Distance oracles for timetable graphs - research

František Hajnovič
Faculty of Mathematics, Physics and Informatics
Comenius University
Bratislava, Slovakia
ferohajnovic@gmail.com

April 2012

Abstract

In this paper, we would like to summarize the work done on distance oracles, shortest path queries and shortest timetable connection queries. The list of papers is not comprehensive, but offers a good look into the state of research in this area.

Keywords: **distance oracles**, **timetable graphs**, **shortest path**

Declaration: This paper is just a summary of the listed articles and the ideas presented are all coming from the respective article being summarized. I re-explain several points deemed important for the purposes of this research. At times, I borrow the exact words of the authors of the article at question.

---

1 List of considered papers

Distance labeling:

1. 2.1 Distance Labeling in Graphs [GPPR04]
2. 2.2 Reachability and distance queries via 2-hop labels [CHKZ03]

Distance oracles:

1. 3.1 Approximate Distance Oracles [TZ05]
2. 3.8 Approximate Shortest Path and Distance Queries in Networks [Som10]
3. 3.2 Exact Distance Oracles for Planar Graphs [MS10]
4. 3.3 f-Sensitivity Distance Oracles and Routing Schemes [CLPR10]
5. 3.4 Improved Distance Oracles for Avoiding Link-Failure [CR02]
6. 3.5 A Compact Routing Scheme and Approximate Distance Oracle for Power-Law Graphs [CSTW09]
7. 3.6 Ramsey Partitions Based Approximate Distance Oracles [Fre08]
8. 3.7 A Sketch-Based Distance Oracle for Web-Scale Graphs [SGNP10]
9. 3.9 Distance Oracles for Sparse Graphs [SVY09]
10. 3.10 Distance Oracles for Spatial Networks [SS09]

Route-planning:

1. 4.1 Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks [GSSD08]
2. 4.2 Engineering Route Planning Algorithms [DSSW09]
3. 4.3 Highway Dimension, Shortest Paths, and Provably Efficient Algorithms [AFGW10]
4. 4.4 Highway Hierarchies Hasten Exact Shortest Path Queries [SS05],
5. 4.5 Route Planning in Road Networks [Sch08]
6. 4.6 TRANSIT Ultrafast Shortest-Path Queries with Linear-Time Preprocessing [BFM06]
7. 4.7 In Transit to Constant Time Shortest-Path Queries in Road Networks [BFM$^+$]

Time-dependent routing:

1. 5.1 Time-Dependent SHARC-Routing [Del08]
2. 5.2 Time Dependent Contraction Hierarchies - Basic Algorithmic Ideas [San08]
3. 5.3 Time-Dependent Contraction Hierarchies [BDSV09]

Timetables related:

1. 6.1 Experimental comparison of shortest path approaches for timetable [PSWZ04]
2. 6.2 Engineering Time-Expanded Graphs for Faster Timetable Information [DPW09]
3. 6.3 Using Multi-level Graphs for Timetable Information in Railway Systems [SWZ02]
4. 6.4 Timetable Information: Models and Algorithms [MHSWZ07]

Others:

1. 7.1 Complex networks: small-world, scale-free and beyond [WC03]

Unclassified:

1. 8.1 Reach for A*: Efficient Point-to-Point Shortest Path Algorithms [GKW06]
2. 8.2 Car or Public Transport–Two Worlds [Bas09]

2 Distance labeling

2.1 Distance Labeling in Graphs

Author(s):

- Cyril Gavoille
- David Peleg
- Stéphane Pérennes
- Ran Raz

Year: 2004

Institution(s):

- Université Bordeaux
- The Weizmann Institute of Science, Israel
- Université de Nice-Sophia Antipolis

Concern: The paper considers the problem of *distance labeling in graphs*. Distance labeling itself is an approach that assigns each node of the graph some label in such a way, that queries for distance between any pair of vertices can be answered using only the information provided by the labels. This paper talks about various lower and upper bounds on the length of such labels for individual classes of graphs, as well as *time complexity of the extraction of the distance* from the labels.

Outline: After some necessary **definitions** and short mention of the **related work**, the results concerning **upper bounds** (sufficient sizes of labels) are presented. Those include an upper bound for general graphs and graphs with a given separator. Next, **lower bounds** are provided for general graphs, graphs with a given separator, planar graphs and trees, most of the results obtained by a generalized theorem proven before. Finally, in the last section, the **time complexity of extracting the actual distance** from the labels is considered, and shown to be too high for practical use for some extreme cases in the class of general graphs.

We should note, that we may think of distance labeling as of some form of distance oracle based method. Obtaining the labels requires some preprocessing time and results in the labels, that together take some space (size of the DO) . Query time is equivalent to the distance decoding (see below). Thus it is just a different (a bit more restricted) approach to the same thing. Let us look closer at the results obtained.

Distance labeling $< L, f >$ for a graph G consists of **node labeling** L and **distance decoder** f. The first is a function that assigns each node a label. Second is a function that computes the distance between the nodes, given their labels. $< L, f >$ is a **distance labeling scheme** for family (class) of graphs $\mathcal{G}$ if it is a correct distance labeling for every $G \in \mathcal{G}$.

Now we are interested in such a distance labeling scheme, that minimizes the maximal label length in all graphs of the given graph family. The obtained results are given in the table 1.

Table 1: Results of [GPPR04] on the necessary length of labels. In each entry we consider a n-vertex subclass of the given class

Class of graphs	Lower bound	Upper bound
general	$n/2 - O(1) = \Omega(n)$	$11n + O(\log n \log\log n) = O(n)$
unweighted binary trees	$\log^2 n/8 - O(\log n) = \Omega(\log^2 n)$	
binary trees with integral weights up to M	$\theta((\log M + \log n)\log n)$	$\theta((\log M + \log n)\log n)$
planar	$\Omega(n^{1/3})$	
bounded degree	$\Omega(\sqrt{n})$	
having $r(n) - separator$	$\sum_{labels} = \Omega(r(n)(n - r(n)) - 2n\log n)$	$O(R(n)\log n + \log^2 n)$

Note: In the last row (of table 1), $R(n) = \sum_{i=0}^{\log_{3/2} n} r(n(2/3)^i)$. We say a graph class $\mathcal{G}$ has a $\boldsymbol{r(n) - separator}$ if for every connected graph $G \in \mathcal{G}$ of n vertices there is a separator of size at most $r(n)$ such that upon its deletion we obtain components that are again graphs from $\mathcal{G}$. Moreover, all of these created components have size at most $r(2n/3)$.

Other results are concerning the time complexity of the distance decoder. The main result is rather technical, so we'll demonstrate it on a simpler corollary:

Let $\mathcal{G}$ be the family of all graphs. For infinitely many n, we have a $G \in \mathcal{G}$, for which:

1. There is an exact (with stretch 1) distance labeling scheme with maximal label $\leq 3\log n + o(\log n)$.
2. However, for any distance labeling scheme with stretch < 2 that satisfies $\sum_{u \in V(G)} |L(u, G)| \leq n^2/2 - O(n \log n)$ (that is also the one from point 1.)), the time and space complexities of distance decoder function f are larger than any constant size stack of exponentials ($2^{2^{\cdots^{2^n}}}$ - constant height of the power stacking).

Put informally, for almost any size of graph, there is such graph for which we can find a labeling producing only short labels. But for that graph, there will not be any distance labeling scheme that would answer distance queries in a small time/space complexity.

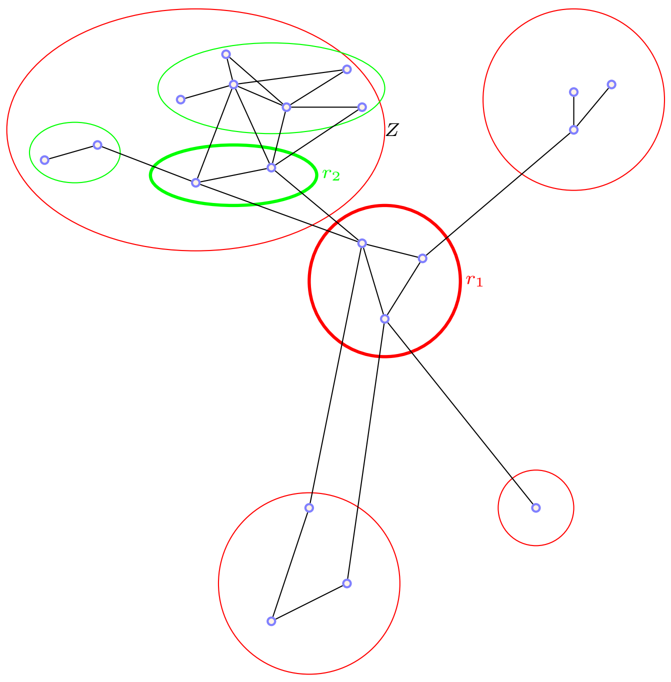

Figure 1: Graph has a recursive 3-separator (so $r(n)$ is constant in this case): None of the components, after removing the separator r_1, contains more than $2/3n$ vertices. This holds recursively (e.g. after removing the separator r_2, the green areas are not too large with respect to Z). There is no 1-separator, however, as there is no way to decompose area Z with separator of size one in a mentioned balanced manner.

2.2 Reachability and distance queries via 2-hop labels

Author(s):

- Edith Cohen
- Eran Halperin
- Haim Kaplan
- Uri Zwick

Year: 2002

Institution(s):

- AT&T Labs Research
- Tel-Aviv University

3 Distance oracles

3.1 Approximate Distance Oracles

Author(s):
- Mikkel Thorup
- Uri Zwick

Year: 2004

Institution(s):
- AT&T Labs Research
- Tel-Aviv University

Concern: Thorup and Zwick provided an important result in this paper, which is further mentioned in other works in the distance oracle area. They showed, that given an undirected weighted graph of n vertices and m edges and a chosen integer $k \geq 1$, we can build a distance oracle answering shortest path queries, the oracle having following properties:
- preprocessing takes $O(kmn^{1/k})$ expected time
- resulting distance oracle is of size $O(kn^{1+1/k})$
- answering queries takes $O(k)$ time
- stretch of the anwser(i.e. the worst ratio of returned path against the optimal value) is $2k-1$ at most

Outline: First, there is a brief **history of the work done** in the area up to the date (2004) and a summary of known methods at that time, compared to the distance oracle of Thorup and Zwick. Following is the presentation of an **approximate distance oracle for metric spaces**, whose preprocessing time is *expeceted* $\mathcal{O}(n^2)$, due to the randomized nature of the preprocessing algorithm. The **derandomization** (with a slight loss in efficiency) follows and the **adjustment** is made for graphs not explicitly embedded in a metric space. In the final section, there is an argument based on the girth conjecture stating that the distance oracle of Thorup and Zwick is **essentially optimal** with regard to amount of storage needed.

The result of Thorup and Zwick says, that we can make compromises - by increasing preprocessing time and the size of the DO, we can achieve better stretch and query time, and vice versa. Note, however, that only values of k up to $\log n$ (when preprocessing and size of DO reach their minimum) are desirable, higher values of k already increase all the factors. Also note, that increasing query time does not increase the accuracy, which is a bit counterintuitive. Christian Sommer's DO-based method (see 3.8) achieves different kind of trade-offs.

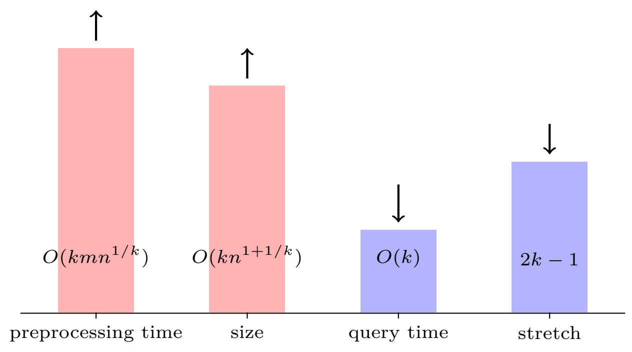

Figure 2: Possible trade-offs among preprocessing time, size of DO, query time and stretch

Lower bound on space

Let us concisely describe, why the proposed distance oracle is essentially optimal, as it is an interesting discussion. First, we need a few definitions. A t-**Spanner** is a subgraph of a graph, that preserves distances between any pair of vertices up to the factor of t. In other words - any distance in the t-spanner is at most t times the distance in the original graph. We should note, that we consider only subgraphs whose set of vertices is the same as in the original graph.

Next, we have the **girth** of a graph, which is the size of its smallest cycle. There is a close relation between spanners and girth of a graph:

Theorem 1: the girth of a graph is at least $t+2 \iff$ no *proper* subgraph of it is a t-spanner

which is trivial to prove. Another result states:

Theorem 2: every n-vertex graph with at least $n^{1+1/k}$ edges contains a cycle of size $\leq 2k$, thus is of girth at most $2k$.

Combining the two theorems, we have:

1. n-vertex graph with $n^{1+1/k}$ edges $\Rightarrow$ girth at most $2k$ (from the second theorem)
2. girth at most $(2k-1)+1 \Rightarrow \exists\ (2k-1)$-spanner (from the first theorem, where $t=(2k-1)$)

Thus we can conclude:

Theorem 3: every graph on n vertices has a $(2k-1)$-spanner with $\mathcal{O}(n^{1+1/k})$ edges

If the original graph is sparse enough, we can take it whole as a spanner, otherwise we apply the two mentioned implications.

The discussion above was concerning undirected and unweighted graphs. However, the last result is valid for weighted graphs as well. We need to extend the definitions of t-spanner and girth - the extension of the definition of a t-spanner for weighted graphs is straightforward and girth stays defined the way it was (that is - the size, i.e. the number of edges of the smallest cycle). The first theorem will no longer hold, however, the necessary right-to-left implication used in point 2. above (where we actually use the inverse of that implication) will still be valid (from any cycle, we can simply remove the most expensive edge while not increasing any distance by the factor more than $(2k-1)$). The second theorem used is not affected by the graph being weighted. Thus the conclusion we drew for unweighted graphs holds for weighted graphs as well.

However, it is conjectured by Erdös (and others as well), that

Conjecture 1: for any $k \geq 1$ there are graphs with $\Omega(n^{1+1/k})$ edges and girth greater than $2k$

The $2k$ can actually be pushed even more to $2k+1$, if we realize, that any graph contains a bipartite graph with at least half the edges. Thus we can obtain a graph with all cycles of an even size and still satisfying $|E| = \Omega(n^{1+1/k})$.

Combining theorem 1 and the conjecture, and repeating theorem 3, we have:

1. $\exists$ graph G of n vertices with $\Omega(n^{1+1/k})$ edges, no proper subgraph of which is a $2k-1$-spanner (conjecture and theorem 1)
2. G is a graphs of n vertices $\Rightarrow G$ has a $(2k-1)$-spanner with $\mathcal{O}(n^{1+1/k})$ edges (theorem 3)

Thus it is shown, that for a $(2k-1)$-spanner $\Omega(n^{1+1/k})$ edges are **sufficient**, though the same amount of edges is **necessary** for some cases (in number 1., we would have to take the whole graph as a spanner). As written in the article, any distance oracle capable of producing paths witnessing the estimated distances must explicitly or implicitly contain an appropriate spanner. Thus provided that the conjecture holds, the size of any distance oracle with stretch $2k-1$ is $\Omega(n^{1+1/k})$.

The final argument above is a bit high-level, and a different one, more formal is provided in the article. Also, the full optimality was not achieved, as the space complexity of the algorithm of Thorup and Zwick was expressed in words, rather than in bits, leaving a logarithmic gap.

3.2 Exact Distance Oracles for Planar Graphs

Author(s):
- Shay Mozes
- Christian Sommer

Year: 2011
Institution(s):
- Brown University
- MIT

3.3 f-Sensitivity Distance Oracles and Routing Schemes

Author(s):
- Shiri Chechik
- Michael Langberg
- David Peleg
- Liam Roditty

Year: 2011
Institution(s):
- The Weizmann Institute of Science
- Open University of Israe
- Bar-Ilan University

3.4 Improved Distance Oracles for Avoiding Link-Failure

Author(s):
- Rezaul Alam Chowdhury
- Vijaya Ramachandran

Year: 2002
Institution(s): The University of Texas at Austin

Concern: Paper attempts to solve the problem of preprocessing an edge-weighted **directed** graph to answer queries for shortest path avoiding specific link.

Outline: **Introduction** and **necessary notations** are followed by the presentation of the two algorithms - DT-1 and DT-2 - that are later upgraded in the paper. CR-1 and CR-2 - the **constructed algorithms** are proposed, each with a code listing, preprocessing description and a justification of correctness.

Two functions are considered:
- $distance(x, y, u, v)$ - shortest distance from x to y avoiding edge (u, v).
- $path(x, y, u, v)$ - shortest path from x to y avoiding edge (u, v).

Assumption is that the time between two successive link failures are long enough to compute a new data structure in the background.

The authors build upon algorithms $DT-1$ and $DT-2$ presented in paper by Demetrescu and Thorup. Those are not very complicated - $DT-1$ divides each shortest path from x to y to $O(\log n)$ segments, storing shortest distance from x to y avoiding each of these segments in a table. $DT-2$ divides each shortest-path tree $T(x)$ into $O(\sqrt{n})$ bands of independent paths. Again, shortest distance avoiding these bands are precomputed.

The algorithms introduced in this paper are called $CR-1$ and $CR-2$. Their respective properties are shown in the table below.

Table 2: Results of [CR02]

Algorithm	**Preprocessing time**	**Space**	**Query time**
$CR-1$	$O(mn^2\log n + n^3\log^2 n)$	$O(n^2\log n)$	$O(1)$
$CR-2$	$O(mn\log^2 n + n^2\log^3 n)$	$O(n^2\log^2 n)$	$O(\log n)$

3.5 A Compact Routing Scheme and Approximate Distance Oracle for Power-Law Graphs

Author(s):
- Wei Chen
- Christian Sommer
- Shang-Hua Teng
- Yajun Wang

Year: 2009
Institution(s):
- Microsoft Research
- MIT

3.6 Ramsey Partitions Based Approximate Distance Oracles

Author(s): Chaya Fredman
Year: 2008
Institution(s): The Open University Of Israel

3.7 A Sketch-Based Distance Oracle for Web-Scale Graphs

Author(s):
- Atish Das Sarma
- Sreenivas Gollapudi
- Marc Najork
- Rina Panigrahy

Year: 2010
Institution(s):
- Microsoft Research
- Georgia Institute of Technology

Concern: This paper considers real-world large-scale graphs, like web graph or social networks, and tries to provide algorithms to answer shortest path queries that would be efficient and giving accurate enough results at the same time.

Outline: A **summary of related methods** is presented at the beginning. Next, the **algorithm is described** (Online-Common-Seed), subsequently **extended to directed graphs**. Finally, **experiments** on a crawl of the web are performed, comparing the algorithm to the Bourgain's method.

The provided algorithm looks the following in the preprocessing phase:

$$SAMPLING \xrightarrow{\log|V|\ times} \forall u \in V:\ SAMPLE[u] \xrightarrow{k\ times} \forall u \in V:\ SKETCH[u]$$

$\boldsymbol{SAMPLING}$ is just picking random vertices (seeds) from V. $\boldsymbol{SAMPLE[u]}$ for some $u \in V$ is a set of couples (w_i, δ_i) where w_i is the closest seed to u from a given sampling, and δ_i is its distance from u. Sampling is done $\log|V|$ times - starting with 1 sampled seed and doubling each subsequent time. Finally, $\boldsymbol{SKETCH[u]}$ is a union of $SAMPLE[u]$ created when we iterated the whole process k times.

The **preprocessing** thus takes $O(k \cdot n^2)$, using one breath-first search from each seed set.
The algorithm $\boldsymbol{ONLINE - COMMON - SEED(x, y)}$ for answering queries compares the two $SKETCH$es of the two vertices and finds a command seed s, which minimizes the distance $d(x, s) + d(s, y)$. In case of no common seed, we output ∞. This algorithm always gives upper bound on actual shortest distance.

Further, there is a proof that for $k = \Theta(n^{1/c} polylog(n))$, we get a $2c - 1$ approximation of the actual distance with high probability.

Another algorithm called $\boldsymbol{ONLINE - BOURGAIN(x, y)}$ is mentioned, since $\boldsymbol{ONLINE - COMMON - SEED(x, y)}$ is later compared to it. $\boldsymbol{ONLINE - BOURGAIN(x, y)}$ provides a lower bound on the actual distance. Unlike $ONLINE - COMMON - SEED$, it takes a maximum of $|d(x, S) - d(y, S)|\ \forall S$ where S are seed sets from $SAMPLING$. Again, a similar theorem applies: for $k = \Theta(n^{1/c})$, we get a $O(2c-1)$ approximation of the actual distance with high probability.

Both algorithms answer queries with time complexity $O(\log n)$.

Algorithms are also modified to directed graphs.

Experiments were run on a large crawl of the web graph, containing about 65000000 web pages and 420000000 distinct URLs. For $k = 1$ (guaranteed approximation of only $\log n$) they still obtained 1.2 approximation ratio with $ONLINE - COMMON - SEED$ and 2.14 ratio with $ONLINE - BOURGAIN$, thus showing that the theoretical guarantee could be possibly improved.

3.8 Approximate Shortest Path and Distance Queries in Networks

Author(s): Christian Sommer
Year: 2010
Institution(s): The University of Tokyo

Concern: In his PhD thesis, Christian Sommer investigates the problem of *efficiently computing exact and approximate shortest path queries* in graphs. Apart from that, the thesis gives a clear understanding of the problematics, terminology and related work in the area.

Outline: A very well organized thesis begins with plenty of **motivation**, continues with extensive **definitions** and **summary** of work done on shortest path query processing. The **three main results** are then devoted a chapter, each beginning with an introduction and ending with a conclusion and open points.

The thesis provides three *main results*:

1. There is no exact DO for sparse general graphs, that would be of size $O(m)$ and have a constant query time. More specifically, given query time t and multiplicative stretch α, a required space for an exact DO is $n^{1+\Omega(1/\alpha t)}$.
2. Adapting the DO of Thorup and Zwick ([TZ05]) and adjusting it for power-law graphs. The adjustment takes advantage of the high-degree nodes in power-law graphs and selects them for

landmarks in the algorithm. A theoretical proof shows why such an approach yields a good heuristics, efficiently approximating shortest paths e.g. in Internet-like topologies.

3. An approximation DO for general undirected graphs with positive edge weights is presented. Based on random sampling and graph Voronoi duals, it provides good theoretical trade-offs between stretch and preprocessing and query times (stretch against the two). Even better is the performance in practice. Compared to DO of Thorup and Zwick, it offers a different type of trade-off: stretch of the answer is reduced when a longer query time is allowed.

3.9 Distance Oracles for Sparse Graphs

Author(s):

- Christian Sommer
- Elad Verbin
- Wei Yu

Year: 2009

Institution(s):

- The University of Tokyo and NII
- Tsinghua University

3.10 Distance Oracles for Spatial Networks

Author(s):

- Jagan Sankaranarayanan
- Hanan Samet

Year: 2009

Institution(s): University of Maryland

4 Route-planning

4.1 Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks

Author(s):
- Robert Geisberger
- Peter Sanders
- Dominik Schultes
- Daniel Delling

Year: 2008
Institution(s):
- Universitat Karlsruhe

Concern: Yet another technique is presented for answering shortest-path queries in road-networks. The idea is based on the notion of contraction. The graph is iteratively contracted, to obtain a set of shortcuts. A special version of bidirectional Dijkstra's algorithm is then run on the enhanced graph to obtain very good speed-up of 4 orders of magnitude against the classical Dijkstra's algorithm.

Outline: The **idea is sketched** at the beginning, a brief comparisons to **related work** (mostly to Highway hierarchies and Transit-node routing) are made. The **preprocessing phase** is described, mostly the heuristics that determine the node ordering in the contraction process. **Query computation** itself is explained, some **experiments** carried out. The **conclusion** gives directions for future improvements and research.

The **contraction** of a node means, that we remove it from the graph and add edges (**shortcuts**) between some of the pairs of its neighbors, so that we preserve all the distances of the original graph. Thus if the contracted vertex v was on some shortest path $P = (u_1, u_2, ..., u_k, v, u_{k+2}, ..., u_s)$, we will add an edge from u_k to u_{k+2} with corresponding distance to preserve the distance between u_1 and u_s in the graph with v contracted.

Now the preprocessing of the algorithm works as follows: Given an ordering of the nodes of the graph, contract them one by one until single vertex remains. The shortcuts created on the way are added to the original edge set.

On such a graph upon query, we run a bidirectional Dijkstra's algorithm, that is allowed to expand only alongside the node-ordering, that is, only vertices with a higher *rank* then the currently scanned vertex are examined (where *rank* is the vertex' position in the node ordering).

Very much depends on the node ordering, as the resulting set of shortcuts created may be too large, given a wrong permutation, thus queries would take too long to compute. Good results were obtained using simple heuristics for node-ordering: Select v as the next node to contract if v maximizes $(deg(v) - add(v))$ where $add(v)$ is the number of added shortcuts upon contraction of v.

4.2 Engineering Route Planning Algorithms

Author(s):
- Peter Sanders
- Dominik Schultes

Year: 2009
Institution(s):
- Universitat Karlsruhe
- Institut für Theoretische Informatik

Concern: Article provides an extensive summary (the bibliography lists over 70 entries) of existing algorithms for route planning in transportation networks.

Outline: An article lists results, classified into several categories. **Static-routing** section gives the lists of the classical results, those exploiting hierarchies, goal-directed searches and various combinations. Following is the **chronological summary** which clearly illustrates the progress obtained throughout the years. Other **generalizations** of the problem at hand follow and **time-dependent routing** and techniques adapted for that purpose are reviewed. **Implementations** and **methodology of experiments** are briefly discussed. **Conclusion** provides open points and future directions of the research.

4.3 Highway Dimension, Shortest Paths, and Provably Efficient Algorithms

Author(s):
- Ittai Abraham
- Amos Fiat
- Andrew V. Goldberg
- Renato F. Werneck

Year: 2010
Institution(s):
- Microsoft Research
- Tel-Aviv University

Concern: Introducing a notion of a highway dimension, this paper established a parameter which considerably influences the efficiency of several *exact* methods for shortest path computing in general graphs. These, while showing very good results in practice, lacked theoretical guarantees on their time complexities. In the results of this paper, low highway dimension was shown to bind the time complexities of the methods to practical values.

Outline: After the **introduction** with interesting facts and motivation, the **definition of highway dimension** and shortest-path cover is introduced. The paper then describes the **notion of shortcuts**, a technique used with many methods for shortest-path computing, particularly with Reach, Contraction Hierarchies, Transit Nodes and SHARC. A shortcut-based **preprocessing** of the input graph is described, which will be common to the mentioned methods. Authors then take the **methods one by one**, and for each (with possible adjustments) show the low query time complexity bound, provided low highway dimension of the input graph. In the last section, a generative model of networks with low highway dimension is suggested. **Conclusion** leaves some open points, most notably the slow time of the preprocessing.

Let us provide the definition of the highway dimension: **Highway dimension** (HD) for an undirected, edge-weighted graph G is the smallest integer h, such that
$\forall r \in R^+, \forall u \in V_G, \exists S \subseteq B_{u,r}, |S| \leq h$, such that $\forall v, w \in B_{u,4r}$:
if $|P(v,w)| > r$ and $P(v,w) \subseteq B_{u,4r}$ then $P(v,w) \cap S \neq \emptyset$
where $B_{u,r} = \{v \in G | d(u,v) \leq r\}$ and is called **ball** of radius r centered at u.

4.4 Highway Hierarchies Hasten Exact Shortest Path Queries

Author(s):
- Peter Sanders
- Dominik Schultes

Year: 2005
Institution(s):
- Universitat Karlsruhe
- Universitat des Saarlandes

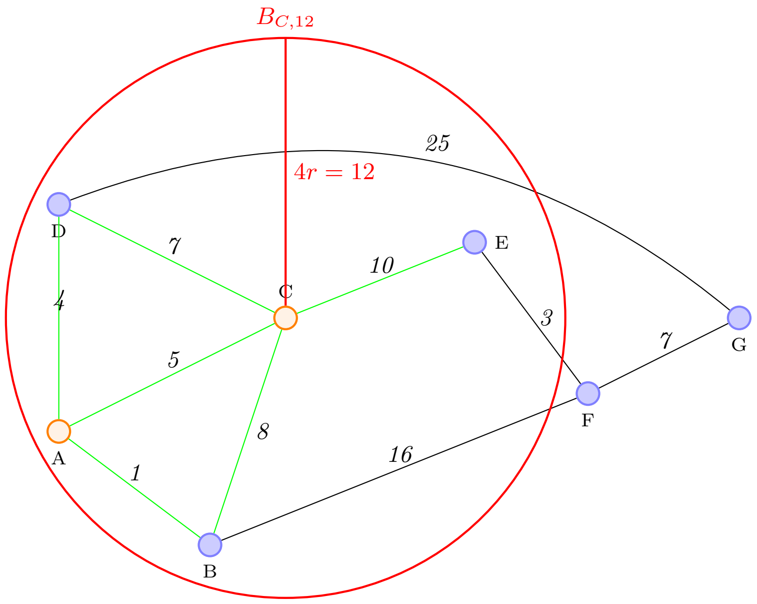

Figure 3: Demonstration of a definition of HD. We chose some r ($r = 3$) and some vertex v ($v = C$) to root the ball $B_{v,4r}$. All the shortest paths *longer* than r *inside* the ball have to contain a vertex from S (orange vertices C and A in our case). The upper bound on $|S|$, considering any ball with any radius, is the required highway dimension. Note: in our case, we had to choose also A to the set S, since a shortest path from B to D does not include C.

Concern: An algorithm for answering exact shortest paths queries in undirected, weighted graphs is presented. Based on a construction of a hierarchy of increasingly smaller and more important highway levels, the authors have reached a very good speedup compared to Dijkstra's algorithm (2000 times faster on the road network of the US).

Outline: After a short **introduction**, there is a section describing **related work**, namely methods based on separators, geometric containers, bit vectors and landmarks. For all of them, a small comparison with presented algorithm is made. Then, **highway hierarchy is defined** theoretically and subsequently its construction is described. A **modification of bidirectional Dijkstra**'s algorithm follows, with slight improvements short after. Finally, **experiments** on the road networks of USA and Europe are presented with clear results in favour of Highway hierarchies algorithm and in the **concluding discussion**, possibilities for further tweaks are mentioned

We will now shortly describe the algorithm and its main idea.

Consider, for each node w of the graph, its H closest neighbors and denote that set by $N_H(w)$. A **highway edge** will be such an edge (u, v), that is on a shortest path from some s to some t, whereas $u \notin N_H(t)$ and $v \notin N_H(s)$.

Basically, the highway hierarchy is built by iterating the following procedure:

$$\text{input graph } G \xrightarrow{\text{select highway edges}} \text{highway network } G_1 \xrightarrow{\text{contract}} \text{contracted highway network } G'_1$$

The selection of highway edges was already described. The contraction means two things:

- Take only the vertices with degree ≥ 2
- Substitute paths whose internal vertices are of degree 2 by one edge.

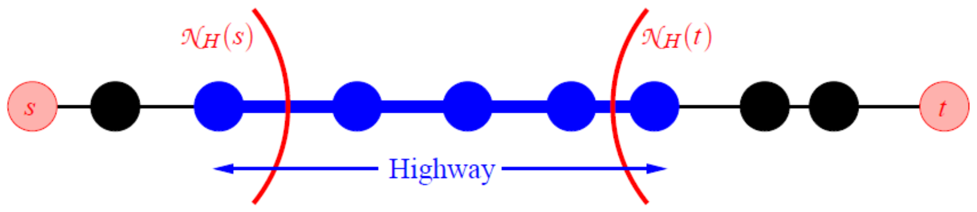

Figure 4: Highway edges

The removed parts of the graph are be called **components**. This way, we continually build smaller and smaller layers of the hierarchy (formed by highway networks - G_1), each time supplying G_1' for the input of the next iteration. The lowest level is the original graph itself.

On such a hierarchy (connected by vertical edges between vertices of the same type on neighboring levels of the hierarchy), a bidirectional Dijkstra's algorithm is run, with two constraints that can be informally described as:

- We do not follow one level of the hierarchy for too long
- Components of a given level of the hierarchy could be entered only when entering the level of the hierarchy itself

There are further optimizations described in the article, concerning speeding up the preprocessing, decreasing the size of the preprocessed information and the time of the query.

4.5 Route Planning in Road Networks

Author(s): Dominik Schultes
Year: 2008
Institution(s): Universitat Karlsruhe

4.6 TRANSIT Ultrafast Shortest-Path Queries with Linear-Time Preprocessing

Author(s):
- Holger Bast
- Stefan Funke
- Domagoj Matijevic

Year: 2006
Institution(s): Max-Planck-Institut fur Informatik, Saarbrucken, Germany

Concern: The article presents a simple, yet very effective method for point-to-point shortest path queries in road networks, based on a precomputation of distances among a small set of so called “transit nodes”. Upon query, the Dijkstra's algorithm is completely replaced by a few table look-ups, yielding impressive speed-ups in computation (6 orders of magnitude against the classical Dijkstra's algorithm).

Outline: A simple, **basic idea** is quickly explained, then **related work** is compared to the proposed TRANSIT method. The **algorithm** is then described, mostly its preprocessing phase. **Modifications** that improve the efficiency are further considered and **experiments** are conducted before **concluding**.

The surprisingly simple and effective idea is the following:

A small set of **transit nodes** is first computed, that would cover all long-enough shortest paths in the graph (note a similarity with the approach in [AFGW10]). Then, for each vertex, a set of the closest transit nodes (so called **access nodes**) will be computed. The algorithm stores all the pair-wise distances among the transit nodes and for every node, the distances to its access nodes. This will not be too much, as for a road network of US (24 million nodes), the number of transit nodes turned out to be about 10000 and number of access nodes 10 on average (per vertex).

A heuristics (e.g. air distance) is used to determine, if a posed query for distance between x and y is local, or not. In the latter case, all pairs of access nodes of x and y are considered and the one that minimizes the total distance is output. In the former case, some other exact method is used to determine the shortest path (e.g. Dijkstra's algorithm). Most queries (99%) are non-local, and thus require just a few table look-ups.

A simple way of obtaining not only the length of the path, but also the path itself is presented as well.

4.7 In Transit to Constant Time Shortest-Path Queries in Road Networks

Author(s):

- Holger Bast
- Stefan Funke
- Domagoj Matijevic
- Peter Sanders
- Dominik Schultes

Year: 2007

Institution(s):

- Universitat Karlsruhe
- Max-Planck-Institut fur Informatik

5 Time-dependent routing

5.1 Time-Dependent SHARC-Routing

Author(s): Daniel Delling
Year: 2008
Institution(s): Universitat Karlsruhe

5.2 Time Dependent Contraction Hierarchies - Basic Algorithmic Ideas

Author(s): Peter Sanders
Year: 2008
Institution(s): Universitat Karlsruhe

5.3 Time-Dependent Contraction Hierarchies

Author(s):
- Gernot Veit Batz
- Daniel Delling
- Peter Sanders
- Christian Vetter

Year: 2009
Institution(s): Universitat Karlsruhe

6 Timetables related

6.1 Experimental comparison of shortest path approaches for timetable

Author(s):

- Evangelia Pyrga
- Frank Schulz
- Dorothea Wagner
- Christos Zaroliagis

Year: 2004

Institution(s):

- University of Karlsruhe
- University of Patras

6.2 Engineering Time-Expanded Graphs for Faster Timetable Information

Author(s):

- Daniel Delling
- Thomas Pajor
- Dorothea Wagner

Year: 2008

Institution(s): University of Karlsruhe

Concern: This article studies quickest connection problem (earliest arrival problem) in time-expanded graphs. An extension of the model is presented, that takes transfers into account. It is then further preprocessed and optimized, so that once a (specifically adjusted) Dijsktra's algorithm is run on such a graph, it would correctly and effectively find the best connection. Existing speed-up techniques - Arc-Flags and ALT - which were designed for road networks are adapted for time-expanded graphs to yield a maximum speed-up factor of cca 56 against the classic Dijkstra's algorithm.

Outline: Article gives a brief **preliminary** and a definition of the time-expanded model. Subsequently, this **model is further refined** by bypassing unnecessary nodes, remodeling stations and introducing parameters that control this fine-tuning - still in the phase of preprocessing. Two **speed-up techniques** using Dijkstra's algorithm are then considered. First one - Node-blocking - devised by authors, is tailored for time-expanded graphs and tries to prune from the search the unnecessary branches of the computation. The second one is an adaptation of a combination of Arc-Flags and ALT, two techniques designed for road networks. Quite extensive **experiments** form much of the last part.

Several notes:

- It was left as an open point, how to handle updates in case of train (or carrier) delays, as updating time-expanded graphs is expensive
- Other techniques for road networks, like Highway Hierarchies, Contraction Hierarchies were considered, but their adaptation was said to be "too challenging"
- A speed-up of 56 is not a bad achievement, but compared to speed-ups achieved in road networks (about 3 million for Transit nodes combined with Edge-flags), one can wonder about possible improvements. However, speeding-up computation in time-dependent settings is much complicated, at least using techniques primarily devised for static routing.

6.3 Using Multi-level Graphs for Timetable Information in Railway Systems

Author(s):

- Frank Schulz
- Dorothea Wagner
- Christos Zaroliagis

Year: 2002

Institution(s):

- University of Konstanz
- University of Patras

6.4 Timetable Information: Models and Algorithms

Author(s):

- Matthias Muller-Hannemann
- Frank Schulz
- Dorothea Wagner
- Christos Zaroliagis

Year: 2006

Institution(s):

- Darmstadt University of Technology
- University of Karlsruhe
- University of Patras

Concern: The article considers various models and techniques how to deal with time-table information, primarily for the purpose of answering queries for shortest connection (solving earliest arrival problem - EAP). Time-expanded and time-dependent models are mostly discussed, both properly explained and later extended to model realistic requirements (like transfer time, cheapest connections, etc...). For multi-criteria optimization, approach that looks for Pareto sets is considered. Comparison between time-expanded and time-dependent models is made, with the latter doing better at EAP while the former being more convenient for multi-criteria optimization.

Outline: A short mention of **historical timetable information systems** (TRAINS, ARIADNE) precedes the **definitions** (time-expanded/dependent models). Next, how Dijkstra's algorithm works with individual models is explained. The **models are then extended** to take into account realistic situations and requirements. The **multi-criteria queries** are considered and the **performance** of individual models is reviewed, summarizing the experimental results of several cited studies. The final part **concludes**.

We will now introduce a few definitions, to make things clearer.

A timetable on a given graph G is a set $T_G = \{(x, y, p, q) | (x, y) \in E_G,\ p, q \in N,\ p < q\}$. An element of T is called an **elementary connection** and x $[y]$ and p $[q]$ are the **departure [arrival] node** and **time**. Graph G is the **underlying graph** of timetable T.

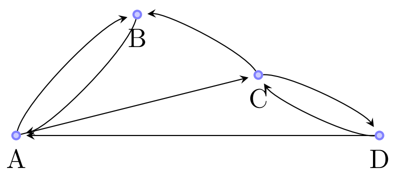

Figure 5: Underlying graph of the timetable in table 3

Place		Time	
From	**To**	**Departure**	**Arrival**
A	B	10:00	10:45
A	B	11:00	11:45
A	B	12:00	12:45
A	C	9:30	10:00
A	C	10:15	10:45
C	D	11:00	11:30
C	D	13:00	13:30
C	D	12:20	12:35
C	D	12:40	12:55
C	D	13:00	13:15
C	B	12:20	12:50
C	B	13:30	14:00
D	A	13:00	14:00

Table 3: An example of a timetable

A connection from x to y in a given timetable T_G is a sequence of *elementary connections* $e_1, e_2, ..., e_k,\ k \geq 1,\ e_i = (x_i, y_i, p_i, q_i)$, such that $x_0 = x,\ x_k = y$ and $\forall i \in \{2, ..., k\} : (x_i = y_{i-1},\ p_i \geq q_{i-1})$. Connection **starts** at the *departure time* p_1. **Size (length)** of the connection is $q_k - p_1$.

The main problem dealt with is the so called **earliest arrival problem (EAP)**: given a graph G, two of its vertices $x \neq y$, time t and a timetable T_G on the graph G, what is the shortest connection from x to y that starts at a time $s \geq t$. Let us now look at the two models.

Let T be a timetable on G. **Time-expanded graph** from T is an oriented graph G_T whose vertices are pairs $[v, t], v \in G$ and $\exists(x, y, p, q) \in T$ such that $p = t$ or $q = t$. Edges of G_T are formed by

1. $([x, p], [y, q]), ;\ \forall(x, y, p, q) \in T$ - connection edges
2. $([x, p], [x, q]), [x, p], [x, q] \in V_{G_T}, p < q$ and $\nexists[x, r] \in V_{G_T} : p < r < q.$ - waiting edges (e.g. waiting for a train in a station)

Weight w of any edge $((x, p), (y, q))$ in E_{G_T} is $w((x, p), (y, q)) = q - p$.

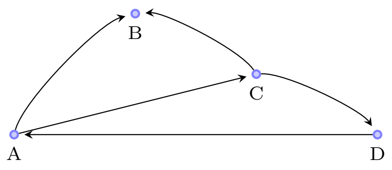

Figure 6: Time-dependent graph of the timetable in table 3

Let T be a timetable on G. **Time-dependent graph** of T is an oriented graph G_T whose vertices are $v \in G$. As for edges, $(u, v) \in E_{G_T} \iff \exists(u, v, p, q) \in T$. Weight w of any edge in E_{G_T} is defined during the run of the specific algorithm. Another point of view might be, that weights of the edges are functions, with time as their domain.

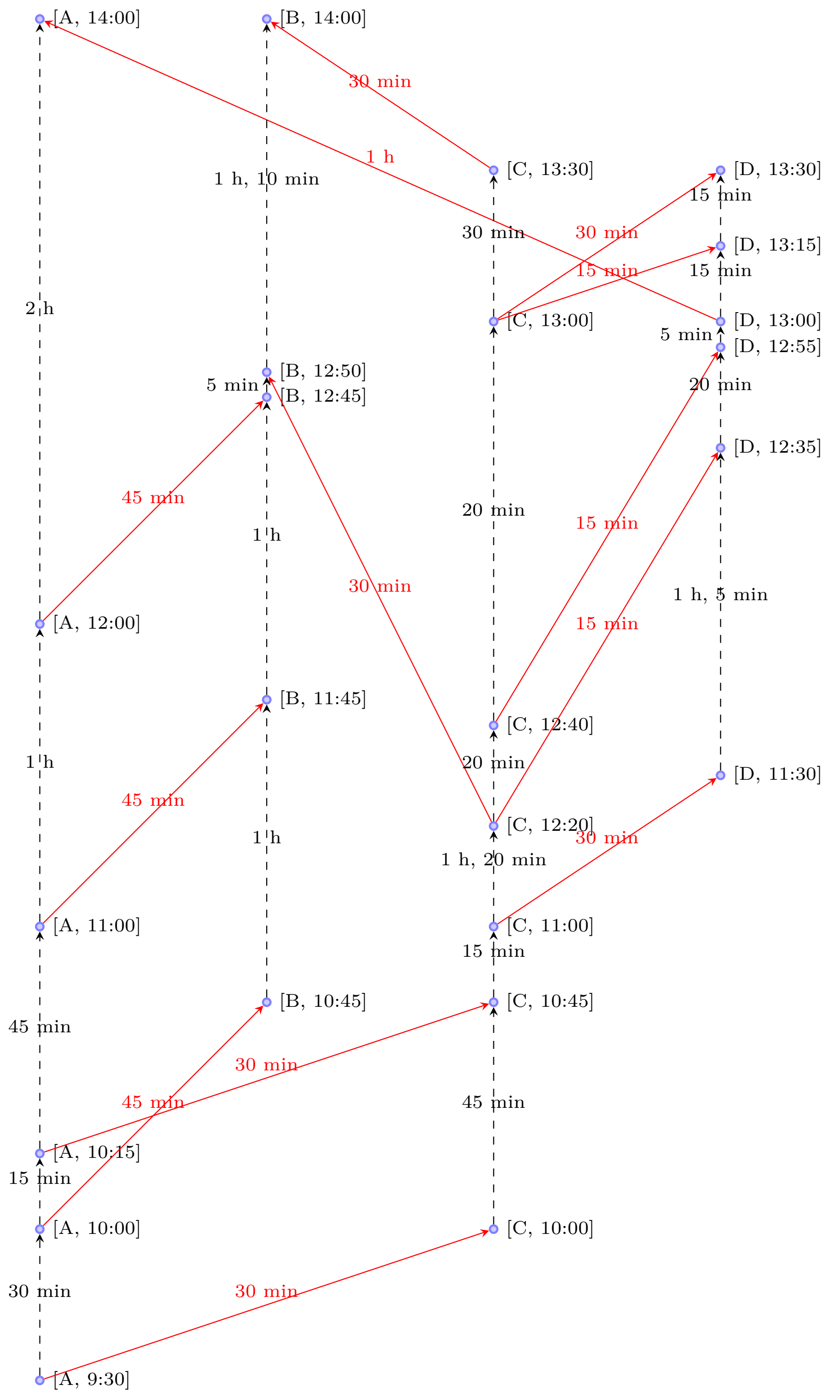

Figure 7: Time-expanded graph from the timetable in table 3

7 Others

7.1 Complex networks: small-world, scale-free and beyond

Author(s):

- Xiao Fan Wang
- Guanrong Chen

Year: 2003

Institution(s):

- Department of Automation, Shanghai Jiao Tong University
- Central Queensland University, Australia

8 Unclassified

8.1 Reach for A*: Efficient Point-to-Point Shortest Path Algorithms

Author(s):
- Andrew V. Goldberg
- Haim Kaplan
- Renato F. Werneck

Year: 2006
Institution(s):
- Microsoft Research
- Tel Aviv University, Israel
- Department of Computer Science, Princeton University

8.2 Car or Public Transport–Two Worlds

Author(s): Hannah Bast **Year:** 2009
Institution(s): Max-Planck-Institute for Informatics

9 List of main definitions

1. A t**-Spanner** is a subgraph of a graph, that preserves distances between any pair of vertices up to the factor of t
2. **girth** of a graph, which is the size of its smallest cycle
3. **Distance labeling** $< L, f >$ for a graph G consists of **node labeling** L and **distance decoder** f. The first is a function that assigns each node a label. Second is a function that computes the distance between the nodes, given their labels. $< L, f >$ is a **distance labeling scheme** for family (class) of graphs $\mathcal{G}$ if it is a correct distance labeling for every $G \in \mathcal{G}$.
4. $R(n) = \sum_{i=0}^{\log_{3/2} n} r(n(2/3)^i)$
5. We say a graph class $\mathcal{G}$ has a $\boldsymbol{r(n)-separator}$ if for every connected graph $G \in \mathcal{G}$ of n vertices there is a separator of size at most $r(n)$ such that upon its deletion we obtain components that are again graphs from $\mathcal{G}$. Moreover, all of these created components have size at most $r(2n/3)$.
6. **Highway dimension** (HD) for an undirected, edge-weighted graph G is the smallest integer h, such that
   $\forall r \in R^+, \forall u \in V_G, \exists S \subseteq B_{u,r}, |S| \leq h$, such that $\forall v, w \in B_{u,4r}$:
   if $|P(v,w)| > r$ and $P(v,w) \subseteq B_{u,4r}$ then $P(v,w) \cap S \neq \emptyset$
   where $B_{u,r} = \{v \in G | d(u,v) \leq r\}$ and is called **ball** of radius r centered at u.
7. **A timetable on a given graph** G is a set $T_G = \{(x,y,p,q)|(x,y) \in E_G,\ p,q \in N,\ p < q\}$. An element of T is called an **elementary connection** and x [y] and p [q] are the **departure** **[arrival] node** and **time**. Graph G is the **underlying graph** of timetable T.
8. **A connection from** x **to** y **in a given timetable** T_G is a sequence of *elementary connections* $e_1, e_2, ..., e_k,\ k \geq 1,\ e_i = (x_i, y_i, p_i, q_i)$, such that $x_0 = x,\ x_k = y$ and $\forall i \in \{2, ..., k\} : (x_i = y_{i-1},\ p_i \geq q_{i-1})$. Connection **starts** at the *departure time* p_1. **Size (length)** of the connection is $q_k - p_1$.
9. **earliest arrival problem (EAP)**: given a graph G, two of its vertices $x \neq y$, time t and a timetable T_G on the graph G, what is the shortest connection from x to y that starts at a time $s \geq t$.
10. Let T be a timetable on G. **Time-expanded graph** from T is an oriented graph G_T whose vertices are pairs $[v,t], v \in G$ and $\exists(x,y,p,q) \in T$ such that $p = t$ or $q = t$. Edges of G_T are formed by
    (a) $([x,p],[y,q]),;\ \forall(x,y,p,q) \in T$ - connection edges

   (b) $([x,p],[x,q]), [x,p],[x,q] \in V_{G_T}, p < q$ and $\nexists [x,r] \in V_{G_T} : p < r < q$. - waiting edges (e.g. waiting for a train in a station)

   Weight w of any edge $((x,p),(y,q))$ in E_{G_T} is $w((x,p),(y,q)) = q - p$.

11. Let T be a timetable on G. **Time-dependent graph** of T is an oriented graph G_T whose vertices are $v \in G$. As for edges, $(u,v) \in E_{G_T} \iff \exists(u,v,p,q) \in T$. Weight w of any edge in E_{G_T} is defined during the run of the specific algorithm.

10 List of main results

1. They showed, that given an undirected weighted graph of n vertices and m edges and a chosen integer $k \geq 1$, we can build a distance oracle answering shortest path queries, the oracle having following properties:
   - preprocessing takes $O(kmn^{1/k})$ expected time
   - resulting distance oracle is of size $O(kn^{1+1/k})$
   - answering queries takes $O(k)$ time
   - stretch of the anwser(i.e. the worst ratio of returned path against the optimal value) is $2k-1$ at most
2. the girth of a graph is at least $t+2 \iff$ no *proper* subgraph of it is a t-spanner
3. every n-vertex graph with at least $n^{1+1/k}$ edges contains a cycle of size $\leq 2k$, thus is of girth at most $2k$
4. every graph on n vertices has a $(2k-1)$-spanner with $\mathcal{O}(n^{1+1/k})$ edges
5. for any $k \geq 1$ there are graphs with $\Omega(n^{1+1/k})$ edges and girth greater than $2k$
6. There exists an exact distance labeling scheme with the overall size $O(nR(n)\log n + n\log^2 n)$ and distance computing in $O(\log n)$ time for a graph having $r(n) - separator$
7. Let $\mathcal{G}$ be the family of all graphs. For infinitely many n, we have a $G \in \mathcal{G}$, for which:
   (a) There is an exact (with stretch 1) distance labeling scheme with maximal label $\leq 3\log n + o(\log n)$.
   (b) However, for any distance labeling scheme with stretch < 2 that satisfies $\sum_{u \in V(G)} |L(u,G)| \leq n^2/2 - O(n\log n)$ (that is also the one from point 1.)), the time and space complexities of distance decoder function f are larger than any constant size stack of exponentials ($2^{2^{\cdots^{2^n}}}$ - constant height of the power stacking).

11 List of main DO methods

1. Distance oracle of **Thorup and Zwick**.
   - for general graphs
   - *preprocessing* - $\mathcal{O}(kmn^{1/k})$, *size* - $\mathcal{O}(kn^{1+1/k})$, *query time* - $\mathcal{O}(k)$, *stretch* - $2k-1$
2. Distance oracle of **Christian Sommer** for general graphs
   - for general graphs
   - based on random sampling and graph Voronoi duals
   - longer query time decreases stretch
3. Distance labeling of **Gavoille**
   - for graphs with $r(n) - separator$
   - *preprocessing* - depends, *size* - $\mathcal{O}(nR(n)\log W + n\log^2 n)$, *query time* - $\mathcal{O}(\log n)$, *stretch* - 1
4. **Highway hierarchies**
   - suitable for road networks
   - based on hierarchies, shortcuts
   - speed-up of 2000 against Dijkstra's algorithm
5. **Transit-node routing**
   - suitable for road networks
   - based on shortcuts, table-lookups
   - speed-up of 1000000 against Dijkstra's algorithm
6. **Contraction hierarchies**
   - suitable for road networks

- based on shortcuts, node contractions
- speed-up of 40000 against Dijkstra's algorithm

7. **Sketch-based distance oracle**
   - suitable for web network, social networks
   - based on landmarks
   -
   - *preprocessing* - $O(k \cdot n^2)$, *size* - $kn \log n$, *query time* - $O(\log n)$, *stretch* - for $k = \Theta(n^{1/c} polylog(n))$, stretch is $2c - 1$ *whp*.

References

[AFGW10] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato Fonseca F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In Moses Charikar, editor, *SODA*, pages 782–793. SIAM, 2010.

[Bas09] Hannah Bast. Efficient algorithms, 2009. ISBN 978-3-642-03455-8. URL `http://dx.doi.org/10.1007/978-3-642-03456-5_24`.

[BDSV09] Gernot Veit Batz, Daniel Delling, Peter Sanders, and Christian Vetter. Time-dependent contraction hierarchies. In Irene Finocchi and John Hershberger, editors, *ALENEX*, pages 97–105. SIAM, 2009.

[BFM$^+$] Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, and Dominik Schultes. In transit to constant time shortest-path queries in road networks.

[BFM06] Holger Bast, Stefan Funke, and Domagoj Matijevic. Transit— ultrafast shortest-path queries with linear-time preprocessing, 2006.

[CHKZ03] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. *SIAM J. Comput.*, 32(5):1338–1355, 2003.

[CLPR10] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-sensitivity distance oracles and routing schemes. In Mark de Berg and Ulrich Meyer, editors, *Algorithms – ESA 2010*, volume 6346 of *Lecture Notes in Computer Science*, pages 84–96. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-15774-5.

[CR02] Rezaul Alam Chowdhury and Vijaya Ramachandran. Improved distance oracles for avoiding link-failure. In Prosenjit Bose and Pat Morin, editors, *ISAAC*, volume 2518 of *Lecture Notes in Computer Science*, pages 523–534. Springer, 2002. ISBN 3-540-00142-5.

[CSTW09] Wei Chen, Christian Sommer, Shang-Hua Teng, and Yajun Wang. A compact routing scheme and approximate distance oracle for power-law graphs. *Distributed Computing*, ?(Disc 2009):379–391, 2009. URL `http://research.microsoft.com/pubs/81716/msr-tr-2009-84.pdf`.

[Del08] Daniel Delling. Time-dependent sharc-routing. In Dan Halperin and Kurt Mehlhorn, editors, *ESA*, volume 5193 of *Lecture Notes in Computer Science*, pages 332–343. Springer, 2008. ISBN 978-3-540-87743-1.

[DPW09] Daniel Delling, Thomas Pajor, and Dorothea Wagner. Engineering time-expanded graphs for faster timetable information. In Ravindra Ahuja, Rolf Möhring, and Christos Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 182–206. Springer Berlin / Heidelberg, 2009. ISBN 978-3-642-05464-8.

[DSSW09] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *ALGORITHMICS OF LARGE AND COMPLEX NETWORKS. LECTURE NOTES IN COMPUTER SCIENCE.* Springer, 2009.

[Fre08] Chaya Fredman. *Ramsey Partitions Based Approximate Distance Oracles.* PhD thesis, The Open University Of Israel, 2008.

[GKW06] Andrew V. Goldberg, Haim Kaplan, and Renato F. Werneck. Reach for a*: Efficient point-to-point shortest path algorithms, 2006.

[GPPR04] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85 – 112, 2004. ISSN 0196-6774. URL `http://www.sciencedirect.com/science/article/pii/S0196677404000884`.

[GSSD08] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In Catherine C. McGeoch, editor, *WEA*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2008. ISBN 978-3-540-68548-7.

[MHSWZ07] Matthias Müller-Hannemann, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, chapter Timetable Information: Models and Algorithms, pages 67 – 90. Springer, 2007.

[MS10] Shay Mozes and Christian Sommer. Exact shortest path queries for planar graphs using linear space. *CoRR*, abs/1011.5549, 2010.

[PSWZ04] Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis. Experimental comparison of shortest path approaches for timetable information. In Lars Arge, Giuseppe F. Italiano, and Robert Sedgewick, editors, *ALENEX/ANALC*, pages 88–99. SIAM, 2004. ISBN 0-89871-564-4.

[San08] Peter Sanders. Time dependent contraction hierarchies – basic algorithmic ideas. *CoRR*, abs/0804.3947, 2008.

[Sch08] Dominik Schultes. *Route Planning in Road Networks.* VDM Verlag, Saarbr&#252;cken, Germany, Germany, 2008. ISBN 383648398X, 9783836483988.

[SGNP10] Atish Das Sarma, Sreenivas Gollapudi, Marc Najork, and Rina Panigrahy. A sketch-based distance oracle for web-scale graphs. In Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu, editors, *WSDM*, pages 401–410. ACM, 2010. ISBN 978-1-60558-889-6.

[Som10] Christian Sommer. *Approximate Shortest Path and Distance Queries in Networks.* PhD thesis, Graduate School of Information Science and Technology, The University of Tokyo, 2010.

[SS05] Peter Sanders and Dominik Schultes. Highway hierarchies hasten exact shortest path queries. In Gerth Stølting Brodal and Stefano Leonardi, editors, *ESA*, volume 3669 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2005. ISBN 3-540-29118-0.

[SS09] Jagan Sankaranarayanan and Hanan Samet. Distance oracles for spatial networks. In *ICDE*, pages 652–663. IEEE, 2009. ISBN 978-0-7695-3545-6.

[SVY09] Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *FOCS*, pages 703–712. IEEE Computer Society, 2009. ISBN 978-0-7695-3850-1.

[SWZ02] Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. Using multi-level graphs for timetable information in railway systems. In David Mount and Clifford Stein, editors, *Algorithm Engineering and Experiments*, volume 2409 of *Lecture Notes in Computer Science*, pages 43–59. Springer Berlin / Heidelberg, 2002. ISBN 978-3-540-43977-6.

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

[WC03] Xiao F. Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *Circuits and Systems Magazine, IEEE*, 3(1):6–20, 2003. ISSN 1531-636X. URL `http://dx.doi.org/10.1109/MCAS.2003.1228503`.