Shortest-Path Queries in Static Networks

Christian Sommer
csom@csail.mit.edu

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 paths (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 questions are posed 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 all 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 arguably high 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

1. INTRODUCTION

We review research on algorithms for the *point-to-point (approximate) shortest-path query problem*, restricted to discrete, static graphs with positive 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; Delling et al. 2011], Geographic Information Systems (GIS) and intelligent transportation systems [Jing et al. 1996], logistics, 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 paths (APSP)* problems in that there are two stages: 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 subsequent phase. After this preprocessing step, applications may ask *queries*, which should be answered efficiently.[1] 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; Bellman 1967; Ford 1956; Fredman and Tarjan 1987; Raman 1997; Thorup 1999; 2000; Hagerup 2000; Pettie and Ramachandran 2002; Goldberg 2008] 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; Seidel 1995; Alon et al. 1997; Shoshan and Zwick 1999; Zwick 2002; Pettie 2004; Pettie and Ramachandran 2005; Chan 2010].[2] Both solutions have their advantages and disadvantages: for the first strategy, no preprocessing is necessary but the query processing is rather slow; for the second 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 *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 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 pre-computation time, storage, query time, and approximation quality?

In this survey, we particularly focus on the tradeoff between the index size (storage) and the 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 major 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, computa-

[1]In computational geometry, this and similar problems are sometimes called *repetitive-mode* (as opposed to *single-shot*) problems [Preparata and Shamos 1985, p. 37].
[2]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).

tional 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, Goldberg, and Werneck [2011], Delling, Sanders, Schultes, and Wagner [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; Shekhar et al. 1997]. This survey is largely based on the author's Ph.D. thesis [Sommer 2010, Chapter 3].

1.1 General Techniques

In the following, let $G = (V, E)$ denote the input graph. On a high level, almost all the 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 serve to represent (potentially approximate) shortest paths. In the following, we refer to such nodes as landmarks. We distinguish three typical design choices:

(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 $\ell : E \to \mathbb{R}^+$), there is also the 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 is not only a "global" analogy, but often also corresponds to "local" decisions on where to use SSSP and where to use APSP.

A potential fourth design choice is to apply recursion at various stages.

Furthermore, almost always when a partial execution of an SSSP algorithm is involved, the designers may choose between *(a)* computing the shortest-path tree at preprocessing time and storing it (as a star), or *(b)* computing the tree at query time, thereby mediating between space and query time in a rather straightforward way. In subsequent sections, whenever possible and appropriate for the surveyed methods, we emphasize the design choices made.

2. THEORETICAL RESULTS ON DISTANCE ORACLES FOR GENERAL GRAPHS

For distance oracles applicable to general graphs, the quantitative tradeoff between the storage requirement 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.

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 non-negative, i.e., $\forall e \in E : \ell(e) \geqslant 0$.

Definition 2.1. An *((α, β)–approximate) distance oracle* for a class of graphs $\mathcal{G}$ consists of a data structure and a query algorithm with the following characteristics:

—The *preprocessing time* is the worst-case time[3] required to construct the data structure for any $G \in \mathcal{G}$.

—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

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

The stretch is also called *distortion*.

In addition to the worst-case measures, the average or expected query time (in particular for client-server systems with multiple servers) 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

[3]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.

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 be proportional to the number of edges on the path, making comparisons 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.

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) the $s-t$–distance.

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

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, which are considered "hard" [Ajtai and Fagin 1990; Patrascu 2011]. The results for *undirected* graphs can be summarized as follows.

—For general graphs, the tradeoffs between space and stretch of existing distance oracles (Section 2.2) are essentially best possible [Thorup and Zwick 2005], up to a widely believed and partially proven conjecture (Erdős' *Girth*[4] *Conjecture* [1964]) in extremal graph theory.

—For sparse graphs, the situation is less clear. Distance oracles with constant stretch and query time require superlinear space [Sommer et al. 2009]. For multiplicative stretch less than 2, quadratic space requirements are conjectured [Cohen and Porat 2010; Patrascu and Roditty 2010]. For multiplicative stretch less than 3, in particular for stretch $\alpha = 3 - 2/\ell$ (for an integer $\ell > 0$), space $\tilde{\Omega}(n^{1+1/(2-1/\ell)})$ is conjectured [Patrascu et al. 2011].

—For special classes of graphs such as planar, bounded-genus, minor-free, and bounded-doubling-dimension graphs no non-trivial lower bounds are known.

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 [Dor et al. 2000, Theorem 5.1] provide a reduction between all-pairs approximate shortest paths and matrix multiplication. If the preprocessing algorithm is faster than the time required for matrix multiplication, then query time $o(m/n)$

[4]The *girth* of a graph is defined as the length of its shortest cycle.

Lower Bound	Condition(s)		Assumption / Proof	Reference
Space	Stretch	Query Time		
$\Omega(\min\{m, n^{1+1/k}\})$	$\alpha < 2k + 1$		girth conjecture [Erdős 1964]	[Thorup and Zwick 2005, Prop. 5.1]
$n^{1+\Omega(1/(\alpha \mathrm{t}))}$	α	t	cell-probe model [Yao 1981]	[Sommer et al. 2009]
$\Omega(n^2)$	$\alpha < 2$	$\mathrm{t} = O(1)$	set intersection, conjecture	[Cohen and Porat 2010; Patrascu and Roditty 2010]
$\tilde{\Omega}(n^{1+1/(2-1/\ell)})$	$\alpha = 3 - 2/\ell$	$\mathrm{t} = O(1)$	set intersection, conjecture	[Patrascu et al. 2011]
$\Omega(n^{3/2})$	$\alpha = 1$		distributed labels	[Gavoille et al. 2004]

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 in the third column. The lower bound of Thorup and Zwick [2005, Proposition 5.1] establishes a relationship between stretch and space, conjectured to be tight for all integers $k \geqslant 1$ (*Girth Conjecture* of Erdős [1964] saying that dense graphs with large girth exist), proven for $k \in \{1, 2, 3, 5\}$. (see also [Matousek 1996] for a similar lower-bound construction). The lower bound of Sommer, Verbin, and Yu [2009] relates the space with both stretch and query time in the cell-probe model [Yao 1981], building on techniques of Pătraşcu [2011]. The lower bound of Cohen and Porat [2010] (see also Pătraşcu and Roditty [2010] and the extension of Pătraşcu, Roditty, and Thorup [2011]) is based on a conjecture on the hardness of *set intersection queries*. The lower bound of Gavoille, Peleg, Pérennes, and Raz [2004] is for distance labels and holds even for degree–3 graphs.

would imply a faster algorithm for boolean matrix multiplication (by answering $O(n^2)$ distance queries).[5]

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 only allowed to access 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].

2.2 Distance Oracles for General Graphs

For an overview of distance oracles for general *undirected* graphs, see Table II. For the stretch vs. space tradeoff, see Fig. 1. We first provide a brief summary.

—*Odd integral stretch* $\geqslant 3$: 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 $O(k)$ [Thorup and Zwick 2005] (improved query time $O(\lg k)$ in [Wulff-Nilsen 2012b]). For many values of k, the distance oracle can be computed in time almost proportional to the space requirements [Wulff-Nilsen 2012a], obtaining an oracle that is optimal (assuming Erdős' girth conjecture) for sufficiently dense graphs with $m = \Omega(n^{1+c/\sqrt{k}})$ and $k = O(1)$. Other improvements of the preprocessing time include [Baswana and Sen 2006; Baswana et al. 2008; Sen 2009; Baswana and Kavitha 2010].

—*Constant query time*: If stretch $\alpha = (2 + \epsilon)k$ (instead of $2k - 1$ as above) is allowed, query time $O(1)$ (instead of $O(k)$ or $O(\lg k)$) can be achieved [Wulff-Nilsen 2012b] (based on an earlier construction providing a distance oracle with

[5]If the preprocessing algorithm is not combinatorial, query time $o(n^{\omega-2})$, where ω denotes the exponent for matrix multiplication).

constant query time and stretch $\alpha = O(k)$ [Mendel and Naor 2007]).

—*Sparse graphs*: 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]. If query time $O(m^{\theta})$ is allowed, there is a distance oracle using space $O(m + m^{1-\theta}n)$ for any $\theta \in (0, \frac{1}{2}]$ [Agarwal et al. 2011].

—*Unit-length graphs*: For unit-length graphs, there is a $(2,1)$–approximate distance oracle using space $O(n^{5/3})$ [Patrascu and Roditty 2010]. The tradeoff extends to general $k \geqslant 2$ with stretch $(2k-2, 1)$ and space $\tilde{O}(n^{1+2/(2k-1)})$ [Abraham and Gavoille 2011]. There is also a method with stretch strictly less than 2 and sublinear query time (albeit only slightly so) [Porat and Roditty 2011].

In a seminal work, Thorup and Zwick [2005] provide both the lower and the matching upper bound: for their oracles, the tradeoff between space complexity and stretch is essentially tight.[6] For any integer stretch parameter $k \geqslant 1$, their randomized algorithm (deterministic version due to Roditty, Thorup, and Zwick [2005]) can preprocess a graph with n nodes and m edges in time $\tilde{O}(mn^{1/k})$ to construct a distance oracle of size $O(kn^{1+1/k})$ with query time $O(k)$ and stretch $\alpha = 2k - 1$. Note that for $k = 1$ this yields an exact distance oracle with preprocessing time $O(mn)$, which equals the running time of many APSP algorithms.

The relationship between size and stretch is almost optimal with respect to the lower bound implied by the girth conjecture. The time complexities (query and preprocessing) are not tight. Both have been improved upon independently. Wulff-Nilsen [2012b], based on earlier work of Mendel and Naor [2007] (see also [Naor and Tao 2010; Naor 2012]), reduces the query time down to $O(1)$, sacrificing an ϵ–factor in the stretch. Baswana and Kavitha [2010], Baswana and Sen [2006], and Baswana, Gaur, Sen, and Upadhyay [2008] (unit-length graphs only) improve the performance of the preprocessing algorithms to quadratic and even subquadratic time (at the cost of constant additive stretch for some oracles). Wulff-Nilsen [2012a] gave further improvements of the preprocessing time using spanners [Baswana and Sen 2007; Baswana and Kavitha 2010].

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 Mendel and Naor [2007] improves the query time to $O(1)$. It would be very useful 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 interesting open question whether such oracles exist for sparse graphs (Agarwal et al. [2011] refer to such oracles as "the holy grail").

Pătraşcu and Roditty [2010] and Agarwal, Godfrey, and Har-Peled [2011] found distance oracles with multiplicative stretch $\alpha = 2$. For both constructions, the space vs. query time tradeoff depends on the number of edges m, providing a better tradeoff for sparser graphs. They also provide linear-space data structures with stretch 3 and query time $O(m^{\epsilon})$ for $\epsilon > 0$ (see Table II and the original papers for details).

[6]To be precise, their tradeoff is provably tight for the values of k for which Erdős' Girth Conjecture has been proven ($k \in \{1, 2, 3, 5\}$). See [Thorup and Zwick 2005, Table II] for an overview of results on the Girth Conjecture.

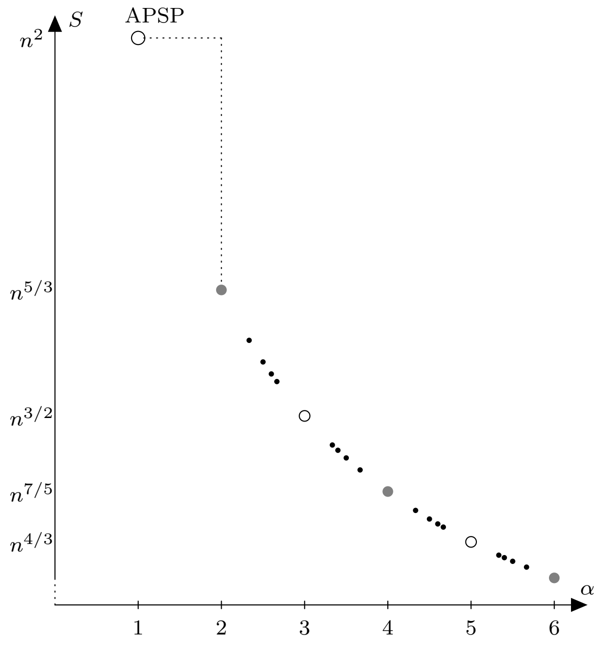

Fig. 1. Distance oracles (with $O(1)$ query time) for sparse graphs (n nodes, $m = O(n)$ edges): the tradeoff between stretch [α] and space [S], depicted using a linear scale for the stretch and a logarithmic scale for the space. Oracles with odd integral stretch (white circles) are due to Thorup and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

A distance oracle for general graphs (which also works for directed graphs) with a different type of worst-case guarantee on the space complexity is due to Cohen, Halperin, Kaplan, and Zwick [2003]. Their method, called 2*–hop covers*, is an *exact* distance labeling scheme, working 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 smallest such cover (*cf.* SetCover for the set of all shortest paths). Their results have been used successfully for road networks (see Section 3.2.2) and also for more complex networks (see Section 4).

2.2.1 *Techniques.* Many distance oracles approximate distances by *triangulation* using a sublinear number of of *landmarks* (also termed *beacons*), selected by random sampling (see also Section 1.1). Nodes store distances to all landmarks $l \in L$ (stars from all landmarks) and query results are computed as $\min_{l \in L} d(s, l) + d(l, t)$.

Preprocessing Time	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
poly	$O(m^{1/3}n^{4/3})$	$O(1)$	$\alpha = 2$	[Patrascu and Roditty 2010]
poly	$O(m + m^{1-\theta}n)$	$O(m^\theta)$	$\alpha = 2$	[Agarwal et al. 2011], $\theta \in (0, 1/2]$
$\tilde{O}(m + n^{1+c/\sqrt{k}})$	$O(kn^{1+1/k})$	$O(\lg k)$	$\alpha = 2k - 1$	[Wulff-Nilsen 2012a; 2012b], 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]
poly	$O(m)$	$O(m^{1/2})$	$\alpha = 3$	[Patrascu and Roditty 2010], implicit
$\tilde{O}(mn^{1/k})$	$O(kn^{1+1/k})$	$O(1/\epsilon)$	$\alpha = (2 + \epsilon)k$	[Wulff-Nilsen 2012b], $\epsilon \in (0, 1]$
poly	$O(m)$	$O(m^{1/(k+1)})$	$\alpha = 4k - 1$	[Agarwal et al. 2011]
poly	$O(m + m^{(1-\theta)(1+1/k)})$	$O(m^\theta)$	$\alpha = 4k - 1$	[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], $k \geqslant 3$
$\tilde{O}(mn^{1/k})$	$O(n^{1+1/k})$	$O(1)$	$\alpha = O(k)$	[Mendel and Naor 2007; Mendel and Schwob 2009]
poly	$\tilde{O}(n^{1+1/(k\pm 1/\ell)})$	$O(1)$	$\alpha = 2k - 1 \pm 2/\ell$	[Patrascu et al. 2011], sparse graphs $m = O(n)$
$O(mn)$	$O(nm^{1-\epsilon/6})$	$O(m^{1-\epsilon/6})$	$(1 + \epsilon, 0)$	[Porat and Roditty 2011], $\epsilon > 0$
poly	$O(n^{5/3})$	$O(1)$	$(2,1)$	[Patrascu and Roditty 2010]
poly	$\tilde{O}(n^{1+2/(2k-1)})$	$O(k)$	$(2k - 2, 1)$	[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 $\alpha > 3$

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 $o(n^2)$.

If any landmark $l \in L$ lies on a shortest path from s to t, the exact distance can be recovered (this is guaranteed by the method based on 2–hop covers [Cohen et al. 2003]). However, most schemes do not guarantee that shortest distances are retrieved. Furthermore, instead of minimizing over all 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 is often not accurate enough if the query nodes are very close to each other (triangulation may incur a detour that is arbitrarily large with respect to the shortest-path distance). Node pairs at short distances are often handled by precomputing and storing shortest-path trees and distances for *open balls* (algorithms limit the sizes of these balls, stars from each node) and by ball intersections (algorithms carefully control the number of balls that intersect). For example, in the distance oracle of Thorup and Zwick [2005], each node u (with nearest landmark l_u) stores distances to all nodes v within distance less than $d(u, l_u)$, i.e., to all the nodes in the open ball $B(u) = \{v \in V : d(u, v) < d(u, l_u)\}$. Then, 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: $d(s, l_t) + d(l_t, t) \leqslant d(s, t) + d(l_t, t) \leqslant 3d(s, t)$.

If landmarks are chosen independently at random with probability p, then the expected ball size is $1/p$. Furthermore, since storing stars from all landmarks requires space proportional to $n^2 p$ in expectation, we can balance the total sizes of all balls to be roughly equal to the space required to store np stars by setting $p := 1/\sqrt{n}$.

Pătraşcu and Roditty [2010] (see also [Agarwal et al. 2011]) found a way to further improve the stretch, at the cost of increasing the space requirements. They 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*, i.e. $B(s) \cap B(t) \neq \emptyset$. We provide some intuition for the simplified construction in [Abraham and Gavoille 2011]. Let the *cluster* of a node v contain all the nodes u that have v in their balls, i.e. $C(v) := \{u : v \in B(u)\}$ — 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 again stores distances to landmarks, to all nodes in its ball $u \in B(v)$, and to all nodes whose ball has a non-empty intersection with $B(v)$ (bounded by $\tilde{O}(n^{2/3})$).

To improve the preprocessing time, many algorithms use *spanners* [Peleg and Schäffer 1989; Althöfer et al. 1993; Bollobás et al. 2003; Elkin and Peleg 2004; Baswana and Sen 2007; Elkin 2008] (see also *emulators* [Thorup and Zwick 2006; Woodruff 2006] and *distance-equivalent networks* [Hu 1969]). Others use metric embeddings and Ramsey partitions [Linial and Saks 1993; Calinescu et al. 2004; Bartal et al. 2005].

3. SHORTEST-PATH QUERIES FOR PLANAR GRAPHS, ROUTE PLANNING FOR ROAD NETWORKS, AND QUERY PROCESSING FOR SPATIAL 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 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.

In this section, we focus on planar graphs and road networks. 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], and planar graphs are well studied in theoretical computer science.

We first give a brief historical overview and then we survey exact and approximate methods for planar graphs and methods for road networks, subdividing approaches 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 an important part of research on shortest-path algorithms. Starting from classical single-source shortest-path algorithms it has been noted that, if

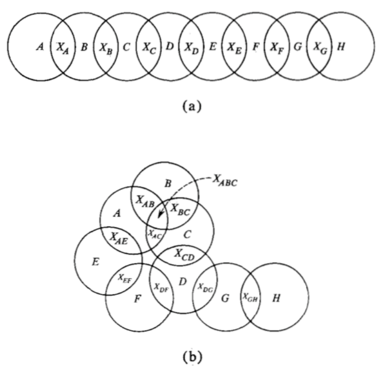

Figure 4 Network decomposition: (a) linearly overlapping sets and (b) nonlinearly overlapping sets.

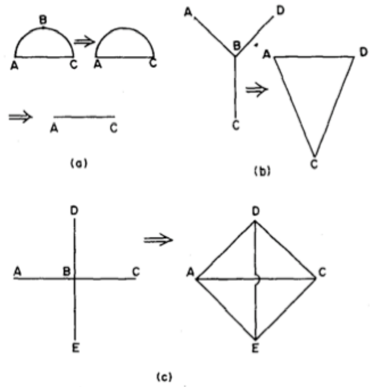

Fig. 5. Spider web transformations for : (a) $e_B = 2$, (b) $e_B = 3$ and (c) $e_B = 4$.

Fig. 2. Illustrations of early preprocessing techniques, extracted from the corresponding papers: *Left:* The network decomposition technique as originally depicted by Hu and Torres [1969]. First, the network is decomposed 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: The *spider web* transformations of van Vliet [1978], 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.*

an application only requires point-to-point distances, many SSSP algorithms can be stopped early. Furthermore, SSSP algorithms may run faster when executed from the source and the target simultaneously — this important technique is also called *bidirectional* search [Nicholson 1966; Boothroyd 1967; Chartres 1967; Murchland 1967; Pohl 1971; Sint and de Champeaux 1977; de Champeaux 1983; Luby and Ragde 1989]. 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. The list can be sorted by starting nodes (such a representation is sometimes termed *forward star form*). 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; Yen 1971; Glover et al. 1974; Lansdowne and Robinson 1982] was used to speed up APSP algorithms on sparse networks (see Fig. 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.[7]

Smolleck models the network by an electric circuit, wherein each edge is mapped to an impedance to efficiently answer *approximate* shortest-path queries.[8] Van Vliet introduced[9] heuristics termed *spider web techniques* [van Vliet 1978, Section 6], which contract nodes and introduce shortcut edges (Fig. 2). Van Vliet in some sense combined APSP and SSSP techniques into a query method, illustrating the tradeoff that shortest-path query methods address.[10]

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; Gelperin 1977; Kung et al. 1986] based on the Euclidean distance have been used to guide the search towards the target [Sedgewick and Vitter 1986]. These heuristics are quite well known, rather easy to implement, and used in numerous implementations. More recent techniques (see Section 3.2 below), however, yield substantially improved speedups.

3.1 Theoretical Results on Distance Oracles for Planar, Bounded-Genus, and Minor-Free Graphs

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. One might also argue that planar graphs form the class of graphs that is closest to road networks *and* that theoreticians know how to design efficient algorithms for. We give a brief overview; summaries can be found in Table III and Fig. 3 for exact and Table IV for approximate shortest paths.

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.

[7]An earlier approach of Bazaraa and Langley [1974] was to preprocess a graph in order to eliminate *negative* lengths such that Dijktra's algorithm can be applied.
Also, a method of Gallo [1980] effectively uses a prior shortest-path computation to speed up future shortest-path computations (see also [Klein 2005]). If, however, the next computation is "far" from the previous one, speedups may be minimal. A possible shortest-path query method might be to extend Gallo's method to allow for a selection among multiple prior shortest-path trees to speed up the query algorithm.
[8]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; the preprocessing time is reported to be 1,000 times slower than the query time.
[9]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.
[10]According to the article, van Vliet's contraction techniques decrease the CPU time for multiple queries by approximately 25%.

For exact shortest-path queries, the currently best result in terms of the tradeoff between space and query time is due to Fakcharoenphol and Rao [2006] (slight improvements in [Mozes and Sommer 2012]). Their data structure of space $\tilde{O}(n)$ can be constructed in time $\tilde{O}(n)$ and processes queries in time $\tilde{O}(\sqrt{n})$. For similar claims but slower preprocessing times, see also [Buchholz 2000]. The fastest preprocessing time is obtained by an algorithm of Klein, Mozes, and Weimann [2010] (implicit, using [Klein 2005]).

Some distance oracles have better query times. Djidjev [1996], improving upon earlier work of 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 $O(n^2/S)$. The preprocessing time is $O(n\sqrt{S})$ for $S \in [n, n^{3/2}]$ and $O(S)$ for larger S. Concurrent results for smaller ranges can be found in [Arikati et al. 1996; Buchholz and Riedhofer 1997; Riedhofer 1997]. These constructions [Djidjev 1996; Arikati et al. 1996; Buchholz and Riedhofer 1997; Riedhofer 1997] only use 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 (but potentially slower preprocessing times) also exist for bounded-genus and minor-free graphs.

For a smaller range, further exploiting planarity, Djidjev also proves that, for any $S \in [n^{4/3}, n^{3/2}]$, there is an exact distance oracle with preprocessing time $O(n\sqrt{S})$, space $O(S)$, and query time $\tilde{O}(n/\sqrt{S})$. Chen and Xu [2000], extending the range, prove that, for any $S \in [n^{4/3}, n^2]$, there is an exact distance oracle using space $O(S)$ with preprocessing time $O(n\sqrt{S})$ and query time $\tilde{O}(n/\sqrt{S})$. Cabello [2011], mainly improving the preprocessing time, proves that, for any $S \in [n^{4/3}, n^2]$, there is an exact distance oracle with preprocessing time and space $O(S)$ and query time $\tilde{O}(n/\sqrt{S})$. Compared to Djidjev's construction, the query time is slower by a logarithmic factor but the range for S is larger and the preprocessing time is faster. Nussbaum [2011] provides a data structure that maintains both the tradeoff from [Djidjev 1996; Chen and Xu 2000] and the fast preprocessing time from [Cabello 2011]. Nussbaum also provides a different data structure with a "clean" tradeoff between space $O(S)$ and query time $O(n/\sqrt{S})$, however spending time $\tilde{O}(S^{3/2}/\sqrt{n})$ in the preprocessing phase. Mozes and Sommer [2012] prove the bounds for essentially the entire range of S, while maintaining preprocessing time $\tilde{O}(S)$. These methods with faster query time $\tilde{O}(n/\sqrt{S})$ make use of *non-crossing* and *Monge* properties [Monge 1781; Hoffman 1963] of planar graphs. It is unclear whether these techniques extend to bounded-genus graphs.

The only lower bounds known are for distance labels, proving that total label length $\Omega(n^{4/3})$ is required for exact distance labels [Gavoille et al. 2004, Theorem 3.8] and $\Omega(n^{3/2})$ 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 $Q = \tilde{O}(n/\sqrt{S})$.

Methods for planar graphs also follow the overall approach described in Section 1.1: roughly speaking, the set of landmarks is chosen as the nodes of a separator (often recursively) and distances among all pairs of separator nodes (within

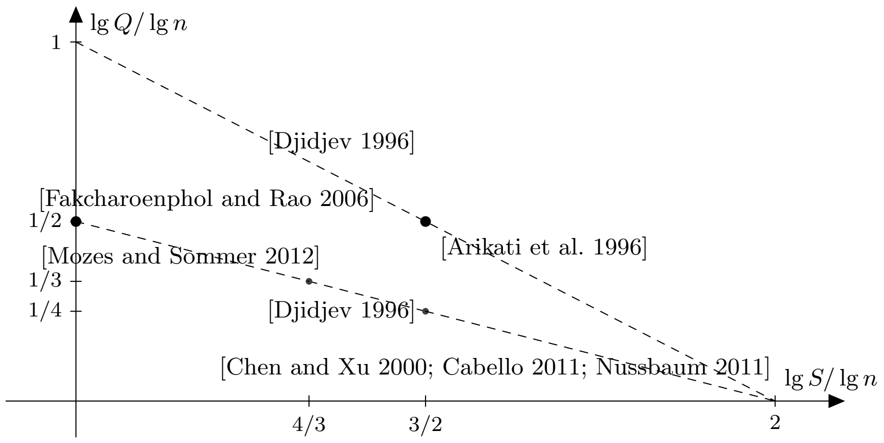

Fig. 3. Distance oracles for planar graphs: The figure illustrates the tradeoff between the Space [S] and the Query time [Q] for different data structures on a doubly logarithmic scale, ignoring constant and logarithmic factors.
The upper line represents the $Q = n^2/S$ tradeoff (completely covered by [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; 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; Cabello 2011] (query time improvements by [Nussbaum 2011]), the point $S = n$ is covered by [Fakcharoenphol and Rao 2006] (see also [Buchholz 2000]), and the full range is covered by [Mozes and Sommer 2012].

a recursive level) are stored explicitly. Distances from the remaining nodes to the separator are either stored, computed at query time using an SSSP algorithm, or retrieved using so-called MSSP data structures [Klein 2005] (see also Section 3.1.3). In order to obtain an exact answer, the minimum among all the paths passing through a landmark node on a *relevant* separator are considered.

Experimental results indicate that real-world road networks appear to have recursive separators of size proportional to roughly $\sqrt[3]{n}$ (as opposed to $O(\sqrt{n})$–sized separators for general planar graphs) [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).

3.1.2 *Approximations.* To obtain constant or polylogarithmic query times while maintaining almost linear space requirements, approximate distance oracles are considered. For an overview of approximate distance oracles for planar graphs, see Table IV. Thorup [2004] presents efficient $(1 + \epsilon)$–approximate distance oracles for planar digraphs. 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; Klein 2002] for related constructions). For directed graphs, the construction is 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. Thorup's oracle (for undirected graphs) has also been implemented and tested for road networks [Muller and Zachariasen 2007]. The results however indicate that, for these road networks, it is not competitive with the specialized methods to be discussed in Section 3.2. Kawarabayashi, Klein, and Sommer [2011] extend Thorup's results to undirected graphs embedded in a surface of Euler genus g, providing a $(1+\epsilon)$–approximate distance oracle with query time $O(g/\epsilon)$, space $O(n(g+\lg n)/\epsilon)$, and preprocessing time $O(n(\lg n)^3\epsilon^{-2} + n(\lg n)g/\epsilon)$. 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]. For any $\epsilon > 0$, their preprocessing algorithm runs in polynomial time, creates a $(1+\epsilon)$–approximate distance oracle using space $O(n\epsilon^{-1}\lg n)$ with query time $O(\epsilon^{-1}\lg n)$. 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 (some techniques are similar to those used for exact tradeoffs illustrated in Fig. 3). Improved tradeoffs have been announced [Sommer 2011].

3.1.3 *Restricted queries.* There are exact distance oracles for two different types of 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] gave a distance oracle that preprocesses a graph and a given face in time and space $O(n \lg n)$ to answer distance queries between any node on the specified face and any other node in time $O(\lg n)$. This data structure is also referred to as a *Multiple Source Shortest Paths* (MSSP) data structure and it is used as an ingredient in other distance oracles. Earlier, Schmidt [1998] had found a similar data structure for grid graphs. Mozes and Sommer [2012] provide an extension that can handle an arbitrary cycle on $O(\sqrt{n})$ nodes instead of a single face but only batched queries can be answered. Cabello and Chambers [2012] give a different and simpler algorithm, which also extends Klein's result to graphs with genus g; they provide an $O(g^2 n \lg n)$–time algorithm to represent the shortest-path tree from all the nodes on one specified face. Any distance from or to a node on this face can be obtained in time $O(\lg n)$.

3.1.4 *Others.* Shortest-path queries for planar graphs have also been considered in the external memory model [Hutchinson et al. 2003] and as part of spatiotemporal queries for a certain class of planar graphs [Gupta et al. 2004].

Preprocessing Time	Space	Query Time	Reference	Restriction (if any)
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]	
$O(S)$	$O(S)$	$O(n^2/S)$	[Djidjev 1996, Thm. 3]	$S \in [n^{3/2}, n^2]$
$O(n\sqrt{S})$	$O(S)$	$O(n^2/S)$	[Djidjev 1996, Thm. 4]	$S \in [n, n^{3/2}]$
$O(n \lg^2 n)$	$O(n \lg n)$	$O(\sqrt{n} \lg^2 n)$	[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],	constant $\epsilon > 0$
$O(n \lg\lg n)$	$O(n)$	$O(n/poly(\lg n))$	Implicit[11] in [Italiano et al. 2011]	
$O(n\sqrt{S})$	$O(S)$	$O((n/\sqrt{S}) \lg n)$	[Djidjev 1996, Thm. 5]	$S \in [n^{4/3}, n^{3/2}]$
$O(n^3/S)$	$O(S)$	$O((S/n) \lg n)$	[Chen and Xu 2000]	$S \in [n^{4/3}, n^{3/2}]$
$O(n\sqrt{S})$	$O(S)$	$O((n/\sqrt{S}) \lg(n/\sqrt{S}))$	[Chen and Xu 2000]	$S \in [n^{3/2}, n^2]$
$O(S \lg n)$	$O(S)$	$O((n/\sqrt{S}) \lg(n/\sqrt{S}))$	[Nussbaum 2011, Thm. 4.1]	$S \in [n^{4/3}, n^2]$
$O(S)$	$O(S)$	$O((n/\sqrt{S}) \lg^{1.5} n)$	[Cabello 2011, Thm. 12]	$S \in [n^{4/3} \lg^{1/3} n, n^2]$
$O(S\sqrt{S/n} \lg^2 n)$	$O(S)$	$O(n/\sqrt{S})$	[Nussbaum 2011, Thm. 5.2]	$S \in [n^{4/3}, n^2]$
$O(S \lg^2 n)$	$O(S)$	$O((n/\sqrt{S}) \lg^{2.5} n)$	[Mozes and Sommer 2012]	$S \in [n \lg\lg n, n^2]$

Table III. Time and space complexities of exact distance oracles for directed planar graphs. The tradeoff between space and query time is illustrated in Fig. 3.
In the large-space result of Chen and Xu [2000], an additive inverse-Ackermann term in the query time is suppressed in this table.

Preprocessing Time	Space	Query Time	Reference
$O(n\epsilon^{-2} \lg(nN) \lg^3 n)$	$O(n\epsilon^{-1} \lg(nN) \lg n)$	$O(\epsilon^{-1} + \lg\lg(nN))$	[Thorup 2004, Thm. 3.16]
$O(n(\epsilon^{-1} + \lg n) \lg(nN) \lg n)$	$O(n\epsilon^{-1} \lg(nN) \lg n)$	$O(\lg n(\epsilon^{-1} + \lg\lg(nN)))$	[Klein 2005, Sec. 7]
$O(n\epsilon^{-2} \lg(nN) \lg^3 n)$	$O(n)$	$O((\epsilon^{-1} \lg(nN) \lg n)^2)$	[Kawarabayashi et al. 2011]
$O(n\epsilon^{-2} \lg^3 n)$	$O(n\epsilon^{-1} \lg n)$	$O(\epsilon^{-1})$	[Thorup 2004, Thm. 3.19]
$O(n\epsilon^{-1} \lg^2 n)$	$O(n\epsilon^{-1} \lg n)$	$O(\epsilon^{-1} \lg n)$	[Thorup 2004, Implicit]
$O(n \lg^2 n)$	$O(n)$	$O(\epsilon^{-2} \lg^2 n)$	[Kawarabayashi et al. 2011]

Table IV. Time and space complexities of $(1+\epsilon)$–approximate distance oracles for planar graphs. N denotes the largest integer length. The upper part lists results for directed, the lower part for undirected planar graphs.

Exact distance oracles have also been found for graphs with bounded tree-width [Chaudhuri and Zaroliagis 2000; Farzan and Kamali 2011; Akiba et al. 2012], for planar graphs with few hammocks [Djidjev et al. 2000; Chen and Xu 2000], for interval and circular-arc graphs [Chen et al. 1998; Sprague and Takaoka 1999] and for permutation graphs [Bazzaro and Gavoille 2005; Sprague 2007].

$(1+\epsilon)$–approximate distance oracles have been found for geometric graphs [Gudmundsson et al. 2008; Sankaranarayanan and Samet 2009; Sankaranarayanan et al. 2009], for graphs with bounded doubling dimension [Har-Peled and Mendel 2006; Abraham et al. 2008; Bartal et al. 2011; Kawarabayashi et al. 2011], and for the intersection graphs of disks and balls [Gao and Zhang 2005; Fürer and Kasiviswanathan 2007] (see also [Funke et al. 2008] for a weighted version and application to wireless networks).

3.2 Route Planning for Road Networks

Route planning for transportation networks (road networks in particular) has been studied intensively [Hoffman and Pavley 1958; Butas 1968; Elliott and Lesk 1982; Ishikawa et al. 1991; Shapiro et al. 1992; Shekhar et al. 1993; Car and Frank 1994; Liu et al. 1994; Zhao and Zaki 1994; Liu 1995; Huang et al. 1996a; 1996b; Jing et al. 1996; Huang et al. 1997a; Zhan 1997; Aydin and Ierardi 1997; Fetterer and Shekhar 1997; Huang et al. 1997a; 1997b; Shekhar and Liu 1997; Shekhar et al.

1997; Pallottino and Scutellà 1998; Zhan and Noon 1998; Jing et al. 1998; Chou et al. 1998; Anwar and Yoshida 2000; 2001; Car et al. 2001; Zhao and Cheng 2001; Jung and Pramanik 2002; Ahuja et al. 2002; Gutman 2004; Brandes et al. 2004; Köhler et al. 2005; Holzer et al. 2005; Goldberg and Werneck 2005; Wagner et al. 2005; Sanders and Schultes 2006; Hu et al. 2006; Möhring et al. 2006; Bast et al. 2007; Baker and Gokhale 2007; Kriegel et al. 2007; Chan and Lim 2007; Kriegel et al. 2008; Kriegel et al. 2008; Kriegel et al. 2008; Holzer et al. 2008; Geisberger et al. 2008; Demir et al. 2008; Kanoh and Hara 2008; Sankaranarayanan and Samet 2009; Bauer et al. 2010; Song and Wang 2011; Hlinený and Moris 2011; Delling et al. 2011] (this list, despite being long, is almost certainly not exhaustive). The 9th DIMACS Implementation Challenge [Demetrescu et al. 2008], which took place in 2006, stimulated a lot of research with impressive results. In the following, we give a brief overview. The tradeoffs between space and query time are summarized in Fig. 4, for absolute performance numbers we refer to [Delling et al. 2009a, Table 1], [Abraham et al. 2011, Table 1], and [Geisberger et al. 2012, Table 2]. For more details on recent results we refer to two surveys on route planning [Delling et al. 2009a; Delling et al. 2011], 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.

We distinguish two types of route planning methods: *(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 very hierarchical nature in available models of 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 updates or incorporating realistic turn costs may have unexpected consequences to the methods' performances). Many the recent methods for road networks effectively exploit properties of the shortest-path metric and the hierarchical structure of road networks.

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, Fig. 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

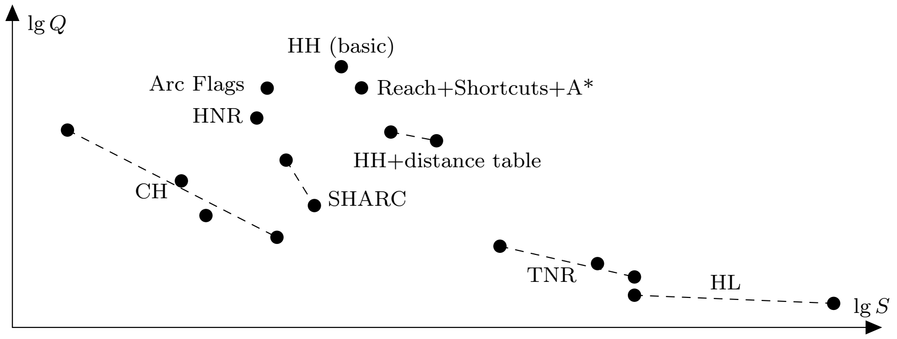

Fig. 4. Route planning for road networks: the tradeoff between space [S] and query time [Q] for recent shortest-path query data structures, depicted using doubly-logarithmic scales (using arbitrary units). The actual performance numbers represented by this figure were extracted from [Delling et al. 2009a, Table 1] and [Abraham et al. 2011, Table 1]. Performance numbers were obtained on different machines and scaled with best effort to make methods comparable. Dashed lines do not carry any meaning; they only 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], *Highway Hierarchies (HH)* [Sanders and Schultes 2005; 2006; Delling et al. 2009b], and *Highway-Node Routing (HNR)* [Schultes and Sanders 2007] can be seen as the "first generation" of CH; *Transit-Node Routing (TNR)* [Bast et al. 2007; Bauer et al. 2010] and *Hub Labels (HL)* [Abraham et al. 2011] dominate the fast-query-time regime.

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 but their performance is significantly 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 Fig. 4). However, estimating travel times for road segments is a highly non-trivial 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 few studies on the *robustness* of these methods, investigating whether the performances would drop significantly upon changes to the length function. Recent theoretical research [Bauer et al. 2009; Bauer et al. 2010; Abraham et al. 2010; Abraham et al. 2011] 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 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 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 towards their use in road networks. Hierarchical preprocessing algorithms usually compute an additional graph structure to speed up shortest-path queries.

3.2.1 *Graph Annotation Approaches.* Shortest-path algorithms based on the annotation approach are sometimes also termed *goal-directed search algorithms.* Additional information is attached to nodes or edges of the graph. Based on this information, the search algorithm decides which part of the graph not to search, or where to *prune* the search space.

*A** [Gelernter 1963; Samuel 1963; Kung et al. 1986; Hart et al. 1968; Doran 1967; Gelperin 1977; Korf 1985] is a popular search technique in Artificial Intelligence. The idea is to direct the search towards the goal. In the priority queue implementation of Dijkstra's algorithm [Dijkstra 1959; Fredman and Tarjan 1987], 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. In such situations, a metric embedding or a drawing [Wagner and Willhalm 2005] can provide coordinates for a potential function.

Goldberg and Harrelson [2005] (see also [Goldberg and Werneck 2005; 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 $l \in L$. 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 highly depend on the number of landmarks. An important 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 [Goldberg and Harrelson 2005] 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. Better speedups can be obtained when combining A* with the bidirectional version of Dijkstra's algorithm. This is, however, not an entirely straightforward combination. The potential function for the forward and the backward search need to be consistent such that the shortest path is found when both searches meet. A good approach for consistent potentials is to take the average of the forward and backward potential functions [Ikeda et al. 1994].

Querying using precomputed cluster distances [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 is directed towards the goal.

Reach-Based Routing [Gutman 2004] is an annotation approach specifically targeted to exploit hierarchical properties of road networks. While this approach is technically an annotation approach, it can, at least in spirit, also be considered a hierarchical approach. It is one of the first methods capturing the hierarchical nature of road networks and exploiting it with provable correctness guarantees.[12] 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 (similar to *Highway Hierarchies*, see Section 3.2.2). 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.

Arc Flags [Lauther 2004; Köhler et al. 2005; Möhring et al. 2006]. The preprocessing algorithm partitions the graph into clusters and then, for each cluster, marks all edges where shortest paths towards nodes in the cluster start. The query algorithm ignores 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, the preprocessing step of the Arc Flag approach is rather expensive (there is a fast parallel preprocessing algorithm [Delling et al. 2012]). 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. 2010].

Shortcuts. One core part of many speedup techniques 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 by 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. It cannot be approximated neither by a constant factor [Bauer et al. 2009, Theorem 1] nor by an additive constant [Bauer et al. 2009, Theorem 2] unless $\mathbf{P} = \mathbf{NP}$. If the shortest paths are unique, a greedy algorithm can find a

[12]Prior industrial methods classified roads according to user-defined heuristic hierarchies (often using road categories), thereby sacrificing correctness.

solution that is optimal up to a constant factor.

3.2.2 *Hierarchical Approaches.* Hierarchical methods to compute shortest paths in graphs have been proposed by many researchers [Kleinrock and Kamoun 1977; Shapiro et al. 1992; Shekhar et al. 1997; Ishikawa et al. 1991; Car and Frank 1993; 1994; Agrawal and Jagadish 1994; Jing et al. 1996; Buchholz and Riedhofer 1997; Riedhofer 1997; Timpf and Frank 1997; Fetterer and Shekhar 1997; Chou et al. 1998; Car et al. 2001; Huang et al. 1997a; Buchholz 2000; Anwar and Yoshida 2000; 2001; Jung and Pramanik 2002; Schulz et al. 2002; Holzer et al. 2008; Sanders and Schultes 2006; Bast et al. 2007; Chan and Lim 2007; Chan and Zhang 2007; Delling et al. 2009; Kriegel et al. 2008; Geisberger et al. 2008; Holzer 2008; Bauer et al. 2010; Miyamoto et al. 2010; Honiden et al. 2010; Delling et al. 2011] (and probably many more). An auxiliary graph is constructed hierarchically. A shortest-path query is then answered by searching only a small part of the auxiliary graph, often using Dijkstra's algorithm. This approach works very well for intrinsically hierarchical graphs. In the following, we give a brief overview of selected recent hierarchical approaches.

Multi-Level Overlay Graphs [Schulz et al. 2002; Holzer et al. 2008; Delling et al. 2009; Delling et al. 2011] 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 are 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 right set of nodes as landmarks 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 paragraph below).
Customizable Route Planning (CRP) [Delling et al. 2011] uses small recursive separators [Delling et al. 2011]. CRP is currently used in Microsoft Bing Maps.[13] See also Section 3.1.1 for very similar 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 target. 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. Similar methods have been and still are used in commercial systems, however, highway edges in those systems are usually selected by heuristics that cannot guarantee correctness. Highway hierarchies — a method with correctness guarantees — is due to Sanders and Schultes, first proposed for undirected graphs [Sanders and Schultes 2005], and later extended to directed graphs [Sanders and Schultes 2006]. Nannicini et al. [2008] give a different approach for directed highway hierarchies. Their main focus is on time-dependent lengths though.

[13]Fore more information, see also `http://www.bing.com/community/site_blogs/b/maps/archive/2012/01/05/bing-maps-new-routing-engine.aspx`.

Contraction Hierarchies (CH) [Geisberger et al. 2008; Geisberger et al. 2012] is the exceedingly popular successor of HH. An integral ingredient of the HH speedup technique 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 shortcuts one would expect based on the node degree, the node is contracted. In the setting of efficiently answering shortest-path queries, a node contraction can also be interpreted as a particularly structured way of adding shortcuts. The method called *Contraction Hierarchies* [Geisberger et al. 2008; Geisberger et al. 2012] uses intelligent heuristics to contract nodes in the "right" order.[14] This order yields a hierarchy, using which the query algorithm can efficiently find shortest paths. Contraction-based techniques perform very well in practice, and the preprocessing step is particularly efficient. For some road networks, the space overhead of one variant of CH is negative, meaning that the CH data structure is actually more space-efficient than just storing the network itself.

Transit-Node Routing (TNR) [Bast et al. 2007; Bast et al. 2007] 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 expensive but the query time is extremely fast, since, for any two given locations, it essentially requires only a few dozen table lookups for all pairs of corresponding landmarks.

Hub Labels (HL) [Abraham et al. 2011; 2012] 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 (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 $L(s), L(t)$, the algorithm simply computes and outputs $\min_{l \in L(s) \cap L(t)} d(s,l) + d(l,t)$. As long as $L(s), L(t)$ are small, the query algorithm is very fast,[15] and, if labels are stored consecutively for each node, the query algorithm also has very good locality, making it extremely efficient. 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. The set of landmarks *covers* all shortest paths; Cohen, Halperin, Kaplan, and Zwick [2003]

[14]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.
The Contraction Hierarchies algorithm is also somewhat related to the nested-dissection algorithm of Lipton, Rose, and Tarjan [1979].

[15]The algorithm runs in $O(|L(s)| + |L(t)|)$ — it is actually enough if one of the two sets is small, say $L(s)$, then we can binary search for each $l \in L(s)$ in $L(t)$ in time $O(|L(s)| \cdot \log |L(t)|)$.

call these labels 2–*hop covers*, for which they provide an $O(n^4)$–time algorithm that computes an $O(\lg n)$–approximation for the optimal cover (see also Section 2.2). For road networks, using heuristics such as Contraction Hierarchies, small labels can be computed efficiently. 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. Even labels based on optimally chosen landmarks would likely occupy more space than the data structure computed by the TNR preprocessing algorithm, and therefore even optimal HL is unlikely to strictly dominate TNR on the space vs. query time tradeoff curve (Fig. 4).

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. 2010]. Two particularly strong combinations are CHASE [Bauer et al. 2010], which combines Contraction Hierarchies with Arc Flags, and SHARC [Bauer and Delling 2009; Brunel et al. 2010], which combines Shortcuts with Arc Flags.

The observed performance of the aforementioned hierarchy-based methods is outstanding, however, complexity results are mostly experimental (exactness and correctness are proven). Recently, Abraham et al. [2010; 2011] found that, if a graph has low *highway dimension*, algorithms based on Reach [Gutman 2004], Contraction Hierarchies [Geisberger et al. 2008], Transit Nodes [Bast et al. 2007], and SHARC [Bauer and Delling 2009] have provable efficiency guarantees. Intuitively, a graph has 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.

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 towards their use in road networks, can be made work on other networks as well, such as those stemming from public transportation [Schulz et al. 2000; Schulz 2005; Müller-Hannemann et al. 2004; Pyrga et al. 2007; Jacob 2008; Bast et al. 2010; Bauer et al. 2011; Delling et al. 2012].

4. COMPLEX NETWORKS

More recently, shortest-path query algorithms and data structures have been studied for more general graphs. In particular, there are 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 benchmarks poses difficulties on 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 [Ball et al. 1997; Watts and Strogatz 1998; Barabási and Albert 1999; Bollobás et al. 2001; Aiello et al. 2000; Chung and Lu 2002; Newman et al. 2002; Reittu and

Norros 2004; Kumar and Latapy 2006; Leskovec and Faloutsos 2007; Lattanzi and Sivakumar 2009] (without being exhaustive). 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*[16]), and *(iii)* there seem to be no small separators (linear-sized *core*). Most of these properties currently cannot be exploited by common algorithmic techniques and many combinatorial optimization problems remain **NP**–hard for complex networks [Ferrante et al. 2008] (the shortest-path problem can still be solved in almost linear time using Dijkstra's algorithm). Other properties such as the one that these networks have nodes with very large 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 experimental nature. I am aware of only few results with worst-case bounds, which are the following *compact routing schemes*.[17] Enachescu et al. [2008] analyze the compact routing scheme of Thorup and Zwick [2001] for $G_{n,p}$ random graphs. They prove that multiplicative stretch $\alpha = 2$ can be achieved with space $\tilde{O}(n^{7/4})$ by selecting $\tilde{O}(n^{3/4})$ landmarks. 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 [2005] using high-degree nodes as landmarks. Given that shortest paths in complex networks are usually 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 very useful.

Implementations and Adaptations. Krioukov et al. [2004] evaluate the compact routing scheme of Thorup and Zwick [2001] for Internet-like inter-domain topologies and random power-law graphs. The Thorup-Zwick distance oracle [2005] uses information precomputed in two steps (see also Section 2.2.1). The query result is either *(i)* a local exact distance or *(ii)* a triangulation via landmarks. Many implementations focus on the triangulation part, providing good estimates for long-range distances. Potamias et al. [2009] experiment on how to select landmarks.See also [Tretyakov et al. 2011; Qiao et al. 2011; Qiao et al. 2012; Cheng et al. 2012]. Das Sarma et al. [2010] provide an implementation of the Thorup-Zwick distance oracle [2005]. Gubichev et al. [2010] extend their work such that the method can also output actual paths. See also [Cao et al. 2011]. Agarwal et al. [2012] focus on the local part, computing distances using ball intersections (which they call *vicini-*

[16]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].
[17]Any compact routing scheme may serve as an approximate shortest-path oracle, retrieving each edge of the path by simulating the decision of a router in a centralized way. We do not attempt to cover results on compact routing in this review. We instead refer to [Gavoille and Peleg 2003; Thorup and Zwick 2001; Abraham et al. 2008] and the references therein.

ties) with landmarks sampled with probability proportional to their degrees. For corresponding worst-case bounds, see also [Patrascu and Roditty 2010; Agarwal et al. 2011]. For 2–hop covers [Cohen et al. 2003], there are several implementations [Schenkel et al. 2004; 2005; Cheng et al. 2006; Cheng et al. 2006; Cheng et al. 2008; Cheng and Yu 2009; Abraham et al. 2012]. Extensions to 3 and more hops have been proposed [Jin et al. 2008; Jin et al. 2009; Chang et al. 2012]. A method of Goldman et al. [1998] can be seen as some kind of 2–hop cover, albeit their method came earlier. Rattigan et al. [2006; 2007] combine a clustering-type approach with landmarks.

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] appear to be rather promising.

Core–Fringe Methods. In the 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 (up to some not-too-large set of edges E'). The scheme uses one routing tree for the core and $|E'|$ trees for the fringe – experiments using random power-law graphs [Aiello et al. 2000] indicate that both D and E' can be chosen to be small simultaneously. 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).

Others. Trißl and Leser [2007] create indices for reachability queries in (very) sparse graphs. Xiao et al. [2009] compress graphs by exploiting symmetries. Instead of treating nodes as a single unit, they work on *orbits* of *automorphism groups.* Honiden et al. [2010] provide an approximation method with fast preprocessing times based on *Voronoi* duals.

5. SUMMARY

Shortest-path query processing in graphs has been studied extensively both by theoreticians and in practitioners. Practical investigations focus mainly on the important 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 towards dynamic and personalized scenarios, developing a navigation assistant that can predict, given sufficient information on current and past traffic conditions, how long it would require a specific driver to use a certain route, and then suggest the best one. 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 important 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; Chen et al. 2012; Mozes and Sommer 2012]. This survey has improved substantially thanks to many valuable comments and suggestions from Daniel Delling, Ron Gutman, Shay Mozes, Dirck van Vliet, and Christian Wulff-Nilsen.

REFERENCES

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

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

Abraham, I., Delling, D., Goldberg, A. V., and Werneck, R. F. 2012. Hierarchical hub labelings for shortest paths. Tech. Rep. MSR-TR-2012-46, Microsoft Research.

Abraham, I., Delling, D., Goldberg, A. V., and Werneck, R. F. F. 2011. A hub-based labeling algorithm for shortest paths in road networks. In *10th International Symposium on Experimental Algorithms (SEA)*. 230–241. Announced as MSR-TR-2010-165 in 2010.

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

Abraham, I. and Gavoille, C. 2006. Object location using path separators. In *25th ACM Symposium on Principles of Distributed Computing (PODC)*. 188–197. Details in LaBRI Research Report RR-1394-06.

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

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

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

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

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

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

Agrawal, R. and Jagadish, H. V. 1994. Algorithms for searching massive graphs. *IEEE Transactions on Knowledge and Data Engineering 6,* 2, 225–238.

Ahuja, R. K., Orlin, J. B., Pallottino, S., and Scutellà, M. G. 2002. Minimum time and minimum cost-path problems in street networks with periodic traffic lights. *Transportation Science 36,* 3, 326–336.

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

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

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

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

Alon, N., Galil, Z., and Margalit, O. 1997. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences 54,* 2, 255–262. Announced at FOCS 1991.

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

Althöfer, I., Das, G., Dobkin, D. P., Joseph, D., and Soares, J. 1993. On sparse spanners of weighted graphs. *Discrete & Computational Geometry 9*, 81–100.

Anwar, M. A. and Yoshida, T. 2000. OORF: An object-oriented route finder. In *15th ACM Symposium on Applied Computing (SAC)*. 301–306.

Anwar, M. A. and Yoshida, T. 2001. Integrating OO road network database, cases and knowledge for route finding. In *16th ACM Symposium on Applied Computing (SAC)*. 215–219.

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

Aydin, C. and Ierardi, D. 1997. Partitioning algorithms for transportation graphs and their applications to routing. In *9th Canadian Conference on Computational Geometry (CCCG)*.

Baker, Z. K. and Gokhale, M. 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.

Ball, F., Mollison, D., and Scalia-Tomba, G. 1997. Epidemics with two levels of mixing. *Annals of Applied Probability 7,* 1, 46–89.

Barabási, A.-L. and Albert, R. 1999. Emergence of scaling in random networks. *Science 286,* 5439, 509–512.

Barrett, C. L., Bisset, K. R., Jacob, R., Konjevod, G., and Marathe, M. V. 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.

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

Bartal, Y., Linial, N., Mendel, M., and Naor, A. 2005. On metric Ramsey type phenomena. *Annals of Mathematics (2) 162,* 2, 643–709. Announced at STOC 2003.

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

Bast, H., Carlsson, E., Eigenwillig, A., Geisberger, R., Harrelson, C., Raychev, V., and Viger, F. 2010. Fast routing in very large public transportation networks using transfer patterns. In *18th European Symposium on Algorithms (ESA)*. 290–301.

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

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

Baswana, S., Gaur, A., Sen, S., and Upadhyay, J. 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.

Baswana, S. and Kavitha, T. 2010. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM Journal on Computing 39,* 7, 2865–2896. Announced at FOCS 2006.

Baswana, S. and Sen, S. 2006. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. *ACM Transactions on Algorithms 2,* 4, 557–577. Announced at SODA 2004.

Baswana, S. and Sen, S. 2007. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms 30,* 4, 532–563. Announced at ICALP 2003.

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

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

Bauer, R. and Delling, D. 2009. SHARC: Fast and robust unidirectional routing. *ACM Journal of Experimental Algorithmics 14*. Announced at ALENEX 2008.

Bauer, R., Delling, D., Sanders, P., Schieferdecker, D., Schultes, D., and Wagner, D. 2010. Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm. *ACM Journal of Experimental Algorithmics 15,* 2.3, 1–31. Special Issue for WEA 2008.

Bauer, R., Delling, D., and Wagner, D. 2011. Experimental study on speed-up techniques for timetable information systems. *Networks 57,* 1, 38–52.

Bazaraa, M. S. and Langley, R. W. 1974. A dual shortest path algorithm. *SIAM Journal on Applied Mathematics 26,* 3, 496–501.

Bazzaro, F. and Gavoille, C. 2005. Distance labeling for permutation graphs. *Electronic Notes in Discrete Mathematics 22*, 461–467.

Bellman, R. E. 1967. *Dynamic Programming.* Princeton University Press.

Bollobás, B., Coppersmith, D., and Elkin, M. L. 2003. Sparse distance preservers and additive spanners. In *14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Bollobás, B., Riordan, O., Spencer, J., and Tusnády, G. E. 2001. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms 18,* 3, 279–290.

Boothroyd, J. 1967. Algorithms: Author's note on algorithms 22, 23, 24; algorithm 22: Shortest path between start node and end node of a network; algorithm 23: Shortest path between start node and all other nodes of a network; algorithm 24: The list of nodes on the shortest path from start node to end node of a network. *The Computer Journal 10,* 3, 306–308. In the Algorithms Supplement.

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

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

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

Brandes, U., Schulz, F., Wagner, D., and Willhalm, T. 2004. Generating node coordinates for shortest-path computations in transportation networks. *ACM Journal of Experimental Algorithmics 9*.

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

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

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

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

Butas, L. F. 1968. A directionally oriented shortest path algorithm. *Transportation Research 2,* 3, 253–268.

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

Cabello, S. 2011. Many distances in planar graphs. *Algorithmica*. Announced at SODA 2006.

CABELLO, S., CHAMBERS, E. W., AND ERICKSON, J. 2012. Multiple-source shortest paths in embedded graphs. *arXiv abs/1202.0314*. Announced at SODA 2007.

CALINESCU, G., KARLOFF, H. J., AND RABANI, Y. 2004. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing 34,* 2, 358–372. Announced at SODA 2001.

CAO, L., ZHAO, X., ZHENG, H., AND ZHAO, B. Y. 2011. Atlas: Approximating shortest paths in social graphs. Tech. Rep. 2011-09, Department of Computer Science, University of California, Santa Barbara.

CAR, A. AND FRANK, A. U. 1993. Hierarchical street networks as a conceptual model for efficient way finding. In *4th European Conference and Exhibition on Geographical Information Systems (EGIS)*.

CAR, A. AND FRANK, A. U. 1994. Modelling a hierarchy of space applied to large road networks. In *International Workshop on Advanced Research in Geographic Information Systems (IGIS)*. 15–24.

CAR, A., TAYLOR, G., AND BRUNSDON, C. 2001. An analysis of the performance of a hierarchical wayfinding computational model using synthetic graphs. *Computers, Environment and Urban Systems 25,* 1, 69–88.

CHAN, E. P. AND LIM, H. 2007. Optimization and evaluation of shortest path queries. *The VLDB Journal 16,* 3, 343–369.

CHAN, E. P. F. AND ZHANG, J. 2007. A fast unified optimal route query evaluation algorithm. In *16th ACM Conference on Conference on Information and Knowledge Management (CIKM)*. 371–380.

CHAN, T. M. 2010. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM Journal on Computing 39,* 5, 2075–2089. Announced at STOC 2007.

CHANG, L., YU, J. X., QIN, L., CHENG, H., AND QIAO, M. 2012. The exact distance to destination in undirected world. *The VLDB Journal*.

CHARTRES, B. A. 1967. Letter concerning Nicholson's paper. *The Computer Journal 10,* 1, 118–119. In Discussion and Correspondence.

CHAUDHURI, S. AND ZAROLIAGIS, C. D. 2000. Shortest paths in digraphs of small treewidth. part I: Sequential algorithms. *Algorithmica 27,* 3, 212–226. Announced at ICALP 1995.

CHEN, D. Z., LEE, D. T., SRIDHAR, R., AND SEKHARAN, C. N. 1998. Solving the all-pair shortest path query problem on interval and circular-arc graphs. *Networks 31,* 4, 249–258.

CHEN, D. Z. AND XU, J. 2000. Shortest path queries in planar graphs. In *32nd ACM Symposium on Theory of Computing (STOC)*. 469–478.

CHEN, W., SOMMER, C., TENG, S.-H., AND WANG, Y. 2012. A compact routing scheme and approximate distance oracle for power-law graphs. *ACM Transactions on Algorithms*. To appear, announced at DISC 2009.

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

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

CHENG, J., YU, J. X., LIN, X., WANG, H., AND YU, P. S. 2006. Fast computation of reachability labeling for large graphs. In *10th International Conference on Extending Database Technology (EDBT)*. 961–979.

CHENG, J., YU, J. X., LIN, X., WANG, H., AND YU, P. S. 2008. Fast computing reachability labelings for large graphs with high compression rate. In *11th International Conference on Extending Database Technology (EDBT)*. 193–204.

CHENG, J., YU, J. X., AND TANG, N. 2006. Fast reachability query processing. In *11th International Conference on Database Systems for Advanced Applications (DASFAA)*. 674–688.

CHERKASSKY, B. V., GOLDBERG, A. V., AND RADZIK, T. 1996. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming 73*, 129–174. Announced at SODA 1994.

CHINDELEVITCH, L., ZIEMEK, D., ENAYETALLAH, A., RANDHAWA, R., SIDDERS, B., BROCKEL, C., AND HUANG, E. 2011. Causal reasoning on biological networks: Interpreting transcriptional changes. In *15th International Conference on Research in Computational Molecular Biology (RECOMB)*.

Chou, Y.-L., Romeijn, H. E., and Smith, R. L. 1998. Approximating shortest paths in large-scale networks with an application to intelligent transportation systems. *INFORMS Journal on Computing 10,* 2, 163–179.

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

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

Cohen, E., Halperin, E., Kaplan, H., and Zwick, U. 2003. Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing 32,* 5, 1338–1355. Announced at SODA 2002.

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

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

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

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

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

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

de Champeaux, D. 1983. Bidirectional heuristic search again. *Journal of the ACM 30,* 1, 22–32.

Delling, D. 2009. Engineering and augmenting route planning algorithms. Ph.D. thesis, Universität Karlsruhe.

Delling, D., Goldberg, A. V., Nowatzyk, A., and Werneck, R. F. 2012. PHAST: Hardware-accelerated shortest path trees. *Journal of Parallel and Distributed Computing*. Announced at IPDPS 2011.

Delling, D., Goldberg, A. V., Pajor, T., and Werneck, R. F. 2011. Customizable route planning. In *10th International Symposium on Experimental Algorithms (SEA)*. 376–387.

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

Delling, D., Goldberg, A. V., and Werneck, R. F. F. 2011. Shortest paths in road networks: From practice to theory and back. *it - Information Technology 53,* 6, 294–301.

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

Delling, D., Pajor, T., and Werneck, R. F. 2012. Round-based public transit routing. In *14th Workshop on Algorithm Engineering and Experiments (ALENEX)*.

Delling, D., Sanders, P., Schultes, D., and Wagner, D. 2009a. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks - Design, Analysis, and Simulation [DFG priority program 1126]*. 117–139.

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

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

DEMIR, E., AYKANAT, C., AND BARLA CAMBAZOGLU, B. 2008. Clustering spatial networks for aggregate query processing: A hypergraph approach. *Information Systems 33,* 1, 1–17.

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

DIAL, R. B., GLOVER, F., KARNEY, D., AND KLINGMAN, D. 1979. A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. *Networks 9*, 215–248.

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

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

DJIDJEV, H., PANTZIOU, G. E., AND ZAROLIAGIS, C. D. 2000. Improved algorithms for dynamic shortest paths. *Algorithmica 28,* 4, 367–389.

DJIDJEV, H. N. 1985. A linear algorithm for partitioning graphs of fixed genus. *Serdica. Bulgariacae mathematicae publicationes 11,* 4, 369–387. Announced in *Comptes Rendus de l'Académie Bulgare des Sciences*, 34:643–645, 1981.

DJIDJEV, H. N. AND SOMMER, C. 2011. Approximate distance queries for weighted polyhedral surfaces. In *19th European Symposium on Algorithms (ESA)*. 579–590.

DOR, D., HALPERIN, S., AND ZWICK, U. 2000. All-pairs almost shortest paths. *SIAM Journal on Computing 29,* 5, 1740–1759. Announced at FOCS 1996.

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

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

DVORAK, Z., KRÁL, D., AND THOMAS, R. 2010. Deciding first-order properties for sparse graphs. In *51st IEEE Symposium on Foundations of Computer Science (FOCS)*. 133–142.

ELKIN, M. L. 2008. Sparse graph spanners. In *Encyclopedia of Algorithms*.

ELKIN, M. L. AND PELEG, D. 2004. (1+epsilon, beta)-spanner constructions for general graphs. *SIAM Journal on Computing 33,* 3, 608–631.

ELLIOTT, R. J. AND LESK, M. 1982. Route finding in street maps by computers and people. In *National Conference on Artificial Intelligence (AAAI)*. 258–261.

ENACHESCU, M., WANG, M., AND GOEL, A. 2008. Reducing maximum stretch in compact routing. In *27th IEEE International Conference on Computer Communications (INFOCOM)*. 336–340.

EPPSTEIN, D. 1999. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications 3,* 3. Announced at SODA 1995.

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

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

ERIKSSON, B., BARFORD, P., AND NOWAK, R. D. 2009. Estimating hop distance between arbitrary host pairs. In *28th IEEE International Conference on Computer Communications (INFOCOM)*. 801–809.

FAKCHAROENPHOL, J. AND RAO, S. 2006. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences 72,* 5, 868–889. Announced at FOCS 2001.

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

FARBEY, B. A., LAND, A. H., AND MURCHLAND, J. D. 1967. The cascade algorithm for finding all shortest distances in a directed graph. *Management Science 14,* 1, 19–28.

FARZAN, A. AND KAMALI, S. 2011. Compact navigation and distance oracles for graphs with small treewidth. In *38th International Colloquium on Automata, Languages and Programming (ICALP)*.

Ferrante, A., Pandurangan, G., and Park, K. 2008. On the hardness of optimization in power-law graphs. *Theoretical Computer Science 393,* 1-3, 220–230.

Fetterer, A. and Shekhar, S. 1997. A performance analysis of hierarchical shortest path algorithms. In *9th International Conference on Tools with Artificial Intelligence (ICTAI)*. 84–93.

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

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

Ford, L. R. 1956. Network flow theory. Report P-923, The Rand Corporation.

Fredman, M. L. and Tarjan, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM 34,* 3, 596–615. Announced at FOCS 1984.

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

Funke, S., Matijevic, D., and Sanders, P. 2008. Constant time queries for energy efficient paths in multi-hop wireless networks. *Journal of Computing and Information Technology 16,* 2, 119–130. Announced at ESA 2003.

Fürer, M. and Kasiviswanathan, S. P. 2007. Spanners for geometric intersection graphs. In *10th International Workshop on Algorithms and Data Structures (WADS)*. 312–324.

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

Gao, J. and Zhang, L. 2005. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing 35,* 1, 151–169.

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

Gavoille, C., Peleg, D., Pérennes, S., and Raz, R. 2004. Distance labeling in graphs. *Journal of Algorithms 53,* 1, 85–112. Announced at SODA 2001, see also LaBRI Research Report RR-1239-00.

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

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

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

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

Gelperin, D. 1977. On the optimality of A*. *Artificial Intelligence 8,* 1, 69–76.

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

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

Glover, F., Klingman, D., and Napier, A. 1974. A note on finding all shortest paths. *Transportation Science 8*, 3–12.

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

Goldberg, A. V. 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.

Goldberg, A. V. 2008. A practical shortest path algorithm with linear expected time. *SIAM Journal on Computing 37,* 5, 1637–1655.

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

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

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

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

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

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

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

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

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

Gutman, R. 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.

Hagerup, T. 2000. Improved shortest paths on the word RAM. In *27th International Colloquium on Automata, Languages and Programming (ICALP)*. 61–72.

Har-Peled, S. and Mendel, M. 2006. Fast construction of nets in low-dimensional metrics and their applications. *SIAM Journal on Computing 35,* 5, 1148–1184. Announced at SoCG 2005.

Hart, P. E., Nilsson, N. J., and Raphael, B. R. 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.

Henzinger, M. R., Klein, P. N., Rao, S., and Subramanian, S. 1997. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences 55,* 1, 3–23. Announced at STOC 1994.

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

Hliněný, P. and Moris, O. 2011. Scope-based route planning. In *19th European Symposium on Algorithms (ESA)*. 445–456.

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

Hoffman, W. and Pavley, R. 1958. Applications of digital computers to problems in the study of vehicular traffic. In *IRE-ACM-AIEE Western Joint Computer Conference: Contrasts in Computers*. 159–161.

Holzer, M. 2008. Engineering planar-separator and shortest-path algorithms. Ph.D. thesis, Universität Karlsruhe.

Holzer, M., Schulz, F., and Wagner, D. 2008. Engineering multilevel overlay graphs for shortest-path queries. *ACM Journal of Experimental Algorithmics 13*. Announced at ALENEX 2006.

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

Honiden, S., Houle, M. E., Sommer, C., and Wolff, M. 2010. Approximate shortest path queries in graphs using Voronoi duals. *Transactions on Computational Science 9*, 28–53. Special Issue on Voronoi Diagrams in Science and Engineering (ISVD 2009).

Hu, H., Lee, D. L., and Lee, V. C. S. 2006. Distance indexing on road networks. In *32nd International Conference on Very Large Data Bases (VLDB)*. 894–905.

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

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

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

Huang, Y.-W., Jing, N., and Rundensteiner, E. A. 1996a. Effective graph clustering for path queries in digital map databases. In *5th International Conference on Information and Knowledge Management (CIKM)*. 215–222.

Huang, Y.-W., Jing, N., and Rundensteiner, E. A. 1996b. Path queries for transportation networks: Dynamic reordering and sliding window paging techniques. In *4th ACM Workshop on Advances in Geographic Information Systems (GIS)*. 9–16.

Huang, Y.-W., Jing, N., and Rundensteiner, E. A. 1997a. A hierarchical path view model for path finding in intelligent transportation systems. *GeoInformatica 1,* 2, 125–159. Announced at GIS 1995.

Huang, Y.-W., Jing, N., and Rundensteiner, E. A. 1997b. Integrated query processing strategies for spatial path queries. In *13th International Conference on Data Engineering (ICDE)*. 477–486.

Hutchinson, D. A., Maheshwari, A., and Zeh, N. 2003. An external memory data structure for shortest path queries. *Discrete Applied Mathematics 126,* 1, 55–82.

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

Ishikawa, K., Ogawa, M., Azuma, S., and Ito, T. 1991. Map navigation software of the electro-multivision of the '91 Toyoto Soarer. In *Vehicle Navigation and Information Systems Conference*. Vol. 2. 463–473.

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

Jacob, R. 2008. Shortest paths approaches for timetable information. In *Encyclopedia of Algorithms.*

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

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

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

Jing, N., Huang, Y.-W., and Rundensteiner, E. A. 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.

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

Jung, S. and Pramanik, S. 2002. An efficient path computation model for hierarchically structured topographical road maps. *IEEE Transactions on Knowledge and Data Engineering 14,* 5, 1029–1046. Announced at ICDE 1996.

Kannan, S., Naor, M., and Rudich, S. 1992. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics 5,* 4, 596–603. Announced at STOC 1988.

KANOH, H. AND HARA, K. 2008. Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network. In *Genetic and Evolutionary Computation Conference (GECCO)*. 657–664.

KARINTHY, F. 1929. *Lancszemek*.

KAWARABAYASHI, K., KLEIN, P. N., AND SOMMER, C. 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.

KITAMURA, M. AND YAMAZAKI, M. 1965. On the connection of the two shortest route systems. In *8th Japanese Road Conference*. 66–68. In Japanese.

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

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

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

KLEIN, P. N., MOZES, S., AND WEIMANN, O. 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. Announced at SODA 2009.

KLEIN, P. N. AND SUBRAMANIAN, S. 1998. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica 22,* 3, 235–249. Announced at WADS 1993.

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

KLEINBERG, J. M., SLIVKINS, A., AND WEXLER, T. 2009. Triangulation and embedding using small sets of beacons. *Journal of the ACM 56,* 6. Announced at FOCS 2004.

KLEINROCK, L. AND KAMOUN, F. 1977. Hierarchical routing for large networks; performance evaluation and optimization. *Computer Networks 1*, 155–174.

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

KORF, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence 27,* 1, 97–109.

KOWALIK, L. AND KUROWSKI, M. 2006. Oracles for bounded-length shortest paths in planar graphs. *ACM Transactions on Algorithms 2,* 3, 335–363. Announced at STOC 2003.

KRIEGEL, H.-P., KRÖGER, P., KUNATH, P., RENZ, M., AND SCHMIDT, T. 2007. Proximity queries in large traffic networks. In *15th ACM International Symposium on Geographic Information Systems (GIS)*.

KRIEGEL, H.-P., KRÖGER, P., KUNATH, P., RENZ, M., AND SCHMIDT, T. 2008. Efficient query processing in large traffic networks. In *24th International Conference on Data Engineering (ICDE)*. 1451–1453.

KRIEGEL, H.-P., KRÖGER, P., AND RENZ, M. 2008. Continuous proximity monitoring in road networks. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (GIS)*.

KRIEGEL, H.-P., KRÖGER, P., RENZ, M., AND SCHMIDT, T. 2008. Hierarchical graph embedding for efficient query processing in very large traffic networks. In *20th International Conference on Scientific and Statistical Database Management (SSDBM)*. 150–167.

KRIOUKOV, D. V., FALL, K. R., AND YANG, X. 2004. Compact routing on Internet-like graphs. In *23rd IEEE International Conference on Computer Communications (INFOCOM)*.

KUMAR, R. AND LATAPY, M. 2006. Preface. *Theoretical Computer Science: special issue on Complex Networks 355,* 1, 1–5.

KUNG, R.-M., HANSON, E. N., IOANNIDIS, Y. E., SELLIS, T. K., SHAPIRO, L. D., AND STONEBRAKER, M. 1986. Heuristic search in database systems. In *1st International Workshop on Expert Database Systems*. 537–548.

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

Lansdowne, Z. F. and Robinson, D. W. 1982. Geographic decomposition of the shortest path problem, with an application to the traffic assignment problem. *Management Science 28,* 12, 1380–1390.

Lattanzi, S. and Sivakumar, D. 2009. Affiliation networks. In *41st ACM Symposium on Theory of Computing (STOC)*. 427–434.

Lauther, U. 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.

Leskovec, J. and Faloutsos, C. 2007. Scalable modeling of real graphs using kronecker multiplication. In *24th International Conference on Machine Learning (ICML)*. 497–504.

Linial, N. and Saks, M. E. 1993. Low diameter graph decompositions. *Combinatorica 13,* 4, 441–454. Announced at SODA 1991.

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

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

Liu, B. 1995. Using knowledge to isolate search in route finding. In *14th International Joint conference on Artificial Intelligence (IJCAI)*. 119–124.

Liu, B., Choo, S.-H., Lok, S.-L., Leong, S.-M., Lee, S.-C., Poon, F.-P., and Tan, H.-H. 1994. Integrating case-based reasoning, knowledge-based approach and Dijkstra algorithm for route finding. In *10th Conference on Artificial Intelligence for Applications*. 149–155.

Luby, M. and Ragde, P. 1989. A bidirectional shortest-path algorithm with good average-case behavior. *Algorithmica 4,* 4, 551–567. Announced at ICALP 1985.

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

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

Mendel, M. and Naor, A. 2007. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society 9,* 2, 253–275. Announced at FOCS 2006.

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

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

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

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

Miyamoto, Y., Uno, T., and Kubo, M. 2010. Levelwise mesh sparsification for shortest path queries. In *21st International Symposium on Algorithms and Computation (ISAAC)*. 121–132.

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

Monge, G. 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.

Moore, E. F. 1959. The shortest path through a maze. In *Annals of the Computation Laboratory of Harvard University*. Harvard University Press, 285–292. Announced at the International Symposium on the Theory of Switching 1957.

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

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

Müller-Hannemann, M., Schulz, F., Wagner, D., and Zaroliagis, C. D. 2004. Timetable information: Models and algorithms. In *4th International Workshop on Algorithmic Methods for Railway Optimization (ATMOS)*. 67–90.

Murchland, J. D. 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.

Murchland, J. D. 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.

Nannicini, G., Baptiste, P., Barbier, G., Krob, D., and Liberti, L. 2008. Fast paths in large-scale dynamic road networks. *Computational Optimization and Applications*.

Naor, A. 2012. An introduction to the Ribe program. *arXiv abs/1205.5993*.

Naor, A. and Tao, T. 2010. Scale-oblivious metric fragmentation and the nonlinear Dvoretzky theorem. *arXiv abs/1003.4013*.

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

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

Newman, M. E. J., Watts, D. J., and Strogatz, S. H. 2002. Random graph models of social networks. *Proceedings of the National Academy of Sciences 99*, 2566–2572.

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

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

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

Pallottino, S. and Scutellà, M. G. 1998. *Equilibrium and Advanced Transportation Modelling.* Chapter Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects, 245–281.

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

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

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

Patrascu, M. 2011. Unifying the landscape of cell-probe lower bounds. *SIAM Journal on Computing 40,* 3, 827–847. Announced at FOCS 2008.

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

Patrascu, M., Roditty, L., and Thorup, M. 2011. A new infinity of distance oracles for sparse graphs. Unpublished manuscript.

Peleg, D. 2000. Proximity-preserving labeling schemes. *Journal of Graph Theory 33*, 167–176. Announced at WG 1999.

Peleg, D. and Schäffer, A. A. 1989. Graph spanners. *Journal of Graph Theory 13,* 1, 99–116.

Pettie, S. 2004. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science 312,* 1, 47–74. Announced at ICALP 2002.

Pettie, S. and Ramachandran, V. 2002. Computing shortest paths with comparisons and additions. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 267–276.

Pettie, S. and Ramachandran, V. 2005. A shortest path algorithm for real-weighted undirected graphs. *SIAM Journal on Computing 34,* 6, 1398–1431. Announced at SODA 2002.

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

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

Porat, E. and Roditty, L. 2011. Preprocess, set, query! In *19th European Symposium on Algorithms (ESA)*. 603–614.

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

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

Pyrga, E., Schulz, F., Wagner, D., and Zaroliagis, C. D. 2007. Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithmics 12.*

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

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

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

Raman, R. 1997. Recent results on the single-source shortest paths problem. *SIGACT News 28,* 2, 81–87.

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

Rattigan, M. J., Maier, M., and Jensen, D. 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.

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

Reittu, H. and Norros, I. 2004. On the power-law random graph model of massive data networks. *Performance Evaluation 55,* 1-2, 3–23. Announced at Internet performance symposium (IPS 2002).

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

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

Roughan, M., Willinger, W., Maennel, O., Perouli, D., and Bush, R. 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.

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

Samuel, A. L. 1963. Some studies in machine learning using the game of checkers. *Computers and Thought*. Originally in IBM Journal 3, 211-229 (1959).

Sanders, P. 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.

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

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

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

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

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

SANTOS, J. L. 2009. Real-world applications of shortest path algorithms. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 74*, 1–19. Papers of the 9th DIMACS Implementation Challenge: Shortest Paths, 2006.

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

SCHENKEL, R., THEOBALD, A., AND WEIKUM, G. 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.

SCHMIDT, J. P. 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. Announced at ISTCS 1995.

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

SCHULTES, D. AND SANDERS, P. 2007. Dynamic highway-node routing. In *6th International Workshop on Experimental Algorithms (WEA)*. 66–79.

SCHULZ, F. 2005. Timetable information and shortest paths. Ph.D. thesis, Universität Karlsruhe.

SCHULZ, F., WAGNER, D., AND WEIHE, K. 2000. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. *ACM Journal of Experimental Algorithmics 5*, 12. Announced at WAE 1999.

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

SCHWARTZ, M. AND STERN, T. E. 1980. Routing techniques used in computer communication networks. *IEEE Transactions on Communications 28,* 4, 539–552.

SEDGEWICK, R. AND VITTER, J. S. 1986. Shortest paths in Euclidean graphs. *Algorithmica 1,* 1, 31–48. Announced at FOCS 1984.

SEIDEL, R. 1995. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of Computer and System Sciences 51,* 3, 400–403. Announced at STOC 1992.

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

SHAPIRO, J., WAXMAN, J., AND NIR, D. 1992. Level graphs and approximate shortest path algorithms. *Networks 22*, 691–717.

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

SHEKHAR, S., FETTERER, A., AND GOYAL, B. 1997. Materialization trade-offs in hierarchical shortest path algorithms. In *5th International Symposium on Advances in Spatial Databases (SSD)*. 94–111.

SHEKHAR, S., KOHLI, A., AND COYLE, M. 1993. Path computation algorithms for advanced traveller information system (ATIS). In *9th International Conference on Data Engineering (ICDE)*. 31–39.

SHEKHAR, S. AND LIU, D.-R. 1997. CCAM: A connectivity-clustered access method for networks and network computations. *IEEE Transactions on Knowledge and Data Engineering 9,* 1, 102–119.

SHOSHAN, A. AND ZWICK, U. 1999. All pairs shortest paths in undirected graphs with integer weights. In *40th IEEE Symposium on Foundations of Computer Science (FOCS)*. 605–615.

SINT, L. AND DE CHAMPEAUX, D. 1977. An improved bidirectional heuristic search algorithm. *Journal of the ACM 24,* 2, 177–191. Announced at IJCAI 1975.

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

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

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

Sommer, C. 2011. More compact oracles for approximate distances in unweighted planar graphs.

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

Song, Q. and Wang, X. 2011. Efficient routing on large road networks using hierarchical communities. *IEEE Transactions on Intelligent Transportation Systems 12*, 132–140.

Sprague, A. P. 2007. $O(1)$ query time algorithm for all pairs shortest distances on permutation graphs. *Discrete Applied Mathematics 155,* 3, 365–373.

Sprague, A. P. and Takaoka, T. 1999. $O(1)$ query time algorithm for all pairs shortest distances on interval graphs. *International Journal of Foundations of Computer Science 10,* 4, 465–472.

Stout, B. 1999. Smart move: Intelligent path-finding. Online at `http://www.gamasutra.com/view/feature/3317/smart_move_intelligent_.php`.

Thorup, M. 1999. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM 46,* 3, 362–394. Announced at FOCS 1997.

Thorup, M. 2000. Floats, integers, and single source shortest paths. *Journal of Algorithms 35,* 2, 189–201. Announced at STACS 1998.

Thorup, M. 2004. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM 51,* 6, 993–1024. Announced at FOCS 2001.

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

Thorup, M. and Zwick, U. 2005. Approximate distance oracles. *Journal of the ACM 52,* 1, 1–24. Announced at STOC 2001.

Thorup, M. and Zwick, U. 2006. Spanners and emulators with sublinear distance errors. In *17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 802–809.

Timpf, S. and Frank, A. U. 1997. Using hierarchical spatial data structures for hierarchical spatial reasoning. In *International Conference on Spatial Information Theory: A Theoretical Basis for GIS (COSIT)*. 69–83.

Tretyakov, K., Armas-Cervantes, A., García-Bañuelos, L., Vilo, J., and Dumas, M. 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.

Trissl, S. and Leser, U. 2007. Fast and practical indexing and querying of very large graphs. In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*. 845–856.

Ungar, P. 1951. A theorem on planar graphs. *Journal of the London Mathematical Society s1-26,* 4, 256–262.

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

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

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

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

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

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

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

Watts, D. J. and Strogatz, S. H. 1998. Collective dynamics of 'small-world' networks. *Nature*, 440–442.

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

Woodruff, D. P. 2006. Lower bounds for additive spanners, emulators, and more. In *47th IEEE Symposium on Foundations of Computer Science (FOCS)*. 389–398.

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

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

Wulff-Nilsen, C. 2012b. Approximate distance oracles with improved query time. *arXiv abs/1202.2336*.

Xiao, Y., Wu, W., Pei, J., Wang, W., and He, Z. 2009. Efficiently indexing shortest paths by exploiting symmetry in graphs. In *12th International Conference on Extending Database Technology (EDBT)*. 493–504.

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

Yen, J. Y. 1971. On Hu's decomposition algorithm for shortest paths in a network. *Operations Research 19,* 4, 983–985.

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

Zhan, F. B. 1997. Three fastest shortest path algorithms on real road networks: Data structures and procedures. *Journal of Geographic Information and Decision Analysis 1,* 1, 69–82.

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

Zhao, J. L. and Cheng, H. K. 2001. Graph indexing for spatial data traversal in road map databases. *Computers & OR 28,* 3, 223–241.

Zhao, J. L. and Zaki, A. 1994. Spatial data traversal in road map databases: A graph indexing approach. In *3rd International Conference on Information and Knowledge Management (CIKM)*. 355–362.

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

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

Ziliaskopoulos, A. K., Kotzinos, D., and Mahmassani, H. S. 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.

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

Zwick, U. 2002. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM 49,* 3, 289–317.

...