Finding Non-Dominated Paths in Uncertain Road Networks

Saad Aljubayrin
Department of Computing and Information Systems
University of Melbourne
Australia
aljubayrin@su.edu.sa

Bin Yang
Department of Computer Science
Aalborg University
Denmark
byang@cs.aau.dk

Christian S. Jensen
Department of Computer Science
Aalborg University
Denmark
csj@cs.aau.dk

Rui Zhang
Department of Computing and Information Systems
University of Melbourne
Australia
rui.zhang@unimelb.edu.au

ABSTRACT

With the rapidly growing availability of vehicle trajectory data, travel costs such as travel time and fuel consumption can be captured accurately as distributions (e.g., travel time distributions) instead of deterministic values (e.g., average travel times). We study a new path finding problem in uncertain road networks, where paths have travel cost distributions. Given a source and a destination, we find optimal, non-dominated paths connecting the source and the destination, where the optimality is defined in terms of the stochastic dominance among cost distributions of paths. We first design an A^* based framework that utilizes the uncertain graph to obtain the most accurate cost distributions while finding the candidate paths. Next, we propose a three-stage dominance examination method that employs extreme values in each candidate path's cost distribution for early detection of dominated paths, thus reducing the need for expensive distributions convolutions. We conduct extensive experiments using real world road network and trajectory data. The results show that our algorithm outperforms baseline algorithms by up to two orders of magnitude in terms of query response time while achieving the most accurate results.

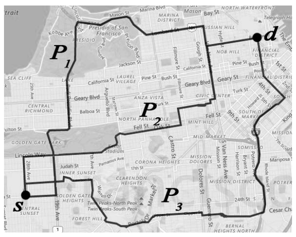
Keywords

Path Finding; Stochastic Dominance; Travel Cost Distributions.

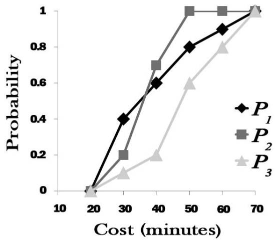
1. INTRODUCTION

It is becoming increasingly possible to capture traffic conditions of road networks in detail either through real-time updates or through historical data. This is due to the rapid increase in available vehicle trajectory data, which enables a more detailed and accurate capture of traffic [4, 8]. Specifically, instead of capturing traffic by means of average travel cost values for road segments, it is possible to derive travel cost distributions from trajectories. Consequently, paths can also be associated with travel cost distributions, and it is no longer appropriate to directly apply the concept of "shortest path" to compare such paths with travel cost distributions. Although any form of travel cost (e.g., travel time, fuel consumption, or greenhouse gas emissions [3, 11]) can be used in the problem settings, we focus on travel time.

WOODSTOCK '97 El Paso, Texas USA
© 2016 ACM. ISBN 00/00/00…$
DOI: 00/00



(a) Three Paths



(b) CDFs

Figure 1: Motivating Example

We employ stochastic dominance to compare paths with cost distributions. In particular, given a source and a destination, we aim at identifying all paths that are not stochastically dominated by other paths. We call such non-dominated paths *optimal paths*. Then, it is up to users to choose a particular path from the optimal paths to travel according to the users' personal preferences. This study enables the accurate and efficient identification of optimal, non-dominated paths.

Figure 1a shows three different paths, $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, from a source s to a destination d. Figure 1b shows the corresponding cumulative distribution functions (CDFs) of the travel times of the three paths. Consider two users u_1 and u_2 who would like to go from hotel s to airport d, but with different time budgets. User u_1 needs to catch a flight and thus wants to make sure to arrive at d in 50 minutes. On the other hand, user u_2 wants to try to meet with his friend who has already arrived at the airport and thus is interested in the path with the highest probability of arriving in the shortest possible time, which is 30 minutes. Based on the cumulative distribution functions, user u_1 should take $\mathcal{P}_2$ since it gives the highest probability of reaching the destination within 50 minutes, while user u_2 should take $\mathcal{P}_1$ as it offers the best probability of arriving early. Path $\mathcal{P}_3$ is not of interest to u_1 or u_2.

The intuition of the example can be explained in terms of stochastic dominance. We say that one distribution stochastically dominates another distribution if for every value, the former distribution has a higher cumulative probability than the latter distribution. For example, $\mathcal{P}_1$'s travel time distribution dominates that of $\mathcal{P}_3$; $\mathcal{P}_2$'s travel time distribution dominates that of $\mathcal{P}_3$; but $\mathcal{P}_1$'s travel time distribution and $\mathcal{P}_2$'s travel time distribution do not dominate each other. Thus, $\mathcal{P}_1$ and $\mathcal{P}_2$ are optimal paths as they can maximize the probability of arriving the destination within a user's specific, time budget, while $\mathcal{P}_3$ can never be such a path, no matter what time budget a user may have.

This example also suggests that travel cost distributions are essential for decision making in many path planning scenarios–simply considering mean travel costs is insufficient. For example, if we only consider the mean travel times of the three paths, $\mathcal{P}_2$ with mean 41 minutes is always the best as the other paths ($\mathcal{P}_1$ at 43 minutes and $\mathcal{P}_3$ at 53 minutes) take longer time. The use of mean travel times fails to meet u_1's and u_2's different needs (i.e., different time budgets).

To solve the optimal route planning problem, we need to consider two aspects. First, we need to consider how to model a road network with uncertain travel costs. Second, we need to consider how to efficiently identify optimal paths in the resulting road network.

To address these aspects, we propose a Hybrid Graph to model the uncertain travel costs in a road network. A hybrid graph is a weighted graph, where not only edges have weights, represented by travel cost distributions, but where paths also have cost distributions. The hybrid graph enables more accurate travel cost estimation than does an ordinary weighted graph, where only edges have weights. Next, we design a solution framework to explore candidate paths in the network and that uses the hybrid graph to obtain the most accurate distribution for each path. In this setting, we propose a three-stage algorithm to detect stochastic dominance between candidate paths.

In summary, we make the following contributions:

1. We propose the new problem of non-dominated path finding in uncertain road networks.

2. We model a road network as a hybrid graph of sub-paths associated with random variables representing the cost distributions. The hybrid graph takes time dependency into account by choosing the right time partition and shifting between time partitions as needed.

3. We design a solution framework that maintains a list of competitor paths and utilizes the hybrid graph to choose the most accurate available distributions for each path.

4. We propose a three-stage effective and efficient algorithm to detect and exclude dominated paths. This algorithm avoids expensive and unnecessary convolutions of dominated paths distributions.

5. We conduct extensive experiments to evaluate the efficiency and the effectiveness of the proposed framework and algorithm, obtaining the following findings:

   (a) In terms of query processing, our algorithm is more than an order of magnitude faster than an existing baseline algorithm and two orders of magnitude faster than a naive algorithm.

   (b) In terms of effectiveness, the set of non-dominated paths computed by our algorithm are more accurate than those computed by the baseline algorithm and as accurate as those obtained using the naive algorithm.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. Section 3 presents the preliminaries . Sections 4 details the solution framework and the proposed algorithm. Section 5 presents and discusses the experimental results, and Section 6 concludes.

2. RELATED WORK

The shortest path routing problem has been studied intensively, and routing in static road networks (e.g., distance based algorithms) [2,7] as well as routing in dynamic (i.e., time-varying) road networks (e.g., travel-time based algorithms) [6] have been considered. However, we are unaware of previous work on the problem of using travel cost distributions to find all non-dominated paths. Most existing studies estimates deterministic travel costs, where the total cost of a path is the sum of the travel costs of the segments in the path [16, 22, 26, 27]. Some studies use real trajectories to extract the travel cost of each segment if there are sufficient trajectories, while other studies use techniques to estimate the costs of road segments without trajectories. For example, they use similar costs for adjacent segments [27] and topologically similar segments [22, 26] to contend with sparseness. Furthermore, some studies (e.g, [22, 26, 27]) consider time dependency to determine the most accurate cost on the network segments (e.g., peak versus off peak hours). All mentioned studies under this category do not apply to our problem since we deal with distributions rather than single-value costs on segments.

Studies exist that are more related to our problem, specifically, uncertain travel cost routing studies [5, 13, 17, 23–25]. Here, the travel costs of road segments are modeled as cost distribution instead of static costs. Some studies assume that the cost distribution of each segment follows a standard distribution (e.g., a Gaussian distribution), However, according to recent studies [23, 24], in many cases, the travel costs generally follow arbitrary distributions. Dependencies between segments is addressed by some studies [5], while other studies assume that segments distribution are independent of each other [13, 17]. All the above studies are inapplicable to our problem as they do not aim to find a set of non-dominated paths.

Our study is built on a recent study [5], that proposes an algorithm to learn a set of random variables from real trajectories for not only edges, but also for paths. This study also offers an algorithm to identify an optimal set of random variables of a path such that the path's cost distribution can be estimated accurately. This work aims to find the most accurate representation of a path distribution by reducing the ratio of entropy in the chosen optimal random variable set. To the best of our knowledge, this is the most accurate method of obtaining travel cost path distributions and we thus use it as the foundation of our study.

Another category of related work is work on skyline path queries [1, 15, 24]. In such studies, the goal is to find all non-dominated paths based on multiple cost types. For example, these cost types can be distance, travel time, and the number of traffic lights [15]. Skyline paths are also used in trip planning queries [1]. The main difference between these studies and our problem is that, the previous studies deal with multiple cost types, while we deal with the same cost types (e.g., travel time only), but yet contend, with different distributions. Therefore, the algorithms proposed in these studies cannot be applied to our problem.

3. PRELIMINARIES

We cover the basic concepts and define the problem.

3.1 Basic Concepts

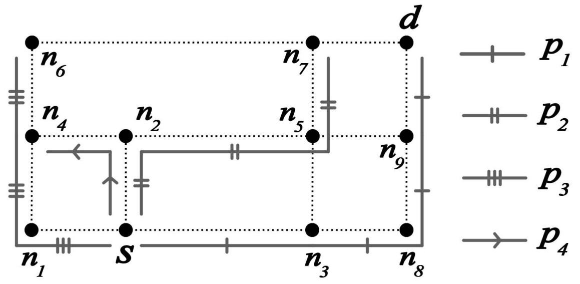


Figure 2: An Example Road Network and Paths

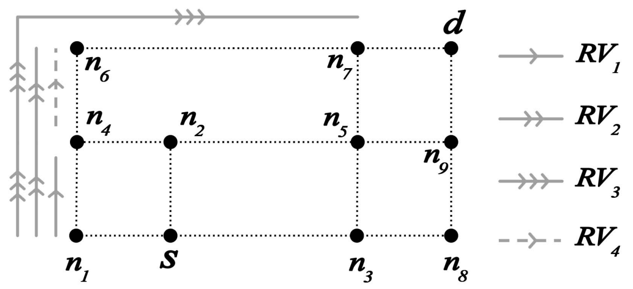


Figure 3: Hybrid Graph with its $\mathcal{RV}s$

A **road network** $\mathcal{N}$ is a directed graph $\mathcal{N} = (V, E)$ where V is a set of nodes representing road intersections and E is a set of edges representing road segments connecting nodes. A path $\mathcal{P} = \langle e_1, e_2, ..., e_i \rangle$ is a sequence of road segments where each adjacent road segment pair must share a node. The cardinality $|\mathcal{P}|$ of a path $\mathcal{P}$ is defined as the number of road segments in $\mathcal{P}$. A path $\mathcal{P}' = \langle q_1, q_2, .., q_j \rangle$ is a sub-path of path $\mathcal{P} = \langle e_1, e_2, .., e_i \rangle$ if there exists a $k \in [0, i - j]$ such that $\langle q_1, q_2, .., q_j \rangle = \langle e_{k+1}, e_{k+2}, .., e_{k+j} \rangle$. Figure 2 shows an example road network with some paths.

A **trajectory** $\mathcal{T} = \langle p_1, p_2, ..., p_i \rangle$ is a sequence of GPS records representing a vehicle trip in the road network. Each GPS record $gps_i = (l, t)$ consists of a location l and a time-stamp t. Map matching is able to map each GPS record to a specific location of a road segment, thus aligning trajectories with paths. Next, we are able to obtain the travel costs of using paths at different times based on the trajectories that occurred on the paths. In particular, the travel time can be obtained by measuring the difference between the time of the last GPS record and the first GPS record on a path; and the fuel consumption for traversing a path can be computed by applying vehicular environmental impact models [9, 10] and 3D road networks [14] to the trajectories associated with the path.

We use a **random variable** $\mathcal{RV}$ to describe the cost distribution of traversing a path $\mathcal{P}$ during a certain time interval I of a day. In particular, a random variable's *probability mass function* (PMF) is represented as a sequence of $(cost_i{:}probability_i)$ pairs, where the sequence is ordered by cost values $cost_i$ and the probabilities sum up to one, i.e., $\sum_i probability_i = 1$. The *cumulative probability function* (CDF) of a random variable is represented as a sequence of $(cost_i : \sum_{k=1}^{i} probability_k)$ pairs, where a cost value $cost_i$'s cumulative probability is the sum of the mass probabilities of the cost values that are less than or equal to $cost_i$.

For example, assume that 100 trajectories occurred on path $\mathcal{P}_1 = \langle e_1, e_2, e_3, e_4 \rangle$ during morning rush hours from 7:00 a.m. to 9:00 a.m., where 20 trajectories took 10 minutes, 50 trajectories took 15 minutes, and 30 trajectories took 17 minutes. We can create a random variable $\mathcal{RV}_1 = \langle 10 : 0.2, 15 : 0.5, 17 : 0.3 \rangle$ for the path during (7:00, 9:00), representing that traversing the path using 10, 15, and 17 minutes have probabilities of 0.2, 0.5, and 0.3, respectively. The corresponding CDF of the random variable is $\langle 10 : 0.2, 15 : 0.7, 17 : 1.0 \rangle$, meaning that traversing the path using less than 10, 15, and 17 minutes have probabilities of 0.2, 0.7, and 1.0, respectively.

3.2 Hybrid Graph

The **Hybrid Graph** (HG) is a weighted graph built on top of the road network $\mathcal{N}$. Essentially, it maintains the following weight function.

$$HG : \{\mathcal{P}\} \times T \rightarrow \{\mathcal{RV}\},$$

where $\{\mathcal{P}\}$ indicates a set of paths in the road network and $\{\mathcal{RV}\}$ denotes a set of random variables, T indicates a set of time stamps. Given a path $\mathcal{P}_i$ and a time stamp t_j, the hybrid graph is able to return a random variable describing the travel cost distribution of traversing path $\mathcal{P}_i$ at t_j.

The hybrid graph is instantiated using historical trajectories [5]. We partition a day into peak and off-peak intervals. If a sufficient number (e.g., 100) of trajectories traverse a path $\mathcal{P}$ during an interval, we derive a random variable for $\mathcal{P}$ for the interval by using the trajectories. The reason that we only instantiate random variables for paths with a sufficient number of trajectories is that we want to avoid derived random variables that are over-fitted to a small number of trajectories' cost values, which may be biased.

A path is a unit path if it has cardinality 1. If a unit path does not have a sufficient number of trajectories, we instantiate a random variable $(\frac{length}{speed\_limit} : 1.0)$ for the path using its length and speed limit.

In the hybrid graph, a random variable $\mathcal{RV}_i$ associated with a path $\mathcal{P}_i$ describes the travel cost distribution of the path. We introduce the concept of *rank* for a random variable. Specifically, the rank $|\mathcal{RV}_i|$ of a random variable $\mathcal{RV}_i$ is the cardinality of the corresponding path, i.e., $|\mathcal{RV}_i| = |\mathcal{P}_i|$.

Thus, for each unit path, the hybrid graph maintains two random variables with rank 1, one for each of the peak and off-peak hour periods. The random variables are either derived from speed limits or from the trajectories that occurred on the unit path. For each non-unit path, the hybrid graph maintains a random variable with rank greater than 1 if a sufficient number of trajectories occurred on the path during the interval considered. This means that, given different historical trajectory sets, the hybrid graph may maintain different random variables with rank greater than 1. Figure 3 shows an example of the hybrid graph during peak hours. We have $|\mathcal{RV}_1| = 1$, $|\mathcal{RV}_2| = 2$, $|\mathcal{RV}_3| = 3$, and $|\mathcal{RV}_4| = 1$.

Next, we describe how to use the hybrid graph to derive the cost distribution of a path. Given a path $\mathcal{P}_i$ along with a departure time t, we first examine the relevant intervals, e.g., peak or off-peak hours. In case that the duration of traversing the path covers both peak and off-peak hours, we split a path into two sub-paths, one for peak hour and one for off-peak hour. Without loss of generality, in the following discussions, we only consider the case for a single time interval.

Given a path, there can be more than a single combination of random variables that is able to compute the cost distribution of the path. For example, in Figure 3, consider the path $\mathcal{P}_i = \langle n_1n_4, n_4n_6 \rangle$. One possible way of deriving the cost distribution of $\mathcal{P}_i$ is to convolute the distribution of edge n_1n_4 and the distribution of edge n_4n_6, i.e., to convolute the two rank 1 random variables $\mathcal{RV}_1$ and $\mathcal{RV}_4$. Another way of deriving the cost distribution of $\mathcal{P}_i$ is to simply use random variable $\mathcal{RV}_2$ with rank 2. The two cost distributions obtained may not be the same. For instance, consider the following example based on Figure 3.

Assume that 200 trajectories go through $\langle n_1n_4\rangle, \langle n_4n_6\rangle$. Out of the 200 trajectories, there are 100 trajectories that traverse $\langle n_1n_4\rangle$ in 10s and then $\langle n_4n_6\rangle$ in 20s, while the other 100 trajectories traverse $\langle n_1n_4\rangle$ in 15s and then $\langle n_4n_6\rangle$ in 25s. Based on the available trajectories, we are able to instantiate two rank 1 random variables: $\mathcal{RV}_1 = \langle 10 : 0.5, 15 : 0.5\rangle$ for unit path $\langle n_1n_4\rangle$ and $\mathcal{RV}_2 = \langle 20 : 0.5, 25 : 0.5\rangle$ for unit path $\langle n_4n_6\rangle$. In addition, a rank 2 random variable $\mathcal{RV}_3 = \langle 30 : 0.5, 40 : 0.5\rangle$ can also be instantiated. However, the convolution of $\mathcal{RV}_1$ and $\mathcal{RV}_2$ produces a random variable $\langle 30 : 0.25, 35 : 0.5, 40 : 0.25\rangle$, that is different from $\mathcal{RV}_3$. The convoluted distribution assumes that the travel costs on the two edges are independent, while $\mathcal{RV}_3$ captures the dependencies between the two edges—if one drives fast (slow) on the first edge and then one is also likely to drive fast (slow) on the second edge.

Following a recent study [5], the distribution obtained from random variables with higher ranks are able to provide more accurate distributions than that derived from random variables with lower ranks. The intuition is that random variables with high ranks capture the cost dependencies among different road segments in the path, which considers various hard-to-formulate traffic factors (e.g., turn costs, waiting at traffic lights). A formal proof is offered in the same study [5].

Based on the above, we propose a method that is able to derive the most accurate cost distribution of a path. Before introducing the method, we define a **direct extension** relationship between two random variables. Assume that two random variables $\mathcal{RV}_i$ and $\mathcal{RV}_j$ represent the cost distributions of path $\mathcal{P}_i = \langle c_1, c_2, \ldots, c_m\rangle$ and path $\mathcal{P}_j = \langle e_1, e_2, \ldots, e_n\rangle$, respectively. Random variable $\mathcal{RV}_i$ is a direct extension of random variable $\mathcal{RV}_j$, if the following two conditions hold.

1. The cardinality of path $\mathcal{P}_i$ is one larger than the cardinality of path $\mathcal{P}_j$, i.e., $m = n+1$ and equivalently $|\mathcal{P}_i| = |\mathcal{P}_j|+1$.
2. The first n edges in path $\mathcal{P}_i$ are identical to the first n edges in path $\mathcal{P}_j$, i.e., $e_1 = c_1, \ldots, e_n = c_n$.

We use symbol $\rightarrow$ to express the direct extension relationship. In particular, $\mathcal{RV}_i{}^{\rightarrow\mathcal{RV}_j}$ denotes that $\mathcal{RV}_i$ is a direct extension of $\mathcal{RV}_j$. For example, in Figure 3, $\mathcal{RV}_2$ is a direct extension of $\mathcal{RV}_1$, denoted as $\mathcal{RV}_2{}^{\rightarrow\mathcal{RV}_1}$.

Consider a path $\mathcal{P}_i = \langle e_1, e_2, \ldots, e_n\rangle$. We aim at selecting the combination of random variables that is able to provide the most accurate travel cost distribution for the path. We call such a combination of random variables the path's *optimal random variable list*, denoted as $RVList_i$. We start considering the first segment e_1 and add its corresponding random variable, say $\mathcal{RV}_{e_1}$, to $RVList_i$. We then consider the next segment e_2. If a random variable $\mathcal{RV}^*$ exists that is a direct extension of $\mathcal{RV}_{e_1}$ and $\mathcal{RV}^*$'s corresponding path contain e_2, we remove $\mathcal{RV}_{e_1}$ from $RVList_i$ and add $\mathcal{RV}^*$ to $RVList_i$. On the other hand, if no such a random variable exists, we add e_2's corresponding rank 1 random variable to $RVList_i$. The procedure continues until the last edge e_n is checked. Finally, $RVList_i$ contains the combination of random variables with the highest ranks and whose corresponding paths cover the path $\mathcal{P}_i$. By convoluting the random variables in $RVList_i$, we achieve the cost distribution for path $\mathcal{P}_i$.

As an example, we consider obtaining the $RVList$ of $\mathcal{P}_i = \langle n_1n_4, n_4n_6, n_6n_7\rangle$ in Figure 2 using the hybrid graph in Figure 3. First we consider the first segment $\langle n_1n_4\rangle$ and add its corresponding random variable $\mathcal{RV}_1$ to $\mathcal{P}_i$'s optimal random variable list $RVList_i$. Next, we consider the segment $\langle n_4n_6\rangle$ and check the random variables associated with it. Since the random variable $\mathcal{RV}_2$ is a direct extension of the just added random variable, i.e., $\mathcal{RV}_1$, we remove $\mathcal{RV}_1$ from $RVList_i$ and add $\mathcal{RV}_2$ to $RVList_i$. Then, when we reach the segment $\langle n_6n_7\rangle$, following the same idea, we find $\mathcal{RV}_3{}^{\rightarrow\mathcal{RV}_2}$, thus, remove $\mathcal{RV}_2$ from $RVList_i$ and add $\mathcal{RV}_3$ to $RVList_i$. Finally, we end up with an optimal random variable list for path $\mathcal{P}_i$ only containing the random variable $\mathcal{RV}_3$.

3.3 Problem Statement

Before formalizing the non-dominated routing problem, we introduce the concept of **stochastic dominance** [12] between two random variables. We say that a random variable $\mathcal{RV}_i$ stochastically dominates a random variable $\mathcal{RV}_j$ such that $\mathcal{RV}_i \prec \mathcal{RV}_j$, if the following two conditions hold.

1. For every possible cost value c, we have $CDF_{\mathcal{RV}_i}(c) \geqslant CDF_{\mathcal{RV}_j}(c)$.
2. There exists a cost value c^*, such that $CDF_{\mathcal{RV}_i}(c^*) > CDF_{\mathcal{RV}_j}(c^*)$.

Assume that $\mathcal{RV}_i$ and $\mathcal{RV}_j$ represents the cost distributions of paths $\mathcal{P}_i$ and $\mathcal{P}_j$ that connect with the same source and destination. If $\mathcal{RV}_i$ stochastically dominates $\mathcal{RV}_j$, we are sure that,

1. Given any time budget, path $\mathcal{P}_i$ always has higher or the same probability of arriving within a time budget as does path $\mathcal{P}_j$.
2. A time budget exists such that path $\mathcal{P}_i$ gives higher probability of arriving within the same time budget than does path $\mathcal{P}_j$.

Thus, no user is interested in using path $\mathcal{P}_j$. This formalizes the intuition in the example in the introduction.

Based on the above, we can state the non-dominated path finding problem.

DEFINITION 1. ***Non-dominated path finding query:*** *Assume that we have a road network $\mathcal{N}$ along with its instantiated hybrid graph HG. Given a source s, a destination d, and a departure time t, the non-dominated path finding query returns the set of all non-dominated paths from s to d. A path $\mathcal{P}$ is a non-dominated path if no other path $\mathcal{P}'$ exists such that $\mathcal{P}'$'s cost distribution stochastically dominates $\mathcal{P}$'s cost distribution.*

4. FINDING NON-DOMINATED PATHS

In this section, we design algorithms for solving the non-dominated path finding problem.

4.1 Solution Framework

We design a general solution framework for solving the problem, which is used by all the proposed algorithms. Different algorithms use different methods to detect stochastic dominance between the cost distributions of paths. The framework is described as follows.

First, starting from the source node s, we perform a graph search towards the destination d using an A^* like approach. But instead of maintaining a priority queue of nodes, we maintain a priority queue of competitor paths. This priority queue is used to order paths in ascending order according to their expected travel cost.

Specifically, every possible path $\mathcal{P}$ from s to d is a competitor path as long as it is not dominated by another path $\mathcal{P}'$ from s to d. The top path in the priority queue, i.e., the path with the least expected cost, is popped up, and a new, unvisited node is added to the path leading to the creation of more paths for every segment emanating from the last node of the path. We call this process of creating more paths "path branching." This is because a path splits into as many branches as there are segments that emanate from a node. For example, in Figure 2, the search starts from s towards d. When segment $\langle sn_2 \rangle$ is added to path $\mathcal{P}_2$, path $\mathcal{P}_4$ is branched at n_2. Since $\mathcal{P}_4$ is branched from $\mathcal{P}_2$, the paths share segment $\langle sn_2 \rangle$.

As in a typical A^* search, we estimate the minimum cost from the last node of a partially explored path to the destination d, in order to direct the search and assign paths appropriate priorities in the priority queue. The estimated cost to d is the Euclidean distance from the last node in a path to d divided by the maximum speed limit in the road network. It is important to note that the minimum estimated total cost is only used to direct the graph search towards d. It does not replace the $\mathcal{RV}$ distributions when checking for dominance between competitor paths. Therefore, it does not affect the accuracy of the results. In addition, even if the minimum expected total cost of a path $\mathcal{P}_i$ is lower than that of $\mathcal{P}_j$ such that $\mathcal{P}_j$ is placed at a better position in the priority queue, this does not mean that $\mathcal{P}_i$ cannot dominate $\mathcal{P}_j$ when they are competing with each other.

The solution framework utilizes the *Hybrid Graph* to obtain the most accurate cost distributions for different paths. This is done by convoluting the random variables in a path's optimal random variable list, as discussed in in Section 3.2.

Next, we propose an early detection and termination approach of dominated paths as the search continues towards d. First, every node n_i keeps a list of non-dominated paths that arrive n_i, represented as $NonDom(n_i) = \{\mathcal{P}_1, \mathcal{P}_2, .., \mathcal{P}_k\}$. Second, when a new path $\mathcal{P}'$ arrives at n_i, we examine the dominance relationship between $\mathcal{P}'$ and the paths in $NonDom(n_i)$. If $\mathcal{P}'$ is dominated by any path in $NonDom(n_i)$, then $\mathcal{P}'$ is terminated and removed from the priority queue. However, if $\mathcal{P}'$ dominates a path $\mathcal{P}$ in $NonDom(n_i)$ then $\mathcal{P}$ is removed from the priority queue and from $NonDom(n_i)$. Further, we remove any other path branching from $\mathcal{P}$. We formalize the terminating branched paths in the following theorem.

THEOREM 1. *Let n_i be a node in road network $\mathcal{N}$, with a list of non-dominated paths $NonDom(n_i)$ including $\mathcal{P}$, i.e., $\mathcal{P} \in NonDom(n_i)$. Let paths $\{\mathcal{P}_j, \mathcal{P}_k, \ldots, \mathcal{P}_l\}$ be branched from $\mathcal{P}$ after passing n_i. Let $\mathcal{P}'$ be a new path to n_i and $\mathcal{P}'$ dominates $\mathcal{P}$ at n_i. We have that any path branching from $\mathcal{P}$ after passing n_i can be terminated and removed from the priority queue.*

PROOF. For paths in $\{\mathcal{P}_j, \mathcal{P}_k, \ldots, \mathcal{P}_l\}$ that are branched from $\mathcal{P}$ after passing n_i, we are able to compose a corresponding set of paths $\{\mathcal{P}'_j, \mathcal{P}'_k, \ldots, \mathcal{P}'_l\}$ such that they are branched from $\mathcal{P}'$ after passing n_i and have the same road segments as $\{\mathcal{P}_j, \mathcal{P}_k, \ldots, \mathcal{P}_l\}$ except for the part prior to reaching n_i.

Since $\mathcal{P}'$ dominates $\mathcal{P}$ at n_i and the distribution of the paths branched from both $\mathcal{P}$ and $\mathcal{P}'$ are the same after n_i, the paths in $\{\mathcal{P}'_j, \mathcal{P}'_k, \ldots, \mathcal{P}'_l\}$ dominate the paths in $\{\mathcal{P}_j, \mathcal{P}_k, \ldots, \mathcal{P}_l\}$. Therefore,the paths in $\{\mathcal{P}_j, \mathcal{P}_k, \ldots, \mathcal{P}_l\}$ can be safely removed from the priority queue. $\square$

As an example, in Figure 2, when the path $\mathcal{P}_2$ visits n_2, it branches and creates a new path $\mathcal{P}_4$. Similarly, assume the path $\mathcal{P}_4$ branches at n_4 ,creating a path (e.g., $\mathcal{P}_5 = \langle sn_2.n_2n_4, n_4n_6 \rangle$) and the node n_4 maintains a list of non-dominated paths $NonDom(n_4) = \{\mathcal{P}_4, \mathcal{P}_5\}$. Later, when path $\mathcal{P}_3$ visit n_4 and meets and dominates $\mathcal{P}_4$ ($\mathcal{P}_3 \prec \mathcal{P}_4$), path $\mathcal{P}_4$ is terminated and removed from the priority queue as well as all paths branched from it ($\mathcal{P}_5$) at the meeting node. This is because $\mathcal{P}_3$ will branch at n_4, creating a new path (e.g., $\mathcal{P}_6$) by adding the segment n_4n_6. Since $\mathcal{P}_3$ has a better cost when reaching n_4 than does $\mathcal{P}_4$, path $\mathcal{P}_4$ and its branches ($\mathcal{P}_5$) can be removed safely.

This approach reduces the number of the paths dramatically, as the paths are dominated by other paths. Thus, detecting them early prevents branching from them and creating additional non-competitive paths. However, we have not covered the process of examining the dominance between competitor paths. The details of this step are different in each algorithm considered next.

Before doing so, the pseudo of the general procedure is provided in Algorithm 1.

Algorithm 1: General Framework

```
Input: A road network $\mathcal{N}$, $s$, $d$, Hybrid graph $HG$,
       Dominance Algorithm $DA$
Output: A set of non-dominated paths
$PathsOpenList = Lowest\_Cost\_Priority\_Queue()$;
$First\_Path.add\_node(s)$;
$PathsOpenList.add(First\_Path)$;
// create paths to $d$
while $PathsOpenList.size > 0$ do
    $Cur\_path \leftarrow PathsOpenList.top()$;
    for $seg \in PathsOpenList.head.edges$ do
        $Curt\_path\_RVList.add(seg.\mathcal{RV}s, HG)$;
        $Cur\_node \leftarrow seg.End\_node()$;
        $Curt\_path\_nodes.add(Cur\_node)$;
        // Check if the path is dominated
        $Cur\_node.nonDom\_paths.check(Cur\_path, DA)$;

        if $Curt\_path$ is not dominated then
            $New\_path = Cur\_path.branch(Cur\_node)$;
            $PathsOpenList.add(New\_path)$;
```

4.2 Stochastic Dominance Checking Baselines

We offer two baseline algorithms for stochastic dominance checking.

4.2.1 Naive Dominance Detection Algorithm

A straightforward algorithm to check the dominance between paths works as follows. First, when two paths meet at a network node, we convolute the random variables in each path's optimal random variable list to get the cost distribution for each path. To increase the algorithm efficiency, we use fast fourier transformation (FFT) based convolution [19]. Next, we check the stochastic dominance relationship between the two paths' distributions by checking the cumulative probability of each cost value.

This baseline has two main limitations. First, it needs to convolute all random variables in a path's optimal random variable list every time a new segment is added to the path. Second, it needs to check every cost value in each path in order to check the dominance relationship between two paths when they meet at a node. The long time consumed by these two procedures is wasted if the checked path is dominated, which is the case for most of the possible paths between s and d.

For example in Figure 2, when the two paths $\mathcal{P}_3 = \langle sn_1, n_1n_4 \rangle$ and $\mathcal{P}_4 = \langle sn_2, n_2n_4 \rangle$ meet at n_4, we first convolute the random variables of each path. Next, we check their dominance relation-

Cost	$PMF(\mathcal{P}_3)$	$PMF(\mathcal{P}_4)$	$CDF(\mathcal{P}_3)$	$CDF(\mathcal{P}_4)$
5	0.7	0	0.7	0
6	0.3	0	1	00
7	0	0.5	1	0.5
8	0	0.2	1	0.7
10	0	0.3	1	1

Table 1: Naive Stochastic Dominance Checking Baseline at n_4

ship by checking the cumulative probability of each possible cost as shown in Table 1. We sort the cost values of both paths in ascending order and compare the cumulative distribution of each cost value. Path $\mathcal{P}_3$ dominates path $\mathcal{P}_4$ as the the stochastic distribution of $\mathcal{P}_3$ is equal or larger than that of $\mathcal{P}_4$.

Checking the dominance relationship using this baseline is easy task at the beginning of the graph search, as the number of random variables that need to be convoluted is small. However, as paths grow, the number of distribution cost values increases dramatically. Although this solution has accurate results, it is extremely slow the convolution is a very expensive operation even with the assistance of FFT. Classical convolution without using FFT has $O(n^2)$ complexity where n is the number of cost values, while FFT-based convolution has $O(n \log n)$ complexity. In addition, the computation of finding stochastic dominance between two distributions with large numbers of cost values is also expensive. Therefore, this solution is not feasible for large data sets.

4.2.2 *Histogram Approximation Based Dominance Detection Algorithm*

The second baseline employs histograms to approximate the distributions and thus improves efficiency. For a random variable with n cost values, it approximates the corresponding distribution by a histogram with m buckets, where m is much smaller than n. To make the histogram best approximate the distribution using the m buckets, we try to maintain a minimum error value between the distribution represented by the original random variable and the distribution represented by the corresponding histogram, e.g., using the V-optimal histogram algorithm [20]. In particular, we use *f-fold cross validation* [21] to ensure that that the optimal boundaries of the m buckets are identified such that the error value is minimized.

The time complexity of histogram based convolution is $O(m \log m)$, where m is the number of buckets, which is usually much smaller than the number of cost values n. Thus, the histogram approximation based baseline is much faster than the naive baseline, but the distribution obtained is less accurate.

For example in Figure 3, when the path $\mathcal{P}_3$ starts from s, we approximate the cost values of every $\mathcal{RV}$ that is added to the path. when $\mathcal{P}_3$ and $\mathcal{P}_4$ reach n_4, we have would several histogram buckets for each path, which are faster to convolute and to check for dominance. We check dominance between paths in the same way as the naive algorithm. But, in the histogram based baseline we deal with fewer of buckets compared to the number of cost values.

4.3 Extreme Values Based Dominance Checking

We propose an efficient algorithm to detect stochastic dominance between paths. Our solution is based on the observation that most dominated paths from s to d can be detected using the two extreme (i.e., minimum and maximum) cost values of each path. We propose an algorithm that utilizes these extreme values to avoid the expensive convolution and examination of all cumulative probability values, thus obtaining all non-dominated paths efficiently. We call this algorithm *ExVDom*. It has three main stages, where the first and the second stages are based on checking the extreme values. The first stage uses the extreme value technique to detect dominance between paths as they grow towards d. The second stage uses the first path that arrives at the destination d to obtain a threshold and uses its extreme values to check dominance with the currently partially-explored paths. In the last stage, we perform full convolution and stochastic dominance examination for paths that are not pruned by the first and second stages to obtain the final set of non-dominated paths.

We first define several concepts to help understand the algorithm. Assume that we have a path $\mathcal{P}_i$ and the random variables in its optimal random variable list. Recall that the cost values in each $\mathcal{RV}$ are sorted in ascending order. After we perform convolution on the $\mathcal{RV}s$, we obtain a sequence of cost values $\langle c_1, c_2, ..., c_n \rangle$, where c_1 is the minimum cost value and c_n is the maximum cost value for $\mathcal{P}_i$, represented as $\mathcal{P}_i min_c$ and $\mathcal{P}_i max_c$, respectively.

Each cost value c_i is associated with a probability value. Thus, we have a cost-probability pair $(cost : prob)$. A cost value c_i of a path $\mathcal{P}_i$, $\mathcal{P}_i c_i$, is said to be larger than another cost value c_j of another path $\mathcal{P}_j$, i.e., $\mathcal{P}_j c_j$, if the *cost* of c_i exceeds that of c_j; or they have the same cost, but the probability of c_i exceeds that of c_j, $prob(c_i) > prob(c_j)$.

The main idea of the algorithm is to maintain the optimal list of $\mathcal{RV}s$ for each path with sorted cost values in each $\mathcal{RV}$ and not to perform any convolution except for the minimum and maximum cost values of each $\mathcal{RV}$. In contrast to the traditional method of examining stochastic dominance, which requires the convolution of all cost values, *ExVDom* only convolutes the minimum and maximum cost values of each $\mathcal{RV}$ to detect obvious dominance between paths. The convolution done by *ExVDom* uses two cost values from each $\mathcal{RV}$ and results in two final extreme cost values. Before we discuss the details of this process, we state corollaries.

COROLLARY 2. *Given two paths $\mathcal{P}_i$ and $\mathcal{P}_j$ and their extreme cost values ($\mathcal{P}_i Min_c$, $\mathcal{P}_i Max_c$) and ($\mathcal{P}_j Min_c$, $\mathcal{P}_j Max_c$), if $\mathcal{P}_i Max_c \leqslant \mathcal{P}_j Min_c$ then $\mathcal{P}_i$ dominates $\mathcal{P}_j$.*

PROOF. When $\mathcal{P}_i Max_c \leqslant \mathcal{P}_j Min_c$, then $\mathcal{P}_i Max_c \leqslant \mathcal{P}_j c_i$, where c_i is any other cost value in $\mathcal{P}_j$. This means that $\mathcal{P}_i$ dominates $\mathcal{P}_j$ according to the definition of stochastic dominance. □

COROLLARY 3. *Given two paths $\mathcal{P}_i$ and $\mathcal{P}_j$ and their extreme cost values ($\mathcal{P}_i Min_c$, $\mathcal{P}_i Max_c$) and ($\mathcal{P}_j Min_c$, $\mathcal{P}_j Max_c$). if $\mathcal{P}_i Min_c < \mathcal{P}_j Min_c$ and $\mathcal{P}_i Max_c > \mathcal{P}_j Max_c$, then $\mathcal{P}_i$ does not dominate $\mathcal{P}_j$ and $\mathcal{P}_j$ does not dominate $\mathcal{P}_i$.*

PROOF. When $\mathcal{P}_i Min_c$ is less than that of $\mathcal{P}_j$ and $\mathcal{P}_i Max_c$ is larger than that of $\mathcal{P}_j$ then for all other cost values in $\mathcal{P}_j$, i.e., $\langle c_1, c_2, .., c_n \rangle$, they are larger than $\mathcal{P}_i Min_c$ and smaller than $\mathcal{P}_i Max_c$. This means that neither path can dominate the other. □

Based on the corollaries, we can guarantee that a path $\mathcal{P}_i$ dominates another path $\mathcal{P}_j$ if Corollary 2 is satisfied, and we can guarantee that the two paths do not dominate each other if Corollary 3 is satisfied.

Stage 1

The first stage of the algorithm proceeds as follows. Similar to the intuitive solution, it uses the solution framework and starts by traversing the graph towards the destination, maintaining an optimal list of high rank $\mathcal{RV}$s. However, it only performs convolution for the maximum and minimum cost values of each $\mathcal{RV}$. When two paths meet at a node, we check their dominance in terms of their extreme values according to Corollaries 2 and 3. The failure

Cost	$PMF(\mathcal{P}_1)$	$PMF(\mathcal{P}_2)$	$PMF(\mathcal{P}_3)$	$CDF(\mathcal{P}_1)$	$CDF(\mathcal{P}_2)$	$CDF(\mathcal{P}_3)$
30	0.4	0.2	0.1	0.4	0.2	0.1
40	0.2	0.5	0.1	0.6	0.7	0.2
50	0.2	0.3	0.4	0.8	1	0.6
60	0.1	0	0.2	0.9	1	0.8
70	00.1	0	0.2	1	1	1

Table 2: Stage 3 Stochastic Dominance

of satisfying Corollary 2 does not convey any dominance relationship between the two paths. Indeed, this means that the dominance relationship is as yet unknown. Therefore, they keep growing in the graph until they are dominated by the extreme values of other paths, or their dominance is examined in a latter stage.

As an example in Figure 2 and Table 1, the minimum and maximum cost values of $\mathcal{P}_3$ and $\mathcal{P}_4$ when they meet at node n_4 are ($\mathcal{P}_3 Min_c = 5 : 0.7$, $\mathcal{P}_3 Max_c = 6 : 0.3$) and ($\mathcal{P}_4 Min_c = 7 : 0.5$, $\mathcal{P}_4 Max_c = 10 : 0.3$) respectively. Since $\mathcal{P}_3 Max_c < \mathcal{P}_4 Min_c$, $\mathcal{P}_3$ dominates $\mathcal{P}_4$.

Stage 2

The second stage of *ExVDom* is to employ the first arriving path threshold. This stage is also based on the extreme values dominance technique, but it uses a different procedure. Let $\mathcal{P}_i$ be the first arriving path at destination d. Let $\mathcal{P}_j$ be the top path in the priority queue and assume that $\mathcal{P}_j$ arrives at n_i. Let $Escst(n_i \rightarrow d)$ be the estimated minimum cost from n_i to d, which can be easily estimated using the Euclidean distance $(\overline{d, n_i})$ between n_i and d divided by the maximum speed limit in $\mathcal{N}$. When we convolute $Escst(n_i \rightarrow d)$ with $\mathcal{P}_j Min_c$ we can get an estimate of the minimum cost value of $\mathcal{P}_j$ when it arrives at d and use the function $Es(\mathcal{P}_j Min_c)$ to obtain it. Since we have $Es(\mathcal{P}_j Min_c)$ and $\mathcal{P}_i max_c$ we can utilize the following corollary.

COROLLARY 4. *Assume that we re given $\mathcal{P}_i$ and $\mathcal{P}_j$, the extreme cost values of $\mathcal{P}_i$, i.e., ($\mathcal{P}_i Min_c$,$\mathcal{P}_i Max_c$), and the estimated minimum cost $Es(\mathcal{P}_j Min_c)$. If $\mathcal{P}_i Max_c \leqslant Es(\mathcal{P}_j Min_c)$, then $\mathcal{P}_i$ dominates $\mathcal{P}_j$ when $\mathcal{P}_j$ arrives at d. Therefore, $\mathcal{P}_j$ can be terminated early and removed.*

PROOF. Since $Escst(n_i \rightarrow d)$ is the minimum estimated cost to d, $Es(\mathcal{P}_j Min_c)$ is the minimum estimated cost value when $\mathcal{P}_j$ arrives at d, which is a lower bound on the actual minimum cost to d. When $\mathcal{P}_i Max_c \leqslant Es(\mathcal{P}_j Min_c)$, then $\mathcal{P}_i Max_c \leqslant Es(\mathcal{P}_j c_i)$, where c_i is any other cost value in $\mathcal{P}_j$. Since the estimated costs are lower bounds of actual costs, we have $\mathcal{P}_i Max_c \leqslant \mathcal{P}_j c_i$. Therefore, $\mathcal{P}_i$ dominates $\mathcal{P}_j$. $\square$

The procedure of obtaining the first arriving path threshold is as follows. First, when a path $\mathcal{P}_i$, arrives at d we use its maximum extreme cost value ($\mathcal{P}_i Max_c$) to create a threshold. Next, when the top path $\mathcal{P}_j$ in the priority queue pops up, we use the function $Es(\mathcal{P}_j Min_c)$ to estimate the minimum cost values of $\mathcal{P}_j$ when it arrives at d. We use Corollary 4 to check if $\mathcal{P}_j$ is dominated by $\mathcal{P}_i$. If $\mathcal{P}_i$ dominates $\mathcal{P}_j$, we terminate and remove $\mathcal{P}_j$ from the priority queue. Otherwise, we keep it for dominance checking in the last stage. We keep applying the process in this stage until the priority queue is empty, upon which we have a set of paths that arrive at d and that may be non-dominated paths. Final check up for non-dominated paths is done in the last stage.

For example, in Figure 2, when $\mathcal{P}_1$ arrives at d it creates the threshold with its maximum cost value (e.g., $\mathcal{P}_1 Max_c = 8 : 0.04$). Next, when $\mathcal{P}_3$ arrives at n_6, we compare it against the threshold path as follows. First, we measure the Euclidean distance $(\overline{n_6, d})$ (e.g., 6 km) and use the maximum speed limit in $\mathcal{N}$ (e.g., 100 km/hour) to estimate the minimum cost to the destination $Escst(n_6 \rightarrow d) = 3.6 : 1$. Next, we convolute $Escst(n_6 \rightarrow d)$ with the minimum cost value of $\mathcal{P}_3$ when it arrives at n_6 (e.g., $\mathcal{P}_3 Min_c = 6 : 0.1$) to obtain the final estimated cost $Es(\mathcal{P}_3 c_i$ of $\mathcal{P}_3$: $\mathcal{P}_3 Min_c \oplus Escst(n_i \rightarrow d) = 9.6 : 0.1$, where $\oplus$ denotes convolution. Finally, since $\mathcal{P}_1 Max_c < Es(\mathcal{P}_3 c_i)$, $\mathcal{P}_3$ is dominated by $\mathcal{P}_1$ and it can be terminated safely.

The First Arriving Path Threshold Improvement: We improve the pruning effectiveness of the second stage by improving the estimation of the minimum cost value of an open path (e.g., $\mathcal{P}_j$) when it arrives at d. We denoted this cost value by $Es(\mathcal{P}_j Min_c)$ in the previous section. Specifically, we replaced the Euclidean distance between the last node of the open path (e.g., n_i) and d, which was denoted as $\overline{d, n_i}$, by the real network distance. To obtain the network distance, we run a one-to-all shortest path query from the destination d; and for each network node n_i we save its distance to d. The use of network distance is more accurate than the use of Euclidean distance and thus improves the ability of the algorithm to detect more dominated paths. For example, in Figure 2, when we use the Euclidean distance and compare $\mathcal{P}_2$ against the threshold path $\mathcal{P}_1$ using the Euclidean distance, $\mathcal{P}_2$ may not be dominated. However, when we use the network distance (d, n_7) instead of $(\overline{d, n_7})$, the chance of $\mathcal{P}_2$ being dominated is higher.

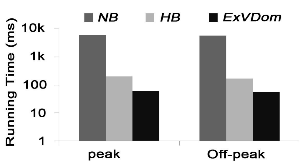
Stage 3

The last stage examines the stochastic dominance between paths that arrive at d and that are not pruned in stages 1 and 2. For each such path, we convolute the random variable in its optimal random variable list. Here, we also use Fast Fourier Transformation (FFT) based convolution to increase the efficiency. Next, we examine the dominance between each pair of the arriving paths using all their possible distributions, same as the naive baseline discussed in Section 4.2.1.

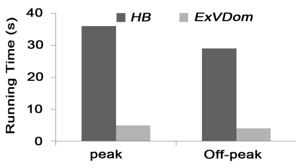
For example, in Table 2, we show the PMFs for $\mathcal{P}_1, \mathcal{P}_2$, and $\mathcal{P}_3$ and assign 0 for unavailable cost values for each path. Next, we create the CDF for each path and examine the dominance relationship. The cumulative probability value of $\mathcal{P}_1$ exceeds that of $\mathcal{P}_2$ at the cost value 30. However, the cumulative probability of $\mathcal{P}_2$ is larger for the remaining costs. Therefore, the paths $\mathcal{P}_1$ and $\mathcal{P}_2$ do not dominate each other. On the other hand, the cumulative probability of $\mathcal{P}_3$ is always lower than that of $\mathcal{P}_1$ and $\mathcal{P}_2$, thus, it is dominated as shown in Figure 1b. Finally, both $\mathcal{P}_1$ and $\mathcal{P}_2$ are returned as the non-dominated paths.

Summary:

To summarize, the *ExVDom* algorithm consists of three stages. In the first, we use an extreme values technique to examine paths dominance when they meet at intermediate nodes. In the second stage, we use the threshold of the first arriving path along with the ex-

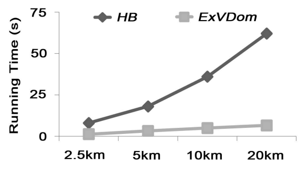


(a) N_2

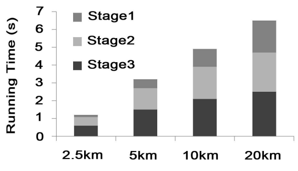


(b) N_1

Figure 4: Efficiency, effect of hybrid graphs

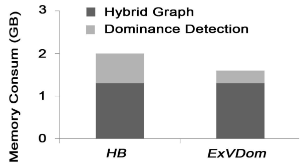


(a) In general

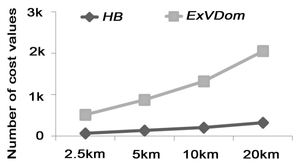


(b) At each stage

Figure 5: Efficiency, effect of path distance



(a) Memory Consumption



(b) Number of cost values

Figure 6

Parameter	Values
Distance between s and d	2.5km, 5km ,**10km**,20km
Hybrid graph	**peak**, off-peak

Table 3: Parameter Settings

treme value technique to examine dominance of paths in the priority queue. In the third stage, we use FFT based convolution to convolute each surviving path's random variables and examine their dominance relationships. The main advantage of *ExVDom* is that it prunes many paths in the first two stages and only performs full convolution for a few final arriving paths. This saves considerable computation, which would otherwise have been wasted performing convolution for dominated paths that are pruned in stages 1 and 2.

5. PERFORMANCE EVALUATION

We evaluate the performance of the three-stage *ExVDom* algorithm against the naive baseline (denoted as *NB*) and the histogram based baseline (denoted as *HB*, cf. Section 4.2). We evaluate the efficiency and effectiveness of ExVDom against both *NB* and *HB*.

5.1 Experimental Setup

Road Network: We use the road network of Aalborg, Denmark, obtained from Open StreetMap, which has 20,195 nodes and 41,276 segments with and average length of 250 m. We refer to this dataset as N_1. We extracted a smaller dataset N_2 (200 nodes) from the center of N_1 in order to evaluate the performance of the naive baseline.

Trajectories: We use a GPS data set containing 37 million GPS records with a sampling rate of 1 Hz (e.g., one record per second) from the same road network. We map match the GPS records using a well-know algorithm based on hidden Markov models [18].

Hybrid Graph: We instantiate the hybrid graph using the GPS records that occurred in peak and off-peak intervals, respectively, thus obtaining two hybrid graphs—one for peak hours and one for off-peak hours.

Parameters: We vary parameters as stated in Table 3, where default values are shown in **bold**. Specifically, we vary the Euclidean distance between s and d from 2.5 km to 20 km. Next, we perform experiments on the peak and off-peak hybrid graphs. We randomly generate 100 queries according to the parameters of each experiment setup. We consider travel time as the travel cost in the experiments, where the finest travel time unit is a second.

5.2 Efficiency Evaluation

5.2.1 Query Processing Time

We measure the query response time of *ExVDom* and the two baselines in varying settings. In addition, we also report the processing time taken by each stage of *ExVDom*.

Effect of hybrid graphs: We measure the run time required by each algorithm to process a query in the smaller road network N_2. Figure 4a shows that, *ExVDom* is two orders of magnitude faster than the *NB* algorithm and more than three times faster than the *HB* algorithm for both hybrid graphs. This confirms that the superiority of *ExVDom* over the naive algorithms.

Figure 4b illustrates the query processing time by *ExVDom* and *HB* for queries on the larger road network N_1. The query response time of *ExVDom* is more than seven times lower than that of the *HB* algorithm for both hybrid graphs. We do not include the results for *NB* because it takes prohibitively long time, which suggests that it is inapplicable to large road networks.

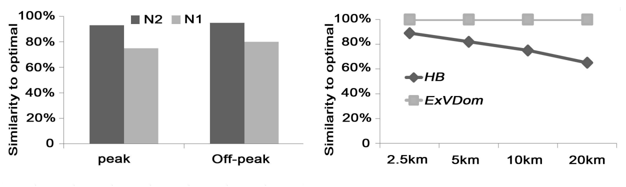
The response time of *ExVDom* is slightly higher for the peak hybrid graph than that of the off-peak hybrid graph, for both road networks. This is due to the complex cost distributions during peak hours resulting in larger differences between the minimum and maximum cost values in the optimal random variable list of each competitor path, thus making extreme value based pruning slightly less effective.

Effect of path distance: We vary the average distance between the source and the destination in N_1 from 2.5 km to 20 km. Figure 5a shows that *ExVDom* is more than an order of magnitude faster than *HB* at the 20 km distance queries. The processing time for both algorithms increases as the distance increases. This is because the size of the optimal random variable list associated with each competitor path increases, which results in more time being needed for convoluting the random variables in the list and for checking stochastic dominance. However, the time of *HB* increases more than that of *ExVDom* because *ExVDom* avoids full convolution when constructing paths.

Next, we report the average run time consumed at each stage of *ExVDom* while varying the distance between the source and the destination, in Figure 5b. As the distance increases, the processing time of the the first two stages increases accordingly. However, we observe a higher increase at the third stage as the distance increases. This is because longer paths maintain more random variables in their optimal random variable lists, which leads to more time required to convolute them and check their stochastic dominance at the third stage.

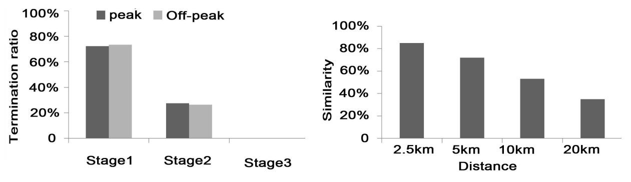
5.2.2 Memory Consumption

We compare the memory consumption of *ExVDom* and *HB*. In



(a) Effect of hybrid graphs (b) effect of distance

Figure 7: Paths accuracy



(a) Paths termination percentage (b) Paths Similarity

Figure 8: Effectiveness

addition, we measure the average number of cost values of the paths returned by both algorithms.

Memory consumption: Figure 6a shows the maximum total memory consumption of *HB* and *ExVDom*. As the figure shows, both algorithms allocate the same memory space for the hybrid graph since it is used by both algorithms. However, *ExVDom* consumes less memory than *HB* when detecting dominance between competitor paths. This is because *ExVDom* uses the extreme values pruning technique and it does not need to store the cost values convolution of each competitor path. On the other hand, *HB* maintains a list of approximated joint distributions of each competitor path, thus, requiring space to store their convolution.

Number of cost values: We compare the number of cost values returned by *ExVDom* and *HB* while varying the distance between the query points. Figure 6b shows that the number of cost values of *HB* is less than that for *ExVDom*. This is because *HB* approximates the many cost values into a limited number of histogram buckets and thus the number of buckets, is much lower than the number of actual cost values. This affects the precision of the cost distributions of the returned paths, thus yielding less accurate lists of non-dominated paths. Although *ExVDom* requires more space to store the convoluted paths at the third stage, the number of these paths is much smaller than the number of paths *HB* convolutes during the network search. Therefore *ExVDom* consumes less memory in general, as shown in Figure 6a.

5.3 Effectiveness Evaluation

We compare the returned non-dominated paths of both *ExVDom* and the baselines in both road networks, while varying the distances between source and destination. Next, we measure the percentage of terminated paths by each stage in *ExVDom*. Finally, we perform three experiments to examine the set non-dominated paths returned by *ExVDom* in terms of cardinality and similarity.

Effect of hybrid graphs: The paths returned by *ExVDom* and the naive baseline *NB* are identical because both compute the exact travel cost distribution, and they do not approximate travel cost distributions. In contrast, the histogram based baseline *HB* returns different paths due to the approximated distributions represented in histograms. In this sense, *ExVDom* and *NB* are considered as exact algorithms, while *HB* is considered as an approximate algorithm. Figure 7a shows the similarity percentage between the approximate algorithm *HB* and the exact algorithms (i.e., *ExVDom* or *NB*) on both road networks. Paths returned by *HB* are different for both road networks. This is due to the histogram approximation used by *HB*. This similarity can be as low as 75% for the peak hybrid graph in N_1. It can be seen, from Figure 7a, that the similarity percentage is slightly less for the peak hybrid graph compared to the off-peak hybrid graph. This is due to that the complex cost distributions during peak hours, resulting in larger approximation influence on the accuracy of the path cost distributions.

Effect of distance between query points: Figure 7b illustrates the similarity percentage between the *ExVDom* and *HB* algorithms for queried paths on road network N_1 using different distances between the source and the destination. The similarity between *HB* paths and *ExVDom* paths decreases as the distance increases. This is because the longer a path is, the more random variables there are in competitor paths' optimal random variable lists, and thus the less accurate the histogram approximation becomes.

Paths termination: Figure 8a shows the percentage of paths terminated at each stage of *ExVDom* for peak and off-peak hybrid graphs. It can be seen that, most of the paths are terminated at the first and second stages with most being terminated in the first stage. The paths terminated at the third stage after the full convolution and stochastic dominance check are less than 1%. However, as we discussed in Figure 5b the third stage consumes about 15% of the query processing time.

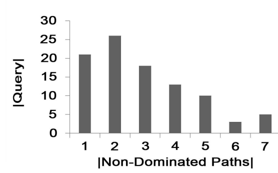
Cardinality of non-dominated paths: We generate 100 queries using the default settings and measure the cardinality of the returned non-dominated paths list. Figure 9a shows that, most of the queries return 1, 2, or 3 non-dominated paths. Only a few queries return more than 5 non-dominated paths, and the maximum number of non-dominated paths is 8.

Effect of distance on the cardinality of non-dominated paths: Using the default settings we generate 25 queries for each setting of the distance between s and d (2.5 km, 5 km, 10 km, 20 km) and measure the cardinality of the returned non-dominated paths list. Figure 9b shows that, when the distance is within 2.5 km, most of the queries returned less than 4 paths. As the distance increases the cardinality of returned paths increases. At the 20 km distance, the average cardinality of the returned paths is 5, and the cardinality can be up to 8.

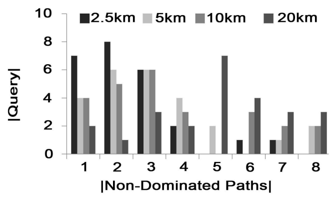
Similarity between non-dominated paths: We measure the similarity among the non-dominated paths if a query returns more than one non-dominated paths. Specifically, we compute the average similarity percentage for every possible pair of the returned non-dominated paths while varying the distance between s and d. Figure 8b shows that at the 2.5 km distance setup, the returned non-dominated paths are quite similar—they share many segments. However, as the distance increases, the similarity between non-dominated paths decreases, and they share only 40% of the segments.

6. CONCLUSIONS

We propose a new path finding problem called non-dominated path finding in uncertain road networks. The problem aims at finding a set of non-dominated paths in term of their cost distributions between a source and a destination. To solve the problem, we instantiate hybrid graphs using trajectories such that edges and some paths are associated with travel cost distributions. The hybrid graph enables accurate travel cost distribution estimation for any path in the given road network. We then introduce an A^* based solution framework that utilizes the hybrid graph to search the road network while capturing the most accurate cost distributions. This frame-



(a) General percentage



(b) Effect of Distance

Figure 9: Effectiveness

work maintains a priority queue of competitor paths and manage paths' branching at intersections to create more paths. We propose a three-stage dominance detection algorithm to be fit to the solution framework to early detect dominance between competitor paths. This algorithm is based on the extreme values dominance technique, where the two extreme (minimum and maximum) cost distributions of each competitor path are used to check their dominance. As shown by the experimental study, our algorithm outperforms both the naive and the HB baseline algorithms in two aspects. First, the time taken to return a set of non-dominated paths by our algorithm is an order of magnitude smaller than that of the histogram based baseline algorithm and two orders of magnitude smaller than that of the naive baseline algorithm. Second, while the naive baseline algorithm cannot be applied on large road networks, the paths returned by our algorithm are more than 35% more accurate than that of the histogram baseline algorithm.

Interesting future work includes investigation of the feasibility of a pre-computing phase in order to improve the algorithm and enable it to scale to larger road networks. It may also be possible to improve the process of choosing the right combination of $\mathcal{RV}s$ to improve the accuracy of the path cost. Finally, it is of interest to consider more time intervals and manage the overlapping between them.

Acknowledgment

Saad Aljubayrin is sponsored by Shaqra University, KSA. In addition this work is supported by Australian Research Council Discovery Project DP130104587, Future Fellowships Project FT120100832, the DiCyPS project, and a grant from the Obel Family Foundation.

7. REFERENCES

[1] S. Aljubayrin, Z. He, and R. Zhang. Skyline trips of multiple pois categories. In *DASFAA*, pages 189–206, 2015.

[2] S. Aljubayrin, J. Qi, C. S. Jensen, R. Zhang, Z. He, and Z. Wen. The safest path via safe zones. In *ICDE*, pages 531–542, 2015.

[3] O. Andersen, C. S. Jensen, K. Torp, and B. Yang. Ecotour: Reducing the environmental footprint of vehicles using eco-routes. In *MDM*, pages 338–340, 2013.

[4] J. Dai, B. Yang, C. Guo, and Z. Ding. Personalized route recommendation using big trajectory data. In *ICDE*, pages 543–554, 2015.

[5] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu. Path cost distribution estimation using trajectory data. *PVLDB*, 10(3), 2016.

[6] B. Ding, J. X. Yu, and L. Qin. Finding time-dependent shortest paths over large graphs. In *EDBT*, pages 205–216, 2008.

[7] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962.

[8] C. Guo, C. S. Jensen, and B. Yang. Towards total traffic awareness. *SIGMOD Record*, 43(3):18–23, 2014.

[9] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul. Ecomark: evaluating models of vehicular environmental impact. In *SIGSPATIAL GIS*, pages 269–278, 2012.

[10] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp. Ecomark 2.0: empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data. *GeoInformatica*, 19(3):567–599, 2015.

[11] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp. Ecosky: Reducing vehicular environmental impact through eco-routing. In *ICDE*, pages 1412–1415, 2015.

[12] J. Hadar and W. R. Russell. Rules for ordering uncertain prospects. *The American Economic Review*, 59(1):25–34, 1969.

[13] Z. Ji. Path finding under uncertainty. *Journal of advanced transportation*, 39(1):19–37, 2005.

[14] M. Kaul, B. Yang, and C. S. Jensen. Building accurate 3d spatial networks to enable next generation intelligent transportation systems. In *MDM*, pages 137–146, 2013.

[15] H.-P. Kriegel, M. Renz, and M. Schubert. Route skyline queries: A multi-preference path planning approach. In *ICDE*, pages 261–272, 2010.

[16] C. Li, Y. Gu, J. Qi, R. Zhang, and G. Yu. A safe region based approach to moving KNN queries in obstructed space. *Knowl. Inf. Syst.*, 45(2):417–451, 2015.

[17] S. Lim, C. Sommer, E. Nikolova, and D. Rus. Practical route planning under delay uncertainty: Stochastic shortest path queries. *Robotics: Science and Systems*, 8(32):249–256, 2013.

[18] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *ACM SIGSPATIAL*, pages 336–343, 2009.

[19] H. J. Nussbaumer. *Fast Fourier transform and convolution algorithms*, volume 2. Springer Science & Business Media, 2012.

[20] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. In *SIGMOD*, volume 25, pages 294–305, 1996.

[21] P. Smyth. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and computing*, 10(1):63–72, 2000.

[22] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *SIGKDD*, pages 25–34, 2014.

[23] B. Yang, C. Guo, and C. S. Jensen. Travel cost inference from sparse, spatio temporally correlated time series using markov models. *VLDB*, 6(9):769–780, 2013.

[24] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang. Stochastic skyline route planning under time-varying uncertainty. In *ICDE*, pages 136–147, 2014.

[25] B. Yang, C. Guo, Y. Ma, and C. S. Jensen. Toward personalized, context-aware routing. *VLDB J.*, 24(2):297–318, 2015.

[26] B. Yang, M. Kaul, and C. S. Jensen. Using incomplete information for complete weight annotation of road networks. *IEEE TKDE*, 26(5):1267–1279, 2014.

[27] J. Zheng and L. M.-S. Ni. Time-dependent trajectory regression on road networks via multi-task learning. In *AAAI*, page 1048, 2013.