ABSTRACT

With the growing volumes of vehicle trajectory data, it becomes increasingly possible to capture time-varying and uncertain travel costs in a road network, including travel time and fuel consumption. The current paradigm represents a road network as a weighted graph; it blasts trajectories into small fragments that fit the underlying edges to assign weights to edges; and it then applies a routing algorithm to the resulting graph. We propose a new paradigm, the hybrid graph, that targets more accurate and more efficient path cost distribution estimation. The new paradigm avoids blasting trajectories into small fragments and instead assigns weights to paths rather than simply to the edges.

We show how to compute path weights using trajectory data while taking into account the travel cost dependencies among the edges in the paths. Given a departure time and a query path, we show how to select an optimal set of weights with associated paths that cover the query path and such that the weights enable the most accurate joint cost distribution estimation for the query path. The cost distribution of the query path is then computed accurately using the joint distribution. Finally, we show how the resulting method for computing cost distributions of paths can be integrated into existing routing algorithms. Empirical studies with substantial trajectory data from two different cities offer insight into the design properties of the proposed method and confirm that the method is effective in real-world settings.

1. INTRODUCTION

Increasing volumes of vehicle trajectories are becoming available that contain detailed traffic information. It is of interest to exploit this data source as well as possible to understand the state of a road network [1, 4]. For instance, it is of interest to know the distributions of travel costs (e.g., travel times or greenhouse gas (GHG) emissions [2, 3]) of paths at a given departure time in order to plan travel or to calculate payments for transportation, e.g., in settings where such services are outsourced.

Consider a scenario where a person wants to reach the airport within 60 min to catch a flight. Figure 1(a) shows the travel time distributions for two alternative paths for a given departure time. If only considering averages, $P_2$ (with mean 51.5 min) is better than $P_1$ (with mean 52 min). However, $P_1$ is preferable because the probability of arriving at the airport within 60 min is 1, while with $P_2$, the probability is 0.9. The example illustrates why being able to compute a distribution rather than a mean is important.

13, 22]. However, the results are only accurate if the edge distributions are independent. We offer evidence that this is generally

Figure 1: Motivating Examples

A natural question is then how to best utilize trajectories to accurately and efficiently derive cost distribution for any path at a given departure time. This is the fundamental problem addressed in this paper. Solving this problem is important in its own right, and it also represents a significant step towards more accurate and efficient stochastic routing. To solve the problem, three challenges must be addressed.

Complex travel cost distributions: Travel time and GHG emissions vary over time and even vary across vehicles traversing the same path at the same time. To exemplify the latter, the bars in Figure 1(b) represent the travel time, derived from GPS trajectories, of a path during the time interval [8:00, 8:30]. Next, distributions do not follow standard distributions. Figure 1(b) shows the Gaussian [17], gamma [20], and exponential [20] distributions obtained using maximum likelihood estimation, illustrating that the cost distribution does not follow any of these standard distributions.

Sparseness: With enough trajectories that contain a path during a particular time interval, we could derive a distribution during the interval for the path using those trajectories. However, we report on analyses showing that even with large volumes of trajectory data, it is practically impossible to cover all paths in a road network with sufficient numbers of trajectories during all time intervals—a road network has a very large number of meaningful paths. We must thus contend with data sparseness.

Dependency: The cost distribution of a path can be estimated by summing up, or convoluting, the cost distributions of its edges [6, 13, 22]. However, the results are only accurate if the edge distributions are independent. We offer evidence that this is generally
not so in our setting. To derive accurate distributions for paths, the dependencies must be considered. The conventional paradigm for path cost estimation fragments trajectories into pieces that fit the individual edges to assign uncertain weights to the edges. Convolution is applied to the uncertain edge weights to compute the cost distribution of the path [6, 11, 13, 22, 26, 28, 29]. This approach falls short in addressing the above challenges and suffers in terms of both accuracy and efficiency. The accuracy suffers because dependencies among different edges are not accounted for. The efficiency suffers because many expensive convolutions must be performed.

We propose part of the foundation for a new paradigm, the hybrid graph, that aims to achieve better accuracy and efficiency by addressing the three challenges. In the hybrid graph, weights are assigned to paths; and path weights are joint distributions that fully capture the cost dependencies among the edges in the paths. Further, multi-dimensional histograms are used to approximate complex distributions accurately and compactly. Given a departure time and a query path, we are then able to select an optimal set of path weights such that the selected paths together cover the query path at the departure time and such that the path weights produce the most accurate joint distribution of the edges in the query path. The joint distribution can then be transferred into the cost distribution of the query path. The capability of deriving the distributions of long paths with insufficient trajectories from the distributions of carefully selected sub-paths with sufficient trajectories addresses the sparseness problem.

The paper’s proposal is compatible with existing stochastic routing algorithms and we show how the proposal can be seamlessly integrated into existing stochastic routing algorithms while boosting the efficiency and accuracy of these algorithms.

To the best of our knowledge, this is the first study to enable accurate and efficient path travel cost distribution estimation using trajectories while contending with sparseness, dependency, and distribution complexity. In particular, we make four contributions. (i) We propose the hybrid graph and a multi-dimensional histogram based method to instantiate path weights using trajectories. (ii) We propose an algorithm that identifies an optimal set of path weights, enabling accurate estimation of the joint distribution of a query path. (iii) We propose a method that derives the cost distribution of the query path, represented as a one-dimensional histogram, from a joint distribution, represented as a multi-dimensional histogram.

(a) Road Network
(b) Trajectories

Figure 2: A Road Network and Trajectories

2. PRELIMINARIES

2.1 Basic Concepts

A road network is modeled as a directed graph \( G = (V, E) \), where \( V \) is a vertex set and \( E \subseteq V \times V \) is an edge set. A vertex \( v_i \in V \) represents a road intersection or an end of a road. An edge \( e_a = (v_i, v_j) \in E \) models a directed road segment, indicating that travel is possible from its start vertex \( v_i \) to its end vertex \( v_j \). We use \( e_a.s \) and \( e_a.d \) to denote the start and end vertices of edge \( e_a \). Two edges are adjacent if one edge’s end vertex is the same as the other edge’s start vertex. Figure 2(a) shows an example road network.

A path \( P = \langle e_1, e_2, \ldots, e_A \rangle \), \( A \geq 1 \), is a sequence of adjacent edges that connect distinct vertices in the graph, where \( e_i \in E \), and \( e_i.d = e_{i+1}.s \) for \( 1 \leq i < A \), and the vertices \( e_1.s, e_2.s, \ldots, e_A.s \), and \( e_A.d \) are distinct. The cardinality of path \( P \), denoted as \( |P| \), is the number of edges in the path. Path \( P' = \langle g_1, g_2, \ldots, g_k \rangle \) is a sub-path of \( P = \langle e_1, e_2, \ldots, e_A \rangle \) if \( |P'| \leq |P| \) and there exists an edge sequence in \( P \) such that \( g_1 = e_i \), \( g_2 = e_{i+1} \), . . . , and \( g_k = e_{i+k-1} \).

Given two paths \( P_1 \) and \( P_2 \), we use \( P_1 \cap P_2 \) to denote the path that is shared by both paths, and we use \( P_1 \setminus P_2 \) to denote the sub-path of \( P_1 \) that exclude edges in \( P_2 \). For instance, if we have \( \langle e_1, e_2, e_3 \rangle \cap \langle e_2, e_3, e_4 \rangle = \langle e_2, e_3 \rangle \) and \( \langle e_1, e_2, e_3 \rangle \setminus \langle e_2, e_3, e_4 \rangle = \langle e_1 \rangle \).

A trajectory \( T = (p_1, p_2, \ldots, p_C) \) is a sequence of GPS records pertaining to a trip, where each \( p_i \) is a \( \{location, time\} \) pair of a vehicle, where \( p_i.time \leq p_{i+1}.time \) if \( 1 \leq i < j \leq C \). Map matching [16] is able to map GPS records in a trajectory \( T \) to specific locations on different edge and thus it aligns trajectory \( T \) with a path. We call this path as the path of trajectory \( T \), denoted as \( P_T \). A trajectory \( T \) occurred on path \( P \) at time \( t \) if and only if path \( P \) is a sub-path of the path of trajectory \( P_T \) and the first GPS record in the first edge in path \( P \) is obtained at \( t \). Figure 2(b) shows 10 trajectories. For example, trajectory \( T_{10} \) occurred on path \( \langle e_1, e_2, e_3, e_4 \rangle \) at 8:01.

The travel cost (e.g., travel time or GHG emissions) of using a path \( P \) can be obtained from the trajectories that occurred on \( P \). Given a trajectory \( T \) that occurred on path \( P \) at \( t \), the travel time of using \( P \) at \( t \) is the difference between the time of the last GPS record and the time of the first GPS record on path \( P \); and the GHG emissions of using \( P \) at \( t \) can be computed from the speeds and accelerations when traversing \( P \), which can be derived from the GPS records on path \( P \), and road grades that are available in 3D road networks [24], using vehicular environmental impact models [8,9].

Problem Definition: Given a large collection of trajectories \( T \) that occurred in a road network \( G \), travel cost distribution estimation takes as input a path \( P \) in \( G \) and a departure time \( t \), and accurately and efficiently estimates the travel cost distribution of traversing path \( P \) at \( t \). The output is a univariate random variable that describes the distribution of the cost of traversing path \( P \) at \( t \).

2.2 Accuracy-Optimal Cost Estimation

The most accurate way of estimating the travel cost distribution of path \( P \) at time \( t \) is to employ a sizable set of qualified trajectories. A trajectory \( T \) is qualified if \( T \) occurred on \( P \) at \( t \) and the difference between \( t' \) and \( t \) is less than a threshold, e.g., 30 minutes. For instance, if we want to estimate the travel cost distribution
of path \(\langle e_2, e_3, e_4 \rangle\) at 8:05, \(T_1, T_2, T_5,\) and \(T_6\) are qualified trajectories, but not \(T_7\) (cf. Figure 2).

A qualified trajectory captures traffic conditions (e.g., the time it takes to pass intersections, wait at traffic lights, and make turns at intersections) of the entire path \(P\) during the interval of interest. Thus, no explicitly modeling of complex traffic conditions at intersections is needed. To ensure an accurate estimation for a path \(P\) at \(t\) and not to overfit to the cost values of a few trajectories, we require the use of more than \(\beta\) qualified trajectories. The effect of parameter \(\beta\) is studied empirically in Section 5.

We regard this method as an accuracy-optimal baseline, and we let the resulting distribution \(D_{GT}(P, t)\) serve the role of a ground truth distribution in our proposal since \(D_{GT}(P, t)\) is the most accurate cost distribution computed from available trajectories.

However, this baseline is not always a practical approach, as it is very often inapplicable due to data sparseness. Figure 3 shows that the maximum number of trajectories that occurred on a path decreases rapidly as the cardinality of the path increases, based on two large trajectory collections from Aalborg and Beijing, with no time constraint applied.

![Data Sparseness Problem](a) Aalborg, Denmark | (b) Beijing, China

**Figure 3: Data Sparseness Problem**

### 2.3 Legacy Graph Model

To contend with the aforementioned data sparseness, existing stochastic route planning use a graph model that operates at edge granularity [6, 13, 17, 22]. The model is a weighted graph \(G = (V, E, W_E)\) with an edge weight function \(W_E : E \times T \rightarrow RV\), where \(T\) is the time domain of a day and \(RV\) denotes a set of random variables. The weight function takes as input an edge \(e \in E\) and a time \(t \in T\) and returns a random variable that represents the travel cost distribution of traversing \(e\) at \(t\).

A recent study [22] instantiates the edge weight function \(W_E\) using the accuracy-optimal baseline on each individual edge, where sparseness is not likely to be a significant problem. Next, independence is assumed so that a path’s travel cost distribution is the convolution of the travel costs distributions of the edges in the path. We denote the resulting distribution by \(D_{LB}(P, t) = \odot_{e_i \in P} W_E(e_i, t_{e_i})\), where \(\odot\) denotes the convolution of two distributions and \(W_E(e_i, t_{e_i})\) denotes the travel cost distribution of edge \(e_i\) at \(t_{e_i}\). \(t_{e_i}\) is the arrival time on edge \(e_i\), which may be different from the departure time \(t_{e_i}\) and needs to be progressively updated according to the travel times of \(e_i\)’s predecessor edges [22].

To examine the effect of dependence on the accuracy of the result of convolution, we consider 500 paths that each consists of two adjacent edges (i.e., with path cardinality being 2) and on which at least 100 trajectories occurred during [7:30, 8:00]. For each path \(P\), we compute the distribution \(D_{GT}\) using the accuracy-optimal baseline and distribution \(D_{LB}\) using the legacy baseline. If the distributions of the two edges in a path are independent, \(D_{GT}\) and \(D_{LB}\) should be identical. To see if this holds, we compute the KL-divergence of \(D_{LB}\) from \(D_{GT}\), denoted as \(KL(D_{GT}, D_{LB})\). The larger the KL-divergence, the more different the two distributions are, meaning that the convoluted distribution is less accurate. Figure 4(a) suggests that most of the adjacent edges are not independent.

![Figure 4](a) KL(D_{GT}, D_{LB}) | (b) Varying |P|

**Figure 4: Examining the Independence Assumption**

Next, we conduct an experiment on 100 paths with different cardinalities and where each path has at least 30 qualifying trajectories during an interval. We compute \(D_{GT}\) and \(D_{LB}\) for each path and then compute the average KL-divergence values between the two distributions for paths with varying cardinality. Figure 4(b) suggests that the more edges a path has, the more different the convoluted distribution \(D_{LB}\) is from the ground truth distribution \(D_{GT}\). Hence, the legacy graph model is likely to yield inaccurate travel cost distributions, especially for long paths.

### 3. HYBRID GRAPH

The analyses of the legacy graph model suggest that the independence assumption does not always hold and that explicitly modeling of the dependency among the travel costs of different edges in a path is needed to achieve accurate results. This motivates us that when computing the cost distribution for a path, we should try to use trajectories that occurred on long sub-paths of the path because they capture the cost dependencies among different edges.

To this end, we propose a novel model—the **hybrid graph** model \(G = (V, E, W_P)\). Instead of having an edge weight function \(W_E\) in the legacy graph model, the hybrid graph maintains a path weight function \(W_P : Paths \times T \rightarrow RV\), where \(Paths\) is a set of paths. Specifically, the path weight function \(W_P\) takes as input a path \(P\) and a time \(t\) and returns a multi-variate random variable that represents the joint distribution of path \(P\)’s edges’ travel costs. The joint distribution fully captures the dependency among the travel costs of different edges in path \(P\). We proceed to describe how to instantiate \(W_P\) using trajectories.

#### 3.1 Instantiating \(W_P\) for Unit Paths

A **unit path** consists of a single edge. We partition a day into a few intervals, where parameter \(\alpha\) specifies the finest-granularity interval of interest in minutes, e.g., 30 minutes. We let \(V_{(e_i)} = p(\alpha)\) denote a random variable that describes the travel cost distribution on unit path \(\langle e_i \rangle\) during interval \(I_j\).

To derive the distribution of \(V_{(e_i)}\), a set of qualified trajectories that occurred on \(\langle e_i \rangle\) at \(t\) where \(t \in I_j\) is obtained. If the trajectory set cardinality exceeds threshold \(\beta\), the same parameter used in accuracy-optimal baseline in Section 2.2, the travel cost values obtained from the qualified trajectories are employed to instantiate the distribution of \(V_{(e_i)}\), which is the ground truth distribution.
If the set cardinality does not exceed $\beta$, the distribution of $V^{f_i}_b$ is derived from the speed limit of edge $e_i$ to avoid overfitting to the limited number of travel costs. We also regard this as the ground truth distribution, since based on the available trajectories, we cannot get a more accurate distribution for the unit path during the interval. Thus, in both cases, $V^{f_i}_b$ represents the ground-truth distribution of unit path $(e_i)$ during interval $I_j$.

**Representing Univariate Distributions** We proceed to discuss how to represent a distribution when having more than $\beta$ trajectories. We represent distributions by histograms because they enable compact approximation of arbitrary, complex distributions. Gaussian mixture models are also able to represent non-standard travel cost distributions [21, 22], but are less compact. In particular, a one-dimensional histogram is employed to represent a univariate distribution.

From the qualified trajectories, we are able to obtain a multiset of cost values of the form $(\text{cost, perc})$, representing that perc percentage of the qualified trajectories took cost value $\text{cost}$. We call this a raw cost distribution. A histogram then approximates the raw cost distribution as a set of pairs: $\{(b_u, pr_u)\}$. A bucket $b_u = [l, u)$ is a range of travel costs, and $pr_u$ is the probability that the travel cost is in the range, and it holds that $\sum pr_i = 1$.

Given the number of buckets $b$, existing techniques, e.g., V-Optimal [12], are able to optimally derive a histogram based on a raw cost distribution such that the sum of errors between the derived histogram and the raw cost distribution is minimized. However, selecting a global value for $b$ is non-trivial because the traffic on different edges, and even the traffic on the same edge during different intervals, may differ significantly. A self-tuning method is desired so that more buckets are used for edges or intervals with more complex traffic conditions.

To this end, we propose a simple yet effective approach to automatically identify the number of buckets. The procedure starts with $b = 1$, i.e., using only one bucket, and computes an error value $E_b$. Next, it incrementally increases $b$ by 1 and computes a new error value $E_b$. Obviously, as the number of buckets increases, the error value keeps decreasing. However, the error values often initially drop quickly, but then subsequently drop only slowly. Based on this, the process stops when the error value of using $b$ does not lead to a significant decrease compared to the error value of using $b = 1$. Then, $b - 1$ is chosen as the bucket number. This yields a compact and accurate representation of the raw data distribution.

The error value $E_b$ of using $b$ buckets is computed using $f$-fold cross validation [18]. First, the multi-set of cost values is randomly split into $f$ equal-sized partitions. Each time, we reserve the cost values in one partition, say the $k$-th partition, and use the cost values in the remaining $f-1$ partitions to generate a histogram with $b$ buckets using V-Optimal, denoted as $H^b_k = \{(b_u, pr_u)\}$. Next, we compute the raw data distribution of the cost values in the reserved partition, denoted as $D^k = \{(\text{cost}_i, \text{perc}_i)\}$. After that, we compute the squared error between $H^b_k$ and $D^k$: $SE(H^b_k, D^k) = \sum_{c\in\text{cost}_i} (H^b_k[c] - D^k[c])^2$. We repeat the procedure $f$ times—once for each partition. The error value of using $b$ buckets, i.e., $E_b$, is the average of the $f$ squared errors.

Take the data in Figure 1(b) as an example. Figure 5(a) shows how the error $E_b$ decreases as the number of buckets $b$ increases. First, $E_b$ decreases sharply and then slowly (i.e., when $b > 4$). Figure 5(b) shows the histogram using $b = 4$ buckets and the original raw data distribution.

**3.2 Instantiating $W_{e_i}$ for Non-Unit Paths**

Based on the distributions of unit paths, we employ a bottom-up procedure to derive joint distributions of non-unit paths, i.e., paths with cardinalities more than one. In particular, the joint distributions of paths with cardinalities $k, k \geq 2$, are computed based on the joint distributions of paths with cardinalities $k - 1$.

Given two paths $P_i$ and $P_j$ with cardinalities $k - 1$, if they share $k - 2$ edges and can be combined into a valid path $P = (e_1, e_2, \ldots, e_k)$ with cardinality $k$, we check if a time interval $I_j$ exists during which more than $\beta$ qualified trajectories occurred on path $P$. If so, a random variable $V^{f_i}_b$ is instantiated based on the qualified trajectories. The travel cost distribution $p(e_1, \ldots, e_k)$ is a ground-truth joint distribution on the $k$ variables $e_1, \ldots, e_k$. This procedure continues until longer paths cannot be obtained.

For example, consider two unit paths $P_a = (e_a)$ and $P_b = (e_b)$, which can be combined into a valid path $P = (e_a, e_b)$. Figure 6(a) shows a raw joint distribution. Point A indicates that 110 trajectories passed $e_a$ with cost 50 s and then $e_b$ with cost 80 s.

**Figure 5: Identifying the Number of Buckets**

**Figure 6: An Example of Multiple Dimensional Histogram**

We use multi-dimensional histograms to describe joint distributions, where a dimension corresponds to the cost of an edge. A multi-dimensional histogram is a set of hyper-bucket and probability pairs: $\{(b_h, pr_h)\}$. A hyper-bucket $b_h = \{b_u^1, \ldots, b_u^n\}$ consists of $n$ buckets that each corresponds to one dimension. Value $pr_h$ equals the probability that the travel costs on multiple edges are in hyper-bucket $b_h$, and it holds that $\sum pr_i = 1$.

To derive a multi-dimensional histogram, we automatically identify the optimal number of buckets for each dimension using the method from Section 3.1. Next, we employ V-optimal to identify the optimal bucket boundaries on each dimension and thus obtain a set of hyper-buckets. Finally, we compute the probability for each hyper-bucket. For example, Figure 6(b) shows a 2-dimensional histogram that corresponds to the joint distribution shown in Figure 6(a). The dimension for $c_{e_a}$ is partitioned into 3 buckets and the dimension for $c_{e_b}$ is partitioned into 2 buckets, yielding 6 hyper-buckets in the 2-dimensional histogram.
3.3 Path Weight Function \( W_P \)

We let \( \mathcal{C}_P \) be a vector of variables \((c_{i1}, c_{i2}, \ldots, c_{ik})\) that corresponds to path \( P = (e_1, e_2, \ldots, e_k) \). We use \( p(\mathcal{C}_P) = p(c_{i1}, c_{i2}, \ldots, c_{ik}) \) to denote the ground-truth joint distribution of using path \( P \) at time \( t \). Following the same idea in Section 2.2, we regard the joint distribution obtained from at least \( \beta \) qualified trajectories as the ground-truth joint distribution.

Given a set of trajectories, we let \( \text{Paths}_{\beta} \) be a set of non-unit paths where each path has at least \( \beta \) qualified trajectories. Note that given different sets of trajectories that occurred on a same road network, \( \text{Paths}_{\beta} \) may contain different non-unit paths.

So far, we are able to instantiate path weight function \( W_P \) for non-unit paths in \( \text{Paths}_{\beta} \) and all unit paths. Specifically, given a path \( P \) and a time \( t \in I_j \), the path weight function \( W_P(P, t) \) returns random variable \( V^P_{ij} \) that represents the travel cost distribution of traversing path \( P \) at \( t \). Since each random variable \( V^P_{ij} \) is obtained by at least \( \beta \) qualified trajectories (or speed limits for some unit paths), they also correspond to ground-truth distributions. Thus, we have

\[
W_P(P, t) = V^P_{ij} = p(\mathcal{C}_P) =
\begin{cases}
  p(c_{i}), & P = (e_i) \text{ is a unit path; } \\
  p(c_{i1}, c_{i2}, \ldots, c_{ik}), & P = (e_i, e_{i2}, \ldots, e_{ik}) \in \text{Paths}_{\beta};
\end{cases}
\]

where (i) if \( P \) is a unit path, the path weight function returns the ground-truth distribution \( p(\mathcal{C}_P) = p(c_{i}) \), represented as a one-dimensional histogram; (ii) if \( P \) is a non-unit path, the path weight function returns the ground-truth joint distribution \( p(\mathcal{C}_P) = p(c_{i1}, c_{i2}, \ldots, c_{ik}) \), represented as a multi-dimensional histogram.

The random variables \( V^P_{ij} \) that are maintained in the path weight function \( W_P \) are called instantiated random variables. The rank of a variable \( V^P_{ij} \) is the cardinality of its path \( |P| \). In the legacy graph model, only random variables with rank one are considered, and they cannot capture distribution dependencies among edges, which are found in trajectories. In contrast, in the proposed hybrid graph model, the random variables with rank larger than one fully capture the distribution dependencies among the edges in a path.

4. GETTING PATH COST DISTRIBUTION

Given any path \( P \) and a departure time \( t \), we perform travel cost distribution estimation using the instantiated hybrid graph, in two steps. First, the joint distribution of path \( P \), which models the travel cost dependency among edges in \( C_P \), is computed. Second, the cost distribution of path \( P \), which captures the cost distribution of traversing the whole path \( P \), is derived based on the path’s joint distribution.

4.1 The Joint Distribution of a Path

The ground truth joint distribution of path \( P = (e_1, e_2, \ldots, e_n) \) at \( t \) is denoted as \( p(\mathcal{C}_P) = p(c_{i1}, c_{i2}, \ldots, c_{in}) \), where \( c_{in} (1 \leq i \leq n) \) is a random variable representing the travel cost distribution of path \( e_i \).

Given a path \( P \) and a departure time \( t \), we aim at estimating its most accurate joint distribution. If we are lucky, the path weight function \( W_P(P, t) \) returns \( V^P_{ij} \), where \( t \in I \). Since \( V^P_{ij} \) corresponds to the ground truth joint distribution \( p(\mathcal{C}_P) \), we are done.

In contrast, if \( W_P(P, t) \) returns an empty result, this means that there does not exist at least \( \beta \) qualified trajectories on path \( P \) around \( t \), so it is impossible to obtain its ground truth distribution. This case occurs often due to the data sparseness problem, as shown in Figure 3, especially for long paths.

To handle this unlucky but common case, we proceed to propose a method that is able to derive an accurate, estimated joint distribution \( p(c_{i1}, c_{i2}, \ldots, c_{in}) \) based on the ground-truth distributions of path \( P \)'s sub-paths, which can be obtained from the instantiated path weight function \( W_P \). While we may be able to obtain multiple estimations of joint distributions using different combinations of sub-paths’ distributions, we aim at identifying and deriving the most accurate one. In the following, we first prove that the combination with the coarsest sub-paths gives the most accurate estimation, and then we propose an efficient way to identify the coarsest combination.

4.1.1 Path Decompositions

To facilitate the following discussions, we first introduce the concept of path decomposition. The decomposition of a path \( P \) is a sequence of paths \( D_E = (P_1, P_2, \ldots, P_k) \), \( k > 1 \), that satisfies the following spatial conditions:

1. Each path \( P_i \in D_E \) is a sub-path of \( P \);
2. All paths in \( D_E \) together cover \( P \), i.e., \( P_1 \cup P_2 \cup \ldots \cup P_k = P \);
3. A path \( P_i \) is not a sub-path of another path \( P_j \), where \( 1 \leq i, j \leq k \) and \( i \neq j \);
4. A path \( P_i \) appears before another path \( P_j \), where \( 1 \leq i < j \leq k \) if and only if the first edge of \( P_i \) appears earlier than the first edge of \( P_j \) in path \( P \).

A path \( P \) may have more than one decomposition. For instance, consider path \( P = (e_1, e_2, e_3, e_4, e_5) \). The following path sequences are possible decompositions for the path.

\[
D_E_1 = ((e_1), (e_2, e_3), (e_4), (e_5)),
D_E_2 = ((e_1, e_2), (e_3, e_4), (e_5)),
D_E_3 = ((e_1), (e_2, e_3), (e_4, e_5)).
\]

Next, we introduce a coarser relationship between two path decompositions \( D_E_1 \) and \( D_E_2 \). We define \( D_E_1 \) to be coarser than \( D_E_2 \) if for each path \( P_i \in D_E_1 \), there is a path \( P_i \in D_E_2 \) such that \( P_i \) is a sub-path of \( P \), and at least one \( P_i \neq P \).

To illustrate, \( D_E_2 \) is coarser than \( D_E_3 \) because \((e_1, e_2, e_3), (e_3, e_4), (e_5)\) are sub-paths of \((e_1, e_2, e_3), (e_3, e_4), (e_5)\), respectively; and \((e_1, e_2, e_3, e_4)\) from \( D_E_2 \) is different from \((e_3, e_4)\) from \( D_E_3 \). Similarly, \( D_E_2 \) is coarser than \( D_E_1 \) as well.

4.1.2 Distribution Estimation using Decompositions

Following the principles of decomposable models \([5, 7, 15]\), a decomposition of path \( P \) corresponds to a set of independence assumptions among the cost variables in \( C_P \). Specifically, given a decomposition \( D_E = (P_1, P_2, \ldots, P_k) \), for any two paths \( P_i \) and \( P_j \) in the decomposition, where \( 1 \leq i, j \leq k \), we have the following cases:

1. If \( P_i \cap P_j = \emptyset \), this indicates that the cost variables in \( C_{P_i} \) are independent of the cost variables in \( C_{P_j} \);
2. If \( P_i \cap P_j \neq \emptyset \), this indicates that the cost variables in \( C_{P_i \backslash P_j} \) are conditionally independent of the cost variables in \( C_{P_j \backslash P_i} \), given the cost variables in \( C_{P_i \cap P_j} \).

Based on the above, given a decomposition \( D_E \), we have a corresponding independence assumption. Based on the independence assumption, we are able to estimate path \( P \)’s joint distributions using the distributions of the paths in \( D_E \) based on Bayes’ theorem.

Formally, given a decomposition \( D_E = (P_1, P_2, \ldots, P_k) \), the joint distribution of path \( P \) is estimated according to Equation 2.

\[
p_{D_E}(\mathcal{C}_P) = \frac{\prod_{P_i \in \mathcal{P}_X} p(\mathcal{C}_{P_i})}{\prod_{P_j \in \mathcal{P}_Y} p(\mathcal{C}_{P_j})},
\]

where path set \( \mathcal{P}_X = \bigcup_{1 \leq i < k} P_i \) consists of all paths in \( D_E \) and \( \mathcal{P}_Y = \bigcup_{2 \leq i < k} P_i \cap P_{i-1} \) consists of the shared paths between all adjacent paths in \( D_E \).
In the running example, $DE_1$ assumes all cost variables $c_{e_1}$, $c_{e_2}$, $c_{e_3}$, and $c_{e_4}$ are independent because no paths in $DE_1$ intersect. According to Equation 2, we have $\tilde{p}_{DE_1}(C_P) = p(c_{e_1}) \cdot p(c_{e_2}) \cdot p(c_{e_3}) \cdot p(c_{e_4})$. This corresponds to the legacy graph model where all cost variables are independent. Next, $DE_2$ assumes that $c_{e_1}$ is conditionally independent of $c_{e_4}$ given $c_{e_2}$ and $c_{e_3}$ because $(e_1, e_2, e_3) \cap (e_2, e_3, e_4) = (e_2, e_3)$; and $c_{e_4}$ is independent of all the other cost variables because $(e_2)$ does not intersect with other paths in $DE$. Thus, Equation 2 gives us $\tilde{p}_{DE_2}(C_P) = \frac{p(c_{e_1} \cap c_{e_2}) \cdot p(c_{e_3} \cap c_{e_4})}{p(c_{e_2}) \cdot p(c_{e_3})} \cdot p(c_{e_4})$.

Given a path $P$, we have more than one path decomposition. For each decomposition, we are able to derive an estimated joint distribution according to Equation 2. The challenge is to identify the decomposition that gives the most accurate estimate.

To measure the accuracy of an estimated distribution with respect to the true distribution, we employ Kullback-Leibler (KL) divergence of the estimated joint distribution $\tilde{p}_{DE}(C_P)$ and the true joint distribution $p(C_P)$, denoted as $KL(p(C_P), \tilde{p}_{DE}(C_P))$. The smaller the divergence, the better. We proceed to prove that the coarser decomposition is the more accurate the estimated joint distribution is, i.e., the more smaller the divergence is.

**Theorem 1.** If path $P'$ is a sub-path of path $P$, we have $\sum_{C_P} p(C_P) \log p(C_P) = -H(C_P')$, where $H(\cdot)$ is entropy function.

**Proof:** Since path $P'$ is a sub-path of path $P$, path $P$ can be represented as $P = P_1 \circ P' \circ P_2$. $P_1$, $P_2$, and $P_3$ are the possibly empty paths before and after path $P'$, and $\circ$ denotes concatenation. Thus, the cost variables in $C_{P_1}$, $C_{P_2}$, or $C_{P_3}$ are a subset of the cost variables in $C_P$. In addition, we have $C_{P_1 \cup P_3} = C_P$. Based on the above, we get

$$\sum_{C_P} p(C_P) \log p(C_P) = \sum_{C_{P_1 \cup P_3}} p(C_{P_1 \cup P_3}) \log p(C_{P_1 \cup P_3}) = \sum_{C_{P_1 \cup P_3}} p(C_{P_1 \cup P_3}) \log p(C_{P_1 \cup P_3}) = -H(C_{P_1 \cup P_3})$$

**Theorem 2.** Given an estimated joint distribution $\tilde{p}_{DE}(C_P)$, we have $KL(p(C_P), \tilde{p}_{DE}(C_P)) = H(D_T(C_P)) - H(C_P)$, where $H(C_P)$ and $H(D_T(C_P))$ are the entropies of random variables $C_P$ under distributions $p(C_P)$ and $\tilde{p}_{DE}(C_P)$, respectively.

**Proof:** $KL(p(C_P), \tilde{p}_{DE}(C_P)) =$ \[\sum_{C_P} p(C_P) \log \left( \frac{p(C_P)}{\tilde{p}_{DE}(C_P)} \right) \]

$$= -H(C_P) - \sum_{C_P} p(C_P) \log \tilde{p}_{DE}(C_P)$$

$$= -H(C_P) - \sum_{C_P} p(C_P) \left( \sum_{P_i \in P \land C_{P_i}} \log p(C_{P_i}) \right)$$

$$= -H(C_P) - \sum_{P_i \in P \land C_{P_i}} \log p(C_{P_i})$$

$$= -H(C_P) + \sum_{P_i \in P \land C_{P_i}} H(C_{P_i})$$

$$= -H(C_P) + \sum_{P_i \in P \land C_{P_i}} H(C_{P_i})$$

By Equation 2.

**Theorem 3.** Given two decompositions $DE$ and $DE'$, where $DE$ is coarser than $DE'$. $DE$ is able to provide a more accurate joint distribution estimation than is $DE'$.

**Proof:** Due to the space limitation, the detailed proof is included in the supplementary document [31].

### 4.1.3 Identifying the Coarsest Decomposition

According to Theorem 3, the estimated joint distribution from the coarsest decomposition is the most accurate. Given a path $P$ and a departure time $t$, we proceed to identify the coarsest decomposition for $P$ based on the instantiated random variables that are maintained in the path weight function $W_P$. Specifically, we first identify the random variables that are spatially relevant to the path $P$ and temporally relevant to the departure time $t$; then we identify the coarsest decomposition from the relevant variables.

Recall that a random variable maintained in the path weight function $W_P$ is in the form of $V_{i(t)}$, which represents the joint distribution of path $P_i$ during interval $I_i$. A random variable $V_{i(t)}$ is spatially relevant to the query path $P$ if the variable’s path $P_i$ is a sub-path of $P$. Next, we test if a spatially relevant random variable $V_{i(t)}$ is also temporally relevant to $t$.

We distinguish two cases. We first consider the case where the first edge in the variable’s path’s $P_i$ is the same as the first edge in the query path $P$. This case is simple because the departure time on $P$ is also the departure time on $P_i$, as both paths start from the same edge. Thus, we just need to test if the departure time $t$ is during $I_i$, i.e., $t \in I_i$. The second case is complicated because the departure time on path $P_i$ is no longer the original departure time $t_i$. We propose a shift-and-enlarge procedure to progressively update the departure time to test the variable’s temporal relevance.

We use an example path $(e_1, e_2)$ and a departure time $t$ to illustrate the procedure. Since the travel time on $e_1$ is uncertain, the departure time on $e_2$ belongs to the interval $[t + V_{i(t)}^{e_1}, \min(t + V_{i(t)}^{e_2}, \max)]$, where $V_{i(t)}^{e_1}$, $\min$ and $V_{i(t)}^{e_2}$, $\max$ denote the minimum and maximum travel times of traversing $e_1$, which are recorded in the random variable $V_{i(t)}^{e_1}$.

Formally, given a time interval $[t, t_e]$ and a random variable $V_{i(t)}^{e_1}$, we define the shifted-and-enlarged interval as $SAE([t, t_e], V_{i(t)}^{e_1}) = [t + V_{i(t)}^{e_1}, \min(t_e + V_{i(t)}^{e_2}, \max)]$.

Given a sub-path $P_i$ of path $P$, assume that the first edge in $P_i$ is the $k$-th edge in $P$, where $2 \leq k \leq |P|$. The updated departure time on $P_i$, based on the original departure time $t$ is $U_{i(k)}$, as defined in Equation 3.

$$U_{i(k)} = \begin{cases} 
SAE([t, t_e], V_{i(t)}^{e_1}), & \text{if } k = 2; \\
SAE(U_{i(k-1)}, V_{i(t)}^{e_{k-1}}), & \text{otherwise.} 
\end{cases}$$

Given a spatially relevant random variable $V_{i(t)}^{e_1}$, if $I_i$ intersects $P_i$’s updated departure time interval $U_{i(k)}$ according to Equation 3 then it is temporally relevant; otherwise, it is not temporally relevant. For a given sub-path $P_i$, if multiple variables $V_{i(t)}^{e_j}$, ($j = 1, 2, \ldots, m$) are temporally relevant, the one with the largest overlap is selected, i.e., the one where $I_j = \max_{j \in [1, m]} \left\lceil \frac{|I_j \cap U_{i(k)}|}{|U_{i(k)}|} \right\rceil$.

Having identified all spatially and temporally relevant variables, we organize them into a two-dimensional candidate array. Each row corresponds to an edge $e_k$ in query path $P$ and contains the instantiated random variables whose corresponding paths start with
edge $e_k$. If more than one variable exist, the variables are ordered by their rank. An example candidate array is shown in Table 1.

<table>
<thead>
<tr>
<th>rank=1</th>
<th>rank=2</th>
<th>rank=3</th>
<th>rank=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{e_1}$</td>
<td>$V_{e_1,e_2}$</td>
<td>$V_{e_1,e_2,e_3}$</td>
<td>$V_{e_1,e_2,e_3,e_4}$</td>
</tr>
<tr>
<td>$V_{e_2}$</td>
<td>$V_{e_2,e_3}$</td>
<td>$V_{e_2,e_3,e_4}$</td>
<td></td>
</tr>
<tr>
<td>$V_{e_3}$</td>
<td>$V_{e_3}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{e_4}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{e_5}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Example Candidate Array

To identify the coarsest decomposition, we consider the path of the random variable with the highest rank for each edge (i.e., the rightmost variable for each row in Table 1). If one random variable’s path is a sub-path of another random variable’s path, the former random variable’s path should be omitted because it otherwise violates spatial condition (3) (cf. Section 4.1.2). The remaining paths constitute the coarsest decomposition. The procedure is summarized in Algorithm 1.

Algorithm 1: Identify the Coarsest Decomposition

Input: Query Path $P$, Departure Time $t$
Output: The Coarsest Decomposition $DE_{coa}$
1 Identify a set of random variables $STRV$ that are spatially relevant to path $P$ from all the instantiated random variables;
2 Eliminate the random variables that are not temporally relevant to departure time $t$ from $STRV$;
3 Organize the spatio-temporally relevant random variables in $STRV$ in an two dimensional array;
4 $DE_{coa} ← null$;
5 for $k = 1; k ≤ |P|; k + +$ do
6 Identify the random variable $V_{P_k}$ with the highest rank from the $k$-th row;
7 if $P_k$ is not a sub-path of any path in $DE_{coa}$ then
8 Append $P_k$ to $DE_{coa}$;
9 return $DE_{coa}$;

Theorem 4. The unique decomposition $DE_{coa}$ returned by Algorithm 1 is the coarsest.

Proof: We prove the theorem by contradiction. Suppose that we cannot identify the coarsest decomposition by Algorithm 1. This happens only if the coarsest decomposition, say $DE_{coa}^*$, contains a sub-path $P_k^*$ that starts with edge $e_k$ but is not the longest sub-path that starts with edge $e_k$. Otherwise, it must be identified by Algorithm 1 since it considers the longest sub-path for each edge in path $P$.

Following the above assumption, we assume that the longest sub-path starting from $e_k$ is $P_k$. By replacing $P_k^*$ by $P_k$, we are able to get a new decomposition, $DE_{coa}$, that is coarser than $DE_{coa}^*$. This contradicts the assumption that $DE_{coa}$ is the coarsest. 

4.2 The Cost Distribution of a Path

The coarsest decomposition $DE_{coa}$ enables accurate estimation of the joint distribution of a path which fully captures the dependencies among edges in the path. Recall that we are interested in knowing the cost distribution of a path $p(V_P)$, where $V_P$ is a univariate random variable indicating the travel cost of path $P$. We proceed to derive $p(V_P)$ based on the joint distribution of a path $p(\tilde{DE}_{coa}(C_P))$ using

$$p(V_P = x) = \sum_{v_1, \ldots, v_n = x} p(\tilde{DE}_{coa}(v_1, \ldots, v_n))$$

Since the estimated joint distribution of a path $p(\tilde{DE}_{coa}(C_P))$ is represented as a multi-dimensional histogram, we need to transform it to a one-dimensional histogram that represents the cost distribution of $P$.

Recall that a multi-dimensional histogram is of the form $\{\langle b_{i_1}, p_{i_1} \rangle, \ldots, \langle b_{i_d}, p_{i_d} \rangle\}$, where $b_{i_d}$ is a hyper-bucket $b_{i_1} = \langle b_{i_1}, \ldots, b_{i_d} \rangle$ consists of $n$ buckets, each corresponding to one dimension. For each hyper-bucket $b_{i_d}$, we derive a bucket $b_{i_d}$ whose upper (lower) bound is the sum of the upper (lower) bounds of the buckets in the hyper-bucket, i.e., $b_{i_d} = \bigcup_{i_1=1}^{n} b_{i_1,j_1}$. Thus, we get a one-dimensional histogram $\{\langle b_{i_d}, p_{i_d} \rangle\}$ as follows. Bucket $b_{i_d}$ is split into three buckets according to the increasing order of $b_{i_1,j_1}, b_{i_1,j_2}, b_{i_1,j_3}$ into three buckets according to the increasing order of $b_{i_1,j_1}, b_{i_1,j_2}, b_{i_1,j_3}$, and each bucket is assigned an adjusted probability. The one-dimensional histogram with the rearranged buckets and the adjusted probabilities represents the final cost distribution.

Figure 7 shows a running example on the aforementioned procedure on path $P_1 = (e_1, e_2)$. The first table in Figure 7 shows the joint distribution of the path. The upper, left hyper-bucket $\langle [20, 30), [20, 40) \rangle$ has value 0.3, which means that when going through path $P_1$, the probability that the travel time on $e_1$ is between 20 and 30 s and the travel time on $e_2$ is between 20 s and 40 s is 0.3. Next, the second table in Figure 7 shows the corresponding cost distribution after transferring each hyper-bucket to a bucket. For example, hyper-bucket $\langle (20, 30), (20, 40) \rangle$ becomes bucket $[40, 70)$.

Figure 7: A Joint Distribution and Its Marginal Distribution

Consider the first two (bucket, probability) pairs shown in the second table, i.e., $\langle [40, 70), 0.30 \rangle$ and $\langle [50, 90), 0.25 \rangle$. Since the two buckets overlap, range $[40, 90)$ is split into $[40, 50)$, $[50, 70)$, and $[70, 90)$. In a histogram, the probability in each bucket is uniformly distributed, so each bucket is assigned an adjusted probability as follows. Bucket $[40, 50)$ is given probability $\frac{[40, 50)}{[40, 70)} \cdot 0.3 = 0.1$, bucket $[50, 70)$ is given probability $\frac{[50, 70)}{[40, 70)} \cdot 0.3 = 0.3$, and bucket $[70, 90)$ is given probability $\frac{[70, 90]}{[40, 70)} \cdot 0.3 = 0.25$. 

Figure 7: A Joint Distribution and Its Marginal Distribution
of a path using our method is more efficient than using the state-
we empirically demonstrate that computing the cost distribution
ran-time of computing the cost distribution of path

In the first scenario, multiple candidate paths, e.g., using or not
using highways or toll roads, are given. Here, the proposed method
can be applied to compute the cost distributions for the given paths,
identifying the most appropriate path based on the requirements,

g, e.g., the one with highest probability of arriving within 60 mins.

The second scenario concerns stochastic routing algorithms, e.g.,
stochastic fastest routing [13], probabilistic top-k routing [10],
stochastic skyline routing [22], that aim to identify a path or set of
paths that satisfy some conditions, e.g., the path with highest prob-
ability of arriving within 60 mins. Such algorithms need to ex-

ore many candidate paths, and pruning of uncompetitive candi-
date paths early is attractive. When making pruning decisions, the
algorithms compare the cost distributions of candidate paths. The
proposed method can be easily incorporated into existing stochastic
routing algorithms by simply using it when the algorithms compute
cost distributions of paths.

Existing stochastic routing algorithms apply a “path + another
edge” pattern to explore candidate paths. Therefore, a path cost dis-
tribution estimation method must satisfy the so-called “incremental
property” that enables reuse of the cost distribution of the existing
path when computing the cost distribution for a new path extended
from this existing path. Our proposal satisfied this property and can
be applied to “path + another edge” path finding algorithms. The
details are covered elsewhere [31].

The run-time of a stochastic routing algorithm is dominated by
\( \sum_{P \in \mathcal{CP}} RT(P, \text{method}) \), where \( \mathcal{CP} \) contains the candidate paths
whose cost distributions need to be evaluated. Thus, \( \mathcal{CP} \) differs
among stochastic routing algorithms with different strategies for
selecting candidate paths. Function \( RT(P, \text{method}) \) refers to the
run-time of computing the cost distribution of path \( P \) using a spe-
cific method, e.g., our method or a legacy method. In Section 5,
we empirically demonstrate that computing the cost distribution of
a path using our method is more efficient than using the state-
of-the-art legacy baselines, i.e., \( RT(P, \text{hybrid\_graph}) < RT(P, \text{legacy\_baseline}) \). We also show that the accuracy of the cost dis-

tributions estimated by our method is higher than that of the legacy
baseline. Thus, integration of our method is able to improves both
the efficiency and accuracy of existing stochastic routing algorithms.

5. EMPIRICAL STUDY

5.1 Experimental Setup

Road networks: Two road networks are used. The Aalborg road
network \( N_1 \) has 20,195 vertices and 41,276 edges, and the Beijing
road network \( N_2 \) has 28,342 vertices and 38,577 edges. Road net-

work \( N_1 \) is obtained from OpenStreetMap and contain all roads,
while road network \( N_2 \) is obtained from the Beijing traffic man-
agement bureau and contains only highways and main roads.

Trajectories: Two GPS data sets are used. The first, \( D_1 \), con-
tains 37 million GPS records that occurred in Aalborg from January
2007 to December 2008. The sampling rate is 1 Hz (i.e., one record
per second). The second, \( D_2 \), contains more than 50 billion GPS
records that occurred in Beijing from September 2012 to November
2012. The sampling rate is at least 0.2 Hz. We apply a well-known
method [16] to map match the GPS records.

Travel Costs: We consider two time-varying, uncertain travel
costs—travel time and GHG emissions. Due to the space limitation,
the results on GHG emissions are included elsewhere [30].

Parameters: We vary the finest time interval \( \alpha \), the qualified tra-
jectory count threshold \( \beta \), the cardinality of a query path \( |P_{query}| \);
see Table 2, where default values are shown in bold.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha ) (min)</td>
<td>15, 30, 45, 60, 120</td>
</tr>
<tr>
<td>( \beta )</td>
<td>15, 30, 45, 60</td>
</tr>
<tr>
<td>(</td>
<td>P_{query}</td>
</tr>
</tbody>
</table>

Table 2: Parameter Settings

Implementation Details: All algorithms are implemented in
Python 2.7 under Linux Ubuntu 14.04. All experiments are con-
ducted on a modern server with 512 GB main memory and 64 2.3
GHz 8-core AMD Opteron(tm) 6376 CPUs.

5.2 Experimental Results

5.2.1 Instantiated Random Variables

We conduct experiments to obtain insight into different aspects of
the instantiated random variables that are maintained in the hy-

brid graph’s weight function \( W_P \) and also describe how to tune
parameters \( \alpha \) and \( \beta \). To highlight the random variables that are in-
stantiated from trajectories, random variables derived from speed
limits are excluded unless stated otherwise.

Tuning \( \alpha \): We vary \( \alpha \) from 15 to 120 minutes. A large \( \alpha \)
suggests that more trajectories may become qualified trajectories,
which instantiates more random variables. We use \( E' \) to denote
the set of edges that are covered by the instantiated random vari-
able and \( E'' \) to denote the set of edges that has at least one GPS
record. Coverage is defined as the ratio between \( |E'| \) and \( |E''| \).
Figure 8(a) shows that as \( \alpha \) increases, the coverage increases as
well on both data sets. However, the coverage ratio remains below
85% for \( \alpha = 120 \). This is because the GPS data is skewed—some
edges have only few GPS records.

Although a large \( \alpha \) enables more instantiated random variables,
they may be inaccurate since traffic may change significantly dur-
ing a long interval, e.g., two hours. We report the average en-
tropies of the instantiated random variables when varying \( \alpha \); see
Figure 8(b). We only show the results on \( D_2 \) as the results on
\( D_1 \) exhibit similar trends. According to Theorem 2, variables with
smaller entropy lead to more accurate joint distribution estimates.
Figure 8(b) shows that using \( \alpha = 30 \) does not significantly in-
crease the entropy compared to using \( \alpha = 15 \). Figure 8(a) shows
that \( \alpha = 30 \) gives a clear increase in the number of instantiated
random variables compared to \( \alpha = 15 \). This suggests that \( \alpha = 30 \)
provides a good trade-off between the accuracy of the random vari-
able and the numbers of random variables. Thus, we use \( \alpha = 30 \)
as the default value.

Tuning \( \beta \): Intuitively, we prefer a large \( \beta \) since having more
qualified trajectories enables instantiated random variables that are
more accurate. However, Figure 9 shows that as \( \beta \) increases, the
number of instantiated random variables drops. This occurs because a large $\beta$ requires more qualified trajectories in order to be able to instantiate $W_{Pr}$. We choose $\beta = 30$ as the default because the number of instantiated random variables is only slightly less than for $\beta = 15$, while achieving more accurate variables.

**Varying Dataset Sizes:** We use 25%, 50%, 75%, and 100% of the trajectories in $D_1$ and $D_2$, respectively. Figure 10 shows that the number of instantiated random variables increases as the number of trajectories increases. The number of instantiated random variables with large rank also increases steadily. This occurs because the more trajectories, the more likely it is to find long paths with more than $\beta$ qualified trajectories, thus resulting in random variables with large rank. It also shows that the instantiated random variables are typically insufficient to enable the accuracy-optimal baseline for arbitrary paths—the sizes of variables with high rank (e.g., $|V| \geq 4$) are small.

**Histogram Approximation:** We evaluate the accuracy and space savings of the histogram representations of the instantiated random variables. Recall that our method is able to automatically identify the number of buckets per dimension (cf. Section 3.1). We call this method Auto. We first compare Auto with methods using standard distributions including Gaussian, Gamma, and exponential distributions. Figure 11(a) shows the KL divergences between the raw data distribution and the distributions represented by different methods. The results of using exponential distributions are not shown since their KL divergences exceed 1.0 and are significantly worse than the latter ones. The results clearly suggest that Auto provides the most accurate estimation and that travel-time distributions typically do not follow standard distributions.

Auto adaptively determines the bucket count for each dimension and then optimally selects the bucket boundaries, thus being able to represent arbitrary distributions. We compare Auto with a static histogram approach that uses a fixed number of buckets per dimension. The method that uses $b$ buckets per dimension is called Sta-b. Figure 11(b) shows the KL divergences between the raw distribution that is obtained from the trajectories’ travel costs and the different histograms. As the number of buckets increases, Sta-b produces a smaller KL divergence value because the more buckets a histogram has, the better accuracy it can achieve. Auto is able to achieve as good accuracy as Sta-b. This suggests that the Auto method is effective.

We evaluate the space-savings achieved by the histogram representation. Intuitively, the more buckets a histogram has, the more storage it needs. We report the space saving ratio $1 - \frac{2n}{S_H}$, where $S_H$ and $S_n$ represent the storage required by the histograms and the underlying raw data distribution, respectively. The raw data distribution is of the form $\text{cost}, \text{frequency}$. The higher the ratio, the better space-savings are achieved by the histograms. Figure 11(c) shows that Auto has a better compression ratio than Sta-4 has. This suggests that Auto achieves a good trade-off between accuracy and space-saving.

**Total Memory Usage:** As the size of the trajectory data set grows, the memory use of recording the instantiated random variables also grows, as shown in Figure 12. Since we use histograms to represent the distributions of instantiated random variables, the memory use is small such that the hybrid graph’s weight function $W_{Pr}$ can be accommodated in main memory. In particular, the instantiated random variables, including the ones that are derived from speed limits, for Aalborg and Beijing occupy around 1.8 GB and 4.2 GB, respectively.

**Run-time:** Since deriving the instantiated random variables is an off-line task, the run-time is not critical. The procedure can be parallelized in a straightforward manner. Using the default parameter setting, it takes less than 2 minutes with 48 threads to learn the random variables from $D_1$, and takes around 45 minutes with 48 threads to learn random variables from $D_2$. This also suggests that when receiving new trajectories regularly, the procedure can be conducted periodically to efficiently instantiate random variables.

### 5.2.2 Path Cost Distribution Computation

We consider four methods. (a) OD: the proposed method using the optimal (i.e., coarsest) decomposition. (b) LB [22]: the legacy baseline as described in Section 2.3. LB is regarded as one of the state-of-the-art approaches used in the conventional paradigm. In our setting, LB only considers the random variables with ranks one. (c) HP [10] assumes that the joint distributions for every pair of edges in a path are known and then computes the joint probability distribution of the path taking these into account. In our setting, this means that HP only considers random variables with ranks two. (d) RD computes an estimated distribution using a randomly chosen decomposition rather than the coarsest decomposition.

**Accuracy Evaluation with Ground Truth:** We select 100 paths where each path has more than $\beta = 30$ trajectories during an interval of $\alpha = 30$ minutes. We use these trajectories to compute the ground-truth distribution using the accuracy-optimal baseline. Next, we exclude these trajectories from the trajectory data set. Thus, we have the data sparseness problem, and the accuracy-optimal baseline does not work.

First, we consider a concrete example shown in Figure 1(b). The distributions estimated using OD, LB, HP, and RD are shown in Figures 13(a)-(d). It is clear that OD captures the main characteristics of the ground-truth distribution. The convolution-based estimation LB seems to approach a Gaussian distribution (cf. the Central Limit Theorem). However, it is clear that a Gaussian distribution is unable to capture the ground-truth distribution, and LB is inaccurate. The distribution computed by HP is inaccurate either, which suggests that the dependencies among the edges in a path cannot be fully captured by only considering the dependencies between adjacent edges. Method RD suggests that a randomly chosen decomposition provides a less accurate estimation compared to the estimation based on the optimal decomposition.

Next, we report results when using paths with different cardinalities. Specifically, we vary $|P_{query}|$ from 5 to 20. Figure 14 shows the KL-divergence values $KL(p, \hat{p})$, where $p$ is the ground-truth distribution derived by the accuracy-optimal baseline and $\hat{p}$ is the estimated distribution using OD, LB, RD, and HP. As the number of edges in a path increases, the benefits of using the proposed OD becomes more significant. In particular, the KL-divergence values of OD grow slowly while the KL-divergence values of LB grow quickly. This is not surprising because LB assumes independencies, and the longer a path is, the more likely it is that the edges in the path are not independent. Next, OD is also better than RD, which suggests that the optimal decomposition produces the most accurate estimation. Further, HP is better than LB because HP considers the correlation between adjacent edges. However, HP always has larger KL-divergence values than do RD and OD. This is because coarser random variable sets have smaller KL-divergence (cf. Theorem 3).

In summary, Figure 14 suggests that the proposed OD is able to accurately estimate travel cost distributions and that it outperforms the other methods, especially for long paths.

**Accuracy Evaluation without Ground Truth:** We consider long paths which do not have corresponding ground truth distributions. We randomly choose 1,000 paths for each cardinality with an arbitrary departure time and report average values; and vary the path cardinality from 20 to 100. Figure 15 shows that OD produces...
the least entropy, which is consistent with the design of identifying the optimal decomposition. This suggests that the proposed method is able to accurately estimate the distribution of a path.

Efficiency, Estimating the Cost Distribution of a Path: Figure 16 reports the run-times of the different methods. We also consider the methods that use instantiated random variables with ranks at most 2, 3, and 4, denoted as OD-2, OD-3, and OD-4, respectively. As the cardinality of a query path increases, the run-time also grows. HP and LB have to consider at least \(|P_{\text{query}}|\) instantiated random variables to compute the joint distribution, yielding higher running times than for the remaining methods. In contrast, OD, OD-x, and RD exploit instantiated random variables with higher ranks. Thus, they use significantly fewer instantiated random variables and are faster than HP and LB. OD uses coarser random variables than does RD, and it is able to use fewer instantiated random variables, making it faster than RD. Following the same reasoning, OD-x is faster than OD-y if \(x > y\). Figure 16 clearly shows that OD is the most efficient.

To further investigate the run-time of OD, a detailed analysis of the three major steps in OD is reported in Figure 17 for paths with cardinality 20 and using differently sized subsets of trajectories. Three steps are involved in OD. First, the optimal decomposition is identified, denoted by \(\Pi\). Thanks to Theorems 4, this part is very efficient. Second, the joint distribution is computed, denoted by \(JC\). This is the most time-consuming part as it goes through many hyper-buckets of the histograms to compute the joint distributions according to Equation 2. However, with more trajectories, there are more instantiated random variables with higher ranks, which improves the run-time of \(JC\). Thus, as data volumes increase, the performance improves. Third, deriving the cost distribution (denoted by \(MC\)) is also very efficient.

Efficiency, Stochastic Routing: We consider the efficiency of using the hybrid graph in existing stochastic routing algorithms following the discussions offered in Section 4.3. In particular, we use an existing depth first search (DFS) based stochastic routing algorithm [10] for answering the question posed in Figure 1(a). We implement the algorithm in versions that use LB, HP, or OD to compute cost distributions, respectively.

We randomly choose 100 source-destination pairs, and we set the travel time budgets to \(S_1 = 10\) min, \(S_2 = 20\) min, and \(S_3 = 30\) min, respectively. The average running times are reported in Figure 18, which shows that OD-DFS always outperforms the other two. We see that replacing the legacy baselines by the paper’s proposal is able to accelerating an existing stochastic routing algorithm. Due to the space limitation, experimental results on the effects on quality of using the paper’s proposal for stochastic routing are covered elsewhere [31].

5.2.3 Summary

The empirical study shows that: (1) In realistic settings with sparse data, the proposed OD method is the most accurate and efficient method and is scalable w.r.t. the path cardinality, meaning that it is able to support long paths. (2) OD is able to approximate arbitrary raw cost distributions well using limited space, making it possible to fit the instantiated random variables into main memory. (3) OD is scalable w.r.t. the number of trajectories. First, as random variable instantiation can be parallelized easily, it is possible to periodically re-instantiate random variables when new trajectories are received. Second, more trajectories yield more random variables with higher ranks, which improves the efficiency and accuracy of the approach. (4) OD can be incorporated into existing stochastic routing algorithms with the effect of improving their efficiency and accuracy.

We conclude that the proposed hybrid-graph and OD method successfully address the challenges caused by data sparseness, complex distributions, and dependencies; and they are able to efficiently provide accurate travel cost distribution estimation, thus enabling efficient and accurate stochastic routing.

6. RELATED WORK

We review recent studies on estimating deterministic and uncertain path costs, respectively.

Estimating Deterministic Path Costs: Most such studies focus on accurate estimation of travel costs of individual edges using trajectory data and loop detector data, based on which the travel cost of a path is then computed as the sum of the travel costs of its edges. In many cases, the available trajectory data is unable to cover all edges in a road network. To address data sparseness, some methods [11,19,26,29] transfer the travel costs of edges that are covered by trajectories to edges that are not covered by trajectories. However, these methods do not support travel cost distributions, and they do not model dependencies among edges. Therefore, they do not apply to the problem we consider.

When all edges have travel costs, the travel cost of any path can be estimated by summing up the travel costs of the edges in the path [11,26,29]. However, using the sum of travel costs of edges as the travel cost of a path can be inaccurate because it ignores hard-to-formalize aspects of travel, such as turn costs. Thus, a method [19] is proposed to identify an optimal set of sub-paths that can be concatenated into a path. The path’s travel cost is then the sum of the travel costs of the sub-paths. This method does not support travel cost distributions, and it assumes independence among sub-paths.

Another study explicitly models turn costs [23], and the path cost is the sum of costs of edges and the costs of turns. However, the study models turn costs based on many assumptions, e.g., maximum turning speeds, but not on real-world traffic data, and the accuracy of the modeling is unknown. In contrast, we do not explicitly model turn costs but use the joint distributions of paths to implicitly model such hard-to-formalize factors. Further, the study [23] does not consider time-dependent and uncertain costs.

Estimating Path Cost Distributions: Studies exist that model the travel cost uncertainty of a path. However, they make assumptions that do not apply in our setting.

First, some studies assume that travel cost distributions follow a standard distribution, e.g., a Gaussian distribution. However, the travel cost distribution of a road segment often follows an arbitrary distribution, as shown in recent studies [21,22,25] and in Figure 1(b) in Section 1 and Figure 11(a) in Section 5. We use multi-dimensional histograms to represent arbitrary distributions.

Second, some studies assume that the distributions on different edges are independent of each other [6,13] or conditionally independent given the arrival times at different edges [22], thus mirroring the LB approach covered in Section 5. The independence assumption often does not hold (cf. Section 2), and our approach outperforms LB, as shown in Section 5. Further, with the exception of one study [22], all the above studies use generated distributions in empirical evaluations; we use large, real trajectory data.

The most advanced method, the HP [10] approach covered in Section 5, does not make the independence assumption. Rather, it assumes that the travel costs of pairs of adjacent edges are dependent, but it does not consider dependencies among multiple edges in a path. We propose a more generic model that employs joint distributions to fully capture the dependencies among all the edges in a path. In addition, we identify distributions from real-world trajectory data and support time-varying distributions, while the HP
approach employs synthetically generated distributions and does not support time-varying distributions.

Although two recent studies [14,27] employ histograms to represent travel cost distributions, they consider travel cost distributions on individual edges and assume that distributions are independent.

7. CONCLUSION AND FUTURE WORK

Accurate estimation of the travel cost distribution of a path is fundamental functionality in spatial-network related applications. We propose techniques that are able to model joint distributions that capture the travel cost dependencies among sub-paths that form longer paths, which in turn enables accurate travel cost estimation of any path using sparse historical trajectory data. Empirical studies in realistic settings offer insight into the design properties of the proposed solution and suggest that it is effective and efficient.

This study provides part of the foundation for a new and promising paradigm where travel costs are associated not only with road-network edges, but with sub-paths. The arguably most pertinent next step is to develop novel stochastic routing algorithms that fully consider the distinct characteristics of the novel paradigm.

Acknowledgments

This research was supported in part by the National Research Foundation, Prime Minister’s Office, Singapore, under its Competitive Research Programme (CRP Award No. NRF CRP8-2011-08), by a grant from the Obel Family Foundation, and by the DiCyPS project. We thank Prof. Beng Chin Ooi and the anonymous reviewers for their insightful comments.

8. REFERENCES