The VLDB Journal (2018) 27:153–178
https://doi.org/10.1007/s00778-017-0491-4

REGULAR PAPER

PACE: a *PA*th-*CE*ntric paradigm for stochastic path finding

Bin Yang[1] · Jian Dai[2] · Chenjuan Guo[1] · Christian S. Jensen[1] · Jilin Hu[1]

Received: 25 May 2017 / Revised: 22 October 2017 / Accepted: 9 November 2017 / Published online: 24 November 2017

Abstract With the growing volumes of vehicle trajectory data, it becomes increasingly possible to capture time-varying and uncertain travel costs, e.g., travel time, in a road network. The current paradigm for doing so is edge-centric: it represents a road network as a weighted graph and splits trajectories into small fragments that fit the underlying edges to assign time-varying and uncertain weights to edges. It then applies path finding algorithms to the resulting, weighted graph. We propose a new *PA*th-*CE*ntric paradigm, *PACE*, that targets more accurate and more efficient path cost estimation and path finding. By assigning weights to paths, *PACE* avoids splitting trajectories into small fragments. We solve two fundamental problems to establish the *PACE* paradigm: (i) how to compute accurately the travel cost distribution of a path and (ii) how to conduct path finding for a source–destination pair. To solve the first problem, given a departure time and a query path, we show how to select an optimal set of paths that cover the query path and such that the weights of the paths enable the most accurate joint cost distribution estimation for the query path. The joint cost distribution models well the travel cost dependencies among the edges in the query path, which in turn enables accurate estimation of the cost distribution of the query path. We solve the second problem by showing that the resulting path cost distribution estimation method satisfies an incremental property that enables the method to be integrated seamlessly into existing stochastic path finding algorithms. Further, we propose a new stochastic path finding algorithm that fully explores the improved accuracy and efficiency provided by *PACE*. Empirical studies with trajectory data from two different cities offer insight into the design properties of the *PACE* paradigm and offer evidence that *PACE* is accurate, efficient, and effective in real-world settings.

Keywords Stochastic routing · Routing · Trajectories · Road networks · Path cost distributions

✉ Jian Dai
daij@comp.nus.edu.sg

Bin Yang
byang@cs.aau.dk

Chenjuan Guo
cguo@cs.aau.dk

Christian S. Jensen
csj@cs.aau.dk

Jilin Hu
hujilin@cs.aau.dk

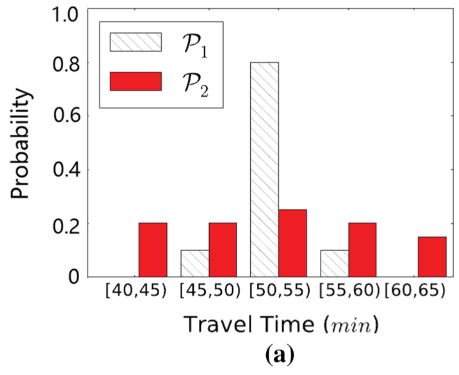
1 Department of Computer Science, Aalborg University, Aalborg, Denmark

2 School of Computer Science, National University of Singapore, Singapore, Singapore

1 Introduction

Increasing volumes of vehicle trajectories are becoming available that contain detailed traffic information. It is of interest to exploit this data as well as possible to understand the state of a road network [11,15]. For instance, it is of interest to know the *distributions* [4,30] of travel costs (e.g., travel times or greenhouse gas (GHG) emissions) 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 an airport within 60 min to catch a flight. Figure 1a shows the travel-time distributions of two alternative paths at a given departure time. If only considering averages, $\mathcal{P}_2$ (with mean 51.5 min) is better than $\mathcal{P}_1$ (with mean 52 min). However, $\mathcal{P}_1$ is preferable because the probability of arriving at the airport



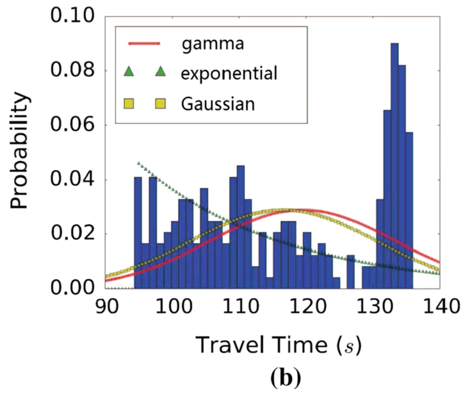


Fig. 1 Motivating examples

within 60 min is 1, while with $\mathcal{P}_2$, the probability is 0.9. The example illustrates why being able to compute a distribution rather than, e.g., a mean, is important.

To enable better routing, a first, natural question is how to best utilize trajectories to accurately and efficiently derive a cost distribution for any path at a given departure time. The next question is how to enable efficient path finding that utilizes such accurate path cost distributions. These are the two fundamental problems addressed in this paper.

To solve the first problem, three challenges must be addressed.

Complex travel cost distributions Travel time [7,37] and GHG emissions [3,18] vary over time and even vary across vehicles traversing the same path at the same time. To exemplify the latter, the bars in Fig. 1b 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 1b shows the Gaussian [31], gamma [35], and exponential [35] 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 [6,24,30,38]. However, the results are only accurate if the edge distributions are independent. We offer evidence that this is generally not the case in our setting. To derive accurate distributions for paths, dependencies must be considered.

The conventional, edge-centric paradigm for path cost estimation fragments trajectories into pieces that fit the individual edges to assign time-varying, uncertain weights to the edges. Convolution is applied to the uncertain edge weights to compute the cost distribution of the path [6,21,24,38–40,42]. This paradigm falls short in addressing the above challenges and suffers in terms of accuracy because dependencies among different edges are not accounted for and suffers in terms of efficiency because many expensive convolutions must be performed.

We propose a data-driven, *PA*th-*CE*ntric paradigm, *PACE*, that aims to achieve better accuracy and efficiency while addressing the three challenges. In *PACE*, 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 paths that together cover the query path and such that the weights of the paths 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 ability to derive the distributions of long paths with insufficient trajectories from the distributions of carefully selected sub-paths with sufficient trajectories addresses the *sparseness* problem.

Next, we study efficient path finding, a.k.a., routing, in *PACE*, i.e., the paper's second problem. Solving this problem is also non-trivial because existing path finding algorithms, ranging from the classical Dijkstra's algorithm [10] to the state-of-the-art algorithms such as contraction hierarchies [13] and hub labeling [1], are designed for the edge-centric paradigm. These algorithms operate on edge weights, which are always disjoint and independent of each other. In contrast, the paths in *PACE* may overlap, and thus the path weights are dependent on each other. This fundamental difference indicates that existing edge-centric path finding algorithms cannot be applied directly in *PACE*.

We propose two complimentary approaches to enable path finding in *PACE*. First, we prove that the path cost distribution estimation method in *PACE* satisfies what we call the *incremental property*, which is existing, edge-centric path finding algorithms rely on for their correctness. This suggests that the method can be integrated seamlessly into existing stochastic routing algorithms while enhancing the efficiency and accuracy of these algorithms. Second, we propose a new path finding algorithm that fully exploits the characteristics of *PACE* thus offering improved accuracy and efficiency.

To the best of our knowledge, this is the first study that enables path-centric path finding, which offers more accurate and efficient path travel cost distribution estimation and thus also more accurate and efficient path finding than edge-centric path finding. In particular, we make four contributions. (1) We propose the *PACE* paradigm and a method to instantiate path weights using trajectories (Sect. 3). (2) We propose an algorithm that identifies an optimal set of paths, the weights of which enable accurate estimation of the joint distribution of a query path, which in turn enables deriving the cost distribution of the query path (Sect. 4). (3) We show that the proposed path cost distribution estimation algorithm satisfies the incremental property, meaning that it can be integrated seamlessly into existing path finding algorithms (Sect. 5.2). (4) We propose a new path finding algorithm capable of fully exploiting the characteristics of *PACE* (Sect. 5.3). (5) We report on empirical studies that demonstrate that the paper's proposal is capable of significantly outperforming existing proposals in terms of both accuracy and efficiency (Sect. 6).

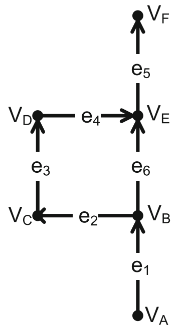
A preliminary report of this work [8] covers part of the foundation of *PACE* by solving the first problem of computing the travel cost distribution for a path. Here, we complement *PACE* by also solving the second problem of conducting path finding in *PACE*. Specifically, this includes (1) showing that the path cost distribution estimation algorithm in *PACE* satisfies the incremental property; (2) devising a generic proposal for how to apply existing edge-centric path finding algorithms in *PACE*; (3) equipping *PACE* with a new path finding algorithm capable of fully exploiting the characteristics of *PACE*; (4) reporting on comprehensive experiments on the efficiency and accuracy of path finding in *PACE*.

Paper outline Section 2 covers basic concepts and baselines. Section 3 introduces *PACE* and the method for instantiating path weights. Section 4 presents the algorithms for estimating the travel cost distribution of a path. Section 5 covers the proposals for conducting path finding in *PACE*. Section 6 reports on the empirical study. Related work is covered in Sects. 7, and 8 concludes.

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 $e_k = (v_i, v_j) \in E$ models a directed road segment, indicating that travel is possible from its *start vertex* v_i to its *end vertex* v_j. We use $e_k.s$ and $e_k.d$ to denote the start and end vertices of edge e_k. Two edges are *adjacent* if one edge's end vertex is the same as the other edge's start vertex. Figure 2a shows an example road network.



(a)

Trajectory	Path of Trajectory	Time
$\mathcal{T}_1$	$\langle e_1, e_2, e_3, e_4\rangle$	8:01
$\mathcal{T}_2$	$\langle e_1, e_2, e_3, e_4\rangle$	8:02
$\mathcal{T}_3$	$\langle e_1, e_2, e_3\rangle$	8:10
$\mathcal{T}_4$	$\langle e_1, e_2, e_3\rangle$	8:07
$\mathcal{T}_5$	$\langle e_2, e_3, e_4\rangle$	8:01
$\mathcal{T}_6$	$\langle e_2, e_3, e_4\rangle$	8:10
$\mathcal{T}_7$	$\langle e_2, e_3, e_4\rangle$	15:21
$\mathcal{T}_8$	$\langle e_4, e_5\rangle$	8:07
$\mathcal{T}_9$	$\langle e_4, e_5\rangle$	8:07
$\mathcal{T}_{10}$	$\langle e_6, e_5\rangle$	8:08

(b)

Fig. 2 A road network and trajectories. **a** Road network. **b** Trajectories

A *path* $\mathcal{P} = \langle e_1, e_2, \ldots, e_A\rangle$, $A \geqslant 1$, is a sequence of adjacent edges that connect *distinct* vertices in the graph, where $e_i \in E$, $e_i.d = e_{i+1}.s$ for $1 \leqslant i < A$, and the vertices $e_1.s, e_2.s, \ldots, e_A.s$, and $e_A.d$ are distinct. The cardinality of path $\mathcal{P}$, denoted as $|\mathcal{P}|$, is the number of edges in the path. Path $\mathcal{P}' = \langle g_1, g_2, \ldots, g_x\rangle$ is a *sub-path* of $\mathcal{P} = \langle e_1, e_2, \ldots, e_a\rangle$ if $|\mathcal{P}'| \leqslant |\mathcal{P}|$ and there exists an edge sequence in $\mathcal{P}$ such that $g_1 = e_i$, $g_2 = e_{i+1}$, …, and $g_x = e_{i+x-1}$.

Given two paths $\mathcal{P}_i$ and $\mathcal{P}_j$, we use $\mathcal{P}_i \cap \mathcal{P}_j$ to denote the path that is shared by both paths, and we use $\mathcal{P}_i \backslash \mathcal{P}_j$ to denote the sub-path of $\mathcal{P}_i$ that exclude edges in $\mathcal{P}_j$. For instance, 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 $\langle e_1, e_2, e_3\rangle \backslash \langle e_2, e_3, e_4\rangle = \langle e_1\rangle$.

A *trajectory* $\mathcal{T} = \langle p_1, p_2, \ldots, p_C\rangle$ is a sequence of GPS records pertaining to a trip [12], where each p_i is a (*location*, *time*) pair of a vehicle, where $p_i.time < p_j.time$ if $1 \leqslant i < j \leqslant C$. Map matching [29] is able to map GPS records in a trajectory $\mathcal{T}$ to specific locations on different edge and thus it aligns trajectory $\mathcal{T}$ with a path. We call this path as the *path of trajectory* $\mathcal{T}$, denoted as $\mathcal{P}_{\mathcal{T}}$. A trajectory $\mathcal{T}$ *occurred on* path $\mathcal{P}$ *at* time t if and only if path $\mathcal{P}$ is a sub-path of the path of trajectory $\mathcal{P}_{\mathcal{T}}$ and the first GPS record in the first edge in path $\mathcal{P}$ is obtained at t. Figure 2b shows 10 trajectories. For example, trajectory $\mathcal{T}_1$ 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 $\mathcal{P}$ can be obtained from the trajectories that occurred on $\mathcal{P}$. Given a trajectory $\mathcal{T}$ that occurred on path $\mathcal{P}$ at t, the travel time of using $\mathcal{P}$ at t is the difference between the time of the last GPS record and the time of the first GPS record on path $\mathcal{P}$; and the GHG emissions of using $\mathcal{P}$ at t can be computed from the speeds and accelerations when traversing $\mathcal{P}$, which can be derived from the GPS records on path $\mathcal{P}$, and road grades that are available in 3D road networks [23], using vehicular environmental impact models [16,17].

2.2 Accuracy-optimal cost estimation

Recall that problem (i) is to estimate accurately the travel cost distribution of a path using trajectories. The most accurate way of estimating the travel cost distribution of path $\mathcal{P}$ at time t is to employ a sizable set of qualified trajectories. A trajectory $\mathcal{T}$ is qualified if $\mathcal{T}$ occurred on $\mathcal{P}$ at t' and the difference between t' and t is less than a threshold, e.g., 30 min. For instance, if we want to estimate the travel cost distribution of path $\langle e_2, e_3, e_4\rangle$ at 8:05, $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_5$, and $\mathcal{T}_6$ are qualified trajectories, but not $\mathcal{T}_7$ (cf. Fig. 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 $\mathcal{P}$ 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 $\mathcal{P}$ at t and to not overfit to the cost values of a few trajectories, we require the use of more than β qualified trajectories. The effect of parameter β is studied empirically in Sect. 6.

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

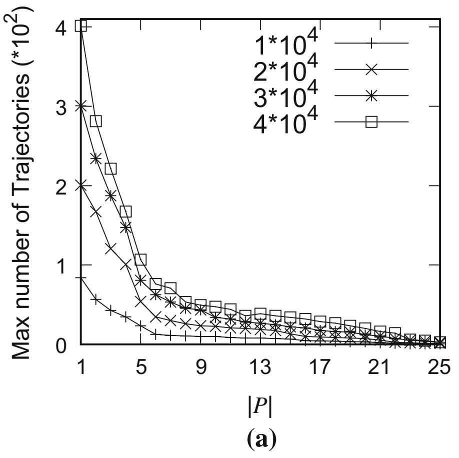
However, this baseline is not always a practical approach, as it is very often inapplicable due to data sparseness. Figure 3 shows that the maximum number of trajectories that occurred on a path decreases rapidly as the cardinality of the path increases, based on two large trajectory collections from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

2.3 Legacy edge-centric paradigm

To contend with the aforementioned data sparseness, existing stochastic route planning uses a graph model that operates at *edge granularity* [6,24,31,38], i.e., the edge-centric model.

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 [38] 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



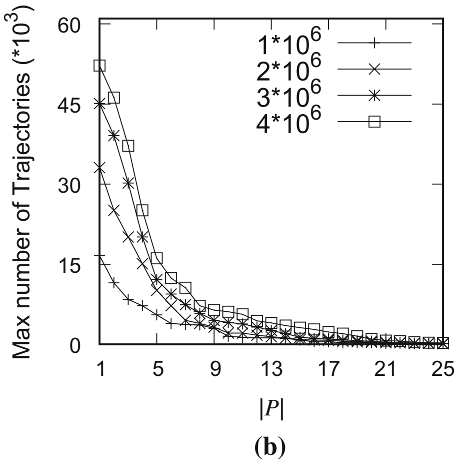


Fig. 3 Data sparseness problem. **a** Aalborg, Denmark. **b** Beijing, China

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 $\mathbf{D}_{LB}(\mathcal{P}, t) = \odot_{e_i \in \mathcal{P}} W_E(e_i, t_{e_i})$, 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 and needs to be progressively updated according to the travel times of e_i's predecessor edges [38].

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 $\mathcal{P}$, 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

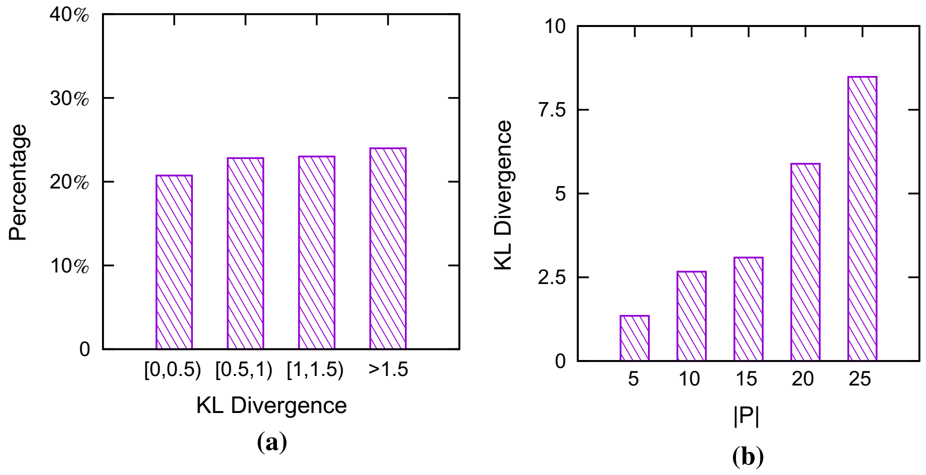


Fig. 4 Examining the independence assumption. **a** $KL(D_{GT}, D_{LB})$. **b** Varying $|\mathcal{P}|$

are, meaning that the convoluted distribution is less accurate. Figure 4a suggests that most of the adjacent edges are not independent.

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 4b 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 edge-centric model is likely to yield inaccurate travel cost distributions, especially for long paths.

3 The PACE model

The analysis of the legacy edge-centric model suggests that the independency assumption does not always hold and that explicitly modeling 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 **PAth CEntric** model—the **PACE** model $G = (V, E, W_P)$. Instead of having an edge weight function W_E in the legacy edge-centric model, the *PACE* model 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 $\mathcal{P}$ and a time t and returns a multi-variate random variable that represents the joint distribution of path $\mathcal{P}$'s edges' travel costs. The joint distribution fully captures the dependency among the travel costs of different edges in path $\mathcal{P}$. 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 α specifies the finest-granularity interval of interest in minutes, e.g., 30 min. We let $V_{\langle e_i \rangle}^{I_j} = p(c_{e_i})$ 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 $V_{\langle e_i \rangle}^{I_j}$, 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 β, the same parameter used in accuracy-optimal baseline in Sect. 2.2, the travel cost values obtained from the qualified trajectories are employed to instantiate the distribution of $V_{\langle e_i \rangle}^{I_j}$, which is the ground-truth distribution.

If the set cardinality does not exceed β, the distribution of $V_{\langle e_i \rangle}^{I_j}$ 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. In such cases, distributions derived from speed limits serve in the role as a ground truth. Thus, in both cases, $V_{\langle e_i \rangle}^{I_j}$ represents the "ground-truth" distribution of unit path $\langle e_i \rangle$ during interval I_j.

Representing univariate distributions We proceed to discuss how to represent a distribution when having more than β trajectories. We represent distributions by histograms because they enable compact approximation of arbitrary, complex distributions. In particular, a one-dimensional histogram is employed to represent a univariate distribution. Gaussian mixture models are also able to represent non-standard travel cost distributions [37,38], but are less compact, especially for representing multi-variate distributions.

From the qualified trajectories, we are able to obtain a multiset of cost values of the form $\langle cost, perc \rangle$, representing that *perc* percentage of the qualified trajectories took cost *cost*.

We call this a *raw cost distribution*. A histogram then approximates the raw cost distribution as a set of pairs: $\{\langle bu_i, pr_i \rangle\}$. A bucket $bu_i = [l, u)$ is a range of travel costs, and pr_i is the probability that the travel cost is in the range, and it holds that $\sum_i pr_i = 1$.

Given the number of buckets b, existing techniques, e.g., V-Optimal [22], 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 nontrivial 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 [33]. First, the multiset of cost values is randomly split into f equal-sized partitions. Each time, we reserve the cost values in one partition, say the kth partition, and use the cost values in the remaining $f - 1$ partitions to generate a histogram with b buckets using V-Optimal, denoted as $H_b^k = \{\langle bu_i, pr_i \rangle\}$. Next, we compute the raw data distribution of the cost values in the reserved partition, denoted as $D^k = \{\langle cost_i, perc_i \rangle\}$. After that, we compute the squared error between H_b^k and D^k: $SE(H_b^k, D^k) = \sum_{c \in costs}(H_b^k[c], D^k[c])^2$. 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 Fig. 1b as an example. Figure 5a 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 5b shows the histogram using $b = 4$ buckets and the original raw data distribution.

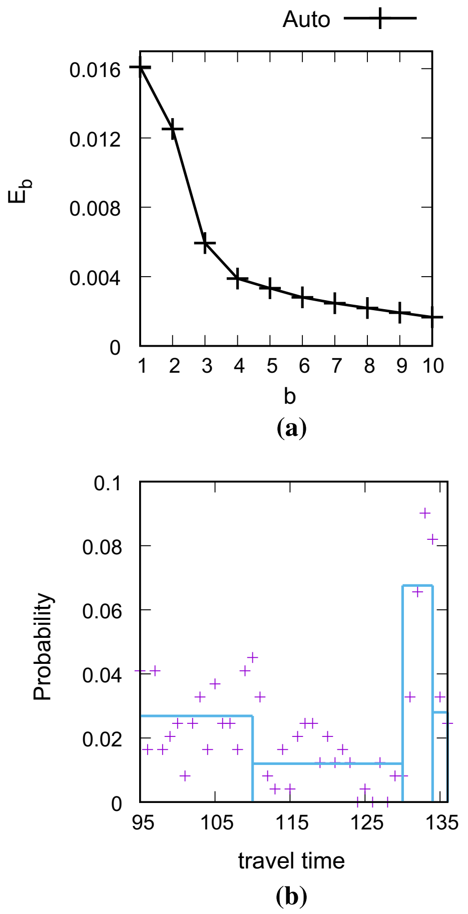


Fig. 5 Identifying the number of buckets. **a** E_b versus b. **b** Raw dist. versus histogram

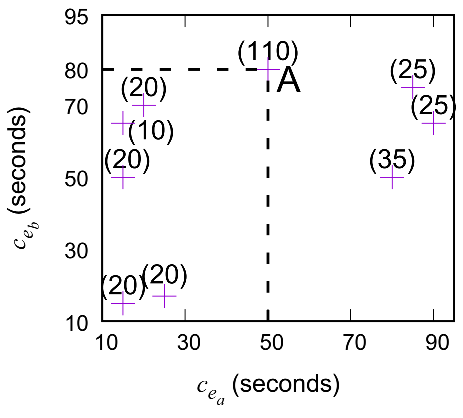
3.2 Instantiating W_P for non-unit paths

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, $k \geqslant 2$, are computed based on the joint distributions of paths with cardinalities $k - 1$.

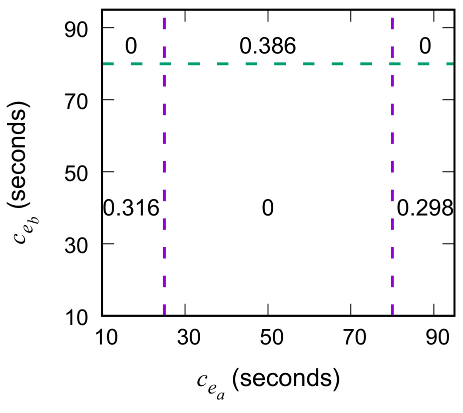
Given two paths $\mathcal{P}_i$ and $\mathcal{P}_j$ with cardinalities $k - 1$, if they share $k - 2$ edges and can be combined into a valid path $\mathcal{P} = \langle e_1, e_2, \ldots, e_k \rangle$ with cardinality k, we check if a time interval I_j exists during which more than β qualified trajectories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

For example, consider two unit paths $\mathcal{P}_a = \langle e_a \rangle$ and $\mathcal{P}_b = \langle e_b \rangle$, which can be combined into a valid path $\mathcal{P} = \langle e_a, e_b \rangle$. Figure 6a 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.

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: $\{\langle hb_i, pr_i \rangle\}$. A hyper-bucket $hb_i =$



(a)



(b)

Fig. 6 An example of multiple dimensional histogram. **a** Actual distribution. **b** 2D histogram

$\langle bu_i^1, \ldots, bu_i^n \rangle$ consists of n buckets that each corresponds to one dimension. Value pr_i equals the probability that the travel costs on multiple edges are in hyper-bucket hb_i, and it holds that $\sum_i pr_i = 1$.

To derive a multi-dimensional histogram, we automatically identify the optimal number of buckets for each dimension using the method from Sect. 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, Fig. 6b shows a 2-dimensional histogram that corresponds to the joint distribution shown in Fig. 6a. 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 $\mathbf{C}_\mathcal{P}$ be a vector of variables $\langle c_{e_1}, c_{e_2}, \ldots, c_{e_k} \rangle$ that corresponds to path $\mathcal{P} = \langle e_1, e_2, \ldots, e_k \rangle$. We use $p(\mathbf{C}_\mathcal{P}) = p(c_{e_1}, c_{e_2}, \ldots, c_{e_k})$ to denote the ground-truth joint distribution of using path $\mathcal{P}$ at time t. Following the same idea in Sect. 2.2, we regard the joint distribution obtained from at least β qualified trajectories as the ground-truth joint distribution.

Given a set of trajectories, we let $Paths_\beta$ be a set of non-unit paths where each path has at least β qualified trajectories. Note that given different sets of trajectories that occurred on a same road network, $Paths_\beta$ may contain different non-unit paths.

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

$$W_P(\mathcal{P}, t) = V_\mathcal{P}^{I_j} = p(\mathbf{C}_\mathcal{P}) = \begin{cases} p(c_{e_i}), & \mathcal{P} = \langle e_i \rangle \text{ is a unit path} \\ p(c_{e_1}, c_{e_2}, \ldots, c_{e_k}), & \mathcal{P} = \langle e_1, e_2, \ldots, e_k \rangle \in Paths_\beta \end{cases} \quad (1)$$

where (i) if $\mathcal{P}$ is a unit path, the path weight function returns the ground-truth distribution $p(\mathbf{C}_\mathcal{P}) = p(c_{e_i})$, represented as a one-dimensional histogram; (ii) if $\mathcal{P}$ is a non-unit path, the path weight function returns the ground-truth joint distribution $p(\mathbf{C}_\mathcal{P}) = p(c_{e_1}, c_{e_2}, \ldots, c_{e_k})$, represented as a multi-dimensional histogram.

The random variables $\{V_\mathcal{P}^{I_j}\}$ that are maintained in the path weight function W_P are called *instantiated random variables*. The *rank* of a variable $V_\mathcal{P}^{I_j}$ is the cardinality of its path $|\mathcal{P}|$. In the legacy edge-centric 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 *PACE* model, the random variables with rank larger than one fully capture the distribution dependencies among the edges in a path.

4 Path cost distribution computation in *PACE*

Given any path $\mathcal{P}$ and a departure time t, we perform travel cost distribution estimation using the instantiated *PACE*, in two steps. First, the joint distribution of path $\mathcal{P}$, which models the travel cost dependency among edges in $\mathcal{P}$, is computed. Second, the cost distribution of path $\mathcal{P}$, which captures the cost distribution of traversing the whole path $\mathcal{P}$, is derived based on the path's joint distribution.

4.1 The joint distribution of a path

The ground-truth joint distribution of path $\mathcal{P} = \langle e_1, e_2, \ldots, e_n \rangle$ at t is denoted as $p(\mathbf{C}_\mathcal{P}) = p(c_{e_1}, c_{e_2}, \ldots, c_{e_n})$, where

c_{e_i} ($1 \leqslant i \leqslant n$) is a random variable representing the travel cost distribution of path $\langle e_i \rangle$.

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

In contrast, if $W_P(\mathcal{P}, t)$ returns an empty result, this means that there does not exist at least β qualified trajectories on path $\mathcal{P}$ 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 Fig. 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 $\hat{p}(c_{e_1}, c_{e_2}, \ldots, c_{e_n})$ based on the ground-truth distributions of path $\mathcal{P}$'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 $\mathcal{P}$ is a sequence of paths $DE = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$, $k > 1$, that satisfies the following *spatial conditions*:

(1) Each path $\mathcal{P}_i \in DE$ is a sub-path of $\mathcal{P}$;
(2) All paths in DE together cover $\mathcal{P}$, i.e., $\mathcal{P}_1 \cup \mathcal{P}_2 \cup \ldots \cup \mathcal{P}_k = \mathcal{P}$;
(3) A path $\mathcal{P}_i$ is not a sub-path of another path $\mathcal{P}_j$, where $1 \leqslant i, j \leqslant k$ and $i \neq j$;
(4) A path $\mathcal{P}_i$ appears before another path $\mathcal{P}_j$ where $1 \leqslant i < j \leqslant k$ if and only if the first edge of $\mathcal{P}_i$ appears earlier than the first edge of $\mathcal{P}_j$ in path $\mathcal{P}$.

A path $\mathcal{P}$ may have more than one decomposition. For instance, consider path $\mathcal{P} = \langle e_1, e_2, e_3, e_4, e_5 \rangle$. The following path sequences are possible decompositions for the path.
$DE_1 = (\langle e_1 \rangle, \langle e_2 \rangle, \langle e_3 \rangle, \langle e_4 \rangle, \langle e_5 \rangle)$,
$DE_2 = (\langle e_1, e_2, e_3 \rangle, \langle e_2, e_3, e_4 \rangle, \langle e_5 \rangle)$,
$DE_3 = (\langle e_1, e_2, e_3 \rangle, \langle e_3, e_4 \rangle, \langle e_5 \rangle)$.

Next, we introduce a *coarser* relationship between two path decompositions DE_i and DE_j. We define DE_i to be coarser than DE_j if for each path $\mathcal{P}_b \in DE_j$, there is a path $\mathcal{P}_a \in DE_i$ such that $\mathcal{P}_b$ is a sub-path of $\mathcal{P}_a$ and at least one $\mathcal{P}_b \neq \mathcal{P}_a$.

To illustrate, DE_2 is coarser than DE_3 because $\langle e_1, e_2, e_3 \rangle$, $\langle e_3, e_4 \rangle$, $\langle e_5 \rangle$ are sub-paths of paths $\langle e_1, e_2, e_3 \rangle$, $\langle e_2, e_3, e_4 \rangle$, $\langle e_5 \rangle$, respectively; and $\langle e_2, e_3, e_4 \rangle$ from DE_2 is different from $\langle e_3, e_4 \rangle$ from DE_3. Similarly, DE_2 is coarser than DE_1 as well.

4.1.2 Distribution estimation using decompositions

Following the principles of decomposable models [9,26], a decomposition of path $\mathcal{P}$ corresponds to a set of independence assumptions among the cost variables in $\mathbf{C}_{\mathcal{P}}$. Specifically, given a decomposition $DE = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$, for any two paths $\mathcal{P}_i$ and $\mathcal{P}_j$ in the decomposition, where $1 \leqslant i, j \leqslant k$, we have the following cases.

(i) If $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$, this indicates that the cost variables in $\mathbf{C}_{\mathcal{P}_i}$ are independent of the cost variables in $\mathbf{C}_{\mathcal{P}_j}$;
(ii) If $\mathcal{P}_i \cap \mathcal{P}_j \neq \emptyset$, this indicates that the cost variables in $\mathbf{C}_{\mathcal{P}_i \setminus \mathcal{P}_j}$ are conditionally independent of the cost variables in $\mathbf{C}_{\mathcal{P}_j \setminus \mathcal{P}_i}$ given the cost variables in $\mathbf{C}_{\mathcal{P}_i \cap \mathcal{P}_j}$.

Based on the above, given a decomposition DE, 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 DE based on Bayes' theorem.

Formally, given a decomposition $DE = (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k)$, the joint distribution of path $\mathcal{P}$ is estimated according to Eq. 3.

$$\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}}) = \frac{\prod_{\mathcal{P}_i \in P_X} p(\mathbf{C}_{\mathcal{P}_i})}{\prod_{\mathcal{P}_j \in P_Y} p(\mathbf{C}_{\mathcal{P}_j})}, \tag{2}$$

$$\hat{pr}_{DE}(\mathbf{C}_{\mathcal{P}}) = \frac{\prod_{m=1}^{k} pr(\mathbf{C}_{\mathcal{P}_m})}{\prod_{m=1}^{k-1} pr(\mathbf{C}_{\mathcal{P}_m \cap \mathcal{P}_{m+1}})}, \tag{3}$$

where path set $P_X = \bigcup_{1 \leqslant i \leqslant k} \mathcal{P}_i$ consists of all paths in DE and $P_Y = \bigcup_{2 \leqslant i \leqslant k} \mathcal{P}_i \cap \mathcal{P}_{i-1}$ consists of the shared paths between all adjacent paths in DE.

In the running example, DE_1 assumes all cost variables c_{e_1}, c_{e_2}, c_{e_3}, c_{e_4}, and c_{e_5} are independent because no paths in DE_1 intersect. According to Eq. 3, we have $\hat{p}_{DE_1}(\mathbf{C}_{\mathcal{P}}) = p(c_{e_1}) \cdot p(c_{e_2}) \cdot p(c_{e_3}) \cdot p(c_{e_4}) \cdot p(c_{e_5})$. This corresponds to the legacy edge-centric 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 $\langle e_1, e_2, e_3 \rangle \cap \langle e_2, e_3, e_4 \rangle = \langle e_2, e_3 \rangle$; and c_{e_5} is independent of all the other cost variables because $\langle e_5 \rangle$ does not intersect with other paths in DE. Thus, Eq. 3 gives us $\hat{p}_{DE_2}(\mathbf{C}_{\mathcal{P}}) = \frac{p(c_{e_1}, c_{e_2}, c_{e_3}) \cdot p(c_{e_2}, c_{e_3}, c_{e_4}) \cdot p(c_{e_5})}{p(c_{e_2}, c_{e_3})}$.

Given a path $\mathcal{P}$, we have more than one path decomposition. For each decomposition, we are able to derive an estimated joint distribution according to Eq. 3. The challenge is to identify the decomposition that gives the most accurate estimation.

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

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

Proof Since path $\mathcal{P}'$ is a sub-path of path $\mathcal{P}$, path $\mathcal{P}$ can be represented as $\mathcal{P} = \mathcal{P}_s \circ \mathcal{P}' \circ \mathcal{P}_e$, where $\mathcal{P}_s$ and $\mathcal{P}_e$ are the possibly empty paths before and after path $\mathcal{P}'$, and $\circ$ denotes concatenation. Thus, the cost variables in $\mathbf{C}_{\mathcal{P}_s}$, $\mathbf{C}_{\mathcal{P}'}$, or $\mathbf{C}_{\mathcal{P}_e}$ must be a subset of the cost variables in $\mathbf{C}_{\mathcal{P}}$. In addition, we have $\mathbf{C}_{\mathcal{P}_s} \cup \mathbf{C}_{\mathcal{P}'} \cup \mathbf{C}_{\mathcal{P}_e} = \mathbf{C}_{\mathcal{P}}$. Based on the above, we get

$$\begin{aligned}
&\sum_{\mathbf{C}_{\mathcal{P}}} p(\mathbf{C}_{\mathcal{P}}) \log p(\mathbf{C}_{\mathcal{P}'}) \\
&\quad = \sum_{\mathbf{C}_{\mathcal{P}_s}, \mathbf{C}_{\mathcal{P}'}, \mathbf{C}_{\mathcal{P}_e}} p(\mathbf{C}_{\mathcal{P}_s}, \mathbf{C}_{\mathcal{P}'}, \mathbf{C}_{\mathcal{P}_e}) \log p(\mathbf{C}_{\mathcal{P}'}) \\
&\quad = \sum_{\mathbf{C}_{\mathcal{P}'}} \left(\log p(\mathbf{C}_{\mathcal{P}'}) \sum_{\mathbf{C}_{\mathcal{P}_s}, \mathbf{C}_{\mathcal{P}_e}} p(\mathbf{C}_{\mathcal{P}_s}, \mathbf{C}_{\mathcal{P}'}, \mathbf{C}_{\mathcal{P}_e}) \right) \\
&\quad = \sum_{\mathbf{C}_{\mathcal{P}'}} p(\mathbf{C}_{\mathcal{P}'}) \log p(\mathbf{C}_{\mathcal{P}'}) = -H(\mathbf{C}_{\mathcal{P}'}) \qquad \square
\end{aligned}$$

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

Proof

$$\begin{aligned}
&KL(p(\mathbf{C}_{\mathcal{P}}), \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})) \\
&\quad = \sum_{\mathbf{C}_{\mathcal{P}}} p(\mathbf{C}_{\mathcal{P}}) \log \left(\frac{p(\mathbf{C}_{\mathcal{P}})}{\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})} \right) \\
&\quad = -H(\mathbf{C}_{\mathcal{P}}) - \sum_{\mathbf{C}_{\mathcal{P}}} p(\mathbf{C}_{\mathcal{P}}) \log \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}}) \\
&\quad = -H(\mathbf{C}_{\mathcal{P}}) - \sum_{\mathbf{C}_{\mathcal{P}}} p(\mathbf{C}_{\mathcal{P}}) \left(\sum_{\mathcal{P}_i \in P_X} \log p(\mathbf{C}_{\mathcal{P}_i}) \right. \\
&\qquad \left. - \sum_{\mathcal{P}_j \in P_Y} \log p(\mathbf{C}_{\mathcal{P}_j}) \right) \quad \text{(due to Eq. 3)} \\
&\quad = -H(\mathbf{C}_{\mathcal{P}}) - \sum_{\mathcal{P}_i \in P_X} \left(\sum_{\mathbf{C}_{\mathcal{P}}} p(\mathbf{C}_{\mathcal{P}}) \log p(\mathbf{C}_{\mathcal{P}_i}) \right) \\
&\qquad + \sum_{\mathcal{P}_j \in P_Y} \left(\sum_{\mathbf{C}_{\mathcal{P}}} p(\mathbf{C}_{\mathcal{P}}) \log p(\mathbf{C}_{\mathcal{P}_j}) \right) \\
&\quad = -H(\mathbf{C}_{\mathcal{P}}) + \sum_{\mathcal{P}_i \in P_X} H(\mathbf{C}_{\mathcal{P}_i}) \\
&\qquad - \sum_{\mathcal{P}_j \in P_Y} H(\mathbf{C}_{\mathcal{P}_j}) \quad \text{(due to Th. 1)} \\
&\quad = H_{DE}(\mathbf{C}_{\mathcal{P}}) - H(\mathbf{C}_{\mathcal{P}}) \qquad \square
\end{aligned}$$

	$Group(\mathcal{P}_i)$	$\mathcal{P}_i^{(a)}$	$\mathcal{P}_i^{(b)}$	$\mathcal{P}_i^{(c)}$
$\mathcal{P}_1 = \langle e_1, e_2, e_3 \rangle$	$\{\langle e_1 \rangle, \langle e_2 \rangle, \langle e_3 \rangle\}$	$\emptyset$	$\langle e_1, e_2, e_3 \rangle$	$\emptyset$
$\mathcal{P}_2 = \langle e_2, e_3, e_4 \rangle$	$\{\langle e_4 \rangle\}$	$\langle e_2, e_3 \rangle$	$\langle e_4 \rangle$	$\emptyset$
$\mathcal{P}_3 = \langle e_5 \rangle$	$\{\langle e_5 \rangle\}$	$\emptyset$	$\langle e_5 \rangle$	$\emptyset$

Fig. 7 A running example on DE_1 and DE_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 Assume $DE = \langle \mathcal{P}_1, \ldots, \mathcal{P}_m \rangle$ and $DE' = \langle \mathcal{P}'_1, \ldots, \mathcal{P}'_n \rangle$ where $m \leqslant n$. Since Theorem 2 holds for both DE and DE', we need to prove that the entropy difference $\Delta = H_{DE}(\mathbf{C}_{\mathcal{P}}) - H_{DE'}(\mathbf{C}_{\mathcal{P}}) < 0$, and we have

$$\begin{aligned}
\Delta = &\sum_{i=1}^{m} H(\mathbf{C}_{\mathcal{P}_i}) - \sum_{i=2}^{m} H(\mathbf{C}_{\mathcal{P}_{i-1} \cap \mathcal{P}_i}) \\
&- \left(\sum_{i=1}^{n} H(\mathbf{C}_{\mathcal{P}'_i}) - \sum_{i=2}^{n} H(\mathbf{C}_{\mathcal{P}'_{i-1} \cap \mathcal{P}'_i}) \right) \quad \text{(due to Eq. 2)}
\end{aligned}$$

Since DE is coarser than DE', we are able to partition the paths in DE' into groups. For each path $\mathcal{P}_i \in DE$, we have a corresponding group $Group(\mathcal{P}_i) = \{\mathcal{P}'_j, \ldots, \mathcal{P}'_{j+k}\}$ such that the paths in $Group(\mathcal{P}_i)$ are sub-paths of $\mathcal{P}_i$. We use the example with DE_1 and DE_2 to illustrate. DE_2 is coarser than DE_1 (cf. Sect. 4.1.1 and the group for each path in DE_2 is shown in Fig. 7.

Next, we introduce the path $\mathcal{P}_i^{(b)} = \cup_{\mathcal{P}' \in Group(\mathcal{P}_i)} \mathcal{P}'$ that is the union of the paths in $Group(\mathcal{P}_i)$. Path $\mathcal{P}_i^{(b)}$ must be a sub-path of $\mathcal{P}_i$. Without loss of generality, assume that $\mathcal{P}_i$ can be represented as $\mathcal{P}_i^{(a)} \cup \mathcal{P}_i^{(b)} \cup \mathcal{P}_i^{(c)}$ where $\mathcal{P}_i^{(a)} = \mathcal{P}_{i-1} \cap \mathcal{P}_i$ and $\mathcal{P}_i^{(c)} = \mathcal{P}_i \cap \mathcal{P}_{i+1}$. It is possible that $\mathcal{P}_i^{(a)}$ or $\mathcal{P}_i^{(c)}$ or both are empty. Figure 7 illustrates this using DE_1 and DE_2.

Based on the above, Δ, i.e., the entropy difference between $H_{DE}(\mathbf{C}_{\mathcal{P}})$ and $H_{DE'}(\mathbf{C}_{\mathcal{P}})$, can be computed as the sum of m sub-differences, where each sub-difference Δ_i corresponds to the entropy difference between path $\mathcal{P}_i$ in DE and the paths in $Group(\mathcal{P}_i) = \{\mathcal{P}'_j, \ldots, \mathcal{P}'_{j+k}\}$. Formally, we have $\Delta = \sum_{i=1}^{m} \Delta_i$, where

$$\Delta_i = \underbrace{H(\mathbf{C}_{\mathcal{P}_i}) - \left(H\left(\mathbf{C}_{\mathcal{P}_i^{(a)}}\right) + H\left(\mathbf{C}_{\mathcal{P}_i^{(c)}}\right)\right)}_{\text{Due to Eq. 2}} - H(Group(\mathcal{P}_i));$$

$$H(Group(\mathcal{P}_i) = \underbrace{\left(\sum_{d=0}^{k} H\left(\mathbf{C}_{\mathcal{P}'_{j+d}}\right) - \sum_{d=1}^{k} H\left(\mathbf{C}_{\mathcal{P}'_{j+d} \cap \mathcal{P}'_{j+d-1}}\right)\right)}_{\text{Due to Eq. 2 on } Group(\mathcal{P}_i)}$$

According to the definition of path $\mathcal{P}_i^{(b)}$, either $\mathcal{P}_i^{(b)}$ itself is the singleton path in $Group(\mathcal{P}_i)$ or the paths in $Group(\mathcal{P}_i)$ are able to form a decomposition of path $\mathcal{P}_i^{(b)}$. No matter which is the case, according to Theorem 2 and the fact that a KL-divergence value is always non-negative, we have $H(Group(\mathcal{P}_i)) \geqslant H(\mathbf{C}_{\mathcal{P}_i^{(y)}})$. Putting this inequality back to Δ_i, we have

$$\Delta_i \leqslant H(\mathbf{C}_{\mathcal{P}_i}) - \left(H\left(\mathbf{C}_{\mathcal{P}_i^{(a)}}\right) + H\left(\mathbf{C}_{\mathcal{P}_i^{(b)}}\right) + H\left(\mathbf{C}_{\mathcal{P}_i^{(c)}}\right)\right)$$

Next, either (1) $\mathcal{P}_i^{(a)}$, $\mathcal{P}_i^{(b)}$, and $\mathcal{P}_i^{(c)}$ are able to form a decomposition of $\mathcal{P}_i$, when $\mathcal{P}_i^{(a)}$ or $\mathcal{P}_i^{(c)}$ is not empty; or (2) $\mathcal{P}_i^{(b)}$ itself is $\mathcal{P}_i$, when both $\mathcal{P}_i^{(a)}$ and $\mathcal{P}_i^{(c)}$ are empty. Since DE is coarser than DE', case (1) should appear at least once. According to Theorem 2 and the fact that KL-divergence values are non-negative (which applies to both case (1) and case (2)) and a KL-divergence value between two different distributions is positive (which applies to case (1)), we have $H(\mathbf{C}_{\mathcal{P}_i^{(a)}}) + H(\mathbf{C}_{\mathcal{P}_i^{(b)}}) + H(\mathbf{C}_{\mathcal{P}_i^{(c)}}) \geqslant H(\mathbf{C}_{\mathcal{P}_i})$ for every $1 \leqslant i \leqslant m$; and there exist at least one i such that $H(\mathbf{C}_{\mathcal{P}_i^{(a)}}) + H(\mathbf{C}_{\mathcal{P}_i^{(b)}}) + H(\mathbf{C}_{\mathcal{P}_i^{(c)}}) > H(\mathbf{C}_{\mathcal{P}_i})$. This implies that $\Delta_i \leqslant 0$ for every $1 \leqslant i \leqslant m$, and there exists at least one i such that $\Delta_i < 0$. This leads to the conclusion that $\Delta = \sum_{i=1}^{m} \Delta_i < 0$. $\square$

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 $\mathcal{P}$ and a departure time t, we proceed to identify the coarsest decomposition for $\mathcal{P}$ 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 $\mathcal{P}$ and *temporally* relevant to the departure time t; we then 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 $V_{\mathcal{P}_i}^{I_j}$, which represents the joint distribution of path $\mathcal{P}_i$ during interval I_j. A random variable $V_{\mathcal{P}_i}^{I_j}$ is spatially relevant to the query path $\mathcal{P}$ if the variable's path $\mathcal{P}_i$ is a sub-path of $\mathcal{P}$. Next, we test whether a spatially relevant random variable $V_{\mathcal{P}_i}^{I_j}$ is also temporally relevant to t.

We distinguish two cases. We first consider the case where the first edge in the variable's path $\mathcal{P}_i$ is the same as the first edge in the query path $\mathcal{P}$. This case is simple because the departure time on $\mathcal{P}$ is also the departure time on $\mathcal{P}_i$, as both paths start from the same edge. Thus, we just need to test if the departure time t is during I_j, i.e., $t \in I_j$. The second case is complicated because the departure time on path $\mathcal{P}_i$ is no longer the original departure time t. 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 $\langle e_1, e_2\rangle$ 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 $[t + V_{\langle e_1\rangle}^{I_j}.min, t + V_{\langle e_1\rangle}^{I_j}.max]$, where $V_{\langle e_1\rangle}^{I_j}.min$ and $V_{\langle e_1\rangle}^{I_i}.max$ denote the minimum and maximum travel times of traversing e_1, which are recorded in the random variable $V_{\langle e_1\rangle}^{I_j}$.

Formally, given a time interval $[t_s, t_e]$ and a random variable $V_{\langle e_k\rangle}^{I_j}$, we define the *shifted-and-enlarged* interval as $SAE([t_s, t_e], V_{\langle e_k\rangle}^{I_j}) = [t_s + V_{\langle e_k\rangle}^{I_j}.min, t_e + V_{\langle e_k\rangle}^{I_j}.max]$.

Given a sub-path $\mathcal{P}_i$ of path $\mathcal{P}$, assume that the first edge in $\mathcal{P}_i$ is the kth edge in $\mathcal{P}$, where $2 \leqslant k \leqslant |\mathcal{P}|$. The updated departure time on $\mathcal{P}_i$ based on the original departure time t is UI_k, as defined in Eq. 4.

$$UI_k = \begin{cases} SAE([t, t], V_{\langle e_1\rangle}^{I_j}), & \text{if } k = 2; \\ SAE(UI_{k-1}, V_{\langle e_{k-1}\rangle}^{I_j}), & \text{otherwise}; \end{cases} \tag{4}$$

Given a spatially relevant random variable $V_{\mathcal{P}_i}^{I_j}$, if I_j intersects $\mathcal{P}_i$'s updated departure time interval UI_k according to Eq. 4; then, it is temporally relevant; otherwise, it is not temporally relevant. For a given sub-path $\mathcal{P}_i$, if multiple variables $V_{\mathcal{P}_i}^{I_j}$, $(j = 1, 2, \ldots, m)$ are temporally relevant, the one with the largest overlap is selected, i.e., the one where $I_j = \arg\max_{j \in [1,m]} \frac{|I_j \cap UI_k|}{|UI_k|}$.

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 $\mathcal{P}$ 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.

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

Table 1 Example candidate array

	rank = 1	*rank* = 2	*rank* = 3	*rank* = 4
e_1	$V_{\langle e_1\rangle}$	$V_{\langle e_1,e_2\rangle}$	$V_{\langle e_1,e_2,e_3\rangle}$	$\boldsymbol{V_{\langle e_1,e_2,e_3,e_4\rangle}}$
e_2	$V_{\langle e_2\rangle}$	$V_{\langle e_2,e_3\rangle}$	$V_{\langle e_2,e_3,e_4\rangle}$	
e_3	$V_{\langle e_3\rangle}$	$V_{\langle e_3,e_4\rangle}$		
e_4	$V_{\langle e_4\rangle}$	$\boldsymbol{V_{\langle e_4,e_5\rangle}}$		
e_5	$V_{\langle e_5\rangle}$			

(3) (cf. Sect. 4.1.2). The remaining paths constitute the coarsest decomposition. The procedure is summarized in Algorithm 1.

To illustrate the procedure, consider the example shown in Table 1. In the beginning, we consider path $\langle e_1, e_2, e_3, e_4\rangle$ and add it to DE_{coa}. Next, since $\langle e_2, e_3, e_4\rangle$ and $\langle e_3, e_4\rangle$ are sub-paths of $\langle e_1, e_2, e_3, e_4\rangle$, both are omitted. Then, $\langle e_4, e_5\rangle$ is added to DE_{coa}. Since $\langle e_5\rangle$ is a sub-path of $\langle e_4, e_5\rangle$, it is omitted. Finally, the coarsest decomposition is $DE_{coa} = (\langle e_1, e_2, e_3, e_4\rangle, \langle e_4, e_5\rangle)$, which is shown in bold in Table 1.

Algorithm 1: Identify the Coarsest Decomposition

```
Input  : Query Path $\mathcal{P}$, Departure Time $t$
Output: The Coarsest Decomposition $DE_{coa}$
1 Identify a set of random variables STRV that are spatially
  relevant to path $\mathcal{P}$ 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 $DE_{coa} \leftarrow null$;
5 for $k = 1$; $k \leqslant |\mathcal{P}|$; $k++$ do
6     Identify the random variable $V^{I_j}_{\mathcal{P}_k}$ with the highest rank from
      the $k$-th row;
7     if $\mathcal{P}_k$ is not a sub-path of any path in $DE_{coa}$ then
8         Append $\mathcal{P}_k$ 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 $\mathcal{P}$.

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 $p(V_{\mathcal{P}})$, where $V_{\mathcal{P}}$ is a univariate random variable indicating the travel cost of path $\mathcal{P}$. We proceed to derive $p(V_{\mathcal{P}})$ based on the joint distribution of a path $\hat{p}_{DE_{coa}}(\mathbf{C}_{\mathcal{P}})$ using

$$p(V_{\mathcal{P}} = x) = \sum_{c_1+\cdots+c_n=x} \hat{p}_{DE_{coa}}(c_{e_1} = c_1, \ldots, c_{e_n} = c_n).$$

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

Recall that a multi-dimensional histogram is of the form $\{\langle hb_i, pr_i\rangle\}$, where hyper-bucket $hb_i = \langle bu_i^1, \ldots bu_i^n\rangle$ consists of n buckets, each corresponding to one dimension. For each hyper-bucket $hb_i = \langle bu_i^1, \ldots bu_i^n\rangle$, we derive a bucket bu_i whose upper (lower) bound is the sum of the upper (lower) bounds of the buckets in the hyper-bucket, i.e., $bu_i = [\sum_{j=1}^n bu_i^j.l, \sum_{j=1}^n bu_i^j.u)$. Thus, we get a one-dimensional histogram $\{\langle bu_i, pr_i\rangle\}$.

The buckets in the obtained one-dimensional histogram may overlap. We need to rearrange the buckets such that they are disjoint and update their corresponding probabilities. We check each pair of buckets as follows. If two buckets bu_i and bu_j are disjoint, keep both buckets. If buckets bu_i and bu_j overlap, range $[\min(bu_i.l, bu_j.l), \max(bu_i.u, bu_j.u))$ is split into three buckets according to the increasing order of $bu_i.l$, $bu_j.l$, $bu_i.u$, and $bu_j.u$, 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 8 shows a running example on the aforementioned procedure on path $\mathcal{P}_1 = \langle e_1, e_2\rangle$. The first table in Fig. 8 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 $\mathcal{P}_1$, 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 and 40 s is 0.3. Next, the second table in Fig. 8 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).

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

	$c_{e_1} \in [20, 30)$	$c_{e_1} \in [30, 50)$
$c_{e_2} \in [20, 40)$	0.30	0.25
$c_{e_2} \in [40, 60)$	0.20	0.25

[40, 70)	[50, 90)	[60, 90)	[70, 110)
0.30	0.25	0.20	0.25

[40, 50)	[50, 60)	[60, 70)	[70, 90)	[90, 110)
0.1000	0.1625	0.2292	0.3833	0.1250

Fig. 8 From joint distribution to final cost distribution

given probability $\frac{|[50,70)|}{|[40,70)|} \cdot 0.3 + \frac{|[50,70)|}{|[50,90)|} \cdot 0.25 = 0.325$, and bucket [70, 90) is associated with probability $\frac{|[70,90)|}{|[50,90)|} \cdot 0.25 = 0.125$. Bucket [40, 50) does not overlap with other buckets, its adjusted probability is the final probability. Since buckets [50, 70) and [70, 90) still overlap with the next bucket [60, 70), their buckets should be further rearranged and their probabilities should be adjusted. The final cost distribution is shown in the third table in Fig. 8.

5 Path finding in *PACE*

To conduct path finding, or routing, in *PACE*, we consider two alternative approaches. First, we show that it is possible to integrate existing edge-centric stochastic path finding algorithms seamlessly into *PACE*. Second, we propose a new stochastic routing algorithm that fully exploits the foundation for improved accuracy offered by *PACE*.

Since both approaches apply a "path + another edge" pattern to explore candidate paths, a path cost distribution estimation method must satisfy the so-called "incremental property" that enables reuse of the cost distribution of a partial path when computing the cost distribution of a path that extends the existing path. We proceed to prove that the path cost distribution estimation method in *PACE* satisfies the incremental property in Sect. 5.1 and then detail the two alternative path finding approaches in Sects. 5.2 and 5.3.

5.1 Incremental property

Consider a path $\mathcal{P} = \langle e_1, e_2, \ldots, e_k \rangle$ and a path $\mathcal{P}' = \langle e_1, e_2, \ldots, e_k, e_{k+1} \rangle$ that extends $\mathcal{P}$ by edge e_{k+1}. The ***incremental property*** requires that the cost distribution of path $\mathcal{P}'$ can be computed by reusing the cost distribution of path $\mathcal{P}$ rather than having to be re-computed from scratch.

We proceed to prove that *PACE* satisfies the incremental property. Consider path $\mathcal{P} = \langle e_1, e_2, \ldots, e_k \rangle$ and suppose that its coarsest decomposition is $DE = \langle \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_j \rangle$, where $k \geqslant j$. Then, the estimated cost of path $\mathcal{P}$ is $\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}}) = \frac{\prod_{i=1}^{j} p(\mathbf{C}_{\mathcal{P}_i})}{\prod_{i=2}^{j} p(\mathbf{C}_{\mathcal{P}_{i-1} \cap \mathcal{P}_i})}$ (cf. Sect. 4.1.2).

Having extended path $\mathcal{P}$ with edge e_{k+1}, we get path $\mathcal{P}' = \langle e_1, e_2, \ldots, e_k, e_{k+1} \rangle$. Suppose that its corresponding coarsest decomposition is DE' and thus has estimated cost $\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'})$. The incremental property amounts to requiring that $\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'})$ can be computed based on $\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})$. In order to show the incremental property, we distinguish three cases.

Case 1 There does not exist an instantiated random variable $V_{\mathcal{P}^\star}$ whose corresponding path $\mathcal{P}^\star$ covers both e_k and e_{k+1}. In this case, we append unit path $\mathcal{P}^* = \langle e_{k+1} \rangle$ to DE to get DE'. Thus, we have $DE' = \langle \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_j, \mathcal{P}^* \rangle$. Consequently, we have

$$\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'}) = \boxed{\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})} \cdot p(\mathbf{C}_{\mathcal{P}^*})$$

Here, path $\mathcal{P}$'s cost distribution $\boxed{\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})}$ can be reused.

Case 2 There exists an instantiated random variable $V_{\mathcal{P}^\star}$ whose path $\mathcal{P}^\star$ covers both e_k and e_{k+1}. Further, $\mathcal{P}^\star$ covers the last path $\mathcal{P}_j$ in DE. In this case, by replacing $\mathcal{P}_j$ by $\mathcal{P}^\star$, we get $DE' = \langle \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{j-1}, \mathcal{P}^\star \rangle$. Then we have

$$\begin{aligned}
\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'}) &= \frac{\left(\prod_{i=1}^{j-1} p\left(\mathbf{C}_{\mathcal{P}_i}\right)\right) \cdot p(\mathbf{C}_{\mathcal{P}^*})}{\left(\prod_{i=2}^{j-1} p\left(\mathbf{C}_{\mathcal{P}_{i-1} \cap \mathcal{P}_i}\right)\right) \cdot p\left(\mathbf{C}_{\mathcal{P}_{j-1} \cap \mathcal{P}^*}\right)} \\
&= \frac{\left(\prod_{i=1}^{j} p\left(\mathbf{C}_{\mathcal{P}_i}\right)\right) \cdot p(\mathbf{C}_{\mathcal{P}^*}) \cdot p\left(\mathbf{C}_{\mathcal{P}_j \cap \mathcal{P}_{j-1}}\right)}{\left(\prod_{i=2}^{j} p\left(\mathbf{C}_{\mathcal{P}_{i-1} \cap \mathcal{P}_i}\right)\right) \cdot p\left(\mathbf{C}_{\mathcal{P}_{j-1} \cap \mathcal{P}^*}\right) \cdot p\left(\mathbf{C}_{\mathcal{P}_j}\right)} \\
&= \boxed{\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})} \cdot \frac{p(\mathbf{C}_{\mathcal{P}^\star}) \cdot p\left(\mathbf{C}_{\mathcal{P}_j \cap \mathcal{P}_{j-1}}\right)}{p\left(\mathbf{C}_{\mathcal{P}_{j-1} \cap \mathcal{P}^\star}\right) \cdot p\left(\mathbf{C}_{\mathcal{P}_j}\right)}
\end{aligned}$$

Again, path $\mathcal{P}$'s cost distribution $\boxed{\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})}$ can also be reused.

Case 3 An instantiated random variable $V_{\mathcal{P}^\star}$ exists whose path $\mathcal{P}^\star$ covers both e_k and e_{k+1}. However, $\mathcal{P}^\star$ does not cover the last path $\mathcal{P}_j$ in DE. In this case, by simply appending $\mathcal{P}^\star$ to DE, we get $DE' = \langle \mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_j, \mathcal{P}^\star \rangle$.

Thus, we have

$$\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'}) = \boxed{\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})} \cdot \frac{p(\mathbf{C}_{\mathcal{P}^\star})}{p(\mathbf{C}_{\mathcal{P}^\star \cap \mathcal{P}_j})}$$

Once again, path $\mathcal{P}$'s cost distribution $\boxed{\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})}$ can be reused.

Examples Consider a path $\mathcal{P} = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ and assume that the coarsest decomposition is $DE = \langle \langle e_1, e_2, e_3 \rangle, \langle e_3, e_4 \rangle, \langle e_4, e_5 \rangle \rangle$. We have the following estimated path cost distribution based on DE:

$$\begin{aligned}
\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}}) &= \frac{\prod_{i=1}^{j} p\left(\mathbf{C}_{\mathcal{P}_i}\right)}{\prod_{i=2}^{j} p\left(\mathbf{C}_{\mathcal{P}_{i-1} \cap \mathcal{P}_i}\right)} \\
&= \frac{p\left(\mathbf{C}_{\langle e_1, e_2, e_3 \rangle}\right) \cdot p\left(\mathbf{C}_{\langle e_3, e_4 \rangle}\right) \cdot p\left(\mathbf{C}_{\langle e_4, e_5 \rangle}\right)}{p\left(\mathbf{C}_{\langle e_3 \rangle}\right) \cdot p\left(\mathbf{C}_{\langle e_4 \rangle}\right)}
\end{aligned}$$

Cases	$V_{\mathcal{P}*}$	DE'	$\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'})$
1	$V_{\langle e_6\rangle}$	$\langle\langle e_1,e_2,e_3\rangle,\langle e_3,e_4\rangle,\langle e_4,e_5\rangle,\langle e_6\rangle\rangle$	$\frac{p(\mathbf{C}_{\langle e_1,e_2,e_3\rangle})\cdot p(\mathbf{C}_{\langle e_3,e_4\rangle})\cdot p(\mathbf{C}_{\langle e_4,e_5\rangle})\cdot p(\mathbf{C}_{\langle e_6\rangle})}{p(\mathbf{C}_{\langle e_3\rangle})\cdot p(\mathbf{C}_{\langle e_4\rangle})} = \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})\cdot p(\mathbf{C}_{\langle e_6\rangle})$
2	$V_{\langle e_4,e_5,e_6\rangle}$	$\langle\langle e_1,e_2,e_3\rangle,\langle e_3,e_4\rangle,\langle e_4,e_5,e_6\rangle\rangle$	$\frac{p(\mathbf{C}_{\langle e_1,e_2,e_3\rangle})\cdot p(\mathbf{C}_{\langle e_3,e_4\rangle})\cdot p(\mathbf{C}_{\langle e_4,e_5,e_6\rangle})}{p(\mathbf{C}_{\langle e_3\rangle})\cdot p(\mathbf{C}_{\langle e_4\rangle})} = \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})\cdot \frac{p(\mathbf{C}_{\langle e_4\rangle})\cdot p(\mathbf{C}_{\langle e_4,e_5,e_6\rangle})}{p(\mathbf{C}_{\langle e_4,e_5\rangle})\cdot p(\mathbf{C}_{\langle e_4\rangle})}$
3	$V_{\langle e_5,e_6\rangle}$	$\langle\langle e_1,e_2,e_3\rangle,\langle e_3,e_4\rangle,\langle e_4,e_5\rangle,\langle e_5,e_6\rangle\rangle$	$\frac{p(\mathbf{C}_{\langle e_1,e_2,e_3\rangle})\cdot p(\mathbf{C}_{\langle e_3,e_4\rangle})\cdot p(\mathbf{C}_{\langle e_4,e_5\rangle})\cdot p(\mathbf{C}_{\langle e_5,e_6\rangle})}{p(\mathbf{C}_{\langle e_3\rangle})\cdot p(\mathbf{C}_{\langle e_4\rangle})\cdot p(\mathbf{C}_{\langle e_5\rangle})} = \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})\cdot \frac{p(\mathbf{C}_{\langle e_5,e_6\rangle})}{p(\mathbf{C}_{\langle e_5\rangle})}$

Fig. 9 Examples of the incremental property

We extend path $\mathcal{P}$ with an edge e_6 to get path $\mathcal{P}' = \langle e_1, e_2, e_3, e_4, e_5, e_6\rangle$. We show examples for the above three cases in Fig. 9.

So far, we have shown the incremental property—when extending a path $\mathcal{P}$ with an edge to get a new path $\mathcal{P}'$, the cost distribution of path $\mathcal{P}'$ can be incrementally computed by reusing the cost distribution of path $\mathcal{P}$. The incremental property enables integration of existing stochastic routing algorithms into *PACE*, which we explain in detail in Sect. 5.2.

5.2 Approach 1: integrating existing stochastic routing algorithms into *PACE*

5.2.1 General procedure

Stochastic routing algorithms, e.g., stochastic fastest routing [24], probabilistic top-*k* routing [20], and stochastic skyline routing [38], aim to identify a path or a set of paths whose cost distributions satisfy given conditions, e.g., identifying the path(s) with the highest probability of arriving within 60 min. Despite the different specific conditions that are applied in different algorithms, these algorithms generally follow the procedure shown in Algorithm 2.

Algorithm 2: General Procedure

```
Input  : Source $s$, Destination $d$, Road network $G$,
         Additional Condition $AC$;
Output: Path(s) that satisfy $AC$;
1  Candidate Path Set $CP \leftarrow \emptyset$;
2  According to source $s$, identify all unit-paths whose starting
   vertices are $s$;
3  Add each such unit-path, with its cost distribution, to the
   candidate path set $CP$;
4  while $CP$ is not empty do
5      Remove the most promising path $\mathcal{P}$ from $\mathcal{CP}$ according to
       $AC$;
6      $u \leftarrow$ the last vertex in $P$;
7      for each edge $(u, v)$ that extends from $u$ do
8          Extend $\mathcal{P}$ with edge $(u, v)$ to get path $\mathcal{P}'$;
9          $\boxed{\text{Compute the cost distribution of path } \mathcal{P}';}$
10         Add path $\mathcal{P}'$ together with its cost distribution to $CP$;
```

Specifically, these algorithms often need to explore many candidate paths, and they use different strategies to choose and extend the most promising candidate path to obtain new candidate paths. When identifying promising candidate paths, the algorithms need to compare the candidate paths' cost distributions. To enable such comparison, whenever a new candidate path $\mathcal{P}'$ is generated, existing algorithms employ the legacy, edge-centric path cost distribution computation method to compute the cost distribution of path $\mathcal{P}'$. This occurs in line 9 of Algorithm 2.

By using the path cost distribution computation method based on *PACE*, rather than existing legacy methods, in line 9, existing stochastic routing algorithms can easily be integrated into *PACE*. The remaining mechanisms in the different algorithms, e.g., the different strategies for identifying the most promising candidate paths, are simply kept unchanged.

The running time of a stochastic routing algorithm is dominated by $\sum_{\mathcal{P}\in CP} RT(\mathcal{P}, method)$, where *CP* contains the candidate paths whose cost distributions need to be evaluated. Thus, *CP* differs among stochastic routing algorithms that have different strategies for selecting candidate paths. Function $RT(\mathcal{P}, method)$ refers to the running time of computing the cost distribution of path $\mathcal{P}$ using a specific method, *method*, e.g., a legacy method or the method based on *PACE*. Although the cost distribution computation method in *PACE* has the same worst case asymptotic complexity as the state-of-the-art legacy edge-centric baseline, we offer empirical evidence (in Sect. 6) that computing the cost distribution of a path using *PACE* is more efficient than using the legacy edge-centric baselines, i.e., $RT(\mathcal{P}, PACE) \leq RT(\mathcal{P}, legacy\_baseline)$. We have also shown that the accuracy of the cost distributions estimated by *PACE* is higher than that of the legacy baseline. Thus, integration of existing stochastic routing algorithms into *PACE* is able to improve the accuracy and efficiency of these algorithms.

5.2.2 A concrete algorithm

We show a concrete example of integrating an existing stochastic routing algorithm for solving a probabilistic threshold path finding problem [20] into *PACE*. A concrete example of the problem is "Which path enables a professor to travel from the university to the airport within 45 min with a probability of at least 90%?"

Given a source s, a destination d, a travel time budget B, and a probabilistic threshold τ, a probabilistic threshold path finding problem returns a path from s to d for which the probability that its travel time is smaller than the time budget

B is at least τ. Here, the additional condition AC, as part of the input to Algorithm 2, consists of the time budget B and probability threshold τ.

This problem can be solved by a depth first search (DFS) based algorithm proposed in an existing study [20]. Algorithm 3 demonstrates how the DFS based algorithm can be integrated into *PACE*. In particular, in lines 18–20, when extending a path $\mathcal{P}$ with edge e_{k+1} to a new path $\mathcal{P}'$, the travel cost distribution of the new path, i.e., $\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'})$, is computed based on the travel cost distribution of path $\mathcal{P}$, i.e., $\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})$, using the incremental property. In contrast, the original algorithm employs the legacy method to compute the cost distribution of $\mathcal{P}'$.

Algorithm 3: DFS-based Stochastic Routing

```
Input  : Source $s$, Destination $d$, Travel Time Budget $B$,
         Probability threshold $\tau$, Road network $G$
Output: A path $\mathcal{P}$ satisfying $t$ with probability at least $\tau$
1  $\mathcal{S} \leftarrow \emptyset$;        /* $\mathcal{S}$ is a stack of vertices */
2  $\mathcal{Q} \leftarrow \emptyset$;   /* $\mathcal{Q}$ is a queue of (path, path cost
   distribution) pairs */
3  for each vertex $u$ adjacent to $s$ do
4      $\mathcal{S}.push(u)$;
5      Use path $\mathcal{P}$ to denote path $\langle(s, u)\rangle$;
6      Estimate the travel cost distribution of path $\mathcal{P}$, i.e.,
       $\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})$, using the paper's proposal;
7      if the probability that $\mathcal{P}$'s travel time is smaller than $B$
       exceeds $\tau$ then
8          if $\mathcal{P}$ ends with $d$ then
9              return $\mathcal{P}$;
10         else
11             $\mathcal{Q}.add(\mathcal{P}, \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}}))$;
12 Set $s$ as visited;
13 while $\mathcal{S} \neq \emptyset$ do
14     $u \leftarrow \mathcal{S}.pop()$;
15     for each path $(\mathcal{P}, \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})) \in \mathcal{Q}$ do
16         if the last edge of $\mathcal{P}$ ends with $u$ then
17             for each unvisited vertex $v$ that is adjacent to $u$ do
18                 $e_{k+1} \leftarrow (u, v)$;
19                 $\mathcal{P}' \leftarrow \mathcal{P} \circ \langle e_{k+1}\rangle$;
20                 Estimate the travel cost distribution of $\mathcal{P}'$, i.e.,
                   $\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'})$, according to the incremental property
                   based on $\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})$;
21                 if the probability that $\mathcal{P}'$'s travel time is smaller
                   than $B$ exceeds $\tau$ then
22                     if $\mathcal{P}'$ ends with $d$ then
23                         return $\mathcal{P}'$;
24                     else
25                         $\mathcal{Q}.add(\mathcal{P}', \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}'}))$;
26             Remove $(\mathcal{P}, \hat{p}_{DE}(\mathbf{C}_{\mathcal{P}}))$ from $\mathcal{Q}$;
27     Set $u$ as visited;
28     for each unvisited vertex $w$ that is adjacent $u$ do
29         $\mathcal{S}.push(w)$;
30 return null;
```

Algorithm 3 exemplifies how an existing stochastic routing algorithm can be integrated seamlessly into the *PACE*. We believe that all stochastic routing algorithms known to us, such as [20,24,38], can be accommodated in a similar way, thus improving the efficiency and effectiveness of existing stochastic routing algorithms.

5.3 Approach 2: new stochastic routing algorithms in *PACE*

5.3.1 Pruning strategies

As discussed, stochastic routing algorithms typically need to explore many candidate paths. To ensure efficiency, existing edge-centric algorithms normally employ an early pruning strategy to avoid exploring uncompetitive candidate paths.

Specifically, at an intermediate vertex, if there is more than one path from the source to the intermediate vertex, the algorithms only keep the best path(s) and prune all remaining paths. Here, the goodness of a path is defined differently according to the specific conditions used in different stochastic routing algorithms. However, existing pruning may not be safe in *PACE*. There is thus a need for new stochastic routing algorithms that take this into account such that the accurate travel cost distribution offered by *PACE* can be utilized fully.

To better explain why the pruning in existing edge-centric algorithms may not be safe, consider the example in the introduction, which is a stochastic on-time arrival problem [30,32,35]. Given a source s, a destination d, a travel-time budget B, we aim at finding the path from s to d that has the largest probability of arriving within the time budget B.

Consider the road network in Fig. 10, where v_A and v_E are the source and the destination, respectively. Two paths exist from source v_A to an intermediate vertex v_D: $\mathcal{P}_1 = \langle e_1, e_2\rangle$ and $\mathcal{P}_2 = \langle e_3, e_4\rangle$.

Assume that the probability that $\mathcal{P}_1$ takes travel time t exceeds that of $\mathcal{P}_2$, meaning that $\mathcal{P}_1$ has a better distribution

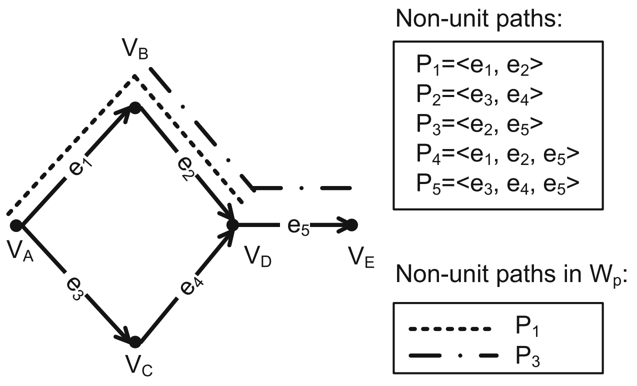


Fig. 10 A road network and non-unit paths

than $\mathcal{P}_2$. Then, in the legacy, edge-centric paradigm, $\mathcal{P}_2$ can be pruned safely and we need not consider any path that is extended from $\mathcal{P}_2$. This is because, for any path $\mathcal{P}_2 \circ \mathcal{P}$ that is extended from $\mathcal{P}_2$ by path $\mathcal{P}$, the alternative path $\mathcal{P}_1 \circ \mathcal{P}$ that is extended from $\mathcal{P}_1$ by the same path $\mathcal{P}$, has a better distribution.

For example, consider path $\mathcal{P}_5 = \langle e_3, e_4, e_5 \rangle$ that is extended from $\mathcal{P}_2$ by path $\langle e_5 \rangle$. The alternative path is $\mathcal{P}_4 = \langle e_1, e_2, e_5 \rangle$ that is extended from $\mathcal{P}_1$ by the same path $\langle e_5 \rangle$. Since existing stochastic routing algorithms use the legacy, edge-centric method to compute the distributions of candidate paths, we have $p(\mathbf{C}_{\mathcal{P}_5}) = p(\mathbf{C}_{\mathcal{P}_2}) \odot p(\mathbf{C}_{\langle e_5 \rangle})$ and $p(\mathbf{C}_{\mathcal{P}_4}) = p(\mathbf{C}_{\mathcal{P}_1}) \odot p(\mathbf{C}_{\langle e_5 \rangle})$, where $\odot$ denotes convolution. This means that the distribution of path $\mathcal{P}_5$ is the convolution of distributions of paths $\mathcal{P}_2$ and $\langle e_5 \rangle$ and the distribution of path $\mathcal{P}_4$ is the convolution of distributions of paths $\mathcal{P}_1$ and $\langle e_5 \rangle$. Further, since $\mathcal{P}_1$ has a better distribution than $\mathcal{P}_2$, after performing convolution with the same distribution, i.e., the cost distribution of path $\langle e_5 \rangle$, path $\mathcal{P}_4$ should have a better distribution than path $\mathcal{P}_5$. Formal proofs of the above intuition can be found elsewhere [32,35].

In contrast, in *PACE*, we can only apply such pruning at *some* intermediate vertices, but not at *all* intermediate vertices. For example, assume that a *PACE* maintains path weights for $\mathcal{P}_1 = \langle e_1, e_2 \rangle$ and $\mathcal{P}_3 = \langle e_2, e_5 \rangle$, as shown in Fig. 10. Then, we cannot apply pruning at intermediate vertex v_D.

The reason is as follows. Still assuming that $\mathcal{P}_1$ has a better distribution than $\mathcal{P}_2$, we cannot guarantee that for any path that is extended from $\mathcal{P}_2$, we always have a corresponding path that is extended from $\mathcal{P}_1$ that has a better distribution. For example, consider paths $\mathcal{P}_4$ and $\mathcal{P}_5$, and recall that *PACE* maintains path weights for $\mathcal{P}_1$ and $\mathcal{P}_3$. Thus, in *PACE*, the cost distribution of $\mathcal{P}_4 = \langle e_1, e_2, e_5 \rangle$ is computed as $\frac{p(\mathbf{C}_{\mathcal{P}_1}) \cdot p(\mathbf{C}_{\mathcal{P}_3})}{p(\mathbf{C}_{\langle e_2 \rangle})}$, i.e., from the cost distributions of $\mathcal{P}_1$, $\mathcal{P}_3$, and $\langle e_2 \rangle$; and the cost distribution of $\mathcal{P}_5 = \langle e_3, e_4, e_5 \rangle$ is computed as $p(\mathbf{C}_{\mathcal{P}_2}) \cdot p(\mathbf{C}_{\langle e_5 \rangle})$, i.e., from the cost distributions of $\mathcal{P}_2$ and $\langle e_5 \rangle$. This suggests that $\mathcal{P}_4$'s distribution is computed as the product of $\mathcal{P}_1$'s distribution and a term that involves the distributions of $\mathcal{P}_3$ and $\langle e_2 \rangle$. Further, $\mathcal{P}_5$'s distribution is computed as the product of $\mathcal{P}_2$'s distribution and a term that involves the distribution of $\langle e_5 \rangle$. Although we know that $\mathcal{P}_1$ has a better distribution than $\mathcal{P}_2$, $\mathcal{P}_4$'s distribution may not be better than that of $\mathcal{P}_5$ since the two terms are no longer the same.

5.3.2 Vertex categorization

We categorize vertices as *safe* or *unsafe*. Recall that, in Sect. 3.3, the path weight function W_P maintains weights for non-unit paths in $Paths_\beta$ and all unit paths. Given a vertex u, if a non-unit path $\mathcal{P} \in Paths_\beta$ exists such that path $\mathcal{P}$ covers vertex u and vertex u is not the starting vertex of the first edge of path $\mathcal{P}$ and is also not the ending vertex of the last edge of path $\mathcal{P}$, vertex u is *unsafe*. Otherwise, u is *safe*.

For example, consider the *PACE* graph in Fig. 10, where $Paths_\beta$ consists of non-unit paths $\mathcal{P}_1$ and $\mathcal{P}_3$. Then, vertex V_B is unsafe due to $\mathcal{P}_1$ and vertex V_D is unsafe due to $\mathcal{P}_3$. All remaining vertices are safe.

Based on this categorization, we formulate two principles for designing stochastic routing algorithms in the *PACE* paradigm:

(1) At a *safe* vertex, pruning can be applied, which is analogs to the legacy case. Assume that two paths $\mathcal{P}_1$ and $\mathcal{P}_2$ meet at a safe vertex and $\mathcal{P}_1$ has a better distribution than $\mathcal{P}_2$. No matter which edge is extended, path $\mathcal{P}_1'$ that is extended from $\mathcal{P}_1$ always has a better distribution than path $\mathcal{P}_2'$ extended from $\mathcal{P}_2$. This is because the distributions of path $\mathcal{P}_1'$ and $\mathcal{P}_2'$ are computed from the distributions path $\mathcal{P}_1$ and $\mathcal{P}_2$ using the *same term*.

(2) At an *unsafe* vertex, pruning cannot be applied. Assume that two paths $\mathcal{P}_1$ and $\mathcal{P}_2$ meet at an unsafe vertex and $\mathcal{P}_1$ has a better distribution than $\mathcal{P}_2$. Path $\mathcal{P}_1'$ extended from $\mathcal{P}_1$ may not have a better distribution than path $\mathcal{P}_2'$ extended from $\mathcal{P}_2$. This is because the distributions of path $\mathcal{P}_1'$ and $\mathcal{P}_2'$ are computed from the distributions of path $\mathcal{P}_1$ and $\mathcal{P}_2$ using *different terms*. Thus, pruning cannot be applied at unsafe vertices.

5.3.3 New algorithms

Based on the vertex categorization and the two principles, we propose a new algorithm, shown in Algorithm 4, for solving the stochastic on-time arrival problem. Algorithms for other stochastic routing problems can be devised similarly.

The new algorithm employs two data structures—a priority queue Q and a hash table H. The priority queue Q maintains a collection of items of the form (u, $\mathcal{P}$, *PRB*), where u is a vertex, $\mathcal{P}$ is a path that is from the source to vertex u, and *PRB* is the probability that the travel time of $\mathcal{P}$ is less than the time budget B. The priority queue is prioritized on the *PRB* field. The hash table H associates each vertex u with a set of items of the form ($\mathcal{P}$, $\hat{p}_{DE}(\mathbf{C}_\mathcal{P})$), where $\mathcal{P}$ is a path from the source to the vertex u and $\hat{p}_{DE}(\mathbf{C}_\mathcal{P}))$ is the travel cost distribution of path $\mathcal{P}$.

The algorithm starts from the source vertex s. We insert s along with an empty path $\langle\rangle$ and 1 as its probability into the priority queue, and we associate the source vertex s with an empty path and 0 as its travel cost in the hash table. In addition, we use *maxProb* to record the maximum probability of arriving within the time budget among all the paths that have been explored so far, and we use *matPath* to record the corresponding path. In the beginning, *maxProb* and *matPath* are initialized to $-\infty$ and *null*.

Algorithm 4: New Stochastic Routing for *PACE*

```
Input  : Source $s$, Destination $d$, Travel Time Budget $B$,
         Road network $G$;
Output: A path $\mathcal{P}$ with the highest probability of arrival within $t$;
1  Initialize a Priority Queue $\mathcal{Q}$ and a Hash Table $\mathcal{H}$;
2  $maxProb \leftarrow -\infty$; $maxPath \leftarrow null$;
3  Insert $(s, \langle\rangle, 1)$ into $\mathcal{Q}$;
4  $\mathcal{H}[s] \leftarrow (\langle\rangle, 0)$;
5  while $\mathcal{Q}$ is not empty do
6      $(u, \mathcal{P}, PRB) \leftarrow \mathcal{Q}.ExtractMax()$;
7      if Path $\mathcal{P}$ reaches destination d, i.e., $u = d$ then
8          if $PRB > maxProb$ then
9              $maxProb \leftarrow PRB$; $maxPath \leftarrow \mathcal{P}$;
10         else
11             if $PRB > 0$ then
12                 return $maxPath$;
13             else
14                 return $null$;
15     for each edge $(u, v)$ from $u$ do
16         if $v$ does not appear in $\mathcal{P}$ then
17             $e_{k+1} \leftarrow (u, v)$;
18             Relax($\mathcal{Q}$, $\mathcal{H}$, $v$, $\mathcal{P}$, $e_{k+1}$);
```

Algorithm 5: Relax Operation

```
Input  : PriorityQueue: $\mathcal{Q}$, HashTable: $\mathcal{H}$, Vertex $v$, Path $\mathcal{P}$,
         Edge $e_{k+1}$;
1  $\mathcal{P}' \leftarrow \mathcal{P} \circ \langle e_{k+1}\rangle$;
2  Estimate the travel cost distribution of $\mathcal{P}'$, i.e., $\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'})$,
   according to the incremental property based on $\hat{p}_{DE}(\mathbf{C}_{\mathcal{P}})$;
3  Boolean $notDominated \leftarrow$ true;
4  if $\mathcal{H}[v]$ is not empty $\wedge$ $v$ is a safe vertex then
5      for each path $\mathcal{P}^*$ in $\mathcal{H}[v]$ do
6          if $\mathcal{P}'$ dominates $\mathcal{P}^*$ then
7              Remove the items with $\mathcal{P}^*$ from both $\mathcal{H}[v]$ and $\mathcal{Q}$;
8          if $\mathcal{P}^*$ dominates $\mathcal{P}'$ then
9              $notDominated \leftarrow$ false;
10             break;
11 if $notDominated$ then
12     $\mathcal{H}[v] \leftarrow \mathcal{H}[v] \cup (\mathcal{P}', \hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'}))$;
13     Insert $(v, \mathcal{P}', EXP(\hat{p}_{DE'}(\mathbf{C}_{\mathcal{P}'})))$ into $\mathcal{Q}$;
```

Next, we iterate on all items in the priority queue. In each iteration, we first identify the item $(u, \mathcal{P}, PRB)$ with the maximum priority, i.e., the item with the largest *PRB*. If path $\mathcal{P}$ has already reached the destination d and its probability of arriving within the time budget *PRB* is larger than the best probability of all the paths that have been explored, *maxProb*, we update *maxProb* and keep exploring paths that are extended from $\mathcal{P}$. Otherwise, path *matPath* is the path with the largest probability of arriving within the time budget.

To consider the paths that are extended from path $\mathcal{P}$, we consider the edges whose starting vertex is u. In particular, for each edge (u, v), where v has not already appeared in $\mathcal{P}$, we may extend $\mathcal{P}$ with edge (u, v). However, it is not always beneficial to extend $\mathcal{P}$ with edge (u, v), e.g., if there already exists another path that also leads to u has a better travel cost distribution. To check whether it is worth to extend $\mathcal{P}$ with edge (u, v), we call function *Relax*(), described in Algorithm 5.

In the *Relax*() function, we extend $\mathcal{P}$ with edge $e_{k+1} = (u, v)$ to get path $\mathcal{P}' = \mathcal{P} \circ \langle e_{k+1}\rangle$. We compute the cost distribution of $\mathcal{P}'$ based on the travel cost distribution of $\mathcal{P}$ using the incremental property. Then, we consider the following three cases to decide whether we should keep considering the extended path $\mathcal{P}'$.

Case 1 $\mathcal{H}[v]$ is empty. This means that $\mathcal{P}'$ is the first path to reach vertex v. Thus, we should keep considering the paths that are extended from $\mathcal{P}'$. We add $\mathcal{P}'$ to both $\mathcal{Q}$ and $\mathcal{H}[v]$—we add vertex v along with path $\mathcal{P}'$ and its expected path cost to $\mathcal{Q}$ and associate vertex v with path $\mathcal{P}'$ and its path cost distribution to $\mathcal{H}$ (lines 12–13).

Case 2 $\mathcal{H}[v]$ is not empty and v is safe (lines 4–10). According to existing studies [32,35], we need to check the *stochastic dominance* [2,35] relationship between path $\mathcal{P}'$ and each path $\mathcal{P}^*$ in $\mathcal{H}[v]$. In particular, if $\mathcal{P}'$ dominates $\mathcal{P}^*$, we remove the corresponding items for $\mathcal{P}^*$ from both $\mathcal{Q}$ and $\mathcal{H}[v]$.

If $\mathcal{P}^*$ dominates $\mathcal{P}'$, we need not consider $\mathcal{P}'$ further. This is because for any path $\mathcal{P}' \circ \mathcal{P}$ that is extended from $\mathcal{P}'$, we always have another path $\mathcal{P}^* \circ \mathcal{P}$ that dominates $\mathcal{P}' \circ \mathcal{P}$. Thus, no matter what cost budget is considered, $\mathcal{P}^* \circ \mathcal{P}$ always has a larger probability of satisfying the cost budget than does $\mathcal{P}' \circ \mathcal{P}$.

If no path in $\mathcal{H}[v]$ can dominate $\mathcal{P}'$, we should consider paths that are extended from $\mathcal{P}'$. Thus, we add $\mathcal{P}'$ to both $\mathcal{Q}$ and $\mathcal{H}[v]$.

Case 3 when $\mathcal{H}[v]$ is not empty and v is unsafe. We cannot apply pruning and thus must consider paths that are extended from $\mathcal{P}'$. Thus, we add $\mathcal{P}'$ to both $\mathcal{Q}$ and $\mathcal{H}[v]$.

6 Empirical study

6.1 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 network N_1 is obtained from OpenStreetMap and contain all roads, while road network N_2 is obtained from the Beijing traffic management bureau, which contains only highways and main roads.

Trajectories Two GPS data sets are used. The first, D_1, contains 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

Table 2 Parameter settings

Parameters	Values
α (min)	15, **30**, 45, 60, 120
β	15, **30**, 45, 60
$\lvert\mathcal{P}_{query}\rvert$	5, 10, 15, **20**, 40, 60, 80, 100
B (min)	15, **20**, 25, 30
d (km)	1, 2, **3**, 4, 5
r (%)	20, 40, 60, 80, **100**

September 2012 to November 2012. The sampling rate is at least 0.2 Hz. We apply a well-known method [29] to map match the GPS records.

Travel costs We consider two time-varying, uncertain travel costs—travel time and GHG emissions.

Parameters Table 2 shows important parameters used in the experiments, where default values are shown in bold. In particular, when instantiating the *PACE* from trajectories, we vary the finest time interval α and the qualified trajectory count threshold β. When studying the accuracy and efficiency of *PACE* based path cost distribution computation method, we vary the cardinality of a query path $|\mathcal{P}_{query}|$. When studying the stochastic routing algorithms based on *PACE*, we vary the time budge B, the Euclidean distance between source and destination d, and percentage of instantiated, high-rank random variables r.

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

6.2 Instantiating *PACE*

We conduct experiments to obtain insight into different aspects of the instantiated random variables that are maintained in the *PACE*'s weight function W_P and also describe how to tune parameters α and β. To highlight the random variables that are instantiated from trajectories, random variables derived from speed limits are excluded unless stated otherwise.

Tuning α We vary α from 15 to 120 min. A large α 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 variables 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 11a shows that as α 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 α enables more instantiated random variables, they may be inaccurate since traffic may change significantly during a long interval, e.g., 2 h. We report the average entropies of the instantiated random variables when varying α; see Fig. 11b. 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 11b shows that using $\alpha = 30$ does not significantly increase the entropy compared to using $\alpha = 15$. Figure 11a 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 variables and the numbers of random variables. Thus, we use $\alpha = 30$ as the default value.

Tuning β Intuitively, we prefer a large β since having more qualified trajectories enables instantiated random variables that are more accurate. However, Fig. 12 shows that as β increases, the number of instantiated random variables drops. This occurs because a large β requires more qualified trajectories in order to be able to instantiate W_P. We choose $\beta = 30$ as the default because the number of instantiated random variables is only slightly less than 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 13 shows that

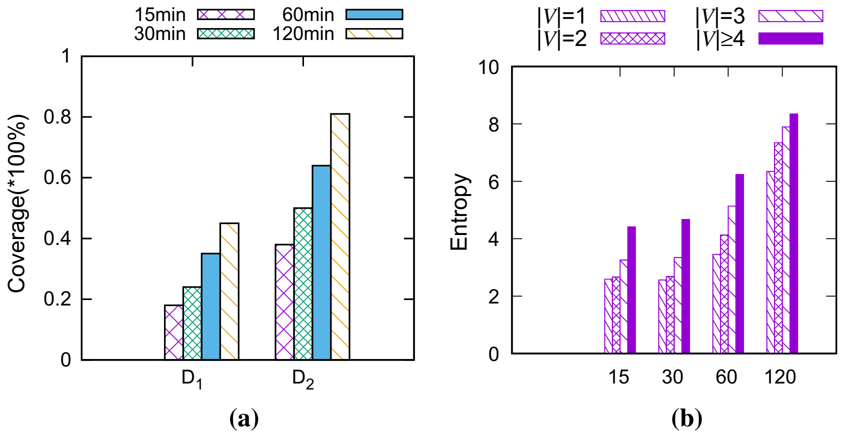


Fig. 11 Effect of α. **a** Coverage. **b** Entropy

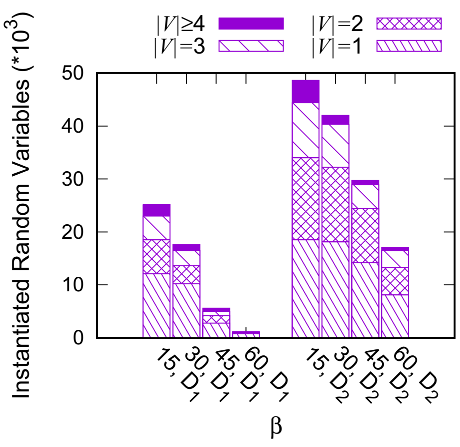


Fig. 12 Effect of β

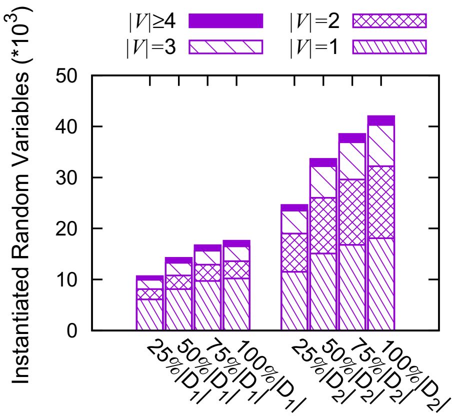


Fig. 13 Varying dataset sizes

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 β 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., greater than 4) are small.

Note that since we do not study accuracy here, there is no need to distinguish between training and testing data. In Sect. 6.3.1, we use separate training and testing data when studying accuracy.

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. Sect. 3.1). We call this method *Auto*. We first compare *Auto* with methods using standard distributions including Gaussian, Gamma, and exponential distributions. Figure 14a 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 other 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 14b 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*-4. 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

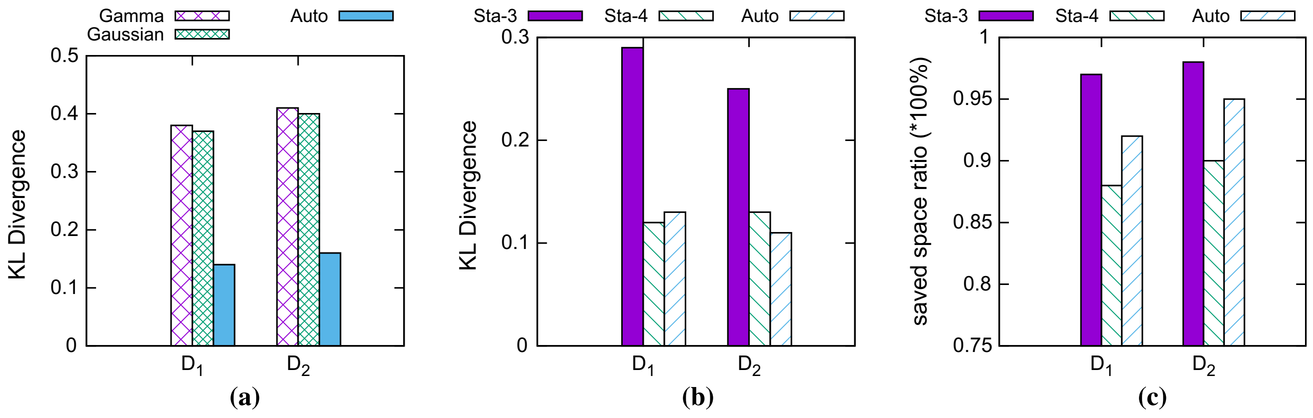


Fig. 14 Performance of multi-dimensional histograms. **a** Varying methods. **b** Varying histograms. **c** Space saving ratio

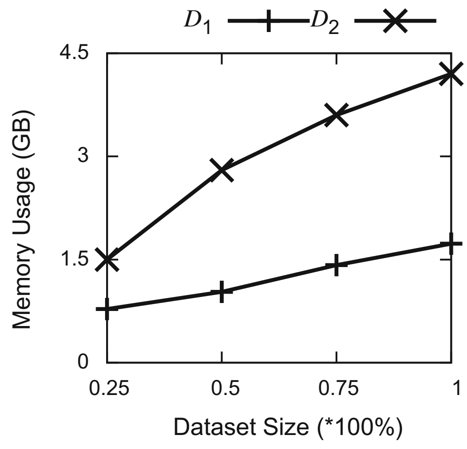


Fig. 15 Memory usage

$1 - \frac{S_H}{S_R}$, where S_H and S_R represent the storage required by the histograms and the underlying raw data distribution, respectively. The raw data distribution is of the form (*cost*, *frequency*). The higher the ratio, the better space savings are achieved by the histograms. Figure 14c 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 Fig. 15.

Since we use histograms to represent the distributions of instantiated random variables, the memory use is small such that the *PACE*'s weight function W_P 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 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 min with 48 threads to learn the random variables from D_1, and takes around 45 min 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.

6.3 Path cost distribution computation with *PACE*

We consider four methods. (a) *OD*: the proposed method for *PACE* using the optimal (i.e., coarsest) decomposition. (b) *LB* [38]: the legacy baseline as described in Sect. 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* [20] 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.

6.3.1 Accuracy

Accuracy evaluation with ground truth We select 100 paths where each path has more than $\beta = 30$ trajectories during an interval of $\alpha = 30$ min. 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 Fig. 1b. The distributions estimated using *OD*, *LB*, *HP*, and *RD* are shown in Fig. 16a–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 using all 100 selected paths, which have different cardinalities. Specifically, the cardinality of a path $|P_{query}|$ varies from 5 to 20. Figure 17 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, Fig. 17 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 1000 paths for each cardinality

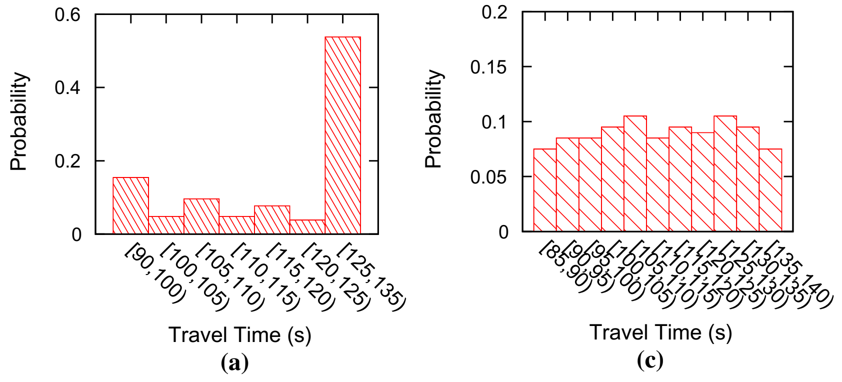
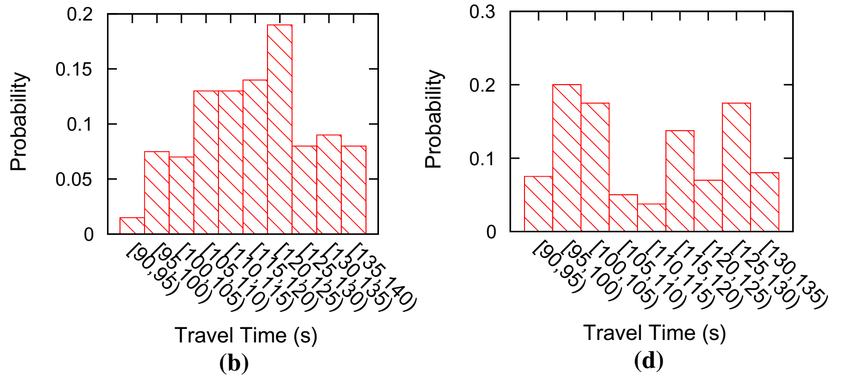



Fig. 16 Accuracy comparison on a particular path. **a** OD. **b** LB. **c** HP. **d** RD

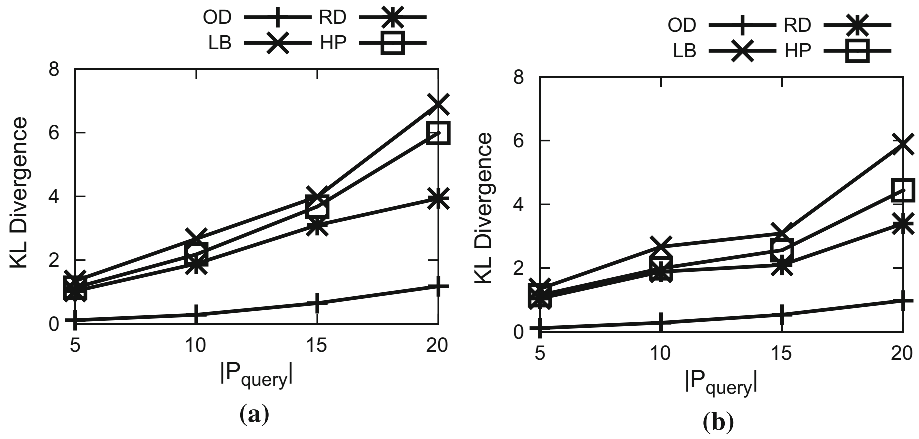


Fig. 17 Accuracy comparison with the ground truth. **a** D_1. **b** D_2

with an arbitrary departure time and report average values; and vary the path cardinality from 20 to 100. Figure 18 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.

6.3.2 Efficiency

Figure 19 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 $|\mathcal{P}_{query}|$ 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 19 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 Fig. 20 for paths with cardinality 20 and using differently sized subsets of trajectories.

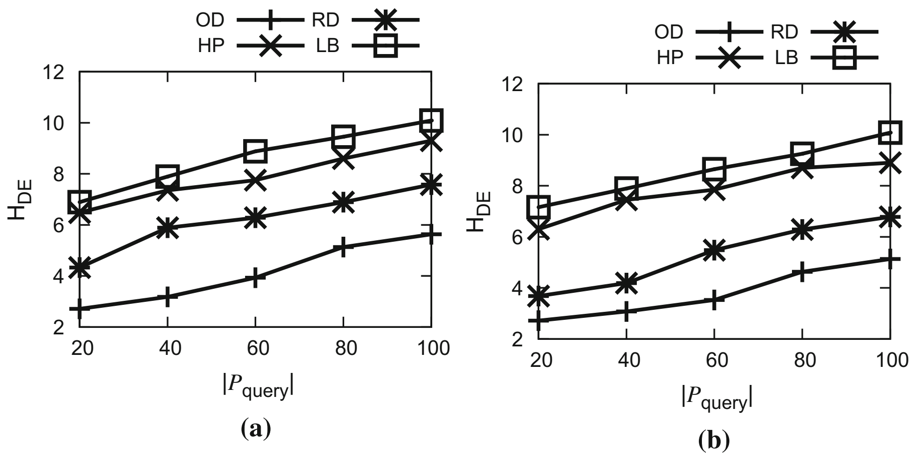


Fig. 18 Entropy comparison. **a** D_1. **b** D_2

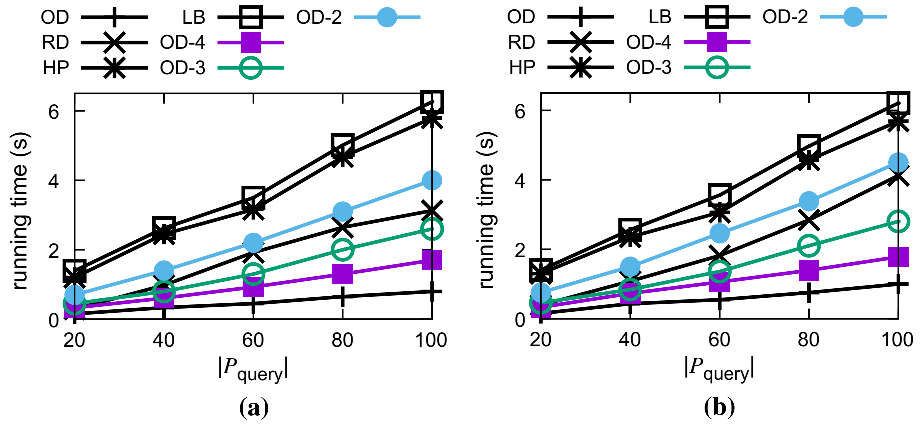


Fig. 19 Efficiency. **a** D_1. **b** D_2

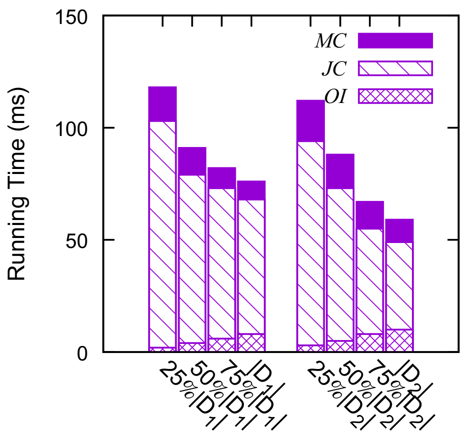


Fig. 20 Run-time analysis

Three steps are involved in *OD*. First, the optimal decomposition is identified, denoted by *OI*. 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 Eq. 3. 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.

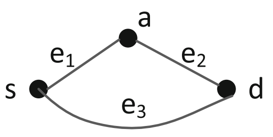
6.4 Stochastic routing with *PACE*

We consider two stochastic routing problems—*PB1*: a probabilistic threshold path finding problem, and *PB2:* a stochastic on-time arrival problem. These are covered in Sects. 5.2 and 5.3, respectively. For each problem, we consider two aspects—effectiveness and efficiency.

6.4.1 Effectiveness

Accurate path cost distributions are essential for achieving accurate, high-quality stochastic routing [20,24,38]. If the computed path cost distributions are inaccurate, the returned paths can deviate from the "correct" paths, thus adversely affecting the service quality.

In the following, we exemplify the quality effect of using accurate path cost distributions as offered by *PACE*, and then we report on experiments that aim to quantify the quality effect at a larger scale.



(a)

Travel Time	Probability
50	0.8
60	0.2

(b)

Travel Time	Probability
5	0.8
10	0.2

(c)

Travel Time	Probability
55	0.8
70	0.2

(d)

Travel Time	Probability
55	0.2
60	0.3
65	0.4
75	0.1

(e)

Travel Time	Probability
55	0.64
60	0.16
65	0.16
70	0.04

(f)

Fig. 21 Effectiveness of accurate travel-time distributions. **a** Example. **b** $p_{GT}(\mathbf{C}_{\langle e_1\rangle})$. **c** $p_{GT}(\mathbf{C}_{\langle e_2\rangle})$. **d** $p_{GT}(\mathbf{C}_{\mathcal{P}_1})$. **e** $p_{GT}(\mathbf{C}_{\mathcal{P}_2})$. **f** $p_{LB}(\mathbf{C}_{\mathcal{P}_1})$

Consider the example in Fig. 21a, where two paths from source S to destination D exist: $\mathcal{P}_1 = \langle e_1, e_2\rangle$ and $\mathcal{P}_2 = \langle e_3\rangle$. Assume that more than β trajectories occurred on $\langle e_1\rangle$, $\langle e_2\rangle$, $\mathcal{P}_1$, and $\mathcal{P}_2$. Thus, we are able to derive the ground-truth distributions $p_{GT}(\mathbf{C}_{\langle e_1\rangle})$, $p_{GT}(\mathbf{C}_{\langle e_2\rangle})$, $p_{GT}(\mathbf{C}_{\mathcal{P}_1})$, and $p_{GT}(\mathbf{C}_{\mathcal{P}_2})$, which are shown in Fig. 21b–e, respectively. These distributions are maintained in *PACE*.

In contrast, in the edge-centric paradigm, only the ground-truth distributions for individual edges, i.e., unit paths, are maintained. To obtain the distribution of non-unit path $\mathcal{P}_1$, the legacy baseline computes the convolution of $p_{GT}(\mathbf{C}_{\langle e_1\rangle})$ and $p_{GT}(\mathbf{C}_{\langle e_2\rangle})$, denoted as $p_{LB}(\mathbf{C}_{\mathcal{P}_1})$, which is shown in Fig. 21f.

It is clear that the cost distribution of $\mathcal{P}_1$ based on the legacy method $p_{LB}(\mathbf{C}_{\mathcal{P}_1})$ is different from the ground-truth cost distribution $p_{GT}(\mathbf{C}_{\mathcal{P}_1})$. A small difference like this may result in significantly different stochastic routing results. For instance, suppose that we consider the stochastic on-time arrival problem of choosing the path with the highest probability of arriving at d within 65 min. When using the legacy method, $\mathcal{P}_1$ has probability 0.96 according to $p_{LB}(\mathbf{C}_{\mathcal{P}_1})$, and $\mathcal{P}_2$ has probability 0.90 according to $p_{LB}(\mathbf{C}_{\mathcal{P}_2})$, which is the same as $p_{GT}(\mathbf{C}_{\mathcal{P}_2})$ since $\mathcal{P}_2$ is a unit-path. Thus, $\mathcal{P}_1$ is returned. However, when using the method based on *PACE*, $\mathcal{P}_2$ is returned. This is because $\mathcal{P}_1$ has probability 0.80 according to $p_{GT}(\mathbf{C}_{\mathcal{P}_1})$, which is maintained as a path weight in *PACE*; and $\mathcal{P}_2$ still has probability 0.90, according to $p_{GT}(\mathbf{C}_{\mathcal{P}_2})$. This example shows that the legacy method may return paths that are significantly different from the correct paths, which *PACE* is able to identify.

To evaluate the impact of using more accurate path cost distributions at a larger scale, we consider stochastic queries for both *PB1* and *PB2*. For each problem, we randomly generate 100 queries for each of road networks D_1 and D_2.

PB1 Given a source s, a destination d, a departure time t, a time budget B, and a probabilistic threshold τ, a query returns a path that enables arrival at destination d within time budget B with probability at least τ.

PB2 Given a source s, a destination d, a departure time t, and a time budget B, a query returns a path $\mathcal{P}$ that enables arrival at d with the largest probability.

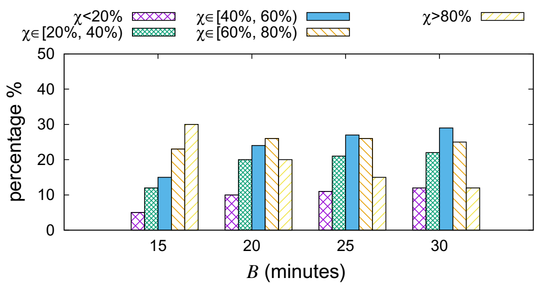
For *PB1*, we consider two algorithms, *PB1-Alg1* and *PB1-Alg2*, based on the DFS-based stochastic routing algorithm. When evaluating path cost distributions, *PB1-Alg1* uses the legacy paradigm, and *PB1-Alg2*, i.e., Algorithm 3 proposed in Sect. 5.2.2, uses *PACE*.

For the same query, suppose that *PB1-Alg1* returns $\mathcal{P}_1$ and *PB1-Alg2* returns $\mathcal{P}_2$. If $\mathcal{P}_1$ and $\mathcal{P}_2$ are different, the more accurate path cost distributions employed by *PB1-Alg2* have a significant impact on the outcome of the stochastic routing.

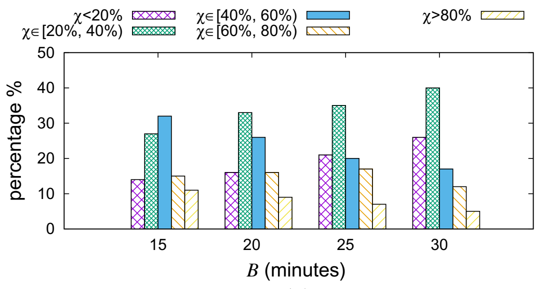
To quantify the significance of the impact, we calculate the path difference ratio χ between $\mathcal{P}_i$ and $\mathcal{P}_2$, where $\chi = \frac{|\mathcal{P}_1 \cap \mathcal{P}_2|}{|\mathcal{P}_1 \cup \mathcal{P}_2|} \times 100\%$, $\mathcal{P}_1 \cap \mathcal{P}_2$ is the set of edges that appear in both paths, and $\mathcal{P}_1 \cup \mathcal{P}_2$ is the set of edges that appear in either $\mathcal{P}_1$ or $\mathcal{P}_2$. The smaller χ is, the more the two paths differ, giving us a measure of the significance of the difference between the results.

We vary the travel-time budget B and show the distributions of the difference ratios in Fig. 22a. In this experiment, we set $\tau = 1$, i.e., identifying a path that guarantees that a person arrives at destination d within time budget B.

The results show that as time budget B grows, the differences between the paths returned by the two algorithms increase. For example, consider the case where χ is less than 20% for *PB1*, meaning that the two paths are very different. When $B = 15$, around 5% of the queries fall into this cate-



(a)



(b)

Fig. 22 Path difference ratios **a** *PB1*. **b** *PB2*

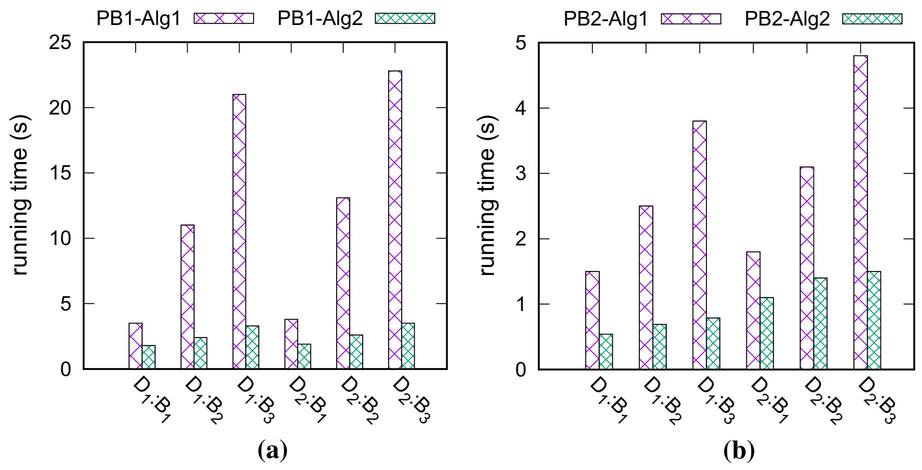


Fig. 23 Effect of B. **a** *PB1*. **b** *PB2*

gory; and when $B = 30$, more than 10% of the queries fall into this category. Next, consider the case where χ exceeds 80%, meaning that the two paths are quite similar. When $B = 15$, around 30% of the queries fall into this category; and when $B = 30$, below 15% of the queries fall into this category. A similar trend occurs for *PB2*. As B changes from 15 to 30, the χ values that exceeds 80% reduces from 10% to 5%. In contrast, the χ values below 20% increases from around 10% to more than 20%.

Next, we consider *PB2*. As before, we consider two algorithms: *PB2-Alg1* uses the legacy paradigm, and *PB2-Alg2*, i.e., Algorithm 4 proposed in Sect. 5.3.3, uses *PACE*, to evaluate path cost distributions. In addition, *PB2-Alg2* applies the pruning strategy on safe vertices according to the vertex categorization, while *PB2-Alg1* applies pruning on all vertices.

Figure 22b shows the difference ratios between the paths returned by *PB2-Alg1* and *PB2-Alg2*. We observe similar trends as for *PB1*—as time budget B increases, the differences between the paths returned by the two algorithms increase.

This set of experiments suggests that the two different routing approaches based on *PACE* are likely to yield significantly different paths compared to the paths that are returned by existing routing algorithms based on the legacy paradigm. As *PACE* based path cost estimation is more accurate than estimation based on legacy baseline, the returned paths are also more accurate.

6.4.2 Efficiency

Next, we consider the running time efficiency of the different routing algorithms. We vary the travel-time budget B, the Euclidean distance d between the source and the destination, and the percentage r of instantiated, high-rank random variables according to Table 2.

Following the setup used in the effectiveness experiments, for *PB1*, we consider two algorithms *PB1-Alg1* and *PB1-Alg2*, and for *PB2*, we consider two algorithms *PB2-Alg1* and *PB2-Alg2*.

Effect of B For each road network, we randomly choose 100 source–destination pairs. We consider travel-time budgets $B_1 = 10$ mins, $B_2 = 20$ mins, and $B_3 = 30$ mins.

The average running times for *PB1* are reported in Fig. 23a, which shows that the algorithm based on *PACE*, i.e., *PB1-Alg2*, is faster. This indicates that the first approach of using *PACE*, where the legacy paradigm is replaced by the *PACE* paradigm, is able to accelerate existing stochastic routing algorithms.

Next, Fig. 23b shows that algorithm *PB2-Alg2*, which is based on *PACE* and applies pruning only on safe vertices, is faster than *PB2-Alg1*. This suggests that the two different approaches to using the *PACE* paradigm are able to provide more efficient stochastic routing.

Effect of d To gain insight into the effect of the distance between source and destination, we vary parameter d. The results are shown in Fig. 24a for *PB1* and Fig. 24b for *PB2*.

As the distance d increases, more candidate paths needs to be explored, meaning that the numbers of path cost distribution computations and comparisons increase. Thus, the running time also increases. Nevertheless, both Fig. 24a, b show the benefits of the algorithms based on *PACE* over existing stochastic routing algorithms. First, the running times of *PB1-Alg2* and *PB2-Alg2* are consistently lower than those of *PB1-Alg1* and *PB2-Alg1*, suggesting that incorporating *PACE* improves efficiency. Second, as d increases, the running time differences between the algorithms with and without using PACE, i.e., *PB1-Alg2* vs. *PB1-Alg1* and *PB2-Alg2* versus *PB2-Alg1*, become more significant, indicating that the *PACE*-based algorithms scale better w.r.t. parameter d.

Effect of r To study the effect of the number of instantiated, high-rank (above 2) random variables, we gradually reduce the number of instantiated random variables. In particular,

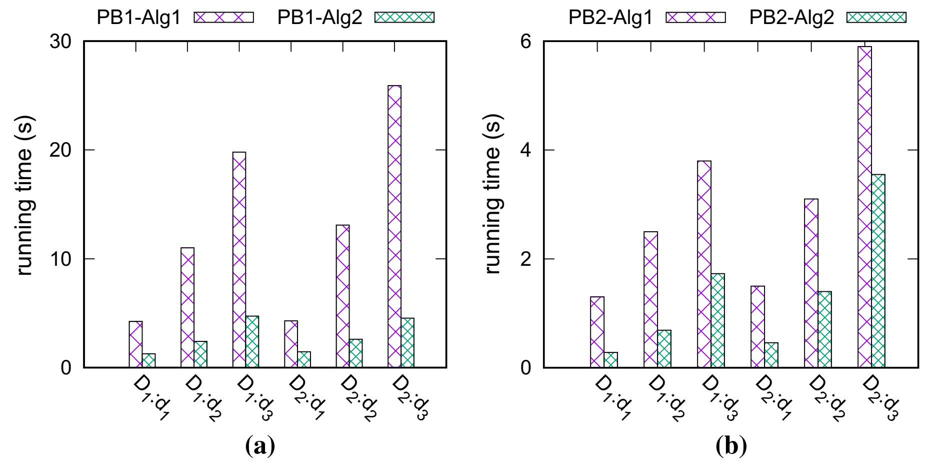


Fig. 24 Effect of d. **a** *PB1*. **b** *PB2*

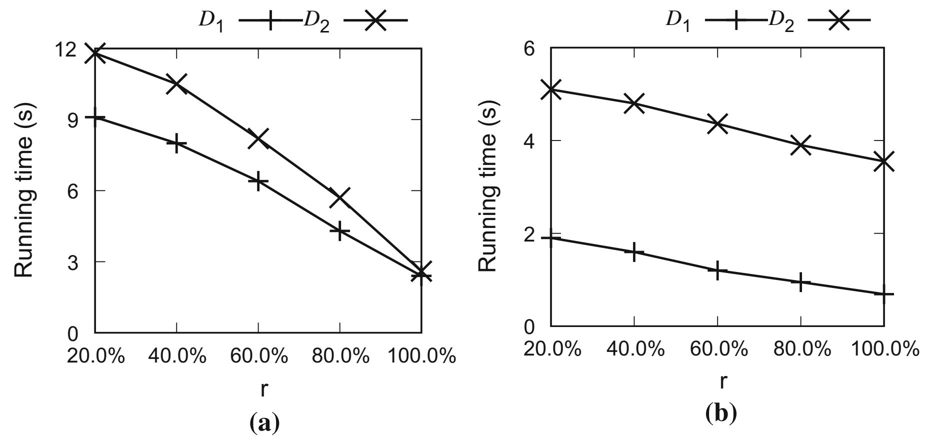


Fig. 25 Effect of r. **a** *PB1, PB1-Alg2*. **b** *PB2, PB2-Alg2*

we use different percentages of instantiated random variables on networks D_1 and D_2, where $r_1 = 20\%$, $r_2 = 40\%$, $r_3 = 60\%$, $r_4 = 80\%$, and $r_5 = 100\%$. Since only the algorithms based on *PACE* employ the instantiated random variables, we only consider *PB1-Alg2* and *PB2-Alg2*. The run-times are reported in Fig. 25.

As r increases, the running time decreases for both problems. This indicates that the more instantiated random variables that *PACE* maintains, the better the acceleration achieved by *PACE*. This is because the availability of more instantiated random variables improves the efficiency of path cost distribution computation based on *PACE*. This means that as more trajectories are collected, more random variables can be instantiated, and thus *PACE* based stochastic routing algorithms become more efficient. This is a very appealing characteristic in real-world applications.

Summary The empirical study can be summarized in four key observations.

(1) In realistic settings with sparse data, the proposed path cost distribution estimation method based on *PACE*, i.e., *OD*, 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) The *PACE* paradigm can accommodate existing stochastic routing algorithms using two different approaches, with the effect of improving their efficiency and accuracy.

We conclude that the proposed *PACE* paradigm successfully address the challenges caused by *data sparseness*, *complex distributions*, and *dependencies*. In particular, the *PACE* based path cost distribution computation method is able to efficiently provide accurate travel cost distribution computation, thus enabling efficient and accurate stochastic routing algorithms based on *PACE*.

7 Related work

We first review recent studies on estimating *deterministic* and *uncertain* path costs, respectively. Then, we review literature on stochastic routing.

Estimating deterministic path costs Most studies in this category 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 [21,34,40,42] 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 [21,40,42]. 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 [34] 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 [14], 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, 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 capture such hard-to-formalize factors. Further, the study [14] 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 [4,37,38] exemplified in Fig. 1b in Sect. 1 and indicated in Fig. 14a in Sect. 6. We use multi-dimensional histograms to represent arbitrary distributions.

Second, some studies assume that the distributions on different edges are independent of each other [6,24] or conditionally independent given the arrival times at different edges [38], thus mirroring the LB approach covered in Sect. 6. The independence assumption often does not hold (cf. Sect. 2), and our approach outperforms LB, as shown in Sect. 6. Further, with the exception of one study [38], all the above studies use synthetic distributions in empirical evaluations; we use large, real trajectory data.

Table 3 Stochastic routing

	Single cost	Multiple costs
Time-homogeneous	[2,6,20,31]	[36]
Time-varying	[2,28,35]	[5,18,27,38]

The most advanced method, the HP [20] approach covered in Sect. 6, 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 [19,41] employ histograms to represent travel cost distributions, they consider travel cost distributions on individual edges and assume that distributions are independent.

Stochastic routing Compared to regular routing algorithms [10,25] that assume deterministic travel costs, stochastic routing algorithms extend to travel costs that are uncertain. Given a source, a destination, a departure time, and some other parameters, e.g., a time budget or a probabilistic threshold, a stochastic routing algorithm identifies a path or a set of paths whose cost distribution satisfy the conditions specified by the input parameters.

Different routing algorithms have been proposed to solve different types of routing problems, e.g., stochastic fastest routing [24], probabilistic top-*k* routing [20], stochastic skyline routing [38], and non-dominated routing [2]. Table 3 summarizes existing stochastic routing algorithms according to two aspects: whether they consider multiple travel costs and whether they consider time-varying uncertainty.

Instead of proposing new types of stochastic routing problems, this paper presents the *PACE* paradigm that enables more accurate and more efficient stochastic routing and shows how to integrate existing stochastic routing algorithms seamlessly into *PACE* to improve both the accuracy and efficiency of these algorithms.

8 Conclusion

This study provides a new and promising paradigm called *PACE*, where travel costs are associated not only with road network edges, but with paths. In *PACE*, we model joint distributions that capture the travel cost dependencies among paths that form longer paths, which in turn enables accurate travel cost estimation of any path using sparse historical trajectory data. In addition, we show how existing stochastic routing algorithms can be integrated seamlessly into *PACE*, resulting in improved accuracy and efficiency. Empirical studies in realistic settings offer insight into the design properties of the *PACE* paradigm and suggest that *PACE* is effective and efficient.

Acknowledgements 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.

References

1. Abraham, I., Delling, D., Goldberg, A.V., Werneck, R., Fonseca, F.: Hierarchical hub labelings for shortest paths. In: ESA, pp. 24–35 (2012)
2. Aljubayrin, S., Yang, B., Jensen, C.S., Zhang, R.: Finding non-dominated paths in uncertain road networks. In: SIGSPATIAL, pp. 15:1–15:10 (2016)
3. Andersen, O., Jensen, C.S., Torp, K., Yang, B.: EcoTour: reducing the environmental footprint of vehicles using eco-routes. In: MDM, pp. 338–340 (2013)
4. Asghari, M., Emrich, T., Demiryurek, U., Shahabi, C.: Probabilistic estimation of link travel times in dynamic road networks. In: SIGSPATIAL, pp. 47:1–47:10 (2015)
5. Chang, T.-S., Nozick, L.K., Turnquist, M.A.: Multiobjective path finding in stochastic dynamic networks, with application to routing hazardous materials shipments. Transp. Sci. **39**(3), 383–399 (2005)
6. Chen, A., Ji, Z.: Path finding under uncertainty. J. Adv. Transp. **39**(1), 19–37 (2005)
7. Dai, J., Yang, B., Guo, C., Ding, Z.: Personalized route recommendation using big trajectory data. In: ICDE, pp. 543–554 (2015)
8. Dai, J., Yang, B., Guo, C., Jensen, C.S., Jilin, H.: Path cost distribution estimation using trajectory data. PVLDB **10**(3), 85–96 (2016)
9. Darroch, J.N., Speed, T.P.: Additive and multiplicative models and interactions. Ann. Stat. **11**(3), 724–738 (1983)
10. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
11. Ding, Z., Yang, B., Chi, Y., Guo, L.: Enabling smart transportation systems: a parallel spatio-temporal. IEEE Trans. Comput. **65**(5), 1377–1391 (2016)
12. Ding, Z., Yang, B., Güting, R.H., Li, Y.: Network-matched trajectory-based moving-object database: models and applications. IEEE Trans. Intell. Transp. Syst. **16**(4), 1918–1928 (2015)
13. Geisberger, R., Sanders, P., Schultes, D., Delling, D.: Contraction hierarchies: faster and simpler hierarchical routing in road networks. In: WEA, pp. 319–333 (2008)
14. Geisberger, R., Vetter, C.: Efficient routing in road networks with turn costs. In: SEA, pp. 100–111 (2011)
15. Guo, C., Jensen, C.S., Yang, B.: Towards total traffic awareness. SIGMOD Rec. **43**(3), 18–23 (2014)
16. Guo, C., Ma, Y., Yang, B., Jensen, C.S., Kaul, M.: Ecomark: evaluating models of vehicular environmental impact. In: SIGSPATIAL, pp. 269–278 (2012)
17. Guo, C., Yang, B., Andersen, O., Jensen, C.S., Torp, K.: Ecomark 2.0: empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data. GeoInformatica **19**(3), 567–599 (2015)
18. Guo, C., Yang, B., Andersen, O., Jensen, C.S., Torp, K.: Ecosky: reducing vehicular environmental impact through eco-routing. In: ICDE, pp. 1412–1415 (2015)
19. Jilin, H., Yang, B., Jensen, C.S., Ma, Y.: Enabling time-dependent uncertain eco-weights for road networks. GeoInformatica **21**(1), 57–88 (2017)
20. Hua, M., Pei, J.: Probabilistic path queries in road networks: traffic uncertainty aware path selection. In: EDBT, pp. 347–358 (2010)
21. Idé, T., Sugiyama, M.: Trajectory regression on road networks. In: AAAI, pp. 203–208 (2011)
22. Jagadish, H.V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K.C., Suel, T.: Optimal histograms with quality guarantees. In: VLDB, pp. 275–286 (1998)
23. Kaul, M., Yang, B., Jensen, C.S.: Building accurate 3D spatial networks to enable next generation intelligent transportation systems. In: MDM, pp. 137–146 (2013)
24. Lim, S., Sommer, C., Nikolova, E., Rus, D.: Practical route planning under delay uncertainty: stochastic shortest path queries. In: Proceedings of "Robotics: Science and Systems VIII", paper number 32 (2012)
25. Liu, H., Jin, C., Yang, B., Zhou, A.: Finding top-k shortest paths with diversity. TKDE, 1–15 (2017). **(online first)**
26. Malvestuto, F.M.: Approximating discrete probability distributions with decomposable models. IEEE Trans. Syst. Man Cybern. **21**(5), 1287–1294 (1991)
27. Miller-Hooks, E., Mahmassani, H.: Optimal routing of hazardous materials in stochastic, time-varying transportation networks. Transp. Res. Rec. J. Transp. Res. Board **1645**, 143–151 (1998)
28. Miller-Hooks, E., Mahmassani, H.S.: Path comparisons for a priori and time-adaptive decisions in stochastic, time-varying networks. Eur. J. Oper. Res. **146**(1), 67–82 (2003)
29. Newson, P., Krumm, J.: Hidden Markov map matching through noise and sparseness. In: SIGSPATIAL, pp. 336–343 (2009)
30. Niknami, M., Samaranayake, S.: Tractable path finding for the stochastic on-time arrival problem. In: SEA, pp. 231–245 (2016)
31. Nikolova, E., Brand, M., Karger, D.R: Optimal route planning under uncertainty. In: ICAPS, pp. 131–141 (2006)
32. Sabran, G., Samaranayake, S., Bayen, A.: Precomputation techniques for the stochastic on-time arrival problem. In: ALENEX, pp. 138–146. SIAM (2014)
33. Smyth, P.: Model selection for probabilistic clustering using cross-validated likelihood. Stat. Comput. **10**(1), 63–72 (2000)
34. Wang, Y., Zheng, Y., Xue, Y.: Travel time estimation of a path using sparse trajectories. In: SIGKDD, pp. 25–34 (2014)
35. Wellman, M.P., Ford, M., Larson, K.: Path planning under time-dependent uncertainty. In: UAI, pp. 532–539 (1995)
36. Wijeratne, A.B., Turnquist, M.A., Mirchandani, P.B.: Multiobjective routing of hazardous materials in stochastic networks. Eur. J. Oper. Res. **65**(1), 33–43 (1993)
37. Yang, B., Guo, C., Jensen, C.S.: Travel cost inference from sparse, spatio-temporally correlated time series using markov models. PVLDB **6**(9), 769–780 (2013)
38. Yang, B., Guo, C., Jensen, C.S., Kaul, M., Shang, S.: Stochastic skyline route planning under time-varying uncertainty. In: ICDE, pp. 136–147 (2014)
39. Yang, B., Guo, C., Ma, Y., Jensen, C.S.: Toward personalized, context-aware routing. VLDB J. **24**(2), 297–318 (2015)
40. Yang, B., Kaul, M., Jensen, C.S.: Using incomplete information for complete weight annotation of road networks. IEEE Trans. Knowl. Data Eng. **26**(5), 1267–1279 (2014)
41. Yuan, J., Zheng, Y., Xie, X., Sun, G.: T-drive: enhancing driving directions with taxi drivers' intelligence. IEEE Trans. Knowl. Data Eng. **25**(1), 220–232 (2013)
42. Zheng, J., Ni, L.M: Time-dependent trajectory regression on road networks via multi-task learning. In: AAAI, pp. 1048–1055 (2013)