Tiering in Contraction and Edge Hierarchies for Stochastic Route Planning

Payas Rajan

payas.rajan@email.ucr.edu

Department of Computer Science & Engineering
University of California
Riverside, California, USA

Chinya V. Ravishankar

ravi@cs.ucr.edu

Department of Computer Science & Engineering
University of California
Riverside, California, USA

ABSTRACT

Stochastic route planning is a hard problem, since it deals with uncertain edge weights, usually modeled as probability distributions. Stochastic shortest path queries are very expensive, as they must compute convolutions of edge weight distributions, whose representations can have a major impact on query costs. Effective speedup techniques for shortest path queries exist for deterministic edge weights, but their extensions to stochastic settings have had limited success, and real-time stochastic routing queries remain beyond reach. We introduce the *tiering* technique for Contraction and Edge Hierarchies (CHs and EHs) to address this challenge. We divide the hierarchy into tiers, and represent edge weights in each tier in ways that permit effective tradeoffs between accuracy, convolution costs, and space use. We show how to use Gaussians to approximate histograms, and bound errors using the KL divergence and Hellinger distance measures. We develop Uncertain Contraction Hierarchies (UCHs) and Uncertain Edge Hierarchies (UEHs) using these methods, and show that they improve both CH and EH performance for three different stochastic query types: probabilistic budget routes, non-dominated routes, and routes to minimize the mean-risk objective. We evaluate our methods using real-world data from Mapbox Traffic Data for a section of Los Angeles. Finally, our results show that query times for EHs can be competitive with CHs for stochastic edge weights, contrary to current belief.

CCS CONCEPTS

• **Theory of computation** → **Shortest paths; Routing and network design problems.**

KEYWORDS

Stochastic Route Planning, Contraction Hierarchies, Edge Hierarchies

ACM Reference Format:

Payas Rajan and Chinya V. Ravishankar. 2021. Tiering in Contraction and Edge Hierarchies for Stochastic Route Planning. In *29th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '21)*, November 2–5, 2021, Beijing, China. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3474717.3484267>

This work is licensed under a Creative Commons Attribution International 4.0 License. *SIGSPATIAL '21*, November 2–5, 2021, Beijing, China
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8664-7/21/11.
<https://doi.org/10.1145/3474717.3484267>

1 INTRODUCTION

For route planning, road networks have traditionally been modeled as graphs with deterministic edge weights representing travel times. In practice, however, no two vehicles travelling along the same road segment can be expected to take the *exactly* same amount of time. A more accurate model would represent edge weights as stochastic quantities, drawn from discrete or continuous travel time distributions. Fixed edge weights yield unique shortest paths, but with stochastic edge weights, one can define shortest paths only in a probabilistic sense. The shortest path computation is also more complex, since edge weight distributions cannot be directly compared. Route planning with stochastic edge weights allows for several types of shortest path queries [1, 2, 12, 14, 15, 23, 28, 29, 31–36, 38, 42–45]. Stochastic route planning is a much harder problem than its deterministic counterpart, due to the high cost of computing convolutions [25]. With deterministic edge weights, routing is feasible in times on the order of microseconds on continental road networks [4, 20, 40]. With stochastic edge weights, however, the best known routing methods, depending on the routing objective, take time on the order of seconds [1, 36, 42] even on city-sized road networks. Even in restricted parameter settings [28], stochastic routing is only known to be feasible in time on the order of milliseconds, using speedup techniques for deterministic edge weight graphs.

1.1 Contraction and Edge Hierarchies

Contraction Hierarchies (CHs) [19, 20] and Edge Hierarchies (EHs) [22] are speedup techniques originally developed for deterministic route planning, and find shortest paths in two stages. In the preprocessing stage, *shortcut edges* are added to the graph. In the query stage, shortcut edges help answer shortest path queries quickly. CHs and EHs are similar, and work as follows: given a road network modeled as a graph with travel times or distances as edge weights, each vertex (or edge) is first assigned a *rank*, and the *contraction* operation is applied to all vertices (or edges) in increasing rank order. Contracting a vertex (or an edge) adds a shortcut edge to the graph if it lies on the shortest path between two of its neighbour vertices (or edges). The query stage runs a bidirectional Dijkstra's algorithm from the source and target vertices, settling only vertices (relaxing only edges) with ranks higher than the source or target.

1.2 Handling Uncertain Edge Weights

CHs have been applied in uncertain settings [36], but no prior work exists on applying EHs in stochastic contexts. For deterministic routing, EHs have higher preprocessing costs since they apply the

Figure 1: Histograms are better lower hierarchy levels, as edge-weight distributions can be complex. Edge weights for shortcuts at higher levels are convolutions, which converge to Gaussians, which are compact, fast, and accurate.

contraction operation per edge, thereby offering a finer grained hierarchy. They are also known to have worse query times than CHs. Nevertheless, EHs are more selective in relaxing edges in the query phase, and hold the promise of better performance when relaxing edges is computationally expensive. Finding stochastic shortest paths is one such application.

When edge weights are modeled as probability distributions, a major cost of finding shortest paths is computing edge weight convolutions. Edge weight distributions can be represented variously, using histograms and continuous functions [33, 42]. These choices can significantly affect shortest path query times, as they make different error, convolution cost, and space usage tradeoffs.

1.2.1 The Tiering Idea. We present the *tiering* technique for CHs and EHs, which works as follows: the set of shortcut edges in a CH or EH with uncertain weights is divided into a series of *tiers*, each of which uses an edge weight representation suitable for that tier. For instance, lower-ranked shortcut edges are likely to connect local vertices, with travel time distributions that do not resemble standard distributions. Histograms represent arbitrary distributions well, and are likely to be better representations here. However, higher ranked shortcuts in the CH or EH connect vertices farther away, and represent travel over many graph edges. Their edge weight distributions are convolutions over multiple distributions, and likely to converge to stable distributions, such as the Gaussian. When edge weight distributions resemble Gaussians (as measured by the KL divergence or Hellinger distance), representing them as Gaussians offers fast convolutions, and is accurate and compact.

We apply *tiering* to both EHs and CHs, and present Uncertain Contraction Hierarchies (UCHs) and Uncertain Edge Hierarchies (UEHs), whose edge weights represent stochastic travel times. Given a graph $G = \langle V, E \rangle$, a source $s \in V$ and destination $t \in V$, we study UCHs and UEHs for three types of stochastic routing queries:

- (1) *Non-Dominated Routes*: Find s - t routes that have edge weight distributions not dominated by other routes.
- (2) *Probabilistic budget routes*: Find a route that maximises the probability of travel cost being within a given budget b .
- (3) *Mean-risk routes*: Given a risk aversion coefficient $c \geq 0$, find routes minimizing (mean travel time + $c * \sqrt{\text{variance}}$).

We show that when coupled with proper heuristics, tiered UCHs and UEHs offer faster stochastic routing queries than their non-tiered variants for all three query types. Further, we build the UEH and UCH using both KL divergence and Hellinger distance measures. These similarity measures offer different tradeoffs: KL divergence yields faster queries, while Hellinger distance permits theoretical bounds on the approximation errors in the shortest path queries.

Finally, contrary to findings in current literature, we show that for all three query types considered, UEHs can have query times comparable to UCHs. UEHs still require higher preprocessing times, since although the query algorithms for both EH and CH are very similar, EHs relax fewer edges due to their finer grained hierarchy, but CHs offer far better stalling performance.

2 BACKGROUND

2.1 Stochastic Route Planning

Stochastic route planning dates to as far back as 1968, when [18] presented a Monte Carlo method to estimate the joint probability distribution of the shortest path in a graph with probabilistic edge weights. Recent work on stochastic routing can be categorized along several axes: i) *path* versus *edge*-centric routing algorithms, which use ii) *static* versus *adaptive* edge weights, which are represented as iii) *discrete* versus *continuous* distributions.

Path vs. edge-centric routing: Most work on both deterministic and stochastic routing is edge-centric [28, 29, 32, 34], assigning to each edge in the routing graph weights or distributions, which are assumed to be independent. However, travel times along network edges are often correlated, so some recent works [2, 16, 42, 43] use paths instead of edges as the smallest unit for routing.

Static vs. adaptive edge weights: Several works model stochastic edge weights as static probability distributions given along with the routing graph [32, 33, 36]. In this case, edge weights can be derived from given distributions by sampling, and we are to find the shortest path for a given definition of ‘shortness’. In other problem definitions, such as the Stochastic On-Time Arrival (SOTA) and the Shortest Path problem considering On-time Arrival Reliability (SPOTAR) problems [2, 25, 31, 38], the exact edge weight is ‘revealed’ only when the search reaches an adjacent vertex. Here, the problems seek an optimal *policy* for the driver to follow in order to have the highest probability of reaching the destination before deadline.

Discrete vs. continuous distributions: Edge weight distributions may be modeled as functions [28, 32–34], or as histograms [2, 36, 42]. Histograms discretize time, and are easy to create from spatiotemporal probe data, but perform well only with sufficient data. Functions are difficult to obtain, but do not depend on the availability of data. A very recent preprint [24] attempts to bridge this divide by combining the advantages of the two representations.

2.2 Speedup Techniques: CHs and EHs

Edge Hierarchies were introduced in [22] as a speedup technique for deterministic routing, and are closely related to Contraction Hierarchies [19, 20]. Both techniques first assign ranks to all vertices or edges, and then apply the *contraction* operation per vertex or edge to add *shortcut* edges to the graph in the preprocessing stage. The query phase then consists of running slightly modified bidirectional Dijkstra’s search, where both forward and backward searches only

settle vertices (or relax edges) that are ranked higher than the source in forward search and the target in the backward search.

For both CHs and EHs, the ranks assigned to vertices or edges can make a significant difference in both preprocessing and query times. Good contraction orders minimize both total number of shortcuts added and query search space sizes, but finding optimal contraction orders for CHs is NP-hard [30]. However, [8] show that nested dissection can be used to derive provable upper bounds on the search space size for CHs. In EHs [22], the edges are ranked in *rounds*. At the beginning of each round, a subset of remaining unranked edges is picked. Edges are picked in increasing order of the number of shortcuts that would be added if they were to be ranked in the current round. The next round begins after all edges in the current round have been ranked.

The query stages of both CHs and EHs use *stalling* techniques to terminate search early, which significantly lower query times. CHs are known to perform well with *stall on demand*, where forward search terminates at $v \in V$ when a shortest path can not be found via the incoming edges in the backward search [22]. Similarly, the backward search terminates at $v \in V$ when a shortest path cannot be found via outgoing edges in the forward search. However, stalling on demand can be wasteful as it may relax every edge twice: once for settling, and once for stalling. Therefore, EHs use *stall in advance*, where the search relaxes every edge at most once. This works as follows: when search reaches vertex v all edges (v, v') that are higher or lower ranked than vertex v in the search are relaxed. The updated distance for lower ranked edges is then stored in a separate label, which can be used to check the distance in the stalling check.

The results in [22] show that CHs outperform EHs significantly, due to better stalling performance. However, we show in Section 5 that this performance disparity is not intrinsic; if relaxing edges is expensive, EH performance can be very similar to that of CHs.

Another variant of CH related to stochastic routing is the Time-Dependent CH (TD-CH) [5–7], which uses edge weight functions to assign travel times corresponding to different arrival times. However, unlike stochastic routing definitions, TD-CH does not allow different travel times for the same arrival time at the same edge.

3 PROBLEM SETUP

We are given a graph $G = \langle V, E \rangle$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges, and a stochastic edge weight function $W : E \rightarrow \mathbf{R}$ mapping each edge e to a random variable $R_e \geq 0$. A *path* is a sequence of vertices $[v_0, v_1, \dots, v_n]$ where $(v_i, v_{i+1}) \in E$. An *s-t path* is a path $[s = v_0, v_1, \dots, v_n = t]$ between s and t .

The cost of path P is the aggregation of all stochastic edge weights along P , and is denoted $\text{cost}(P)$. For path $P = [e_1, e_2 \dots e_k]$, with weight distributions $W(e_1), W(e_2), \dots, W(e_k)$, we obtain the aggregate distribution along P using the convolution $\otimes_{i=1}^k W(e_i)$.

We assume that the edge weights satisfy the *First-In-First-Out (FIFO) property*. This ‘no overtake’ rule guarantees that vehicles using the same path complete the trip in the same order that they started it [29, 36, 45]. We are interested in three types of stochastic routing queries: probabilistic budget routing, non-dominated routes, and routes that optimize the mean-risk objective.

In *probabilistic budget routing* [36], we are given a source $s \in V$, a target $t \in V$ and a cost budget $b \geq 0$. We are to find an *s-t path* P that

maximizes the probability that $\text{cost}(P) \leq b$. The standard example of such a route would be a driver trying to reach an airport before a deadline b , which is the budget. We want a route that maximizes the probability of reaching the airport before the deadline b .

Non-dominated routing or *Pareto-optimal routing* seeks the full set of paths between source and destination vertices that are not *dominated* by other paths. A path P is said to *dominate* another path P' if the travel times for P are always lower than that for P' .

The *mean-risk* objective minimizes a linear combination of the mean and standard deviation of edge weights along the path [32]. A typical application is to model travel time *delays* along a path. We have a graph $G = \langle V, E \rangle$ and two edge weight functions $\mu : E \rightarrow \mathbf{R}_{\geq 0}$ and $\tau : E \rightarrow \mathbf{R}_{\geq 0}$ that map every edge e to the mean μ_e and variance τ_e of travel time delay for each $e \in E$, a source $s \in V$, a target $t \in V$, and a risk-aversion coefficient $c \geq 0$. Our objective is to find an *s-t path* P that minimizes $\sum_{e \in P} (\mu_e + c\sqrt{\tau_e})$. A quasi-linear shortest-path algorithm under the mean-risk model first appeared in [32]. It approximates the convex hull of the level set of feasible solutions using linear separation oracles without restricting c . However, by restricting c , [28] achieved sub-linear query times by applying distance oracles [41] to speed up queries.

4 UNCERTAIN HIERARCHIES

Three major factors affect the performance of speedup techniques beyond low-level optimizations: the structure of the road network, the available ‘hierarchy’ in edge weights that can be exploited by adding shortcuts to the graph, and the runtime cost of basic operations required to compute edge weights. For instance, CHs are known to perform well for graphs with a low road or skeleton dimension, and perform much better if travel times are used as edge weights, rather than physical distances between the vertices [10, 26]. Further, the cost of operations required on edge weights is a significant component of the algorithm engineering required for data structures such as the Time-dependent Uncertain Contraction Hierarchies [36]. Our goal in this section is to develop Uncertain Contraction and Edge Hierarchies for stochastic routing.

4.1 Tiering in Hierarchies

Travel-time distributions are often derived from collected trajectory data [13, 28, 36, 42] or other traffic sensors [3]. Broadly, there are two ways to represent uncertain edge weights: using histograms [2, 36, 42] or using continuous functions [23, 28, 33, 43]. Speed-up techniques for graphs with uncertain edge weights benefit greatly if the edge weight representations have the following properties:

- (1) *Accuracy*: The representation should capture all the information about the edge cost distribution without errors.
- (2) *Cheap convolutions*: Convolution is a basic operation in finding shortest paths, so representations that offer cheaper convolutions can improve query performance.
- (3) *Space efficiency*: Compact edge weight representations can have improve cache performance, reducing query times.

Histograms and continuous distributions make different tradeoffs between these properties. Most real-world data is collected by periodic, not continuous sampling, so histograms are usually the most accurate representations of available information. However, the source distributions can be complex, so histogram convolutions

can be expensive. In contrast, convolutions are very fast when edge weights match stable distributions such as the Gaussian [28, 33].

Given two Gaussians $F = \mathcal{N}(\mu_1, \sigma_1^2)$ and $G = \mathcal{N}(\mu_2, \sigma_2^2)$, we have $F \otimes G = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$, so convolution is just two additions. More complex representations, like Gaussian Mixture Models, can be less compact and have high convolution costs [43].

Definition 4.1. A tier T in a CH or EH is the set of shortcut edges with ranks $\tau_{\min}^T \leq r(e) < \tau_{\max}^T$ for given thresholds $\tau_{\min}^T$ and $\tau_{\max}^T$.

A tier T is marked as a *histrogram tier* or a *function tier* depending on whether histograms or continuous functions are used to represent the edge weights in T . In a histogram tier, a histogram with a fixed bucket width w and b buckets is used to represent edge weights. Similarly, edge weights in a function tier are represented by a mixture of one or more stable probability distributions.¹

Definition 4.2. A *tiered* contraction or edge hierarchy is a series of tiers $[T_1, T_2, \dots, T_N]$ such that $\tau_{\min}^{T(i+1)} = \tau_{\max}^{T_i} + 1$, $i = 1, \dots, N$.

The next problem is to choose the number and type of tiers for the hierarchy. Here, the Central Limit Theorem suggests a useful heuristic: edges with weights derived from a large number of convolutions are likely to have distributions that approximate the Gaussian. Using this heuristic, we use a *two-tiered* contraction or edge hierarchy, which contains a histogram tier H with thresholds $\tau_{\min}^H$ and $\tau_{\max}^H$, and a Gaussian tier G with thresholds $\tau_{\min}^G$ and $\tau_{\max}^G$ that uses Gaussian edge weights in G .

A similar heuristic for pruning Dijkstra's search is used in [36]. However, an important difference is that we use the Central Limit Theorem to alter the *structure* of the contraction or edge hierarchy while building the hierarchies, while [36] uses the heuristic in the query stage. Also, we do not consider time-varying edge weights.

We model Uncertain Edge Hierarchies (UEHs) as two-tiered hierarchies with Histogram and Gaussian tiers. We now show their construction and use to answer the three types of stochastic routing queries under consideration.

4.1.1 Preprocessing. As in [22], all edges in G are ranked in rounds. We iterate over unranked edges, and compute the number of edges that would be added to G if they were ranked in this round. Then, we pick a set of edges that would add the minimum number of edges among their neighbor edges and add them to the ranking set R in increasing order. We then rank edges in R in the current round.

Let $r(u, v)$ denote the rank of edge (u, v) . In a round, on every iteration, an unranked edge (u, v) is picked, and the *contraction* operation applied to it as follows: a Dijkstra's run is used to determine if (u, v) lies on the shortest path between any unranked edges (u', u) and (v, v') . If it does, shortcut edges (u, v') and (u', v) are added to a set S . Next, the algorithm computes a minimum vertex cover of the bipartite graph in S . The edges adjacent to vertices that remain after the minimum vertex cover computation over S are then added to G and (u, v) is removed.

¹A stable probability distribution is one such that the linear combination of two or more random variables with the distribution results in the same distribution with different parameters.

Figure 2: Tiering in shortcut hierarchies. Distributions are represented as histograms in Tier H, and as Gaussian approximations in Tier G.

4.2 Uncertain Edge Hierarchies

4.2.1 Determining the Tier Thresholds. We must next find suitable thresholds for the histogram tier H and the Gaussian tier G of the UEH. Since H is the lower tier, $\tau_{\min}^H = 1$.

We find $\tau_{\min}^G$ as follows. Since the number of edges is very large, we use sampling, and examine only one in λ edges. Let e be a sampled edge with edge weight histogram $W(e)$. We first construct a Gaussian distribution $\mathcal{N}(\mu_e, \sigma_e^2)$ from $W(e)$ [11]. We next sample this Gaussian and construct a histogram $G(e)$. Finally, we compute the KL divergence [27] (or Hellinger distance [21]) between $G(e)$ and $W(e)$ for the shortcut edges e being added. Given two discrete distributions G, W on a probability space χ ,

$$D_{KL}(W||G) = \sum_{x \in \chi} W(x) \log \frac{W(x)}{G(x)}.$$

If the KL-divergence is less than a predetermined similarity threshold σ_T for all the edges to be added, we set $\tau_{\min}^G = \lambda$ and $\tau_{\max}^H = \lambda - 1$. Algorithm 1 shows the pseudocode for building UEHs. The preprocessing phases for UEHs built using both KL divergence and Hellinger distance measures are identical, except for the use of different measures on Line 17 of Algorithm 1.

The sampling frequency λ and similarity threshold σ_T are UEH configuration parameters. λ depends on the number of edges in the EH, so $1 \leq \lambda \leq |E|$. The parameter λ presents a tradeoff between preprocessing times and query times. Similarly, σ_T trades off the *accuracy* for query times. A low λ means that we check for threshold of tiers H and G more often while building the UEH, possibly lowering query times but raising preprocessing costs. Conversely, setting λ too high would put more edges in tier H than strictly required, offsetting the benefits of cheaper convolutions in tier G . On the other hand, setting a high σ_T means we approximate edge weight histograms with Gaussian distributions even when the two differ significantly, reducing query accuracy. Finally, setting σ_T too low slows down queries as the UEH contains mostly edge weight histograms, for which convolutions are expensive.

4.3 Stochastic Query Processing

Since each edge is assigned an integer rank, as in the deterministic EH, there exists an *up-down* path between source $s \in V$ and target $t \in V$ [22]. For every vertex in G , the search algorithm is a

Algorithm 1 Building the two-level Uncertain Edge Hierarchy

```

1: procedure BUILDEDGEHIERARCHY
2:    $currentRank \leftarrow 0$ 
3:   Set current tier to  $H$ 
4:   while unranked edges remain in  $G$  do
5:     Pick unranked edge  $(u, v)$ ;  $r(u, v) \leftarrow currentRank$ 
6:     for unranked  $(u', u)$  do
7:       for unranked  $(v, v')$  do
8:         if  $(u, v)$  lies on shortest path from  $u'$  to  $v'$  then
9:            $S \leftarrow S \cup \{(u', u), (v, v')\}$ 
10:        end if
11:      end for
12:    end for
13:     $MVC(S) \leftarrow$  Bipartite Minimum Vertex Cover over  $S$ 
14:    if  $currentRank$  is a multiple of  $\lambda$  then
15:      for  $e \in MVC(S)$  do
16:         $\mathcal{N}(\mu_e, v_e^2) \leftarrow$  Gaussian approximation of  $W(e)$ 
17:         $KL \leftarrow KL \cup \{D_{KL}[W(e) \parallel \mathcal{N}(\mu_e, v_e^2)]\}$ 
18:      end for
19:      if all edges in  $KL$  have similarity  $< \sigma_T$  then
20:        Set current tier to  $G$ 
21:      end if
22:    end if
23:    Add edges in  $MVC(S)$  to current tier
24:     $currentRank \leftarrow currentRank + 1$ 
25:  end while
26: end procedure

```

bidirectional Dijkstra's run from s and t , which first sets the vertex rank labels at s and t to 0. Then, search in both directions expands only edges with rank greater than the rank label of v after reaching vertex v . If the distance of a vertex v from source is updated in the priority queue while relaxing edge (u, v) , rank label of u is set to $r(u, v)$. Since UEHs have edge weights represented as either histograms or continuous functions in different tiers, as the search progresses, it may relax edges from one or both tiers H and G . Whenever a Dijkstra's search starting from tier $T \in \{H, G\}$ reaches some edge $e \notin T$, we call e a *boundary edge*.

LEMMA 4.3. *A shortest path between any two vertices s and t in a UEH can contain at most two boundary edges.*

PROOF. Since all shortest paths in a UEH are up-down paths, let $P = [s = v_1, v_2, \dots, v_m, \dots, v_n = t]$ be a shortest path such that $r((v_i, v_{i+1})) < r((v_{i+1}, v_{i+2}))$ for $1 \leq i < m$, and $r((v_j, v_{j+1})) > r((v_{j+1}, v_{j+2}))$ for $m \leq j \leq n$. Only one of two cases can arise.

First, if $\text{rank } r((v_{m-1}, v_m)) < \tau_{min}^G$, all edges in P lie in tier H , and P has no boundary edges. Second, if $r((v_{m-1}, v_m)) \geq \tau_{min}^G$, we must have $i, j \in [1, n]$ such that $r((v_i, v_{i+1})) \leq \tau_{min}^G \leq r((v_{i+1}, v_{i+2}))$, and $r((v_j, v_{j+1})) \geq \tau_{min}^G \geq r((v_{j+1}, v_{j+2}))$. Then, (v_{i+1}, v_{i+2}) and (v_{j+1}, v_{j+2}) are the two boundary edges in P . $\square$

4.3.1 Error Bounds for KL Divergence. In a UEH, the weight of a shortcut edge e is the convolution of all edge weight distributions that the shortcut replaces. However, if e lies in tier G , we store only a Gaussian *approximation* of the exact distribution such that the

maximum KL divergence between the two is σ_T . The maximum error this approximation induces for an edge weight is given by Pinsker's inequality [37]:

$$\|W(e) - \mathcal{N}(\mu_e, v_e^2)\|_1 \leq \sqrt{2\sigma_T}, \text{ where} \quad (1)$$

$$\|W(e) - \mathcal{N}(\mu_e, v_e^2)\|_1 = \sup\{|W(e)(x) - \mathcal{N}(\mu_e, v_e^2)(x)|, x \in \mathbb{R}\}$$

Here, $\|W(e) - \mathcal{N}(\mu_e, v_e^2)\|_1$ is the L_1 distance between the exact edge weight distribution of edge e , $W(e)$ and $\mathcal{N}(\mu_e, v_e^2)$, its Gaussian approximation. The L_1 distance is the maximum difference between two values for any observation $x \in \mathbb{R}$ for which they are defined.

As longer shortcuts are added to the UEH at higher levels, we can determine the rate at which the edge weight distributions converge to Gaussians. Using Lemma 4.3, an s - t shortest path P can be divided into a sequence of three subpaths $[P_H, P_G, P_{H'}]$ where P_H and $P_{H'}$ lie in tier H and P_G lies in tier G . The edge weights in P_H and $P_{H'}$ are exact histograms that induce no error, while those in P_G use approximations with Gaussian distributions. Let $s(e)$ represent the edges in G that each shortcut edge $e \in P_G$ replaces. Then, the total number of "unpacked" edges in P_G is $\sum_{e \in P_G} s(e)$.

The rate at which convolutions of edge weights in P_G converge to a Gaussian distribution is given by the Berry-Esseen theorem [9, 17]. Let $X_1, X_2, \dots, X_n$ be independent random variables with $E[X_i] = 0, E[X_i^2] = \sigma_i^2 > 0, E[|X_i|^3] = \rho < \infty$. Let F_n be the CDF of $S_n = (\sum_{i=1}^n X_i) / \sqrt{\sum_{i=1}^n \sigma_i^2}$. Now,

$$\sup_{x \in \mathbb{R}} |F_n(x) - \Phi(x)| \leq C_0 \left(\sum_{i=1}^n \sigma_i^2 \right)^{-\frac{3}{2}} \sum_{i=1}^n \rho_i, \quad (2)$$

where $\Phi(x)$ is the standard Gaussian, and $0.4097 \leq C_0 \leq 0.56$ [39]. Equation 2 can be used to get the rate of convergence and error in a shortest path query only after all the edges along the shortest path from s to t are known.

4.3.2 Using the Hellinger Distance. An alternative approach to the UEH would be to use the Hellinger Distance (HD) [21] instead of KL-Divergence for constructing the hierarchy. For discrete probability distributions $P = \{P_1, P_2, \dots, P_k\}$ and $Q = \{Q_1, Q_2, \dots, Q_k\}$, the Hellinger distance is given by:

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^k (\sqrt{P_i} - \sqrt{Q_i})^2}$$

Unlike KL Divergence, HD satisfies the triangle inequality, which can be used to reduce the approximation error by storing the HD in each edge weight in tier G , and including it in the dominance criterion in the query phase. Each stochastic routing query then returns the pareto-optimal set of routes with two objectives: to minimize the distance from s to t , and to minimize the approximation error. We call this variant of UEHs the *HD-UEH*, and the originally introduced version the *KLD-UEH*.

4.3.3 Query Processing Details. We now consider the three types of queries described in Section 4.

(1) *Non-dominated routes:* To find non-dominated routes, we maintain two label sets at all vertices. At $v \in V$, $L_{un}(v)$ and $L_{set}(v)$ respectively store the unsettled and settled labels for a Dijkstra

Figure 3: Computing approximation error on a path in UEH. Gaussian approximations are used only on the tier-G subpath. When this subpath is known, KLD-UEH uses Pinsker’s inequality (Equation 1) to find the total error, while HD-UEH propagates the error term in Dijkstra’s search labels.

search. Before each query, all sets are emptied. A bidirectional Dijkstra search starts from source s and target t , setting $r(s) = r(t) = 0$. On reaching vertex v , a Dijkstra label $\ell_v = \langle \text{dist}(v), \xi_v \rangle$ is added to $L_{un}(v)$, where $\text{dist}(v)$ represents the convolution of all edge weights on the current path from s to v , and ξ_v is the error term. In HD-UEHs, when the search reaches a vertex u , on relaxing edge (u, v) , we set $\xi_v = \xi_u + HD(u, v)$. Since HD satisfies the triangle inequality, $HD(s, v) < HD(s, u) + HD(u, v)$. However, KL divergences do not satisfy the triangle inequality, and can not be added directly to give error bounds when using KLD-UEH. Therefore, we set ξ_v to a null value, and exclude it from Dijkstra’s search labels (see Figure 3).

We bound the approximation error in an HD-UEH as follows: Assume that $P = [s = v_1, \dots, v_b, v_{b+1}, \dots, v_{b'}, v_{b'+1}, \dots, v_n = t]$ is a shortest path, where (v_b, v_{b+1}) and $(v_{b'}, v_{b'+1})$ are the boundary edges. P comprises of three subpaths $S = [v_1, \dots, v_b]$, $S' = [v_{b+1}, \dots, v_{b'}]$ and $S'' = [v_{b'+1}, \dots, v_n]$, where S and S'' lie in tier H , and S' lies entirely in tier G . Only S' induces approximation error in P . We can then quantify the error in S' as follows: Assume that a Dijkstra search starts from v_b and reaches $v_{b'}$, spawning a label ℓ' at $v_{b'}$. Let $\xi_{b'} \in \ell'$ be the error term in label ℓ' . Let $C_{S'}$ be the convolution of the edge weight histograms along the path, so $C_{S'} = \otimes_{e \in S'} W(e)$. Let $\text{cost}_{S'}$ be the convolution of the Gaussian approximations along the path. Then, $\xi_{b'}$ is an upper bound on the Hellinger distance between $\text{cost}_{S'}$ and $C_{S'}$. By the definition of Hellinger distance,

$$\xi_{b'} > \frac{1}{\sqrt{2}} \|\text{cost}_{S'} - C_{S'}\|_2$$

To compute non-dominated routes, the Dijkstra search terminates when the priority queue becomes empty, not when it reaches t .

(2) *Probabilistic budget routes* A Probabilistic Budget Route query runs exactly like a non-dominated route query, except for an additional pruning criterion: on relaxing an edge (u, v) , we compare the distance from the s to v with b . The search is terminated if reaches a vertex with distance from source greater than b .

(3) *Minimizing the mean-risk objective* Our method is similar to [28], which creates a set K of distance oracles [41] with deterministic edge weights to ϵ -approximately answer shortest path queries that minimize the mean-risk objective. In the original settings [28], for each $e \in E$, $W(e) = \mathcal{N}(\mu_e, \tau_e^2)$. Then, given an ϵ , they set $\xi = \sqrt{\frac{\epsilon}{1+\epsilon}}$, L to the minimum τ_e in the graph, and U equal to the maximum variance along any path in G . Finally, they show that it suffices to

build distance oracles with edge lengths $l_k = k \cdot \mu_e + \tau_e$ for each $k \in [L, (1 + \xi)L, (1 + \xi)^2 L, \dots, U]$, and collected in a set K .

In [28], all the edge weights distributions are Gaussian. In a UEH, in contrast, only edges weights in tier G are Gaussian. Hence, we build a set K of deterministic EHs for the subset of graph in tier G . We also set ϵ to a very low value to avoid errors from both Gaussian approximations and the route planning method. Therefore, we get ξ close to 1 and derive L and U empirically from the dataset.

A query for a given risk aversion coefficient c runs like one for Non-dominated routes, except after a Dijkstra’s search from s reaches a boundary edge b_e , the search in tier G runs a shortest path query on all EHs in set K . The path with minimum $(\mu + c\sqrt{\tau})$ among all shortest path queries is the shortest path in tier G . After the second boundary edge $b_{e'}$ along the search is reached, it progresses to target $t \in V$ by convolving of histograms on the path.

4.4 Uncertain Contraction Hierarchies

We also model Uncertain Contraction Hierarchies (UCHs) as two-tiered hierarchies.

4.4.1 Preprocessing. We proceed as with UEHs. First, all vertices are ranked using a standard heuristic [19, 20]. Then, the vertices are contracted in order of rank, and on every λ^{th} vertex contracted, we compute the KL divergence (or Hellinger distance) between the edge weight of the shortcut being added and a Gaussian with equal mean and variance. If on contracting $v \in V$, the KLD or HD of all shortcuts being added to the graph is less than σ_T , we set the threshold τ_{min}^G to the rank of vertex v .

4.4.2 Query Processing. All three query types can be handled with algorithms similar to those for UEHs, except when building UCHs to minimize the mean-risk objective, we create CHs instead of EHs for the shortcut edges in tier G . As with their deterministic versions, we use *stall on demand* for UCHs and *stall in advance* for UEHs.

4.5 Other Stable Distributions and Limitations

Our tiering heuristic uses edge weight representations at the lower levels more faithful to real-world data, but approximates with stable distributions at higher tiers, so convolutions are cheaper. However, this heuristic may not be universal. For example, some works use the log-normal and beta distributions as edge weights [25]. Our current heuristic also relies on convergence of edge weight distributions to stable distributions such as Gaussians for longer routes. We find that this works well for travel times, but further work is needed to validate this for more general distributions. Other statistical limit theorems may be helpful for constructing suitable heuristics.

5 EXPERIMENTS

To evaluate our methods, we implemented our algorithms in Rust and compiled them with rustc 1.54.0—nightly with full optimizations. All experiments were then run on an Ubuntu Linux machine running kernel 5.4.0 equipped with an Intel i5-8600K 3.6 GHz processor with 1.5 MB of L2 and 9 MB of L3 cache. The machine has 64 GBs of DDR4-2133 Mhz RAM.

Graph	Source	Vertices	Edges
New York	DIMACS	264346	733846
Bay Area	DIMACS	321270	800172
California & Nevada	DIMACS	1890815	4657742
USA West	DIMACS	6262104	15248146
Los Angeles Area	OSM	2549286	1666283
Tile 0230123 (contracted)	OSM	244728	453942

Table 1: Description of the graphs used to evaluate our method. Tile 0230123 is a part of the LA area OSM graph between Long Beach and Oxnard, covering most of LA city. We contract all vertices with degree ≤ 2 for Tile 0230123.

5.1 Baselines for Deterministic Routing

To evaluate our CH and EH implementations for deterministic edge weights, we compare the preprocessing and query times against reference implementations of Contraction and Edge Hierarchies. We use the CH implementation from the RoutingKit library² and the EH made available by the authors³. Both reference implementations recommend using the GCC compiler toolchain, so they were compiled with GCC 10.2 with full optimizations.

The graph datasets used are taken from either the 9th DIMACS challenge⁴, or an instance of the road network of Los Angeles area taken from Open Street Maps (OSM)⁵. The edge weights in DIMACS instances are pre-populated. For the OSM dataset, we first find the Vincenty’s distance between coordinates of adjacent vertices, and divide the distance by the maximum speed allowed on the road type to obtain travel times. Table 1 shows the source and the sizes of the datasets.

The work in [22] suggests that CHs outperform EHs significantly. However, their implementation uses adjacency arrays for CH, and adjacency lists for EH, confounding the effects of this difference with possible slowdown factors intrinsic to EHs. Adjacency arrays offer far better locality of reference than adjacency lists when iterating over edges in the graph, making them the preferred representation for routing with deterministic edge weights in fast libraries like RoutingKit. However, for graphs labeled with composite edge weights such as histograms, continuous functions, or both, the benefits of adjacency arrays are less clear. Further, some optimizations, like reordering edges to improve performance, are not well defined for composite edge weights. Therefore, we choose the simpler adjacency lists to implement our algorithms. The effects of this choice are clearly visible in Table 2. Our EH implementation has much better preprocessing *and* query performance than the implementation in [22]. This is because our implementation shows better cache utilization than that of [22]. For CHs, our implementation has somewhat better preprocessing times with travel times as edge weights, but worse preprocessing times with the distance metric. The query times of our CH implementation are worse than RoutingKit for both metrics due to the difference in graph representations.

²<https://github.com/RoutingKit/RoutingKit>

³<https://github.com/Hespian/EdgeHierarchies>

⁴<http://www.diag.uniroma1.it/~challenge/>

⁵<http://openstreetmap.org>

5.2 Stochastic Routing

Our CH and EH implementations are generic, and support both scalar or composite edge weights such as histograms and continuous functions. We benchmark the three kinds of stochastic queries on Tileset 0230123, a subset of the Los Angeles OSM graph.⁶

The travel time distributions are obtained from the Mapbox Traffic Data⁷ for Tile ID 0230123, containing the edge travel times sampled at 5-minute intervals. The dataset contains traffic data for four and a half months between 15th July and 30th November 2019, giving us 42,299 travel time updates of the underlying graph edges. These updates were grouped into 30-minute intervals over the 24 hours in a day, then histograms extracted separately for weekdays and weekends. The weekend travel time histograms are much sparser than those for weekdays, and are therefore not used for our experiments. This is because both UCHs and UEHs use histograms to represent edge weights at lower levels of the hierarchy, which are known to be inaccurate when number of observations is low [24].

To ensure reasonable query processing times, we contracted the vertices in the road network with degree ≤ 2 . This reduces the number of graph vertices and edges significantly, and speeds up the preprocessing and query times without losing accuracy.

Preprocessing Times: Figure 4 shows preprocessing times for all three query types. Non-Dominated Routes and Probabilistic Budget Routes can both be computed using the same underlying graph, so we construct only one UEH and UCH of each type for these query types. The preprocessing times for Mean-risk routes, however, are much larger than for other queries. This is because we must construct a *set* of EHs or CHs for a subset of the graph to answer mean-risk queries. For all hierarchies, we set $\sigma_T = 0.2$ and $\lambda = 100$.

We see that the untiered hierarchies take the longest to build, followed by the HD and KLD variants of UEH, and the HD and KLD variants of UCH. This is because of the expensive convolutions on histograms. The untiered CH and EH lack tier G which offers cheap convolutions, and incur a high cost for witness searches, increasing preprocessing time. UCH has lower preprocessing time than UEH for the same reasons as in deterministic routing—it offers a coarser hierarchy, preprocesses far fewer vertices, and avoids the costly Minimum Vertex Cover computation on each edge contraction.

Query times The query times are shown in Figure 5. Routes minimizing the mean-risk objective tend to be the fastest to compute. This is due to having a set of deterministic EHs and CHs for shortcuts in tier G. The very low cost of finding routes in a deterministic hierarchy outweighs having to run K shortest path queries, one for each EH or CH in K . For Probabilistic Budget routes, the query time increases as the budget increases, because the search can reach vertices farther from the source. However, for mean-risk routes, the correlation between c and query times is not well-defined, and can depend on the locations of source and target vertices, variance in edge weights in the vicinity, etc.

Next, we see that for all three query types, untiered hierarchies are the slowest, followed by Hellinger Distance variants of the UEH and UCH. The KLD variants tend to be the fastest. This is due

⁶<https://labs.mapbox.com/what-the-tile/>

⁷<https://www.mapbox.com/traffic-data>

Dataset		Contraction Hierarchies						Edge Hierarchies					
		Preprocessing (ms)			Query (μ s)			Preprocessing (ms)			Query (μ s)		
		RtKt	Ours	% Gain	RtKt	Ours	%Gain	Ref [22]	Ours	%Gain	Ref [22]	Ours	%Gain
Travel Time	New York	7183	7130	0.7%	15	23	-53.3%	427456	285471	33.2%	41	35	14.6%
	Bay Area	4329	4220	2.5%	10	15	-50.0%	267031	241691	9.5%	27	22	18.5%
	States of CA & NV	28065	27297	2.7%	16	29	-81.3%	1592972	1258829	21.0%	46	39	15.2%
	USA West	96388	90301	6.3%	22	51	-131.8%	4783188	3772306	21.1%	62	53	14.5%
	Los Angeles Area	6158	5543	10.0%	13	20	-53.8%	384892	318524	17.2%	33	27	18.2%
	Tile 0230123	3954	4523	14.4%	9	16	-77.8%	297123	279923	5.8%	26	20	23.1%
Distance	New York	11728	12948	-10.4%	31	45	-45.2%	750250	685268	8.7%	82	71	13.4%
	Bay Area	6574	6930	-5.4%	19	27	-42.1%	448575	329455	26.6%	52	44	15.4%
	States of CA & NV	46366	50537	-9.0%	39	71	-82.1%	3061041	268851	91.2%	117	102	12.8%
	USA West	156994	167487	-6.7%	59	108	-83.1%	9150456	7752139	15.3%	186	152	18.3%
	Los Angeles Area	8853	9412	-6.3%	26	44	-69.2%	661573	543212	17.9%	72	63	12.5%
	Tile 0230123	5423	8121	-49.8%	13	22	-69.2%	450032	392191	12.9%	43	32	25.6%

Table 2: Deterministic routing: Our CH and EH implementation uses an adjacency list representation for both speedup techniques, and performs better than the original EH implementation but is slower than RoutingKit. The performance gap between CH and EH techniques when using the same underlying graph representation is smaller than originally reported in [22]

Figure 4: Preprocessing times for the tiered and untiered uncertain EHs and CHs on the contracted Tile 0230123 road network.

to three reasons: first, HD-UEH and HD-UCH store the Hellinger Distance on each edge of the road graph, and on running queries, include it in the dominance criterion for Dijkstra’s labels. In other words, in KLD-UCH or UEH, a Dijkstra’s search label at vertex v , ℓ_v dominates ℓ'_v iff the route represented by ℓ_v always takes more time to traverse than one represented by ℓ'_v . For the HD variants presented, in addition to travel time dominance, ℓ_v must have a smaller Hellinger Distance as compared to ℓ'_v . This makes it harder for a label to dominate another, and fewer searches can be pruned, which results in slower query times.

Effect of approximation error on routes Since tiered hierarchies approximate edge weight histograms in tier G , they trade off some accuracy for better query processing times. Table 3 shows the percentage change in mean travel times for a 100 random stochastic routing queries running on untiered versus tiered hierarchies.

In Table 3, the travel time generally increases when approximate edge weights are used. This is because our approach to approximating histograms as Gaussians using the method of [11] maintains overall fidelity to the histogram, but may cause slight changes to means and variances (see Table 1, [11]). Regardless, the errors are well under 10% of the reference solution using untiered hierarchies, except for very short probabilistic budget routes ($b = 5$ minutes).

6 CONCLUSIONS AND FUTURE WORK

Shortest-path queries are much harder when edge weights are stochastic than when they are deterministic, since they must compute expensive convolutions, making them many orders of magnitude slower than deterministic queries. We have addressed this problem by presenting the novel approach of “tiering”, which uses different representations for the distributions at different tiers of a shortcut hierarchy, allowing the best representation at each tier.

(a) Non-Dominated and Probabilistic Budget routes with $b \in \{5, 15, 25\}$ minutes.

(b) Non-dominated and Probabilistic Budget routes with $b \in \{5, 15, 25\}$ minutes

(c) Mean-risk routes with $c \in \{0.5, 1.0, 1.5\}$

(d) Mean-risk routes with $c \in \{0.5, 1.0, 1.5\}$

Figure 5: Query times for all three query types, using the tiered and untiered uncertain CHs and EHs on the contracted road network for Tile 0230123.

Change in mean travel times (%)	Uncertain Contraction Hierarchies			Uncertain Edge Hierarchies		
	Untiered	KLD	HD	Untiered	KLD	HD
Non-Dominated Routes	0	+5.9%	+3.3%	0	+7.1%	+5.4%
Probabilistic Budget Routes						
$b = 5$ minutes	0	-13.6%	-8.6%	0	+4.6%	+4.7%
$b = 15$ minutes	0	+6.2%	+5.3%	0	+9.5%	+7.7%
$b = 25$ minutes	0	+8.1%	+6.7%	0	+5.9%	+5.9%

Table 3: Effect of approximation error on route travel times in UEH and UCH. Routes generally take slightly more time when edge weight approximations are used as compared to untiered hierarchies.

We have developed Uncertain Contraction Hierarchies (UCH) and Uncertain Edge Hierarchies (UEH), by applying tiering to conventional Contraction and Edge Hierarchies. We have used two tiers in our work in this paper, and shown how to construct these tiers using the KL divergence and the Hellinger distance measures.

We have studied these techniques in depth, and characterized their performance for three important types of stochastic shortest path queries, showing good speedups. We present extensive experiments using real-world instances from Mapbox Traffic Data.

Our work has shown that tiering is both practical and useful, and provides good speedups without a significant loss in accuracy. In the future, we propose to study the value of more tiering levels, and more detailed experiments with multiple convolution methods between histograms [36].

ACKNOWLEDGMENTS

The authors gratefully acknowledge access to Mapbox Traffic Data provided by the Mapbox Community team⁸. This work was supported in part by NSF grant IIS-1527984.

REFERENCES

- [1] Saad Aljubayrin, Bin Yang, Christian S Jensen, and Rui Zhang. 2016. Finding non-dominated paths in uncertain road networks. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Burlingame, California) (SIGSPATIAL '16, Article 15). Association for Computing Machinery, New York, NY, USA, 1–10.
- [2] Georgi Andonov and Bin Yang. 2018. Stochastic Shortest Path Finding in Path-Centric Uncertain Road Networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. 40–45.
- [3] Mohammad Asghari, Tobias Emrich, Ugur Demiryurek, and Cyrus Shahabi. 2015. Probabilistic Estimation of Link Travel Times in Dynamic Road Networks. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Seattle, Washington) (SIGSPATIAL '15). ACM, New York, NY, USA, 47:1–47:10.
- [4] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. 2016. Route Planning in Transportation Networks. In *Algorithm Engineering*. Springer, Cham, 19–80.
- [5] Gernot Veit Batz, Daniel Delling, Peter Sanders, and Christian Vetter. 2009. Time-dependent Contraction Hierarchies. In *Proceedings of the Meeting on Algorithm Engineering & Experiments* (New York, New York). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 97–105.
- [6] Gernot Veit Batz, Robert Geisberger, Sabine Neubauer, and Peter Sanders. 2010. Time-Dependent Contraction Hierarchies and Approximation. In *Experimental Algorithms*. Springer Berlin Heidelberg, 166–177.
- [7] Gernot Veit Batz, Robert Geisberger, Peter Sanders, and Christian Vetter. 2013. Minimum time-dependent travel times with contraction hierarchies. *J. Exp. Algorithmics* 18 (Dec. 2013), 1.1–1.43.
- [8] Reinhard Bauer, Tobias Columbus, Ignaz Rutter, and Dorothea Wagner. 2016. Search-space size in contraction hierarchies. *Theor. Comput. Sci.* 645 (Sept. 2016), 112–127.
- [9] Andrew C Berry. 1941. The accuracy of the Gaussian approximation to the sum of independent variates. *Trans. Amer. Math. Soc.* 49, 1 (Jan 1941), 122–122.
- [10] Johannes Blum and Sabine Storandt. 2020. Lower bounds and approximation algorithms for search space sizes in contraction hierarchies. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.
- [11] Lawrence D Brown and J T Gene Hwang. 1993. How to Approximate a Histogram by a Normal Density. *Am. Stat.* 47, 4 (Nov. 1993), 251–255.
- [12] Zhiguang Cao, Hongliang Guo, Jie Zhang, Dusit Niyato, and Ulrich Fastenrath. 2014. A data-driven method for stochastic shortest path problem. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. 1045–1052.
- [13] Lily Chai and Morgan Herlocker. 2019. Generating accurate speed estimations using aggregated telemetry data.
- [14] Anthony Chen and Zhaowang Ji. 2005. Path finding under uncertainty. *J. Adv. Transp.* 39, 1 (Sept. 2005), 19–37.
- [15] Bi Yu Chen, William H K Lam, Agchai Sumalee, Qingquan Li, and Mei Lam Tam. 2014. Reliable Shortest Path Problems in Stochastic Time-Dependent Networks. *J. Intell. Transp. Syst.* 18, 2 (April 2014), 177–189.
- [16] Jian Dai, Bin Yang, Chenjuan Guo, Christian S Jensen, and Jilin Hu. 2016. Path Cost Distribution Estimation Using Trajectory Data. *Proceedings VLDB Endowment* 10, 3 (Nov. 2016), 85–96.
- [17] Carl-Gustav Esseen. 1942. On the Liapunoff limit of error in the theory of probability. *Ark. Mat. Astron. Fys.* A28, 9 (1942), 1–19.
- [18] H Frank. 1969. Shortest Paths in Probabilistic Graphs. *Oper. Res.* 17, 4 (1969), 583–599.
- [19] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *Experimental Algorithms*. Springer, Berlin, Heidelberg, 319–333.
- [20] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. 2012. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transportation Science* 46, 3 (Aug. 2012), 388–404.
- [21] Ernst Hellinger. 1909. Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. *J. Reine Angew. Math.* 1909, 136 (July 1909), 210–271.
- [22] Demian Hesse and Peter Sanders. 2019. More Hierarchy in Route Planning Using Edge Hierarchies. In *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019) (OpenAccess Series in Informatics (OASIS))*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbrücken, Germany, Dagstuhl, Germany, 10:1–10:14.
- [23] Jilin Hu, Bin Yang, Chenjuan Guo, and Christian S Jensen. 2018. Risk-aware path selection with time-varying, uncertain travel costs: a time series approach. *VLDB J.* 27, 2 (April 2018), 179–200.
- [24] Tobias Skovgaard Jepsen, Christian S Jensen, and Thomas Dyhre Nielsen. 2021. UniTE – The Best of Both Worlds: Unifying Function-Fitting and Aggregation-Based Approaches to Travel Time and Travel Speed Estimation. (April 2021). arXiv:2104.13321 [cs.LG]
- [25] Moritz Kobitzsch, Samitha Samaranyake, and Dennis Schieferdecker. 2014. Pruning Techniques for the Stochastic on-time Arrival Problem – An Experimental Study. (July 2014). arXiv:1407.8295 [cs.DS]
- [26] Adrian Kosowski and Laurent Viennot. 2017. Beyond Highway Dimension: Small Distance Labels Using Tree Skeletons. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (Barcelona, Spain) (SODA '17). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1462–1478.
- [27] Solomon Kullback and Richard A Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (March 1951), 79–86.
- [28] Sejoon Lim, Christian Sommer, Evdokia Nikolova, and Daniela Rus. 2013. Practical route planning under delay uncertainty: Stochastic shortest path queries. In *Robotics: Science and Systems*, Vol. 8. books.google.com, 249–256.
- [29] Elise Deborah Miller-Hooks. 1997. *Optimal routing in time-varying, stochastic networks: Algorithms and implementations*. Ph.D. Dissertation. The University of Texas at Austin.
- [30] Nikola Milosavljević. 2012. On optimal preprocessing for contraction hierarchies. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science* (Redondo Beach, California) (IWCTS '12). Association for Computing Machinery, New York, NY, USA, 33–38.
- [31] Mehrdad Niknami and Samitha Samaranyake. 2016. Tractable Pathfinding for the Stochastic On-Time Arrival Problem. In *Experimental Algorithms*. Springer International Publishing, 231–245.
- [32] Evdokia Nikolova. 2010. Approximation Algorithms for Reliable Stochastic Combinatorial Optimization. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer Berlin Heidelberg, 338–351.
- [33] Evdokia Nikolova, M Brand, and David R Karger. 2006. Optimal Route Planning under Uncertainty. *ICAPS* (2006).
- [34] Evdokia Nikolova, Jonathan A Kelner, Matthew Brand, and Michael Mitzenmacher. 2006. Stochastic Shortest Paths Via Quasi-convex Maximization. In *Algorithms – ESA 2006*. Springer Berlin Heidelberg, 552–563.
- [35] Axel Parmentier and Frédéric Meunier. 2014. Stochastic Shortest Paths and Risk Measures. (Aug. 2014). arXiv:1408.0272 [cs.DS]
- [36] Simon Aagaard Pedersen, Bin Yang, and Christian S Jensen. 2019. Fast stochastic routing under time-varying uncertainty. *VLDB J.* (Oct. 2019).
- [37] Mark Semenovich Pinsker. 1964. *Information and Information Stability of Random Variables and Processes*. Holden-Day.
- [38] Guillaume Sabran, Samitha Samaranyake, and Alexandre Bayen. 2013. Pre-computation techniques for the stochastic on-time arrival problem. In *2014 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*. Society for Industrial and Applied Mathematics, 138–146.
- [39] Irina G Shevtsova. 2010. An improvement of convergence rate estimates in the Lyapunov theorem. *Dokl. Math.* 82, 3 (Dec. 2010), 862–864.
- [40] Christian Sommer. 2014. Shortest-path queries in static networks. *ACM Computing Surveys (CSUR)* 46, 4 (April 2014), 45.
- [41] Mikkel Thorup. 2004. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM* 51, 6 (Nov. 2004), 993–1024.
- [42] Bin Yang, Jian Dai, Chenjuan Guo, Christian S Jensen, and Jilin Hu. 2018. PACE: a PATH-Centric paradigm for stochastic path finding. *VLDB J.* 27, 2 (April 2018), 153–178.
- [43] Bin Yang, Chenjuan Guo, Christian S Jensen, Manohar Kaul, and Shuo Shang. 2014. Stochastic skyline route planning under time-varying uncertainty. In *2014 IEEE 30th International Conference on Data Engineering*. 136–147.
- [44] Jie Zheng, Ling Wang, Shengyao Wang, Yile Liang, and Jize Pan. 2021. Solving two-stage stochastic route-planning problem in milliseconds via end-to-end deep learning. *Complex & Intelligent Systems* 7, 3 (June 2021), 1207–1222.
- [45] Athanasios K Ziliaskopoulos and Hani S Mahmassani. 1993. A Time-Dependent Shortest Path Algorithm for Real-Time Intelligent Vehicle/Highway System. *Transportation Research Record Journal of the Transportation Research Board* 1408 (Jan. 1993), 94–100.

⁸<https://mapbox.com/community/>