{\mathcal{P}_{ht}}|)$ time for path concatenation. Line 9 takes linear time $\mathcal{O}(|\widehat{\mathcal{P}_{sh}}| + |\widehat{\mathcal{P}_{ht}}|)$ to prune the paths. Since there are $|Hoplinks|$ iterations, the time complexity follows straightaway.

*2) Independent Variables:*

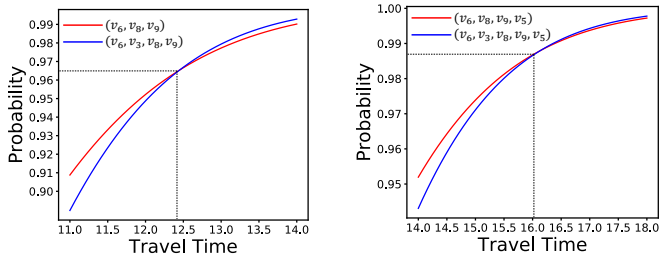
**Separate Cases by M-V Dominance.** We start with an intuitive dominance based on Definition 7.

*Proposition 1 (M-V Dominance [18]):* For two paths $p_1$ and $p_2$, $p_1 \prec p_2$ on $\alpha \in (0.5, 1)$ if $\mu_1 \leq \mu_2$ and $\sigma_1 < \sigma_2$. Similarly, $p_1 \prec p_2$ on $\alpha \in (0, 0.5)$ if $\mu_1 \leq \mu_2$ and $\sigma_1 > \sigma_2$.

*Proof:* $F_{p_1 \oplus p_3}^{-1}(\alpha) = \mu_1 + \mu_3 + Z_\alpha\sqrt{\sigma_1^2 + \sigma_3^2}$ and $F_{p_2 \oplus p_3}^{-1}(\alpha) = \mu_2 + \mu_3 + Z_\alpha\sqrt{\sigma_2^2 + \sigma_3^2}$. $F_{p_1 \oplus p_3}^{-1}(\alpha) < F_{p_2 \oplus p_3}^{-1}(\alpha)$ holds on the two intervals because $Z_\alpha > 0$ for $\alpha \in (0.5, 1)$ and $Z_\alpha < 0$ for $\alpha \in (0, 0.5)$. ■

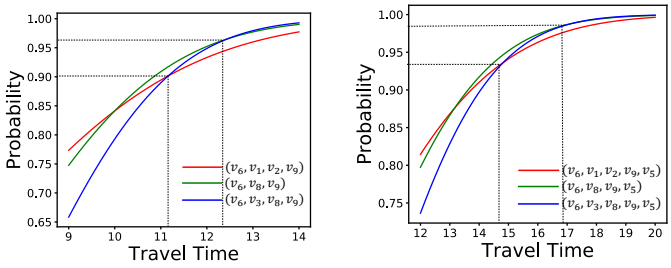
Note that we omit the case where $\sigma_1 = \sigma_2$ since the dominance holds on $(0,1)$ and we can easily prune such dominated paths first. Based on the above observation, we would separately consider three cases: 1) $\alpha = 0.5$, 2) $\alpha \in (0.5, 1)$, and 3) $\alpha \in (0, 0.5)$. For each label $L(v)$, for each $X(v)$'s ancestor $X(u)$, instead of storing the set of all non-dominated paths $\mathcal{P}_{uv}$ w.r.t. $\alpha \in (0,1)$, our index preprocesses the three sets of non-dominated paths w.r.t. $\alpha$'s three intervals, denoted by $\mathcal{P}_{uv}^{0.5}$, $\mathcal{P}_{uv}^{>0.5}$, and $\mathcal{P}_{uv}^{<0.5}$, respectively.

For the first case where $\alpha = 0.5$, $Z_\alpha = 0$ and $F_p^{-1}(\alpha) = \mu_p$. By Definition 7, $p_1 \prec p_2$ iff $\mu_1 + \mu_3 < \mu_2 + \mu_3$ (i.e. $\mu_1 < \mu_2$).




(a) Before concatenating $(v_9, v_5)$ (b) After concatenating $(v_9, v_5)$

Fig. 4: Dominance w.r.t. the path set $P = \{(v_9, v_5)\}$




(a) Before concatenating $(v_9, v_5)$ (b) After concatenating $(v_9, v_5)$

Fig. 5: Upper Bound Maximizer of $(v_6, v_3, v_8, v_9)$

We only need to store only one $u$-$v$ path with the minimum mean value in $\mathcal{P}_{uv}^{0.5}$ (where ties are broken arbitrarily). This special case is equivalent to finding the shortest path on a special network where edges' travel times are deterministic constants equal to the mean values.

For the second case where $\alpha > 0.5$, $\mathcal{P}_{uv}^{>0.5}$ includes all the non-dominated paths w.r.t. $\alpha \in (0.5, 1)$.

*Definition 9 (Pruned path set $\mathcal{P}_{uv}^{>0.5}$):* For each $p \in \mathcal{P}_{uv}^{>0.5}$, it is not dominated by any other $u$-$v$ path on $(0.5, 1)$. If w.l.o.g. $\mathcal{P}_{uv}^{>0.5} = \{p_1, p_2, \ldots, p_k\}$ where $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_k$, $\sigma_1 > \sigma_2 > \ldots > \sigma_k$.

*Example 8:* In the example, $\mathcal{P}_{v_6v_9}^{>0.5} = \{p_1, p_2, p_3\}$ where $p_1 = (v_6, v_1, v_2, v_9), p_2 = (v_6, v_8, v_9), p_3 = (v_6, v_3, v_8, v_9)$ with their means $\mu_1 = 6$, $\mu_2 = 7$, $\mu_3 = 8$ and standard deviations $\sigma_1 = 4$, $\sigma_2 = 3$, $\sigma_3 = \sqrt{6}$, respectively.

We do not use split points other than 0.5 because it is hard to check the dominance (Definition 7). For example, if we use $\mathcal{P}_{uv}^{>0.1587}$, where $Z_\alpha = -1$ with $\alpha \approx 0.1587$, there are no concise conditions (i.e., $\mu_1 \leq \mu_2$ and $\sigma_1 < \sigma_2$ in Proposition 1) to ensure that $\mu_1 + \mu_3 + Z_\alpha\sqrt{\sigma_1^2 + \sigma_3^2} < \mu_2 + \mu_3 + Z_\alpha\sqrt{\sigma_2^2 + \sigma_3^2}$ for any $Z_\alpha > -1$ since $Z_\alpha$ can be positive or negative. Furthermore, we will discuss such more refined dominance on flexible intervals later by Propositions 2 and 3.

We omit the third case since similar results can be done by symmetry, and users usually set the confidence level $\alpha$ to be greater than 0.5 to avoid a high risk in practice. In the following, we only discuss the second case for simplicity. NRP builds $\mathcal{P}_{uv}^{>0.5}$ during index construction (explained in Section IV). We next discuss our further pruning techniques used in Line 9 of Algorithm 1.

**Intersection Dominance.** When $\alpha > 0.5$, for any two paths $p_1, p_2 \in \mathcal{P}_{sh}^{>0.5}$, we know that if $\mu_1 \leq \mu_2$, we have $\sigma_1 > \sigma_2$. Though the intersection is above 0.5 and the fact that $p_1 \prec p_2$ on $\alpha \in (0.5, 1)$ does not hold, we can observe that the red line is still on the left side of the blue line from 0.5 to the y-value of the intersection. Besides, since the paths in $\mathcal{P}_{sh}^{>0.5}$ could only be concatenated with those in $\mathcal{P}_{ht}^{>0.5}$, we hope to find dominance w.r.t. a specific interval of $\alpha$ and $\mathcal{P}_{ht}^{>0.5}$ based on Definition 8. Formally, we propose the *intersection dominance*. When we prune $\mathcal{P}_{sh}^{>0.5}$, the dominance is w.r.t. the path set $\mathcal{P}_{ht}^{>0.5}$, and vice versa. We only need to find the interval of $\alpha$ to make the condition hold. Let $\sigma_{\min}(P) = \min_{p \in P} \sigma_p$ and $\sigma_{\max}(P) = \max_{p \in P} \sigma_p$. Suppose that NRP has precomputed $\sigma_{\min}(\mathcal{P}_{uv}^{>0.5})$ and $\sigma_{\max}(\mathcal{P}_{uv}^{>0.5})$ for each $X(v)$'s ancestor $X(u)$.

*Proposition 2 (Intersection Dominance):* For paths $p_1$ and $p_2$, $p_1 \prec p_2$ on $\alpha \in (0.5, \Phi(\frac{\mu_2 - \mu_1}{\sqrt{\sigma_1^2 + \sigma_{\min}(P)^2} - \sqrt{\sigma_2^2 + \sigma_{\min}(P)^2}}))$ w.r.t. path set $P$ when $\mu_1 \leq \mu_2$ and $\sigma_1 > \sigma_2$.

*Proof:* We first show that $F^{-1}_{p_1 \oplus p_3}(\alpha) < F^{-1}_{p_2 \oplus p_3}(\alpha)$ for $\alpha \in (0.5, \Phi(\frac{\mu_2 - \mu_1}{\sqrt{\sigma_1^2 + \sigma_3^2} - \sqrt{\sigma_2^2 + \sigma_3^2}}))$ and any $p_3$ with any $\sigma_3$. By Lemma 2, $F^{-1}_{p_1 \oplus p_3}(\alpha)$ and $F^{-1}_{p_2 \oplus p_3}(\alpha)$ have only one intersection since the two standard deviations (i.e., $\sqrt{\sigma_1^2 + \sigma_3^2}$ and $\sqrt{\sigma_2^2 + \sigma_3^2}$) are not equal. Consider the function $g(\alpha) = F^{-1}_{p_1 \oplus p_3}(\alpha) - F^{-1}_{p_2 \oplus p_3}(\alpha)$. The only one root of $g(\alpha)$ is $\alpha' = \Phi(\frac{\mu_2 - \mu_1}{\sqrt{\sigma_1^2 + \sigma_3^2} - \sqrt{\sigma_2^2 + \sigma_3^2}})$, similar to the one in Lemma 2, and $g(\alpha) < 0$ on $(0.5, \alpha')$ and $g(\alpha) > 0$ on $(\alpha', 1)$.

Since we need to find an interval of $\alpha$ such that $g(\alpha) < 0$ holds for any path $p_3 \in P$ on this interval, we need to take the minimum $\alpha'$ for each $p_3$. Let $A(x) = \sqrt{\sigma_1^2 + x^2} - \sqrt{\sigma_2^2 + x^2}$. Since $A(x)$ is decreasing on $x \geq 0$ (by $A'(x) < 0$), $A(\sigma_3) \leq A(\sigma_{\min}(P))$ (by $\sigma_3 \geq \sigma_{\min}(P)$). Since $\Phi(x)$ is a monotonically increasing function, $\alpha' \geq \Phi(\frac{\mu_2 - \mu_1}{A(\sigma_{\min}(P))})$. ■

To get some intuition, we may imagine that the y-value of the intersection will increase when we concatenate any $p_3$ with any $\sigma_3$. We need to use the minimum y-value of intersections since the dominance has to hold for any paths in the set $P$.

*Example 9:* Back to the previous example, for $h = v_9$, we concatenate the paths in $\mathcal{P}_{v_6v_9}^{>0.5}$ and $\mathcal{P}_{v_9v_5}^{>0.5} = \{(v_9, v_5)\}$. Now consider two paths $(v_6, v_8, v_9)$ and $(v_6, v_3, v_8, v_9)$ in $\mathcal{P}_{v_6v_9}^{>0.5}$, shown by red and blue lines, respectively, in Figure 4. Before concatenating $(v_9, v_5)$, we know that the red line is on the left side of the blue line from 0.5 to the y-value of the intersection in Figure 4a. Since we will concatenate the two paths with $(v_9, v_5)$, the y-value of the intersection is $\Phi(\frac{10-9}{\sqrt{9+4} - \sqrt{6+4}}) = 0.988$ in Figure 4b, larger than that in Figure 4a. We know that $(v_6, v_8, v_9) \prec (v_6, v_3, v_8, v_9)$ on $(0.5, 0.988)$ w.r.t. $\mathcal{P}_{v_9v_5}^{>0.5}$.

For each path $p$, any other paths in $\mathcal{P}_{sh}^{>0.5}$ with smaller means can dominate $p$ with different intervals w.r.t. a path set $\mathcal{P}_{ht}^{>0.5}$. The one that gives the largest upper bound could be applied to more $\alpha$ values. Our NRP index stores this *upper bound maximizer* for each path $p$, denoted by $p_{\max}$. Note that $p_{\max}$ does not change.

*Definition 10 (Upper Bound Maximizer):* We consider the path set $\overline{S_p} = \{p' \in \mathcal{P}_{uv}^{>0.5} : \mu_{p'} < \mu_p\}$ consisting of paths with their means smaller than $\mu_p$. For each path $p \in \mathcal{P}_{uv}^{>0.5}$, we define its upper bound maximizer $p_{\max} =$

$\arg\max_{p' \in \overline{S_p}} \Phi(\frac{\mu_p - \mu_{p'}}{\sqrt{\sigma_{p'}^2 + \sigma_{\min}(P)^2} - \sqrt{\sigma_p^2 + \sigma_{\min}(P)^2}})$ w.r.t. a path set $P$. It can be found that $p_{\max} = \arg\max_{p' \in \overline{S_p}} \Phi(\frac{\mu_p - \mu_{p'}}{\sigma_{p'} - \sigma_p})$.

*Example 10:* Consider $\mathcal{P}_{v_6 v_9}^{>0.5}$ in the running example. For the path $(v_6, v_3, v_8, v_9)$, $\overline{S_p} = \{(v_6, v_1, v_2, v_9), (v_6, v_8, v_9)\}$ since their means are smaller. In Figure 5, $(v_6, v_1, v_2, v_9)$, $(v_6, v_8, v_9)$, and $(v_6, v_3, v_8, v_9)$ are represented by red, green, and blue lines, respectively. For $(v_6, v_3, v_8, v_9)$, the other two paths can dominate it on different intervals w.r.t. $\mathcal{P}_{v_9 v_5}^{>0.5}$ as shown in Figure 4b. Its upper bound maximizer is $(v_6, v_8, v_9)$ since $\Phi(\frac{8-7}{3-\sqrt{6}}) > \Phi(\frac{8-6}{4-\sqrt{6}})$. Note that its maximizer does not change as in Figure 4b where we concatenate $(v_9, v_5)$.

**Reverse Intersection Dominance.** It can be also found in Figure 4 that the blue line is on the left side of the red one from the y-value of the intersection to 1. Similarly, we can find an interval based on Definition 8 and propose the *reverse intersection dominance* (since now the paths with larger means can dominate the others).

*Proposition 3 (Reverse Intersection Dominance):* For two paths $p_1$ and $p_2$ and a path set $P$, $p_2 \prec p_1$ on $\alpha \in (\Phi(\frac{\mu_2 - \mu_1}{\sqrt{\sigma_1^2 + \sigma_{\max}(P)^2} - \sqrt{\sigma_2^2 + \sigma_{\max}(P)^2}}), 1)$ w.r.t. $P$ when $\mu_1 \leq \mu_2$ and $\sigma_1 > \sigma_2$, where $\sigma_{\max}(P) = \max_{p \in P} \sigma_p$.

*Proof:* Similar to the previous proof, we define $\alpha'$ and $A(x)$. We know that $g(\alpha) > 0$ on $(\alpha', 1)$ for any path $p_3$. We need to take the maximum $\alpha'$ w.r.t. all $p_3 \in P$. We obtain $\alpha' \leq \Phi(\frac{\mu_2 - \mu_1}{A(\sigma_{\max}(P))})$ ∎

Similarly, we want to find the lowest (or best) lower bound with the *lower bound minimizer*, denoted by $p_{min}$.

*Definition 11 (Lower Bound Minimizer):* Let $\underline{S_p} = \{p' \in \mathcal{P}_{uv}^{>0.5} : \mu_{p'} > \mu_p\}$ be the set of paths with their means greater than $\mu_p$. For each path $p \in \mathcal{P}_{uv}^{>0.5}$, its lower bound minimizer $p_{\min} = \arg\min_{p' \in \underline{S_p}} \Phi(\frac{\mu_{p'} - \mu_p}{\sigma_p - \sigma_{p'}})$.

*Example 11:* For the path $(v_6, v_1, v_2, v_9)$, we can obtain its $\underline{S_p} = \{(v_6, v_8, v_9), (v_6, v_3, v_8, v_9)\}$. Its lower bound minimizer is $(v_6, v_8, v_9)$ since $\Phi(\frac{7-6}{4-3}) < \Phi(\frac{8-6}{4-\sqrt{6}})$.

Note that the upper bound maximizer and the lower bound minimizer could be preprocessed in the NRP index since they are fixed whatever the path set $P$ is.

Let $B_p(p_m, x) = \Phi(\frac{\mu_{p_m} - \mu_p}{\sqrt{\sigma_p^2 + x^2} - \sqrt{\sigma_{p_m}^2 + x^2}})$ be the function to compute the two bounds, where $p_m$ is the upper bound maximizer or the lower bound minimizer and $x$ is $\sigma_{\min}(P)$ or $\sigma_{\max}(P)$, respectively. If $\alpha$ lies in the interval between the two bounds, we can prune the path $p$. Algorithm 2 illustrates the pruning procedure. We only discuss $\alpha > 0.5$ for simplicity. In Lines 1–3, for each $p \in \mathcal{P}_{sh}^{>0.5}$, it can be pruned if $\alpha$ is smaller than the upper bound or greater than the lower bound, computed by using the upper bound maximizer or the lower bound minimizer in $B_p(p_m, x)$, respectively. In Lines 4–6, we similarly prune $\mathcal{P}_{ht}^{>0.5}$.

*Example 12:* For $\mathcal{P}_{v_6 v_9}^{>0.5}$, $\sigma_{\min}(\mathcal{P}_{v_9 v_5}^{>0.5}) = \sigma_{\max}(\mathcal{P}_{v_9 v_5}^{>0.5}) = 2$. For $(v_6, v_1, v_2, v_9)$, it does not have $p_{\max}$ but $p_{\min} = (v_6, v_8, v_9)$, we can prune it since $\alpha = 0.95 > B_p(p_{\min}, \sigma_{\max}(\mathcal{P}_{ht}^{>0.5})) = \Phi(\frac{7-6}{\sqrt{16+4} - \sqrt{9+4}}) = 0.88$. For $(v_6, v_8, v_9)$, its $p_{\max} = (v_6, v_1, v_2, v_9)$ and $p_{\min} = (v_6, v_3, v_8, v_9)$. The two bounds for $p_{\max}$ and $p_{\min}$ are 0.88 and 0.99, respectively. We need to include $(v_6, v_8, v_9)$ in $\widehat{\mathcal{P}_{v_6 v_9}}$ since $\alpha \geq 0.88$ and $\alpha \leq 0.99$. For $(v_6, v_3, v_8, v_9)$, it does not have $p_{\min}$ but $p_{\max} = (v_6, v_3, v_8, v_9)$. It can also be pruned since $\alpha = 0.95 < B_p(p_{\max}, \sigma_{\min}(\mathcal{P}_{ht}^{>0.5})) = 0.99$. For $(v_9, v_5) \in \mathcal{P}_{v_9 v_5}^{>0.5}$, it does not have the maximizer and the minimizer. Finally, $\widehat{\mathcal{P}_{v_6 v_9}} = \{(v_6, v_8, v_9)\}$ and $\widehat{\mathcal{P}_{v_9 v_5}} = \{(v_9, v_5)\}$.

**Algorithm 2:** Pruning Dominated Paths

**input :** $\mathcal{P}_{sh}^{>0.5}$, $\mathcal{P}_{ht}^{>0.5}$, and $\alpha$
**output:** $\widehat{\mathcal{P}_{sh}}$ and $\widehat{\mathcal{P}_{ht}}$
1 **foreach** $p \in \mathcal{P}_{sh}^{>0.5}$ *s.t.* $B_p(p_{\max}, \sigma_{\min}(\mathcal{P}_{ht}^{>0.5})) \leq \alpha \leq B_p(p_{\min}, \sigma_{\max}(\mathcal{P}_{ht}^{>0.5}))$ **do**
2  $\widehat{\mathcal{P}_{sh}} \leftarrow \widehat{\mathcal{P}_{sh}} \cup \{p\}$
3 **foreach** $p \in \mathcal{P}_{ht}^{>0.5}$ *s.t.* $B_p(p_{\max}, \sigma_{\min}(\mathcal{P}_{sh}^{>0.5})) \leq \alpha \leq B_p(p_{\min}, \sigma_{\max}(\mathcal{P}_{sh}^{>0.5}))$ **do**
4  $\widehat{\mathcal{P}_{ht}} \leftarrow \widehat{\mathcal{P}_{ht}} \cup \{p\}$

It can be easily seen that the time complexity of Algorithm 2 is $\mathcal{O}(|\mathcal{P}_{sh}^{>0.5}| + |\mathcal{P}_{ht}^{>0.5}|)$ since there are only two for-loops.

*3) Correlated Variables:*

By extending the M-V dominance for independent variables, we can similarly consider the three separate cases and focus on the one where $\alpha \in (0.5, 1)$. We first propose the correlated M-V dominance for $\mathcal{P}_{uv}^{>0.5}$ and then a dominance relationship based on the lower and upper bounds of the variance.

**Separate Cases by Correlated M-V Dominance.** For each path $p = (e_1, e_2, \ldots, e_{|p|})$, $W_p = \sum_{i=1}^{k} W_{e_i}$ is a normal variable with its mean $\sum_{i=1}^{k} \mu_{e_i}$ and variance $\sum_{i=1}^{k} \sigma_{e_i}^2 + 2\sum_{i<j} \sigma_{e_i, e_j}$, where $\sigma_{e_i, e_j}$ is the covariance between $W_{e_i}$ and $W_{e_j}$. Let $\sigma_{p_i, p_j}$ be the covariance between $W_{p_i}$ and $W_{p_j}$.

To extend Proposition 1, we want to make $F_{p_1 \oplus p_3}^{-1}(\alpha) < F_{p_2 \oplus p_3}^{-1}(\alpha)$ hold for any path $p_3$, which further requires that $\mu_1 \leq \mu_2$ and $\sigma_1^2 + 2\sigma_{p_1, p_3} + \sigma_3^2 < \sigma_2^2 + 2\sigma_{p_2, p_3} + \sigma_3^2$ as in Proposition 1. However, it is costly to check the second condition since there are infinite paths to be $p_3$. In practice, the correlation between two edges is often weak when they are far from each other [7], [8], [33]. Let the *$K$-hop neighborhood* of $v$, denoted by $Nei_K(v)$, be the set of all simple paths that start from $v$ and have at most $K$ edges. For each $u$-$v$ path $p$, assume that there are only correlations between edges in $p$ and those of the paths in $Nei_K(u) \cup Nei_K(v)$ for some $K$. We then use $Nei_K(u) \cup Nei_K(v)$ to check the covariance.

*Proposition 4 (Correlated M-V Dominance):* For two $u$-$v$ paths $p_1$ and $p_2$, $p_1 \prec p_2$ on $\alpha \in (0.5, 1)$ if 1) $\mu_1 \leq \mu_2$ and 2) for $p \in Nei_K(u) \cup Nei_K(v)$, $\sigma_1^2 + 2\sigma_{p_1, p} < \sigma_2^2 + 2\sigma_{p_2, p}$.

*Proof:* We need to show that for any path $p_3$, $F_{p_1 \oplus p_3}^{-1}(\alpha) = \mu_1 + \mu_3 + Z_\alpha \sqrt{\sigma_1^2 + 2\sigma_{p_1, p_3} + \sigma_3^2} < \mu_2 + \mu_3 + Z_\alpha \sqrt{\sigma_2^2 + 2\sigma_{p_2, p_3} + \sigma_3^2} = F_{p_2 \oplus p_3}^{-1}(\alpha)$. The path $p_3$ can be always divided into two subpaths, one in $Nei_K(u) \cup Nei_K(v)$ (i.e., some $p$) and the remaining part. Since $\sigma_{p_1, p_3} = \sigma_{p_1, p}$, $\sigma_{p_2, p_3} = \sigma_{p_2, p}$, $\mu_1 \leq \mu_2$, and $Z_\alpha > 0$ on $\alpha \in (0.5, 1)$, we have $F_{p_1 \oplus p_3}^{-1}(\alpha) < F_{p_2 \oplus p_3}^{-1}(\alpha)$. ∎

Our NRP index would then store $\mathcal{P}_{uv}^{>0.5} = \{p_1, p_2, \ldots, p_k\}$ and w.l.o.g. $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_k$. Then, for any two paths $p_i$

and $p_j$ with $i < j$, $\sigma_i^2 + 2\sigma_{p_i,p} > \sigma_j^2 + 2\sigma_{p_j,p}$ for one path $p \in Nei_K(u) \cup Nei_K(v)$, since otherwise it would contradict the condition in Proposition 4. Note that if the edges in $p$ is independent of the edges in $Nei_K(u)$ (or $Nei_K(v)$), we do not need to check $Nei_K(u)$ (or $Nei_K(v)$). If both $Nei_K(u)$ and $Nei_K(v)$ can be ignored, we can directly use the techniques for independent variables.

*Example 13:* Consider the graph in Example 2 where the two covariances $\sigma_{(v_6,v_4),(v_4,v_7)} = -2$ and $\sigma_{(v_4,v_7),(v_7,v_5)} = 1$ exist between adjacent edges, which further means that $K = 1$ is sufficient. Consider two $v_6$-$v_7$ paths $p_1 = (v_6, v_4, v_7)$ and $p_2 = (v_6, v_8, v_7)$. We can derive that $p_1 \prec p_2$ because $\mu_1 = 6 < 13 = \mu_2$ and for $(v_7, v_5) \in Nei_1(v_7)$, $\sigma_1^2 + 2\sigma_{p_1,p_3} = (5 + 5 - 2 \times 2) + 2 \times 1 = 8 < \sigma_2^2 + 2\sigma_{p_2,p_3} = 12$.

**Correlated Bound Dominance.** To prune $\mathcal{P}_{sh}^{>0.5}$ further, we could also apply Definition 8 to find a dominance relationship since the paths in $\mathcal{P}_{sh}^{>0.5}$ will be concatenated with those in $\mathcal{P}_{ht}^{>0.5}$. Using the maximum variance $\sigma_{\max}(P) = \max_{p \in P} \sigma_p$, we propose the following *correlated bound dominance*.

*Proposition 5 (Correlated Bound Dominance):* For two paths $p_1$ and $p_2$, $p_1 \prec p_2$ for some $\alpha$ w.r.t. the path set $P$ if $\mu_1 + Z_\alpha(\sigma_1 + \sigma_{\max}(P)) < \mu_2$.

*Proof:* For any path $p \in P$, we have $F^{-1}_{p_1 \oplus p}(\alpha) = \mu_1 + \mu_p + Z_\alpha \sqrt{\sigma_1^2 + 2\sigma_{p_1,p} + \sigma_p^2} \le \mu_1 + \mu_p + Z_\alpha(\sigma_1 + \sigma_p) \le \mu_1 + \mu_p + Z_\alpha(\sigma_1 + \sigma_{\max}(P)) < \mu_2 \le \mu_2 + \mu_p + Z_\alpha \sqrt{\sigma_2^2 + 2\sigma_{p_2,p} + \sigma_p^2} = F^{-1}_{p_2 \oplus p}(\alpha)$, where the first inequality is because $|\sigma_{p_1,p}| \le \sigma_1 \sigma_p$. ∎

*Example 14:* Back to the previous example, we can derive that $p_1 = (v_6, v_4, v_7) \prec p_2 = (v_6, v_8, v_7)$ w.r.t. $\mathcal{P}_{v_7 v_5}^{>0.5} = \{(v_7, v_5)\}$ since $\mu_1 + Z_\alpha(\sigma_1 + \sigma_{\max}(\mathcal{P}_{v_7 v_5}^{>0.5})) = 6 + 1.645 \times (\sqrt{6} + \sqrt{3}) = 12.88 < 13 = \mu_2$.

The whole procedure takes $\mathcal{O}(|\mathcal{P}_{sh}^{>0.5}| + |\mathcal{P}_{ht}^{>0.5}|)$ time.

## IV. Index Construction

The main task is to preprocess the sets $\mathcal{P}_{uv}^{>0.5}$ of non-dominated reliable paths. We basically follow the structure of the tree decomposition to build the sets recursively on the tree. To accelerate the procedure, we can apply Propositions 1 and 4 for independent and correlated variables.

The tree decomposition is built by *contracting* each vertex following a given vertex order $\pi$, where $\pi(v)$ denotes $v$'s position in the order. When contracting $v$, we are removing it and its incident edges and assigning $X(v)$ as the set including $v$ and its current neighbors. When $v$'s two neighbors $u$ and $w$ are removed, we need to preserve the information about non-dominated paths (which means that $\mathcal{P}_{uv}^{>0.5}$ for any $u$ and $v$ remains unchanged on the new graph after the removal of $v$) by adding a new edge $(u, w)$ if $(u, w)$ does not exist and associating it with an *edge-driven* path set $P_{(u,w)}$ (for each $e \in E$, $P_e$ initially contains one single path corresponding to $e$). $P_{(u,w)}$ is updated by $RF(P_{(u,w)} \cup (P_{(u,v)} \oplus P_{(v,w)}))$, where the *refining* operation $RF(P)$ for a path set $P$ keeps only the non-dominated paths in $P$. After generating all edge-driven path sets, we can then obtain $\mathcal{P}_{uv}^{>0.5}$ by $RF(\bigcup_{w \in X(v) \setminus \{v\}} P_{(v,w)} \oplus \mathcal{P}_{uw}^{>0.5})$ (where $\mathcal{P}_{uw}^{>0.5}$ has been computed in previous labels) in the tree recursively since $X(v) \setminus \{v\}$ includes $v$'s neighbors during the contraction of $v$.

**Algorithm 3:** Index Construction

**input :** A road network $G$
**output:** The tree decomposition $T$ and the labels $L$

1 $P_{(u,v)} \leftarrow \{(u, v)\}$ for $(u, v) \in E$
2 $T, \pi \leftarrow$ Algorithm 6 in [26] on $G$
3 **foreach** *tree node* $X(v)$ *in the contraction order* $\pi$ **do**
4  **foreach** $(u, w)$ *where* $u \neq w$ *and* $u, w \in X(v)$ **do**
5  $P_{(u,w)} \leftarrow RF(P_{(u,w)} \cup (P_{(u,v)} \oplus P_{(v,w)}))$
6 **foreach** *tree node* $X(v)$ *in a top-down manner* **do**
7  **foreach** *ancestor* $X(u)$ *of* $X(v)$ **do**
8  **foreach** $w \in X(v) \setminus \{v\}$ **do**
9  $\mathcal{P}_{uv}^{>0.5} \leftarrow RF(\mathcal{P}_{uv}^{>0.5} \cup RF(P_{(v,w)} \oplus \mathcal{P}_{uw}^{>0.5}))$
10  $L(v) \leftarrow L(v) \cup (u, \mathcal{P}_{uv}^{>0.5})$

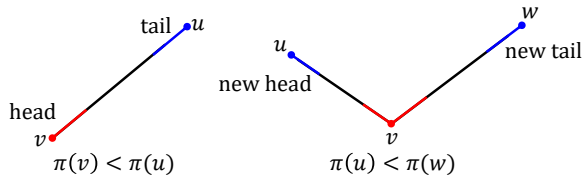



Fig. 6: The head and tail in the path concatenation

Algorithm 3 summarizes the whole procedure. In Lines 1–2, we initialize $P_e$ for each $e \in E$ and obtain the tree decomposition. In Lines 3–5, we build the edge-driven path sets following the contraction order. In Lines 6–10, we build $\mathcal{P}_{uv}^{>0.5}$ in a top-down manner.

Since Lines 5 and 9 use the path concatenation $P_1 \oplus P_2$ and the refining operation $RF(P)$, we show next how to do them efficiently for independent and correlated variables.

**Independent Variables.** Let $(\mu_p, \sigma_p^2)$ represent each path $p$. The path concatenation $p_1 \oplus p_2$ of two paths $p_1$ and $p_2$ has its $(\mu_{p_1} + \mu_{p_2}, \sigma_{p_1}^2 + \sigma_{p_2}^2)$. For the refining operation, given a path set $P = \{p_1, p_2, \ldots, p_k\}$, it can be easily done by first sorting all the paths in $P$ in the increasing order of their $\mu$ values (i.e., $\mu_1 \le \mu_2 \le \ldots \le \mu_k$) and then sequentially removing those with $\sigma$ values larger than their previous ones (to make $\sigma_1 > \sigma_2 > \ldots > \sigma_k$ hold), as in Proposition 1. After obtaining $\mathcal{P}_{uv}^{>0.5}$ in each label, we can easily obtain the upper bound maximizer, the lower bound minimizer, $\sigma_{\min}(\mathcal{P}_{uv}^{>0.5})$, and $\sigma_{\max}(\mathcal{P}_{uv}^{>0.5})$ by their definitions.

In practice, $\alpha \le 0.999$ can satisfy most user requirements. After sorting paths in the increasing order of $\mu$ values, we can refine them by maintaining $\mu_1 + 3.1\sigma_1 > \mu_2 + 3.1\sigma_2 > \ldots > \mu_k + 3.1\sigma_k$, which uses $F_p^{-1}(\alpha)$ when $\alpha = 0.999$ and $Z_\alpha = 3.1$. This can be easily proved by using the monotonicity of $g(\alpha)$ in Lemma 2. Note that $\sigma_1 > \sigma_2 > \ldots > \sigma_k$ corresponds to the extreme case where $\alpha \to 1$ and $Z_\alpha \to +\infty$.

**Correlated Variables.** Since we need to consider the covariances in $K$ hops during path concatenation, we maintain for each $u$-$v$ path $p$ its *head* and *tail*. Suppose w.l.o.g. $v$ is

contracted before $u$ (i.e., $\pi(v) < \pi(u)$). We define its head and tail as the $K$ edges starting from $v$ and the $K$ edges ending at $u$, respectively, as shown in the red and blue parts in Figure 6. Now suppose that we are concatenating a $u$-$v$ path with a $v$-$w$ path as shown in Figure 6. We first find the common vertex $v$, then compute the covariance between them by using the $2K$ edges, and finally maintain the new head and tail by comparing $\pi(u)$ with $\pi(w)$. Note that the common vertex $v$ can be the end vertex of either the head or tail. Also note that the path $p$'s head or tail has fewer than $K$ edges when $|p| < K$.

For the refining operation $RF(P)$, given an upper bound of $\alpha$, we can apply the same technique of using $F_p^{-1}(\alpha)$ as in the independent case. Proposition 4 can be checked with the support of paths' heads and tails. Besides, we maintain a flag for each vertex $v$, which is true if there are edges in $Nei(v)$ that have correlations with other edges and false otherwise. We then ignore $Nei(v)$ if $v$'s flag is false.

Let $\omega = \max_{v \in V} |X(v)|$ be the treewidth and $\eta$ be the treeheight (which is an upper bound for the number of $X(v)$'s ancestors). Let $\overline{|\mathcal{P}_{uv}^{>0.5}|}$ be the average size of $\mathcal{P}_{uw}^{>0.5}$ for any $u$ and $v$. Algorithm 3's time complexity is $\mathcal{O}(|V|\eta\omega\overline{|\mathcal{P}_{uv}^{>0.5}|}^2 \ln \overline{|\mathcal{P}_{uv}^{>0.5}|})$ for the independent case and $\mathcal{O}(|V|\eta\omega\overline{|\mathcal{P}_{uv}^{>0.5}|}^2 \ln \overline{|\mathcal{P}_{uv}^{>0.5}|} |\max_v |Nei_K(v)|)$ for the correlated case, where the difference lies in the refining operation. The space cost of NRP index is $\mathcal{O}(|V|\eta\overline{|\mathcal{P}_{uv}^{>0.5}|})$ since $\mathcal{P}_{uv}^{>0.5}$ is built for each $X(v)$ and each of its ancestor $X(u)$.

*Example 15:* We use the independent case for ease of illustration. Suppose that we are given the tree decomposition in Figure 2 and the contraction order is from $v_1$ to $v_9$. When we contract $v_1$, $X(v_1) = \{v_1, v_2, v_6\}$. For its two neighbors $v_2$ and $v_6$, we add an edge $(v_2, v_6)$ with $P_{(v_2,v_6)} = P_{(v_1,v_2)} \oplus P_{(v_1,v_6)} = \{(4, 10)\}$ and remove $(v_1, v_2)$ and $(v_1, v_6)$. We can similarly contract $v_2$. When we contract $v_3$, $X(v_3) = \{v_3, v_6, v_8\}$. Since $(v_6, v_8)$ exists, we update $P_{(v_6,v_8)} = \{(2, 4), (3, 1)\}$. We similarly process $v_4, v_5, \ldots, v_9$. We next build the labels. We set $L(v_9) = \emptyset$ since $X(v_9)$ has no ancestor. For $X(v_8)$, $\mathcal{P}_{v_8v_9}^{>0.5} = P_{(v_8,v_9)} \oplus \mathcal{P}_{v_9v_9}^{>0.5} = P_{(v_8,v_9)} = \{(5, 5)\}$. For $X(v_7)$, $\mathcal{P}_{v_7v_9}^{>0.5} = P_{(v_7,v_8)} \oplus \mathcal{P}_{v_8v_9}^{>0.5} = \{(4, 7)\}$. We similarly build the rest labels.

## V. Index Maintenance

We need to update the index when the distribution of one edge's travel time changes. To detect such changes, we directly adopt a widely used method since it is not our focus. Updating NRP essentially requires us to recompute non-dominated path sets $\mathcal{P}_{uv}^{>0.5}$. Since $\mathcal{P}_{uv}^{>0.5}$ is built based on the $e$-driven path set $P_e$, we will talk about how to first update $P_e$ and then $\mathcal{P}_{uv}^{>0.5}$.

We detect such distribution changes in a canonical manner as statistically-significant deviations [34]; specifically, the sign of one change is that one sample of the travel time $W_e \sim \mathcal{N}(\mu_e, \sigma_e)$ is outside $\mu_e \pm 2\sigma_e$ at the 5% significance level. It has been shown that the number of changes can be smaller than 10 even in peak hours lasting for one hour [3], [35].

We first update affected $P_e$ in the index. When $W_{(u,w)}$ of an edge $(u, w)$ changes, we need to find all the "center" vertices $v$ such that $P_{(u,w)}$ could be affected as in Line 5 of Algorithm 3

**Algorithm 4:** Updating $P_{(u,w)}$

**input :** The edge $(u, w)$
**output:** The new $P_{(u,w)}$
1 swap $u$ and $w$ if $u$ is contracted after $w$, i.e., $\pi(u) > \pi(w)$
2 initialize $P_{(u,w)}$ with its latest mean and standard deviation
3 **foreach** $v \in C((u, w))$ **do**
4 $\quad P_{(u,w)} \leftarrow RF(P_{(u,w)} \cup (P_{(u,v)} \oplus P_{(v,w)}))$
5 **if** $P_{(u,w)}$ *is different from the original one* **then**
6 $\quad$ **foreach** $v \in X(u)\backslash\{u, w\}$ **do**
7 $\quad\quad$ update $P_{(v,w)}$ by Algorithm 4

**Algorithm 5:** Index Maintenance

**input :** The new travel time $W'_{(u,w)}$ and NRP index
**output:** The updated NRP index
1 update $P_{(u,w)}$ by Algorithm 4
2 **foreach** $X(v)$ *in the subtree rooted at the* $X(r)$ *in a top-down manner* **do**
3 $\quad$ use Lines 7–10 of Algorithm 3 to update $L(v)$

(i.e., $u$ and $w$ are both the neighbors of $v$ during the contraction of $v$). Suppose that the set of such center vertices for each edge $e$ is denoted by $C(e)$. We can similarly initialize $P_{(u,w)}$ by using the latest mean and variance, and for each $v \in C((u, w))$, update it by $RF(P_{(u,w)} \cup (P_{(u,v)} \oplus P_{(v,w)}))$.

Suppose that w.l.o.g. $u$ is contracted before $w$ (i.e., $\pi(u) < \pi(w)$). If $P_{(u,w)}$ changes, some other $P_e$ could also be affected. During the contraction of $u$, for each $u$'s neighbor $v$ other than $w$, $P_{(u,w)}$ would be used to update $P_{(v,w)}$ in Line 5 of Algorithm 3. Note that $X(u)$ include all $u$'s neighbors during the contraction of $u$. We can then check if we have to update each $P_{(v,w)}$ for each $v \in X(u)\backslash\{u, w\}$. If $P_{(v,w)}$ is also affected, we need to do the update procedure recursively.

Algorithm 4 illustrates how to update $P_{(u,w)}$ in a bottom-up manner (following the contraction order). In Line 2, we initialize the edge with its latest mean and standard deviation. We use $C((u, w))$ to update $P_{(u,w)}$ in Lines 3–4. In Lines 5–7, we recursively update $P_{(v,w)}$ for each $v \in X(u)\backslash\{u, w\}$ if $P_{(v,w)}$ is affected.

After we process all the affected $P_e$, we can update $\mathcal{P}_{uw}^{>0.5}$ in a top-down manner as in the construction of labels. For each updated $P_{(u,w)}$, we update $r$ as $u$ if $u$ is contracted after $r$. In this way, the label $L(v)$ for each $v$ that is contracted after $r$ is not affected. For each node $v$ in the subtree rooted at $X(r)$, we update $L(v)$ by Lines 7–10 of Algorithm 3. Algorithm 5 summarizes the whole procedure. In Line 1, we update $P_{(u,w)}$ by Algorithm 4. In Lines 2–3, we rebuild labels of nodes in the subtree rooted at $X(r)$. Note that the whole procedure can be extended to handle a batch of updates by first processing $P_e$ bottom-up and then $L(v)$ top-down in one go.

Let $|V_{sub}|$ be the number of tree nodes in the subtree rooted

at $X(r)$. For each tree node $X(v)$, all of the path sets in its label $L(v)$ are updated in the worst case. Algorithm 5 needs $\mathcal{O}(|V_{sub}|\eta\omega\overline{|\mathcal{P}_{uv}^{\geq 0.5}|}^2 \ln \overline{|\mathcal{P}_{uv}^{\geq 0.5}|})$ time for the independent case, and $\mathcal{O}(|V_{sub}|\eta\omega\overline{|\mathcal{P}_{uv}^{\geq 0.5}|}^2 \ln \overline{|\mathcal{P}_{uv}^{\geq 0.5}|}| \max_v |Nei_K(v)|)$ time for the correlated case.

*Example 16:* For ease of illustration, we assume that all variables are independent. The only difference from the correlated case lies in the refining operation. Suppose that $W_{(v_6,v_8)}$ changes from $\mathcal{N}(2,4)$ to $\mathcal{N}(2,2)$. For $(v_6, v_8)$, we initialize $P_{(v_6,v_8)} = \{(2,2)\}$. We obtain its $C((v_6, v_8)) = \{v_3\}$ and update $P_{(v_6,v_8)} = \{(2,2),(3,1)\}$. Since $X(v_6)\backslash\{v_6, v_8\} = \{v_7, v_9\}$, we check that $P_{(v_7,v_8)}$ and $P_{(v_8,v_9)}$ do not change and stop updating $P_e$. Finally, we find the vertex $r = v_6$ that is last contracted. For each $X(v)$ in the subtree rooted at $X(v_6)$, we rebuild $L(v_6)$, $L(v_2)$, $L(v_3)$, $L(v_4)$, and $L(v_1)$.

## VI. Experiments

### A. Experimental Setup

All experiments were conducted on a machine with two Intel Xeon Gold 5220R 2.2GHz processors and 512GB RAM running CentOS 7 Linux distribution. All algorithms were implemented in C++ and compiled by GNU C++ compiler with O3 optimization.

**Datasets.** We collected three publicly available city road networks from DIMACS [36], including NY, BAY, and COL, with details shown in Table I. They should cover most scenarios since users usually issue RSP queries within a city. The last column gives the approximate diameter $d_{max}$ of each network, which is the maximum shortest distance between two vertices. For the probability distributions of travel times, since the network data from DIMACS provide deterministic travel times for each edge, we directly used them as the mean values $\mu_e$. Following existing work [8], we generated the standard deviation $\sigma_e$ by multiplying the mean with the Coefficient of Variation (CV), defined as the ratio $CV_e = \frac{\sigma_e}{\mu_e}$. Each $CV_e$ was uniformly sampled from $(0, CV)$ where $CV$ is in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The covariances $\sigma_{e_i,e_j}$ were subsequently generated by $\rho_{e_i,e_j}\sigma_{e_i}\sigma_{e_j}$, where $\rho_{e_i,e_j} \in [-1, 1]$ is the correlation coefficient. Since it has been proved that considering the covariances in $K = 4$ hops is sufficient to approximate the path's travel time with at least 99.1% accuracy [7], we only generated covariances between pairs of edges that can be reached to each other in 5 hops. To make the covariance matrix positive semi-definite, we randomly selected $\rho_{e_i,e_j}$ on $[-0.2, 1]$, where $-0.2$ is the minimum feasible value for 6 random variables (corresponding to 5 hops) with pairwise equal correlation coefficients. We evaluated our algorithms by varying the $CV = \{0.1, 0.3, \underline{0.5}, 0.7, 0.9\}$ and $K = \{1, 2, 3, 4, \underline{5}\}$, where the default setting is underlined. For RSP queries, following [26], [37], we tested the effect of the distance between the source and destination of queries and the confidence level $\alpha$. We generated 5 query sets $Q_i$ for varying distances, where $i = \{1, 2, \underline{3}, 4, 5\}$, each of which includes 1,000 queries with random source-destination pairs with their shortest distances lie in $[d_{max}/2^{6-i}, d_{max}/2^{5-i}]$ and $\alpha$ randomly in $[0.7, 0.8]$, and another 5 query sets that all use the 1000 source-destination pairs in $Q_3$ with $\alpha_i$ randomly drawn from $[0.4 + 0.1i, 0.5 + 0.1i]$ for $i = [1, 2, \underline{3}, 4, 5]$.

TABLE I: Real dataset description

| Dataset | Region | $\|V\|$ | $\|E\|$ | $d_{max} \approx$ |
|---|---|---|---|---|
| NY | New York City | 264,346 | 733,846 | 154 km |
| BAY | San Francisco Bay Area | 321,270 | 800,172 | 320 km |
| COL | Colorado | 435,666 | 1,057,066 | 832 km |

We also extracted distributions of real historic travel times from NYC open data provided by the Department of Transportation (DOT) [38]. We first mapped the sensors to the edges by considering the midpoints of the edges and finding the sensors' nearest midpoints. For each sensor, we used all of its recorded travel times from 7:00 am to 7:15 am in September of 2022 to obtain the normal distributions of edges' travel times by using the maximum likelihood estimates.

**Compared algorithms.** Since we considered the correlated case with covariances in our problem, we compared our solution NRP with four baselines for RSP queries that can handle spatial correlations. Note that we omit [1] because it uses the variance-covariance matrix of size $\mathcal{O}(|E|^2)$, which is infeasible in any network.

- SMOGA [17]: a genetic algorithm that repeatedly generates a set of paths, called a population, and optimizes them by changing the current paths in several rounds.
- SDRSP-A* [7]: an A*-based algorithm that uses the M-V dominance to maintain non-dominated paths for labels.
- ERSP-A* [8]: an A*-based algorithm that extends SDRSP-A* and additionally utilizes the M-B dominance.
- TBS [16]. The state-of-the-art algorithm that uses the precomputed reversed paths to prune the search space.

For SMOGA, we set the population size and the number of rounds as 10 and 20 in order to speed up the algorithm. We did not consider the other solutions that are infeasible on large road networks as explained in related work.

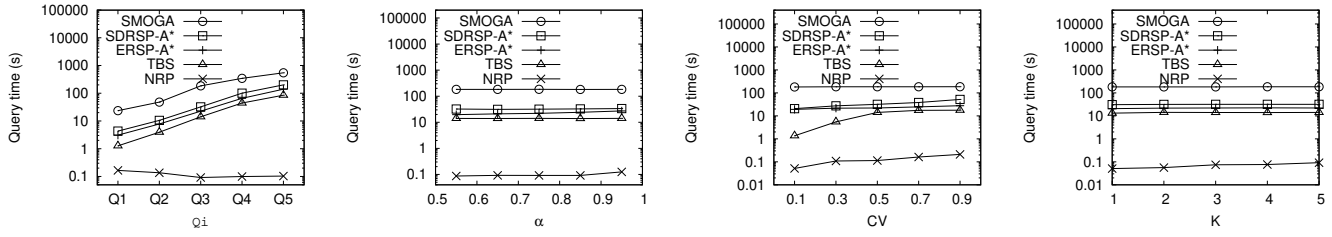
### B. Experiment Results

*1) Query Performance:*

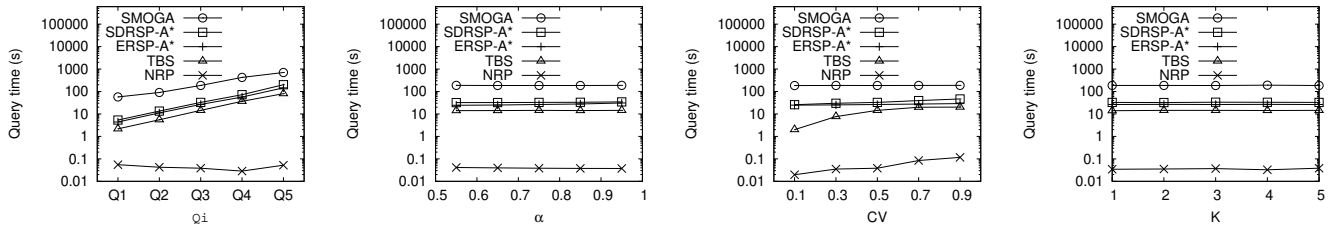
Figure 7 reports the query processing time for a workload of 1,000 queries by varying the four factors stated above.

**Effect of $Q$.** The first column of Figure 7 shows the results of different $Q$. As the distance between the source and destination of each query increases, it can be observed that all algorithms except NRP tend to take more time on all networks. This is because the other four algorithms all search the path from the source to the destination, which suggests that their query times depend on the search range. For NRP, since it utilizes the preprocessed labels for query processing, its query time depends on the number of label lookups and is independent of the distance.

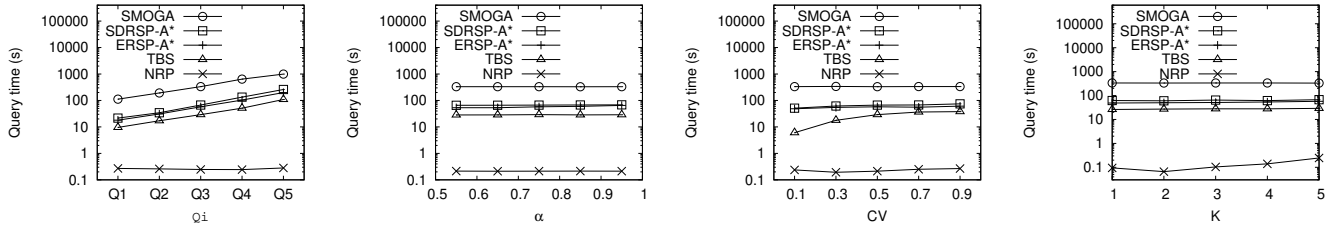
Among all algorithms, NRP runs faster than the others by orders of magnitude due to its small number of label lookups on the preprocessed index. It takes around 0.1 seconds to finish a workload. TBS is the second-best algorithm since its precomputed reversed path can prune many search branches




(a) Time versus $Q$ (NY) (b) Time versus $\alpha$ (NY) (c) Time versus $CV$ (NY) (d) Time versus $K$ (NY)




(e) Time versus $Q$ (BAY) (f) Time versus $\alpha$ (BAY) (g) Time versus $CV$ (BAY) (h) Time versus $K$ (BAY)




(i) Time versus $Q$ (COL) (j) Time versus $\alpha$ (COL) (k) Time versus $CV$ (COL) (l) Time versus $K$ (COL)

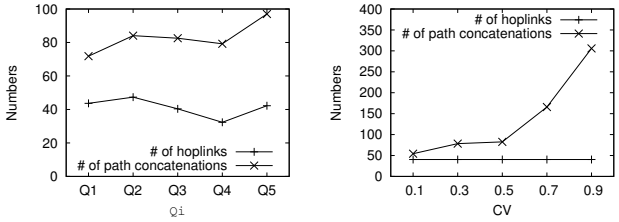
Fig. 7: Query times by varying $Q$, $\alpha$, $CV$, and $K$ in $Nei_K$ on NY, BAY, and COL

during query processing. However, it needs over 100 seconds to finish a workload on COL, which is unacceptable in practice. SMOGA is the worst one due to its multiple iterations of updating the entire path.

**Effect of $\alpha$.** The second column of Figure 7 presents the query time when we vary the confidence level $\alpha$. It can be found that all algorithms are insensitive to $\alpha$. For SMOGA, the population size and the number of rounds are fixed and irrelevant to $\alpha$. The search space of the other algorithms is slightly affected by $\alpha$. Similar performance results can be made. NRP's query time is at least two orders of magnitude shorter than those of the other three, which is mainly due to the index power of fast query processing.

**Effect of $CV$.** We report the query time of different $CV$ in the third column of Figure 7. Recall that $CV$ is the ratio of the standard deviation over the mean. When $CV$ becomes greater, the variance of the path's travel time $W_p$ also increases, which further indicates that there will be more non-dominated paths for each query since the dominance conditions are decided by the variance. It can be seen that all algorithms except SMOGA need slightly more time to answer queries when $CV$ is larger. SMOGA's query time is not affected because the number of its updated paths is independent of the variance. Furthermore, we can similarly find that that NRP is the best one.

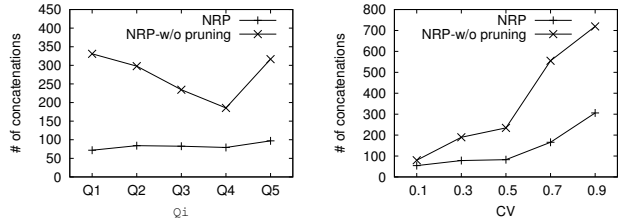
**Effect of $K$.** In the last column of Figure 7, we depict the results of different $K$, where the $K$ value implies that the covariances exist between edges that are $K$ hops away. We can find that the query times of all the algorithms are independent of $K$. For SMOGA, the reason is the same as before. For the other four, the non-dominated paths are essentially affected by the variance of the path's travel time, to which the covariance can make a limited variation. In addition, NRP beats the others by a large margin as before.




(a) Statistics versus $Q$ (b) Statistics versus $CV$

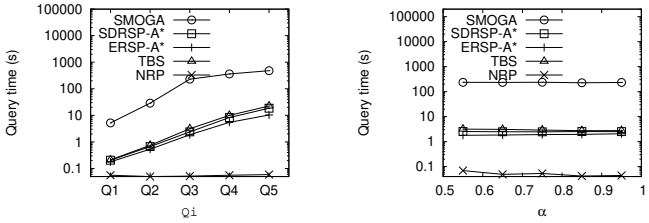
Fig. 8: Numbers of hoplinks and path concatenations (NY)

**Numbers of hoplinks and path concatenations.** To analyze NRP's performance further, we record the average numbers of used hoplinks $h$ and performed path concatenations per query in Figure 8. Note that the latter is the dominant term in the query time complexity. In Figure 8a, we can find that the numbers of hoplinks and path concatenations are insensitive to $Q$ with the same reason as before. In Figure 8b, the number of hoplinks remains the same because we use the same queries in $Q_3$. For the hoplinks, we select the separator based on the LCA of the two tree nodes $X(s)$ and $X(t)$. Therefore, the number of hoplinks is only relevant to the source and destination of




(a) Statistics versus $Q$ (b) Statistics versus $CV$

Fig. 9: Ablation study on NY




(a) Time versus $Q$ (b) Time versus $\alpha$

Fig. 10: Real NYC Open Data from DOT

each query. For the number of path concatenations, it increases as $CV$ becomes larger since the larger variance would result in more non-dominated paths.

**Ablation study.** We compare the variant without using the pruning techniques, called "NRP-w/o pruning" in terms of the number of path concatenations in Figure 9. In both two figures, it can be observed that there is a dramatic decrease in the number when the pruning is applied under each setting, which further demonstrates the effectiveness of the pruning techniques. Moreover, we can observe a similar trend to that of NRP's query time since the number of path concatenations dominates the query time complexity.

**Real travel times.** The query times of varying $Q$ and $\alpha$ on NYC open data from DOT is shown in Figures 10a and 10b, respectively. We can find similar results that all the algorithms tend to take more query times when the distance is larger and that they are insensitive to $\alpha$. Among all algorithms, NRP performs the best and takes less than 0.1 seconds.

*2) Index Cost:*

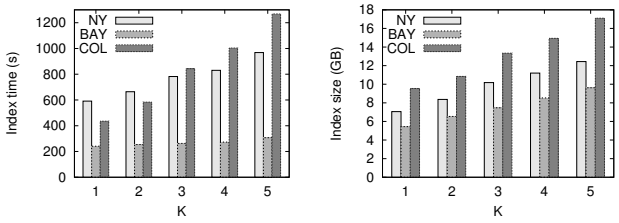
We omit SMOGA, SDRSP-A*, and ERSP-A* since they do not build any index. The index costs of TBS and NRP are summarized in Table II. The second to the last columns list the treewidth $\omega$, the treeheight $\eta$, NRP's index time, NRP's index size, TBS's index time, and TBS's index size, respectively.

**Index construction time.** We explore the index time by varying $K$ in Figure 11a. It can be seen that the time for each network gradually increases linearly with $K$. This is because we need to compute more terms about covariances between pairs of two paths and also spend more time checking the dominance conditions. NY could take more time than BAY and COL when $K$ is 1 and 2 because NY has a more densely distributed grid layout, but the other two networks are sparsely distributed since there can be bays and mountains. It can be found that NRP takes smaller indexing time than TBS.

**Index storage size.** The space consumption of different $K$ is shown in Figure 11b. We can obtain a similar conclusion that the space cost grows linearly with $K$. The last column of Table II shows the index size of the default setting. NRP's index size is also much smaller than TBS's, and NRP supports faster query processing.

TABLE II: Index cost

| Dataset | $\omega$ | $\eta$ | NRP | | TBS | |
|---|---|---|---|---|---|---|
| | | | Time | Size | Time | Size |
| NY | 148 | 330 | 968s | 12.4 GB | 14,076s | 130.1 GB |
| BAY | 100 | 238 | 307s | 9.6 GB | 21,460s | 192.2 GB |
| COL | 143 | 423 | 1269s | 17.1 GB | 37,058s | 353.5 GB |




(a) Index time versus $K$ (b) Index size versus $K$

Fig. 11: NRP's index costs on NY

**Index maintenance cost.** Table III records NRP's average processing time over 1,000 updates of travel times of randomly selected edges and the extra storage about the set of center vertices $C(e)$. Note that we omit TBS here since it does not consider distribution changes [16]. Following existing work about path index maintenance [27], we increase and decrease the original $\mu_e$ to a random value in $[0.5\mu, \mu]$ and $[\mu, 2\mu]$, respectively, and do similarly for $\sigma_e$. The results show that the time cost is insensitive to specific operations. The extra storage is relatively small compared with the entire index.

*3) Case Study:*

In Figure 12, we conducted a case study in New York. Suppose that a businessman at the blue point wants to reach the red point to attend an important meeting during the morning rush hour. The fastest path (shown by the brown line) that uses deterministic values as edges' estimated travel times is expected to take around 16 minutes for 14.4 km. However, it traverses the Cross Bronx Expressway in the shaded region, which can be congested with potential traffic accidents [39], and its real travel time can be 39 minutes long. Considering the uncertainty, we set a high confidence level of 0.95 to find the RSP (shown by the navy blue line), which takes at most 25 minutes for 20.8 km and does not traverse the Cross Bronx Expressway. It would be more reliable for the businessman.

### C. Summary

*(i)* Our proposed *NRP* index-based solution has its query processing time orders of magnitude faster than state-of-the-art competitors. For a workload of 1,000 queries, it only needs around 0.1 seconds in city road networks, whereas existing RSP solutions can take 100 seconds.

*(ii)* NRP's index cost on city road networks are acceptable.

*(iii)* The proposed pruning techniques can improve both the time and space efficiencies.

TABLE III: Index update time (sec) and extra storage (NY)

| Dataset | Inc. $\mu$ | Dec. $\mu$ | Inc. $\sigma$ | Dec. $\sigma$ | Extra Storage |
|---|---|---|---|---|---|
| NY | 0.9142 | 0.9272 | 0.9093 | 0.9125 | 63.58 MB |
| BAY | 0.2791 | 0.2776 | 0.2774 | 0.2759 | 31.55 MB |
| COL | 1.2819 | 1.2799 | 1.2713 | 1.2674 | 30.97 MB |

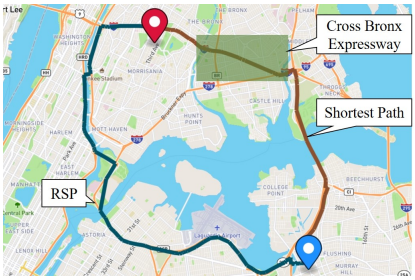



Fig. 12: Case study in New York

## VII. Related Work

### A. RSP for the Independent Case

The seminal work proposed the most reliable shortest path (RSP) that has the maximum arrival probability under a given travel time upper bound [40]. Following it, the $\alpha$-reliable shortest path (i.e., our focus) was introduced [6]. It was shown that the two optimal answers to the two problems are equivalent under some conditions [41]. For the most RSP, it has been proven to be NP-hard by using the reduction from the *longest path problem* [42]. However, it is often hard for users to set the upper bound in the most RSP when they are unfamiliar with the trips, and using inappropriate small values can make the arrival probability less than 0.5, which makes the event unlikely to happen. For $\alpha$-RSP, early solutions were based on the label-correcting algorithms that search from the source, maintain a label from the source to each visited vertex (which is essentially a non-dominated path set), and choose a promising path from labels for expansion [20], [41]. The size of the non-dominated path set could be reduced under different dominance definitions. The first-order stochastic dominance was first applied [41]. Then, [20] considered the M-V dominance [18] and the M-B dominance [19] to prune more paths and also utilized the idea of the A* algorithm to speed up the search. They expanded the currently best path with the minimum $F_p^{-1}(\alpha)$ value plus the distance to the destination. Other studies considered the skyline definition of path queries [10], [43] and the reachability [44]. However, they all assume that travel times are independent, which is unrealistic. Moreover, though the high-level idea of path dominance has been used, our proposed dominance conditions are tailored to our solution and thus more efficient for stochastic routing.

A line of research considered the time-dependent shortest path query [45]–[53], where travel times are predictable and given by piecewise linear functions. Its problem is different from ours due to its deterministic travel times. Though it can handle time-of-day dependent variations, RSP can model the inherent uncertainty in travel times caused by unpredictable factors. Furthermore, RSP can maintain the index to adapt to distribution changes during one day. The distributions of travel times can be dependent on the time of day.

### B. RSP for the Correlated Case

Since we consider the correlation between edges' travel times, our work falls into this category. To handle the correlated case, extensive solutions have been proposed [1], [7]–[9], [11]–[17]. Most of them were search-based routing algorithms that expand the path answer incrementally by comparing the network edges sequentially. They include the depth-first search [9], [12], the two-directional search [15], and the A* guided-search with tailored lower bounds [7], [8], [11], [13], [14], [16]. Specifically, the state-of-the-art TBS used the bounds obtained from a reversed search to prune the search space [16]. The original TBS finds the most RSP introduced before. Since the most RSP is equivalent to $\alpha$-RSP [41], we adapt TBS to return exactly the same route as NRP's to compare their performance in experiments. However, all search-based solutions are still inefficient because they need to search the whole network. Our NRP can preprocess partial path answers in the index and fetch them efficiently from the index by table lookups during query processing. [1] decomposed the variance-covariance matrix and utilized Lagrangian relaxation to solve the dual problem. However, the matrix could be large and matrix operations are time-consuming with their time complexities $\Omega(|E|^2)$, which are prohibitive in large road networks. [17] applied the heuristic genetic algorithm to repeatedly optimize the answer. In each round, it first selects a set of candidate paths, then uses a defined fitness function to evaluate these paths, and generates new paths by performing some common operations, such as crossover and mutation, on the paths with high fitness values. However, it may not find the exact answer and need many iterations of updating a number of candidate $s$-$t$ paths. In sum, all these solutions are inefficient and unscalable to large road networks. Other studies considered how to select the best path from a candidate set [54] and a different arrival window [55], but their problem settings are completely different from ours.

## VIII. Conclusion

This paper addresses the problem of efficiently answering the reliable shortest path (RSP) queries in stochastic road networks. We propose an efficient index-based solution called NRP, which runs faster than state-of-the-art ones by orders of magnitude. NRP stores non-dominated paths in the labels and processes queries by simply looking up a limited number of labels. The experiments conducted on three real networks verified the superiority of NRP, which can answer each query in around 100 $\mu$s on New York's road network. For future work, we may explore how to efficiently update the index when the weight distributions of the edges frequently change. We may also study some time-dependent reliability metrics.

## Acknowledgements

We thank reviewers for their valuable comments and effort to improve this manuscript. Libin Wang and Raymond Chi-Wing Wong are supported by fund WEB24EG01-H.

## REFERENCES


[1] W. Zeng, T. Miwa, Y. Wakita, and T. Morikawa, "Application of lagrangian relaxation approach to $\alpha$-reliable path finding in stochastic networks with correlated link travel times," *Transportation Research Part C: Emerging Technologies*, vol. 56, pp. 309–334, 2015.

[2] B. Y. Chen, C. Shi, J. Zhang, W. H. Lam, Q. Li, and S. Xiang, "Most reliable path-finding algorithm for maximizing on-time arrival probability," *Transportmetrica B: Transport Dynamics*, vol. 5, no. 3, pp. 248–264, 2017.

[3] P. Chen, R. Tong, G. Lu, and Y. Wang, "Exploring travel time distribution and variability patterns using probe vehicle data: case study in beijing," *Journal of Advanced Transportation*, vol. 2018, 2018.

[4] Z. Zang, X. Xu, K. Qu, R. Chen, and A. Chen, "Travel time reliability in transportation networks: A review of methodological developments," *Transportation Research Part C: Emerging Technologies*, vol. 143, p. 103866, 2022.

[5] Z. Zang, R. Batley, X. Xu, and D. Z. Wang, "On the value of distribution tail in the valuation of travel time variability," *Transportation Research Part E: Logistics and Transportation Review*, vol. 190, p. 103695, 2024.

[6] A. Chen and Z. Ji, "Path finding under uncertainty," *Journal of advanced transportation*, vol. 39, no. 1, pp. 19–37, 2005.

[7] B. Y. Chen, W. H. Lam, A. Sumalee, and Z.-l. Li, "Reliable shortest path finding in stochastic networks with spatial correlated link travel times," *International Journal of Geographical Information Science*, vol. 26, no. 2, pp. 365–386, 2012.

[8] B. Y. Chen, W. H. Lam, and Q. Li, "Efficient solution algorithm for finding spatially dependent reliable shortest path in road networks," *Journal of advanced transportation*, vol. 50, no. 7, pp. 1413–1431, 2016.

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

[10] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *ICDE*, 2014.

[11] G. Andonov and B. Yang, "Stochastic shortest path finding in path-centric uncertain road networks," in *MDM*, 2018.

[12] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, "PACE: a path-centric paradigm for stochastic path finding," *VLDB J.*, vol. 27, no. 2, pp. 153–178, 2018.

[13] S. A. Pedersen, B. Yang, and C. S. Jensen, "Anytime stochastic routing with hybrid learning," *PVLDB*, vol. 13, no. 9, pp. 1555–1567, 2020.

[14] ——, "A hybrid learning approach to stochastic routing," in *ICDE*, 2020.

[15] ——, "Fast stochastic routing under time-varying uncertainty," *VLDB J.*, vol. 29, no. 4, pp. 819–839, 2020.

[16] C. Guo, R. Xu, B. Yang, Y. Ye, T. Kieu, Y. Zhao, and C. S. Jensen, "Efficient stochastic routing in path-centric uncertain road networks," *PVLDB*, 2024.

[17] Z. Ji, Y. S. Kim, and A. Chen, "Multi-objective $\alpha$-reliable path finding in stochastic networks with correlated link costs: A simulation-based multi-objective genetic algorithm approach (smoga)," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1515–1528, 2011.

[18] S. Sen, R. Pillai, S. Joshi, and A. K. Rathi, "A mean-variance model for route guidance in advanced traveler information systems," *Transportation Science*, vol. 35, no. 1, pp. 37–49, 2001.

[19] K. R. Hutson and D. R. Shier, "Extended dominance and a stochastic shortest path problem," *Comput. Oper. Res.*, vol. 36, no. 2, pp. 584–596, 2009.

[20] B. Y. Chen, W. H. Lam, A. Sumalee, Q. Li, H. Shao, and Z. Fang, "Finding reliable shortest paths in road networks under uncertainty," *Networks and spatial economics*, vol. 13, no. 2, pp. 123–148, 2013.

[21] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "LC-RNN: A deep learning model for traffic speed prediction," in *IJCAI*, 2018.

[22] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *SIGKDD*, 2018.

[23] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *SIGMOD*, 2020.

[24] L. Zou, J. Wang, M. Cheng, and J. Hang, "Travel time reliability estimation in urban road networks: Utilization of statistics distribution and tensor decomposition," *Journal of Advanced Transportation*, vol. 2024, no. 1, p. 4912642.

[25] Y. Jiang and X. Li, "Travel time prediction based on historical trajectory data," *Ann. GIS*, vol. 19, no. 1, pp. 27–35, 2013.

[26] D. Ouyang, L. Qin, L. Chang, X. Lin, Y. Zhang, and Q. Zhu, "When hierarchy meets 2-hop-labeling: Efficient shortest distance queries on road networks," in *SIGMOD*, 2018.

[27] Z. Chen, A. W. Fu, M. Jiang, E. Lo, and P. Zhang, "P2H: efficient distance querying on road networks by projected vertex separators," in *SIGMOD*, 2021.

[28] L. Wang and R. C. Wong, "QHL: A fast algorithm for exact constrained shortest path search on road networks," in *SIGMOD*, 2023.

[29] Y. Qiu, D. Wen, L. Qin, W. Li, R. Li, and Y. Zhang, "Efficient shortest path counting on large road networks," *PVLDB*, 2022.

[30] L. Wang and R. C. Wong, "PCSP: efficiently answering label-constrained shortest path queries in road networks," *PVLDB*, vol. 17, no. 11, pp. 3082–3094, 2024.

[31] N. Robertson and P. D. Seymour, "Graph minors. III. planar tree-width," *J. Comb. Theory, Ser. B*, vol. 36, no. 1, pp. 49–64, 1984.

[32] L. Chang, J. X. Yu, L. Qin, H. Cheng, and M. Qiao, "The exact distance to destination in undirected world," *VLDB J.*, vol. 21, no. 6, pp. 869–888, 2012.

[33] M. El Esawey and T. Sayed, "Travel time estimation in urban networks using limited probes data," *Canadian Journal of Civil Engineering*, vol. 38, no. 3, pp. 305–318, 2011.

[34] S. R. Jeffery, M. N. Garofalakis, and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *VLDB*, 2006.

[35] Z. Chen and W. D. Fan, "Analyzing travel time distribution based on different travel time reliability patterns using probe vehicle data," *International Journal of Transportation Science and Technology*, vol. 9, no. 1, pp. 64–75, 2020.

[36] "9th dimacs implementation challenge," 2010, http://www.diag.uniroma1.it/challenge9/download.shtml.

[37] Z. Liu, L. Li, M. Zhang, W. Hua, P. Chao, and X. Zhou, "Efficient constrained shortest path query answering with forest hop labeling," in *ICDE*, 2021.

[38] "NYC department of transportation," 2025, https://data.cityofnewyork.us/Transportation/DOT-Traffic-Speeds-NBE/i4gi-tjb9/about_data.

[39] "Cross bronx expressway," 2019, https://traffictickets.com/blog/3-nyc-roads-named-most-congested-in-u-s/.

[40] H. Frank, "Shortest paths in probabilistic graphs," *Oper. Res.*, vol. 17, no. 4, pp. 583–599, 1969.

[41] Y. M. Nie and X. Wu, "Shortest path problem considering on-time arrival probability," *Transportation Research Part B: Methodological*, vol. 43, no. 6, pp. 597–613, 2009.

[42] Y. Xiao, K. Thulasiraman, X. Fang, D. Yang, and G. Xue, "Computing a most probable delay constrained path: Np-hardness and approximation schemes," *IEEE Trans. Computers*, vol. 61, no. 5, pp. 738–744, 2012.

[43] S. Aljubayrin, B. Yang, C. S. Jensen, and R. Zhang, "Finding non-dominated paths in uncertain road networks," in *SIGSPATIAL*, 2016.

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

[45] L. Li, S. Wang, and X. Zhou, "Time-dependent hop labeling on road network," in *ICDE*, 2019.

[46] S. Wang, W. Lin, Y. Yang, X. Xiao, and S. Zhou, "Efficient route planning on public transportation networks: A labelling approach," in *SIGMOD*, 2015.

[47] Y. Wang, G. Li, and N. Tang, "Querying shortest paths on time dependent road networks," *PVLDB*, 2019.

[48] Y. Yuan, X. Lian, G. Wang, Y. Ma, and Y. Wang, "Constrained shortest path query in a large time-dependent graph," *PVLDB*, 2019.

[49] L. Li, W. Hua, X. Du, and X. Zhou, "Minimal on-road time route scheduling on time-dependent graphs," *PVLDB*, 2017.

[50] L. Li, S. Wang, and X. Zhou, "Fastest path query answering using time-dependent hop-labeling in road network," *TKDE*, vol. 34, no. 1, pp. 300–313, 2022.

[51] L. Li, K. Zheng, S. Wang, W. Hua, and X. Zhou, "Go slow to go fast: minimal on-road time route scheduling with parking facilities using historical trajectory," *VLDB J.*, 2018.

[52] D. Chen, Y. Yuan, W. Du, Y. Cheng, and G. Wang, "Online route planning over time-dependent road networks," in *ICDE*, 2021.

[53] Z. Gong, Y. Zeng, and L. Chen, "Querying shortest path on large time-dependent road networks with shortcuts," in *ICDE*, 2024.

[54] J. Hu, B. Yang, C. Guo, and C. S. Jensen, "Risk-aware path selection with time-varying, uncertain travel costs: a time series approach," *VLDB J.*, vol. 27, no. 2, pp. 179–200, 2018.

[55] S. A. Pedersen, B. Yang, C. S. Jensen, and J. Møller, "Stochastic routing with arrival windows," *ACM Trans. Spatial Algorithms Syst.*, vol. 9, no. 4, pp. 30:1–30:48, 2023.