Highway Dimension and Provably Efficient Shortest Path Algorithms

ITTAI ABRAHAM and DANIEL DELLING, Microsoft Research Silicon Valley
AMOS FIAT, Tel Aviv University
ANDREW V. GOLDBERG and RENATO F. WERNECK, Microsoft Research Silicon Valley

Computing driving directions has motivated many shortest path algorithms based on preprocessing. Given a graph, the preprocessing stage computes a modest amount of auxiliary data, which is then used to speed up online queries. In practice, the best algorithms have storage overhead comparable to the graph size and answer queries very fast, while examining a small fraction of the graph. In this article, we complement the experimental evidence with the first rigorous proofs of efficiency for some of the speedup techniques developed over the past decade or variations thereof. We define highway dimension, which strengthens the notion of doubling dimension. Under the assumption that the highway dimension is low (at most polylogarithmic in the graph size), we show that, for some algorithms or their variants, preprocessing can be implemented in polynomial time, the resulting auxiliary data increases the storage requirements by a polylogarithmic factor, and queries run in polylogarithmic time. This gives a unified explanation for the performance of several seemingly different approaches. Our best bounds are based on a result that may be of independent interest: we show that unique shortest paths induce set systems of low VC-dimension, which makes them combinatorially simple.

Categories and Subject Descriptors: E.1 [**Data Structures**]: Graphs and Networks; G.2.2 [**Graph Theory**]: Graph Algorithms

General Terms: Algorithms, Shortest Paths, Set Systems

Additional Key Words and Phrases: Shortest paths, VC-dimension, algorithm

ACM Reference Format:
Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. 2016. Highway dimension and provably efficient shortest path algorithms. J. ACM 63, 5, Article 41 (December 2016), 26 pages.
DOI: http://dx.doi.org/10.1145/2985473

1. INTRODUCTION

Gaius Octavius Thurinus (aka Augustus Caesar) was put in charge of Roman roads (*viae*) in 20 BC, and he made the roads easier to navigate. Formally, all *viae* began at the Temple of Saturn in Rome, with milestones giving distances to the Forum in Rome.

Most of this work was done while the first, second, fourth, and fifth authors were at Microsoft Research Silicon Valley. Part of this work was done while the third author was visiting Microsoft Research Silicon Valley.
This article includes the results from two conference papers [Abraham et al. 2010, 2011a] as well as some additional contributions, including an alternative definition of highway dimension that we show to be a strengthening of the definition of doubling dimension (Section 9) and also allows us to analyze the transit node algorithm (Section 5.2).
Authors' addresses: I. Abraham, VMWare Research, 3425 Hillview Ave, Palo Alto, CA 94304, USA; email: ittaia@cs.huji.ac.il; D. Delling, 655 Boise Ct, Sunnyvale, CA 94087, USA; email: daniel.delling@gmail.com; A. Fiat, School of Computer Science, Tel Aviv University, Tel Aviv, Israel; email: fiat@tau.ac.il; A. V. Goldberg and R. F. Werneck, Amazon.com, Inc., 1900 University Avenue, E. Palo Alto, CA 94303, USA; emails: avg@alum.mit.edu, rwerneck@gmail.com.

DOI: http://dx.doi.org/10.1145/2985473

According to the Cosmographia Julius Honorius, Consuls Julius Caesar, and Mark Anthony sent out four scholars to map the world: Nicodemus, Didymus, Thodotus, and Polycletes (see, e.g., Brock [2012]). This task took them 32 years, 1 month, and 20 days. The Roman map, however, is tiny by modern standards, and one can find good routes in it by hand.

The volume of available data on road networks is much higher today, and computing shortest paths is not trivial. Dijkstra's algorithm [Dijkstra 1959] allows us to answer point-to-point shortest path queries on any network with nonnegative arc lengths in essentially linear time, both in theory and in practice (see, e.g., Fredman and Tarjan [1987] and Goldberg [2008]). Unfortunately, this is too slow for common applications on large road networks, such as those of North America or Europe, on which Dijkstra's algorithm takes a few seconds on a modern server. For example, a system for online driving directions must support thousands of queries per second. Alternatively, an application running on a GPS device with limited RAM stores graph data in external memory, resulting in a slowdown of several orders of magnitude.

Motivated by computing driving directions, several algorithms use a two-stage framework. In a preprocessing stage, they compute auxiliary data, such as additional edges (shortcuts) and labels or values associated with vertices or edges. The auxiliary data is then used to accelerate point-to-point (s–t) shortest path queries, typically by pruning or directing Dijkstra's algorithm. Algorithms within this framework are based on a wide variety of ideas, such as arc flags [Lauther 2009; Hilger et al. 2009; Bauer and Delling 2009], A^* search with landmarks [Goldberg and Harrelson 2005], separator-based multilevel graphs [Holzer et al. 2008; Delling et al. 2011a; Delling and Werneck 2013], highway hierarchies [Sanders and Schultes 2012], reach [Gutman 2004; Goldberg et al. 2009], transit nodes [Bast et al. 2007], contraction hierarchies [Geisberger et al. 2012], and hub labeling [Abraham et al. 2011b, 2012b]. In experiments using real-world data, these algorithms achieve amazing speedups over plain Dijkstra: visiting a few hundred vertices is enough to answer a random query on road networks with tens of millions of intersections. These methods are exact: they are guaranteed to find the actual shortest path, not an approximation. Moreover, preprocessing is practical and produces a modest amount of auxiliary data.

Unfortunately, these excellent practical results have been purely experimental, with no provably good time guarantees. In fact, it is not hard to construct inputs for which these algorithms fail to achieve any meaningful speedup. No analysis of the algorithms on any nontrivial graph classes was known. Furthermore, there was no theoretical understanding of which properties of road networks make the algorithms work well; previous work in this direction has been mostly experimental [Bauer et al. 2011]. Eppstein and Goodrich [2008] take an orthogonal approach and model road networks as graphs with small separators.

The lack of theoretical understanding of the practical hierarchical shortest path algorithms suggests the following natural questions, which are the subject of this article. Can one prove sublinear query bounds for these algorithms on a nontrivial class of networks? For what graphs does the two-stage framework lead to algorithms with provably good performance? Specifically, what properties of road networks imply provably good performance for the (*de facto* excellent) preceding algorithms? Finally, is there a plausible explanation as to why real road networks actually satisfy such conditions?

Note that previously studied graph properties such as planarity and small doubling dimension do not explain the observed algorithm performance. In fact, Eppstein and Goodrich [2008] argue that road networks are nonplanar. Furthermore, the recent algorithms perform worse on two-dimensional grids with uniformly random edge weights than they do on road networks [Goldberg et al. 2009; Bauer et al. 2010a, 2011].

To explain these empirical observations, we introduce the notion of *highway dimension*. Our motivation for the definition of highway dimension is the experiments performed by Bast et al. [2009], who exploited a very intuitive observation: when driving on a shortest path from a compact region of a road network to points that are "far away," one must pass through one of a very small number of *access nodes*. Intuitively, a graph has highway dimension h if, for every scale $r > 0$ and for every vertex v, there is a set of at most h vertices that hits all shortest paths within distance $O(r)$ from v that are longer than r. We show that our definition of highway dimension strengthens the notion of doubling dimension [Gupta et al. 2003]. In particular, a graph with highway dimension h has doubling dimension $\log h$ (all logarithms in this article have base 2); the doubling dimension of (the shortest path metric of) a graph is defined in Section 9.

We show that low highway dimension gives provable guarantees of efficiency for the *Reach (RE)*, *Contraction Hierarchies (CH)*, and *Transit Nodes (TN)* algorithms, which were originally developed with road networks in mind. We prove even better bounds for the *Hub Labeling (HL)* algorithm [Cohen et al. 2003; Gavoille et al. 2004], which was initially proposed (in a theoretical setting) for general graphs, but had never been tested on large road networks. However, recent work (motivated by a conference version of this article [Abraham et al. 2010]) shows that HL is indeed extremely fast on such graphs [Abraham et al. 2011b, 2012b].

More precisely, given a connected, simple graph with n vertices, m edges, highway dimension h, and positive edge weights. Define D to be the diameter of the network obtained from the input network by dividing all edge length by the minimum one (so that the minimum edge length becomes one).

—Preprocessing takes time polynomial in m (and n), h, and $\log D$.
—The auxiliary data it produces has size linear in m and polynomial in $\log n$, h, and $\log D$.
—An s–t query returns the distance from s to t in time polynomial in h and $\log D$.

Our preprocessing algorithm uses a new result on the relationship between shortest paths and VC-dimension, which shows that collections of shortest paths are simple in a combinatorial sense. Kleinberg [2003] showed that VC-dimension can be useful in the analysis of graph algorithms. We extend this observation by showing that it can be used to design graph algorithms. To facilitate this use, the formal definition of highway dimension is based on set systems.

One can view modeling road networks by graphs with low highway dimension as being somewhat analogous to small-world models for social networks [Milgram 1967; Kleinberg 2000]. Although real social networks do not look exactly like small-world graphs, the latter give insight into and allow the asymptotic analysis of various routing algorithms. Similarly, while real road networks may have local features that violate the small highway dimension assumption, the notion of highway dimension gives insight into and allows rigorous analysis of algorithms that work very well in practice.

As already mentioned, RE, CH, and TN have been developed to be efficient on road networks. TN is the fastest among these algorithms in practice, winning the DIMACS Challenge on shortest paths [Demetrescu et al. 2009]. Our theoretical analysis for TN indeed shows that it should be asymptotically faster than RE and CH for long-range queries (and equivalent for local ones). Our theoretical bound is even better for HL, which motivated follow-up work [Abraham et al. 2011b, 2012b] on a practical implementation of HL that does outperform TN on road networks, with the fastest known point-to-point shortest path queries.

Another contribution of this article is a generative model for road networks based on three assumptions: (i) the road network is embedded in a metric space of small doubling dimension (such as the Earth's surface) with a corresponding distance metric; (ii) the speed of vehicles is higher on longer road segments (i.e., highways are faster than local roads); and (iii) roads are built in an incremental manner over time. We show that the model produces networks with constant highway dimension.

This article is organized as follows. Section 2 establishes some basic notation, definitions, and background. Section 3 formally introduces the notion of highway dimension. Section 4 introduces the related concept of sparse hitting sets (and multiscale variants). Section 5 uses multiscale sparse hitting sets to obtain provably efficient implementations of HL and TN, while Section 6 extends the multiscale hitting sets approach to deal with the shortcut-based algorithms CH and RE. Section 7 shows that shortest paths have a remarkably simple combinatorial structure: they induce set systems of VC-dimension two. This result is used in Section 8 to show how multiscale hitting sets can be approximated efficiently, leading to polynomial-time preprocessing for shortest path algorithms. Section 9 studies the relationship between highway dimension and doubling dimension. Section 10 presents our generative model. Section 11 discusses the relationship to continuous graphs, which gives an intuition for our definitions. Section 12 has final remarks, including an overview of follow-up work enabled by the notion of highway dimension, developed since the conference versions of our results [Abraham et al. 2010, 2011a] appeared.

2. PRELIMINARIES

A *network* (G, ℓ) consists of a graph $G = (V, E)$ and a length function $\ell : E \to \mathbb{R}_+$. The preprocessing phase of a two-stage algorithm takes as input the network (G, ℓ), and outputs additional data that can be used in the query phase. In addition, the query takes as input a *source* vertex s and a *target* vertex t and returns the distance $\mathrm{dist}(s, t)$ from s to t.

In this article, we focus on bounds for implementing *distance oracles*, that is, we are interested in finding the shortest path *length*, not the path itself. The algorithms we analyze can be extended to find the corresponding shortest paths in time linear or near-linear in the size (number of vertices) of the path. Although this bound is optimal if we require explicit paths in the output, it may be exponentially worse than a distance oracle query.

Unless mentioned otherwise, we assume that the graph is undirected and connected, and that the length function is strictly positive (zero-length edges can be contracted). We also assume that there are no parallel edges or self-loops. To simplify notation, we scale edge lengths by dividing them by the smallest arc length, so that the minimum edge length is one. In the rest of the article, we assume that the input lengths are scaled this way. Note that this is not necessary if the input lengths are integral, as is the case in most practical implementations.

A path is a sequence of vertices $P = (v_1, \ldots, v_k)$ such that there is an edge $(v_i, v_{i+1}) \in E$ for every $1 \leq i < k$. We allow trivial paths with a single vertex. The length of a path is defined by $\ell(P) = \sum_{i=1}^{k-1} \ell(v_i, v_{i+1})$. In a slight abuse of notation, we will sometimes identify a path $(v_1, \ldots, v_k)$, with its set of vertices $\{v_1, \ldots, v_k\}$.

For a pair of vertices s, t, $\mathrm{dist}(s, t)$ denotes the length of a shortest path from s to t. We also assume that the shortest path $P(s, t)$ between any two vertices s and t is unique; since we consider undirected graphs, $P(s, t) = P(t, s)$. This is a common assumption that can be made without loss of generality, as one can perturb the input to ensure uniqueness with high probability. Finally, we assume that every edge is a shortest path between its endpoints; we can delete edges that do not satisfy this condition.

The *diameter* D of the network is defined by $D = \max_{u,v \in V} \mathrm{dist}(u, v)$. Recall that we scale edge lengths by the minimum one, so the diameter is scaled as well. We use the diameter in our complexity bounds. As the lengths are real valued, without scaling one can divide the lengths by the same constant and obtain an equivalent problem with an arbitrary small value of D.

Dijkstra's algorithm [1959] is an efficient implementation of the scanning method for graphs with nonnegative edge lengths (see, e.g., Tarjan [1983]). For every vertex v, it maintains the length $d(v)$ of the shortest path from s to v found so far, as well as the predecessor $p(v)$ of v on the path. Initially $d(s) = 0$, $d(v) = \infty$ for all other vertices v, and $p(v) = null$ for all v.

Dijkstra's algorithm maintains a priority queue of unscanned vertices with finite d values. At each step, the algorithm extracts the minimum valued vertex v from the queue and scans it. To scan v, the algorithm looks at all edges $(v, w) \in E$ and, if $d(v) + \ell(v, w) < d(w)$, sets $d(w) = d(v) + \ell(v, w)$ and $p(w) = v$. The algorithm terminates when the target t is extracted, without scanning it.

The bidirectional version of Dijkstra's algorithm is similar, but runs a forward search from s and a backward search from t. When an edge (v, w) is scanned by the forward search and w has already been scanned by the backward search, the concatenation of path s–v, edge (v, w), and path w–t is a new path P from s to t (the same holds in the backward search). The algorithm maintains the shortest such path $\overline{P}$ found during the execution. It terminates when one of the searches extracts a vertex that has been scanned by the other, at which point the path $\overline{P}$ is optimal [Pohl 1971].

3. HIGHWAY DIMENSION

We now formally define the notion of highway dimension using the concepts of r-significant and (r, d)-close paths. While these concepts may seem somewhat contrived, they are natural in the context of continuous graphs, as formalized in Section 11. Intuitively, an r-significant path is a discretization of a continuous path of length greater than r. Similarly, a path (r, d)-close to v is a discretization of a continuous path of length greater than r whose distance to v is at most d.

Formally, we define an r-significant path as follows.

Definition 3.1. A shortest path P is r-significant if it has an r-witness path (see Definition 3.2).

Definition 3.2. Given a shortest path $P = (v_1, \ldots, v_k)$ and $r > 0$, a shortest path P' is an r-witness for P if and only if $\ell(P') > r$ and one of the following conditions holds:

(1) $P' = P$; or
(2) $P' = (v_0, v_1, \ldots, v_k)$; or
(3) $P' = (v_1, \ldots, v_k, v_{k+1})$; or
(4) $P' = (v_0, v_1, \ldots, v_k, v_{k+1})$.

In other words, P can be extended by at most one vertex at each end to a shortest path of length greater than r. Note that, if $(v, w) \in E$ and $\ell(v, w) > r$, then the trivial paths (v) and (w) are r-significant, since (v, w) is the shortest path from v to w. Also note that if $(v_1, v_2, \ldots, v_k)$ is r-significant, then so is $(v_k, v_{k-1}, \ldots, v_2, v_1)$ (s Figure 1).

Let $\mathcal{P}_r$ denote the set of all r-significant paths. For example, in the graph of Figure 1, for all $0 < r < 3$ the set P_r consists of all shortest paths (including singleton vertices). For $3 \le r < 5$, the set $P_r = P_{3-\epsilon} \backslash \{(w)\}$. For $5 \le r < 6$, $P_r = P_{5-\epsilon} \backslash \{(x)\}$. The set of all 19-significant paths is $\mathcal{P}_{19} = \{(p, q, u, v), (w, p, q, u, v), (p, q, u, v, z), (w, p, q, u, v, z)\}$ (and the same paths in the reverse direction). There is no other 19-significant path. The path (w, p, q, u, v, z) is the longest shortest path and has length 19; for any $r > 19$, $\mathcal{P}_r = \emptyset$.

Given a vertex v and a path P, we define the distance from v to P as $\text{dist}(v, P) = \min_{w \in P} \text{dist}(v, w)$.

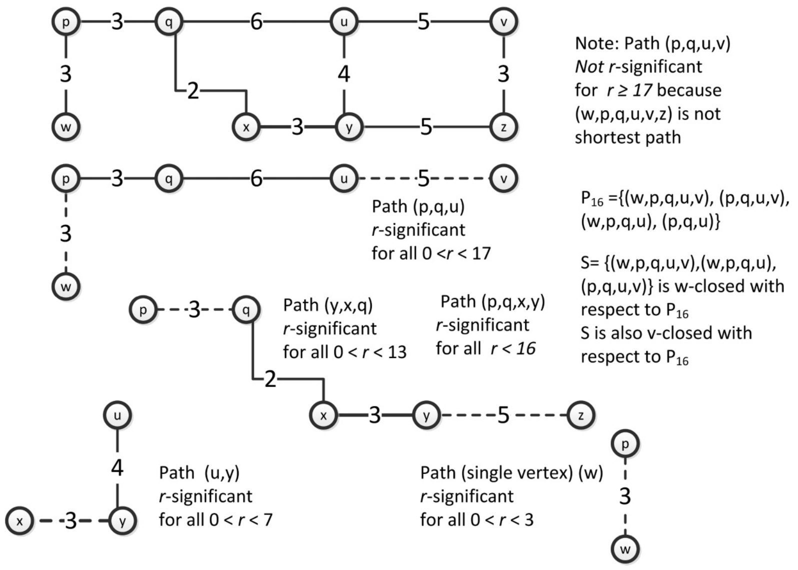

Fig. 1. Illustrating the definition of r-significant and (r, d)-close paths.

Definition 3.3. A shortest path P is (r, d)-close to a vertex v if P is r-significant with an r-witness path P' such that $\text{dist}(v, P') \leq d$.

Note that if P is (r, d)-close to v, then for $0 < r' \leq r$, $0 \leq d \leq d'$, it follows that P is also (r', d')-close to v.

To illustrate this definition, consider the graph in Figure 1. For $P = (u, y)$ we have that P is $(6, 3)$-close to q, $(6, 0)$-close to x, and $(6, 5)$-close to p. To see this, consider the shortest path $P' = (u, y, x)$ that shows that (u, y) is 6-significant. The distances from P' to q, x, and u are 3, 0, and 5, respectively.

Given $r \geq 0$ and $v \in V$, we define the *ball* of radius r centered at v, $B_r(v)$ to be the set of all vertices w such that $\text{dist}(v, w) \leq r$. For example, in Figure 1, $B_4(y) = \{u, x\}$.

Definition 3.4. A network (G, ℓ) has *Highway Dimension (HD) h* if h is the smallest integer such that, for all $r > 0$ and all $v \in V$, there exists a set $H \subseteq V$ (that depends on v and r) such that $|H| \leq h$ and $H \cap P \neq \emptyset$ for all paths P that are $(r, 2r)$-close to v.

As discussed in Section 12, this definition is somewhat different from the one introduced in the conference version of our work [Abraham et al. 2010, 2011a].

Let the *r-neighborhood* of v, denoted by $S_r(v)$, be the set of all paths P that are $(r, 2r)$-close to v. Given a set of paths $\mathcal{P}$, we say that $H \subseteq V$ is a *hitting set* for $\mathcal{P}$ if every path in $\mathcal{P}$ contains a vertex in H. Figure 2 gives an example. Definition 3.4 states that the highway dimension of a network (G, ℓ) is at most h if, for any $r > 0$ and any $v \in V$, there exists a hitting set H for $S_r(v)$ with $|H| \leq h$.

The highway dimension of a graph bounds its degree:

Lemma 3.5. *If (G, ℓ) has highway dimension h, then the degree of any vertex in G is at most h.*

Proof. Consider a vertex v and take any $0 < r < 1$ (e.g., $r = 1/2$). Since we assume that every edge is a shortest path and edge lengths are at least one, for every neighbor w of v the path $P' = (w, v)$ is an r-witness for the trivial path $P = (w)$. The distance from $P' = (w, v)$ to v is zero, hence $P = (w)$ is (r, d)-close to v for all $d \geq 0$; in particular,

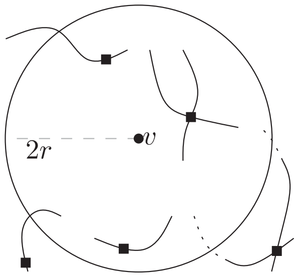

Fig. 2. Example for $S_r(v)$. All r-significant paths are hit.

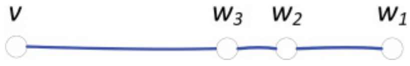

Fig. 3. v-closed sets.

it is $(r, 2r)$-close to v. This implies that $(w) \in S_r(v)$, and therefore any hitting set of $S_r(v)$ must contain w. Thus, the cardinality of any hitting set for $S_r(v)$ is at least the degree of v. It follows that the degree of v is at most h. □

4. SPARSE HITTING SETS

A notion related to highway dimension is that of a *Sparse Shortest Path Hitting Set (SPHS)*, which we need to prove our bounds for HL, TN, CH, and RE in Sections 5 and 6.

Definition 4.1. For $r > 0$, an (h, r)-SPHS is a hitting set $C \subseteq V$ for $\mathcal{P}_r$ such that $\forall v \in V,\ |B_{2r}(v) \cap C| \le h$.

SPHS and HD are closely related.

THEOREM 4.2. *If the highway dimension of (G, ℓ) is h, then for any $r > 0$, a minimum hitting set for $\mathcal{P}_r$ is an (h, r)-SPHS.*

PROOF. Let H be a minimum hitting set for $\mathcal{P}_r$ and suppose for contradiction that for some u, the set $U = H \cap B_{2r}(u)$ is such that $|U| > h$. Note that U hits only those paths in $\mathcal{P}_r$ that belong to $\mathcal{S}_r(u)$. By the definition of h, there is a hitting set U' for $\mathcal{S}_r(u)$ with $|U'| \le h$. By the definition of $\mathcal{S}_r(u)$, U' hits all shortest paths in $\mathcal{P}_r$ hit by U. Therefore, $H' = (H \backslash U) \cup U'$ is smaller than H and hits all shortest paths in $\mathcal{P}_r$, contradicting the optimality of H. □

A *multiscale SPHS* of (G, ℓ) is a collection of sets C_i for $0 \le i \le \lceil \log D \rceil$, where each C_i is a $(h, 2^{i-1})$-SPHS. In particular, note that $C_0 = V$, since every vertex is an (1/2)-significant path. The notion of multiscale SPHS is a basic building block of our preprocessing algorithms.

An important property of multiscale SPHS is that the sequence of set sizes ($|C_i|$) does not have big gaps. To prove this, we need the concept of *v-closed sets*. For a fixed r, we say that a set $S \subseteq \mathcal{P}_r$ is *v-closed* if either $S = \emptyset$ or (i) v is an endpoint of some path in S and (ii) for all paths $P \in S$ with endpoint v, every subpath $P' \in \mathcal{P}_r$ of P that has v as an endpoint also belongs to S. In particular, note that $S_r(v)$ is v-closed.

In Figure 3, let $P_0 = (v)$, $P_1 = (v, w_3, w_2, w_1)$, $P_2 = (v, w_3, w_2)$, and $P_3 = (v, w_3)$. Examples of v-closed sets are $\{P_0\}$, $\{P_0, P_3\}$, and $\{P_0, P_1, P_2, P_3\}$. The set $\{P_0, P_2\}$ is not v-closed because it does not contain P_3, which is a subpath of P_2.

LEMMA 4.3. *Let C be a hitting set for a v-closed set $S \subseteq \mathcal{P}_r$. Then, for any $P \in S$ that has v as an endpoint, there is a vertex $w \in C \cap P$ such that* $\text{dist}(v, w) \le r$.

PROOF. The lemma is obvious if $\ell(P) \le r$ because for any $w \in P$, we have $\text{dist}(v, w) \le r$. Assume that $\ell(P) > r$. There is an edge (x, y) on P such that $\text{dist}(v, x) \le r$ and $\text{dist}(v, y) > r$. Let P' be the subpath of P between v and x; then $\ell(P') \le r$. Since S is v-closed, $P' \in S$, so P' is hit by a vertex $z \in C$. Since z is on P', we have $\text{dist}(v, z) \le r$, completing the proof. □

With this result, we can prove that the sequence of set sizes of a multiscale SPHS does indeed have small gaps.

LEMMA 4.4. *If the highway dimension of (G, ℓ) is h, then there exists a multiscale SPHS such that $|C_{\lceil \log D \rceil}| \le h$ and $|C_i| \le h|C_{i+1}|$ for $0 \le i < \lceil \log D \rceil$.*

PROOF. To prove the first claim, pick a vertex v. Because the graph is connected, all vertices are within distance D from v. Therefore, $S_{2^{\lceil \log D \rceil - 1}}(v)$ contains all $(2^{\lceil \log D \rceil - 1})$-significant paths. The hitting set H for $S_{2^{\lceil \log D \rceil - 1}}(v)$ hits all these paths and $|H| \le h$. Thus, $C_{\lceil \log D \rceil} = H$ is the desired set.

For $i < \lceil \log D \rceil$, we build C_i from C_{i+1} in decreasing order of i, as follows. Let H be the union of the hitting sets for $S_{2^{i-1}}(v)$ for all $v \in C_{i+1}$; obviously $|H| \le h|C_{i+1}|$. By Lemma 4.3, every vertex $w \in V$ is at distance at most 2^i from some vertex in C_{i+1}, and therefore H is a hitting set for $\mathcal{P}_{2^{i-1}}$.

Let $\hat{h}$ be the minimal value of $x \le n$ such that H is an $(x, 2^{i-1})$-SPHS. If $\hat{h} \le h$, we are done. Otherwise, if $h < \hat{h}$, then there must exist some $u \in V$ such that $U = H \cap B_{2r}(u)$ has $|U| = \hat{h} > h$. Because the highway dimension of (G, ℓ) is h, there is a hitting set U' for the set $S_r(u)$, where $|U'| \le h$. As in the proof of Theorem 4.2, we replace U with U'. The new set H' is also a hitting set for $\mathcal{P}_{2^{i-1}}$, and has fewer vertices. So after at most n iterations, the resulting set is a hitting set for $P_{2^{i-1}}$ and is $(h, 2^{i-1})$ sparse. □

Although Theorem 4.2 guarantees the existence of good hitting sets, no polynomial-time algorithm is known for computing the minimum hitting set H used in the proof. As a result, no polynomial-time algorithm is known for computing an optimum multiscale SPHS, which are important building blocks of the preprocessing phase of the algorithms we analyze in Sections 5 and 6. Note that in these sections, we analyze the query times assuming optimum (exponential-time) preprocessing.

Section 8 discusses polynomial-time approximation algorithms for SPHS computation, which enable polynomial-time preprocessing with a slight degradation of the query bounds. Although still impractical for large road networks, these polynomial-time preprocessing routines provide some justification for practical variants of preprocessing. These variants run in polynomial time and produce solutions that have no theoretical quality guarantee, but work well on real-world problems.

5. ANALYSIS FOR HUB LABELS AND TRANSIT NODE ROUTING

This section analyzes the HL and TN routing algorithms. In both cases, we assume that we have computed a multiscale SPHS as the initial step of the preprocessing stage.

5.1. Hub Labels

In this section, we study the HL algorithm in the context of undirected graphs. During preprocessing, a labeling algorithm [Peleg 2000] computes a *label* for each vertex of the graph such that $\text{dist}(s, t)$ can be computed from the labels of s and t only, without looking at the graph. The *HL* algorithm [Gavoille et al. 2004] is a special kind of labeling algorithm. For each vertex $v \in V$, HL builds a label $L(v)$, which consists of a set of vertices (the *hubs* of v), together with the distances $\text{dist}(v, w)$ for each $w \in L(v)$.

The labels obey the *cover property*: for any two vertices s and t, $L(s) \cap L(t)$ contains at least one vertex on the shortest s–t path. For an s–t query, among all vertices $w \in L(s) \cap L(t)$ we pick the one minimizing $\text{dist}(s, w) + \text{dist}(w, t)$ and return this sum. If the entries in each label are sorted by hub ID, this can be done with a coordinated sweep over the two labels, as in mergesort.

We say that the *label size of* v, $|L(v)|$, is the number of hubs in $L(v)$. The runtime of an s–t query is $O(|L(s)| + |L(t)|)$, or $O(M)$ if M is the maximum label size. The *labeling* $\mathcal{L}$ is the set of all labels; we denote its *size* by $|\mathcal{L}| = \sum_v |L(v)|$.

Although hub labels can be quite large in general, they are provably small for some graphs [Gavoille et al. 2004]. Computing labelings with the smallest total size or minimizing the size of the biggest label may be difficult, however. Cohen et al. [2003] give a polynomial-time algorithm to approximate the smallest labeling size within a factor of $O(\log n)$ (see also Krauthgamer and Roughgarden [2011]). In addition, Babenko et al. [2013] give an $O(\log n)$ approximation algorithm for the maximum label size and discuss simultaneous approximation of the total and maximum label size.

Next, we prove the following result:

THEOREM 5.1. *If (G, ℓ) has highway dimension h, then it has a hub labeling with maximum label size $O(h \log D)$.*

PROOF. Consider a multiscale SPHS $\{C_i : i = 0, \ldots, \lceil \log D \rceil\}$, with C_i as in Theorem 4.2. We define the label of a vertex v by

$$L(v) = \{v\} \cup \bigcup_{i=0}^{\lceil \log D \rceil} (C_i \cap B_{2^i}(v)).$$

By the definition of SPHS, $|L(v)| = O(h \log D)$. Next, we show that L has the cover property. Consider two vertices s and t. Let P be the shortest path from s to t and consider the smallest i such that $\ell(P) \leq 2^i$. Then, $P \in \mathcal{P}_{2^{i-1}}$ and there exists a vertex $x \in P \cap C_i$. By the definition of P, $x \in B_{2^i}(s)$ and $x \in B_{2^i}(t)$, so x must be in $L(s) \cap L(t)$. □

COROLLARY 5.2. *HL queries take $O(h \log D)$ time.*

The actual shortest path can be extracted in time linear in the number of edges on the path. This requires some changes in the preprocessing to produce an additional data structure [Abraham et al. 2012a].

5.2. Transit Nodes

The TN algorithm [Bast et al. 2007, 2009; Sanders and Schultes 2009] motivated our definition of highway dimension. TN is based on the observation that anyone driving (on an optimal path) from a small region to a faraway destination must pass through one of a small number of *access nodes*. The union of all access nodes constitutes the set of *transit nodes*. The preprocessing algorithm for TN finds a set of transit nodes that is not too large and locally sparse, that is, any vertex v has a small set $A(v)$ of access nodes contained in the set of transit nodes. For each vertex v, the algorithm outputs $A(v)$ and $\text{dist}(v, w)$ for each $w \in A(v)$. Finally, it computes and outputs a table of distances between all pairs of transit nodes; the size of the table is quadratic in the number of transit nodes.

The query algorithm looks at all three-hop paths of the form s–s'–t'–t, where $s' \in A(s)$ and $t' \in A(t)$. If the s–t distance is large enough (*global query*), then the shortest such path is the shortest path from s to t. As we precomputed distances between vertices and their access nodes and between all pairs of transit nodes, the global query time is $O(|A(s)| \cdot |A(t)|)$. To distinguish between global and *local* queries, the algorithm uses a fast *locality filter* that estimates how close s and t are. For local queries, an alternative algorithm is used to find the shortest path: in particular, Dijkstra-based algorithms with pruning, such as CH, are relatively fast when s and t are close. Optionally, additional levels of transit nodes can be used to handle medium-range queries [Sanders and Schultes 2009].

We will describe CH, another algorithm for the shortest path problem, in Section 6.1. For the discussion in this section, we only use the fact that CH orders vertices by importance during preprocessing.

The most efficient variants of TN [Geisberger et al. 2012; Arz et al. 2013] use CH in two ways: to choose transit nodes and to run local queries. For the former, if τ is the desired number of transit nodes, the τ most important vertices in the CH ordering are chosen as transit nodes. In practice, global queries are remarkably fast, much more so than the CH-based local queries.

In this section, we propose *TN-MS*, an analyzable implementation of TN based on multiscale SPHS, and derive sublinear time bounds for it. The implementation is similar in spirit to those of Geisberger et al. [2012] and Arz et al. [2013]. First, we pick a bound τ on the number of transit nodes. With a bigger value of τ, more queries are global. Choosing $\tau = O(\sqrt{m})$ guarantees that the additional memory requirements are bounded by the input graph size. Next, we compute a multiscale SPHS with elements C_i as in Section 4. Finally, we define the set of T of transit nodes to be C_q, where q is the smallest number such that $|C_q| \leq \tau$. Note that Lemma 4.4 implies that $|T| \geq \tau/h$, that is, T is not much smaller than the target size τ.

To compute the access nodes $A(v)$ for a vertex v, we build the shortest path tree D_v rooted at v and add a vertex $w \in T$ to $A(v)$ if w is the first node of T on the v–w path in D_v. (If $w \in T$, then $A(w) = \{w\}$.) For each access node w of v, we store $\text{dist}(v, w)$.

The output of the TN-MS preprocessing algorithm is the value of q, the set T, the table of pairwise distances for vertices in T, and the distances between every vertex and its access nodes. The local query algorithm may need additional information; Section 6.1 will explain the requirements of CH.

For comparison, the algorithm of Geisberger et al. [2012] takes the top τ vertices in the CH ordering as transit nodes, while our algorithm takes those in C_q. Although these two sets of transit nodes should be quite similar, they are not identical.

A TN-MS query from s to t is quite simple. We first run a global TN query. If s and t are far enough, we return the corresponding distance; otherwise, we run a local query. More precisely, we run the global TN query by computing, for all pairs (v_s, v_t) of access nodes of s and t, the length of the path s–v_s–v_t–t. Let the minimum such length be $g(s, t)$. If $g(s, t) \geq 5 \cdot 2^{q-1}$, we return $g(s, t)$. Otherwise, the query may be local, so we run a local (CH-based) s–t query and return the value it finds.

To prove that this algorithm is indeed correct, we need to establish a few lemmas. First, we note that access nodes are close to the corresponding vertex.

LEMMA 5.3. *For all $v \in V$ and all $w \in A(v)$,* $\text{dist}(v, w) \leq 2^{q-1}$.

PROOF. Assume there exists a vertex $v \in V$ and a vertex $w \in A(v)$ such that $\text{dist}(v, w) > 2^{q-1}$. Then, by Lemma 4.3 (applied to C_q), the shortest path from v to w must contain a vertex $u \in C_q$ with $\text{dist}(v, u) \leq 2^{q-1}$. But this implies $w \notin A(v)$, a contradiction. □

For large distances, global queries are correct.

LEMMA 5.4. *If* $\text{dist}(s, t) > 2^{q-1}$, *then* $g(s, t) = \text{dist}(s, t)$.

PROOF. Consider the shortest path P from s to t. If $\text{dist}(s, t) > 2^{q-1}$, then P contains at least one vertex of $C_q = T$. Let v_s be the first vertex of T on P and let v_t be the last vertex. By construction, $v_s \in A(s)$ and $v_t \in A(t)$, which implies the lemma. □

As we do not know $\text{dist}(s, t)$, we need the following lemma to decide when to run a local query.

LEMMA 5.5. *If* $\text{dist}(s, t) < 2^{q-1}$*, then* $g(s, t) < 5 \cdot 2^{q-1}$.

PROOF. Let the path corresponding to $g(s, t)$ be s–s'–t'–t. We know that $\text{dist}(s', t') \leq \text{dist}(s', s) + \text{dist}(s, t) + \text{dist}(t, t') < 2^{q-1} + 2^{q-1} + 2^{q-1} = 3 \cdot 2^{q-1}$. By triangle inequality, $g(s, t) \leq \text{dist}(s, s') + \text{dist}(s', t') + \text{dist}(t', t) \leq 2 \cdot 2^{q-1} + \text{dist}(s', t') < 5 \cdot 2^{q-1}$. □

TN-MS has a natural locality filter: if the global query result is too big, run a local query. In contrast, locality filters in most previous TN implementations [Bast et al. 2009; Sanders and Schultes 2009; Geisberger et al. 2012] require vertex coordinates and use Euclidean distances. The only exception is the recent algorithm of Arz et al. [2013], which uses partial hub labels for local queries.

To analyze the running time of our algorithm, we first need to bound the number of access nodes.

LEMMA 5.6. *Every vertex v has $O(h)$ access nodes.*

PROOF. If $v \in T$, then $A(v) = \{v\}$ by construction, so assume $v \notin T$. By Lemma 5.3, for any vertex $w \in A(v)$ we have $w \in B_{2 \cdot 2^{q-1}}(v)$. The fact that T is a $(2^{q-1}, h)$-SPHS completes the proof. □

THEOREM 5.7. *TN-MS can be implemented so that global queries take $O(h^2)$ time.*

Local queries have the same (higher) bound as CH, which we discuss next.

6. SHORTCUT-BASED ALGORITHMS

Neither HL nor TN need the actual input graph at query time (ignoring local queries for TN); instead, they operate only on the precomputed auxiliary data. In this section, we turn our attention to two important graph-based algorithms: CH and RE.

6.1. Contraction Hierarchies and Shortcuts

Both CH and RE use the notion of *shortcuts* [Schulz et al. 2000]. Shortcuts are especially natural in the context of CH, which we describe in this section.

Given two vertices $u, w \in V$, a *shortcut* is a new edge $e = (u, w)$ with length $\text{dist}(u, w)$. The *shortcut operation* [Geisberger et al. 2012] deletes a vertex v from the graph and adds edges between its neighbors to maintain the shortest path information. In particular, for any neighbors u and w of v such that the concatenation of (u, v) and (v, w) is the unique shortest u–w path, we add (u, w) with $\ell(u, w) = \ell(u, v) + \ell(v, w)$. The shortcut edges are added to the original graph. This breaks the invariant that shortest paths are unique; in the augmented graph, they are not. Breaking ties by favoring paths with fewer edges is sufficient for our purposes, even though it does not eliminate all ties.

Given the notion of shortcuts, CH preprocessing is straightforward: pick a total order among the vertices and shortcut them sequentially in this order, until a single vertex remains. The output of preprocessing is the set E^+ of shortcut edges and the vertex order. We denote the position of a vertex v in the ordering by $rank(v)$.

An s–t CH query runs a pruned bidirectional Dijkstra search on the graph $G^+ = (V, E \cup E^+)$. When scanning v, only the edges (v, w) with $rank(v) < rank(w)$ are examined. The search may safely terminate when the priority queues for both search directions become empty. (A tighter criterion is to stop each search when the top element of its priority queue is greater than the shortest path seen so far.) At this point, each vertex v has estimates $d_s(v)$ and $d_t(v)$ on distances from s to v and from v to t. (Unlike in Dijkstra's algorithm, these estimates may be greater than the actual distances for some vertices; in particular, they will be infinity for unvisited ones.) As proven by Geisberger et al. [2012], a vertex u minimizing $d_s(u) + d_t(u)$ is guaranteed to be on a shortest path from s to t, given by the concatenation of the s–u and u–t paths.

Although queries are correct for any vertex ordering, their running times and the size of the auxiliary data required may vary greatly. An online ordering heuristic that works well in practice [Geisberger et al. 2012] selects the next vertex to shortcut based on its current degree and the number of new edges added to the graph, among other factors.

Note that the query returns a path in the augmented graph (and its length). In applications where the corresponding original path is required, we must translate each shortcut into a sequence of original edges. This can be done in time linear in the size of the sequence [Geisberger et al. 2012], as long as we remember (as part of the auxiliary data) which were the two elements (edges or shortcuts) combined to create each shortcut added during preprocessing. This requires $O(|E^+|)$ space.

This remarkably simple algorithm is surprisingly efficient on road networks. On a standard benchmark instance representing the road network of Western Europe [PTV AG - Planung Transport Verkehr 1979; Demetrescu et al. 2009], random queries visit fewer than 500 vertices (out of 18 million) on average. Moreover, the preprocessing algorithm adds fewer shortcuts than there are original edges.

6.2. Computing Shortcuts via Multiscale SPHS

We now describe an algorithm to order vertices and add shortcuts using multiscale SPHS $\{C_i : i = 0, \dots, \lceil \log D \rceil\}$. For $0 \le i \le \lceil \log D \rceil$, let $Q_i = C_i \setminus \bigcup_{j=i+1}^{\lceil \log D \rceil} C_j$ be the set of vertices that appear in C_i but no higher. The sets Q_i partition V into *levels*. Define a contraction order as follows: for every i, the vertices in Q_i come after those in Q_{i+1}; within each Q_i, the order is arbitrary.

Next, we prove some bounds for the shortcuts resulting from this order.

LEMMA 6.1. *For fixed v and j, the number of edges $(v, w) \in E^+$ with $w \in Q_j$ is at most h.*

PROOF. Suppose $v \in Q_i$ and let $\gamma = \min\{i, j\}$. The shortcut (v, w) is created when we remove the last internal vertex z on the shortest path $P = (v = u_1, u_2, \dots, u_t = w)$ between v and w in the underlying graph. Therefore, $z \in Q_x$, for some $x \le \gamma$.

Suppose for contradiction that $\ell(P) > 2^\gamma$. Note that since (v, w) is a shortcut, we add $(v, w) \notin E$, so P contains more than one edge. Let P' be the path obtained by deleting (v, u_2) if $rank(v) > rank(w)$ and deleting (u_{t-1}, w) otherwise. Since P' is an $\ell(P)$-significant path, it must contain a vertex $u \in Q_y$ for $y > \gamma$. But this is impossible, so $\ell(P) \le 2^\gamma$. Therefore, $w \in B_{2^\gamma}(v)$. Also, recall that $w \in Q_j \subseteq C_j$ for $j \ge \gamma$. Since $|Q_j \cap B_{2^\gamma}(v)| \le |C_j \cap B_{2^\gamma}(v)| \le |C_j \cap B_{2^j}(v)| \le h$, the lemma follows. □

Combined with the results of Lemma 3.5, these lemmas imply the following bounds on preprocessing (to be used in the CH and Reach algorithms):

THEOREM 6.2. *If (G, ℓ) has highway dimension h, then preprocessing based on multiscale SPHS produces a set of shortcuts E^+ such that degree of every vertex in $G(V, E \cup E^+)$ is at most $h + h\lceil \log D \rceil$ and $|E^+| \le nh\lceil \log D \rceil$.*

6.3. Remarks on Dijkstra's Algorithm

In Sections 6.4 and 6.5, we study pruned variants of the bidirectional Dijkstra's algorithm on the graph $G(V, E \cup E^+)$ and show that they examine $O(h \log D)$ vertices. By Theorem 6.2, vertices in this graph have degree $O(h \log D)$. The following lemma bounds the complexity of the algorithm (excluding the computation related to pruning, which will be amortized).

LEMMA 6.3. *The bidirectional Dijkstra's algorithm can be implemented so that an execution of it scans $O(h \log D)$ vertices on a graph with maximum degree $O(h \log D)$ runs in $O((h \log D)^2 + h \log D \log n)$ time if ℓ is real valued and in $O((h \log D)^2)$ time if ℓ is integral.*

PROOF. The implementation of the bidirectional Dijkstra's algorithm using Fibonacci heaps [Fredman and Tarjan 1987] runs in time $O(m' + n' \log n)$, where n' is the number of vertices scanned by the algorithm and m' is the number of edges adjacent to these vertices. This gives the $O((h \log D)^2 + h \log D \log n)$ bound.

When edge lengths are integral, the multilevel bucket data structure [Denardo and Fox 1979] yields an $O((h \log D)^2 + h \log D \log C)$ bound, where C is the maximum edge length. Since we assume that every edge is a shortest path, we have $D \geq C$, which implies the $O((h \log D)^2)$ bound. □

To simplify the statements of the following Theorems 6.4 and 6.6, we will only consider integral lengths. For real-valued lengths, the theorems hold with an extra additive term of $O(h \log(D) \log n)$ in the bounds.

6.4. CH Query Analysis

To analyze a CH query on a graph with highway dimension h, we would like to show that it visits at most $O(h)$ vertices in each level Q_i. It would be sufficient to show that all vertices visited by the forward search at level Q_i are within distance at most 2^i from the source s (a similar argument would hold for the backward search). This is almost true. If a vertex $v \in Q_i$ is such that $\ell(P(s, v)) > 2^i$, then there must be a vertex $u \in Q_j$ (with $j > i$) on $P(s, v)$. This means that $rank(u) > rank(v)$, so the forward CH search would never follow path $P(s, v)$ in its entirety. Because of pruning, however, not every branch of the search tree followed by CH is a shortest path. Conceivably, the search could find an alternative (not shortest) path of increasing ranks from s to v.

A simple way to avoid this issue is to add *range optimization* to CH: when scanning an edge (v, w) with $w \in Q_j$, if the candidate label for w from the scan is $d(w) > 2^j$, we prune the search by not adding w to the the priority queue. (Because a shortest path of length greater than 2^j must pass through a vertex in Q_{j+1}, we know the current path to w is not the shortest.) This modification requires knowing the level of a vertex (in the multiscale SPHS), which can be implemented with a constant overhead per vertex at query time. The stall-on-demand pruning technique [Geisberger et al. 2012] can be viewed as a heuristic solution to the same problem.

We get the following time bound.

THEOREM 6.4. *For (G, ℓ) with highway dimension h and integral ℓ, there is an ordering of vertices such that a CH query implemented using range optimization takes $O((h \log D)^2)$ time.*

PROOF. It is enough to show that the bidirectional Dijkstra's algorithm scans $O(n \log D)$ vertices; we can then apply Lemma 6.3. Consider the forward search from s (the reverse search is similar). Because of range optimization, the search scans $w \in Q_j$ only if $d(w) \leq 2^j$, and therefore $\text{dist}(s, w) \leq 2^j$. Since $Q_i \subseteq C_i$ and $|C_i \cap B_2^i(s)| \leq h$, for a fixed i, the number of scanned vertices from Q_i is at most h, and the total over all i is $O(h \log D)$. □

6.5. The Reach Algorithm

The RE algorithm [Goldberg et al. 2009] is based on the notion of *reach* [Gutman 2004]. Given a path P and a vertex $v \in P$ that divides P into P_1 and P_2, the reach of v with respect to P is $r_P(v) = \min\{\ell(P_1), \ell(P_2)\}$. Let $\mathcal{P}(v)$ be the set of *shortest* paths containing v. The *reach* of v (with respect to the entire graph) is $r(v) = \max_{P \in \mathcal{P}(v)} r_P(v)$. Intuitively, a vertex has high reach if it is close to the middle of a long shortest path.

The preprocessing stage of the original RE algorithm heuristically adds shortcuts to the graph and computes upper bounds r on reaches in the augmented graph. (Because we break ties by number of edges, adding shortcuts reduces the reaches of bypassed vertices.) An s–t query performs bidirectional Dijkstra search pruned by reach. More precisely, consider what happens when the forward search labels a vertex v with distance label $d(v)$. If v has already been processed by the backward search, a new candidate path has been found; the length of the best known path is updated accordingly. Otherwise, we check if $r(v)$ is smaller than the minimum of $d(v)$ and b^*, the distance label of the top priority queue element of the backward search. If so, we do not add v to the priority queue (if v were on the shortest path from s to t, its reach would be at least $\min\{d(v), b^*\}$). A symmetric pruning is done for the backward search.

Bidirectional Dijkstra search will give correct answers irrespectively of when one switches between forward and backward searches. By default, we consider *balanced RE*, the variant that balances the two searches by distance traversed: in each iteration, choose the direction (backward or forward) whose minimum labeled vertex distance is smaller, breaking ties arbitrarily.

To obtain provably good query times, we use the shortcuts E^+ induced by the multiscale SPHS, as described in Section 6.2.

LEMMA 6.5. *For any $v \in Q_i$, $r(v) \leq 2^i$ in $G^+ = (V, E \cup E^+)$.*

PROOF. Suppose for contradiction that $r(v) > 2^i$. Then, there is a shortest path P in G^+ between a vertex x and a vertex y such that (i) P contains v and (ii) both the subpath P_1 from x to v and the subpath P_2 from v to y are longer than 2^i. Note that both P_1 and P_2 must contain vertices $u \in Q_j$ for $j > i$. Among these, let u_1 and u_2 be the closest vertices to v on P_1 and P_2, respectively. It follows that all vertices of P between u_1 and u_2 (including v) will be shortcut before u_1 and u_2, which means (u_1, u_2) must be a shortcut. The x–y path using the shortcut has the same length as P but a smaller number of hops. Since we break ties by number of hops, we prefer it over P. This contradicts the assumption that P is the shortest path. □

With these bounds, we can prove the following time bound on RE queries.

THEOREM 6.6. *For (G, ℓ) with highway dimension h and integral ℓ, there is an ordering of vertices such that a balanced RE query takes $O((h \log D)^2)$ time.*

PROOF. For the forward search from s, consider the ball $B_{2 \cdot 2^{i-1}}(s)$. The search does not scan any $v \in Q_i$ such that v is outside the ball. This is because $r(v) \leq 2^i$ (by Lemma 6.5), which implies that v is either scanned by the backward search or not scanned at all. Therefore, the search scans $O(h \log D)$ vertices. A similar argument holds for the backward search. The fact that vertex degrees are bounded by $O(h \log D)$ completes the proof. □

Note that the RE query algorithm does not need the vertex ordering but needs the reach bounds. For a vertex $v \in Q_i$, we can store 2^i (which can be represented by the exponent) as its bound. The query time bound also holds for any reach upper bounds that are at least as good as those computed by our preprocessing. In particular, it holds for optimal reach values in the graph with added shortcuts, which can be computed in polynomial time.

Recall that the query returns an implicit representation of the shortest path containing shortcuts. One can use the method outlined in Section 6.1 to expand this path into the underlying path (using only original edges) in linear time.

7. VC-DIMENSION AND USP SYSTEMS

In this section, we show a relationship between shortest paths and VC-dimension, a measure of set system complexity used in learning and computational geometry. Intuitively, we show that (unique) shortest paths form a simple set system. This result is independent of the concept of highway dimension. In Section 8, we use this result to obtain better approximation algorithms for constructing multiscale SPHS for small highway dimension.

A *set system* $(X, \mathcal{R})$ consists of a *base set* X and a collection $\mathcal{R}$ of its subsets. A *hitting set* H is a subset of X that intersects every element of $\mathcal{R}$. Given a set $Y \subseteq X$, let $(Y, \mathcal{R}_{|Y})$, where $\mathcal{R}_{|Y} = \{S \cap Y | S \in \mathcal{R}\}$, be the set system *induced by* $\mathcal{R}$. We say that Y is *shattered* by $\mathcal{R}$ if $\mathcal{R}_{|Y} = 2^Y$ (i.e., it includes every subset of Y). The set system $(X, \mathcal{R})$ has *VC-dimension d* if d is the smallest integer such that no subset $Y \subseteq X$ with $|Y| = d + 1$ can be shattered [Vapnik and Chervonenkis 1971].

Some algorithmic problems become simpler on set systems with low VC-dimension. In particular, while the classical algorithm for the hitting set problem achieves an $O(\log |X|)$ approximation [Chvátal 1979], one can do better when the VC-dimension is small, as shown by Clarkson [1988, 1993] (randomized version) and Brönnimann and Goodrich [1995] (derandomization).

THEOREM 7.1. *For a set system with VC-dimension d and smallest hitting set size h, there is a polynomial-time algorithm to find a hitting set of size* $O(hd \log(hd))$.

This result is based on a boosting-type algorithm from learning theory. Even et al. [2005] proposed an alternative algorithm based on linear programming.

In this article, we study set systems $(V, \mathcal{R})$ where V is the set of vertices of a graph, and each element of $\mathcal{R}$ corresponds to (the set of vertices of) a shortest path. We refer to such set systems for graphs with Unique Shortest Paths as *USP systems*.

The following theorem shows that USP systems have low VC-dimension.[1]

THEOREM 7.2. *A USP system* $(V, \mathcal{R})$ *has VC-dimension at most two.*

PROOF. We need to show that no three-vertex set $Y = \{a, b, c\}$ can be shattered. If there is no path $P \in \mathcal{R}$ such that $Y \subseteq P$, Y is clearly not shattered. So assume there exists a shortest path $P \in \mathcal{R}$ such that $Y \subseteq P$. Without loss of generality, assume that b lies between a and c on P. Consider $Z = \{a, c\}$. By uniqueness of shortest paths, any shortest path containing a and c must contain b, so Y is not shattered. □

In combination with Theorem 7.1, this implies the following result.

COROLLARY 7.3. *For a USP system with an optimal hitting set of size h, there is a polynomial-time algorithm to find a hitting set of size* $O(h \log h)$.

8. POLYNOMIAL-TIME PREPROCESSING

We do not know how to compute optimal (h, r)-SPHS in polynomial time. In this section, we show how to efficiently construct approximate SPHS, that is, (h', r)-SPHS such that h' is (slightly) bigger than h. In particular, we use Corollary 7.3 to get $h' = O(h \log h)$.

[1]This relationship has been independently discovered by Tao et al. [2011].

Consider a simple greedy algorithm [Johnson 1974], which in each iteration adds to the solution the vertex that hits the most uncovered paths. We show that this algorithm produces an $O(\log n)$ approximation.

LEMMA 8.1. *If G has highway dimension h, then for any r the greedy algorithm computes an* $(O(h \log n), r)$*-SPHS.*

PROOF. Pick $v \in V$ and consider $B_{2r}(v)$. By the definition of highway dimension, there is a hitting set H for $\mathcal{S}_r(v)$ with $|H| \leq h$. We call a path in $\mathcal{S}_r(v)$ *relevant* if it is not yet hit by the set of vertices already picked by the algorithm.

Suppose at some step the algorithm chooses a vertex w in $B_{2r}(v)$. Every path newly hit by w must be in $\mathcal{S}_r(v)$, and by the greedy choice of w and the fact that H is a hitting set of $\mathcal{S}_r(v)$, w hits at least $1/h$ of the relevant paths. As the initial number of relevant paths is $O(n^2)$, the algorithm can choose $O(h \log n)$ vertices in $B_{2r}(v)$. $\square$

A straightforward implementation of the greedy algorithm computes all pairs of shortest paths in every iteration and runs in $\tilde{O}(n^2 m)$ time using linear space. An alternative implementation [Abraham et al. 2012b] runs in $\tilde{O}(nm)$ time but needs $\Theta(n^2)$ space.

Using the results derived in Section 7, we can do even better. We now describe a polynomial-time algorithm to compute an $O(\log h)$ approximation of SPHS, which is a significant improvement for $h \ll n$ (e.g., h is polylogarithmic in n). The result is slightly more complicated than a direct application of Corollary 7.3 because we want to get a sparse hitting set, and not just a small one.

THEOREM 8.2. *There exists a polynomial-time algorithm that, given a graph* (G, ℓ) *with the highway dimension of h and an* $r > 0$*, computes an* $(O(h \log h), r)$*-SPHS.*

PROOF. Let c be the constant hidden by the big-"O" notation in Theorem 7.1. By Corollary 7.3, we can efficiently compute a hitting set of size at most $h' = 2hc \log(2h)$ for any $\mathcal{S}_r(u)$. Our goal is to build an (h', r)-SPHS.

We maintain a hitting set H' for $\mathcal{P}_r$. We can start with any such H' computed in polynomial time, or simply take $H' = V$. We show that if H' is not sparse, then we can replace H' by a smaller hitting set for $\mathcal{P}^r_{2r}$ in polynomial time. Iterating this construction, we will stop with a sparse H' in at most n steps.

While H' is not sparse, there exists a vertex u and a set U such that $U = H' \cap B_{2r}(u)$ and $|U| > h'$. Note that U hits only paths in $\mathcal{S}_r(u)$. We compute a hitting set H for $\mathcal{S}_r(u)$ with $|H| \leq h'$. Now $(H' \backslash U) \cup H$ is a hitting set for $\mathcal{P}_r$ that is smaller than H'. $\square$

Since computing (h, r)-SPHS is the only nonpolynomial part of preprocessing for the variants of HL, TN-MS, CH, and RE for which we obtain good query bounds, Theorem 8.2 allows us to get variants of these algorithms with polynomial-time preprocessing. However, instead of (h, r)-SPHSs, we use $(O(h \log h), r)$-SPHSs, causing some $O(h)$ terms in the query bounds to change to $O(h \log h)$. For completeness, we state the bounds corresponding to those in Theorems 5.1, 5.7, 6.2, 6.4, and 6.6:

THEOREM 8.3. *There exists a polynomial-time algorithm that, given a graph with HD h, computes a hub labeling with maximum label size* $O(h \log h \log D)$*. For this labeling, an HL query takes* $O(h \log h \log D)$ *time.*

THEOREM 8.4. *For a variant of TN-MS with polynomial-time preprocessing, global queries take* $O((h \log h)^2)$ *time.*

THEOREM 8.5. *The polynomial-time preprocessing based on multiscale SPHS produces* E^+ *such that the degree of every vertex in* $G(V, E \cup E^+)$ *is at most* $O(h \log h \log D)$ *and* $|E^+| = O(nh \log h \log D)$.

THEOREM 8.6. *With polynomial preprocessing, CH queries with range optimization take* $O((h \log h \log D)^2)$ *time.*

THEOREM 8.7. *With polynomial preprocessing, a balanced RE query takes* $O((h \log h \log D)^2)$ *time.*

9. RELATION TO DOUBLING DIMENSION

We now discuss the relationship between highway dimension and the well-known concept of doubling dimension [Gupta et al. 2003]. A *metric space* is a pair (M, μ) where M is a set and μ is a function $M \times M \to \mathbb{R}$ that, for any $x, y, z \in M$, satisfies (i) $\mu(x, y) \geq 0$, (ii) $\mu(x, y) = 0$ if and only if $x = y$, (iii) $\mu(x, y) = \mu(y, x)$, and (iv) $\mu(x, y) + \mu(y, z) \geq \mu(x, z)$. A metric space has *doubling dimension* d if every ball of radius r can be covered by 2^d balls of radius $r/2$. Given an undirected graph $G = (V, E)$ and a positive length function ℓ, it is easy to check that (V, dist) is a metric space. We define the doubling dimension of (G, ℓ) to be the doubling dimension of this metric space.

Constant doubling dimension does not imply constant highway dimension (consider a unit weight two-dimensional grid), but the converse is true.

THEOREM 9.1. *If* (G, ℓ) *has highway dimension h, then its doubling dimension is at most* $\log(h + 1)$.

PROOF. Consider any vertex v, and let H be a hitting set for $S_r(v)$ with $|H| \leq h$. We show that the union of the radius-r balls centered at $H \cup \{v\}$ covers the ball $B_{2r}(v)$. Consider $w \in B_{2r}$ and assume $w \notin B_r(v)$. Then the shortest path P from v to w is in $S_r(v)$ and it is easy to see that $S_r(v)$ is w-closed. By Lemma 4.3, there is $u \in P \cap H$ such that $\text{dist}(w, u) \leq r$. Therefore, $w \in B_r(u)$. □

This bound is fairly tight. Consider a star graph with n vertices and all edges of length two. It is easy to see that the doubling dimension of this graph is $\log n$ and the highway dimension is n.

Note that requiring a graph to have low doubling dimension is not enough to explain empirical observations. Consider square two-dimensional grid networks with unit lengths and n vertices, and perturb arc lengths to make shortest paths unique. Although these networks have constant doubling dimension, RE, CH, and TN perform much worse on them than on road networks of similar size [Goldberg et al. 2009; Bauer et al. 2010a, 2011]. The highway dimension of such grids is $\Theta(\sqrt{n})$, which is relatively large and more consistent with the empirical results. This example also shows that these algorithms may have worse performance on general planar graphs than on road networks.

Although we do not require Theorem 9.1 for any of the algorithms or analyses presented previously, it is interesting as it relates HD to previous work and suggests that results holding for small doubling dimension may be strengthened under the small HD assumption. Furthermore, it provides intuition for the performance of A* search with landmarks [Goldberg and Harrelson 2005] on road networks under the small HD assumption (which implies small doubling dimension): Kleinberg et al. [2009] show that if the doubling dimension is small, a small number of landmarks yields good distance bounds for almost all vertex pairs.

10. EMERGENCE OF NETWORKS WITH CONSTANT HIGHWAY DIMENSION

The formation of real road networks is a complex process involving geographic, economic, engineering, political, and cultural aspects. In this section, we propose a simple model that captures some of these aspects. We then show that networks formed by this process have constant highway dimension. This provides a partial explanation for the emergence of low HD networks. We do not claim our model captures all aspects of road network formation, but the assumptions we make are enough to yield networks with constant highway dimension.

We would like to capture three properties of road network formation.

(1) Roads are built incrementally over time, and each decision is typically done in a local manner, without necessarily taking into account a centralized global planner. Hence, we consider a *decentralized* and *online* process of forming a road network.
(2) The underlying geometric space on which roads are built has some low-dimensional structure. To capture this property, we assume that the underlying geometric space has low doubling dimension. We denote the distance between two points v and w in this space as $\mu(v, w)$.
(3) Longer highways are typically faster than short roads. (For example, interstate highways are typically faster than state highways, which are faster than local interneighborhood roads, which are faster than small interneighborhood roads, and so on.) To formalize this we introduce a *speedup parameter* $0 < \delta < 1$ and define the *traversal time* $\tau(u, v)$ of a road segment with endpoints u, v to be $\mu(u, v)^{1-\delta}$. We are interested in computing shortest paths with respect to τ.

Consider the following model, which is similar in spirit to the dynamic spanner construction of Gottlieb and Roditty [2008]. Start with a metric space (M, μ) with doubling dimension $\log \alpha$. An adversary supplies a sequence of distinct points $v_1, \ldots, v_n \in M$. Let $V = \{v_1, \ldots, v_n\}$. We scale μ so that $\min_{v \neq w} \mu^{(1-\delta)}(v, w) = 1$. Then, we process the vertices from v_1 to v_n in order; when processing v_t, we connect it to nearby peers $v_i : i < t$ at appropriate scales. Intuitively, if v_t is a new city, we connect it to nearby cities; if it is a new neighborhood, we connect it to nearby neighborhoods; and so on. However, as neighborhoods are created farther and farther away from the city center, there comes a point where we connect a new neighborhood not only to nearby neighborhoods, but also to nearby city centers.

Formally, let $\Delta = \max \mu(v, w)$. For each integer $0 \leq i \leq \log D$, we maintain a set $C_i \subseteq V$ such that any two vertices in C_i are at least 2^i apart in the metric space. We say that C_i is a 2^i-packing. The packings we maintain are hierarchical, that is, if $v \in C_i$, v is in C_j for all $j < i$. All vertices are in C_0. We say that the *level* of v is i if i is the maximum integer for which $v \in C_i$.

The first point (vertex) supplied by the adversary is assigned to level Δ (i.e., belongs to all C_i sets). The algorithm for processing each subsequent vertex v is as follows:

(1) Let i be the largest index such that $\mu(v, C_i) \geq 2^i$. Add v to C_j for all $0 \leq j \leq i$.
(2) For each j such that $v \in C_j$, add edges from v to all vertices $\{u \in C_j \mid \mu(u, v) \leq 6 \cdot 2^j\}$. Note that this set may contain only v, in which case no edges are added. For each such new edge $e = (u, v)$, note that $2^j \leq \mu(u, v) \leq 6 \cdot 2^j$.
(3) Add an edge from v to the closest vertex in C_{i+1}, where i is the level of v (unless $i = \log \Delta$). Note that the length of such an edge is at most 2^{i+1} because we have not added v to C_{i+1}. The length is also greater than 2^i, since the vertex we connect v to is in C_i (by our choice of i and the hierarchical property).

LEMMA 10.1. *For an edge (v, w), if vertex v has level i, then $\mu(v, w) \leq 6 \cdot 2^i$.*

PROOF. If we process v after w and add (v, w), then either $w \in C_i$ and $\mu(v, w) \leq 6 \cdot 2^i$, or $w \in C_{i+1}$ and $\mu(v, w) \leq 2 \cdot 2^i < 6 \cdot 2^i$. If we process v before w and add (v, w), then either w is at a level $j \leq i$ and $\mu(v, w) \leq 6 \cdot 2^j$, or w is at a level $j < i$ and $\mu(v, w) \leq 2 \cdot 2^j \leq 2^i < 6 \cdot 2^i$. □

Next, we show that the number of edges added in this process is not too large.

LEMMA 10.2. *Processing of a new point v causes at most $\alpha^4(\log \Delta + 1)$ edge insertions.*

PROOF. We claim that, for each v, we add at most α^4 edges for each $0 \leq i \leq \log \Delta$. To see this, we need to prove that the ball of radius $6 \cdot 2^i$ around v contains at most α^4 vertices in C_i. By the assumption of doubling dimension α, this ball can be covered by α^4 balls of radius $6 \cdot 2^{i-4} < 2^{i-1}$. By the definition of C_i, each of the balls in the cover contains at most one vertex of C_i. □

Next, we prove that the graph created by this process has small highway dimension. Recall that we have a speedup parameter $\delta \in (0, 1)$, so an edge of length $\mu(u, v)$ (in the metric space) has traversal time $\tau(u, v) = \mu(u, v)^{1-\delta}$. To simplify the analysis, we fix $\delta = 1/4$. The following proofs can be modified to show that Theorem 10.6 holds for any constant $\delta \in (0, 1)$.

We shall refer to shortest paths with respect to traversal times τ as *fastest paths*. By shortest paths we will mean shortest paths with respect to μ. Given a path P in the graph, we denote by $\mu(P)$ the length of the path (i.e., the sum over all edges of distances between endpoints) and by $\tau(P)$, the total traversal time of the path P.

For the next three lemmas, let $x \neq y \in V$ and i be such that $2^i \leq \mu(x, y) < 2^{i+1}$.

LEMMA 10.3. *For $\delta = 1/4$, there exists an x–y path P such that $\tau(P) < 9 \cdot 2^{i \cdot 3/4}$.*

PROOF. Consider the following recursive construction of a sequence of vertices $x = x_0, x_1, \ldots, x_i$. Given x_{j-1}, let x_j be x_{j-1} if $x_{j-1} \in C_j$; otherwise, let x_j be the vertex in C_j we connected x_{j-1} to when adding it to the graph. Note that $\mu(x_{j-1}, x_j) \leq 2^j$. Therefore, there is a path P_x from x to $x_i \in C_i$ of length $\mu(P_x) \leq \sum_{j=1}^{i} 2^j \leq 2 \cdot 2^i$. The transit time $\tau(P_x)$ is at most

$$\sum_{j=1}^{i} 2^{j \cdot 3/4} \leq \frac{2^{(i+1) \cdot 3/4} - 1}{2^{3/4} - 1} < \frac{2^{3/4}}{2^{3/4} - 1} \cdot 2^{i \cdot 3/4} < 2.5 \cdot 2^{i \cdot 3/4}.$$

Similarly, there is a path P_y from y to $y_i \in C_i$ with $\mu(P_y) \leq 2 \cdot 2^i$ and $\tau(P_y) < 2.5 \cdot 2^{i \cdot 3/4}$.

Both x_i and y_i are in C_i and $\mu(x_i, y_i) \leq \mu(x_i, x) + \mu(x, y) + \mu(y, y_i) \leq 6 \cdot 2^i$. Without loss of generality, assume that y_i has been added after x_i. Then, we added the edge (x_i, y_i) as well. In combination with P_x and P_y, this edge gives an x–y path P with

$$\tau(P) < 5 \cdot 2^{i \cdot 3/4} + 6^{3/4} 2^{i \cdot 3/4} \leq 9 \cdot 2^{i \cdot 3/4},$$

using the inequality $6^{3/4} < 4$. □

LEMMA 10.4. *For $\delta = 1/4$, the longest edge $a = (u, v)$ on the fastest s–t path Q satisfies $\mu(u, v) \geq 9^{-4} \cdot 2^i$.*

PROOF. Note that the average speed (distance/time) on Q is at most the speed on a, which is $\mu(u, v)^{1/4}$. Thus, the traversal time of Q is at least 2^i divided by the speed, and applying Lemma 10.3, we get

$$\frac{2^i}{\mu(u, v)^{1/4}} \leq \tau(Q) \leq 9 \cdot 2^{i \cdot 3/4}.$$

Therefore,

$$\mu(u, v) \geq \left(\frac{2^i}{9 \cdot 2^{i \cdot 3/4}} \right)^4 = 9^{-4} 2^i. \quad \square$$

LEMMA 10.5. *For $\delta = 1/4$, the fastest x–y path goes through a vertex $v \in C_k$ with $i - 16 < k$; this implies that the level of v is greater than $i - 16$.*

PROOF. Consider the longest edge $a = (u, v)$ as in Lemma 10.4 and without loss of generality assume that u has been added to the graph after v. The edge has been added because $v \in C_k$ for some k, and either $u \in C_k$, in which case $\mu(u, v) \leq 6 \cdot 2^k$, or $u \in C_{k-1}$ and $\mu(u, v) \leq 2^{k+1}$. In both cases, the former bound applies: $2^k \leq \mu(u, v) \leq 6 \cdot 2^k$. Using Lemma 10.4, we get $9^{-4} \cdot 2^i \leq \mu(u, v) \leq 6 \cdot 2^k$ and therefore $k > i - 3 - 4(\log_2 9) > i - 16$. □

THEOREM 10.6. *For $\delta = 1/4$ and a length function τ, the highway dimension of a network constructed as previously is α^{20}.*

PROOF. Consider $u \in V$, $r > 0$, and i such that $2^i \leq r < 2^{i+1}$. We will show that, for $k > i - 17$, $H = B_{4r}(u) \cap C_k$ is a hitting set for $S_r(u)$. To bound the size of H, we note that $B_{4r}(u)$ can be covered by $\alpha^{\log(4r)/2^{i-17}} < \alpha^{20}$ balls of radius 2^{i-17}. Each of the latter balls contains at most one vertex of C_k, so $|H| \leq \alpha^{20}$.

Recall that S_r contains r-significant paths P with $\tau(u, P) \leq 2r$. It is enough to find a small hitting set for the set of minimal P such that P is r-significant and $\tau(u, P) \leq 2r$; this set will hit all the nonminimal paths as well. Let P be such a minimal path.

If $\tau(P) \leq r/2$, then since P is r-significant; P can be extended to a fastest path of length greater than r by adding an edge on one or both ends. Denote the only or the longer of such edges by (v, w), where $v \in P$. Then, $\tau(v, w) > r/4$ and $\mu(v, w) > (r/4)^{4/3}$. By Lemma 10.1, if k is the level of v, then $6 \cdot 2^k \geq \mu(v, w) > (r/4)^{4/3}$. Therefore, $k > (4/3)\log(r) - 2 - \log 6 > (4/3)i - 5 > i - 17$. In addition, $\tau(u, v) \leq 2r + \tau(P) \leq 2r + r/4 < 4r$.

If $\ell(P) > r/2$, then by Lemma 10.5, P contains a vertex v at level $k > \log(r/2) - 16 > i - 17$. We have two cases: $\tau(u, v) \leq r$ and $\tau(u, v) > r$. In the latter case, let (w, v) be the last edge on the u–v portion of P. By the minimality of P, $\tau(u, w) \leq r$ and $\tau(w, v) \leq r$ (otherwise we could truncate P at w, so P would not be minimal). Therefore, $\tau(u, v) \leq 2r + r + r = 4r$. □

The theorem shows that a fairly simple model can be used to generate networks with constant highway dimension. Note that the constant hidden by the big "O" notation is large. To keep the model simple, we do not attempt to model several important properties of real-life road networks. For example, road networks have small separators. They also have the "Steiner" property: a new vertex may be connected to a point on an existing edge, which corresponds to creating a new intersection on an existing road segment. Furthermore, we allow adversarial vertex placement, but in real life new vertices are usually added close to existing access points. A more sophisticated model may lead to tighter bounds.

11. HIGHWAY DIMENSION ON CONTINUOUS GRAPHS

Our definition of highway dimension (Definition 3.1) can be seen as the discrete analog of a natural definition for continuous graphs. In this context, the definitions of r-significant and (r, d)-close paths become natural. Continuous graphs [Smyth 1995] fit real road networks well. For these networks, edges correspond to road segments and vertices correspond to intersections. User positions and addresses need not be at the intersections: they can be at some point along a road segment, and continuous graphs model this phenomenon.

As the goal of this section is to provide more intuition into the definition of HD, we treat continuous graphs somewhat informally. For formal definitions (e.g., of shortest paths), see Smyth [1995].

Given a finite graph G, we define the corresponding continuous graph $|G|$ as follows. To each edge in G, we assign a unit interval $[0, 1]$; endpoints of these intervals are glued together at vertices of G. Note the $|G|$ has infinitely many vertices and that each vertex of $|G|$ corresponds either to a vertex of G or to a pair (e, α), where $e \in E$ and $\alpha \in (0, 1)$. Let (V, d) be the metric space defined by a weighted graph $G = (V, E)$. We define a metric space $(|G|, \bar{d})$ where the distance $\bar{d}$ between two vertices $(e = (u, v), \alpha)$ and $(e' = (u', v'), \alpha')$ is defined as follows.

If $e \neq e'$, then $\bar{d}((e, \alpha), (e', \alpha'))$ is the shortest path distance between vertices x and x' in a discrete graph $G^* = (V \cup \{x, x'\}, E \cup \{(u, x), (x, v), (u', x'), (x', v')\})$ with $\ell(u, x) = \alpha\ell(u, v)$, $\ell(x, v) = (1 - \alpha)\ell(u, v)$, $\ell(u', x') = \alpha'\ell(u', v')$, and $\ell(x', v') = (1 - \alpha')\ell(u', v')$; the remaining edges keep their original lengths.

If $e = e' = (u, v)$, without loss of generality assume $\alpha \leq \alpha'$. Then $\bar{d}((e, \alpha), (e, \alpha'))$ is the shortest path distance between vertices x and x' in a discrete graph $G^* = (V \cup \{x, x'\}, E \cup \{(u, x), (x, x'), (x', v)\})$ with $\ell(u, x) = \alpha\ell(u, v)$, $\ell(x, x') = (\alpha' - \alpha)\ell(u, v)$, and $\ell(x', v) = (1 - \alpha')\ell(u, v)$; the remaining edges keep their original lengths.

We are interested in computing continuous shortest paths in G, and we restrict our attention to the paths between two vertices that are not interior vertices of the same edge. This is because for two interior vertices s, t of an edge (v, w), our assumption that every edge in G is the shortest path between its endpoints implies that the portion of the edge between s and t is the shortest s-t path. This motivates restricting hitting sets to subsets of V in the next definition.

Definition 11.1. The *Continuous Highway Dimension (CHD)* of $|G|$ is h if h is the smallest integer such that for all $r > 0$ and for all $v \in V(|G|)$, there exists a set $H \subseteq V$ such that $|H| \leq h$ and $H \cap P \neq \emptyset$, for any continuous shortest path P in $|G|$ with $P \cap V \neq \emptyset$, $\ell_{\bar{d}}(P) > r$, and $\bar{d}(v, P) \leq 2r$.

Here $\ell_{\bar{d}}(P)$ is the length of the continuous shortest path (geodesic) P with respect to the metric $\bar{d}$.

Next, we bound HD of G and CHD of $|G|$ in terms of each other. We start with a lemma that gives the intuition for the definitions of r-significant and (r, d)-close (to v) paths in G by showing that the latter are proxies for the continuous shortest paths in $|G|$ of lengths greater than r and within distance at most d to v.

LEMMA 11.2. *For any $v \in V$ and $r, d > 0$, if P is a shortest path that is (r, d)-close to v in G, then there is a continuous shortest path P'' in $|G|$ such that $\ell_{\bar{d}}(P'') > r$, $\mathrm{dist}(v, P'') \leq 2r$, and $P'' \cap V = P$ (i.e., all original vertices on P'' are on P).*

PROOF. By the definition of an (r, d)-close path, there is a shortest path P' in G (containing P, with potential one-hop extensions at each end) with $\ell(P') > r$ and $\mathrm{dist}(v, P') \leq 2r$. If $P' = P$, then $P'' \cap V = P'$ is the desired path. Otherwise, P' is an extension of P. Let v and w be the end vertices of P.

Suppose the extension is at one end, and let (x, v) be the extension edge. Pick a sufficiently small ϵ, for example, $\epsilon = \min(\ell(x, v), \ell(P') - r)/2$. Then the continuous shortest path P'' in $|G|$ between $(x, v, \epsilon/\ell(x, v))$ and w is the desired path.

Finally, suppose the extensions are at both ends and let (x, v) and (w, y) be the extension edges. Set $\epsilon = \min\{\ell(x, v), \ell(w, y), \ell(P'') - r\}/2$. Then the continuous shortest path P'' in $|G|$ between $(x, v, \epsilon/\ell(x, v))$ and $(w, y, 1 - \epsilon/\ell(w, y))$ is the desired path. $\square$

THEOREM 11.3. *If the HD of G is h and the CHD of $|G|$ is h', then $h \leq h' \leq 2h$.*

PROOF. To prove the first inequality, consider $v \in V$ and $r > 0$. Let $H \subseteq V$, $|H| \leq h'$ be the hitting set for all shortest paths P'' in $|G|$ such that $P'' \cap V \neq \emptyset$, $\ell(P'') > r$ and $\bar{d}(v, P'') \leq 2r$. We show that H is the hitting set for $S_r(v)$. Let P be a shortest path that is $(r, 2r)$-close to v. Let P'' be as in Lemma 11.2. Then P'' must be hit by an element x of H and by the Lemma 11.2, $x \in P$.

To prove the second inequality, consider $v \in V(|G|)$ and $r > 0$. We construct a hitting set $H \subseteq V$ for shortest paths P' in $|G|$ such that $P' \cap V \neq \emptyset$, $\ell(P') > r$, and $\mathrm{dist}(v, P') \leq 2r$ as follows.

If v corresponds to an interior point of an edge, we define $x, y \in V$ to be the endpoints of this edge. Otherwise, v corresponds to a vertex of V, and we define $x = y = v$. Since the HD of G is h, there are hitting sets H_x for $S_r(x)$ and H_y for $S_r(y)$ of size at most h each. We set $H' = H_x \cup H_y$. Clearly $|H'| \leq 2h$.

Consider a path P' in $|G|$ that we need to hit. Since $P' \cap V$ is nonempty, the path P induced by the sequence of vertices of P' belonging to V is nonempty. So P' is not contained in the interior of the continuous edge (x, y), and therefore P' is at least as close to one of x or y as it is to v. Without loss of generality, assume that $\bar{d}(v, P') \leq \bar{d}(x, P')$. Then, P' is the witness path that certifies that P is $(r, 2r)$-close to x. Therefore, H_x hits P, so H' hits P'. □

Note that the definition of CHD refers to a discrete graph by requiring the paths that are hit to have the property $P \cap V$. An alternative is to define the highway dimension of a continuous graph without referring to a discrete graph by requiring P to have at least one vertex of degree other than two (i.e., degree two vertices are "continuous" and other vertices are "discrete").

Definition 11.4. The *Extended Continuous Highway Dimension (ECHD)* of $|G|$ is h if h is the smallest integer such that for all $r > 0$ for all $v \in V(|G|)$, there exists a set $H \subseteq V$ such that (i) $|H| \leq h$ and (ii) $H \cap P \neq \emptyset$ for any continuous shortest path P in $|G|$ such that P contains a vertex of degree other than two, $\ell_{\bar{d}}(P) > r$, and $\bar{d}(v, P) \leq 2r$.

This definition is almost equivalent to Definition 11.1 since we can eliminate degree-two vertices in G as follows: if v has two neighbors (u and w), we can delete v and replace (u, v) and (v, w) by a shortcut edge (u, w) with $\ell(u, w) = \ell(u, v) + \ell(v, w)$. Note the continuous graph still has a vertex that corresponds to v, but we do not treat is as an original vertex. The only difference between the two definitions is that the first one allows such vertices in hitting sets, while the second one does not. If G has no degree-two vertices, the vertices of G correspond to vertices of $|G|$ of degree other than two, and the definitions are equivalent.

12. CONCLUDING REMARKS

We have introduced the notion of highway dimension (HD), and have shown that small HD formally guarantees good query performance for variants of several recent shortest paths algorithms for road networks (HL, TN, CH, and RE). Our results shed light on what might be the underlying reason for their remarkably good performance on road networks. When they first appeared [Abraham et al. 2010], our results suggested that HL might perform better than the other algorithms we consider (TN, CH, and RE), motivating experimental studies [Abraham et al. 2011b, 2012b] that indeed confirmed this conjecture.

Our definition of HD is motivated by the transit node (TN) algorithm [Bast et al. 2009]. The key to its efficiency is the empirically discovered fact that if a map is partitioned into regions, then for any region, all sufficiently long shortest paths originating from it can be hit by a small number of vertices. In our case, a region is roughly defined as a ball of radius $2r$ and "sufficiently long" means r-significant.

Note that the definition of HD we consider here is slightly different from the one we used in our conference papers [Abraham et al. 2010, 2011a]. The main difference is the use of r-significant paths instead of shortest paths of length greater than r. Besides being cleaner, the new definition allows us to prove Lemma 4.3, which yields a better bound for the TN algorithm. (The lemma does not hold under the old definition.) In addition, HD is a strengthening of doubling dimension under the new definition, as shown in Theorem 9.1. Under the old definition, constant highway dimension does not imply constant doubling dimension. It is easy to see that an n-vertex star graph with unit edge lengths has doubling dimension $\Omega(\log n)$ but highway dimension one under the old definition (the center vertex covers all paths of nonzero length). The new definition requires both ends of r-significant edges to be hit, and the highway dimension of the star graph becomes n.

Note that the highway dimension of a network depends not only on the underlying graph structure, but also on the length function ℓ. This is natural, as HD is a strengthening of doubling dimension, which is a property of the metric induced by ℓ.

The theory of HD is helpful for developing other shortest path algorithms and route planning services for road networks. In particular, our notion of SPHS has inspired practical algorithms for computing improved contraction orderings [Abraham et al. 2012b]. Moreover, the ideas behind HD have been used to develop new algorithms to compute alternative paths [Abraham et al. 2013] and shortest path corridors [Delling et al. 2012], as well as a new approach to computing shortest path trees that can take advantage of modern CPU and GPU architectures [Delling et al. 2013].

We have also introduced a model of road network formation that leads to constant HD networks. Although the constant we obtain is large, in a more realistic setting, discussed in Section 10, it should be much smaller. An experimental study of this model [Bauer et al. 2010b] shows that hierarchical algorithms (CH and RE) perform well on these networks, further validating our theoretical analysis.

The notion of highway dimension explains some of the behavior of routing algorithms, but the algorithms seem to perform better than our bounds imply. Although it is possible that our time and space bounds on query complexity can be improved, a recent paper [Milosavljevic 2012] shows that the $O(nh \log D)$ bound on the number of shortcuts is tight for a wide range of parameters h, n, and D. Other properties of road networks, such as the fact that they have very small separators [Delling et al. 2011b] (better than what planarity would guarantee), may also contribute to good practical performance. In fact, algorithms based only on this fact can be quite fast in practice [Delling et al. 2011a], though not as fast as those that use the shortest path structure on graphs with strong hierarchies. However, hierarchy-based algorithms tend to be more sensitive to metric changes. A related recent paper [Bauer et al. 2013] studies the theoretical performance of CH with respect to the maximum HD value over all possible metrics.

While our theoretical results apply to undirected graphs, real road networks are directed, and practical variants of the algorithms we study work very well on such inputs. One can try to extend our results to directed networks. One possible approach is to define directed balls in a natural way. Although one could define HD and SPHS in the directed case, our proof of Theorem 4.2 would not work. One way to extend the results would be to use an alternative definition of HD: a graph has HD h if h is the smallest integer for which an (r, h)-SPHS exists for all r. Another way to extend the results is to assume that the asymmetry is bounded, that is, $\text{dist}(v, w)$ and $\text{dist}(w, v)$ are within a constant factor of each other. Note, however, that for directed graphs the relationship to doubling dimension will not hold, as metrics are symmetric by definition.

Other variants of HD may prove to be interesting as well. One could, for example, use a cardinality-based definition, where instead of a balls of radius r one considers balls of cardinality R. Another possibility is to study the average-case highway dimension: at each scale r, instead of bounding the maximum r-neighborhood hitting set size, bound the average hitting set size. This definition may give a better explanation for the practical performance of various algorithms.

An interesting avenue for future research is an experimental analysis of HD (and related concepts) of real road networks. In particular, it would be useful to measure the HD (which is determined by the maximum hitting set size) for specific inputs, as well as the distribution of the hitting set sizes for different r and different centers. Unfortunately, the underlying problem of computing a minimum hitting set is NP-hard, and even our polynomial-time approximation algorithm is too slow to be practical for continental road networks. One can also attempt to derive lower bounds on the HD of real networks; a related paper [Eisner and Funke 2012] proves instance-specific lower bounds on the number of transit nodes.

ACKNOWLEDGMENTS

The authors would like to thank Pino Italiano, Haim Kaplan, Ruslan Savchenko, and Sabine Storandt for illuminating interactions and helpful suggestions. We are also grateful to the anonymous referees for their detailed comments, which greatly improved the article.

REFERENCES

Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. 2011a. VC-dimension and shortest path algorithms. In *Proceedings of the 38th International Colloquium on Automata, Languages, and Programming (ICALP'11)*, Lecture Notes in Computer Science, Vol. 6755. Springer, 690–699.

Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. 2012a. HLDB: Location-based services in databases. In *Proceedings of the 20th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (GIS'12)*. ACM Press, 339–348.

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. 2011b. A hub-based labeling algorithm for shortest paths on road networks. In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, Lecture Notes in Computer Science, Vol. 6630. Springer, 230–241.

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. 2012b. Hierarchical hub labelings for shortest paths. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA'12)*, Lecture Notes in Computer Science, Vol. 7501. Springer, 24–35.

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. 2013. Alternative routes in road networks. *ACM Journal of Experimental Algorithmics* 18, 1 (2013), 1–17.

Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato F. Werneck. 2010. Highway dimension, shortest paths, and provably efficient algorithms. In *Proceedings of the 21st Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'10)*. 782–793.

Julian Arz, Dennis Luxen, and Peter Sanders. 2013. Transit node routing reconsidered. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, Lecture Notes in Computer Science, Vol. 7933. Springer, 55–66.

Maxim Babenko, Andrew V. Goldberg, Anupam Gupta, and Viswanath Nagarajan. 2013. Algorithms for hub label optimization. In *Proceedings of the 30th ICALP*, F. V. Fomin, R. Freivalds, M. Kwiatkowska, and D. Peleg (Eds.). Lecture Notes in Computer Science, Vol. 7965, Springer, 69–80.

Holger Bast, Stefan Funke, and Domagoj Matijevic. 2009. Ultrafast shortest-path queries via transit nodes. In *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson (Eds.). DIMACS Book, Vol. 74. American Mathematical Society, 175–192.

Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. 2007. Fast routing in road networks with transit nodes. *Science* 316, 5824 (2007), 566.

Reinhard Bauer, Tobias Columbus, Ignaz Rutter, and Dorothea Wagner. 2013. Search-space size in contraction hierarchies. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP'13)*, Lecture Notes in Computer Science, Vol. 7965. Springer, 93–104.

Reinhard Bauer and Daniel Delling. 2009. SHARC: Fast and robust unidirectional routing. *ACM Journal of Experimental Algorithmics* 14, 2.4 (August 2009), 1–29. Special Section on Selected Papers from ALENEX 2008.

Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. 2010a. Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm. *ACM Journal of Experimental Algorithmics* 15, 2.3 (January 2010), 1–31. Special Section devoted to WEA'08.

Reinhard Bauer, Daniel Delling, and Dorothea Wagner. 2011. Experimental study on speed-up techniques for timetable information systems. *Networks* 57, 1 (January 2011), 38–52.

Reinhard Bauer, Marcus Krug, Sascha Meinert, and Dorothea Wagner. 2010b. Synthetic road networks. In *Proceedings of the 6th International Conference on Algorithmic Aspects in Information and Management (AAIM'10)*, Lecture Notes in Computer Science, Vol. 6124. Springer, 46–57. http://dx.doi.org/10.1007/978-3-642-14355-7_6

John F. Brock. 2012. Four surveyors of Caesar: Mapping the world! *International Federation of Surveyers* 1 (2012), 1–14.

Hervé Brönnimann and Michael T. Goodrich. 1995. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry* 14, 4 (1995), 463–479.

Vašek Chvátal. 1979. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research* 4, 3 (1979), 233–235.

Kenneth L. Clarkson. 1988. A Las Vegas algorithm for linear programming when the dimension is small. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science (FOCS'88)*. IEEE Computer Society, 452–456.

Kenneth L. Clarkson. 1993. Algorithms for polytope covering and approximation. In *Proceedings of the 3rd International Workshop on Algorithms and Data Structures (WADS'93)*, Lecture Notes in Computer Science, Vol. 709. Springer, 246–252.

Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. 2003. Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing* 32, 5 (2003), 1338–1355.

Daniel Delling, Andrew V. Goldberg, Andreas Nowatzyk, and Renato F. Werneck. 2013. PHAST: Hardware-accelerated shortest path trees. *Journal of Parallel and Distributed Computing* 73, 7 (2013), 940–952.

Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. 2011a. Customizable route planning. In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*, Lecture Notes in Computer Science, Vol. 6630. Springer, 376–387.

Daniel Delling, Andrew V. Goldberg, Ilya Razenshteyn, and Renato F. Werneck. 2011b. Graph partitioning with natural cuts. In *Proceedings of the 25th International Parallel and Distributed Processing Symposium (IPDPS'11)*. IEEE Computer Society, 1135–1146.

Daniel Delling, Moritz Kobitzsch, Dennis Luxen, and Renato F. Werneck. 2012. Robust mobile route planning with limited connectivity. In *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*. SIAM, 150–159.

Daniel Delling and Renato F. Werneck. 2013. Faster customization of road networks. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, Lecture Notes in Computer Science, Vol. 7933. Springer, 30–42.

Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson (Eds.). 2009. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*. DIMACS Book, Vol. 74. American Mathematical Society.

Eric V. Denardo and Bennett L. Fox. 1979. Shortest-route methods: 1. Reaching, pruning, and buckets. *Operations Research* 27, 1 (1979), 161–186.

Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* 1 (1959), 269–271.

Jochen Eisner and Stefan Funke. 2012. Transit nodes—Lower bounds and refined construction. In *Proceedings of the 14th International Workshop on Algorithm Engineering and Experiments*. 141–149.

David Eppstein and Michael T. Goodrich. 2008. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'08)*. ACM, New York, NY, Article 16, 10 pages.

Guy Even, Dror Rawitz, and Shimon Shahar. 2005. Hitting sets when the VC-dimension is small. *Information Processing Letters* 95, 2 (2005), 358–362.

Michael L. Fredman and Robert E. Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM* 34, 3 (July 1987), 596–615.

Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. 2004. Distance labeling in graphs. *Journal of Algorithms* 53 (2004), 85–112.

Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. 2012. Exact routing in large road networks using contraction hierarchies. *Transportation Science* 46, 3 (August 2012), 388–404.

Andrew V. Goldberg. 2008. A practical shortest path algorithm with linear expected time. *SIAM Journal on Computing* 37 (2008), 1637–1655.

Andrew V. Goldberg and Chris Harrelson. 2005. Computing the shortest path: A* search meets graph theory. In *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'05)*. SIAM, 156–165.

Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. 2009. Reach for A*: Shortest path algorithms with preprocessing. In *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson (Eds.). DIMACS Book, Vol. 74. American Mathematical Society, 93–139.

Lee-Ad Gottlieb and Liam Roditty. 2008. An optimal dynamic spanner for doubling metric spaces. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA'08)*, Lecture Notes in Computer Science, Vol. 5193. Springer, 478–489.

A. Gupta, R. Krauthgamer, and J. R. Lee. 2003. Bounded geometries, fractals, and low-distortion embeddings. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 534–543.

Ronald J. Gutman. 2004. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX'04)*. SIAM, 100–111.

Moritz Hilger, Ekkehard Köhler, Rolf H. Möhring, and Heiko Schilling. 2009. Fast point-to-point shortest path computations with arc-flags. In *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson (Eds.). DIMACS Book, Vol. 74. American Mathematical Society, 41–72.

Martin Holzer, Frank Schulz, and Dorothea Wagner. 2008. Engineering multi-level overlay graphs for shortest-path queries. *ACM Journal of Experimental Algorithmics* 13, 2.5 (December 2008), 1–26.

David S. Johnson. 1974. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences* 9, 3 (1974), 256–278.

J. M. Kleinberg, A. Slivkins, and T. Wexler. 2009. Triangulation and embedding using small sets of beacons. *Journal of the Association for Computing Machinery* 56, 6 (2009).

Jon M. Kleinberg. 2000. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing (STOC'00)*. 163–170.

Jon M. Kleinberg. 2003. Detecting a network failure. *Internet Mathematics* 1, 1 (2003), 37–55.

Robert Krauthgamer and Tim Roughgarden. 2011. Metric clustering via consistent labeling. *Theory of Computing* 7, 1 (2011), 49–74.

Ulrich Lauther. 2009. An experimental evaluation of point-to-point shortest path calculation on roadnetworks with precalculated edge-flags. In *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson (Eds.). DIMACS Book, Vol. 74. American Mathematical Society, 19–40.

Stanley Milgram. 1967. The small world problem. *Psychology Today* 1, 1 (1967), 60–67.

Nikola Milosavljevic. 2012. On optimal preprocessing for contraction hierarchies. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS)*. Paper 6.

David Peleg. 2000. Proximity-preserving labeling schemes. *Journal of Graph Theory* 33, 3 (2000), 167–176.

Ira Pohl. 1971. Bi-directional search. In *Machine Intelligence*. Vol. 6. Edinburgh University Press, Edinburgh, 124–140.

PTV AG - Planung Transport Verkehr. 1979. http://www.ptv.de. (1979).

Peter Sanders and Dominik Schultes. 2009. Robust, almost constant time shortest-path queries in road networks. In *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson (Eds.). DIMACS Book, Vol. 74. American Mathematical Society, 193–218.

Peter Sanders and Dominik Schultes. 2012. Engineering highway hierarchies. *ACM Journal of Experimental Algorithmics* 17, 1 (2012), 1–40.

Frank Schulz, Dorothea Wagner, and Karsten Weihe. 2000. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. *ACM Journal of Experimental Algorithmics* 5, 12 (2000), 1–23.

M. B. Smyth. 1995. Semi-metrics, closure spaces and digital topology. *Theoretical Computer Science* 151, 1 (1995), 257–276.

Yufei Tao, Cheng Sheng, and Jian Pei. 2011. On k-skip shortest paths. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data (SIGMOD'11)*. ACM, New York, NY, 421–432.

Robert Tarjan. 1983. *Data Structures and Network Algorithms*. SIAM.

Vladimir N. Vapnik and Alexey Ya. Chervonenkis. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications* 16, 2 (1971), 264–280.

Received September 2013; revised July 2016; accepted August 2016