Shortest-Path Queries in Static Networks

CHRISTIAN SOMMER, MIT, CSAIL

We consider the *point-to-point (approximate) shortest-path query problem*, which is the following generalization of the classical *single-source (SSSP)* and *all-pairs shortest-path (APSP)* problems: we are first presented with a *network* (*graph*). A so-called *preprocessing* algorithm may compute certain information (a *data structure* or *index*) to prepare for the next phase. After this preprocessing step, applications may ask shortest-path or distance *queries*, which should be answered as fast as possible.

Due to its many applications in areas such as transportation, networking, and social science, this problem has been considered by researchers from various communities (sometimes under different names): algorithm engineers construct fast *route planning* methods; database and information systems researchers investigate *materialization tradeoffs*, query processing on *spatial networks*, and *reachability queries*; and theoretical computer scientists analyze *distance oracles* and *sparse spanners*. Related problems are considered for *compact routing* and *distance labeling* schemes in networking and distributed computing and for *metric embeddings* in geometry as well.

In this survey, we review selected approaches, algorithms, and results on shortest-path queries from these fields, with the main focus lying on the tradeoff between the index size and the query time. We survey methods for general graphs as well as specialized methods for restricted graph classes, in particular for those classes with arguable practical significance such as planar graphs and complex networks.

Categories and Subject Descriptors: G.2.2 [**Discrete Mathematics**]: Graph Theory; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Shortest path, shortest-path query, distance oracle

ACM Reference Format:
Christian Sommer. 2014. Shortest-path queries in static networks. ACM Comput. Surv. 46, 4, Article 45 (March 2014), 31 pages.
DOI: http://dx.doi.org/10.1145/2530531

1. INTRODUCTION

We review research on algorithms for the *point-to-point (approximate) shortest-path query problem*, restricted to discrete, static graphs with nonnegative edge lengths (also called weights or costs). The only criterion on the optimality of a path shall be its length, which is defined as the sum over all the edges on the path of their lengths. Edge and path lengths can be used to represent various quantities such as travel times, ticket prices, or fuel costs.

The shortest-path problem in general has countless applications; the shortest-path *query* problem in particular occurs in applications such as route planning and navigation [Zaroliagis 2008; Goldberg et al. 2009; Delling et al. 2009a], geographic information systems (GISs) and intelligent transportation systems [Jing et al. 1996], logistics,

Authors' addresses: Christian Sommer; email: csom@csail.mit.edu.

DOI: http://dx.doi.org/10.1145/2530531

traffic simulations [Ziliaskopoulos et al. 1997; Barrett et al. 2002; Raney and Nagel 2004; Baker and Gokhale 2007], computer games [Stout 1999; Bulitko et al. 2010], server selection [Ng and Zhang 2002; Dabek et al. 2004; Costa et al. 2004; Shavitt and Tankel 2008; Eriksson et al. 2009], XML indexing [Schenkel et al. 2004, 2005], proximity search in databases [Goldman et al. 1998], reachability in object databases [Butterworth et al. 1991], packet routing [Schwartz and Stern 1980], metabolic networks [Rahman et al. 2005], causal regulatory networks [Chindelevitch et al. 2011], web search ranking [Vieira et al. 2007], and path finding in social networks [Karinthy 1929; Milgram 1967; Kleinberg 2000; Newman 2001]. See also Santos [2009].

The shortest-path query problem is different from the classical *single-source (SSSP)* and *all-pairs shortest path (APSP)* problems in that there are two stages: *preprocessing* and answering *queries*. We are first presented with a *network* (also termed *graph*). A so-called *preprocessing* algorithm may compute certain information (a *data structure* or *index*, in the theory community referred to as a *distance oracle* [Thorup and Zwick 2005]) to prepare for the second phase. After this preprocessing step, applications may ask *queries*, which should be answered efficiently. In computational geometry, this and similar problems are sometimes called *repetitive-mode* (as opposed to *single-shot*) problems [Preparata and Shamos 1985, p. 37].

A lazy solution to the shortest-path query problem is not to precompute any data structure at all but to use an SSSP algorithm [Dijkstra 1959; Fredman and Tarjan 1987] to answer queries. Answering a query then requires time roughly linear in the network size. An eager solution is to precompute the results for all possible queries using an APSP algorithm [Floyd 1962; Warshall 1962; Johnson 1977]. We assume no knowledge about the query distribution. In practice, an application designer may potentially take advantage of different frequencies for user queries, in particular if certain pairs are queried significantly more often than others, or if some pairs are expected to never be queried at all. For example, in a route planning system, one might assume that for most user queries origin and destination are within a few hundred miles (while the maximum distance might be a few thousand miles). Both solutions have their advantages and disadvantages: for the SSSP strategy, no preprocessing is necessary but the query processing is rather slow. For the APSP strategy, the query execution is extremely fast: one table lookup suffices to obtain the shortest-path distance, but the preprocessing step is expensive and the space consumption is prohibitively large for many real-world networks, spanning millions or even billions of nodes.

In the shortest-path query scenario, we mediate between these two extremes; that is, we analyze the *tradeoff* between space, preprocessing time, and query time. If the query algorithm is allowed to return an approximate shortest path, the worst-case *accuracy* (often called *stretch*) is also an important factor of the tradeoff.

Designing a shortest-path query processing method raises questions such as: How can these data structures be computed efficiently? What amount of storage space is necessary? How much improvement of the query time is possible? How good is the approximation quality of the query result? What are the tradeoffs between precomputation time, space, query time, and approximation quality?

In this survey, we focus on the tradeoff between space and query time. In the first part, we survey theoretical results on distance oracles for general graphs (Section 2). In the second part, we consider two application scenarios and the corresponding graph classes, namely, (i) distance oracles for planar graphs, motivated by route planning problems for road networks (Section 3), and (ii) distance oracles for complex networks, motivated by practical problems in online social networks, web search, computer networking, computational biology, and social science (Section 4).

Recent related surveys include the following. Sen [2009] surveys distance oracles for general graphs with a special focus on fast preprocessing. Zwick [2001] surveys exact and approximate shortest-path algorithms. Gavoille [2003] surveys similar tradeoffs for routing problems. Delling et al. [2009a], Goldberg [2007] (see also Goldberg et al. [2009]); and Wagner and Willhalm [2007] survey route-planning methods (see also Fu, Sun, and Rilett [2006] for heuristics). Bast [2009, Section 3] surveys the "tricks of the trade" for fast routing on transportation networks. The related topic of *materialization* tradeoffs is considered by Agrawal and Jagadish [1989] and Shekhar et al. [1997]. This survey is largely based on the author's Ph.D. thesis [Sommer 2010, Chapter 3].[1]

1.1. Problem Specification

Thorup and Zwick [2005] coined the term *distance oracle*, which is a data structure that, after preprocessing a graph $G = (V, E)$, allows for efficient (approximate) distance and shortest-path queries. Let ℓ denote the *edge length* function $\ell : E \to \mathbb{R}^+$, which we assume to be nonnegative, that is, $\forall e \in E : \ell(e) \geqslant 0$.

Definition 1.1. An *((α, β)–approximate) distance oracle* for a class of graphs $\mathcal{G}$ consists of a data structure and a query algorithm.

—The *preprocessing time* is the worst-case time required to construct the data structure $S(G)$ for any $G \in \mathcal{G}$. For randomized preprocessing algorithms, the preprocessing time is, as usual, defined as the maximum over all $G \in \mathcal{G}$ of the expected preprocessing time for G.
—The *space complexity* refers to the worst-case size of the data structure for any $G \in \mathcal{G}$.

After preprocessing $G = (V, E)$, the data structure S (which depends on G) supports *(approximate) distance queries* for all pairs of nodes $u, v \in V$, returning a value $\tilde{d}_S(u, v)$. The query algorithm and its result are characterized as follows.

—The *query time* is the worst-case time required to compute $\tilde{d}_S(u, v)$ among all $G \in \mathcal{G}$ and $u, v \in V$.
—A distance oracle S is said to have *stretch* (α, β) with $\alpha \geqslant 1$ and $\beta \geqslant 0$ if for all $G \in \mathcal{G}$ and $u, v \in V$, its query algorithm satisfies

$$d_G(u, v) \leqslant \tilde{d}_S(u, v) \leqslant \alpha \cdot d_G(u, v) + \beta,$$

where $d_G(u, v)$ denotes the shortest-path distance from u to v in G. The stretch is also called *distortion*.

In addition to the worst-case measures, the average or expected query time and stretch may also be of interest. If not explicitly stated otherwise, in this survey, *stretch* means the worst-case stretch. Note that additive stretch $\beta > 0$ is most meaningful for unit-length graphs ($\forall e \in E : \ell(e) = 1$); for more general length functions, we usually have $\beta = 0$ and stretch means multiplicative stretch ($\alpha \geqslant 1$).

Let us emphasize that the query time corresponds to the time to compute the shortest-path *distance* (or an estimate thereof), as opposed to an actual path. For many efficient data structures, the time to actually report a shortest path is dominated by the time required to explicitly output each edge. Obviously, this time must at least be proportional to the number of edges on the path, making comparisons somewhat more difficult. For most methods, after having computed the distance, there is an implicit representation of the path such that each edge can be output efficiently (often in constant time). In the following, query times correspond to the times required to compute (approximate) distances.

[1]This survey has been modified from its original version. It has been formatted to fit this journal's page limit and edited for content. Several references had to be removed for brevity; they can be found in Sommer [2010].

An *(approximate) distance labeling scheme* [Peleg 2000; Gavoille et al. 2004] (inspired by *adjacency labels* [Kannan et al. 1992]) can be thought of as the *distributed* version of a distance oracle. The data structure is distributed among the nodes such that each node u is assigned a *label* $\mathcal{L}(u)$. The space complexity is defined as the maximum label length (the average label length is also of interest). At query time, the algorithm is given *only* the two labels $\mathcal{L}(s)$, $\mathcal{L}(t)$ of the query nodes s, t, respectively, using which it must compute (an estimate of) $d_G(s, t)$.

1.2. General Techniques

On a high level, many methods use some concept of *landmarks*, *portals*, *hubs*, *beacons*, *seeds*, or *transit nodes*, each corresponding to a set of carefully selected points (often a subset $L \subseteq V$ of the node set), which represent (potentially approximate) shortest paths. In the following, we refer to such nodes as *landmarks*. We distinguish three typical degrees of freedom:

(1) *Global Landmark Selection*: different methods use different selection strategies to choose a *global* set of landmarks $L \subseteq V$. Popular strategies include (i) random sampling, (ii) high-degree nodes, and (iii) nodes on *separators*. Depending on the shortest-path metric (defined by the length/weight/cost function ℓ), there is also a strategy to (iv) choose those nodes that lie on shortest paths of certain lengths (particularly effective for road networks with edge lengths corresponding to estimates of driving times).

(2) *Local Landmark Selection*: each node $u \in V$ is *connected* to certain landmarks (potentially to all), which usually means that u stores the shortest-path distance to its landmark set $L(u) \subseteq L$. For some methods, there is one distinguished landmark $l_u \in L$ associated with each node u. Usually, l_u is chosen as a nearest landmark.

(3) *Distances Among / From / To Landmarks*: methods may differ in how they represent distances among landmarks (sometimes only a subset of all the $L \times L$ distances is represented) and from nodes to landmarks (and vice versa). In general, methods store *stars* (representing SSSP distances from one node to a subset of nodes) and *cliques* (representing APSP distances among a subset of nodes), which are of course unions of stars. When we say that oracle constructions mediate between SSSP and APSP, it not only is a “global” analogy but often also corresponds to “local” decisions on where to use SSSP and where to use APSP.

A potential fourth degree of freedom is to use multiple levels or recursion. For example, many methods use landmarks at various *scales*.

Furthermore, almost always when a partial execution of an SSSP algorithm is involved, the designers may choose between (i) computing the shortest-path tree at preprocessing time and storing it (as a star) or (ii) computing the tree at query time, thereby mediating between space and query time in a rather straightforward way.

2. THEORETICAL RESULTS ON DISTANCE ORACLES FOR GENERAL GRAPHS

For distance oracles applicable to general graphs, the quantitative tradeoff between the space requirements and the approximation quality (stretch) is known up to constant factors. For distance oracles that take advantage of the properties of certain classes of graphs, however, the tradeoff is less well understood: for some classes of sparse graphs such as planar graphs, there are data structures that enable query algorithms to efficiently compute distance estimates of much higher precision than what the tradeoff for general graphs would predict.

In the following, we summarize the known theoretical results for general graphs, with space complexity lower bounds in Section 2.1 and algorithmic upper bounds in Section 2.2.

Table I. Space Lower Bounds of Distance Oracles for Graphs on n Nodes and m Edges, up to Polylogarithmic Factors
The table is supposed to be read and interpreted as follows: the lower bound on the space (leftmost column) holds if the conditions on stretch and query time in the second column are met, potentially with further assumptions on the model or conjectures (third column).

Lower Bound	Condition(s)			
Space	Stretch	Query	Assumption / Proof	Reference
$\Omega(\min\{m, n^{1+1/k}\})$	$\alpha < 2k+1$		girth conjecture [Erdös 1964]	[Thorup and Zwick 2005]
$n^{1+\Omega(1/(\alpha t))}$	α	t	cell-probe model [Yao 1981]	[Sommer et al. 2009]
$\Omega(n^2)$	$\alpha < 2$	$t = O(1)$	set intersection, conjecture	[Cohen and Porat 2010]
$\tilde{\Omega}(n^{1+1/(2-1/\ell)})$	$\alpha = 3-2/\ell$	$t = O(1)$	set intersection, conjecture	[Patrascu et al. 2012]
$\Omega(n^{3/2})$	$\alpha = 1$		distributed labels	[Gavoille et al. 2004]

2.1. Lower Bounds on the Space of Distance Oracles

Known lower bounds on the space requirements of distance oracles are listed in Table I. In the following, we consider lower bounds for *undirected graphs*, which extend to directed graphs. However, for directed graphs, even stronger lower bounds are known: note that any distance oracle for directed graphs with arbitrary finite stretch can also answer *reachability* queries, for which no worst-case efficient data structure is known [Ajtai and Fagin 1990; Patrascu 2011]. The results for *undirected* graphs can be summarized as follows.

—The lower bound of Thorup and Zwick [2005, Proposition 5.1.] (see also Matousek [1996] for a similar construction) establishes that, for general graphs, the tradeoffs between space, stretch, and query time of existing distance oracles (Section 2.2) are essentially best possible. More precisely, they prove that any distance oracle with multiplicative stretch $\alpha < 2k + 1$ must use space $\Omega(n^{1+1/k})$, assuming Erdős's *Girth Conjecture* [1964], which is widely believed and partially proven. The *girth* of a graph is defined as the length of its shortest cycle; the conjecture says that dense graphs with large girth exist (see Thorup and Zwick [2005, Table II] for an overview of results on the Girth Conjecture).
—For sparse graphs, the situation is less clear. Distance oracles with constant stretch and query time require superlinear space [Sommer et al. 2009]. Higher space lower bounds have been given based on the assumption that constant-time *set intersection* queries require quadratic space: for multiplicative stretch $\alpha < 2$, quadratic space is required [Cohen and Porat 2010; Patrascu and Roditty 2010]; for multiplicative stretch $\alpha < 3$, in particular for stretch $\alpha = 3 - 2/\ell$, space $\tilde{\Omega}(n^{1+1/(2-1/\ell)})$ is required [Patrascu et al. 2012].[2]
—For special classes of graphs such as planar, bounded-genus, minor-free, and bounded-doubling-dimension graphs, I do not know of nontrivial lower bounds.

Lower Bounds on the Lengths of Distance Labels. Since distance labeling schemes are in some sense distributed distance oracles, space lower bounds on oracles extend to bounds on label lengths in a straightforward way. However, lower bounds on label lengths can be higher, since the query algorithm is allowed to access only two labels (as opposed to an entire data structure). Several such lower bounds are due to Gavoille, Peleg, Pérennes, and Raz [2004] (see also Section 3.1.1). One of their lower bounds says that, even for graphs with maximum degree 3, exact distance labels require total label length $\Omega(n^{3/2})$ [Gavoille et al. 2004, Theorem 3.7].

[2]Asymptotic notation as in $\tilde{O}(\cdot)$ or $\tilde{\Omega}(\cdot)$ hides polylogarithmic factors in the number of nodes n.

Table II. Time and Space Complexities of Distance Oracles for General, Undirected Graphs, Sorted by Stretch (More Precisely, by the Minimum Stretch Possible)
For the oracles in the lower part, the bounds apply for unit-length graphs only and the stretch often also involves an additive term β. Approximate distance oracles are included only if the space requirement is at most $o(n^2)$.

Preprocessing	Space	Query Time	Stretch (α, β)	Reference
$O(mn)$	$O(n^2)$	$O(1)$	exact	APSP
none	$O(m)$	$O(m + n \lg n)$	exact	SSSP
$\tilde{O}(mn/\sigma)$	$\tilde{O}(m + n^2/\sigma)$	$O((\sigma m/n)^{t})$	$\alpha = 1 + 1/t$	[Agarwal 2013], $\sigma \in [1, n]$
$\tilde{O}(mn/\sigma)$	$O(m + n^2/\sigma)$	$O((2\sigma m/n)^{t})$	$\alpha = 1+2/(t+1)$	[Agarwal and Godfrey 2013], $\sigma \in [1, n]$
$O(\text{poly}(n))$	$O(m^{1/3}n^{4/3})$	$O(m^{1/3}n^{1/3})$	$\alpha = 1 + 2/3$	[Fakcharoenphol and Saranurak 2010]
$O(\text{poly}(n))$	$O(m^{1/3}n^{4/3})$	$O(1)$	$\alpha = 2$	[Patrascu and Roditty 2010]
$O(\text{poly}(n))$	$O(m + m^{1-\theta}n)$	$O(m^{\theta})$	$\alpha = 2$	[Agarwal et al. 2011], $\theta \in (0, 1/2]$
$\tilde{O}(mn^{1/k})$	$O(kn^{1+1/k})$	$O(1)$	$\alpha = 2k - 1$	[Chechik 2013]
$\tilde{O}(m+n^{1+c/\sqrt{k}})$	$O(kn^{1+1/k})$	$O(\lg k)$	$\alpha = 2k - 1$	[Wulff-Nilsen 2012, 2013], fixed $c = \Theta(1)$
$\tilde{O}(mn^{1/k})$	$O(kn^{1+1/k})$	$O(k)$	$\alpha = 2k - 1$	[Thorup and Zwick 2005; Roditty et al. 2005]
$\tilde{O}(n^2)$	$O(n^{3/2})$	$\Theta(\lg n)$	$\alpha = 3$	[Baswana and Kavitha 2010]
$O(\text{poly}(n))$	$O(m)$	$O(m^{1/2})$	$\alpha = 3$	[Patrascu and Roditty 2010], implicit
$O(\text{poly}(n))$	$O(m)$	$O(m^{1/(k+1)})$	$\alpha = 4k - 1$	[Agarwal et al. 2011]
$O(\text{poly}(n))$	$O(m+m^{(1-\theta)(1+1/k)})$	$O(m^{\theta})$	$\alpha = 4k - 1$	[Agarwal et al. 2011], $\theta \in (0, 1)$
$\tilde{O}(n^2)$	$O(kn^{1+1/k})$	$O(k)$	$\alpha = 2k - 1$	[Baswana and Kavitha 2010], $k \geqslant 3$
$O(\text{poly}(n))$	$\tilde{O}(n^{1+1/(k\pm1/\ell)})$	$O(k + \ell)$	$\alpha = 2k-1\pm2/\ell$	[Patrascu et al. 2012], sparse graphs
$O(mn)$	$O(nm^{1-\epsilon/6})$	$O(m^{1-\epsilon/6})$	$(1+\epsilon, 0)$	[Porat and Roditty 2013], $\epsilon > 0$
$O(\text{poly}(n))$	$O(n^{5/3})$	$O(1)$	$(2, 1)$	[Patrascu and Roditty 2010]
$O(\text{poly}(n))$	$\tilde{O}(n^{1+2/(2k-1)})$	$O(k)$	$(2k - 2, 1)$	[Abraham and Gavoille 2011], $k \geqslant 2$
$O(n^2)$	$O(kn^{1+1/k})$	$O(k)$	$(2k - 1, 0)$	[Baswana and Sen 2006]
$O(m + n^{23/12})$	$O(n^{3/2})$	$O(1)$	$(3, 10)$	[Baswana et al. 2008], similar for $\alpha > 3$

Preprocessing Time. There is also a connection between boolean matrix multiplication and distance oracles with multiplicative stretch $\alpha < 2$ or additive stretch $\beta = O(1)$. Dor, Halperin, and Zwick [2000, Theorem 5.1] provide a reduction between approximate APSP and matrix multiplication: if the preprocessing algorithm is faster than the time required for matrix multiplication, then query time $o(m/n)$ would imply a faster algorithm for boolean matrix multiplication.

2.2. Distance Oracles for General Graphs

For an overview of distance oracles for general *undirected* graphs, see Table II. For the stretch versus space tradeoff, see Figure 1.

Many distance oracles approximate distances by *triangulation* using a sublinear number of *landmarks* (also termed *beacons*), selected by random sampling. Nodes store distances to all landmarks $l \in L$ (stars from all landmarks) and query results are computed as $\min_{l \in L} d(s, l) + d(l, t)$. If any landmark $l \in L$ lies on a shortest path from s to t, the exact distance can be recovered. However, most schemes do not guarantee that shortest distances are retrieved. Furthermore, instead of minimizing over all

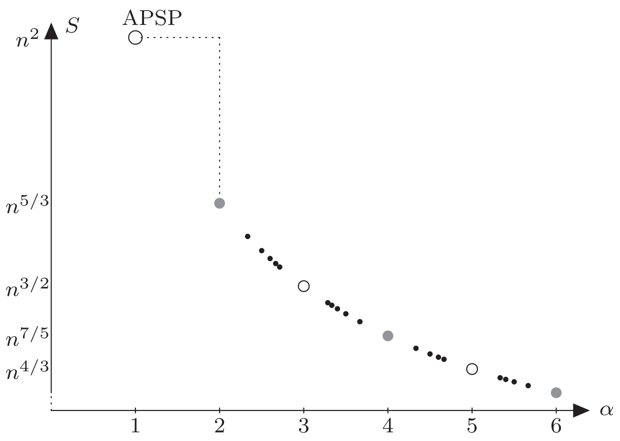

Fig. 1. Distance oracles (with $O(1)$ query time) for sparse graphs (n nodes, $m = \tilde{O}(n)$ edges): the tradeoff between stretch [α] and space [S], depicted using a linear scale for the stretch and a logarithmic scale for the space. Oracles with odd integral stretch (white circles) are due to Thorup and Zwick [2005]. Oracles with even integral stretch (gray circles) are due to Pătraşcu and Roditty [2010] and Abraham and Gavoille [2011]. Oracles in between (black dots) are due to Pătraşcu, Roditty, and Thorup [2012]. Their results say that, for many stretch values $\alpha \geqslant 2$, there is a distance oracle using space $S = O(n^{1+2/(\alpha+1)})$.

landmarks (in time $O(|L|)$), in many schemes, nodes designate a nearest landmark for triangulation. Let l_s (l_t) denote a landmark that is closest to s (t). Then, the result is simply $\min\{d(s,l_s) + d(l_s,t), d(s,l_t) + d(l_t,t)\}$.

The approximation obtained by triangulation can be rather inaccurate if s and t are close to each other. Triangulation may incur a detour that is arbitrarily large with respect to the shortest-path distance. The approximate distance oracles described in the following differ mainly in their handling of short distances.

Odd Integral Stretch. In a seminal work, Thorup and Zwick [2005] provide both the lower and the matching upper bound: for any integer $k \geqslant 1$, there is a distance oracle using space $O(kn^{1+1/k})$ with stretch $\alpha = 2k-1$ and query time $O(k)$. For their oracles, the tradeoff between space complexity and stretch is essentially tight. Note that for $k = 1$, this yields an exact distance oracle with $\tilde{O}(mn)$ preprocessing time, which essentially corresponds to APSP [Zwick 2001].

The relationship between size and stretch is almost optimal with respect to the lower bound implied by the girth conjecture (for earlier results, see Awerbuch et al. [1998], Cohen [1998], and Matousek [1996]). The time complexities (query and preprocessing) are not tight; both have been improved upon independently. Chechik [2013], based on earlier work of Mendel and Naor [2007] (see also Mendel and Schwob [2009]) and Wulff-Nilsen [2013], reduces the query time down to $O(1)$. Baswana and Kavitha [2010], Baswana and Sen [2006], and Baswana, Gaur, Sen, and Upadhyay [2008] improve the performance of the preprocessing algorithms to quadratic and even subquadratic time. Wulff-Nilsen [2012] proves that the distance oracle can be computed in time almost proportional to the space requirements, obtaining an oracle that is optimal for sufficiently dense graphs.

For $k = 2$ (multiplicative stretch $\alpha = 3$), the Thorup–Zwick distance oracle works as follows: landmarks are chosen independently at random with probability $p := 1/\sqrt{n}$ (see Roditty et al. [2005] for deterministic landmark selection). Storing stars from all np landmarks (in expectation) requires space proportional to $n^2 p$. Node pairs at short distances are handled by precomputing and storing shortest-path trees and distances for *open balls*. Each node u (with nearest landmark l_u) stores distances to all nodes v

within distance strictly less than $d(u, l_u)$, that is, to all the nodes in the open ball $B(u) = \{v \in V : d(u, v) < d(u, l_u)\}$. The expected ball size is $1/p$, which yields expected total space requirements of $n^2 p + n/p = O(n\sqrt{n})$. Given a pair of nodes (s, t) at query time, the algorithm checks whether $s \in B(t)$ or $t \in B(s)$, in which case the exact distance has been precomputed and can be returned. Otherwise, the error introduced by triangulation can be bounded using the triangle inequality: $d(s, l_t) + d(l_t, t) \leqslant d(s, t) + d(t, l_t) + d(l_t, t) \leqslant 3d(s, t)$.

For $k > 2$, landmarks are selected at k levels: at level i, landmarks are sampled with probability $p := n^{-i/k}$. Balls at higher levels are defined to contain only landmarks of the level below; hence, balls at each level have expected size proportional to $n^{1/k}$. The query algorithm alternates between landmarks of source and target. For details, see Thorup and Zwick [2005] and Wulff-Nilsen [2013].

Even Integral Stretch. Pătraşcu and Roditty [2010] (see also Agarwal et al. [2011]) observe that the worst-case stretch is attained when neither $s \in B(t)$ nor $t \in B(s)$, but *almost*, meaning that the *balls intersect*, that is, $B(s) \cap B(t) \neq \emptyset$. They prove that, for unit-length graphs, there is a $(2, 1)$–approximate distance oracle using space $O(n^{5/3})$. The tradeoff extends to weighted sparse graphs and to general $k \geqslant 2$ with stretch $(2k - 2, 1)$ and space $\tilde{O}(n^{1+2/(2k-1)})$ [Abraham and Gavoille 2011].

In the following, we briefly describe the simplified construction of Abraham and Gavoille [2011], building on Patrascu and Roditty [2010]. Let the *cluster* of a node v contain all the nodes u that have v in their balls, that is, $C(v) := \{u : v \in B(u)\}$. An algorithm that carefully resamples landmarks can find a set of landmarks of size $|L| = \tilde{O}(n^{2/3})$ such that, for all $v \in V$, both $|B(v)| = \tilde{O}(n^{1/3})$ and $|C(v)| = \tilde{O}(n^{1/3})$ [Thorup and Zwick 2001]. Then, each node v stores distances to landmarks, to all nodes in its ball $u \in B(v)$, and to all nodes whose ball has a nonempty intersection with $B(v)$ (bounded by $\tilde{O}(n^{2/3})$).

The stretch $(2, 1)$ bound is proven as follows. If $s \in B(t)$, $t \in B(s)$, or $B(s) \cap B(t) \neq \emptyset$, the exact distance has been precomputed and can be returned. Otherwise, since $B(s) \cap B(t) = \emptyset$, and since the open ball $B(u)$ has radius $d(u, l_u) - 1$,

$$\begin{aligned} (d(s, l_s) - 1) + (d(t, l_t) - 1) &< d(s, t) \\ d(s, l_s) + d(t, l_t) &\leqslant d(s, t) + 1. \end{aligned}$$

Without loss of generality, let us assume that the radius of $B(t)$ is at most the radius of $B(s)$, which is equivalent to $d(t, l_t) \leqslant d(s, l_s)$. Therefore,

$$d(t, l_t) \leqslant (d(s, t) + 1)/2.$$

Then $d(s, l_t) + d(l_t, t) \leqslant d(s, t) + 2d(l_t, t) \leqslant 2d(s, t) + 1$.

Sparse Graphs. The Pătraşcu–Roditty result extends to graphs with general nonnegative edge lengths and the additive term $\beta = 1$ in the stretch can be avoided: for graphs on m edges, there is a distance oracle using space $O(m^{1/3} n^{4/3})$ with stretch $\alpha = 2$ and query time $O(1)$ [Patrascu and Roditty 2010]. More generally, for many stretch values between $\alpha = 2$ and $\alpha = 3$, there is a distance oracle using space $\tilde{O}(m^{1+1/(k \pm 1/\ell)})$ with stretch $\alpha = 2k - 1 \pm 2/\ell$ [Patrascu et al. 2012]. For an illustration of the tradeoff, see Figure 1.

For multiplicative stretch $\alpha < 2$, oracles with subquadratic space and constant query time are unlikely to exist (Section 2.1). However, for sparse graphs, oracles with subquadratic space and sublinear query time have been found [Fakcharoenphol and Saranurak 2010; Porat and Roditty 2013; Agarwal and Godfrey 2013]. Agarwal and Godfrey [2013] provide smooth tradeoffs between query time and stretch and also between query time and space. Further improvements have been announced [Agarwal 2013].

The oracle of Thorup and Zwick [2005] achieves $\tilde{O}(n)$ space for $k = \lg n$ with $O(\lg n)$ multiplicative stretch and $O(\lg n)$ query time. The oracle of Chechik [2013] improves the query time to $O(1)$. It would be interesting to reduce the stretch to $O(1)$ instead. The girth-based space lower bound proves that this is impossible for dense graphs. It is an open question whether such oracles exist for sparse graphs (Agarwal et al. [2011] refer to such oracles as *the holy grail*).

Exact Distances. An exact distance oracle for general graphs (which also works for directed graphs) with a different type of worst-case guarantee on the space and query complexities is due to Cohen, Halperin, Kaplan, and Zwick [2003]. Their *two-hop cover* is a distance labeling scheme, which works as follows. Each node u precomputes and stores distances to a set of landmarks $L(u)$ such that, for any pair of nodes s, t, at least one node on a shortest $s - t$ path is in $L(s) \cap L(t)$ (each shortest path is *covered* by a landmark). The query algorithm simply returns the best distance using a landmark $l \in L(s) \cap L(t)$. There is no absolute guarantee on the size of $L(\cdot)$; however, their polynomial-time preprocessing algorithm returns an $O(\lg n)$–approximation for the cover with minimum total size (*cf.* SetCover for the set of all shortest paths). Babenko, Goldberg, Gupta, and Nagarajan [2013] provide an $O(\lg n)$–approximation algorithm for the cover minimizing the maximum label size (and more general objective functions), thereby minimizing the worst-case query time (up to a log factor); two-hop covers have been implemented, engineered, and evaluated for road networks (see Section 3.2.2) and also for more complex networks (see Section 4).

3. THEORY AND PRACTICE OF ROUTE PLANNING FOR ROAD NETWORKS

Efficiently finding "good" routes in transportation networks is arguably the main application scenario for shortest-path query methods. Due to its immediate practical implications, this scenario stimulated a large body of research.

As mentioned in the introduction, we assume that all relevant information is incorporated into the network and the length function. In the following, we shall not attempt to cover various modeling aspects, despite their practical importance. Let us just briefly mention that some methods discussed in this section (e.g., [Holzer et al. 2008; Delling et al. 2009, 2013a]) can in addition to mere expected travel times also effectively incorporate selected aspects of real-world road networks such as times spent at intersections and turn restrictions, route complexities (as, e.g., the number of turns), various uncertainties, fuel costs, and dynamic traffic information or time dependencies derived from historical traffic data. Most methods in this section, however, may require substantial modifications to fully incorporate such cost models.

We first give a brief historical overview and then we survey exact and approximate methods for planar graphs and methods for road networks, further subdividing into hierarchical and graph-labeling approaches.

Researchers started investigating point-to-point shortest-path problems immediately after the introduction of the general shortest-path problem [Minty 1957; Dantzig 1960; Klee 1964; Smolleck 1975]. Experimental evaluation [Hitchner 1968; Bourgoin and Heurgon 1969; Dreyfus 1969; Gilsinn and Witzgall 1973; Pape 1974; Golden 1976; Cherkassky et al. 1996; Zhan and Noon 1998; Demetrescu et al. 2008] has always been a central part of research on shortest-path algorithms. Starting from classical single-source shortest-path algorithms, it has been noted that, if an application requires only point-to-point distances, many SSSP algorithms can be stopped early. Furthermore, SSSP algorithms may run faster when executed from

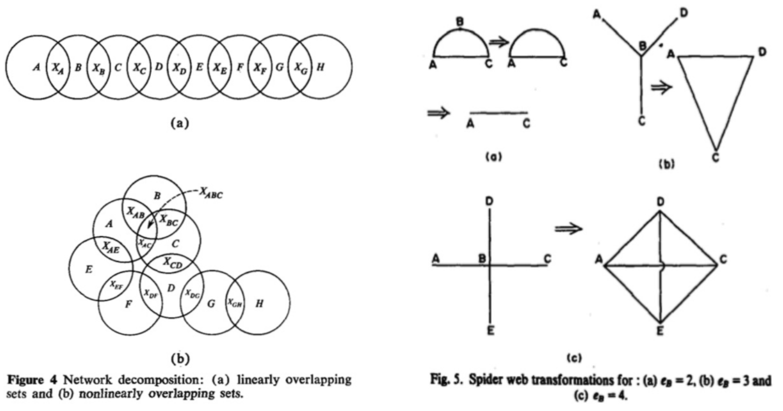
(a)
(b)
Figure 4 Network decomposition: (a) linearly overlapping sets and (b) nonlinearly overlapping sets.
(a)
(b)
(c)
Fig. 5. Spider web transformations for : (a) $e_B = 2$, (b) $e_B = 3$ and (c) $e_B = 4$.

Fig. 2. Illustrations of early preprocessing techniques: *Left:* Extracted from Hu and Torres [1969], used with permission of IBM, © 1969. The technique of Hu and Torres first decomposes the network into overlapping subnetworks. Next, with each subnetwork treated separately, conditional shortest paths are obtained using triple operations. Finally, these conditional shortest paths are used to obtain the shortest paths between paired nodes in the original network by matrix mini-summation. *Right:* Extracted from van Vliet [1978], used with permission of Elsevier, © 1978. The *spider web* transformations of van Vliet, illustrated by contractions for nodes of degrees 2, 3, and 4. Van Vliet's methods contract nodes such that *groups of two or more links from the original network are combined into single links representing minimum distance paths between their end nodes.*

the source and the target simultaneously—this technique is also called *bidirectional* search [Dantzig 1963; Nicholson 1966; Boothroyd 1967; Chartres 1967; Murchland 1967; Pohl 1971]. Bidirectional search can be a very powerful technique for networks other than transportation networks as well.

Researchers further found that the representation of a graph in memory greatly affects the performance of the algorithm. For sparse graphs, representing the graph by an *adjacency list* is quite efficient, sorting each list by starting nodes (this representation is sometimes termed *forward star form* [Mehlhorn and Sanders 2008]). It may be efficient to also sort the edges of a node by their length [Dial et al. 1979]. Such a sorting step can be seen as preprocessing the graph in order to speed up the query algorithm (albeit without decreasing the worst-case query time).

Reordering nodes and edges was just the beginning. Researchers tried to further speed up the shortest-path algorithms of Dijkstra [1959], D'Esopo [Pollack and Wiebenson 1960; Pape 1974], and Moore [1959]. Network decomposition [Kitamura and Yamazaki 1965; Mills 1966; Land and Stairs 1967; Farbey et al. 1967; Hu 1968; Hu and Torres 1969] was used to speed up APSP algorithms on sparse networks (see Figure 2). Other than the articles on the network decomposition technique, to the best of my knowledge, the thesis of Smolleck [1975] (see also Smolleck and Chen [1981]) and an article of van Vliet [1978] appear to be among the first reports on the shortest-path query problem with considerable preprocessing. Somewhat related, a method of Gallo [1980] effectively uses a prior shortest-path computation to speed up subsequent shortest-path computations (*cf.* Klein [2005]). If, however, the next computation is "far" from the previous one, speedups may be minimal.

Smolleck models the network by an electric circuit, wherein each edge is mapped to an impedance to efficiently answer *approximate* shortest-path queries. According to Deo and Pang [1984], Smolleck achieves a speedup of 30 compared to Dijkstra's algorithm (on a graph with 2,047 nodes and 2,547 edges); the paths are on average 1.9% longer than the optimal path, and the preprocessing time is reported to be 1,000 times slower than the query time. Van Vliet introduced[3] heuristics termed *spider web techniques* [van Vliet 1978, Section 6], which contract nodes and introduce shortcut edges (Figure 2). Van Vliet in some sense combined APSP and SSSP techniques into a query method, illustrating the tradeoff that shortest-path query methods address. According to the article, van Vliet's contraction techniques decrease the CPU time for multiple queries by approximately 25%.

For road networks, if in addition to the graph the *geographical coordinates* of the nodes are known, A* heuristics [Gelernter 1963; Samuel 1963; Doran 1967; Hart et al. 1968] based on the Euclidean distance have been used to guide the search toward the target [Sedgewick and Vitter 1986]. These heuristics are quite well known, rather easy to implement, and widely used. More recent techniques (see Section 3.2), however, yield substantially improved speedups.

3.1. Theoretical Results on Distance Oracles for Planar Graphs and Generalizations

Due to the importance of planar graphs as a more or less accurate model for road networks, shortest-path queries for planar graphs have been studied extensively. Real-world road networks may not actually be planar graphs, but they seem to share some properties with planar graphs such as small separators and some sense of orientation [Eppstein and Goodrich 2008]. One might also argue that, among the graph classes theoreticians know how to design efficient algorithms for, planar graphs and their extensions are the closest to road networks. Other related graph classes with known approximate distance oracles are geometric graphs [Gudmundsson et al. 2008; Sankaranarayanan and Samet 2009; Sankaranarayanan et al. 2009] and graphs with bounded doubling dimension [Har-Peled and Mendel 2006; Abraham et al. 2008a; Bartal et al. 2011; Kawarabayashi et al. 2011].

3.1.1. Exact Shortest Paths. The contents of this section were partially extracted from Mozes and Sommer [2012, Section 1.1]. In the following, as above, $\tilde{O}(\cdot)$–notation accounts for logarithmic factors. We give a brief overview of results; a summary can be found in Table III, illustrated in Figure 3.

Djidjev [1996], improving upon Feuerstein and Marchetti-Spaccamela [1991], proves that, for any $S \in [n, n^2]$, there is an exact distance oracle using space $O(S)$ with query time $O(n^2/S)$. Concurrent results for smaller ranges can be found in Arikati et al. [1996], Buchholz and Riedhofer [1997], and Riedhofer [1997]. These constructions use only recursive $O(\sqrt{n})$–separators [Ungar 1951; Lipton and Tarjan 1979; Djidjev 1985; Gilbert et al. 1984; Alon et al. 1990], and consequently, oracles with these space–query time tradeoffs also exist for bounded-genus and minor-free graphs. Experimental results indicate that real-world road networks appear to have recursive separators of size proportional to roughly $\sqrt[3]{n}$ [Delling et al. 2011], except that road networks contain some grids of considerable sizes as subgraphs (with separators of size $\Omega(k)$ for a $k \times k$–grid).

Djidjev's method also follows the overall approach described in Section 1.2: the set of landmarks is chosen as the set of *boundary nodes* of an *r–division* [Frederickson 1987; Klein et al. 2013]. An r–division is essentially a partition of the edges into $O(n/r)$ *regions* of size r such that each region R has at most $O(\sqrt{r})$ *boundary nodes* ∂R (a node

[3]Van Vliet attributes the idea to Hu [1969], who termed it *distance-equivalent networks*. There may be a connection to the *minimum-route transformations* of Akers [1960] and William S. Jewell (no reference). These network changes are based on *Wye-Delta*–transformations of electrical networks. However, the transformations appear to be restricted to planar networks and to two or three terminals. Hu and Torres [1969, p.390] attribute smaller *flow-equivalent networks* to Akers [1960]. Van Vliet also relates it to *triple operations* [Floyd 1962; Murchland 1965; Hu 1968]. Such a triple operation compares an edge length with the lengths of paths with two edges using an intermediate *pivot* node. The method is mainly used in APSP algorithms.

Table III. Time and Space Complexities of Exact Distance Oracles for Directed Planar Graphs
The tradeoff between space and query time is illustrated in Figure 3. For the large-space result of Chen and Xu 2000, an additive inverse-Ackermann term in the query time is suppressed in this table.

Preprocessing	Space	Query Time	Restriction (if any)	Reference(s)
none	$O(n)$	$O(n)$		[Henzinger et al. 1997]
$o(n^2)$	$o(n^2)$	$O(1)$		[Wulff-Nilsen 2010]
$O(n^{3/2})$	$O(n^{3/2})$	$O(\sqrt{n})$		[Arikati et al. 1996; Djidjev 1996]
$O(S)$	$O(S)$	$O(n^2/S)$	$S \in [n^{3/2}, n^2]$	[Djidjev 1996, Thm. 3]
$O(n\sqrt{S})$	$O(S)$	$O(n^2/S)$	$S \in [n, n^{3/2}]$	[Djidjev 1996, Thm. 4]
$O(n \lg^2 n)$	$O(n \lg n)$	$O(\sqrt{n} \lg^2 n)$		[Fakcharoenphol and Rao 2006; Klein et al. 2010]
$O(n \lg n)$	$O(n)$	$O(n^{1/2+\epsilon})$		[Nussbaum 2011; Mozes and Sommer 2012]
$O(n \lg \lg n)$	$O(n)$	$O(n/\text{poly}(\lg n))$		[Italiano et al. 2011]
$O(n\sqrt{S})$	$O(S)$	$O((n/\sqrt{S}) \lg n)$	$S \in [n^{4/3}, n^{3/2}]$	[Djidjev 1996, Thm. 5]
$O(n^3/S)$	$O(S)$	$O((S/n) \lg n)$	$S \in [n^{4/3}, n^{3/2}]$	[Chen and Xu 2000]
$O(n\sqrt{S})$	$O(S)$	$O((n/\sqrt{S}) \lg(n/\sqrt{S}))$	$S \in [n^{3/2}, n^2]$	[Chen and Xu 2000]
$O(S \lg n)$	$O(S)$	$O((n/\sqrt{S}) \lg(n/\sqrt{S}))$	$S \in [n^{4/3}, n^2]$	[Nussbaum 2011, Thm. 4.1]
$O(S)$	$O(S)$	$O((n/\sqrt{S}) \lg^{1.5} n)$	$S \in [n^{4/3} \lg^{1/3} n, n^2]$	[Cabello 2012, Thm. 12]
$O(S\sqrt{S/n} \lg^2 n)$	$O(S)$	$O(n/\sqrt{S})$	$S \in [n^{4/3}, n^2]$	[Nussbaum 2011, Thm. 5.2]
$O(S \lg^2 n)$	$O(S)$	$O((n/\sqrt{S}) \lg^{2.5} n)$	$S \in [n \lg \lg n, n^2]$	[Mozes and Sommer 2012]

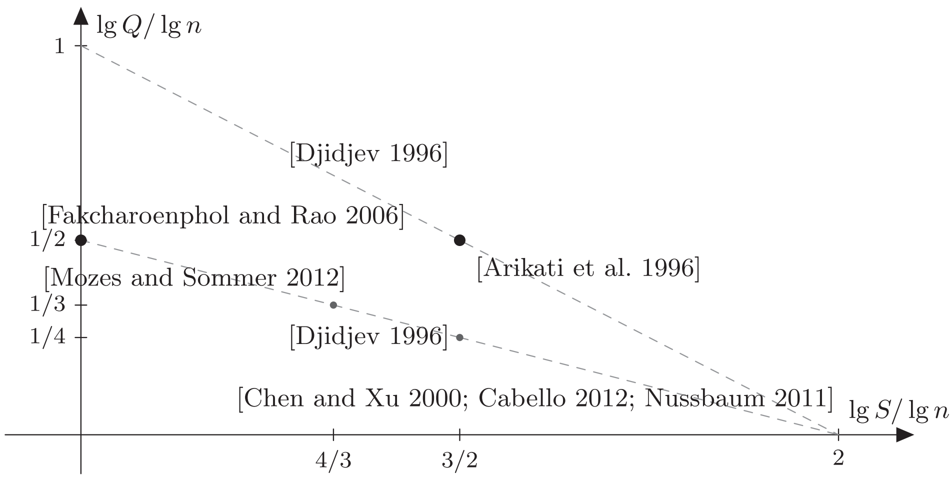

Fig. 3. Distance oracles for planar graphs: The figure illustrates the tradeoff between the space [S] and the query time [Q] for different data structures on a doubly logarithmic scale, ignoring constant and logarithmic factors. The upper line represents the $Q = n^2/S$ tradeoff (completely covered by Djidjev [1996]; the range $S \in [n^{3/2}, n^2]$ is covered by Arikati et al. [1996]; for $S = n^{3/2}$ see also Buchholz and Riedhofer [1997] and Riedhofer [1997]; SSSP ($S = Q = n$) and APSP ($S = n^2$) also lie on this line). Planarity is not necessary; only recursive separators of size $O(\sqrt{n})$ are assumed to achieve this tradeoff. The lower line represents the $Q = n/\sqrt{S}$ tradeoff; the range $S \in [n^{4/3}, n^{3/2}]$ is covered by Djidjev [1996]; extended to $S \in [n^{4/3}, n^2]$ by Chen and Xu [2000] and Cabello [2012] (query time improvements by Nussbaum [2011]), the point $S = n$ is covered by Fakcharoenphol and Rao [2006] (similar claims in Buchholz [2000]), and the full range is covered by Mozes and Sommer [2012].

is called a boundary node if it is adjacent to edges in different regions). Next, pairwise distances among all landmarks are computed and stored. The space requirements for this distance table are $S = O((n/\sqrt{r})^2)$. The query algorithm, given a pair of nodes (s, t), first searches (using SSSP [Henzinger et al. 1997]) both regions R_s, R_t. If s and t are in the same region R, and if the shortest path is entirely contained in R, the shortest-path distance has been found in time $O(r)$ (short-range query). Otherwise, exact distances to all corresponding landmarks (boundary nodes in ∂R_s, ∂R_t, respectively) have been computed, and the distance is the minimum among all pairs of landmarks $(l_s, l_t) \in \partial R_s \times \partial R_t$ of $d(s, l_s) + d(l_s, l_t) + d(l_t, t)$. Since the number of boundary node pairs is bounded by $O((\sqrt{r})^2)$, the query time for long-range queries is also $O(r)$.

For a smaller range, also exploiting noncrossing properties of planar graphs, Djidjev proves that, for any $S \in [n^{4/3}, n^{3/2}]$, there is an exact distance oracle with space $O(S)$ and query time $\tilde{O}(n/\sqrt{S})$. For further improvements and extensions, see Chen and Xu [2000], Cabello [2012], Nussbaum [2011], and Mozes and Sommer [2012]. Djidjev observes that, if the boundary nodes ∂R of each region R form a simple cycle,[4] then not all pairs $(l_s, l_t) \in \partial R_s \times \partial R_t$ need to be considered for long-range queries: the intersection pattern among shortest paths between nodes on two disjoint cycles of a planar graph is limited such that, instead of exploring $O((\sqrt{r})^2)$ pairs, the intermediate minimization step for long-range distances can be computed in time $O(\sqrt{r} \lg r)$ (the intersection pattern is restricted only if intermediate distances are computed in $G \backslash (R_s \cup R_t)$ and not in G). Distances from each node to its region's boundary nodes are then either stored, computed at query time using an SSSP algorithm, or retrieved using multiple *MSSP* data structures [Klein 2005] (see also Section 3.1.3). Short-range distances can be computed faster by recursively computing distance oracles for each region.

Fakcharoenphol and Rao [2006] further exploit the noncrossing property (which they call the *Monge* property [Monge 1781; Hoffman 1963]). They call the complete bipartite graph among $\partial R_s \times \partial R_t$ in $G \backslash (R_s \cup R_t)$ a *Dense Distance Graph (DDG)*. Their query algorithm can efficiently handle multiple DDGs simultaneously in time roughly proportional only to the number of nodes in these DDGs (as opposed to the number of edges, which for DDGs is quadratic). Their technique is used for various distance oracles with low space and preprocessing complexities [Fakcharoenphol and Rao 2006; Italiano et al. 2011; Nussbaum 2011; Mozes and Sommer 2012].

The only lower bounds known are for distance labels, proving that total label length $\Omega(n^{3/2})$ is required for planar graphs with edge lengths [Gavoille et al. 2004, Corollary 3.11] (the best upper bound uses total label length $O(n^{3/2} \log n)$ [Gavoille et al. 2004, Corollary 2.5]). There are no lower bounds on distance oracles for planar graphs. It is an open problem whether there exists another tradeoff curve strictly below $Q = \tilde{O}(n/\sqrt{S})$.

3.1.2. Approximations. To obtain constant or polylogarithmic query times while maintaining almost linear space, approximate distance oracles are considered. Thorup [2004] presents efficient $(1 + \epsilon, 0)$–approximate distance oracles for directed planar graphs. One of the main ingredients of Thorup's construction is a special separator consisting of a constant number of shortest paths (instead of a general set of $O(\sqrt{n})$ nodes as in the Lipton-Tarjan separator theorem [1979]). Each node computes and stores shortest-path distances to a set of $O(1/\epsilon)$ landmarks per level, recursively for $O(\lg n)$ levels (see also Klein and Subramanian [1998] and Klein [2002] for related constructions). For directed graphs, the construction is actually more involved and the bounds show a moderate dependency on the largest edge length. Distances among subsets of landmarks (those on the same shortest path Q) can be represented in a very compact way by just storing the position on Q. Improved tradeoffs have been announced [Kawarabayashi et al. 2013].

Thorup's oracle for undirected graphs has been implemented and evaluated for road networks [Muller and Zachariasen 2007]. The results, however, indicate that, for these road networks, it is not competitive with the specialized methods discussed in Section 3.2.

[4]More generally, $O(1)$ cycles can be handled; this simplified overview assumes a single boundary cycle per region.

Kawarabayashi, Klein, and Sommer [2011] extend Thorup's results to undirected graphs embedded in a surface of Euler genus g. Abraham and Gavoille [2006] extend Thorup's result to minor-free graphs. They prove that minor-free graphs can be recursively separated using a (large) constant number of shortest paths. Based on these shortest-path separators, they then construct approximate distance oracles as in Thorup [2004]. Kawarabayashi et al. [2011] provide tunable tradeoffs for the aforementioned approximate distance oracles for planar, bounded-genus, and minor-free graphs such that the space requirements can be made linear in the graph size while maintaining polylogarithmic query time (with techniques similar to those used for exact tradeoffs illustrated in Figure 3).

3.1.3. Restricted Queries.

Bounded-length queries. Kowalik and Kurowski [2006] prove that, for unit-length planar graphs, there is a distance oracle with linear preprocessing time and space requirements that answers queries for distances bounded by a constant h in constant time (which is an improvement over the $O(\lg n)$ query time in Eppstein [1999, Theorem 12]). Dvorak, Král, and Thomas [2010] extend their result to essentially all sparse graphs (sparse as defined in Nesetril and de Mendez [2006]).

One-face queries. Klein [2005] gives a distance oracle that preprocesses a graph with a specified face f in time and space $O(n \lg n)$ to answer distance queries between any node incident to f and any other node (incident to an arbitrary face) in time $O(\lg n)$. This data structure is also referred to as a *multiple-source shortest path* (MSSP) data structure and it is used as an ingredient in other distance oracles. Schmidt [1998] provides a similar data structure for grid graphs. Cabello and Chambers [2012] give a different and simpler algorithm, which also extends Klein's result to graphs with genus g.

3.2. Route Planning for Road Networks

Route planning for transportation networks (road networks in particular) has been studied intensively for many years. Recently, the 9th DIMACS Implementation Challenge, which took place in 2006, stimulated a lot of research with impressive results [Demetrescu et al. 2008]. In the following, we give a brief overview. The tradeoffs between space and query time are summarized in Figure 4. For more details on recent results, we refer to the survey on route planning [Delling et al. 2009a], the survey on A*–based point-to-point shortest-path queries [Goldberg 2007] (see also Goldberg et al. [2009]), the overview on engineering large network applications [Zaroliagis 2008], and the Ph.D. theses of Schultes [2008] and Delling [2009]. Route planning is also strongly related to efficient path query processing on spatial networks [Papadias et al. 2003; Gupta et al. 2004; Demir et al. 2008; Samet et al. 2008; Sankaranarayanan et al. 2009; Sankaranarayanan and Samet 2009]. Let us re-emphasize that the focus of this survey is on static networks. There are numerous methods that work with more dynamic transportation networks (for various definitions and interpretations of *dynamic*), but these methods are not considered in this survey.

From a technical perspective, two types of route planning methods can be distinguished: (i) methods that obtain improvements mainly by exploiting structural properties of the input graph (somewhat related to the exact methods for planar graphs, as described in Section 3.1.1), and (ii) methods that also exploit properties of the shortest-path metric (induced by the underlying edge lengths), which appears to be of a hierarchical nature in road networks. Methods of that second type tend to yield significantly improved tradeoffs at the cost of, potentially, being somewhat less "robust" (meaning that changes to the edge length function such as dynamic

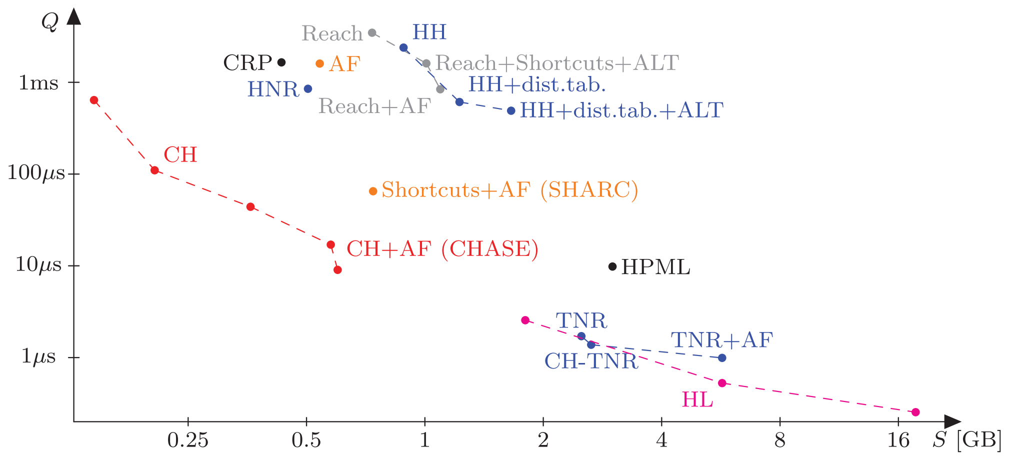

Fig. 4. Route planning for road networks: the tradeoff between space [S] and query time [Q] for recent shortest-path query data structures, depicted using doubly logarithmic scales. The performance numbers represented by this figure were extracted from Delling et al. [2009a, Table 1], Bauer et al. [2010b, Table 8], Abraham et al. [2011b, Table 1], Arz et al. [2013, arXiv Table 5], and Delling et al. [2013b, Table 2]. Performance numbers were obtained on different machines and scaled with best effort to make methods comparable. Colors and dashed lines do not carry any meaning; lines serve the purpose of visually connecting dots corresponding to different implementations or different variants of the same method.
Methods using *contraction hierarchies (CH)* [Geisberger et al. 2008; Sanders et al. 2008; Geisberger et al. 2012] dominate the low-space regime; methods based on *reach* [Gutman 2004; Goldberg et al. 2009] and *highway hierarchies (HH)* [Sanders and Schultes 2005, 2006; Delling et al. 2009b] can be seen as the "first generation" of CH; *transit-node routing (TNR)* [Bast et al. 2007a; Bauer et al. 2010b; Arz et al. 2013] and *hub labels (HL)* [Abraham et al. 2011b; Delling et al. 2013b] dominate the fast-query-time regime.

updates or incorporating realistic turn costs may have unexpected consequences to the methods' performances).

Efficient practical methods to answer shortest-path queries are often devised by following a feedback loop that consists of four steps: design, analysis, implementation, and experimentation. This approach is also called *algorithm engineering* [Sanders 2009, Figure 1]. Since experimentation is an integral part of the feedback loop, the choice of the datasets may highly influence the outcome of the algorithm engineering process. Whenever possible, experiments are run with input graphs that are actually used in practice. Route planning methods discovered by an algorithm engineering process include, for example, *Highway Hierarchies (HH)* [Sanders and Schultes 2005, 2006] and its exceedingly popular successor called *Contraction Hierarchies (CH)* [Geisberger et al. 2008]. Both methods depend on structural properties of the input graph and rather heavily on the edge lengths and the shortest-path metric they impose. If the length function is chosen such that edge lengths correspond to Euclidean distances, the methods still work well but their performance is worse than the performance when edge lengths correspond to (estimated) travel times. It is for the so-called *travel time metric*, where these hierarchical methods excel, and where the performances obtained are truly impressive (see also other methods, as illustrated in Figure 4). However, estimating travel times for road segments is a highly nontrivial task in itself and it is not entirely clear to what extent the estimates used in research datasets are accurate representations for actual travel times observed in the real world. To the best of my knowledge, there are only a few studies on the *robustness* of these methods, investigating whether the performances would drop significantly upon changes to the length function; see, for example, Delling et al. [2013a]. Recent theoretical research (Section 3.2.4) strives to explain the success of these speedup techniques, analyzing the running times of preprocessing and query algorithms by appropriately modeling graph and metric properties of road networks.

In our overview of recent methods, two preprocessing strategies (mostly orthogonal to the aforementioned types) are distinguished. Approaches based on *graph annotation* attach additional information to each node or edge, based on which, at query time, the search tree can be prioritized or pruned. These approaches are inherently based on an SSSP algorithm such as Dijkstra's algorithm. They are quite general in nature, and some also work very well on graphs other than those stemming from road networks. *Hierarchical approaches* are often somewhat more tailored toward their use in road networks. These algorithms usually compute an additional graph structure to speed up shortest-path queries.

3.2.1. Graph Annotation Approaches. An annotation approach is to attach additional information to nodes or edges of the graph. Based on this information, the query algorithm decides how to prioritize nodes in the queue, or which part of the graph not to search (i.e., how to *prune* the search space). A subset of these methods is sometimes also called *goal-directed search algorithms*.

*A** [Gelernter 1963; Samuel 1963; Kung et al. 1986; Hart et al. 1968; Doran 1967] is a popular search technique in artificial intelligence. The idea is to direct the search toward the goal. In the priority queue implementation of Dijkstra's algorithm, at each iteration, the node with the shortest distance to the source is extracted from the queue. In the A* algorithm, instead of ordering nodes by their distance from the source, nodes in the queue are ordered by their distance from the source plus a *potential*, which estimates the remaining distance to the target. By adding a potential to the priority of each node, the order in which nodes are removed from the priority queue is altered. A good potential function increases the priority of nodes that lie on a shortest path to the target (usually by decreasing the priority of other nodes). In road networks, for example, if the coordinates of the target are known, the Euclidean distance provides a reasonable potential function [Sedgewick and Vitter 1986]. This has been exploited and applied successfully. In general, however, the coordinates may not be known. Metric embeddings or drawings [Wagner and Willhalm 2005] may provide coordinates.

Goldberg and Harrelson [2005] (see also Goldberg and Werneck [2005] and Goldberg et al. [2006, 2007]) propose to use a set of landmarks $L \subseteq V$ and the triangle inequality to compute node potentials (their method is sometimes called *ALT*, short for A*, landmarks, and triangulation). Analogous to the distance oracle of Thorup and Zwick 2005, all nodes $v \in V$ know the distance to all landmarks $l \in L$. This auxiliary information fits perfectly into the framework of Section 1.2. For two nodes $u, v \in V$ and a landmark $l \in L$, the triangle inequality yields that $d(u, v) \geqslant d(u, l) - d(v, l)$. Taking the maximum difference over all $l \in L$ yields the best estimate, which is used as a potential in the A* search. The quality of the lower bound highly depends on the landmark selection. Since in the preprocessing phase the distances to all landmarks need to be computed and stored, the preprocessing time and the space consumption also depend on the number of landmarks. A central question is how to select few but good landmarks. Random selection is a straightforward approach, but it may not necessarily provide good coverage, meaning that some nodes are far from all landmarks. Several heuristics have been proposed to improve coverage [Goldberg and Harrelson 2005; Goldberg and Werneck 2005], or to choose "important" nodes [Potamias et al. 2009]. Theoretical results on beacon-based triangulations [Kleinberg et al. 2009] characterize, to some extent, the strengths and weaknesses of ALT: for graphs with bounded doubling dimension, triangulation using a constant number of landmarks yields $(1+\epsilon, 0)$–approximate distances for a $(1-\sigma)$–fraction of the nodes; it is also shown that this *slack* σ is necessary. While

A* with landmarks works for general graphs, it can be expected to perform particularly well on graphs with low doubling dimension.

A* is easy to implement and it yields decent speedups. Bidirectional A*, however, is not entirely straightforward: either the termination criterion is changed [Pohl 1971; Kwa 1989], or the potential functions for the forward and the backward search need to be consistent; averaging the forward and backward potential yields a consistent potential function [Ikeda et al. 1994].

Precomputed Cluster Distances (PCD) [Maue et al. 2009] is a somewhat similar approach. The network is partitioned into clusters, and distances between any pair of clusters are precomputed. These cluster distances yield upper and lower bounds for distances, based on which the search space can be prioritized or pruned.

Arc Flags (AF) [Lauther 2004; Köhler et al. 2005; Möhring et al. 2006]. The preprocessing algorithm partitions the graph into clusters and then, for each cluster C, marks all edges where shortest paths toward nodes in C start. The query algorithm prunes edges that are not marked with the target cluster. A related approach uses geometric containers [Wagner and Willhalm 2003; Wagner et al. 2005]. On its own, AF preprocessing is rather expensive (there is a fast parallel preprocessing algorithm [Delling et al. 2013]). However, when applied within a hierarchy [Möhring et al. 2006] or when combined with other techniques, it can be very efficient [Bauer and Delling 2009; Bauer et al. 2010b].

Reach-Based Routing [Gutman 2004] is technically an annotation approach; however, it should, at least in spirit, also be considered a hierarchical approach. Reach is one of the first methods specifically capturing the hierarchical nature of road networks and exploiting it with provable correctness guarantees. Prior industrial methods classified roads according to heuristic hierarchies (often using road categories), thereby sacrificing correctness. In reach-based routing, each node is assigned a so-called *reach value*, which determines whether a particular node should be considered during Dijkstra's algorithm. To have a high reach value, a node must lie on a shortest path that extends a long distance in both directions from the node. A node is excluded from consideration if its reach value is small, that is, if it does not contribute to any path long enough to be of use for the current query. When combined with shortcuts [Goldberg et al. 2009], *Reach* is rather similar to many hierarchical approaches.

3.2.2. Hierarchical Approaches. Hierarchical methods to compute shortest paths in graphs have been proposed by many researchers. Many methods effectively exploit the inherent hierarchical nature of road networks. However, in this section, *hierarchical* does not exclusively refer to this hierarchy of roads. Many methods construct an auxiliary graph with multiple levels: a hierarchy of graphs. A shortest-path query is then answered by searching only a small part of the auxiliary graph, often using Dijkstra's algorithm. In the following, we give a brief overview of selected recent hierarchical approaches.

Multilevel Overlay Graphs [Jung and Pramanik 1996; Jing et al. 1998; Schulz et al. 2002; Holzer et al. 2008; Delling et al. 2009, 2013a] build a hierarchy of graphs with node sets at higher levels chosen as subsets of the node sets at lower levels (*cf.* a hierarchy of landmarks). Two nodes u, v at level i may be connected by an edge with length corresponding to the distance in G if the shortest path in level $i - 1$ does not use any other node of level i. Selecting the landmark set on higher levels is one of the most critical components of these methods; several selection heuristics are proposed and evaluated. *Highway-Node Routing (HNR)* [Schultes and Sanders 2007] effectively uses *highway nodes* as landmarks (see also next paragraph). *Customizable Route Planning (CRP)* [Delling et al. 2013a] uses small recursive separators [Delling et al. 2011]. CRP is currently used in Microsoft Bing Maps. See also Section 3.1.1 for similar separator-based methods for planar graphs. Previous methods based on separators were significantly less efficient; among other reasons, the performance of CRP is very good since the preprocessing algorithm puts substantial effort into minimizing the sizes of the separators [Delling et al. 2011].

Highway Hierarchies (HH) [Sanders and Schultes 2005, 2006] are based on the observation that a certain class of edges (the *highway* edges) tend to have greater representation among the portion of the shortest paths that are not in the vicinity of either the source or the target (similar to high *reach* values [Gutman 2004]). A recursive computation of these edges, paired with a contraction step, leads to a hierarchy of graphs that enables an impressive speedup at query time.

Contraction Hierarchies (CH) [Geisberger et al. 2008, 2012] is the exceedingly popular successor of HH. An integral ingredient of HH is its initial contraction step. Nodes with low degree can be contracted, since their removal does not cause many additional edges (an observation related to van Vliet's *spider web* [van Vliet 1978] and Hu's *distance-equivalent networks* [Hu 1969]). This observation can be generalized [Geisberger et al. 2008]: for each node, the number of potential *shortcut* edges is computed. If for a node under consideration the number of shortcuts is smaller than the number of expected shortcuts based on the node degree, the node is contracted. A node contraction can also be interpreted as a particularly structured way of adding shortcuts. Thinking about shortest-path queries in road networks, one almost immediately notes that many nodes have degree 2 (in the undirected sense) and that these can be contracted. Van Vliet [1978] contracts nodes up to degree 4; it is reported that contractions of nodes with higher degrees did not yield any speedup but a slowdown. CH uses intelligent heuristics to contract nodes in the "right" order. This order π defines a directed star for each node as follows: node u with rank $\pi(u)$ must be connected to a node v with $\pi(v) > \pi(u)$ if the shortest path from u to v does not use any node $w \neq v$ with $\pi(w) > \pi(u)$. The union of all these directed stars defines the forward (or upward) CH. The backward CH is defined analogously. Using these compact auxiliary graphs, the bidirectional query algorithm can efficiently find shortest paths. Contraction-based techniques perform very well in practice, the space overhead is small, and the preprocessing step is particularly efficient.

Transit-Node Routing (TNR) [Bast et al. 2007a, 2007b; Arz et al. 2013] is based on the following observation: when driving somewhere sufficiently far away, drivers usually leave their current location via one of only a few access routes to a relatively small set of landmarks called *transit nodes*. These landmarks are then interconnected by a network relevant for long-distance travel. The TNR method precomputes all shortest paths to landmarks (stars) and all shortest paths among landmarks (clique). The preprocessing is quite expensive but the query time is very low, since, for any two locations far enough, it essentially requires only a few dozen table lookups for all pairs of corresponding landmarks. A recent variant of TNR [Arz et al. 2013] can be interpreted as a sophisticated combination of CH with a distance table: transit nodes are chosen as the top–k nodes in contraction order, short-range queries are computed using CH, and long-range queries correspond to several table lookups. Depending on the needs of the application, the number of transit nodes k can be varied, thereby determining the tradeoff between space and query time.

Hub Labels (HL) [Abraham et al. 2011b, 2012b; Delling et al. 2013b] are used in the method that currently offers the fastest query times. During preprocessing, each node u computes and stores the distance to a set of carefully chosen landmarks $L(u)$ in its label (stars; for directed graphs, different landmarks $L^+(u), L^-(u)$ may be used for forward and backward distances). At query time, given two labels corresponding to distances to landmarks $L(s), L(t)$, respectively, the algorithm simply computes and outputs $\min_{l \in L(s) \cap L(t)} d(s, l) + d(l, t)$. This can even be implemented in SQL, which allows for more general queries involving, for example, points of interest [Abraham et al. 2012a]. As long as $L(s), L(t)$ are small, the query algorithm is efficient. Furthermore, if labels are stored consecutively for each node, the query algorithm also has good locality. The main difficulty lies in choosing $L(u)$ for a node u: for any two nodes s, t, the intersection of their landmark sets $L(s) \cap L(t)$ shall contain at least one node on a shortest $s - t$ path; that is, the set of landmarks must *cover* all shortest paths (see Cohen et al. [2003] and Section 2.2). For road networks, small labels can be computed efficiently using CH search spaces.

In the HL method, shortest paths are represented by two hops; in the TNR method, three hops are used. The extremely fast query times are paid for by rather high space requirements (compression in Delling et al. [2013b]).

3.2.3. Combinations. Combining graph annotation and hierarchical approaches often yields powerful methods. Several combinations have been investigated and evaluated empirically [Holzer et al. 2005; Bauer et al. 2010b]. Particularly strong combinations are Reach with Shortcuts [Goldberg et al. 2009]; CHASE [Bauer et al. 2010b], which combines Contraction Hierarchies with Arc Flags; and SHARC [Bauer and Delling 2009; Brunel et al. 2010], which combines Shortcuts with Arc Flags.

To sum up, the "tricks of the trade" [Bast 2009, Section 3] for fast routing on transportation networks are bidirectional search, exploiting hierarchy, graph contraction, goal direction, and distance tables. Let us conclude by noting that some of the methods described in this section, despite being tailored toward their use in road networks, can be adapted to work on other networks as well, such as those stemming from public transportation, albeit with somewhat reduced speedups [Berger et al. 2009; Bast 2009].

3.2.4. Analysis. The observed performance of the aforementioned methods is outstanding; however, complexity results are mostly experimental (exactness and correctness are proven).

There are some worst-case results for various speedup techniques. One core part of many speedup techniques, particularly the hierarchical ones, is the insertion of *shortcuts*; a shortcut is an additional edge (u, v) whose length is equal to the distance from u to v, and that represents shortest $u - v$–paths in the graph. Let the *hop-length* of a path be defined as the number of edges on a shortest path. The shortcut problem [Bauer et al. 2009] consists of adding a fixed number of shortcuts to a graph such that the sum of the hop lengths of hop-minimal shortest paths on the graph is minimized. This optimization problem is difficult to solve both optimally and approximately unless $\mathbf{P} = \mathbf{NP}$ [Bauer et al. 2009]. If the shortest paths are unique, a greedy algorithm can find a solution that is optimal up to a constant factor.

Contraction Hierarchies can be seen as a structured way of adding shortcuts by contracting nodes in a sophisticated order [Geisberger et al. 2008, 2012; Abraham et al. 2012b]. However, computing or even approximating the optimal ordering is **NP**–hard [Bauer et al. 2010a]. For graphs with small recursive separators such as planar graphs, there are bounds on CH preprocessing and space [Milosavljevic 2012] as well as query time [Bauer et al. 2013] (some based on a relation to nested dissection [Lipton et al. 1979]).

Abraham et al. [2010, 2011a] found that, if a graph has a low *highway dimension*, algorithms based on Reach [Gutman 2004; Goldberg et al. 2009], Contraction Hierarchies [Geisberger et al. 2008], Hub Labels [Abraham et al. 2011b], and SHARC [Bauer and Delling 2009] have provable efficiency guarantees. Intuitively, a graph has a small highway dimension if, for every radius $r > 0$, there is a sparse set of nodes S_r such that every shortest path of length greater than r includes a node from S_r. A set is deemed sparse if every ball of radius $O(r)$ contains only a small number of elements of S_r. Computing and analyzing the highway dimension of real road networks remains an open problem.

4. COMPLEX NETWORKS

Recently, shortest-path query algorithms and data structures have been studied for more general graphs, motivated by potential applications for real-world networks such as social networks or regulatory networks from biology. This section of the survey is rather vague, since research on shortest-path queries for complex networks seems to currently be evolving quite rapidly. Also, the absence of commonly agreed-on benchmarks poses difficulties in evaluation and comparison of existing algorithms. Similar difficulties arise for theoretical work, where there is currently a rather large variety of random-graph models for complex networks. For most of these complex network models, the following common properties have been identified: (i) complex networks appear to have small diameters (proportional to roughly $\lg n$; this property is referred to as the *small-world* property), (ii) oftentimes there is a large variety of node degrees (*scale-free* networks; the degree sequence obeys a *power law*[5]), and (iii) there seem to be no small separators (linear-sized *core*). Most of these properties currently cannot be exploited by common algorithmic techniques; other properties, such as the one that these networks have nodes with high degree (so-called *hubs*), have been exploited successfully in (approximate) shortest-path query algorithms.

Most results on shortest-path queries in complex networks are of an experimental nature. I am aware of only a few results with worst-case bounds, which are the following *compact routing schemes*.[6] Enachescu et al. [2008] analyze the compact routing scheme of Thorup and Zwick [2001] for $G_{n,p}$ random graphs. They prove that stretch $\alpha = 2$ can be achieved with space $\tilde{O}(n^{7/4})$ by selecting $\tilde{O}(n^{3/4})$ landmarks (later dominated by the $\tilde{O}(n^{5/3})$–space oracle of Pătraşcu and Roditty [2010] for general graphs). They also claim (without proof in the proceedings version) that multiplicative stretch α can be achieved with space $\tilde{O}(n^{1+2/(\alpha+1)+\epsilon})$. See also Krioukov et al. [2004] for results on the stretch distribution. Chen et al. [2012] provide an approximate distance oracle with stretch $\alpha = 3$ using space $O(n^{4/3})$ for certain random power-law graphs [Aiello et al. 2000; Chung and Lu 2002] (the actual space requirements may be smaller, depending on the power-law exponent). Their oracle is an adaptation of the Thorup-Zwick distance oracle using high-degree nodes as landmarks. Given that shortest paths in complex networks are typically very short (*small worlds*, *six degrees of separation*), it is, however, questionable whether worst-case guarantees on the multiplicative stretch of $\alpha = 2, 3$ are particularly useful.

Implementations and Adaptations. Krioukov et al. [2004] evaluate the compact routing scheme of Thorup and Zwick [2001] for Internet-like interdomain topologies and random power-law graphs. The Thorup-Zwick distance oracle uses information precomputed in two steps (see also Section 2.2). The query result is either a local exact distance or a triangulation via landmarks. Many implementations focus on the triangulation part, providing good estimates for long-range distances by carefully selecting landmarks [Potamias et al. 2009; Das Sarma et al. 2010; Gubichev et al. 2010; Tretyakov

[5]Whether or not many of these degree sequences actually obey power laws is a controversial question [Faloutsos et al. 1999; Clauset et al. 2009; Achlioptas et al. 2009; Roughan et al. 2011].

[6]Any compact routing scheme may serve as an approximate shortest-path oracle, retrieving each edge of the path by simulating the decision of each router. We do not attempt to cover results on compact routing in this review, instead referring to Gavoille and Peleg [2003], Thorup and Zwick [2001], and Abraham et al. [2008b] and the references therein.

et al. 2011; Cao et al. 2011; Qiao et al. 2011, 2012; Cheng et al. 2012]. Agarwal et al. [2012] focus on the local part, computing distances using ball intersections (which they call *vicinities*) with landmarks sampled with probability proportional to their degrees. For corresponding worst-case bounds, see also Patrascu and Roditty [2010] and Agarwal et al. [2011].

For two-hop covers [Cohen et al. 2003], there are several implementations [Schenkel et al. 2004, 2005; Cheng et al. 2006, 2008; Cheng and Yu 2009; Abraham et al. 2012b; Akiba et al. 2013]. Extensions to three and more hops have been proposed [Jin et al. 2008, 2009; Chang et al. 2012]. A method of Goldman et al. [1998] can be seen as some kind of two-hop cover, albeit their method actually came earlier. Rattigan et al. [2006, 2007] combine a clustering-type approach with landmarks.

Core–Fringe Methods. In the stretch–$(1, D)$ routing scheme of Brady and Cowen [2006], the algorithm first computes a shortest-path tree from the node with the highest degree. All nodes up to distance $D/2$ for some parameter D form the *core* with diameter D. The remaining nodes form the *fringe*, which is claimed to be almost a forest. At query time, distance estimates are computed as the minimum among the distances in multiple trees. Wei [2010] explores the property that many scale-free networks have reasonably small *tree-width* outside the *core* (see also Akiba et al. [2012] for further speedups).

Embeddings. Zhao et al. [2010] provide an embedding into Euclidean space. Das Sarma et al. [2010] provide an implementation of Bourgain's embedding [1985]. Embeddings into hyperbolic spaces [Shavitt and Tankel 2008; Cvetkovski and Crovella 2009; Papadopoulos et al. 2010; Zhao et al. 2011] and trees [Herzen et al. 2011] appear to be rather promising.

5. SUMMARY

Shortest-path query processing in graphs has been studied extensively by both theoreticians and practitioners. Practical investigations focus mainly on the class of transportation networks, for which substantial speedups with respect to classical SSSP algorithms can be achieved. For transportation networks, the focus of practical research efforts appears to be shifting toward dynamic and personalized scenarios. For complex networks, methods have been proposed only recently; their efficiency and optimality is still under investigation. Common benchmark networks have not crystalized yet.

Recent theoretical research on distance oracles for general graphs has been centered around improving preprocessing and query times (due to restrictive space lower bounds). For restricted graph classes such as sparse graphs, planar graphs, and power-law graphs, various questions remain to be solved. Distance oracles for directed graphs of restricted classes are mostly unknown territory.

ACKNOWLEDGMENTS

Many thanks to Michael E. Houle for his encouragement in writing this survey and for his valuable comments on an earlier version [Sommer 2010, Chapter 3]. I learned almost everything included in this survey from great collaborators while working on Sommer et al. [2009], Honiden et al. [2010], Djidjev and Sommer [2011], Gavoille and Sommer [2011], Kawarabayashi et al. [2011], Akiba et al. [2012], Lim et al. [2012], Chen et al. [2012], Mozes and Sommer [2012], and Kawarabayashi et al. [2013]. This survey has improved substantially thanks to many comments and suggestions from Daniel Delling, Ron Gutman, Shay Mozes, Dirck van Vliet, Christian Wulff-Nilsen, and the anonymous referees of this journal.

REFERENCES

I. Abraham, Y. Bartal, and O. Neiman. 2008a. Embedding metric spaces in their intrinsic dimension. In *19th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 363–372.

I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. F. Werneck. 2011a. VC-dimension and shortest path algorithms. In *38th International Colloquium on Automata, Languages and Programming (ICALP)*. 690–699.

I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. F. Werneck. 2012a. HLDB: location-based services in databases. In *SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*. 339–348.

I. Abraham, D. Delling, A. V. Goldberg, and R. F. F. Werneck. 2011b. A hub-based labeling algorithm for shortest paths in road networks. In *10th International Symposium on Experimental Algorithms (SEA)*. 230–241.

I. Abraham, D. Delling, A. V. Goldberg, and R. F. F. Werneck. 2012b. Hierarchical hub labelings for shortest paths. In *20th European Symposium on Algorithms (ESA)*. 24–35.

I. Abraham, A. Fiat, A. V. Goldberg, and R. F. F. Werneck. 2010. Highway dimension, shortest paths, and provably efficient algorithms. In *21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 782–793.

I. Abraham and C. Gavoille. 2006. Object location using path separators. In *25th ACM Symposium on Principles of Distributed Computing (PODC)*. 188–197.

I. Abraham and C. Gavoille. 2011. On approximate distance labels and routing schemes with affine stretch. In *25th International Symposium on Distributed Computing (DISC)*. 404–415.

I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. 2008b. Compact name-independent routing with minimum stretch. *ACM Transactions on Algorithms* 4, 3.

D. Achlioptas, A. Clauset, D. Kempe, and C. Moore. 2009. On the bias of traceroute sampling: Or, power-law degree distributions in regular graphs. *Journal of the ACM* 56, 4.

R. Agarwal. 2013. The space-stretch-time trade-off in distance oracles. Preprint.

R. Agarwal, M. Caesar, P. B. Godfrey, and B. Y. Zhao. 2012. Shortest paths in less than a millisecond. In *5th Workshop on Online Social Networks (WOSN)*.

R. Agarwal and P. B. Godfrey. 2013. Distance oracles for stretch less than 2. In *24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 526–538.

R. Agarwal, P. B. Godfrey, and S. Har-Peled. 2011. Approximate distance queries and compact routing in sparse graphs. In *30th IEEE International Conference on Computer Communications (INFOCOM)*. 1754–1762.

R. Agrawal and H. V. Jagadish. 1989. Materialization and incremental update of path information. In *5th International Conference on Data Engineering (ICDE)*. 374–383.

W. Aiello, F. R. K. Chung, and L. Lu. 2000. A random graph model for massive graphs. In *32nd ACM Symposium on Theory of Computing (STOC)*. 171–180.

M. Ajtai and R. Fagin. 1990. Reachability is harder for directed than for undirected finite graphs. *Journal of Symbolic Logic* 55, 1, 113–150. Announced at FOCS 1988.

S. B. Akers. 1960. The use of Wye-Delta transformations in network simplification. *Operations Research* 8, 3, 311–323. Announced at Rand Symposium on Mathematical Programming 1959.

T. Akiba, Y. Iwata, and Y. Yoshida. 2013. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 349–360.

T. Akiba, C. Sommer, and K. Kawarabayashi. 2012. Shortest-path queries for complex networks: Exploiting low tree-width outside the core. In *15th International Conference on Extending Database Technology (EDBT)*. 144–155.

N. Alon, P. D. Seymour, and R. Thomas. 1990. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society* 3, 4, 801–808. Announced at STOC 1990.

S. R. Arikati, D. Z. Chen, L. P. Chew, G. Das, M. H. M. Smid, and C. D. Zaroliagis. 1996. Planar spanners and approximate shortest path queries among obstacles in the plane. In *4th European Symposium on Algorithms (ESA)*. 514–528.

J. Arz, D. Luxen, and P. Sanders. 2013. Transit node routing reconsidered. In *12th International Symposium on Experimental Algorithms (SEA)*. 55–66.

B. Awerbuch, B. Berger, L. Cowen, and D. Peleg. 1998. Near-linear time construction of sparse neighborhood covers. *SIAM Journal on Computing* 28, 1, 263–277.

M. A. Babenko, A. V. Goldberg, A. Gupta, and V. Nagarajan. 2013. Algorithms for hub label optimization. In *40th International Colloquium on Automata, Languages, and Programming (ICALP)*. 69–80.

Z. K. Baker and M. Gokhale. 2007. On the acceleration of shortest path calculations in transportation networks. In *15th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 23–32.

C. L. Barrett, K. R. Bisset, R. Jacob, G. Konjevod, and M. V. Marathe. 2002. Classical and contemporary shortest path problems in road networks: Implementation and experimental analysis of the TRANSIMS router. In *10th European Symposium on Algorithms (ESA)*. 126–138.

Y. Bartal, L.-A. Gottlieb, T. Kopelowitz, M. Lewenstein, and L. Roditty. 2011. Fast, precise and dynamic distance queries. In *22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 840–853.

H. Bast. 2009. Car or public transport—two worlds. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*. 355–367.

H. Bast, S. Funke, D. Matijevic, P. Sanders, and D. Schultes. 2007a. In transit to constant time shortest-path queries in road networks. In *9th Workshop on Algorithm Engineering and Experiments (ALENEX)*.

H. Bast, S. Funke, P. Sanders, and D. Schultes. 2007b. Fast routing in road networks with transit nodes. *Science* 316, 5824, 566.

S. Baswana, A. Gaur, S. Sen, and J. Upadhyay. 2008. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In *35th International Colloquium on Automata, Languages and Programming (ICALP)*. 609–621.

S. Baswana and T. Kavitha. 2010. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM Journal on Computing* 39, 7, 2865–2896.

S. Baswana and S. Sen. 2006. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. *ACM Transactions on Algorithms* 2, 4, 557–577.

R. Bauer, T. Columbus, B. Katz, M. Krug, and D. Wagner. 2010a. Preprocessing speed-up techniques is hard. In *7th International Conference on Algorithms and Complexity (CIAC)*. 359–370.

R. Bauer, T. Columbus, I. Rutter, and D. Wagner. 2013. Search-space size in contraction hierarchies. In *40th International Colloquium on Automata, Languages, and Programming (ICALP)*. 93–104.

R. Bauer, G. D'Angelo, D. Delling, and D. Wagner. 2009. The shortcut problem—complexity and approximation. In *35th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*. 105–116.

R. Bauer and D. Delling. 2009. SHARC: Fast and robust unidirectional routing. *ACM Journal of Experimental Algorithmics* 14.

R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. 2010b. Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm. *ACM Journal of Experimental Algorithmics* 15, 2.3, 1–31.

A. Berger, D. Delling, A. Gebhardt, and M. Müller-Hannemann. 2009. Accelerating time-dependent multicriteria timetable information is harder than expected. In *9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*.

J. Boothroyd. 1967. Algorithms: Author's note on algorithms 22, 23, 24. *Computer Journal* 10, 3, 306–308.

J. Bourgain. 1985. On lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics* 52, 1–2, 46–52.

M. H. Bourgoin and E. M. J. Heurgon. 1969. Study and comparison of algorithms of the shortest path through planned experiments. In *Project Planning by Network Analysis*. 106–118.

A. Brady and L. Cowen. 2006. Compact routing on power law graphs with additive stretch. In *8th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 119–128.

E. Brunel, D. Delling, A. Gemsa, and D. Wagner. 2010. Space-efficient SHARC-routing. In *9th International Symposium on Experimental Algorithms (SEA)*. 47–58.

F. Buchholz. 2000. Hierarchische Graphen zur Wegesuche. Ph.D. thesis, Universität Stuttgart.

F. Buchholz and B. Riedhofer. 1997. Hierarchische Graphen zur kürzesten Wegesuche in planaren Graphen. Tech. rep. 13, Universität Stuttgart.

V. Bulitko, Y. Björnsson, N. R. Sturtevant, and R. Lawrence. 2010. Real-time heuristic search for pathfinding in video games. In *Artificial Intelligence for Computer Games*.

P. Butterworth, A. Otis, and J. Stein. 1991. The GemStone object database management system. *Communications of the ACM* 34, 10, 64–77.

S. Cabello. 2012. Many distances in planar graphs. *Algorithmica* 62, 1–2, 361–381.

S. Cabello, E. W. Chambers, and J. Erickson. 2012. Multiple-source shortest paths in embedded graphs. *arXiv abs/1202.0314*.

L. Cao, X. Zhao, H. Zheng, and B. Y. Zhao. 2011. Atlas: Approximating shortest paths in social graphs. Tech. rep. 2011-09, Department of Computer Science, University of California, Santa Barbara.

L. Chang, J. X. Yu, L. Qin, H. Cheng, and M. Qiao. 2012. The exact distance to destination in undirected world. *VLDB Journal* 21, 6, 869–888.

B. A. Chartres. 1967. Letter concerning Nicholson's paper. *Computer Journal* 10, 1, 118–119.

S. Chechik. 2013. Approximate distance oracle with constant query time. *arXiv abs/1305.3314*.

D. Z. Chen and J. Xu. 2000. Shortest path queries in planar graphs. In *32nd ACM Symposium on Theory of Computing (STOC)*. 469–478.

W. Chen, C. Sommer, S.-H. Teng, and Y. Wang. 2012. A compact routing scheme and approximate distance oracle for power-law graphs. *ACM Transactions on Algorithms* 9, 1, 4:1–26.

J. Cheng, Y. Ke, S. Chu, and C. Cheng. 2012. Efficient processing of distance queries in large graphs: a vertex cover approach. In *ACM SIGMOD International Conference on Management of Data*. 457–468.

J. Cheng and J. X. Yu. 2009. On-line exact shortest distance query processing. In *12th International Conference on Extending Database Technology (EDBT)*. 481–492.

J. Cheng, J. X. Yu, X. Lin, H. Wang, and P. S. Yu. 2006. Fast computation of reachability labeling for large graphs. In *10th International Conference on Extending Database Technology (EDBT)*. 961–979.

J. Cheng, J. X. Yu, X. Lin, H. Wang, and P. S. Yu. 2008. Fast computing reachability labelings for large graphs with high compression rate. In *11th International Conference on Extending Database Technology (EDBT)*. 193–204.

B. V. Cherkassky, A. V. Goldberg, and T. Radzik. 1996. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming* 73, 129–174.

L. Chindelevitch, D. Ziemek, A. Enayetallah, R. Randhawa, B. Sidders, C. Brockel, and E. Huang. 2011. Causal reasoning on biological networks: Interpreting transcriptional changes. In *15th International Conference on Research in Computational Molecular Biology (RECOMB)*. 34–37.

F. R. K. Chung and L. Lu. 2002. The average distances in random graphs with given expected degrees. *Internet Mathematics* 99, 15879–15882.

A. Clauset, C. R. Shalizi, and M. E. J. Newman. 2009. Power-law distributions in empirical data. *SIAM Review* 51, 4, 661–703.

E. Cohen. 1998. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing* 28, 1, 210–236.

E. Cohen, E. Halperin, H. Kaplan, and U. Zwick. 2003. Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing* 32, 5, 1338–1355.

H. Cohen and E. Porat. 2010. On the hardness of distance oracle for sparse graph. *arXiv abs/1006.1117*.

M. Costa, M. Castro, A. I. T. Rowstron, and P. B. Key. 2004. PIC: Practical internet coordinates for distance estimation. In *24th International Conference on Distributed Computing Systems (ICDCS)*. 178–187.

A. Cvetkovski and M. Crovella. 2009. Hyperbolic embedding and routing for dynamic graphs. In *28th IEEE International Conference on Computer Communications (INFOCOM)*. 1647–1655.

F. Dabek, R. Cox, F. Kaashoek, and R. Morris. 2004. Vivaldi: a decentralized network coordinate system. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. 15–26.

G. B. Dantzig. 1960. On the shortest route through a network. *Management Science* 6, 2, 187–190.

G. B. Dantzig. 1963. *Linear Programming and Extensions*. Princeton University Press.

A. Das Sarma, S. Gollapudi, M. Najork, and R. Panigrahy. 2010. A sketch-based distance oracle for web-scale graphs. In *3rd International Conference on Web Search and Web Data Mining (WSDM)*. 401–410.

D. Delling. 2009. *Engineering and Augmenting Route Planning Algorithms.* Ph.D. thesis, Universität Karlsruhe.

D. Delling, A. V. Goldberg, A. Nowatzyk, and R. F. Werneck. 2013. PHAST: Hardware-accelerated shortest path trees. *Journal of Parallel and Distributed Computing* 73, 7, 940–952.

D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. 2013a. *Customizable Route Planning in Road Networks*. Retrieved from http://research.microsoft.com/pubs/198358/crp_web_130724.pdf.

D. Delling, A. V. Goldberg, I. Razenshteyn, and R. F. F. Werneck. 2011. Graph partitioning with natural cuts. In *25th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*. 1135–1146.

D. Delling, A. V. Goldberg, and R. F. Werneck. 2013b. Hub label compression. In *12th International Symposium on Experimental Algorithms (SEA)*. 18–29.

D. Delling, M. Holzer, K. Müller, F. Schulz, and D. Wagner. 2009. High-performance multi-level routing. In *The Shortest Path Problem: 9th DIMACS Implementation Challenge*. Vol. 74. 73–92.

D. Delling, P. Sanders, D. Schultes, and D. Wagner. 2009a. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks - Design, Analysis, and Simulation*. 117–139.

D. Delling, P. Sanders, D. Schultes, and D. Wagner. 2009b. Highway hierarchies star. In *The Shortest Path Problem: 9th DIMACS Implementation Challenge*. Vol. 74. 141–174.

C. Demetrescu, A. V. Goldberg, and D. S. Johnson. 2008. Implementation challenge for shortest paths. In *Encyclopedia of Algorithms*.

E. Demir, C. Aykanat, and B. Barla Cambazoglu. 2008. Clustering spatial networks for aggregate query processing: A hypergraph approach. *Information Systems* 33, 1, 1–17.

N. Deo and C.-Y. Pang. 1984. Shortest path algorithms: Taxonomy and annotation. *Networks* 14, 257–323.

R. B. Dial, F. Glover, D. Karney, and D. Klingman. 1979. A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. *Networks* 9, 215–248.

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

H. Djidjev. 1996. Efficient algorithms for shortest path problems on planar digraphs. In *22nd International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*. 151–165.

H. N. Djidjev. 1985. A linear algorithm for partitioning graphs of fixed genus. *Serdica. Bulgariacae mathematicae publicationes* 11, 4, 369–387.

H. N. Djidjev and C. Sommer. 2011. Approximate distance queries for weighted polyhedral surfaces. In *19th European Symposium on Algorithms (ESA)*. 579–590.

D. Dor, S. Halperin, and U. Zwick. 2000. All-pairs almost shortest paths. *SIAM Journal on Computing* 29, 5, 1740–1759.

J. E. Doran. 1967. An approach to automatic problem-solving. *Machine Intelligence* 1, 105–124.

S. E. Dreyfus. 1969. An appraisal of some shortest-path algorithms. *Operations Research* 17, 3, 395–412.

Z. Dvorak, D. Král, and R. Thomas. 2010. Deciding first-order properties for sparse graphs. In *51st IEEE Symposium on Foundations of Computer Science (FOCS)*. 133–142.

M. Enachescu, M. Wang, and A. Goel. 2008. Reducing maximum stretch in compact routing. In *27th IEEE International Conference on Computer Communications (INFOCOM)*. 336–340.

D. Eppstein. 1999. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications* 3, 3.

D. Eppstein and M. T. Goodrich. 2008. Studying (non-planar) road networks through an algorithmic lens. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (GIS)*. 16.

P. Erdös. 1964. Extremal problems in graph theory. *Theory of Graphs and its Applications, Proceedings of the Symposium Held in Smolenice*, 29–36.

B. Eriksson, P. Barford, and R. D. Nowak. 2009. Estimating hop distance between arbitrary host pairs. In *28th IEEE International Conference on Computer Communications (INFOCOM)*. 801–809.

J. Fakcharoenphol and S. Rao. 2006. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences* 72, 5, 868–889.

J. Fakcharoenphol and T. Saranurak. 2010. Improving stretch bound on the Patrascu–Roditty distance oracle. Preprint.

M. Faloutsos, P. Faloutsos, and C. Faloutsos. 1999. On power-law relationships of the Internet topology. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. 251–262.

B. A. Farbey, A. H. Land, and J. D. Murchland. 1967. The cascade algorithm for finding all shortest distances in a directed graph. *Management Science* 14, 1, 19–28.

E. Feuerstein and A. Marchetti-Spaccamela. 1991. Dynamic algorithms for shortest paths in planar graphs. In *17th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*. 187–197.

R. W. Floyd. 1962. Algorithm 97: Shortest path. *Communications of the ACM* 5, 6, 345.

G. N. Frederickson. 1987. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing* 16, 6, 1004–1022.

M. L. Fredman and R. E. Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM* 34, 3, 596–615.

L. Fu, D.-H. Sun, and L. R. Rilett. 2006. Heuristic shortest path algorithms for transportation applications: State of the art. *Computers & Operations Research* 33, 11, 3324–3343.

G. Gallo. 1980. Reoptimization procedures in shortest path problems. *Rivista di Matematica per le Scienze Economiche e Sociali* 3, 3–13.

C. Gavoille and D. Peleg. 2003. Compact and localized distributed data structures. *Distributed Computing* 16, 2–3, 111–120.

C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. 2004. Distance labeling in graphs. *Journal of Algorithms* 53, 1, 85–112.

C. Gavoille and C. Sommer. 2011. Sparse spanners vs. compact routing. In *23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 225–234.

R. Geisberger, P. Sanders, D. Schultes, and D. Delling. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *7th International Workshop on Experimental Algorithms (WEA)*. 319–333.

R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. 2012. Exact routing in large road networks using contraction hierarchies. *Transportation Science* 46, 3, 388–404.

H. L. Gelernter. 1963. Realization of a geometry theorem proving machine. *Computers and Thought*.

J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. 1984. A separator theorem for graphs of bounded genus. *Journal of Algorithms* 5, 3, 391–407.

D. E. Gilsinn and C. Witzgall. 1973. A performance comparison of labeling algorithms for calculating shortest path trees. Technical Note 772, National Institute of Standards and Technology.

A. Goldberg, H. Kaplan, and R. F. F. Werneck. 2006. Reach for A*: Efficient point-to-point shortest path algorithms. In *8th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 129–143.

A. V. Goldberg. 2007. Point-to-point shortest path algorithms with preprocessing. In *33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*. 88–102.

A. V. Goldberg and C. Harrelson. 2005. Computing the shortest path: A* search meets graph theory. In *16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 156–165.

A. V. Goldberg, H. Kaplan, and R. F. Werneck. 2009. Reach for A*: Shortest path algorithms with preprocessing. In *The Shortest Path Problem: 9th DIMACS Implementation Challenge*. Vol. 74. 93–139.

A. V. Goldberg, H. Kaplan, and R. F. F. Werneck. 2007. Better landmarks within reach. In *6th International Workshop on Experimental Algorithms (WEA)*. 38–51.

A. V. Goldberg and R. F. F. Werneck. 2005. Computing point-to-point shortest paths from external memory. In *7th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 26–40.

B. Golden. 1976. Shortest-path algorithms: A comparison. *Operations Research* 24, 6, 1164–1168.

R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. 1998. Proximity search in databases. In *24th International Conference on Very Large Data Bases (VLDB)*. 26–37.

A. Gubichev, S. J. Bedathur, S. Seufert, and G. Weikum. 2010. Fast and accurate estimation of shortest paths in large graphs. In *19th ACM Conference on Information and Knowledge Management (CIKM)*. 499–508.

J. Gudmundsson, C. Levcopoulos, G. Narasimhan, and M. H. M. Smid. 2008. Approximate distance oracles for geometric spanners. *ACM Transactions on Algorithms* 4, 1.

S. Gupta, S. Kopparty, and C. Ravishankar. 2004. Roads, codes, and spatiotemporal queries. In *23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 115–124.

R. Gutman. 2004. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *6th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 100–111.

S. Har-Peled and M. Mendel. 2006. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing* 35, 5, 1148–1184.

P. E. Hart, N. J. Nilsson, and B. R. Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths in graphs. *IEEE Transactions of Systems Science and Cybernetics SSC-4*, 2, 100–107.

M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian. 1997. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences* 55, 1, 3–23.

J. Herzen, C. Westphal, and P. Thiran. 2011. Scalable routing easy as PIE: A practical isometric embedding protocol. In *19th IEEE International Conference on Network Protocols (ICNP)*. 49–58.

L. E. Hitchner. 1968. A comparative investigation of the computational efficiency of shortest path algorithms. Technical Report ORC 68-17, University of California at Berkeley.

A. J. Hoffman. 1963. On simple linear programming problems. In *Symposia in Pure Mathematics VII*. 317–327.

M. Holzer, F. Schulz, and D. Wagner. 2008. Engineering multilevel overlay graphs for shortest-path queries. *ACM Journal of Experimental Algorithmics* 13.

M. Holzer, F. Schulz, D. Wagner, and T. Willhalm. 2005. Combining speed-up techniques for shortest-path computations. *ACM Journal of Experimental Algorithmics* 10.

S. Honiden, M. E. Houle, C. Sommer, and M. Wolff. 2010. Approximate shortest path queries in graphs using Voronoi duals. *Transactions on Computational Science* 9, 28–53.

T. C. Hu. 1968. A decomposition algorithm for shortest paths in a network. *Operations Research* 16, 1, 91–102.

T. C. Hu. 1969. *Integer Programming and Network Flows*. Addison Wesley.

T. C. Hu and W. T. Torres. 1969. Shortcut in the decomposition algorithm for shortest paths in a network. *IBM Journal of Research and Development* 13, 4, 387–390.

T. Ikeda, M.-Y. Hsu, H. Imai, S. Nishimura, H. Shimoura, T. Hashimoto, K. Tenmoku, and K. Mitoh. 1994. A fast algorithm for finding better routes by AI search techniques. In *Vehicle Navigation and Information Systems Conference*. 291–296.

G. F. Italiano, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. 2011. Improved algorithms for min cut and max flow in undirected planar graphs. In *43rd ACM Symposium on Theory of Computing (STOC)*. 313–322.

R. Jin, Y. Xiang, N. Ruan, and D. Fuhry. 2009. 3-HOP: a high-compression indexing scheme for reachability query. In *35th SIGMOD International Conference on Management of Data (SIGMOD)*. 813–826.

R. Jin, Y. Xiang, N. Ruan, and H. Wang. 2008. Efficiently answering reachability queries on very large directed graphs. In *34th ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 595–608.

N. Jing, Y.-W. Huang, and E. A. Rundensteiner. 1996. Hierarchical optimization of optimal path finding for transportation applications. In *5th International Conference on Information and Knowledge Management (CIKM)*. 261–268.

N. Jing, Y.-W. Huang, and E. A. Rundensteiner. 1998. Hierarchical encoded path views for path query processing: An optimal model and its performance evaluation. *IEEE Transactions on Knowledge and Data Engineering* 10, 3, 409–432.

D. B. Johnson. 1977. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM* 24, 1, 1–13.

S. Jung and S. Pramanik. 1996. HiTi graph model of topographical roadmaps in navigation systems. In *12th International Conference on Data Engineering (ICDE)*. 76–84.

S. Kannan, M. Naor, and S. Rudich. 1992. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics* 5, 4, 596–603.

F. Karinthy. 1929. *Lancszemek*.

K. Kawarabayashi, P. N. Klein, and C. Sommer. 2011. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *38th International Colloquium on Automata, Languages and Programming (ICALP)*. 135–146.

K. Kawarabayashi, C. Sommer, and M. Thorup. 2013. More compact oracles for approximate distances in undirected planar graphs. In *24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 550–563.

M. Kitamura and M. Yamazaki. 1965. On the connection of the two shortest route systems. In *8th Japanese Road Conference*. 66–68.

V. L. Klee. 1964. A "string algorithm" for shortest path in directed networks. *Operations Research* 12, 3, 428–432.

P. N. Klein. 2002. Preprocessing an undirected planar network to enable fast approximate distance queries. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 820–827.

P. N. Klein. 2005. Multiple-source shortest paths in planar graphs. In *16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 146–155.

P. N. Klein, S. Mozes, and C. Sommer. 2013. Structured recursive separator decompositions for planar graphs in linear time. In *45th ACM Symposium on Theory of Computing (STOC)*. 505–514.

P. N. Klein, S. Mozes, and O. Weimann. 2010. Shortest paths in directed planar graphs with negative lengths: A linear-space $O(n \log^2 n)$-time algorithm. *ACM Transactions on Algorithms* 6, 2.

P. N. Klein and S. Subramanian. 1998. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica* 22, 3, 235–249.

J. M. Kleinberg. 2000. Navigation in a small world. *Nature* 406, 6798, 845.

J. M. Kleinberg, A. Slivkins, and T. Wexler. 2009. Triangulation and embedding using small sets of beacons. *Journal of the ACM* 56, 6.

E. Köhler, R. H. Möhring, and H. Schilling. 2005. Acceleration of shortest path and constrained shortest path computation. In *4th International Workshop on Experimental and Efficient Algorithms (WEA)*. 126–138.

L. Kowalik and M. Kurowski. 2006. Oracles for bounded-length shortest paths in planar graphs. *ACM Transactions on Algorithms* 2, 3, 335–363.

D. V. Krioukov, K. R. Fall, and X. Yang. 2004. Compact routing on Internet-like graphs. In *23rd IEEE International Conference on Computer Communications (INFOCOM)*.

R.-M. Kung, E. N. Hanson, Y. E. Ioannidis, T. K. Sellis, L. D. Shapiro, and M. Stonebraker. 1986. Heuristic search in database systems. In *1st International Workshop on Expert Database Systems*. 537–548.

J. B. H. Kwa. 1989. BS*: An admissible bidirectional staged heuristic search algorithm. *Artificial Intelligence* 38, 1, 95–109.

A. H. Land and S. W. Stairs. 1967. The extension of the cascade algorithm to large graphs. *Management Science* 14, 1, 29–33.

U. Lauther. 2004. An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. In *Geoinformation und Mobilität—von der Forschung zur praktischen Anwendung*. Vol. 22. 219–230.

S. Lim, C. Sommer, E. Nikolova, and D. Rus. 2012. Practical route planning under delay uncertainty: Stochastic shortest path queries. In *Robotics: Science and Systems VIII*. 249–256.

R. J. Lipton, D. J. Rose, and R. E. Tarjan. 1979. Generalized nested dissection. *SIAM Journal on Numerical Analysis* 16, 346–358.

R. J. Lipton and R. E. Tarjan. 1979. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics* 36, 2, 177–189.

J. Matousek. 1996. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics* 93, 1, 333–344.

J. Maue, P. Sanders, and D. Matijevic. 2009. Goal-directed shortest-path queries using precomputed cluster distances. *ACM Journal of Experimental Algorithmics* 14, 3.2–3.27.

K. Mehlhorn and P. Sanders. 2008. *Algorithms and data structures: the basic toolbox*. Springer.

M. Mendel and A. Naor. 2007. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society* 9, 2, 253–275.

M. Mendel and C. Schwob. 2009. Fast C-K-R partitions of sparse graphs. *Chicago Journal of Theoretical Computer Science*, 1–18.

S. Milgram. 1967. The small world problem. *Psychology Today* 1, 61–67.

G. Mills. 1966. A decomposition algorithm for the shortest route problem. *Operations Research* 14, 279–286.

N. Milosavljevic. 2012. On optimal preprocessing for contraction hierarchies. In *5th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS)*. 33–38.

G. J. Minty. 1957. A comment on the shortest-route problem. *Operations Research* 5, 5, 724.

R. H. Möhring, H. Schilling, B. Schütz, D. Wagner, and T. Willhalm. 2006. Partitioning graphs to speedup Dijkstra's algorithm. *ACM Journal of Experimental Algorithmics 11*.

G. Monge. 1781. Mémoire sur la théorie des déblais et de remblais. *Histoire de l'Académie Royale des Sciences de Paris, avec les Mémoires de Mathématique et de Physique pour la même année*, 666–704.

E. F. Moore. 1959. The shortest path through a maze. In *Annals of the Computation Laboratory of Harvard University*. Harvard University Press, 285–292.

S. Mozes and C. Sommer. 2012. Exact distance oracles for planar graphs. In *23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 209–222.

L. F. Muller and M. Zachariasen. 2007. Fast and compact oracles for approximate distances in planar graphs. In *15th European Symposium on Algorithms (ESA)*. 657–668.

J. D. Murchland. 1965. A new method for finding all elementary paths in a complete directed graph. Tech. rep. LBS-TNT-22, London Business School, Transport Network Theory Unit.

J. D. Murchland. 1967. The "once-through" method of finding all shortest distances in a graph from a single origin. Tech. rep. LBS-TNT-56, London Business School, Transport Network Theory Unit.

J. Nesetril and P. O. de Mendez. 2006. Linear time low tree-width partitions and algorithmic consequences. In *38th ACM Symposium on Theory of Computing (STOC)*. 391–400.

M. E. J. Newman. 2001. Scientific collaboration networks. II. shortest paths, weighted networks, and centrality. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 64*.

T. S. E. Ng and H. Zhang. 2002. Predicting internet network distance with coordinates-based approaches. In *21st IEEE International Conference on Computer Communications (INFOCOM)*.

T. A. J. Nicholson. 1966. Finding the shortest route between two points in a network. *The Computer Journal* 9, 3, 275–280.

Y. Nussbaum. 2011. Improved distance queries in planar graphs. In *12th International Symposium on Algorithms and Data Structures (WADS)*. 642–653.

D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. 2003. Query processing in spatial network databases. In *29th International Conference on Very Large Data Bases (VLDB)*. 802–813.

F. Papadopoulos, D. V. Krioukov, M. Boguñá, and A. Vahdat. 2010. Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In *29th IEEE International Conference on Computer Communications (INFOCOM)*. 2973–2981.

U. Pape. 1974. Implementation and efficiency of Moore-algorithms for the shortest route problem. *Mathematical Programming* 7, 1, 212–222.

M. Patrascu. 2011. Unifying the landscape of cell-probe lower bounds. *SIAM Journal on Computing* 40, 3, 827–847.

M. Patrascu and L. Roditty. 2010. Distance oracles beyond the Thorup-Zwick bound. In *51st IEEE Symposium on Foundations of Computer Science (FOCS)*. 815–823.

M. Patrascu, L. Roditty, and M. Thorup. 2012. A new infinity of distance oracles for sparse graphs. In *53rd IEEE Symposium on Foundations of Computer Science (FOCS)*. 738–747.

D. Peleg. 2000. Proximity-preserving labeling schemes. *Journal of Graph Theory* 33, 167–176.

I. S. Pohl. 1971. Bi-directional search. *Machine Intelligence* 6, 127–140.

M. Pollack and W. Wiebenson. 1960. Solutions of the shortest-route problem—a review. *Operations Research* 8, 2, 224–230.

E. Porat and L. Roditty. 2013. Preprocess, set, query! *Algorithmica* 67, 4, 516–528.

M. Potamias, F. Bonchi, C. Castillo, and A. Gionis. 2009. Fast shortest path distance estimation in large networks. In *18th ACM Conference on Information and Knowledge Management (CIKM)*. 867–876.

F. P. Preparata and M. I. Shamos. 1985. *Computational geometry: an introduction*.

M. Qiao, H. Cheng, L. Chang, and J. X. Yu. 2012. Approximate shortest distance computing: A query-dependent local landmark scheme. In *28th International Conference on Data Engineering (ICDE)*.

M. Qiao, H. Cheng, and J. X. Yu. 2011. Querying shortest path distance with bounded errors in large graphs. In *23rd International Conference on Scientific and Statistical Database Management (SSDBM)*. 255–273.

S. A. Rahman, P. Advani, R. Schunk, R. Schrader, and D. Schomburg. 2005. Metabolic pathway analysis web service (Pathway Hunter Tool at CUBIC). *Bioinformatics* 21, 7, 1189–1193.

B. Raney and K. Nagel. 2004. Iterative route planning for large-scale modular transportation simulations. *Future Generation Computer Systems* 20, 7, 1101–1118.

M. J. Rattigan, M. Maier, and D. Jensen. 2006. Using structure indices for efficient approximation of network properties. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 357–366.

M. J. Rattigan, M. Maier, and D. Jensen. 2007. Graph clustering with network structure indices. In *24th International Conference on Machine Learning (ICML)*. 783–790.

B. Riedhofer. 1997. Hierarchische Straßengraphen. M.S. thesis, Universität Stuttgart.

L. Roditty, M. Thorup, and U. Zwick. 2005. Deterministic constructions of approximate distance oracles and spanners. In *32nd International Colloquium on Automata, Languages and Programming (ICALP)*. 261–272.

M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 2011. 10 lessons from 10 years of measuring and modeling the Internet's autonomous systems. *IEEE Journal on Selected Areas in Communications* 29, 9, 1810–1821.

H. Samet, J. Sankaranarayanan, and H. Alborzi. 2008. Scalable network distance browsing in spatial databases. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 43–54.

A. L. Samuel. 1963. Some studies in machine learning using the game of checkers. *Computers and Thought*. 3, 211–229.

P. Sanders. 2009. Algorithm engineering—an attempt at a definition. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*. 321–340.

P. Sanders and D. Schultes. 2005. Highway hierarchies hasten exact shortest path queries. In *13th European Symposium on Algorithms (ESA)*. 568–579.

P. Sanders and D. Schultes. 2006. Engineering highway hierarchies. In *14th European Symposium on Algorithms (ESA)*. 804–816.

P. Sanders, D. Schultes, and C. Vetter. 2008. Mobile route planning. In *16th European Symposium on Algorithms (ESA)*. 732–743.

J. Sankaranarayanan and H. Samet. 2009. Distance oracles for spatial networks. In *25th International Conference on Data Engineering (ICDE)*. 652–663.

J. Sankaranarayanan, H. Samet, and H. Alborzi. 2009. Path oracles for spatial networks. *Proceedings of the VLDB Endowment* 2, 1, 1210–1221.

J. L. Santos. 2009. Real-world applications of shortest path algorithms. In *The Shortest Path Problem: 9th DIMACS Implementation Challenge*. Vol. 74. 1–19.

R. Schenkel, A. Theobald, and G. Weikum. 2004. HOPI: An efficient connection index for complex XML document collections. In *9th International Conference on Extending Database Technology (EDBT)*. 237–255.

R. Schenkel, A. Theobald, and G. Weikum. 2005. Efficient creation and incremental maintenance of the HOPI index for complex XML document collections. In *21st International Conference on Data Engineering (ICDE)*. 360–371.

J. P. Schmidt. 1998. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM Journal on Computing* 27, 4, 972–992.

D. Schultes. 2008. Route planning in road networks. Ph.D. thesis, Universität Karlsruhe.

D. Schultes and P. Sanders. 2007. Dynamic highway-node routing. In *6th International Workshop on Experimental Algorithms (WEA)*. 66–79.

F. Schulz, D. Wagner, and C. D. Zaroliagis. 2002. Using multi-level graphs for timetable information in railway systems. In *4th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 43–59.

M. Schwartz and T. E. Stern. 1980. Routing techniques used in computer communication networks. *IEEE Transactions on Communications* 28, 4, 539–552.

R. Sedgewick and J. S. Vitter. 1986. Shortest paths in Euclidean graphs. *Algorithmica* 1, 1, 31–48. Announced at FOCS 1984.

S. Sen. 2009. Approximating shortest paths in graphs. In *3rd International Workshop on Algorithms and Computation (WALCOM)*. 32–43.

Y. Shavitt and T. Tankel. 2008. Hyperbolic embedding of internet graph for distance estimation and overlay construction. *IEEE/ACM Transactions on Networking* 16, 25–36.

S. Shekhar, A. Fetterer, and B. Goyal. 1997. Materialization trade-offs in hierarchical shortest path algorithms. In *5th International Symposium on Advances in Spatial Databases (SSD)*. 94–111.

H. A. Smolleck. 1975. Application of fast sparse-matrix techniques and an energy estimation model for large transportation networks. Ph.D. thesis, University of Texas at Arlington.

H. A. Smolleck and M.-S. Chen. 1981. A new approach to near-optimal path assignment through electric-circuit modeling. *Networks* 11, 335–349.

C. Sommer. 2010. Approximate shortest path and distance queries in networks. Ph.D. thesis, The University of Tokyo.

C. Sommer, E. Verbin, and W. Yu. 2009. Distance oracles for sparse graphs. In *50th IEEE Symposium on Foundations of Computer Science (FOCS)*. 703–712.

B. Stout. 1999. Smart move: Intelligent path-finding. Retrieved from http://www.gamasutra.com/view/feature/3317/smart_move_intelligent_.php.

M. Thorup. 2004. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM* 51, 6, 993–1024.

M. Thorup and U. Zwick. 2001. Compact routing schemes. In *ACM Symposium on Parallelism in Algorithms and Architectures*. 1–10.

M. Thorup and U. Zwick. 2005. Approximate distance oracles. *Journal of the ACM* 52, 1, 1–24.

K. Tretyakov, A. Armas-Cervantes, L. García-Bañuelos, J. Vilo, and M. Dumas. 2011. Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. In *20th ACM Conference on Information and Knowledge Management (CIKM)*. 1785–1794.

P. Ungar. 1951. A theorem on planar graphs. *Journal of the London Mathematical Society*s 1–26, 4, 256–262.

D. van Vliet. 1978. Improved shortest path algorithms for transport networks. *Transportation Research* 12, 1, 7–20.

M. V. Vieira, B. M. Fonseca, R. Damazio, P. B. Golgher, D. de Castro Reis, and B. A. Ribeiro-Neto. 2007. Efficient search ranking in social networks. In *16th ACM Conference on Information and Knowledge Management (CIKM)*. 563–572.

D. Wagner and T. Willhalm. 2003. Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *11th European Symposium on Algorithms (ESA)*. 776–787.

D. Wagner and T. Willhalm. 2005. Drawing graphs to speed up shortest-path computations. In *7th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 17–25.

D. Wagner and T. Willhalm. 2007. Speed-up techniques for shortest-path computations. In *24th Symposium on Theoretical Aspects of Computer Science (STACS)*. 23–36.

D. Wagner, T. Willhalm, and C. D. Zaroliagis. 2005. Geometric containers for efficient shortest-path computation. *ACM Journal of Experimental Algorithmics 10*.

S. Warshall. 1962. A theorem on boolean matrices. *Journal of the ACM* 9, 1, 11–12.

F. Wei. 2010. TEDI: efficient shortest path query answering on graphs. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 99–110.

C. Wulff-Nilsen. 2010. Algorithms for planar graphs and graphs in metric spaces. Ph.D. thesis, University of Copenhagen.

C. Wulff-Nilsen. 2012. Approximate distance oracles with improved preprocessing time. In *23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 202–208.

C. Wulff-Nilsen. 2013. Approximate distance oracles with improved query time. In *24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 539–549.

A. C.-C. Yao. 1981. Should tables be sorted? *Journal of the ACM* 28, 3, 615–628.

C. Zaroliagis. 2008. Engineering algorithms for large network applications. In *Encyclopedia of Algorithms*.

F. B. Zhan and C. E. Noon. 1998. Shortest path algorithms: An evaluation using real road networks. *Transportation Science* 32, 1, 65–73.

X. Zhao, A. Sala, C. Wilson, H. Zheng, and B. Y. Zhao. 2010. Orion: shortest path estimation for large social graphs. In *3rd Conference on Online Social Networks (WOSN)*. 9–9.

X. Zhao, A. Sala, H. Zheng, and B. Y. Zhao. 2011. Efficient shortest paths on massive social graphs. In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. 77–86.

A. K. Ziliaskopoulos, D. Kotzinos, and H. S. Mahmassani. 1997. Design and implementation of parallel time-dependent least time path algorithms for intelligent transportation systems applications. *Transportation Research Part C: Emerging Technologies* 5, 2, 95–107.

U. Zwick. 2001. Exact and approximate distances in graphs—a survey. In *9th European Symposium on Algorithms (ESA)*. 33–48.

Received June 2012; revised September 2013; accepted September 2013