Approximate Shortest Path and Distance Queries in Networks

Christian Sommer
Born on 8 March 1983 in Zurich, Switzerland
Citizen of Winterthur (ZH) and Sumiswald (BE), Switzerland
Master of Science ETH Zurich (2006)

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

January 2010

Department of Computer Science
Graduate School of Information Science and Technology
The University of Tokyo

Abstract

Computing shortest paths in graphs is one of the most fundamental and well-studied problems in combinatorial optimization. Numerous real-world applications have stimulated research investigations for more than 50 years. Finding routes in road and public transportation networks is a classical application motivating the study of the shortest path problem. Shortest paths are also sought by routing schemes for computer networks: the transmission time of messages is less when they are sent through a short sequence of routers. The problem is also relevant for social networks: one may more likely obtain a favor from a stranger by establishing contact through personal connections.

This thesis investigates the problem of efficiently computing exact and approximate shortest paths in graphs, with the main focus being on shortest path query processing. Strategies for computing answers to shortest path queries may involve the use of pre-computed data structures (often called *distance oracles*) in order to improve the query time. 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 precomputation time, storage, query time, and approximation quality?

For distance oracles applicable to general graphs, the quantitative tradeoff between the storage requirement and the approximation quality 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. The first main contribution of this thesis is a proof that such data structures cannot exist for all sparse graphs. We prove a space lower bound implying that distance oracles with good precision and very low query costs require large amounts of space. A second contribution consists of space- and time-efficient data structures for a large family of complex networks. We prove that exploiting well-connected nodes yields efficient distance oracles for scale-free graphs. A third contribution is a practical method to compute approximate shortest paths. By means of random sampling and graph Voronoi duals, our method successfully accommodates both highly structured graphs stemming from transportation networks and less structured graphs stemming from complex networks such as social networks.

Acknowledgements

I thank my advisor and patron Shinichi Honiden for the wonderful Japanese environment he provided, in which I learnt the essentials of Japanese student life and culture, for the freedom he gave me, and for his very generous financial support, which allowed me to travel to many conferences all around the world, and which also allowed me to work with high-quality equipment.

I thank my co-advisor and mentor Michael E. Houle for the many fruitful and interesting discussions about algorithms and everything else, for announcing the shortest path project, for his advice, guidance, and intelligent questions, for his efforts as a collaborator, for teaching me how to improve my writing, and, honestly, for him patiently insisting on important things I did not want to hear and believe.

I am very honored to have Hiroshi Imai as chair and David M. Avis, Takeo Igarashi, Kunihiko Sadakane, and Tetsuo Shibuya as members of the committee for my PhD thesis. Many thanks for investing their time in studying my work and for examining this thesis.

I profited and learned enormously from my co-authors I worked with on the results of this thesis. I am indebted to Wei Chen, who first told me about the connection between path queries and compact routing, Shang-Hua Teng, for empowering and inspiring me, Elad Verbin, for his fascinating questions and his great intuition (and the invitation to Tsinghua), Yajun Wang, for insisting and working hard on the subtle differences between random graph models, Martin Wolff, for his precious help on bootstrapping my thesis and for his impressive skills as a non-native editor, and Wei Yu, for working on all the tedious and nasty calculations.

I also profited from collaborating with Cyrille Artho, Hristo Djidjev, Stephan Eidenbenz, Daisuke Fukuchi, Nicolas W. Hengartner, Pierre-Loïc Garoche, Shiva Kasiviswanathan, Ken-ichi Kawarabayashi, Yusuke Kobayashi, Martin Mevissen, Johan Nyström, Yoshio Okamoto, David Roberts, Sunil Thulasidasan, and Takeaki Uno. I got precious advice from Ittai Abraham, Erik D. Demaine, Jittat Fakcharoenphol, Cyril Gavoille, Stephan Langermann, Xiang-Yang Li, Mikkel Thorup, and Uri Zwick.

This thesis has improved substantially by the valuable comments of those who read parts of preliminary versions of it. I would like to thank Michael E. Houle, Cyrille Artho, Jacopo Grazzini, and Johan Nyström for their helpful suggestions and proofreading. The remaining errors and omissions are entirely the authors responsibility.

I owe a big thanks to countless individuals who I was fortunate to meet and to spend time with in one way or the other — *o-sewa ni nari mashita!* in the Honiden laboratories: Miki Nakagawa (and Family Nishida in Takatsuki), Kyoko Oda, Akiko Shimazu, Shuko Yamada, Ai Tobimatsu, Mizuki Inoue, Kenji Taguchi, Yasuyuki Tahara, Nobukazu Yoshioka, Fuyuki Ishikawa, Kenji Tei, Rihoko Inoue, Nik Nailah Binti Abdullah, Rey Abe, Hikari Aikawa, Yukino Baba, Valentina Baljak, Takuo Doi , Katsushige Hino, Satoshi Kataoka, Yojiro Kawamata, Kazutaka Matsuzaki, Hirotaka Moriguchi, Mohammad Reza Motallebi, Hiroyuki Nakagawa, Yoshiyuki Nakamura, Hikotoshi Nakazato, Eric Platon, José Ghislain Quenum, Yuichi Sei, Ryota Seike, Shunichiro Suenaga, Ryuichi Takahashi, Ryu Tatsumi, Susumu Toriumi, Eric Tschetter, Kayoko Yamamoto, Adrian Klein, Maxim Makatchev, Daniele Quercia, and Martin Rehak; at NII: Shigeko Tokuda, Michael Nett, Weihuan Shu, Nizar Grira, Sebastien Louis, Sébastien Duval, Takeshi Ozawa, Yuzuru Sawato and all the other nice and friendly guards; at Microsoft Research Asia and in Beijing: Kun Chen, Yuki Arase, Yasuyuki Matsushita, 'Tommy', Yuan Zhou, Jialin Zhang, Lolan Song, Peter & Ursi Zürcher, Gabriel Schweizer, and Jaimie Hwang; at the Los Alamos National Laboratories and in Los Alamos: Ulrike Campbell & Glenn, Family Eidenbenz, Douglas D. Kautz, Jacopo Grazzini, Nicolas Jegou, Leticia Cuellar, Vishwanath Venkatesan, Guanhua Yan, Keren Tan, Lukas Kroc, Leonid Gurvits, Nandakishore Santhi, and Carrie Manore; on my trips to the US and around the world: Alicia Aponte, Jittat Fakcharoenphol, Daniele Quercia, Thomas, Kati & Lars Spirig, and Andrew Yao; a special thanks to my running mates from Tokyo's International Running Club "Namban Rengo" for many enjoyable hours of training, racing, and travelling (and partying): thanks Guillaume Bouvet, Gary & Mami Chandler, Kazuo Chiba, Chad Clark, Edward Clease, Joachim Dirks, Jon Holmes, Jay Johannesen, Chika Kanai, Gordon Kanki Knight, Daniel Kershner, Stephen Lacey, Christiane Lange, Brett Larner, Jason Lawrence, Omar Minami, Teruyuki Minegishi, Keren Miers, Satohi Numasawa, Paddy O'Connor, Rie Onodera, Bob Poulson, Gareth Pughe, Fabrizio Raponi, Gerard Robb, David Motozo Rubenstein, Philip Ryan, Daniel Seite, Yuka Shigihara, Mika Tokairin, Martin Verdier, Jürgen Wittstock, Kiyonari Yoshida, and others; from the Swiss club in Japan: Jörg Aschwanden, Daniel & Monika Hagemeier, Felix Mösner, Hans Prisi, Christoph Saxer, Uwe Sievers, Martin Wenk, and Hermann Werner; Japan friends (tea ceremony & BBQ) Mitsuhiko & Yuko Kusuyama, Chikako, Tomoaki, & Noriaki Sawada, Machiko & Yoshinobu Nagura, Junichi 'Andy' Kimura, Hitomi Sakamoto, Tomoko Sawada, Toshiro 'Joe' Suzuki, Akemi Okado, Masae Ono, John & Yuki Mettraux; more friends in Japan: Ryudo Tsukizaki, Koichi Matsumoto, Filiz Gencer, Jumi Klaus, Cedric S. Rutishauser, Jerry Ray, my extremely patient and lovely Japanese teacher Chieko Okamoto, Katsuro Ishimasa of Idaten, and Ken Yamagata of Iroha for making the best eel sushi in town.

It was great to have friends visiting me in Japan; thanks to Samuel Burri, Adrian Doswald, Martin Halter, Roger Herzog, Julia Imhof, Tanja Isker, Moritz Isler, Hannes Schneebeli, Georg Troxler, and Stefan Wolf.

Last but definitely not least, I wish to thank my wonderful family. None of this would ever have been possible without the love and the fantastic support of Hansruedi, Hermine, and Stefan. They taught me that shortest paths are not always optimal and desirable, and that *"der Weg ist das Ziel"*. And I wish to thank my Japanese family in Miyazaki, Kayashima, and Tottori, especially my beloved Rie, for everything.

Christian Sommer, December 2009

Table of Contents

MILLE VIAE DUCUNT HOMINEM
PER SAECULA ROMAM.
(All roads lead to Rome.)
(Es führen viele Wege nach Rom.)

Alanus de Insulis
in the Liber Parabolarum
12th century AD

1 Introduction

Imagine you wanted to travel from central Tokyo to visit the place where I grew up, which is the village of Ottenbach in Switzerland. What is the fastest way to get there?

Imagine you wanted to get in touch with Nelson Mandela using a sequence of personal introductions through friends and friends of friends. What is the shortest such sequence?

Imagine you wanted to access a webpage on the Internet. Which routers should be used such that the necessary information is downloaded to your computer fastest?

These questions have something in common, in that their (optimal) solution is the shortest path between two points of a network: a transportation network, a social network, and a router network. All roads may lead to Rome, but we wish to arrive as soon as possible. The aim of this thesis is to provide means to efficiently compute shortest paths in networks.

1.1 Networks and Graphs

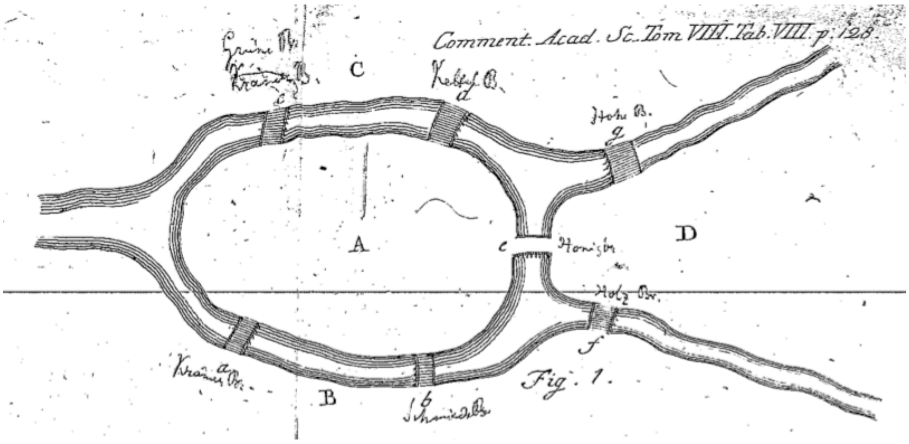

Figure 1.1: *The Königsberg bridges as depicted by Leonhard Euler in his article "Solutio Problematis ad Geometriam Situs Pertinentis" on page 129 in volume 8 of* Commentarii Academiae Scientiarum Petropolitanae *in 1741.*

You may have heard of the seven Königsberg bridges and the question as to whether one can, in one walk, cross each bridge exactly once. Leonhard Euler resolved this question in 1735 by proving that there is no such walk. His proof works as follows. Consider the island denoted by the letter $\mathcal{A}$ on the illustration in Figure 1.1. Euler made the following important observation: although the island contains many buildings, streets, and paths, the relevant information is the island's *connections* (the bridges) to the other parts of the city. This observation led him to create an abstract discrete structure, later termed a *graph*. He identified each landmass with a *node* and each bridge with an *edge* connecting the two corresponding nodes in the graph. A walk in Königsberg corresponds to a walk in the graph; crossing a bridge is represented by traversing an edge. Euler then noted that the number of edges adjacent to a node is essential. Except for the endpoints of the walk, all intermediate nodes must have an even number of adjacent edges, since any walk must leave the node exactly once for every time entering it. Since all of the four nodes have an odd number of edges, there cannot be any walk that traverses each edge exactly once.

Modeling the bridges by the edges of a graph helped Euler to solve the problem of the Königsberg bridges. Ever since, graphs have been used as an abstraction of structures in the real world:

> Graphs are, of course, one of the prime objects of study in Discrete Mathematics. However, graphs are among the most ubiquitous models of both natural and human-made structures. In the natural and social sciences they model relations among species, societies, companies, etc. In computer science, they represent networks of communication, data organization, computational devices as well as the flow of computation, and more. In mathematics, Cayley graphs are useful in Group Theory. Graphs carry a natural metric and are therefore useful in Geometry, and though they are "just" one-dimensional complexes, they are useful in certain parts of Topology, e.g. Knot Theory. In statistical physics, graphs can represent local connections between interacting parts of a system, as well as the dynamics of a physical process on such systems. [HLW06]

In a graph representing a road network, intersections and streets can be modeled by nodes and edges, respectively. Two nodes have an edge in between if there is a street connecting the two corresponding intersections. For a computer network, routers and the connecting network cables are mapped to nodes and edges, respectively. In a social network, the connections are not physical. Individuals can be modeled by nodes; two nodes are connected by an edge whenever the corresponding individuals are friends. In other social networks, an edge may also indicate a private or professional relationship other than friendship.

Different streets of a road network have different lengths. This is modeled by assigning each edge a number, called its *edge weight*. This cost can reflect real-world values such as distance, travel time, transmission time, and latency. In a graph modeling a social network, the edge weight can also reflect the quality of a friendship, although this is arguably difficult to capture appropriately by a single number [XNR10].

In the remainder of this introduction, we first review various example structures, for which a graph serves as a suitable model. We then consider one central problem that can be solved using graphs: we investigate applications of the *shortest path problem* (Section 1.2), in particular contemporary applications of the *point-to-point shortest path problem* (Section 1.2.2). We conclude the chapter by stating the contribution of this thesis (Section 1.3).

1.1.1 Transportation Networks

Transportation networks are an integral part of the infrastructure in many countries. For this reason, the study of transportation networks is an important field of research. We give three examples of transportation networks: road, railway, and airline networks. A more realistic model of transportation would ideally integrate networks of all three types [Fra08, DPW09, DPWZ09].

Road Networks

As mentioned before, intersections and streets are represented by graph nodes and edges, respectively. Also, edges may be assigned weights such as expected travel times or distances. All the information included in the graph model can essentially be found by consulting a road map. The reverse is not true. We cannot draw the original road map by considering the graph only, since some information about the original drawing (or embedding) such as coordinates is not included. The original map is just one possible drawing of the graph. A general graph is completely independent from its embedding.

A graph that can be drawn on a plane such that no two edges cross (except for the endpoints), is called *planar*. Since road networks often contain many bridges and tunnels, the corresponding graphs are in general not planar. However, road networks and planar graphs share some important properties, which render road networks tractable for many optimization problems. Navigation, for example, is not as difficult as it could be on a general graph, since road networks have some geometric and geographical orientation. Another characteristic of road networks is that, at every intersection, the number of streets (and thus the number of choices when navigating) is quite low. Still, planning efficient routes through a road network is very challenging since these networks may be huge (millions of nodes) and dynamic (travel times depend on various factors such as the current traffic situation and road maintenance).

Railway Networks

For railway networks, a straightforward model, analogous to road networks, would map stations and tracks to the nodes and edges of a graph, respectively. Taking a train is represented by traversing an edge. Again, edges can be weighted with distances or travel times. This model arguably does not represent a real public transportation system very well. In road networks, one can conceivably use a particular street at any time (except for small delays due to traffic lights). In railway networks, however, trains are bound to a timetable. This may cause significant waiting time at a station, which needs to be captured by a realistic model. Indeed, modeling railway networks with timetable information is more involved than modeling road networks [MHSWZ04]. In the time-expanded model for example, nodes represent both a location and a specific time simultaneously. One station then corresponds to several nodes in the graph. Edges have different types; traversing an edge means either taking a train, waiting at a station, or walking to a different track within a station. Edge weights can be travel times (possibly walking or waiting times) or ticket prices. In general, the graphs derived from railway networks and timetables are considerably more complex than the graphs representing road networks.

Airline Networks

Air traffic networks can also be represented by graphs. Airports can be modeled by the nodes of a graph. Two nodes are connected by an edge if there is an aircraft that can start and land on and cover the distance between the corresponding airports. In this graph, almost all nodes are connected. While in road networks, each intersection had up to half a dozen of connections, in air traffic networks, airports can have hundreds of connections. Consequently, the corresponding graphs are significantly more dense. For the moment, let us restrict the edges of the graph to the routes that commercial airlines offer. Even when considering the graph with this restricted edge set, some nodes have hundreds of adjacent edges, due to the fact that airlines often have a small number of hubs (usually big airports) where all their routes connect. The airline network is a so-called *small-world* network [WS98]: one can connect any two airports in the network by following a sequence of only one to five connections [ASBS00]. From a graph-theoretic point of view, airline networks are structurally very different from road and railway networks. Networks without apparent structure are often called complex networks.

1.1.2 Complex Networks

Various complex systems are highly interconnected: phenomena that were assumed to be local only are sometimes unexpectedly shown to influence the other end of the system. Researchers from fields such as physics [BA99, New00, WS98], mathematics [BRST01, BR04, FFV07, Lov09], computer science [KL06, WM03], biology [MV06, LT09], and social science [LPA+09] analyze these systems to explain *how* (and sometimes *why*) everything is connected [Str01, AB02, New03, GGM06, BLM+06]. The objectives of network scientists are (1) to make connections in real-world systems explicit ("to connect the dots" [Hay06]), (2) to analyze and understand the network structures formed by these connections, and, possibly, (3) to exploit the structural properties of these systems. The challenges are manifold. Extracting the connections of complex systems can already be very difficult since these systems tend to be inherently distributed and very large. Once the connections have been made explicit, that is, once a graph (in this context also termed *complex network*) has been created, the next challenge is awaiting. Due to the often massive size of the graph (millions of nodes and edges are not uncommon), analyzing its structure requires sophisticated methods, techniques, and tools. Researchers often use methods to decompose the graph into different clusters and communities of smaller size, which are easier to understand than the whole graph. Another approach is to focus on 'important' nodes within the graph. Important nodes are conjectured to be those that are well connected with most of the other nodes, or those that enable important connections between other nodes.

Although these systems and the corresponding complex networks appear to lack structure, they still have some important commonalities: they show a large variation in the number of links connected to each node; these networks often have a few nodes that are very well connected while a majority of the nodes have only few interactions; the number of interactions appears to follow a *power law* [Mit03, New05, CSN07]. These networks are called *scale-free* networks [dSP65, BA99, BAJ00, New00, WS98]. Also, many complex networks are *small worlds*, which means that, although many nodes only have a few direct interactions, they are still connected to all the other nodes through very short chains of interactions.

In the following, we discuss some specific complex networks in more detail.

Citations among scientific articles

The graph modeling citations among scientific articles is a directed graph, which means that its edges are oriented. The node corresponding to article A has a directed edge to the node corresponding to article B if and only if article A cites article B. The citation of a research article by another one is an explicit relationship indicating influence and dependence. An article that gets cited by many other articles is likely to have had some influence on many researchers. For this reason, the impact of an article is often measured by the number of citations an article received. Researchers and managers evaluate the success of a research article based on its impact. There are also potential applications other than evaluating and ranking articles and researchers. For example, the citation graph may be of use in a system that automatically suggests reviewers for articles submitted to a journal.

The network of citations among scientific articles was in fact one of the first complex networks ever analyzed [dSP65]. It has been found that, over time, articles that are cited by many others, acquire even more citations [dSP76]. This cumulative advantage, (later termed *preferential attachment* [BA99]), appears to be very common in complex networks.

Web Graph

The web consists of billions of pages that are connected by hyperlinks. For the web structure, a directed graph is a suitable model [KKR+99, BKM+00, DLL+06]. Web pages and hyperlinks are mapped to nodes and edges, respectively. The resulting structure is also called *web graph*. Similar to the citation between two scientific papers, the hyperlink between two web pages indicates some dependence and influence. A scientific article is considered important if it is cited by many others; analogously, a webpage is considered important if it is linked to by many others. Modern search engines often make use of this network structure when ranking search results [PBMW99, FVC07].

An accurate snapshot of the web graph is hard to obtain: the number of web pages is massive, the pages reside on different servers all around the world, and many pages change constantly. Due to these reasons, researchers analyze subsets and samples of the web. Based on such small samples, it is conjectured that the web graph is *scale-free* [BAJ00], which means that the number of incoming links per web page obeys a power law. It is also conjectured that most pages that are inter-linked, are connected by rather short chains of links [AJB99]. Based on the characteristics of sample web graphs, researchers also build mathematical models [KRR+00, BBCR03, FFV07, CKL+09], using which the future structure of the web graph can be predicted.

Weblogs ("blogs") form a very popular part of the web [Shi03]. Since good content on the web is proportionally rare and since most blog writers (termed "bloggers") also read other blogs, interesting ideas, rumors, and stories propagate very quickly within the blogosphere. Some blogs are read by millions of readers and the corresponding writers have the power to influence their audience to a certain extent. It is thus no surprise that companies try to make use of this "network of influence" when launching new products or services. The analysis of the blogosphere is indeed a vibrant research topic [LMF+07, LNMG09].

Router Connections

Since the Internet has become an integral network handling a significant percentage of business transactions both among companies and between companies and individuals, any technical failure might have drastic consequences in the real world. The Internet is nowadays considered being critical infrastructure almost at the level of transportation and power networks.

Although the Internet is a physical network consisting of routers and cables (modeled respectively by nodes and edges of a graph), there is no accurate map available. Researchers try to create such a map by sending messages to random computers while monitoring (using the tool `traceroute`) which routers the messages went through [FFF99, VPSV02, Coo03]. These measurements yield that the Internet (or some of its subgraphs) also has properties of power-law graphs. However, it is not clear whether these properties were obtained due to measurement bias in the map creation process [ACKM09].

The "speed of the internet" highly depends on the efficiency of the internet protocols, in particular the routing protocols. The performance of a protocol highly depends on the underlying network structure. Since speed matters and due to the importance of the Internet as critical infrastructure, the network of router connections is a very important network to understand and analyze.

Networks in Biology

Traditionally, proteins are identified based on their actions as building blocks, catalysts, or signaling molecules. The network view identifies a protein by its physical interactions with other proteins. This yields its contextual role in a protein–protein interaction network [UGC+00, CHC+00, ICO+01, RSR+01, GBB+03, LAB+04]. Proteins and interactions are represented by nodes and edges of a graph, respectively.

Recently, many networks have been extracted from biological data [MV06]. Examples other than protein interaction networks include metabolic networks, which encode biochemical reactions between metabolic substrates [RSM+02], and transcriptional regulatory networks, which describe the regulatory interactions between different genes [IFB+02, SLSP03].

Network science may help to understand the human body and its internal actions, for example by determining the role, function, and essentiality of proteins or genes [SL03, JOB03] by analyzing shared interaction partners in the protein–protein interaction network. This knowledge may then help to find treatments for diseases such as cancer by identifying drug targets [VLL00]. Another potential application in medicine lies in the intersection with neuroscience. Abnormal interaction patterns in the brain could help diagnosing neurological disorders [SG05]. These applications indicate that the potential of an interplay between biology and network science is enormous.

Social Networks

Around 1929, the Hungarian writer Frigyes Karinthy [Kar29] formulated the following challenge: find a person who you can not be connected to by a friendship chain of at most five people. In other words, Karinthy conjectured that everybody knows everybody else through a short chain of personal connections. Milgram [Mil67], in his famous small-world experiment [Mil67, TM69], tried to verify this conjecture with an empirical study. Participants in the Milgram study were supposed to deliver a letter addressed to a specific person, not by mailing it directly, but by forwarding it to somebody they know on a first-name basis — somebody who is more likely to know the addressee. Milgram was interested in the number of forwarding steps, which corresponds directly to the length of a friendship chain. Although most letters never arrived, many letters reached the addressee after a very small number of forwarding steps only.[1] *"It's a small world!"* Milgram's result exhibits a stronger statement than Karinthy's conjecture. For two individuals, the friendship chains connecting the two are not only short, it is also possible to "find" such a chain.

The small-world phenomenon has been fascinating people around the world for many years. For example, the popular Hollywood movie "Six degrees of separation" [Gua93] is devoted entirely to small-world phenomena. As another example, the members of some communities "measure" their interaction distance to distinguished individuals. Actors are interested in their co-starring distance to Kevin Bacon, Mathematicians care about their collaboration distance to Paul Erdős,[2] and players of the popular game *Go* measure their distance to Honinbo Shusaku.[3]

In social networks [dSPK78], we consider individuals as nodes and relationships as edges of a graph. If the relationship is friendship, social networks are far more than a scientific playground: online social networking platforms such as Facebook, MySpace, mixi, and others have

[1] Some researchers doubt the significance of Milgram's results [Kle02b, Kle02c].

[2] My own Erdős number is currently 3. Sommer–Houle–Avis–Erdős and Sommer–Teng–F. Yao–Erdős are two disjoint shortest paths containing 3 publications each.

[3] See http://senseis.xmp.net/?ShusakuNumber — Erdős apparently has a Shusaku number of at most 6.

conquered the web and their corresponding websites attract millions of users. Privacy aside, these social networking websites generate massive data sets that are of huge interest for advertising companies and marketeers. Some of this data is made available to scientists as well (usually after anonymization). Companies also analyze their internal communication patterns to improve productivity and to identify leader personalities [GBL08]. Social networks can also be extracted from phone call [OSH+07] and instant messaging [LH08] data. Other data on social relationships may be harder to collect. If an edge of the graph indicates sexual interaction (not necessarily a subset of the friendship edges), not everybody may be willing to reveal all connections. In a sample of 2,810 Swedes, the number of sexual interactions per person showed the structure of a power law [LEA+01]. Although the exact connections were not retrieved, the power-law distribution in the number of connections is already consequential. Epidemics tend to arise and propagate very fast in power-law networks [WS98, PSV01]. It is also known that the speed of disease propagation can potentially be reduced by prevention campaigns that strategically target those individuals with a large number of partners [LEA+01]. The previous example indicates that knowledge and understanding about the structure of complex networks may have an impact on the real world.

Network of networks and techno-social systems

The spread of a general infectious disease depends on interactions of different types. One network is not enough; several networks have to be considered simultaneously to find effective containment strategies in urban social networks [New02, MPSV02, EGK+04, MPN+05, GCE+09]. Techno-social systems consist of a technological component, often physical infrastructure such as transportation systems, and a human component, which could be any form of communication, potentially influenced by a social network. Understanding the interplay of both components is necessary to predict, and eventually control and influence, techno-social systems.

> It now seems possible to imagine the creation of computational forecasting infrastructures that will help us design better energy-distribution systems, plan for traffic-free cities, anticipate the demands of Internet connectivity, or manage the deployment of resources during health emergencies. [Ves09]

1.2 Shortest Paths

Once we have performed the abstraction by making connections explicit and modeling them by a graph, we can work with this model and compute properties of the graph without considering the exact details of the underlying network. Interesting properties could be graph *distances* between nodes. Tobler [Tob70] invoked

> ... the first law of geography: everything is related to everything else, but near things are more related than distant things.

This law ought to be generalized to a law holding for many networks. An edge of a graph essentially indicates a relationship between the two corresponding entities: between two interacting proteins, between two friends, between two intersections in a road network, or between two routers connected by a cable. If this relationship is somewhat transitive,[4] which means that, for example,

[4] In some graphs, two nodes are connected by an edge if there is some conflict between the corresponding edges. For example, two neighboring wireless devices cannot send at the same time due to interferences. These conflict graphs often occur in scheduling problems, which can be solved for example by coloring the graph. In this thesis, however,

if Paul is friends with Linton and Stanley, then Linton and Stanley are probably closer to each other than if they were not friends with Paul. Or, to consider another example, if protein A interacts with and influences protein B, and B influences C, then A has an indirect influence on C. The shorter the chain of interactions, the stronger the influence. This is the **first law of networks**.

In graphs, the distance between two nodes s and t (source and target, respectively) is defined as follows. If s and t are connected by an edge, their distance is 1. If they are not directly connected, the distance is defined by the length of a *shortest path* between s and t, which is a sequence of adjacent edges. In a *weighted* graph, the length of a path is defined by the sum of the weights of the edges on the path. Consequently, shortest paths are defined with respect to these weights. Note that, in weighted graphs, even if two nodes are connected by an edge, depending on its weight, the edge is not necessarily part of any shortest path.

To each of the three questions at the beginning of this chapter the answer is a shortest path in the corresponding network: the fastest way to get to Ottenbach, the shortest sequence of friends to get in touch with Nelson Mandela, and the fastest route to a webserver. Many other questions also have shortest paths in graphs as answers. Whenever we must traverse a network or send someone or something through a network between two points in a fast, cheap, or reliable way, it is likely that solving a shortest path problem will provide the optimal solution. Other applications of shortest path computations include (but are by no means limited to) network optimization [GN67, BMW89], packet routing [SS80], image segmentation [CK96, MB98, FUS$^+$98], computer-assisted surgery [MDMCM01], computer games [Sto99], DNA analysis [Sch98], injection molding [Joh97], postman problems [EJ73], operator scheduling [BOR80], production planning [Kle63, GN64, Whi69, Elm77, LN94], re-allocation of resources [Wri75], approximation of piecewise linear functions [II86, p. 33], and VLSI physical design [CKL97]. Furthermore, the countless optimization problems that can be solved by deterministic dynamic programming [CLRS01, Chapter 15] (Knapsack, for example [Fri76]), can also be solved by a shortest path algorithm (on a very large graph) by identifying the stages of the dynamic program with the nodes of an acyclic, directed network. The shortest path problem also has a deep connection to the minimum cost flow problem, which is an abstraction for various shipping and distribution problems, the minimum weight perfect matching, and the minimum mean-cycle problem. More sophisticated algorithms for combinatorial optimization problems such as network flows often need a shortest path subroutine.

1.2.1 Classical Results

Methods to find a shortest path were discovered and analyzed already in the late 50's and early 60's by Bellman [Bel58], Bock and Cameron [BC58], Caldwell [Cal61], Dantzig [Dan57, Dan60], Dijkstra [Dij59], Floyd [Flo62], Ford [For56], Fulkerson [FF58], Hu [Hu67], Klee [Kle64], Leyzorek, Gray, Johnson, Ladew, Meaker, Petry, and Seitz [LGJ$^+$57], Minty [Min57, Min58], Moore [Moo59], Mori and Nishimura [MN67], Parikh [Par60], Rapaport and Abramson [RA59], Shimbel [Shi53, Shi55], Warshall [War62], Whiting and Hillier [WH60], and probably others.[5] Despite the

we restrict ourselves to graphs of the type encountered in the previous examples, assuming that an edge represents a somewhat positive relationship and not a conflict.

[5] Due to the large amount of literature, providing an exhaustive list appears to be difficult. The first survey article reviewing shortest path algorithms was apparently written and published in 1960 [PW60] and complemented in the same year with four additional solutions [PRB60]. Neither article mentions Dijkstra's algorithm.
Robacker [Rob56] and Gallai [Gal58] (and potentially Egerváry [Ege31]) studied the shortest path problem without giving an algorithm. Research on the strongly related (but harder) Traveling Salesman Problem started earlier, see for example [Hel53, Kuh55, DFJ54].

exponential growth of computing research, many of their algorithms are still in use today in one form or the other, most prominently Dijkstra's algorithm, which literally every computer science student learns as an undergraduate. Dijkstra's algorithm solves the Single Source Shortest Path (SSSP) problem: the algorithm starts at one point (the source) and explores the whole graph in all directions until the distances to all the other nodes are known. For an illustration of the shortest path tree in a transportation network, see Figure 1.2. If the user is interested in the distance to one target node only, it is possible to stop the search as soon as this specific target has been found. Some of the other classical algorithms solve the All Pairs Shortest Path (APSP) problem: the distance and the shortest path between all pairs of nodes is computed.

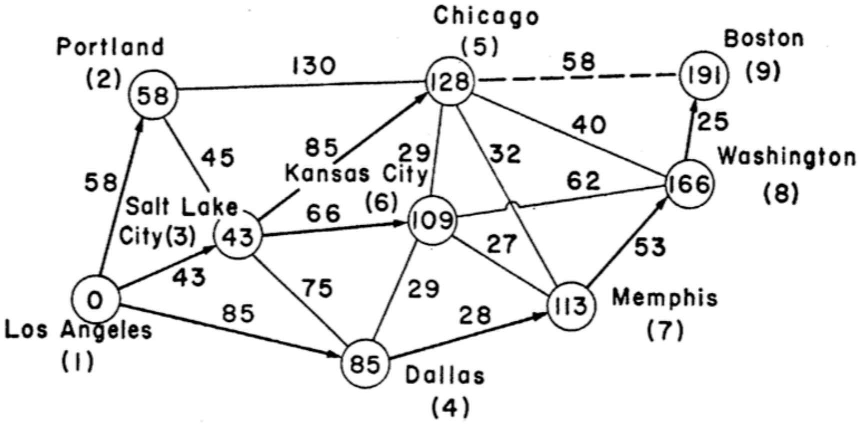

***Figure 1.2**: The shortest path tree starting at the node representing the city of Los Angeles. Original by George B. Dantzig [Dan57].* "Hence the optimal path is from Los Angeles to Salt Lake City, then to Chicago, and finally to Boston."

The importance of the problem and the numerous applications have stimulated research efforts for more than fifty years.[6] A large fraction of these efforts targets the shortest path problem in transportation networks, since route planning is arguably one of the most important applications of shortest path algorithms. Intelligent navigation systems know the current location using GPS and guide cars along the shortest route to minimize travel time, distance, or fuel and energy consumption. Some of these systems also react to changing traffic conditions and, in the future, cars may communicate with each other and negotiate routes to regulate the overall traffic flow and manage congestion. The end users are often interested in trip planning, which means that they want to know the distance and the shortest, cheapest, or most reliable path between two specific points.

1.2.2 Point-to-Point Shortest Path Queries

We ask for the distance between two points of a network. Recall that Dijkstra's algorithm solves the Single Source Shortest Path (SSSP) problem by exploring the whole graph starting from the source until the distance to all other nodes is known. Dijkstra's algorithm can be stopped prematurely, as soon as the target has been found. Another improvement to decrease the running

[6] A search for "shortest path" on Google scholar `scholar.google.com`, a search engine for scholarly literature, returns more than 150,000 results. Hu [Hu71] states: "This is an area in which people keep writing papers."

time is to start a search from both the source and the target and stop when the two searches meet [Nic66, Mur67b, Poh71, SdC77, dC83]. Still, Dijkstra's algorithm may explore the whole graph. A complete exploration is too expensive. The objective is to develop a method that is much faster than Dijkstra's algorithm. In return, the method is allowed to *pre-compute* a data structure that, later, assists shortest path computations, called *queries*. We thus assume that the graph is known some time before the first query is asked.

Shortest path query processing resembles a typical data base problem: create an index by materializing information to speed up certain queries [Han87, SFG97]. One strategy would be to precompute the result for every possible query. This is expensive in terms of time and space; storing all the results may be prohibitive since the number of possible queries is quadratic in the number of nodes. No precomputation, the other extreme, is too slow at query time. We aim for a mid-point, a good compromise, between "no precomputation and complete processing" at the time of query and "complete precomputation and a simple look-up" at the time of query [AJ89].

Shortest path query processing is an integral part of many applications, in particular Geographic Information Systems (GIS) and intelligent transportation systems [JHR96]. A challenge for traffic information systems or public transportation systems is to process a vast number of customer queries on-line while keeping the space requirements as small as possible [Zar08]. Transportation planning systems and GIS form arguably the most established application scenarios for shortest path algorithms. The following sections list — without being exhaustive — other motivational examples of contemporary applications, where computing short paths is a key component and where a method that allows for efficient shortest path queries may potentially speed up the total computation significantly.

Traffic Simulations

Two objectives of traffic simulations [HP58, CWM94, SBA+95, HLW98, PW99, EW03, RN04a, TE09, TKE+09] are to forecast future traffic patterns and to predict the consequences of certain changes to the road network. While sometimes the consequences are easily predictable and intuitively clear, there are paradoxical situations where closing a road actually improves the overall traffic situation [Bra68]. Simulations can help to identify these counterproductive roads. Simulation results may also help in urban planning [SG67, SL67, RTMS05, Bat08] to reason about the economic and social impact of building new roads. With realistic estimates of population mobility and parameterized models for simulating the progress and transmission of a disease, simulations may improve predictions in public health and epidemiology [EGK+04, MPN+05, Ves09].

Network-based simulators assume that cars drive along shortest paths. Within the simulator, at each virtual time step, entities are routed one step further from the source to the destination along a shortest path. The simulation requires the computation of a large number of shortest paths [ZKM97, BBJ+02, Hol04, RN04a, BG07, TE09, TKE+09]. Therefore, various simulators can potentially reduce the total running time by exploiting a fast query processing method. Since in the real world drivers rarely use the exact shortest path, even a method returning approximate shortest paths may be of help.

Image Segmentation

Image segmentation [CK96, MB98, FUS+98] is an integral part of image processing applications such as accident disposal, medical images analysis, and photo editing. The image segmentation problem is to group together neighboring pixels whose properties are coherent. The grouping process often relies on shortest path computations. Object surfaces are in some sense continuous.

Continuous shortest paths are computed with the fast marching method [Tsi95, Set96, HPCD96], which can be seen as a continuous version of Dijkstra's algorithm. However, since the discrete lattice [Bes74] is the standard reconstruction of image data, discrete algorithms are also used in many applications. Graph-based algorithms[7] consider the image as a graph in which each pixel is a node, connected by an edge to each of the 4 (or 8 or more) neighboring pixels. An important part is to assign appropriate weights to the edges, based on the image "potential" of the two pixels and their Euclidean distance [Bor84, TM92].

The graph-based intelligent scissors algorithm [MB98] heavily relies on shortest paths. The user provides two endpoints; the algorithm computes the result of the corresponding point-to-point shortest path query and the resulting path is interpreted as a part of the object boundary. In other words, given two endpoints of a contour, the algorithm determines the maximum likelihood contour connecting them. The edge weights are probabilities and, by taking the negative logarithm, the algorithm can just sum up the weights on the path to retrieve the optimal contour.[8] For boundary computations in 3 dimensions, the intelligent scissors algorithm must be adapted [KH05, Gra06, Gra08]. A boundary can be found by combining many shortest paths until a closed surface is obtained [KH05].

In computer vision [Gra08, PC06], shortest path algorithms on weighted graphs have found numerous applications other than segmentation such as centerline finding [BKS01, BSB^{+}00], radiation therapy [CLW08], mesh morphing [LDSS99, ADG^{+}06], video summarization [PMT03], and finding roads and trails on satellite images [FTW87, HSP92, MZ93, SMR97, MB08].

Drug Target Identification

Shortest paths in interaction graphs are important in systems biology. Signaling paths for example are routes along which one molecule can affect another one [KvK09, Ari00]. The average path length already reveals valuable information about a cell or a body.

> In a sense, the average path length in a network is an indicator of how readily "information" can be transmitted through it. Thus, the small world property observed in biological networks suggests that such networks are efficient in the transfer of biological information: only a small number of intermediate reactions are necessary for any one protein/gene/metabolite to influence the characteristics or behavior of another. [MV06, Section 3, p. 8]

[7] We note that image segmentation based on minimum spanning trees in these graphs is also a suitable approach [Zah71, FH04].

[8] This likelihood transformation can be applied to Markov chains [MT93] in general [Met07, Chapter 6]. We may interpret a Markov chain with state set S, described by its (stochastic) transition matrix $P = (p_{ij})_{i,j \in S}$, as a directed graph $G = (V, E)$. The edge weights can be set with a *likelihood approach*, where the edge $(i,j) \in E$ has weight $\mathbf{w}_{LL}(i,j) := -\lg p_{ij}$ and the path length is the negative log-likelihood. The shortest path between two states with respect to the weights $\mathbf{w}_{LL}(i,j)$ is the path with the highest probability (*reaction pathway*) and the result of a point-to-point distance query yields a lower bound on the transition probability between the corresponding states. For rather long paths, the likelihood approach may produce misleading results. The *Free Energy Approach* tries to overcome this problem. We assume that the stationary distribution is unique, $\pi = (\pi_i)_{i \in S}$. Now, the discrete free energy of a state is $F_i := -\lg \pi_i > 0$. The edge weights are constructed such that the shortest path between two states overcomes the lowest discrete free energy barriers. This yields weights $\mathbf{w}_{FE}(i,j) := |F_j - F_i|$. Assigning likelihood or free energy edge weights may be a suitable model for social networks as well. The probability of favor serves as a possible interpretation. Here, the sum of the weights of edges adjacent to one node need not necessarily to sum up to 1.

The average path length is related to a graph's Wiener index[9] [Wie47, Rou86], which could potentially be approximated using random sampling and (approximate) point-to-point shortest path computations.

The next potential application leverages information on paths other than their average length. We wish to know which proteins, genes, and metabolites are more important and powerful than others, meaning that they have a strong influence on others. These may be suitable targets for medication [VLL00]. Various graph-theoretic centrality measures somehow correlate with the importance of a node. The degree centrality for example counts the number of neighbors of a node. Intuitively, the more interaction partners a protein has, the higher its potential to influence others [Fla63]. More complex measures distinguish purely local effects (such as the number of interaction partners) from global organizational effects [WS03]. For example, if a protein is on many reaction pathways or signaling paths, it has some potential for control [Fre77]. It can stop — or at least slow down — the reaction by breaking the chain of interactions, due to the fact that it lies between two indirectly interacting proteins. The number of shortest paths that a node takes part of is called its *betweenness centrality* [Bav48, Shi53, Sha54, Fre77]. Betweenness accounts for direct and indirect influences of proteins at distant network sites and hence it allows one to relate local network structure to global network topology. *Closeness centrality* [Bav50, Bea64, Sab66] measures how far away a node is from all the other nodes. *Degree centrality* counts the mere number of interaction partners.

An important application of network centrality in pharmaceutical research could be drug target identification. One of these potential target genes is p53 [VLL00]. The loss of p53 function is very damaging: p53 is among the genes most likely to be mutated in cancers. In fact, p53 function loss occurs in nearly all human cancers. It turns out that p53 corresponds to a highly connected node in the interaction graph.

Centrality also allows to predict protein essentiality [JOB03, JBIH05]. Interesting proteins are those with high betweenness-centrality, yet low local connectivity. Their low connectivity would imply that they are unimportant, but their high betweenness suggests that these proteins may have a global impact, acting as important links between modules. The removal of a protein can have different phenotypic effects including lethal or non-lethal effects and a slow-down of growth. There is a positive correlation between lethality and connectivity [JMBO01]. The most highly connected proteins in the cell are the most important ones for its survival. The more essential a gene (or its associated protein) is to a pathogen or to a cancerous cell, the more attractive it is as a drug target [MV06, Section 4]. So far, however, there is not much evidence that for two nodes with the same number of interactions the node with higher betweenness centrality is significantly more important [JBIH05]. Currently, degree centrality serves as a good indicator. Nevertheless:

> In most of the studies, [...] the centrality score of a node was found to be indicative of its likelihood to be essential. In particular, this appears to be true for degree centrality, betweenness centrality, and eigenvector centrality measures. [MV06]

The computation of centrality indices has been studied [Bra01] and since exact sequential methods are rather slow, there are efforts to parallelize the computations [MEJ$^+$09] and to find approximate solutions [EW04, BKMM07]. Since the biological networks are often sampled and have some errors, centrality measures are not exact anyway [CV03a]. Depending on the application, approx-

[9] In chemistry, the Wiener index of a molecule is the sum of all shortest path lengths between non-hydrogen atoms in the graph defined by the structure of the molecule. In general, the Wiener index of a graph is the sum of all shortest path lengths among all nodes.

imations may be sufficient. Centrality algorithms rely on the computation of functional pathways and they could thus benefit from fast (approximate) path computations [RMJ06].

Community Detection

Complex networks are often huge and thus difficult to analyze. One way to obtain an understanding of complex networks is to decompose the network at hand into related components, communities, and clusters. Several algorithms for clustering and community detection have been proposed [GN02, NG04, New04, CNM04, APF+06, Dji06]. The algorithm by Girvan and Newman [GN02, NG04] has been applied successfully to a variety of networks, including several social and collaboration networks, metabolic networks, and gene networks. Their algorithm iteratively removes edges with high betweenness centrality. If a network contains highly-connected communities that are only loosely connected by a few edges between clusters, then all shortest paths between different communities must go along one of these few edges, which will therefore have high edge betweenness. Removing these edges separates the communities from each other. The method works very well for small graphs but it does not scale due to its high computational demand [BLM+06, Section 7.1.3]. Again, computing the betweenness centrality is the bottleneck. If there are only few edges between different clusters, any approximate shortest path query method would arguably also detect these edges. Approximating betweenness centrality or just finding edges with high betweenness centrality can potentially be sped up using fast point-to-point shortest path queries[10] [RMJ07].

Social Search

The sheer size of the web (currently, the web consists of billions of pages), renders the search for relevant information very challenging. Search engines are expected to find the "needle in the haystack." The search interface is supposed to be kept simple and the average user is not entering much information; the search engine must find relevant information without knowing what the user actually wants (sometimes he may not even know it himself or he may not be able to express it appropriately). Imagine that a search engine would know basically everything about the user (this scenario may actually already be reality). The query term combined with what a user's friends were looking for and which results they liked could enable the search engine to make a well-educated guess on what the user would want to see. Almost like a recommender system [GZC+09], the engine can rank the documents matching the search query term just for one user, based on his interests — a personalized search [JW03]. It would aggregate the ratings for the pages retrieved and then assign a higher ranking to those pages that people similar to that user, for example his friends, liked best. To do so, efficiently retrieving proximity information in the social graph is essential.

Involving context and social connections in ranking is a hot topic for search engines [VFD+07, YBLS08, SR08, UCDG08, SCK+08, PBCG09] and computing social distances may soon be a an important primitive, both for search engines handling keyword queries and for online stores recommending items for purchase.

[10] Distance queries are inherent in graph clustering: a constant-factor k–clustering can be computed with t queries to the distance oracle if and only if a graph k–partition can be computed with t queries to the adjacency matrix of G [GMMO00].

Social Networking

Shortest paths in social networks seem to be of interest for end users. On the website oracleofbacon.org, for example, users can enter two names of actors and "the database server uses a breadth-first search (BFS) to find the shortest path between pairs of actors."[11] Such a webpage also exists for Mathematicians.[12] In professional networking sites such as LINKEDIN or XING, users can add their business contacts in order to get in touch with potential clients or employers through a short chain of personal introductions. The corresponding webservers compute point-to-point shortest paths in an online setting.

Analogous to the Erdős number project there is a social network based on collaboration on scientific articles in various fields. Two scientists are considered connected if they have co-authored one or more scientific papers together [New01]. One may assign weights to these connections based on how many papers two authors share. Such weightings potentially capture the strength of a relationship more appropriately. Shortest paths [New01, Sec. A] between scientists may help to establish new professional contacts by following a sequence of personal referrals. Such systems have been built years before online social networks became popular [KSS97a, KSS97b, Sha97a]. Scientific collaboration networks are also used to enhance communication at conferences. Some researchers built a system called DEAIEXPLORER [KIK$^+$06] that visualizes how two scientists standing in front of a screen may know each other (a common affiliation such as that they wrote a paper together, they both gave a talk at the same conference, they have a common co-author, or they cite each other's papers), which is supposed to help them communicating. The DEAIEXPLORER system computes relationships and connections between persons up to distance 4.

All these systems could potentially benefit from fast shortest path query processing.

Message Routing

Forwarding a message from a sender to a receiver through a network is called routing. Many routing algorithms are variants, in one form or another, of shortest path algorithms that route packets over a path of minimal cost [SS80]. The cost of an edge may reflect transmission capacity, latency, congestion, error rates, and other features. On the chosen path, routers must know where to forward a packet to. These decisions are made based on the information in the packet and the routing table at the router. It is integral that this information is kept small (compact) while paths remain short. Research in *compact routing* addresses this tradeoff. Compact routing focuses on distances and ignores other influencing factors such as the quality of service provided (see [Hui00, Chapter 11: Policy Routing]). It is assumed that these factors are abstracted out and aggregated in the edge weight function. Large routing tables are difficult to cope with [Hui00, Section 9.1.2: Routing Table Explosion, pp. 202–203].

> As more and more networks get connected, the memory required for storing the routing tables grows. This memory requirement varies a lot with the routing protocol and with the router's architecture. In fact, the problem may appear in multiple ways. The phrase "routing table explosion" is merely a catchall term for all the problems posed by the manipulation of very large routing tables.

The tradeoff between routing table size and route quality is basically the distributed version of shortest path query processing. Each router gets some part of the index, based on which

[11] Written by Patrick Reynolds on http://www.oracleofbacon.org/how.php (as of December 2009).

[12] See http://www.ams.org/mathscinet/collaborationDistance.html.

it has to make best-effort decisions. The routing tables should be small and routing decisions quick [DBCP97] while paths remain short.

Distance approximations for networks may be of interest for end users as well. If a sender can choose among different destinations, for example before downloading a large file from one of several replicated servers holding the same data, it is beneficial to predict the round trip time for each of the servers prior to actually communicating. This proximity estimate helps choosing the optimal server and connection [NZ02, DCKM04].

Conclusion

The numerous applications of point-to-point shortest path query processing make the following claim easy to believe. The organizers of the DIMACS implementation challenge on shortest paths, Demetrescu, Goldberg, and Johnson [DGJ08], state that

> ... shortest path problems are among the most fundamental combinatorial optimization problems with many applications, both direct and as subroutines in other combinatorial optimization algorithms. Algorithms for these problems have been studied since the 1950's and still remain an active area of research.

This thesis investigates the tradeoffs between pre-computation time, storage, query time, and approximation quality — both from a theoretical and a practical point of view.

1.3 Contribution

> Theory and Practice. Both of these English words come from the Greek language, and their root meanings are instructive. The Greek $\theta\epsilon\omega\rho\acute{\iota}\alpha$ means seeing or viewing, while $\pi\rho\alpha\kappa\tau\iota\kappa\acute{\eta}$ means doing, performing. [...] Theory and practice are not mutually exclusive; they are intimately connected. They live together and support each other.
>
> *Donald Knuth* [Knu89]

This thesis aims to contribute to both the theoretical and the practical side of the shortest path and distance query problems. The main contributions are summarized in the following. The precise statements are deferred to subsequent chapters.

Theoretical

Space lower bound

The first main contribution is a theoretical analysis of the space requirements for data structures that assist shortest path queries. Given is a graph with n nodes. A preprocessing algorithm computes a data structure of size $\mathcal{S}$. If an algorithm computes approximate shortest paths with multiplicative stretch at most α, it must access the data structure at a certain number $\mathtt{t}$ of locations of the data structure. This three-way tradeoff between size $\mathcal{S}$, stretch α, and query time $\mathtt{t}$ is analyzed in Chapter 4.

This result was achieved in joint work with Elad Verbin and Wei Yu. An extended abstract was published in the proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS) [SVY09].

Time and space upper bounds for power-law graphs

One class of graphs that appears to be very common in real-world networks is the class of power-law graphs: the node degrees obey a power law, which roughly means that there are many nodes with few neighbors and only a few nodes with many neighbors. One property of these graphs is that shortest paths often pass through nodes with many neighbors. This property allows for efficient data structures. Routing through nodes with large degrees is a natural and very common heuristic. Empirical evidence indicates that it is also a powerful heuristic in practice. We make an attempt to bridge the gap between theory and practice by a rigorous efficiency proof of this heuristic for certain random power-law graphs. Details are in Chapter 5.

This result was achieved in joint work with Wei Chen, Shang-Hua Teng, and Yajun Wang. An extended abstract was published in the proceedings of the 23rd International Symposium on Distributed Computing (DISC) [CSTW09a]. The full version is available as a technical report [CSTW09b].

Practical

A third contribution of this thesis is an efficient practical method (with theoretical guarantees) to compute approximate shortest paths in undirected graphs. The preprocessing step consists of computing the analogue of a Voronoi dual for graphs and the query step consists of searching a shortest path in the dual and refining it in the original graph (primal). Compared to many existing practical methods, the method described in Chapter 6 computes approximate shortest paths but it also works for graphs other than road networks (such as *complex networks*).

This result was achieved in joint work with Shinichi Honiden, Michael E. Houle, and Martin Wolff. An extended abstract was published in the proceedings of the 6th Annual International Symposium on Voronoi Diagrams in Science and Engineering (ISVD) [HHSW09]. An outdated version is available as a technical report [SHWH08].

1.4 Outline

This thesis is organized as follows. Chapter 2 contains preliminaries such as various definitions from graph theory and a review of shortest path algorithms for the single source and the all pairs shortest path problems. Related work considering the shortest path query problem is reviewed in Chapter 3. Chapters 4, 5, and 6 contain the main contributions of this thesis, as outlined in Section 1.3.

I never commit to memory anything that can easily be looked up in a book.

Albert Einstein (1879–1975)

2 Preliminaries

This chapter consists of (1) a list of necessary definitions from graph theory and algorithmics, and (2) a review of work related to the shortest path problem in graphs. A detailed review of shortest path query processing is postponed to Chapter 3.

We use the following convenient notation throughout the thesis. For $n \in \mathbb{N}^+$, we define $[n] := \{1, 2, \ldots n\}$. Unless stated otherwise, lg denotes the logarithm with base 2. $poly(x)$ means a polynomial in x of unspecified constant degree. We write modular congruences by $p \equiv_q r$ and $p = r \mod q$.

2.1 Graphs

A graph is a collection of entities (nodes) linked by some relationship (edges).

Definition 1. *A* graph *G is a pair $G = (V, E)$ consisting of a set of nodes V and a set of edges $E \subseteq \binom{V}{2}$.*

Nodes are also referred to as *vertices*.

Definition 2. *A graph $G' = (V', E')$ is a* subgraph *of $G = (V, E)$, if $V' \subseteq V$ and $E' \subseteq E$. An* induced subgraph *is a subset of the vertices of a graph together with any edges whose endpoints are both in this subset.*

Definition 3. *Two nodes $u, v \in V$ of a graph $G = (V, E)$ are called* adjacent *if there is an edge between u and v, that is, $\{u, v\} \in E$. For a graph $G = (V, E)$, the set of* neighbors *of a vertex v, denoted by $\Gamma_G(v)$, is defined as the set of nodes adjacent to v, that is, $\Gamma_G(v) := \{u : \{u, v\} \in E\}$.*

For a set of nodes $U \subseteq V$, let $\Gamma_G(U) := \bigcup_{u \in U} \Gamma_G(u)$.

Definition 4. *For a graph $G = (V, E)$, the* degree *of a vertex v, denoted by $\deg_G(v)$, is defined as the number of its neighbors, that is, $\deg_G(v) := |\Gamma_G(v)|$.*

If the graph G is clear from the context, we omit subscripts. For example, we write the set of neighbors and the degree of a node by $\Gamma(v)$ and $\deg(v)$, respectively.

A graph is called *r–regular* if all vertices have degree r.

The sum of all node degrees divided by two equals the number of edges:

$$\sum_{v \in V} \deg(v) = 2 \cdot |E|.$$

In this thesis, if not stated otherwise, we consider undirected graphs (as in Definition 1). In some networks, relationships between entities are inherently directed, for example one-way streets in road networks or hyperlinks in the World Wide Web. *Directed* graphs can be used to model these networks.

Definition 5. *A* directed graph *(digraph) D is a pair $D = (V, A)$ consisting of a set of nodes V and a set of edges (also called* arcs*) $A \subseteq V \times V$.*

For digraphs, we may distinguish between *in-neighbors* and *out-neighbors*.

Definition 6. *For a digraph $D = (V, E)$, the set of* in-neighbors *of a vertex v is defined as $\Gamma^-_D(v) := \{u : (u, v) \in E\}$, and its set of* out-neighbors *is defined as $\Gamma^+_D(v) := \{u : (v, u) \in E\}$. We define the neighbors of a vertex v as the union of the set of in-neighbors and the set of out-neighbors, $\Gamma_D(v) := \Gamma^-_D(v) \cup \Gamma^+_D(v)$. Its* in-degree *is $\deg^-_D(v) := |\Gamma^-_D(v)|$ and its* out-degree *is $\deg^+_D(v) := |\Gamma^+_D(v)|$.*

Note that $\deg_D(v) \leqslant \deg^-_D(v) + \deg^+_D(v)$ and equality does not necessarily hold. We may again omit the subscript if D is clear from the context.

Weighted graphs also capture relationships of different cost, length, and strength. In what follows, we only consider *edge-weighted* graphs (as opposed to node-weighted graphs). We may still restrict the edge weights to 1, which yields an *unweighted* graph.

Definition 7. *An* edge-weighted graph *(digraph) is a graph (digraph) associated with a weight function $\mathbf{w} : E \to \mathbb{R}$.*

An edge weight can be interpreted as representing a value in the real world such as distance, time, cost, penalty, or loss. In the following, if not stated otherwise, we shall only consider weight functions with positive range, that is, $\mathbf{w} : E \to \mathbb{R}^+$. This explicitly excludes edges with weight 0. This is no restriction, since we may just contract (definition below) edges with weight 0, which yields a single vertex instead.

Definition 8. *A* path *in G from a node u_0 to a node u_h is a sequence of (undirected or directed) edges $((u_0, u_1), (u_1, u_2), \dots, (u_{h-1}, u_h))$. We also interpret such a path as a node sequence $(u_0, u_1, \dots, u_h)$, as a node set $\{u_0, u_1, \dots, u_h\}$, or as a subgraph, when this simplifies the notation. The* length *of a path P is the sum of its edge weights $\ell(P) := \sum_{i=0}^{h-1} \mathbf{w}(u_i, u_{i+1})$. The* hop-length *of a path P is the number of edges h on P.*

Note that for any path of an unweighted graph, the hop-length and the path length are equal.

Definition 9. *A* subpath *P' of a path $P = (u_0, u_1, \dots u_h)$ is a path constructed from a subsequence of nodes $P' = (u_i, u_{i+1}, \dots u_j)$, $0 \leqslant i < j \leqslant h$. A* simple path *is a path without repeated vertices. Two paths are called* vertex-disjoint *if they do not have any vertices in common except for, possibly, the endpoints.*

Distances in graphs are computed based on the shortest path metric.[1]

[1] This connection to metrics is one reason to restrict the range of the edge-weight function $\mathbf{w}(\cdot)$ to $\mathbb{R}^+$ and the graphs to undirected. This thesis does not heavily rely on the general concepts of a metric space but since metrics are inherently designed to measure distance, we briefly outline the basic definition. A *metric space* is a set for whose elements a distance (called a *metric*) is defined. This distance metric is supposed to satisfy three conditions:

1. $d(x, y) = 0$ if and only if $x = y$,

Definition 10. *Let $\mathcal{P}_G(u,v)$ denote the set of paths from u to v in G. The* distance $d_G(u,v)$ *between two nodes u, v is the length of a shortest path from u to v; that is,*

$$d_G(u,v) = \min_{P \in \mathcal{P}_G(u,v)} \ell(P).$$

If $\mathcal{P}_G(u,v) = \emptyset$ then $d_G(u,v) := \infty$. The distance $d_G(u,V')$ between a node u and a subset of the nodes $V' \subseteq V$ is defined as $d_G(u,V') := \min\limits_{v \in V'} d_G(u,v)$. The distance between two subsets of the nodes $U', V' \subseteq V$ is defined as $\min\limits_{u \in U'} d_G(u,V')$.

If *unique* shortest paths are needed, one may perturb the edge weights by adding random infinitesimal weights[2] [MVV87, EHP04].

Definition 11. *An undirected graph $G = (V,E)$ is* connected *if $d_G(u,v)$ is finite for all $u, v \in V$. A directed graph $D = (V', A)$ is* connected *if for all $u, v \in V'$ at least one of $d_D(u,v)$ and $d_D(v,u)$ is finite. A directed graph $D = (V', A)$ is* strongly connected *if for all $u, v \in V'$ both $d_D(u,v)$ and $d_D(v,u)$ are finite.*

The (strongly) connected components of a graph can be extracted efficiently [CLRS01, Section 22.5].

Definition 12. *A* cycle *is a path where both endpoints coincide. A cycle is thus a node sequence $(u_0, u_1, \ldots, u_h)$ for which $(u_i, u_{i+1}) \in E$ for all $i \in \{0, 1, \ldots h-1\}$ and $u_0 = u_h$. The length of a cycle is defined as the number of edges $h \geqslant 3$.*

A cycle of length 3 is also called *triangle*.

Definition 13. *In a graph $G = (V,E)$, the* open ball with radius *r around $v \in V$ is defined by*

$$B_G^r(v) := \{u \in V : d_G(v,u) < r\}$$

Accordingly, the closed ball with radius *r is defined by $B_G^{r+\epsilon}(v) := \{u \in V : d_G(v,u) \leqslant r\}$.*

The open (closed) ball relative to a subset of the nodes $U \subseteq V$ is defined as the open (closed) ball with radius $d(v,U)$.

Definition 14. *Define the* multiplicative stretch *of a path P from s to $t \neq s$ relative to the distance from s to t as the ratio $\ell(P)/d_G(s,t)$ and define the* additive stretch *as the difference $\ell(P) - d_G(s,t)$.*

The stretch of a path is also called *distortion*.

2. symmetry: $d(x,y) = d(y,x)$, and

3. the triangle inequality: $d(x,z) \leqslant d(x,y) + d(y,z)$.

These conditions also imply non-negativity $d(x,y) \geqslant 0$. Let ℓ_p^{dim} denote the Euclidean space of dimension dim, denoted by $\mathbb{R}^{\mathsf{dim}}$, equipped with the ℓ_p–*norm*. For $1 \leqslant p \leqslant \infty$, the ℓ_p–norm on a dim–dimensional space is defined as $||\vec{x}||_p := \sqrt[p]{\sum_{i=1}^{\mathsf{dim}} |x_i|^p}$, set $||\vec{x}||_\infty := \max_i |x_i|$.

[2] The isolation lemma [MVV87] states that, for a finite set of distinct weights $\mathbf{w}(e_1), \mathbf{w}(e_2), \ldots \mathbf{w}(e_{|E|})$, any collection of subsets of weights has a unique minimum with probability at least $1/2$. Edge weights can be made unique by adding infinitesimal weights. Define $\mathbf{w}'(e) := \mathbf{w}(e) + \epsilon \cdot \iota(e)$, where $\iota(e)$ for each edge e is chosen independently at random from $[n^2] = \{1, 2, \ldots n^2\}$.

2.1.1 Graph Properties

Definition 15. *The* diameter $\operatorname{diam}(G)$ *of a graph* $G = (V, E)$ *is the maximum distance between two vertices:*

$$\operatorname{diam}(G) := \max_{u,v \in V} d_G(u, v).$$

We define the empty graph to have infinite diameter and the graph with one vertex to have zero diameter. All other graphs have a diameter in $\mathbb{R}^+ \cup \{\infty\}$. We usually abbreviate $\Delta = \operatorname{diam}(G)$ when G is clear from the context.

Definition 16. *The* radius *of a graph* $G = (V, E)$ *is the least* r *such that there is a vertex* v *whose closed ball* $B_G^{r+\epsilon}(v)$ *covers all vertices.*

Definition 17 (Girth)**.** *The* girth *of a graph* $G = (V, E)$*, denoted by* $\mathsf{g}(G)$ *is the length of its shortest cycle.*

A connected undirected graph without cycles is called a *tree*. A tree with n nodes has exactly $n - 1$ edges. An undirected graph without cycles (but not necessarily connected) is called a *forest*. Since there are no cycles in trees and forests, these graphs have infinite girth. A subgraph that is a tree on all nodes is called a *spanning tree*.

The tree-width of a graph was introduced by Halin [Hal76], but it went unnoticed until it was rediscovered by Robertson and Seymour [RS86] and, independently, by Arnborg and Proskurowski [AP89]. The *tree-width* of a graph is defined as follows.

Definition 18. *Let* G *be a graph,* T *a tree and let* $\mathcal{V} = \{V_t \subseteq V(G) \mid t \in V(T)\}$ *be a family of vertex sets of* G *indexed by the vertices* t *of* T*. The pair* $(T, \mathcal{V})$ *is called a* tree-decomposition *of* G *if it satisfies the following three conditions:*

- $V(G) = \bigcup_{t \in T} V_t$
- *for every edge* $e \in G$ *there exists a* $t \in T$ *such that both ends of* e *lie in* V_t
- *If* $t, t', t'' \in V(T)$ *and* t' *lies on the path of* T *between* t *and* t''*, then* $V_t \cap V_{t''} \subseteq V_{t'}$*.*

The width *of* $(T, \mathcal{V})$ *is the number* $\max\{|V_t| - 1 \mid t \in T\}$ *and the* tree-width $\operatorname{tw}(G)$ *of* G *is the minimum width of any tree-decomposition of* G*.*

Definition 19 (Doubling Dimension)**.** *The* doubling dimension *(also:* Assouad dimension *[Ass83]) of a graph is the minimum* dim *such that any ball of radius* r *can be covered by at most* 2^{dim} *balls of radius* $r/2$*.*

A metric with diameter Δ and doubling dimension dim has at most $\Delta^{\mathcal{O}(\mathsf{dim})}$ points

Definition 20 (Edge Contraction)**.** *In an undirected graph* $G = (V, E)$*, the* contraction *of an edge* $e = \{u, v\}$ *with endpoints* u *and* v *is the replacement of* u *and* v *by a single vertex* u' *such that the edges incident to the new vertex* u' *are the edges other than* e *that were incident with* u *or* v*.*

Definition 21 (Graph Minor [RS83])**.** *A graph* H *is a* minor *of a graph* G *if a copy of* H *can be obtained from* G *via repeated edge deletion and/or edge contraction.*

Graph Class	Excluded Minor
trees	K_3
series-parallel	K_4
outerplanar	K_4 and $K_{2,3}$
planar	K_5 and $K_{3,3}$

***Table 2.1**: Examples of minor-free graph classes.*

2.1.2 Graph Classes

A bipartite graph is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. All forests are bipartite.

Definition 22. *A graph G is* planar *if it can be drawn in the plane such that the edges are represented by line intervals and do not intersect in their interiors.*

There is a famous statement by Leonhard Euler, stating that planar graphs with $n \geqslant 3$ nodes have at most $3n - 6$ edges. If a graph only has few edges ($m \leqslant n \cdot poly(\lg n)$), it is called *sparse*. Often, algorithms run faster on sparse graphs. However, sparsity alone does not necessarily imply that a graph is an 'easy' instance for an algorithm. Besides sparsity, planar graphs have other special structural properties that allow for efficient algorithms: planar graphs can be cut into different pieces without cutting too many edges (the planar separator theorem captures this property, see Theorem 4 in a subsequent section).

Outerplanar graphs are the "easiest" planar graphs.

Definition 23. *An* outerplanar graph *is a planar graph that can be embedded in the plane such that all nodes lie on one face.*

A planar graph can be efficiently decomposed into outerplanar subgraphs (called *hammocks*) [Fre91, Fre95, KPSZ96]. The number of hammocks required induces a hierarchy on the class of planar graphs.

Often, graphs modeling practical networks are 'almost' planar. The planarity property has been extended in several ways.

Planar graphs require the existence of an embedding in the plane without any two edges crossing. This can be generalized to orientable surfaces of larger *genus* (for example genus 0 is a plane and genus 1 is a torus). If the genus is restricted to a constant, these graphs are called *bounded-genus graphs*.

Graphs characterized by forbidden *minors* [RS83] are another special class of graphs. A graph belongs to a *minor-closed* family if and only if it does not have a minor from a certain specified list. Some examples are given in Table 2.1. All graphs in these classes characterized by a finite set of forbidden minors are sparse.

Definition 24. *The* thickness *of a graph $G = (V, E)$ is the minimum number θ of planar subgraphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots G_\theta(V, E_\theta)$ such that $E = \bigcup_{i=1}^{\theta} E_i$.*

There is literally a class of graphs called *bounded X graphs* for any of the aforementioned properties X, where the corresponding property is bounded by a constant $\mathcal{O}(1)$. For example: bounded-degree graphs are graphs in which the degree of each vertex is bounded by a constant.

Another example class is the class of graphs with bounded tree-width (Definition 18). The tree-width is a good measure of the algorithmic tractability of graphs. It is known that a number of hard problems on graphs can be solved efficiently when the given graph has small tree-width [AP89]. A graph has tree-width 1 if and only if it is a forest, and families of graphs with tree-width at most 2 include outer-planar graphs and series-parallel graphs.

A *ball graph* is an intersection graph of balls in $\mathbb{R}^{\mathsf{dim}}$. It consists of n balls with centers v_i and radii r_i. Two centers v_i, v_j are connected in the intersection graph iff their balls intersect in $\mathbb{R}^{\mathsf{dim}}$. A *disk graph* is a ball graph with $\mathsf{dim} = 2$. In *unit-disk* and *unit-ball graphs*, all radii are equal. The class of disk graphs contains the class of planar graphs [Koe36].

2.1.3 Synthetic Graph Models

Ideally, an algorithm would work well for all instances. However, more often than not, one can construct an adversarial graph (also termed worst-case graph) for which certain algorithms show a very bad performance. Almost ideally, an algorithm would work well for all practical instances, or at least for a typical (average) instance. Even though many datasets are made public these days,[3] the number of available real-world networks is still rather limited. Furthermore, the algorithm designer may not know in advance which graph the user will work with. From a theoretical perspective, creating an algorithm for one particular graph instance is trivial: since code size is not measured and evaluated, all solutions can be encoded in advance. Instead, we often evaluate algorithms on certain restricted classes of graphs (see Section 2.1.2). Many real-world networks, however, do not fall into any of these classes.

A large branch of research investigates models of the real world. Since we wish to capture the essential features of multiple networks, the models have some degree of freedom, which is often modeled by randomness. In some random graphs [ER60, Gil59, Bol01] for example each possible edge is in the graph with probability p. In general, these models may help to understand characteristics of certain real-world networks but also to evaluate and test [ASS09] the performance of algorithms. [4]

In the following we briefly review models for the networks discussed in this thesis: road networks and complex networks.

Road Networks

Often planar graphs are used to model road networks.

Eppstein and Goodrich [EG08] and Eppstein et al. [EGS09] explicitly state that road networks are non-planar. Instead, they use a model called *multiscale-dispersed graphs*, formalized in terms of disk graphs (which contain planar graphs [Koe36]). They prove that these networks have small *separators* (see also Theorem 4 in a subsequent section), which can be found efficiently.

Abraham et al. [AFGW10] introduce the notion of *highway dimension*, which means that for every radius $r > 0$, there is a small set of vertices S_r, which all shortest paths of length greater than r pass through.

[3] There are even online platforms to trade datasets, for example `infochimps.org`

[4] Another approach to evaluate the average-case performance of algorithms and to generate test instances is to collect many real-world graphs and perturb edge weights at random [Iri92].

Network	Degree distribution	Example	Model
Single-scale	Gaussian or exponential		Erdős-Rényi [Gil59, ER60]
Scale-free	Power law	Metabolic networks, food webs, Web graphs, and numerous others	Pref. attachment [BA99]; fixed (exp.) degree sequence [BBK72, ACL00]
Broad-scale	Power-law distrib. with sharp 'cut-off' (decay of the tail)	Movie actor	

***Table 2.2**: Complex networks of different scales [ASBS00]*

Complex Networks

Complex networks at first appear not to have any particular structure — this is why they are called complex. For the moment, let us focus on the degree distribution. Amaral et al. [ASBS00] distinguish three classes of networks based on their degree distribution: *single-scale*, *scale-free*, and *broad-scale networks* (see Table 2.2).

Single-Scale Networks. Erdős-Rényi random graphs [ER60, Gil59] have been studied intensively for more than 50 years. There are two common models for undirected graphs with n nodes. In the $G_{n,p}$ model, each of the $\binom{n}{2}$ edges is in the graph independently at random with probability p. In the $G_{n,M}$ model, all graphs with n nodes and M edges have the same probability. Many properties of Erdős-Rényi graphs are well understood. For example, the $G_{n,p}$ random graph with edge probability p proportional to $n^{1/d}$ (where d denotes an integer) has diameter at most $d+1$ with high probability [Bol01]. For more results we refer to [Bol01]. Erdős-Rényi graphs serve as suitable probability distributions for the average-case analysis of many algorithms. However, graphs with power-law degree distribution are very unlikely in the Erdős-Rényi random graph distribution. Since many real-world networks do have power-law degree distributions, researchers also consider other random graph models.

Scale-Free Networks. The node degree sequence of scale-free graphs obeys a *power law* [Mit03, New05, CSN07]. Power-law distributions are referred to as scale-free distributions, since they look the same on any scale. Mathematically speaking, a power-law degree distribution is defined as follows: the probability that a node has degree x is proportional to $x^{-\tau}$ for some τ, which is called the *power-law exponent*. For most practical scenarios, the power-law exponent lies in the interval $2 < \tau < 3$. These inequalities are assumed to hold in the following. Formally, a degree sequence obeys a power law if $\mathsf{Pr}[\deg(v) = x] = C \cdot x^{-\tau}$ for some constant C. The expected degree can be computed as follows.

$$\mathsf{E}\left[\deg(v)\right] = \sum_{x=1}^{n-1} x \cdot \mathsf{Pr}[\deg(v) = x] \leqslant \int_{1}^{\infty} C x^{-\tau+1} \mathrm{d}x = \frac{C}{\tau - 2} \tag{2.1}$$

For constant values of C, the expected number of edges is linear, which makes scale-free networks sparse. The power-law degree sequence is just one important feature of many real-world complex networks. Another characteristic is that distances are very short. This characteristic is called the *small world* effect.

Two broad classes of network models [BS05, Mit03, CF06, TGJ+02] are distinguished based on the method the graphs are generated with. In *pure random graphs*, the number of nodes and the parameters are set at the beginning and then all the edges are generated. These models are satisfactory to analyze complex networks but they do not explain the reasons for the scale-free nature of complex networks. In *random evolving graphs*, the graph is generated by a random process that adds node by node to the graph and connects the new node at random to the existing graph. This process can be stopped at any time. For a generated graph by either model, let n denote the number of nodes. The details for the different models vary greatly. Commonalities other than the power-law degree sequence are that, usually, the diameter is proportional to $\lg n$ and the average distance is proportional to $\lg \lg n$. The goal is to find a model that is both realistic and easy to work with.

The *configuration model* [BBK72, RN04b] works as follows: we specify a degree sequence $\vec{d} := (d_1, \dots d_n)$. The edges are generated such that all graphs $G = (V, E)$ with $\forall v_i \in V : \deg(v_i) = d_i$ have the same probability. Where in the Erdős-Rényi random graph model all edges were *independent*, the edges in the configuration model are *dependent*. Once an edge between two vertices v_i, v_j has been assigned, the potential of both u and v to acquire more edges decreases by 1. Note that, for a degree sequence $\vec{d}$ to be realizable as a graph, there are some conditions on $\vec{d}$ such as $\sum\limits_{i=1}^{n} d_i$ must be even and others [EG60, Hak62].

In the *fixed expected degree random graph model* [ACL00, CL02, NR06], edges are independent. We again specify a sequence $\vec{w} := (w_1, \dots w_n)$. For this model, w_i is interpreted as the *expected degree* of v_i. Each edge $\{v_i, v_j\}$ is in the graph independently at random with probability $\frac{w_i w_j}{\sum_k w_k}$. Note that it is required to restrict $\vec{w}$ such that $\forall i, j : w_i w_j \leqslant \sum_k w_k$. In Chapter 5, we use an adapted version of this model to analyze distance oracles for random power-law graphs.

In the *re-wired lattice model* [BMST97, NW99, Kle00], each vertex is connected to all of its neighbors within constant distance by an undirected edge. In addition, a number of shortcuts (long-range links) are added between randomly chosen pairs of nodes. In a variant, instead of adding edges, some of the connections to neighbors are removed and "re-wired" to random nodes.

In *affiliation networks* [LS09], we start with a bipartite graph. The nodeset is divided into actor nodes and affiliation nodes. Each node representing an actor is connected to certain nodes representing affiliations such as companies, orchestras, and sports clubs. Then, the bipartite graph is "unfolded" into a social network, which consists of actor nodes only; edges are generated such that two actors connected by a path of length 2 in the affiliation graph get connected in the social network.

In the *preferential attachment model* [BA99, DMS00], the network is growing in time in such a way that new vertices are more likely to be connected to vertices that already have a high degree. A new vertex connects to a node with degree d_i with probability $\frac{d_i}{\sum_k d_k}$. This model offers a convincing explanation for the emergence of scale-free networks. The *copy model* [KRR+00] is in some sense a variant of the preferential attachment model, where a new node, upon generation, copies a fraction of the links of a random node.

2.2 Graph Algorithms

Graph algorithms is a research field at the intersection between graph theory and computer science. We are interested in (efficiently) computing certain properties of graphs. An algorithm, given a graph $G = (V, E)$ and optional inputs such as subsets of the nodeset or edgeset or constants, decides or computes certain properties of G. *Decision problems* are those questions for which the answer is “yes” or “no.” *Optimization problems* are the questions for which the solution is a subgraph, potentially ordered, minimizing or maximizing an objective function.

The *efficiency* of algorithms is of integral interest.

> For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, “efficient” means “adequate in operation or performance.” This is roughly the meaning I want.
>
> *Jack Edmonds* [Edm65]

Suppose that we want to evaluate an algorithm for a problem. Objective evaluation criteria include the quality of the result (correctness, exactness, approximation quality), the computing time (also: time complexity), measured in terms of the input size, and the memory consumption (also: space complexity), again measured relative to the input size. For graph algorithms, the input consists of a graph $G = (V, E)$ and optional parameters. Unless stated otherwise, $n := |V|$ denotes the number of nodes and $m := |E|$ denotes the number of edges. An important aspect in the evaluation of an algorithm is its *scalability*. For theoretical work, scalability means the asymptotic behavior of an algorithm in terms of n and m. Let $f_{\mathcal{A}}(n, m)$ denote the least upper bound on the cost of applying algorithm $\mathcal{A}$ to graphs with n nodes and m edges. It is claimed that a constant number of instructions every now and then does not influence the running time too much. It is often also convenient not to analyze these constant overheads in great detail. This is captured in the Bachmann-Landau [Bac94, Lan09] $\mathcal{O}$–notation [CLRS01, Chapter 3]. We say that *the running time of an algorithm is (in)* $\mathcal{O}(g(n, m))$, meaning that the actual running time as a function $f_{\mathcal{A}}(n, m)$ increases, or grows, at most proportionally to $g(n, m)$, ignoring the exact value $f_{\mathcal{A}}(n, m)$. The precise definitions of the $\mathcal{O}$–notation are listed in Table 2.3.

Notation	Definition
$f(n) \in \mathcal{O}(g(n))$	$\exists n_0, c_1, c_2$ such that $\forall n > n_0 : c_1 \cdot g(n) + c_2 \geqslant f(n)$
$f(n) \in o(g(n))$	$\exists n_0, c_1, c_2$ such that $\forall n > n_0 : c_1 \cdot g(n) + c_2 > f(n)$
$f(n) \in \Omega(g(n))$	$g(n) \in \mathcal{O}(f(n))$
$f(n) \in \omega(g(n))$	$g(n) \in o(f(n))$
$f(n) \in \Theta(g(n))$	$f(n) \in \mathcal{O}(g(n)) \wedge f(n) \in \Omega(g(n))$
$f(n) \in \widetilde{\mathcal{O}}(g(n))$	$\exists c'$ such that $f(n) \in \mathcal{O}(g(n) \cdot \lg^{c'} n)$

Table 2.3: *$\mathcal{O}$–notation for the asymptotic behavior of functions f, g.*

Computational problems are classified according to their difficulty, which is defined by the existence of an algorithm running in a certain time. The class of decision problems for which there exists an algorithm that outputs the correct answer in time $\mathcal{O}(poly(n))$ is called $\mathbf{P}$. The class of decision problems for which there exists an algorithm that, given some evidence, verifies the correct answer in time $\mathcal{O}(poly(n))$ is called $\mathbf{NP}$. The complexity classes $\mathbf{P}$ and $\mathbf{NP}$ are only mentioned in some parts of the chapter on related work, but profound knowledge on the $\mathbf{P}$ vs. $\mathbf{NP}$ problem is not essential to understand this thesis. For more on computational complexity theory, we refer to [GJ90].

2.2.1 Computational Models

Time and space complexities are measured differently depending on the machine model [vEB90]. The traditional model of computation consists of a *Turing machine* [Tur37], which is a state machine operating on an infinite tape divided into cells. In the *word RAM model* [CR73] with integral *word length* $w \geqslant 1$, the contents of all memory cells are integers in the range $\{0, \dots, 2^w - 1\}$ and operations such as addition, subtraction, bit shifts, and bit-wise boolean operations are assumed to be executable in constant time (analogous to programming languages such as `C`). This model often allows for fast algorithms (faster compared to addition/comparison models by a logarithmic factor) if for example the edge weights are integers and the largest integer weight W satisfies $W \leqslant 2^w - 1$. This model is often used for upper bounds on the time complexity of a *specific* algorithm.

For lower bounds on the time complexity of *any* algorithm, the related *cell-probe model* is very common.

Definition 25 (Cell-probe model [Yao81, Mil99])**.** *In the* cell-probe model*, a memory cell has w bits (also called word length) and the* space *of a data structure is measured as the number of cells it occupies, denoted by $\mathcal{S}$. The query time is measured by the worst-case number of cells* $\mathtt{t}$ *that a query reads.*

Both for the word RAM and the cell-probe model, the most typical values for the word length are $w = \lg n$ or $w = polylog(n) = poly(\lg n)$, but larger (or smaller) values may be interesting as well.

For problems involving huge data sets, often I/O is the bottleneck of computations. The data does not fit into main memory; instead it is read block by block from disk. To analyze external memory algorithms [AV88, VS94], often a cell-probe-like model is used for upper bounds as well. Operations may read a block of size B into main memory of size M.

2.2.2 Approximation Algorithms

For certain optimization problems, the optimal solution with respect to an objective function is hard to compute. Often the computation of a 'close-to-optimal' solution can be done much faster. An approximate solution may still be acceptable if the quality of the solution is sufficiently good. The quality is measured as follows. Let OPT denote the value of an optimal solution (as deemed by the objective function) and let ALG denote the value of the solution the algorithm returned. We say that the algorithm has approximation quality (α, β) if the inequalities (2.2) hold for *all* allowed inputs. The approximation quality is thus a worst-case measure.

$$\mathsf{OPT} \leqslant \mathsf{ALG} \leqslant \alpha \cdot \mathsf{OPT} + \beta \tag{2.2}$$

(Note that this definition is tailored to minimization problems and in particular to the shortest path problem.)

In the case of distances, this approximation quality is also called *stretch* or *distortion*. Let $\widetilde{d}(u, v)$ denote the result of the approximation algorithm when asked for the distance between u

and v. For an algorithm computing (α, β)*–approximate distances*, for all $u, v \in V$, the result must satisfy

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

(The combination of multiplicative and additive stretch only makes sense for multiple node pairs. For a single path we consider either its multiplicative or its additive stretch (Definition 14).)

2.3 Common Techniques

This section consists of a non-exhaustive list of techniques that are commonly used to solve problems related to shortest paths.

2.3.1 Spanners and Emulators

For most graph algorithms, the performance depends on the number of nodes and edges of the input graph. The running time can potentially be reduced by altering the graph, in particular by adding or deleting edges. After altering the graph, we wish that the answer to the question we ask concerning the graph (in our case the distances between nodes) does not change by much. When edges are deleted only, we obtain a subgraph, which, if it preserves distances to a certain extent, is called a *spanner* [PS89, ADD$^+$93, Coh98, DHZ00, Kor01, BCE03, EP04, TZ06, Pet07, Elk08a, Elk08b, BS08].

Definition 26 ((Graph) Spanner)**.** *An* (α, β)–spanner *of a graph* $G = (V, E)$ *is a subgraph* $G' = (V, E')$ *that approximately preserves distances such that for all pairs of nodes* $(u, v) \in V \times V$*,*

$$d_G(u,v) \leqslant d_{G'}(u,v) \leqslant \alpha \cdot d_G(u,v) + \beta.$$

We say that this spanner has *stretch* (or *distortion*) (α, β).

Spanners are useful in various applications such as constructing routing tables, where the edges of a subgraph are used to route messages, and computing approximate shortest paths.

The more edges we delete, the smaller the input size for the next algorithm, the faster the running time. The amount of edges we can delete before some distances change substantially often depends on the girth of the graph. Recall that the girth of a graph is the length of its shortest cycle (Definition 17). Intuitively, if there are short cycles, we may delete an edge of a cycle, since for a shortest path using this edge, there is an alternative, reasonably short path using the cycle. If there are no short cycles, there is no alternative short path; the redundancy is low and the deletion of an edge may cause a large distortion.

Let $m_g(n)$ denote the maximum number of edges in a graph with n vertices and girth at least g.

Theorem 1 (Althöfer et al. [ADD$^+$93])**.** *For any integer* $\alpha \geqslant 3$*, every graph* $G = (V, E)$ *on* $|V| = n$ *vertices has a spanner with stretch* $(\alpha, 0)$ *and* $m_{\alpha+2}(n)$ *edges.*

Their construction uses a greedy algorithm (similar to Kruskal's algorithm to construct a minimum spanning tree [CLRS01, p. 568]). The upper bound is actually tight. The corresponding lower bound is not very difficult: in a graph with girth $g = \alpha + 2$, removing any edge increases the distance between its endpoints from 1 to at least $\alpha + 1$. The only multiplicative $(\alpha, 0)$–spanner is the graph itself.

Since no edges can be removed from graphs with large girth without significantly altering distances, graphs with many edges (dense graphs) and large girth are important worst-case instances for spanner and distance oracle constructions. In extremal combinatorics, determining $m_g(n)$ is a research field of its own [EJ08, Big98, Hoo02]. For example, for $g = 4$ the question is the following: how many edges can be added to the empty graph on n nodes without closing a triangle? Intuitively, the more edges that were already added, the harder it gets to add another one. For $g = 4$, the complete bipartite graph is asymptotically optimal. For general g, the construction of the graph is much more involved; for some g the value of $m_g(n)$ is not even known.[5] Erdős' girth conjecture [Erd64, ES63] predicts that, for an integer $k \geqslant 1$, $m_{2k+1}(n) = m_{2k+2}(n) = \Omega(n^{1+1/k})$. The corresponding upper bound is known to be tight [AHL02] and the conjectured lower bound is a theorem for certain values of k (1, 2, 3, and 5); for an overview, see Table 2.4.

Girth	$\|E\|$	Reference
4	$\Theta(n^2)$	complete bipartite graphs
6	$\Theta(n^{3/2})$	[Rei58, ERS66, Bro66, Wen91]
8	$\Theta(n^{4/3})$	[Tit59, Ben66, Wen91]
10	$\mathcal{O}(n^{5/4})$ $\Omega(n^{6/5})$	[Tit59, Ben66, LU93]
12	$\Theta(n^{6/5})$	[Tit59, Ben66, Wen91, LU93]
14	$\mathcal{O}(n^{7/6})$ $\Omega(n^{9/8})$	[LUW95, LUW96]
16	$\mathcal{O}(n^{8/7})$ $\Omega(n^{10/9})$	[WU93, LUW95]
$4r+2$	$\mathcal{O}(n^{\frac{2r+1}{2r}})$ $\Omega(n^{1+\frac{1}{3r-1}})$	[LUW95, LUW96]
$4r$	$\mathcal{O}(n^{\frac{2r}{2r-1}})$ $\Omega(n^{1+\frac{1}{3r-3}})$	[LUW95, LUW96]

***Table 2.4**: Results on Erdős' girth conjecture, overview from [TZ05, Table II]. Maximum size of the edge set E for a graph $G = (V, E)$ with $|V| = n$ nodes and given girth (length of a shortest cycle, Def. 17).*

For multiplicative spanners, the tradeoff between space and stretch is well understood. Not so for spanners with additive stretch. Aingworth et al. [ACIM99] found a $(1, 2)$–spanner with $\mathcal{O}(n^{3/2})$ edges and Baswana et al. [BKMP05] found a $(1, 6)$–spanner with $\mathcal{O}(n^{4/3})$ edges. Woodruff [Woo06] gives a strong lower bound for additive graph spanners independent of Erdős' girth conjecture. He proves that for an integer $k = o\left(\frac{\lg n}{\lg \lg n}\right)$, there are graphs for which any $(1, 2k-1)$–spanner has $\Omega\left(n^{1+1/k}/k\right)$ edges.

Recently [TZ06, Pet07], spanners with non-constant additive stretch $\beta(\cdot)$ are under investigation. For these spanners, β is required to be sublinear in $d(u, v)$. We refer to the overview by Pettie [Pet07, Fig. 2]. For a result on spanners for directed graphs, see [RTZ08].

Spanners are subgraphs. If we just care about distances and not about the actual paths, the subgraph requirement may be too restrictive. *Emulators* are graphs restricted to the same nodeset

[5] For directed graphs, the problem seems to be even more involved [CH78, CS83].

but not to the same edgeset.

Definition 27 (Emulator [DHZ00])**.** *An edge-weighted graph $F = (V, E')$ (α, β)-*emulates *a graph $G = (V, E)$ if for every $u, v \in V$*

$$d_G(u, v) \leqslant d_F(u, v) \leqslant \alpha \cdot d_G(u, v) + \beta.$$

*F is called an (α, β)–*emulator.

Consequently, we say that such an emulator has stretch (α, β). Note that, for both spanners and emulators, distances may only increase. Dor et al. [DHZ00] give a $(1, 4)$–emulator with $\widetilde{\mathcal{O}}(n^{4/3})$ edges. Thorup and Zwick [TZ06], for an arbitrary integer $k \geqslant 2$, construct emulators with $\mathcal{O}(kn^{1+1/(2^k-1)})$ edges (in expectation), such that for pairs with distance ℓ, the distance in the emulator is at most $\ell + \mathcal{O}(k\ell^{1-1/(k-1)})$. The lower bounds by Woodruff [Woo06] can be extended to emulators as well.

2.3.2 Distance Labelings and Metric Embeddings

The objective of distance labelings is to assign each node of a graph a label such that the distance (or an approximation thereof) between two nodes can be computed based on the corresponding labels only [Pel00]. Such labelings are used in the real world to a certain extent. For example, postal addresses include countries, cities, and street names, using which we can get an estimate of how close two addresses are.[6] The idea is formalized in the following definition.

Definition 28 (Distance Labeling [Pel00, GPPR04])**.** *An (α, β)–*approximate distance labeling scheme *for a graph $G = (V, E)$ is an assignment of labels to nodes $L : V \to \{0, 1\}^*$ such that the estimated distance $\widetilde{d}(u, v)$ computed by the scheme from the labels $L(v)$ and $L(v)$ satisfies*

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

Distance labeling schemes with short labels are derivable for highly regular graph classes, such as rings, meshes, and hypercubes. An interesting question is whether more general graph classes can also be labeled in this fashion. For general graphs, it is known that any distance labeling scheme must label some graphs with n vertices with labels of size $\Omega(n)$ [GP03b]. Even for planar graphs, some nodes must have labels of size $\Omega(n^{1/3})$ [GPPR04].

If we restrict the function $\widetilde{d}(L(u), L(v))$ to ℓ_p norms, we obtain *embeddings* [IM04, Lin02]. The idea is to map a metric space (here: the shortest path metric of a graph) into a simpler one, in such a way that the distances between points do not change too much. More formally, an embedding of a (weighted) graph $G = (V, E, \mathbf{w})$ with distance function d into a target metric space (V', d') (where d' denotes the distance function of this space) is a map $\varphi : V \to V'$. An embedding with good distortion yields good approximation algorithms [Ind01].

The Johnson-Lindenstrauss Lemma can be used to reduce the number of dimensions of a metric space without introducing a large error. It states that, for any ℓ_2^{DIM}, a random linear mapping into ℓ_2^{dim} preserves distances up to a factor of $1 \pm \epsilon$ with probability at least $1 - e^{-\epsilon^2 \mathsf{dim}}$. More precisely:

[6] Latitude and longitude coordinates would of course be more precise.

Lemma 2 (Johnson and Lindenstrauss [JL84]). *For any $0 < \epsilon < 1$ and any integer n, let* dim *be a positive integer such that*

$$\mathsf{dim} \geqslant 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n.$$

Then for any set V of n points in $\mathbb{R}^{\mathsf{DIM}}$ there is a map $f : \mathbb{R}^{\mathsf{DIM}} \to \mathbb{R}^{\mathsf{dim}}$ such that for all $u, v \in V$,

$$(1-\epsilon)||u-v||^2 \leqslant ||f(u) - f(v)||^2 \leqslant (1+\epsilon)||u-v||^2.$$

The map f can be found in polynomial time.

We can thus project any n–dimensional Euclidean space to an $\mathcal{O}(\lg n/\epsilon^2)$–dimensional space such that the distance between any two points changes by at most $1 \pm \epsilon$. Such projections are almost best possible: at least $\Omega(\lg n/(-\epsilon^2 \lg \epsilon))$ dimensions are needed [Alo03, Theorem 9.3].

We would like to generalize this result to all metric spaces, in particular to graphs and the shortest path metric.

Theorem 3 (Bourgain [Bou85]). *For every n–point metric space there exists an embedding into Euclidean space (ℓ_2) with distortion $(\mathcal{O}(\lg n), 0)$.*

Also, this bound is tight [LLR95]. For the special case of planar graphs, the multiplicative distortion is $\Theta(\sqrt{\lg n})$ [Rao99, NR03]. The optimal distortion of an embedding into a constant-dimensional Euclidean space is hard to compute [MS08a].

Embedding a graph into a hypercube (wherein the ℓ_1 norm is computed as the *Hamming distance* between coordinates) is another technique [Djo73]. The *Squashed Cube Conjecture* by Graham and Pollak [GP72], proven by Winkler [Win83] implies an exact distance labeling scheme with labels of size $n \lg_2 3$. Note that computing the Hamming distance takes time proportional to the label size.

2.3.3 Planar Graph Techniques

Separators

A powerful approach when designing an algorithm is to use *divide & conquer*: we split the problem at hand into a bunch of smaller problems, solve these, and then combine their solutions. Ideally the sub-problems are completely independent such that the combination of their solutions is straightforward. In general, the sub-problems are not independent. It is then crucial to cut the problem into pieces with as few interdependencies as possible. For some problems, such a cutting is indeed possible. Given an instance of the restricted class of planar graphs with n nodes, it is known [LT80, AST94] that we can split the graph into two parts of roughly half the original size separated by a rather small third set of nodes (with size $\mathcal{O}(\sqrt{n})$).

Theorem 4 (Lipton and Tarjan's Planar Separator Theorem [LT80]). *The n vertices of a planar graph can be partitioned into three sets A, B, S such that*

- *no edge connects a vertex in A with a vertex in B,*
- *A and B each contain at most $n/2$ vertices, and*
- *S contains at most $\mathcal{O}(\sqrt{n})$ vertices.*

Results for planar graphs often extend to other classes of graphs. The separator theorem has been generalized to bounded-genus graphs by Gilbert et al. [GHT84] and to minor-free graphs by Alon et al. [AST90].

Road networks, although non-planar (the networks may have many bridges and tunnels), often also have structural properties that are similar to the ones for planar graphs. Most importantly, they appear to have small separators as well [EG08].

Edge Orientability

The edge set of a simple undirected planar graph $G = (V, E)$ can be oriented (denoted by $\hat{G} = (V, \hat{E})$) such that the out-degree of every vertex v satisfies $\deg^{+}_{\hat{G}}(v) = \mathcal{O}(1)$. The upper bound on the degree can be chosen to be 3 [CE91]. Note that the nodes' in-degrees $\deg^{-}_{\hat{G}}(v)$ remain unbounded.

Using this orientation, adjacency queries can be answered in constant time by inspecting both nodes. This technique can be seen as a labeling [Bre66, KNR92]: each node gets a label of size $\mathcal{O}(\lg n)$ bits such that the adjacency of two nodes can be computed by looking at the two corresponding labels only. The result on edge orientability extends to minor-free graphs [GL07].

2.3.4 Well-Separated Pairs

The well-separated pair decomposition by Callahan and Kosaraju [CK95] is important for algorithms operating on point sets in $\mathbb{R}^{\mathsf{dim}}$. The corresponding definition for graphs is as follows. For a graph $G = (V, E)$ and a set $A \subseteq V$, let $G(A)$ denote the subgraph induced by A.

Definition 29 (WSPD [CK95]). *For a graph $G = (V, E)$, two sets of nodes $A, B \subseteq V$ are* ϵ–separated *if* $\max\{\mathrm{diam}(G(A)), \mathrm{diam}(G(B))\} \leqslant \epsilon \cdot d_G(A, B)$*. For a parameter $\epsilon > 0$, a* well-separated pair decomposition *of a graph G is a set of s pairs $\mathcal{W} = \{\{A_1, B_1\}, \ldots \{A_s, B_s\}\}$ such that for all pairs $u, v \in V$*

$$(u, v) \in \bigcup_{i=1}^{s} A_i \times B_i \cup B_i \times A_i,$$

and that for all $i \in [s]$

- *$A_i, B_i \subset V$,*
- *$A_i \cap B_i = \emptyset$, and*
- *A_i and B_i are ϵ–separated.*

In other words, for any pair of nodes $u, v \in V$, there is exactly one pair $\{A_i, B_i\} \in \mathcal{W}$ such that $u \in A_i$ and $v \in B_i$.

2.4 Shortest Paths

> Shortest paths should be of inherent interest to any lazy creature living on a sphere like the surface of planet Earth.
>
> *Mikkel Thorup* [Tho04a]

> Even in very primitive (even animal) societies, finding short paths and searching (for instance, for food) is essential.
>
> *Alexander Schrijver* [Sch05a, p. 1]

The distance between two nodes $s, t \in V$ of a graph $G = (V, E)$ is defined by the length of a shortest path (Definitions 8 and 10). The objective is to find the shortest possible path that connects the source and the target.

In this thesis, we restrict ourselves to unconstrained, (approximate) shortest paths in static, discrete graphs with positive edge weights. For geometric shortest paths we refer to [Mit97, Che96] (for motion planning see [Sha97b]). For paths on surfaces and meshes we refer to [MMP87, ADG$^+$06, VS09]. For the dynamic version of the problem we refer to [DI08, Ita08]. For time-dependent weights, see [CH66, OR90, KS93a]. For SSSP algorithms on graphs with negative edge weights, we refer to [GT89, Gol95, FR06].

We classify [DP84] the shortest path algorithms according to the problem type (single source, all pairs, one pair), the graph class, and the techniques. We also refer to Schrijver's book [Sch03, Chapter 7], Pettie's thesis [Pet03], and the surveys by Zwick [Zwi01] and Sen [Sen09]. The review in this section is made with best efforts; however, the list of methods, algorithms, and techniques is by far not complete.

2.4.1 Single Source Shortest Path (SSSP) Algorithms

For a brief overview, we refer to the Encyclopedia of Algorithms [Pet08b]. For a comprehensive historical overview, we refer to Schrijver [Sch03, Section 7.5b]. The following outline is partially based on Ahuja, Magnanti, and Orlin's book [AMO93]. Historical information is from [Dre69, GM77, DP84, Sch05a].

The Single Source Shortest Path (SSSP) problem asks for a *shortest path* from one node $s \in V$ (called the *source*) to all other nodes in $V \setminus \{s\}$. In particular, after running a single source shortest path algorithm, we know the *distance* from the source to all other nodes. After termination, each node $u \in V$ is *labeled* with $d(s, u)$. At the beginning $d(s, s) = 0$ and all other labels are set to ∞. SSSP algorithms iterate and assign tentative distance labels $\hat{d}(s, u)$ (upper bounds on the true distance $d(s, u) \leqslant \hat{d}(s, u)$) at each step. We distinguish *label-setting*, *label-correcting*, and other algorithms. In label-setting algorithms, each iteration "produces" one optimal label. This property does not hold for label-correcting algorithms, for which all labels are optimal after the final iteration only. Label-setting algorithms are restricted to non-negative lengths, but they often have a better worst-case time complexity than label-correcting algorithms. Also, if we are interested in one particular target node t, it is possible to stop the algorithm as soon as the distance label for t has been produced.

Label-Setting Algorithms

The most famous label-setting algorithm is *Dijkstra's algorithm* [Dij59], see also [Sch03, Section 7.2] or any algorithms textbook [CLRS01, p. 595] of your choice. The original implementation runs in time $\mathcal{O}(n^2)$. According to an article on the history of combinatorial optimization [Sch05a], the same or a similar algorithm has been proposed independently by Leyzorek et al. [LGJ$^+$57], Dantzig [Dan60], and Whiting and Hillier [WH60]; but it is known as Dijkstra's algorithm. The algorithm starts a search at the source. At each step, among the nodes with tentative labels, the algorithm selects the node u with the shortest tentative distance $\hat{d}(s, u)$ and deems the label of u as permanent. Then, the algorithm updates the labels of all the neighbors of u if necessary. That is, if a neighbor $v \in \Gamma(u)$ has a tentative distance $\hat{d}(s, v)$ larger than $d(s, u) + \mathbf{w}(u, v)$, its label is decreased. The algorithm does not need to backtrack: once a label is finalized, it is correct and it can never decrease. The computational bottleneck of the algorithm is the node selection: we need to efficiently find the node u with the shortest tentative distance.

For this node selection step, sorting may be used [Joh72]. Dial [Dia69] proposes to maintain sorted distances by using buckets. Let the weight function $\mathbf{w} : E \to \mathbb{N}^+$ be restricted to integers and let W denote the largest integer weight. Dial's implementation maintains buckets, which could require a large amount of memory; the running time is $\mathcal{O}(m + nW)$. Further improvements were made by Wagner [Wag76], Dial et al. [DGKK79], and Denardo and Fox [DF79].

Another approach to efficient node selection is to use a *priority queue*, which is a data structure we use to manage the nodes [Mur67a]. This data structure supports efficient operations to insert an element, to retrieve the minimum element, to delete the minimum element, and to decrease the value of an element in the queue. These are exactly the operations necessary for Dijkstra's algorithm. Each update of a label involves a queue operation. Using a *binary heap* as priority queue, insertions, deletions, and decrease operations can be done in time $\mathcal{O}(\lg n)$, which yields time $\mathcal{O}(m \lg n)$ for Dijkstra's algorithm [Wil64]. For very dense graphs with $m = \omega\left(\frac{n^2}{\lg n}\right)$ edges this running time is actually slower than the $\mathcal{O}(n^2)$ running time of the original implementation, but for sparse graphs the running time decreases significantly. Johnson [Joh77] uses a priority queue with fixed depth to get running time $\mathcal{O}(m)$ for dense graphs ($m = \Omega(n^{1+\epsilon})$ for some $\epsilon > 0$).

The faster the operations of the priority queue, the better the overall running time. Using a d–heap, insertions and decrease operations take time $\mathcal{O}(\lg_d n)$ and deletions take time $\mathcal{O}(d \lg_d n)$, which yields total running time $\mathcal{O}(m \lg_d n + nd \lg_d n)$. The optimal value for the parameter d is $d = \max\{2, \lfloor m/n \rfloor\}$, which yields total running time $\mathcal{O}(m \lg_d n)$. The *Fibonacci heap* [FT87] supports deletions in time $\mathcal{O}(\lg n)$ and all the other operations in amortized constant time $\mathcal{O}(1)$, which yields the currently best time bound of $\mathcal{O}(m + n \lg n)$. Alternatively, one may use *relaxed heaps* [DGST88] or *rank-pairing heaps* [HST09], which are data structures with the same performance guarantees.

If we restrict the range of the edge weight function $\mathbf{w}(\cdot)$ to integers (or floats), better bounds are possible by using special priority queues designed for the word RAM model (as defined in Section 2.2.1). For an overview of the running times, see Table 2.5.

Component Hierarchy Algorithms

Sorting and priority queues are strongly related [Tho07]. For sorting, there is an information-theoretic time lower bound of $\Omega(n \lg n)$. To circumvent this bound, sorting must be avoided. An SSSP algorithm is not actually required to sort the distances to all nodes. Indeed, the sorting bottleneck can be avoided. Thorup [Tho99, Tho00a] gives an $\mathcal{O}(m)$ algorithm for undirected graphs with integer or floating point weights. He first constructs a *component hierarchy* (in linear time) and then revisits the nodes. Although the algorithm is theoretically best-possible, it appears to be hard to implement and not very efficient in practice [AI00].

Hagerup [Hag00b] generalizes the idea of the component hierarchy to directed graphs, for which his algorithm (using an analogue of minimum spanning trees for directed graphs) runs in time $\mathcal{O}(m \lg \lg W)$. Actually, constructing the component hierarchy itself takes this time; an SSSP search using the hierarchy requires time $\mathcal{O}(m + n \lg \lg n)$.

Pettie and Ramachandran [PR02] generalize the component hierarchy to real weights. For

Time	Reference
$\mathcal{O}(m \lg n)$	[Wil64]
$\mathcal{O}(m + n \lg n)$	[FT87, DGST88]
$\mathcal{O}(m \lg \lg W)$	[Joh82, vEBKZ77]
$\mathcal{O}(m + n\sqrt{\lg W})$	[AMOT90]
$\mathcal{O}(m\sqrt{\lg n})$	[FW93]
$\mathcal{O}(m + n\frac{\lg n}{\lg \lg n})$	[FW94]
$\mathcal{O}(m \lg \lg n)$	[Tho00b]
$\mathcal{O}(m + n \lg^{1/2+\epsilon} n)$	[Tho00b]
$\mathcal{O}(m + n\sqrt{\lg n \lg \lg n})$	[Ram96b]
$\mathcal{O}(m + n \lg^{1/3+\epsilon} n)$	[Ram97]
$\mathcal{O}(m + n \lg \lg n)$	[Tho04b]
$\mathcal{O}(m + n\sqrt{\lg \lg n})$	[HT02]

***Table 2.5**: Running times for different implementations of Dijkstra's algorithm. W denotes the largest integer weight. The table is in large parts excerpted from [Tho99, p. 364]. The algorithms in the first two rows work for both the comparison/addition model and the word RAM model. The analysis of the algorithms shown in row 3 and below only works in the word RAM model.*

real weights and undirected graphs, if the ratio of any two edge weights is polynomial in n, their algorithm runs in time $\mathcal{O}(m + n \lg \lg n)$. Furthermore, the algorithm appears to be efficient in practice as well [PRS02]. The algorithm's idea is to enforce a certain degree of balance in the component hierarchy and, when computing the SSSP, to use a specialized priority queue that takes advantage of this balance [Pet08b]. Unfortunately, it has been found that a hierarchy-based algorithm can not improve upon Dijkstra's algorithm to run in $o(n \lg n)$ for general weights [PR02, Pet04]. See also [Pet03, Section 3.6].

Note that the component hierarchy approach already captures a certain notion of having preprocessing and query stages [AI00].

Label-Correcting Algorithms

In label-correcting algorithms, all distance labels are temporary and they are guaranteed to be exact and optimal in the end only.[7] Label-correcting algorithms can solve more general problems (they can, for example, find a cycle with negative length) and they are more flexible — but they are usually slower. Some version of the algorithm often solves the All-Pairs Shortest Path (APSP) problem. It is actually unknown whether computing the result of a point-to-point shortest path query is easier than computing APSP on arbitrarily weighted graphs.

The generic label-correcting algorithm runs in time $\mathcal{O}(\min\{n^2 mW, m2^n\})$ [For56, Moo59, FF58, GKP85, GKPS85]. There is an efficient FIFO implementation, which runs in time $\mathcal{O}(mn)$ [Bel58] (the well-known *Bellman-Ford algorithm* [Bel67, For56] (for an outline, see also [CLRS01, Chapter 24.1])) and a dequeue implementation running in time $\mathcal{O}(\min\{mnW, m2^n\})$ but being very efficient in practice [Pap74, Pap80, GP86, Ber93].

The Simplex algorithm can also be used to compute shortest paths [Orl83, Akg88, GHK90, AO92, GJ99, SNGM09].

[7] This convergence behavior makes the correctness of an algorithm more difficult to prove.

Restricted Graph Classes, Average-Case Analysis, Expected Running Times, and Practical Considerations

Due to the importance of the shortest path problem, researchers have been interested in parallel algorithms [Coh00, PK85, KS93b, KS96, BTZ98, TZ00, CMMS98, SS99, HTB01, MS03, SPZ06], I/O–efficient algorithms [MZ08, MZ03, BFMZ04, JZ05, MZ06, ALZ07, MO09, Mey09], algorithms for special graph classes such as planar graphs [Fre87, HKRS97], sparse graphs [GOY76, Wag76], or Euclidean graphs [SV86], and in the expected performance (average-case analysis) of algorithms [NMM78, Nos85, Mey03, Gol08, Hag06]. Also, comparisons [Dre69, ZN00] and experimental evaluation for practical purposes [Hit68, BH69, GW73, Pap74, Gol76, II84, IHI$^+$94, CGR96, ZN98, CGS99, JMN99, Shi00, Gol01, PRS02, DI04] have been an important part of investigations.

Point-to-point shortest path problems have been considered early on [Min57, Dan60, Kle64, Smo75]. In practice, point-to-point shortest path algorithms can be computed faster when bidirectional search [Nic66, Boo67, Cha67, Mur67b, Poh71, SdC77, dC83] is used. For road networks, if in addition to the graph the node *coordinates* are known, A* heuristics [Gel63, Sam63, KHI$^+$86, Dor67, HNR68, Gel77] based on geometry help to guide the search towards the target [SV86]. Approaches to the problem of computing shortest paths using an *additional* preprocessed data structure are reviewed in Chapter 3.

2.4.2 All Pairs Shortest Path (APSP) Algorithms

In the following, we consider the static version of the APSP problem, which is for example surveyed by Schrijver [Sch03] and Pettie [Pet08a]. For the dynamic version of the problem, we refer to the Encyclopedia of Algorithms [DI08, Ita08]. We do not list parallel algorithms.

Given a graph $G = (V, E)$, the All Pairs Shortest Paths (APSP) problem is to compute a shortest path, respectively the distance, between all pairs of nodes $s, t \in V$. Since there are $\binom{n}{2} = \Theta(n^2)$ pairs of nodes for which the distance needs to be output, the time complexity must be at least $\Omega(n^2)$. Strong lower bounds are hard to prove [YAR77, GYY80]. Kerr [Ker70] and Nakamori [Nak72] give lower bounds for algorithms with restricted operations. Karger et al. [KKP93] show that any algorithm based on path comparison must take time $\Omega(mn)$. For general algorithms, the gap between the lower and the upper bound is huge.

Given an SSSP algorithm with running time $S(n, m, W)$, the straightforward approach to solve APSP is to run the SSSP algorithm for each node. This approach yields a running time of $\mathcal{O}(n \cdot S(n, m, W))$. For non-negative weights, we may use Dijkstra's algorithm, which yields time $\mathcal{O}(mn + n^2 \lg n)$. The algorithm of Johnson [Joh77] matches this bound also for negative weights (by using Dijkstra's algorithm [Dij59] and the Bellman-Ford algorithm [Bel67, For56]). For very sparse graphs with $m = \mathcal{O}(n)$ edges, this is already best possible.

In practice, for sparse graphs, an initial graph decomposition step may potentially decrease the total computation time [LS67, FLM67, KY65, Mil66, Hu68, HT69, Yen71, GKN74, LR82] (see Figure 2.1). For some graphs we can save a few unnecessary operations by considering *essential* edges only [KKP93, McG95]. Also, there are algorithms that are efficient on average [Spi73, TM80, Blo83, FG85, MT87, MP97, MC09]. In both cases, the worst-case complexity does not change.

The question is: can we do better for dense graphs (with algorithms not based on path comparison)? Label-correcting algorithms may help to improve the running time. The famous *Floyd-Warshall algorithm* [Flo62, War62] is based on dynamic programming and it runs in time $\mathcal{O}(n^3)$ (see also [CLRS01, Section 25.2]). This is optimal if only *triple operations* on paths and edge

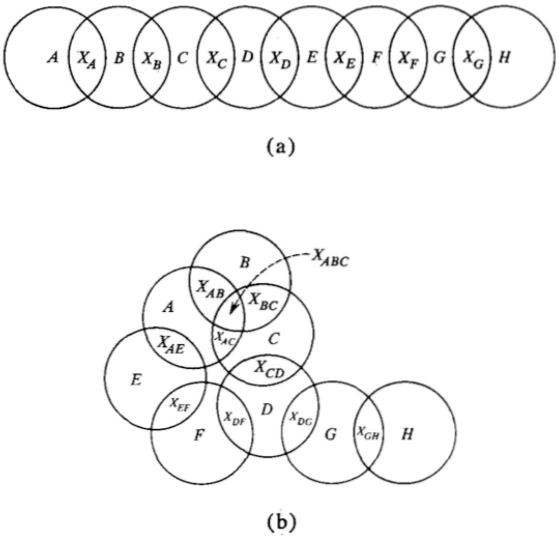

Figure 4 Network decomposition: (a) linearly overlapping sets and (b) nonlinearly overlapping sets.

***Figure 2.1**: Illustration of the network decomposition technique as originally depicted by Hu and Torres [HT69, Figure 4].*

costs are allowed [IN72]. Yen [Yen72, Yen73] and Moffat and Takaoka [MT84] further investigate the constants hidden in the $\mathcal{O}$–notation. Fredman improves the time complexity to $\mathcal{O}\left(n^3\left(\frac{\lg\lg n}{\lg n}\right)^{1/3}\right)$ [Fre76]. Based on Fredman's algorithm, Takaoka [Tak92] improves the running time to $\mathcal{O}\left(n^3\sqrt{\frac{\lg\lg n}{\lg n}}\right)$. Feder and Motwani [FM95] give an $\mathcal{O}\left(mn\frac{\lg(n^2/m)}{\lg n}\right)$–time algorithm for weighted graphs and an $\mathcal{O}\left(n^3\frac{1}{\lg n}\right)$–time algorithm for unweighted graphs. Takaoka gives an algorithm running in time $\mathcal{O}\left(n^3\frac{\lg\lg n}{\lg n}\right)$ [Tak05]. For directed graphs with real edge lengths, Zwick [Zwi06] gives an $\mathcal{O}\left(n^3\frac{\sqrt{\lg\lg n}}{\lg n}\right)$–time algorithm. The algorithm of Chan [Cha07] runs in time $\mathcal{O}\left(n^3\frac{\lg\lg n}{\lg^2 n}\right)$. Blelloch et al. [BVW08] save a lg–factor compared to Feder and Motwani [FM95]. They give the currently fastest combinatorial algorithm, which requires time proportional to $\mathcal{O}\left(mn\frac{\lg(n^2/m)}{\lg^2 n}\right)$. For an overview, see Table 2.6.

In forty years of research, "only" a logarithmic improvement in the running time has been achieved. A combinatorial algorithm running in truly sub-cubic time would be a major breakthrough. Better theoretical bounds exist through an algebraic [Car71] approach: the APSP problem can be solved using matrix multiplication (APSP for directed graphs is at least as hard as boolean matrix multiplication). It is also possible to apply algorithms for fast matrix multiplication [Str69]. We refer to the survey [Tak08] and briefly state the result. Although very important in theory, the algorithms are unfortunately impractical to implement. Let $M(n)$ denote the time it takes to multiply two $n \times n$ matrices. The fastest known algorithm is due to Coppersmith and Winograd [CW90] with $M(n) = \mathcal{O}(n^\omega)$, where $\omega < 2.376$. The APSP problem for undirected graphs with small weights can be solved in time $\widetilde{\mathcal{O}}(M(n))$ [Yuv76, Rom80, GM93, GM97, AGMN92, AGM97, Sei95, Zwi98, SZ99, Zwi02].

Time $\mathcal{O}$	Reference
$mn\cdot \quad 1$	[Bel67, For56]
$n^3 \cdot \quad 1$	[Flo62, War62]
$n^3 \cdot \left(\frac{\lg\lg n}{\lg n}\right)^{1/3}$	[Fre76]
$n^3 \cdot \left(\frac{\lg\lg n}{\lg n}\right)^{1/2}$	[Tak92]
$mn\cdot \frac{\lg\left(n^2/m\right)}{\lg n}$	[FM95]
$n^3 \cdot \left(\frac{\lg\lg n}{\lg n}\right)^{5/7}$	[Han04]
$n^3 \cdot \frac{\lg\lg n}{\lg n}$	[Tak05]
$n^3 \cdot \frac{\sqrt{\lg\lg n}}{\lg n}$	[Zwi06]
$n^3 \cdot \frac{1}{\lg n}$	[Han08a]
$n^3 \cdot \left(\frac{\lg\lg n}{\lg n}\right)^{5/4}$	[Han08b]
$n^3 \cdot \frac{\lg\lg n}{\lg^2 n}$	[Cha07]
$mn\cdot \frac{\lg\left(n^2/m\right)}{\lg^2 n}$	[BVW08]

***Table 2.6**: Combinatorial algorithms for the All Pairs Shortest Path problem.*

All Pairs Approximate Shortest Path (APASP) Algorithms

For some applications, the computation of exact distances may be too expensive. The computational complexity of approximation algorithms is better. We also refer to the survey by Sen [Sen09, Section 5].

For undirected, unweighted graphs, Aingworth et al. [ACIM99] solve APASP with stretch $(1, 2)$ in time $\mathcal{O}(n^{2.5}\lg n)$ and Dor et al. [DHZ00] solve APASP with stretch $(1, \mathcal{O}(\lg n))$ in time $\widetilde{\mathcal{O}}(n^2)$. Cohen and Zwick [CZ01] solve APASP with stretch $(2, 0)$ in time $\widetilde{\mathcal{O}}(n^{3/2}m^{1/2})$, with stretch $(7/3, 0)$ in time $\widetilde{\mathcal{O}}(n^{7/3})$, and with stretch $(3, 0)$ in time $\widetilde{\mathcal{O}}(n^2)$. Baswana and Kavitha [BK06], Berman and Kasiviswanathan [BK07], and Baswana et al. [BGS09] give an $\widetilde{\mathcal{O}}(n^2)$–time algorithm for all pairs $(2, W)$–approximate shortest paths, where W denotes the largest edge weight. Based on matrix multiplication, Zwick [Zwi02] obtains running time $\mathcal{O}(n^\omega/\epsilon)$ for $(1 + \epsilon, 0)$–approximate distances. Roditty and Shapira [RS08] give a "smooth" transition between Zwick's APASP result and the exact APSP algorithms based on matrix multiplication. Their distances have sublinear additive distortion.

2.4.3 Many Pairs Shortest Path (MPSP) Algorithms

We may want to compute the shortest paths between pairs from a restricted subset of vertices $U \subseteq V$. For this, an APSP algorithm might be computing too much. Let $s := |U|$. For $s = \omega(1)$ pairs, the algorithm of Pettie and Ramachandran [PR02] is currently the fastest method to compute exact shortest paths.

Restricted graph classes. The algorithm by Cabello [Cab06] computes s–many distances in planar graphs in time $\mathcal{O}\left(s^{2/3}n^{2/3}\lg n + n^{4/3}\lg^{1/3} n\right)$. The algorithm makes use of an efficient distance oracle for planar graphs (see Section 3.1.3).

Many Pairs Approximate Shortest Path (MPASP) Algorithms

For certain applications, approximate distances between a subset of the nodes may be enough. For an overview of results, see [Elk05, Table I]. Similar to APASP, many algorithms use spanners (subgraphs that approximate distances, for details see Section 2.3) to reduce the problem size. Elkin [Elk05, p. 284] notes that the existence of an algorithm for constructing an (α, β)–spanner with $\mathcal{O}(n^{1+\zeta})$ edges ($\zeta > 0$) in time $\mathcal{O}(T)$ implies the existence of an algorithm for the MPASP problem with s sources with running time $\mathcal{O}(T + s \cdot n^{1+\zeta})$.

Also, any (α, β)–approximate distance oracle (to be defined in the next chapter) with preprocessing time $\mathcal{O}(T)$ and query time $\mathcal{O}(Q)$ can trivially solve MPASP in time $\mathcal{O}(T + s \cdot Q)$. It is interesting to investigate whether it helps when we know the s pairs in beforehand (*offline* scenario).

Aingworth et al. [ACIM99] compute $(1, 2)$–approximate distances in time $\mathcal{O}(n^{1.5}\sqrt{s \lg n} + n^2 \lg^2 n)$. Dor et al. [DHZ00] compute $(1, \lceil 1/\zeta \rceil)$–approximate shortest paths in time $\widetilde{\mathcal{O}}(n^{2+\zeta})$. Cohen [Coh94] computes $(1 + \epsilon, polylog(n))$–approximate shortest paths in time $\mathcal{O}(mn^{\epsilon'} + sn^{1+\zeta})$. Elkin [Elk05] computes $(1 + \epsilon, \mathcal{O}(1))$–approximate shortest paths in the same time. Roditty et al. [RTZ05] construct a distance oracle for a subset of the nodes $U \subseteq V$. Based on this, they can compute $(2k - 1, 0)$–approximate distances in time $\widetilde{\mathcal{O}}(ms^{1/k} + ks)$.

2.4.4 Shortest Path and Distance Queries

Processing shortest path and distance queries in graphs can be seen as a generalization of the APSP problem and the MPSP problem, for which we do not know the pairs in beforehand (*online* scenario). The efficient processing of these queries and the corresponding data structures are the main focus of this thesis. The problem is defined in Chapter 3, wherein we also review related work.

3 Review of Short Path Query Processing

Die Aufgabe [...] besteht darin, ein Verfahren anzugeben, wie man sich aus einem Labyrinthe herausfindet.
(The task is to give a method how to find your way out of a maze.)

Ludwig Christian Wiener (1826–1896) [Wie73]

The eminent importance of the shortest path problem has stimulated a large body of research, both of theoretical and practical nature. Some important results for the general problem (without being exhaustive) are mentioned in Section 2.4. In this chapter, we review results on the *point-to-point (approximate) shortest path query problem*. In this overview, the query problem is restricted to discrete, static graphs with positive edge weights. Also, the only criteria on the optimality of a path is its length (for example, there are no turn penalties [Cal61, KP69]). If paths are required to satisfy constraints other than distance (for example bounded leg paths [DP08]), the corresponding results are not covered in this review.

For distance and path queries in computational geometry, we refer to [Mit97]. For the specific problem in computational geometry concerning robot path planning, we refer to [Sha97b, KLMR98, Lat91]. For questions regarding query processing in dynamic graphs, see [Ram96a, pp. 30–100] and [Ita08, Ber09, BHS07, RZ04, MN67], for time-dependent weights, see [OR90], and for negative weights, see [YZ05].

The query scenario is different from the 'classical' scenario described in the previous sections (2.2 and 2.4). We are first presented with a usually large graph $G = (V, E)$. A so-called *preprocessing* algorithm may compute certain information — or a data structure — to prepare for the next phase. After this preprocessing algorithm has been executed, users will ask *queries*, which should be answered efficiently. In computational geometry, this and similar scenarios are sometimes called *repetitive-mode* (as opposed to *single-shot*) scenarios [PS85, p. 37].

A lazy strategy would be not to precompute a data structure but to use a classical SSSP algorithm to answer queries. The query would then take roughly linear time. An eager strategy would be to precompute the result for all possible queries using an APSP algorithm.[1] Both strategies have its advantages and disadvantages: for the first strategy, no preprocessing is necessary but the query processing is very slow; for the second strategy, the query execution is extremely fast but the preprocessing step is very expensive and the space consumption is prohibitively large for many graphs. In the path query scenario, we mediate between these two extremes: 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.

In the following, we review related work on shortest path and distance query processing for graphs. Theoretical results are presented in Section 3.1 and practical results are summarized in Section 3.2. The review in this chapter is made with best efforts; however, the list of methods and algorithms is not exhaustive.

[1] We assume no knowledge about the query distribution. In practice, we might actually know that some queries are more frequent than others.

3.1 Theoretical — Distance Oracles

Thorup and Zwick [TZ05] 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.

Definition 30. *An* ((α, β)–approximate) distance oracle *for a class of graphs $\mathcal{G}$ consists of a data structure S and a query algorithm with the following characteristics:*

- *The* preprocessing time *is the worst-case time 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 supports (approximate) distance queries *for all pairs of vertices $u, v \in V$, returning a value $\widetilde{d}_S(u, v)$. The query algorithm and its result are characterized as follows.*

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

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

  The stretch is also called distortion*.*

In addition to the worst-case measures, the average-case behavior of the time and space complexities, and, especially, the stretch, may also be of interest. For some distance oracles, only the average stretch is guaranteed. For other distance oracles, the stretch condition is satisfied except for σn^2 pairs of nodes. This fraction $\sigma \in [0, 1)$ is called *slack*. If not explicitly stated otherwise, *stretch* means the worst-case stretch as in Definition 30.

We summarize the known results for general graphs (lower bounds in Section 3.1.1 and upper bounds in Section 3.1.2), and we give an overview of the distance oracles for restricted classes of graphs (Section 3.1.3). Some common techniques are explained in Section 2.3.

3.1.1 Lower Bounds

For directed graphs, distance oracles are closely related to *reachability oracles*. We do not review upper bounds on algorithms for reachability oracles. Any lower bound on the space complexity of reachability oracles directly implies a lower bound on any distance oracles with finite stretch. The first part of the definition for reachability oracles is equivalent to the first part of the definition for distance oracles (Definition 30) except that reachability oracles always consider directed graphs.

Definition 31. *A* reachability oracle *for a class of directed graphs $\mathcal{G}$ consists of a data structure S and a query algorithm with the following characteristics:*

- *The* preprocessing time *is the worst-case time 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 supports reachability queries *for all pairs of vertices $u, v \in V$, returning a boolean value* $\mathsf{reach}_S(u, v) \in \{\top, \bot\}$*. The query algorithm and its result are characterized as follows.*

- *The* query time *is the worst-case time required to compute* $\mathsf{reach}_S(u, v)$ *among all $G = (V, E) \in \mathcal{G}$ and $u, v \in V$.*

- *The query algorithm returns*

$$\mathsf{reach}_S(u, v) = \begin{cases} \top & \textit{if and only if } d_G(u, v) \neq \infty \\ \bot & \textit{otherwise}. \end{cases}$$

For general directed graphs, any distance oracle with finite worst-case stretch requires space $\Omega(n^2)$ bits. Consider a complete bipartite digraph $D = (V_1 \cup V_2, A)$, where all arcs are directed from V_1 to V_2. Any scheme that encodes reachability [AF90] information for D and all its $2^{\Omega(n^2)}$ subgraphs requires quadratic space for some subgraphs.

For sparse graphs with $\mathcal{O}(n)$ edges, the following tradeoff between space complexity and query time has been proven. For details, see Section 4.2.1.

Theorem 5 (Pătraşcu [Pat08a, Theorem 2])**.** *A reachability oracle using space $\mathcal{S}$ in the cell-probe model with w–bit cells, requires query time* $\mathsf{t} = \Omega\left(\lg n / \lg \frac{\mathcal{S}w}{n}\right)$.

Corollary 6. *A distance oracle for directed graphs with finite stretch using space $\mathcal{S}$ in the cell-probe model with w–bit cells, requires query time* $\mathsf{t} = \Omega\left(\lg n / \lg \frac{\mathcal{S}w}{n}\right)$.

For dense graphs, the information-theoretic argument extends to undirected graphs as follows.

For undirected graphs, Thorup and Zwick [TZ05] prove a lower bound on the size of distance oracles for a certain stretch. The worst-case instances are dense graphs with large girth. Recall that the *girth* of a graph is the length of its shortest cycle and recall that $m_g(n)$ denotes the maximum number of edges in a graph with n vertices and girth at least g (for details, see Section 2.3.1).

Theorem 7 (Girth-based lower bound [TZ05, Proposition 5.1])**.** *For any integer $k \geqslant 0$ (called* stretch parameter*), any distance oracle for graphs with n nodes and multiplicative stretch less than $\alpha < 2k + 1$ needs space at least $\Omega(m_{2k+1}(n))$ bits.*

Erdős' girth conjecture [Erd64, ES63] predicts that $m_{2k+1}(n) = m_{2k+2}(n) = \Omega(n^{1+1/k})$. For the values of k for which the conjecture has been proven (including 1, 2, 3, and 5, see Table 2.4), this yields a space lower bound of $\Omega(n^{1+1/k})$ bits.

Thorup and Zwick's lower bound proof [TZ05, Proposition 5.1] roughly works as follows: Recall the argument of the lower bound for graph spanners (Definition 26): in a graph with girth $g = \alpha + 2$, removing any edge increases the distance between its endpoints from 1 to at least $\alpha + 1$. Let G be a graph with n nodes, $m_g(n)$ edges, and girth g. All $2^{m_g(n)}$ subgraphs G' of the graph G also have large girth $\mathsf{g}(G') \geqslant \mathsf{g}(G)$. The distance oracle must distinguish between any two different subgraphs G' and G'', and it cannot omit any edges since there is no alternative short path. Thus, for a distance oracle with stretch less than g, at least $\lg 2^{m_g(n)} = \Omega(m_g(n))$ bits of space are necessary.

Note that the tradeoff is between space and stretch only. The lower bound does not refer to the query time in any way; it even holds if the query algorithm is allowed to access the complete data structure. The lower bound essentially states that we cannot compress certain dense graphs with large girth to less than their original size, or, alternatively, that the size of the data structure must be at least $\Omega(m)$.

3.1.2 General Graphs

In this section, if not stated otherwise, we consider distance oracles for general *undirected* graphs. An overview is listed as Table 3.1. For a recent survey, we refer to Sen [Sen09, Section 4].

Thorup and Zwick, in their seminal work [TZ05], provide both the lower and the matching upper bound: the space complexities of their distance oracles are tight with respect to the space lower bound presented in the previous section (for those values of the stretch parameter k, for which the girth conjecture has been proven). For any integer stretch parameter $k \geqslant 1$, their randomized algorithm (deterministic version by Roditty et al. [RTZ05]) can preprocess a graph with n nodes and m edges in time $\widetilde{\mathcal{O}}(mn^{1/k})$ to construct a distance oracle of size $\mathcal{O}(kn^{1+1/k})$ with query time $\mathcal{O}(k)$ and stretch $2k-1$. Note that for $k=1$ this yields an exact distance oracle with preprocessing time $\mathcal{O}(mn)$, which equals the running time of many APSP algorithms.

Let us look at their method in more detail. For the sake of exposition, let $k=2$. The preprocessing step is outlined as Algorithm 1 (original in [TZ05, Fig. 5]) and the query step is outlined as Algorithm 2 (original in [TZ05, Fig. 2]). For efficient query times, preprocessed information is stored in a hash table [FKS84] for each node). If the actual approximate shortest path is needed, each edge of the path can be generated in constant time $\mathcal{O}(1)$. For a graph $G=(V,E)$ and a node $u \in V$, the open ball with respect to a set $S \subseteq V$ is defined by $B_S(u) := \{v : d(u,v) < d(u,S)\}$ (Definition 13). Let $\mathcal{L}(u)$ denote the node in S that is nearest to u. We also call this node the *landmark* of u.

Algorithm 1 Preprocess $(G=(V,E))$

```
$S = \emptyset$
for each $v \in V$ do
    with probability $n^{-1/2}$, add $v$ to $S$
end for
for each $v \in S$ do
    run SSSP search from $v$ in $G$
    for each node $u \neq v$, store $d(u,v)$ and let FirstNode$_u(v)$ be the penultimum node on the
    shortest path; update $\mathcal{L}(u)$ if $v$ is nearest landmark
end for
for each $u \in V$ do
    compute and store $B_S(u)$ (including distances)
    for each $v \in B_S(u)$ let FirstNode$_u(v)$ be the first node on the shortest path to $v$.
end for
```

Algorithm 2 Distance (s,t)

```
if $s \in B_S(t)$ or $t \in B_S(s)$ then
    return local distance $d(s,t)$ from the information at $s$ or $t$.
else
    return $d(s,\mathcal{L}(t)) + d(\mathcal{L}(t),t)$
end if
```

As mentioned before, the relationship between data structure size and stretch is almost optimal with respect to their lower bound in Theorem 7. For those values of the stretch parameter k for which the girth conjecture by Erdős has been proven (see Table 2.4), $\Omega(n^{1+1/k})$ bits of space are necessary for any distance oracle with multiplicative stretch less than $2k+1$. Thorup and Zwick achieve space $\mathcal{O}(kn^{1+1/k})$ words for multiplicative stretch $2k-1$.

The time complexities (preprocessing and query) are not tight. Both have been improved upon independently.

For weighted graphs, Baswana and Kavitha [BK06] (by using a new APASP scheme) reduce the preprocessing time down to $\mathcal{O}(n^2 \lg n)$. For $k > 2$ they maintain query time $\mathcal{O}(k)$, but for $k = 2$ the query time is $\Theta(\lg n)$. For unweighted graphs, Baswana and Sen [BS06, Theorem 1.2] (by using a $(2,1)$–spanner) reduce the preprocessing time down to expected quadratic time. For unweighted graphs, with an additional additive term (stretch $(3,8)$), Baswana et al. [BGSU08] achieve sub-quadratic preprocessing time $\mathcal{O}(\min\{m+n^{23/12}, mn^{1/2}\})$, space $\mathcal{O}(n^{3/2})$ and query time $\mathcal{O}(1)$. In a recent survey, Sen claims the stretch to be $(3,10)$ instead [Sen09, Table 1]. For general $k \geqslant 3$, Baswana et al. obtain stretch $(2k-1, 2k-2)$, space $\mathcal{O}(kn^{1+1/k})$, and preprocessing time $\mathcal{O}\left(m + kn^{\frac{3}{2}+\frac{4k-3}{k(4k-6)}}\right)$.

Mendel and Naor [MN06], based on Ramsey partitions [LS93, CKR04, BLMN05], reduce the query time down to constant $\mathcal{O}(1)$, but they sacrifice a constant factor in the stretch. For stretch $(\mathcal{O}(k), 0)$, their oracle needs space $\mathcal{O}(n^{1+1/k})$. For their data structure, the best-kown construction time of $\mathcal{O}(mn^{1/k} \lg^2 n)$ is due to Mendel and Schwob [MS08b].

For general graphs, Bourgain's theorem (Theorem 3) yields an $(\mathcal{O}(\lg n), 0)$–approximate distance oracle with query time $\widetilde{\mathcal{O}}(1)$ and space complexity $\widetilde{\mathcal{O}}(n)$.

Derungs et al. [DJW07] consider the setting, where the query algorithm may access both the data structure (termed *index graph*) *and* the original graph (whose representation must remain unchanged). The access to the original graph is limited. In practice, a graph may be stored in external memory but a small index can be stored in internal memory (the definition differs from the definition for typical external memory algorithms [AV88, VS94] — it is related to the *indexing model* [Yao90, DLO03]). Their tradeoff is between the size of the data structure (which may be accessed for free at query time), the number of edges read from the original graph at query time, and the stretch. The index graph is supposed to have sublinear size $o(n)$. The *probe factor* r is defined as the number of edges read from the original graph divided by the number of edges on the result path. Derungs et al. provide an index graph with $\left(\frac{n \lg n}{\sqrt{r}}\right)^{1+2\sigma}$ edges, using which they compute paths of stretch $(4\sigma+1, 0)$, where σ is another parameter. They also prove two lower bounds. For probe factor 0 (no edges can be read from the original graph), the index requires space at least $\frac{n \lg n}{2}$ bits. For probe factor $\frac{n}{10}$ and index size $\frac{n \lg n}{10}$ bits, no stretch less than $(5,0)$ is possible.

For an overview of distance oracles for general graphs, see Table 3.1.

Suppose that we need a distance oracle with quasi-linear space consumption. The oracle of Thorup and Zwick [TZ05] achieves this for $k = \lg n$ with $\mathcal{O}(\lg n)$ multiplicative stretch and $\mathcal{O}(\lg n)$ query time. The oracle of Mendel and Naor [MN06] improves the query time to $\mathcal{O}(1)$. It would be very useful to reduce the stretch to $\mathcal{O}(1)$ instead. The space lower bound proves that such a reduction is impossible for dense graphs. It is however open whether such oracles exist for sparse graphs.

To conclude this section, note that any APSP or APASP algorithm constructs a complete distance table, which may serve as a distance oracle. The space requirement is $\mathcal{O}(n^2)$ and the query time is $\mathcal{O}(1)$. Distances in the table are optimal if computed by an APSP algorithm. For APASP algorithms, the tradeoff between stretch and space is not optimal with respect to the lower bound of Section 3.1.1. APASP algorithms may improve upon APSP algorithms in terms of the preprocess-

Preprocessing	Space	Query	Stretch	Reference
$\mathcal{O}(mn)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	$(1,0)$	APSP
$\mathcal{O}(1)$	$\mathcal{O}(m)$	$\mathcal{O}(m)$	$(1,0)$	BFS
$\widetilde{\mathcal{O}}(kmn^{1/k})$	$\mathcal{O}(kn^{1+1/k})$	$\mathcal{O}(k)$	$(2k-1,0)$	[TZ05, RTZ05]
$\widetilde{\mathcal{O}}(mn^{1/k})$	$\mathcal{O}(n^{1+1/k})$	$\mathcal{O}(1)$	$(\mathcal{O}(k),0)$	[MN06, MS08b]
$\widetilde{\mathcal{O}}(n^2)$	$\mathcal{O}(n^{3/2})$	$\Theta(\lg n)$	$(3,0)$	[BK06]
$\widetilde{\mathcal{O}}(n^2)$	$\mathcal{O}(kn^{1+1/k})$	$\mathcal{O}(k)$	$(2k-1,0)$	[BK06], $k \geqslant 3$
$\mathcal{O}(n^2)$	$\mathcal{O}(kn^{1+1/k})$	$\mathcal{O}(k)$	$(2k-1,0)$	[BS06]
$\mathcal{O}(m+n^{23/12})$	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(1)$	$(3,10)$	[BGSU08]
	$\Omega(n^{1+1/k})$		$<(2k+1,0)$	[TZ05]
	$n^{1+\Omega(1/\mathrm{t})}$	t	$<(1,2)$	Lemma 11
	$n^{1+\Omega(1/\alpha\mathrm{t})}$	t	$<(\alpha,0)$	Theorem 8

***Table 3.1**: Time and space complexities of distance oracles for general, undirected, unweighted graphs (some upper bounds extend to weighted graphs). The upper part of the table lists upper bounds; the lower part lists lower bounds (some restrictions on k and α apply). For the result by Baswana et al. [BGSU08], the stretch is taken from [Sen09, Table 1]. Approximate distance oracles are included only if the space requirement is at most $o(n^2)$.*

ing time. Recall that the distance oracle of Thorup and Zwick [TZ05] is restricted to stretch factors of uneven integers. If multiplicative stretch less than 3 is desired, the preprocessing algorithm of their oracle can only function as an APSP algorithm. The APASP algorithms by Cohen and Zwick [CZ01] yield a $(2,1)$–approximate distance oracle with preprocessing time $\widetilde{\mathcal{O}}(n^{3/2}m^{1/2})$ and a $(7/3,0)$–approximate distance oracle with preprocessing time $\widetilde{\mathcal{O}}(n^{7/3})$. The algorithms by Baswana and Kavitha [BK06] improve this to a $(2,0)$–approximate distance oracle with preprocessing time $\widetilde{\mathcal{O}}(mn^{1/2}+n^2)$ and a $(7/3,0)$–approximate distance oracle with preprocessing time $\widetilde{\mathcal{O}}(m^{2/3}n+n^2)$.

If the input is restricted to special classes of graphs, the girth-based lower bound may not necessarily apply. Distance oracles with better stretch/space tradeoffs exist.

3.1.3 Restricted Graph Classes

Given the prohibitive girth lower bound (Theorem 7), the natural question arises whether we can construct a better distance oracle for certain classes of graphs. Graphs that actually appear in practical settings are of particular interest. Indeed, there are better constructions for restricted graph classes, in particular for several sparse graphs.

Planar Graphs (and Graphs with Bounded Genus)

Due to the importance of planar graphs, short path queries for planar graphs have been studied extensively. A complete review of the corresponding results would deserve a chapter itself. We give a brief overview; a summary can be found in Table 3.2.

For exact shortest path queries, the currently best result in terms of the tradeoff between space and query time is by Fakcharoenphol and Rao [FR06, FR08]. The data structure requires space $\mathcal{O}(n \lg^3 n)$ (improvement for non-negative weights to $\mathcal{O}(n \lg^2 n)$ by Klein [Kle05], another improvement by a $\lg \lg$ factor may be possible [MWN09]) and processes queries in time

$\mathcal{O}(\sqrt{n}\lg^2 n)$. Note that their result holds for graphs with negative weights as well.

Some distance oracles by Cabello [Cab06] (and others) have better query times. The time and space complexities of his distance oracles depend on a parameter r that can be specified by the application. He proves that for any $r \in [n^{4/3}\lg^{1/3} n, n^2]$ there is an exact distance oracle with preprocessing time and space $\mathcal{O}(r)$ with query time $\mathcal{O}\left(\frac{n}{\sqrt{r}}\lg^{3/2} n\right)$. This oracle yields an improvement over complexities of the oracles by Arikati et al. [ACC$^+$96, p. 517] and Djidjev [Dji96]. They prove that, for any $r \in [n^{3/2}, n^2]$, there is a distance oracle with preprocessing time and size $\mathcal{O}(r)$ that supports queries in time $\mathcal{O}(n^2/r)$. Djidjev [Dji96] obtains two more results. For any $r \in [n, n^{3/2}]$, the preprocessing step takes time $\mathcal{O}(n\sqrt{r})$, the other features remain the same. For any $r \in [n^{4/3}, n^{3/2}]$, the query time is $\mathcal{O}\left(\frac{n}{\sqrt{r}}\lg n\right)$. The size remains $\mathcal{O}(r)$; the preprocessing time remains $\mathcal{O}(n\sqrt{r})$. For $r = n^{3/2}$, we obtain query time $\mathcal{O}(n^{1/4}\lg n)$ and space $\mathcal{O}(n^{3/2})$. Cabello's tradeoff [Cab06] yields better preprocessing times for the same amount of space.

The *hammock decomposition* [FJ88, Fre91, Fre95] of a planar graph is used to construct distance oracles optimized for certain classes of planar graphs. Let $q = q(G) \in \{1, 2, \ldots n-1\}$ denote the minimum number of outerplanar subgraphs of a planar graph G (proportional to the minimum number of faces covering all vertices of G; the minimum is taken over all embeddings of G in the plane). The number of hammocks is a topological measure imposing a natural hierarchy on the class of planar graphs. Djidjev et al. [DPZ95] give a distance oracle with preprocessing time and space $\mathcal{O}(n + q\lg q)$ and distance query time $\mathcal{O}(\lg n + q)$. Recall that, for some planar graphs, q may be $\Theta(n)$. If only distance queries are to be answered (no path queries), space $\mathcal{O}(n)$ suffices. Djidjev et al. [DPZ00, DPZ91] also give a dynamic distance oracle for outerplanar graphs with $\mathcal{O}(n)$ preprocessing and space and $\mathcal{O}(\lg n)$ query time. They generalize this distance oracle to planar graphs (and graphs with genus at most $\mathcal{O}(n^\epsilon)$) with q hammocks; the preprocessing and space complexities are $\mathcal{O}(n + q^{3/2})$ and the distance query time is $\mathcal{O}(\lg n + \sqrt{q})$. They obtain a fast linear-space distance oracle for planar graphs with at most $q = \mathcal{O}(n^{2/3})$ hammocks. Chen and Xu [CX00] present a data structure that, for a parameter $1 \leqslant r \leqslant q$, uses $\mathcal{O}(n + q^2/r + q\sqrt{r})$ space, $\mathcal{O}(n + q^2/\sqrt{r} + qr^{3/4})$ preprocessing time, and $\mathcal{O}(\sqrt{r}\lg r + \hat{a}(n))$ query time. $\hat{a}(n)$ denotes the inverse of the Ackermann function. $\hat{a}(n)$ is an extremely slowly-growing function.

Gupta et al. [GKR04] address spatiotemporal queries on objects moving along the edges of a certain class of planar graphs. Spatiotemporal queries (including distance queries) can be processed in time $\mathcal{O}(\sqrt{n})$ on a data structure of size $\mathcal{O}(n\sqrt{n})$ with preprocessing time $\mathcal{O}(n\sqrt{n})$. Their tradeoff matches the tradeoff for the results by Arikati et al. and Djidjev with $r = n^{3/2}$ [ACC$^+$96, Dji96].

Hutchinson et al. [HMZ03] consider shortest path queries in planar graphs in the parallel disk model. This model is used for external memory algorithms [AV88, VS94]. Their data structure uses $\mathcal{O}(n^{3/2}/B)$ blocks of external memory and allows for a shortest path query to be answered in $\mathcal{O}\left(\frac{\sqrt{n}+L}{DB}\right)$ I/O operations, where B is the block size, L is the number of vertices on the reported path, and D is the number of parallel disks. Their result essentially also matches the tradeoffs in [ACC$^+$96, Dji96].

Restricted Queries. Kowalik and Kurowski [KK03] generalize adjacency queries in unweighted planar graphs to queries for distances bounded by a constant h. The preprocessing time and space complexities are $\mathcal{O}(n)$ and the query time is $\mathcal{O}(1)$ (which is an improvement over the $\mathcal{O}(\lg n)$ query time in Eppstein [Epp99, Theorem 12]). To obtain the space bound, they prove that any constant power of a planar graph still has constant thickness (Definition 24). Then, answering a distance query translates to combining the results of a constant number of adjacency queries. Note that for unweighted planar graphs, any distance oracle with stretch $(1, \mathcal{O}(1))$ immediately yields a $(1+\epsilon, 0)$–approximate distance oracle by combining it with the result by Kowalik and Kurowski.

Distances in undirected, unweighted grids are directly determined by grid coordinates. For a weighted, directed $(p \times q)$–grid, Schmidt [Sch98] gives an $\mathcal{O}(pq \lg p)$–time algorithm to build a distance oracle that supports distance queries in time $\mathcal{O}(\lg p)$ for paths starting in the left-most column. In a straightforward way, this yields an $\mathcal{O}(pq \lg pq)$–time algorithm to build a distance oracle that supports distance queries in time $\mathcal{O}(\lg p + \lg q)$ starting at any node on the outer face. For a $(\sqrt{n} \times \sqrt{n})$–grid, the preprocessing and query times are $\mathcal{O}(n \lg n)$ and $\mathcal{O}(\lg n)$, respectively.

This result can be generalized (with rather different techniques). For an undirected graph with genus g, Cabello and Chambers [CC07] provide an $\mathcal{O}(g^2 n \lg n)$–time algorithm to represent the shortest path tree from all the vertices on one specified face. Any query distance from a vertex on this face can be obtained in time $\mathcal{O}(\lg n)$.

Approximations. The tradeoff between space and query time of all exact distance oracles is such that for space $\mathcal{O}(n^{2-\epsilon})$ the query time is at best $\mathcal{O}(n^{\epsilon/2})$ [Cab06]. Constant query time with space $o(n^2)$ has not been achieved yet. To obtain fast query times of $\mathcal{O}(1)$ or $\widetilde{\mathcal{O}}(1)$ with lower space requirements, approximate distance oracles are considered. Recall that, for general graphs, any distance oracle with multiplicative stretch less than $\alpha < 3$ requires space $\Omega(n^2)$. In the special case of planar graphs, the corresponding tradeoff is much better.

Thorup [Tho04a] presents efficient $(1+\epsilon)$–approximate distance oracles for planar digraphs. The main ingredient of Thorup's construction [Tho04a] is a special separator consisting of a set of shortest paths (instead of a general set of $\mathcal{O}(\sqrt{n})$ nodes as in the Lipton-Tarjan [LT80] separator theorem (Theorem 4)). Each node computes the distance to certain nodes of these separator paths. His method achieves the fastest (and only) provable query times for planar digraphs. His oracle has also been implemented and tested for large road networks [MZ07]. The results indicate that, for these road networks, it is not competitive with the specialized methods to be discussed in Section 3.2.3. For undirected graphs, Thorup provides two oracles.

1. The first distance oracle has query time $\mathcal{O}(1/\epsilon)$, preprocessing time $\mathcal{O}\left(n\epsilon^{-2} \lg^3 n\right)$, and space requirement $\mathcal{O}(n\epsilon^{-1} \lg n)$ [Tho04a, Theorem 3.19]
   Klein [Kle02a] independently obtains the same oracle for space and query time by improving Thorup's second oracle.

2. For the slower query time of $\mathcal{O}\left(\lg n (\lg\lg \Delta + \epsilon^{-1})\right)$, space $\mathcal{O}(n\epsilon^{-1} \lg n \lg \Delta)$ and preprocessing time $\mathcal{O}(n\epsilon^{-1} \lg^2 n \lg \Delta)$ are sufficient [Tho04a, Proposition 3.14].
   Klein [Kle05, Section 7] improves the preprocessing time to $\mathcal{O}(n(1/\epsilon + \lg n) \lg n \lg \Delta)$.

The original separator theorem [LT80] can be generalized to minor-free graphs [AST90]. An extension is also possible for Thorup's shortest path separator theorem [Tho04a]. Abraham and Gavoille [AG06] generalize it to minor-free graphs. Based on these shortest path separators, they construct distance oracles for graphs without fixed minors (including planar graphs). The graph at hand is recursively cut into pieces of almost equal size, separated by at most $\mathcal{O}(1)$ shortest paths. In polynomial time, they obtain a $(1+\epsilon, 0)$–approximate distance oracle with space $\mathcal{O}\left(\frac{n}{\epsilon} \lg n\right)$ and query time $\mathcal{O}\left(\frac{\lg n}{\epsilon}\right)$.

For an overview, see Table 3.2. Recall that Δ denotes the diameter (largest finite distance) of a graph.

Preprocessing	Space	Query	α	Reference
$\mathcal{O}(n \lg^2 n)$	$\mathcal{O}(n \lg^2 n)$	$\mathcal{O}(\sqrt{n} \lg^2 n)$	1	[FR06, Kle05]
$\mathcal{O}(n^{3/2})$	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(\sqrt{n})$	1	[ACC$^+$96, Dji96]
$\widetilde{\mathcal{O}}(n^{4/3})$	$\widetilde{\mathcal{O}}(n^{4/3})$	$\mathcal{O}(n^{1/3} \lg^{4/3} n)$	1	[Cab06, $r : n^{4/3}$]
$\mathcal{O}(n^{3/2})$	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(n^{1/4} \lg^{3/2} n)$	1	[Cab06, $r : n^{3/2}$]
$\mathcal{O}(n^{7/4})$	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(n^{1/4} \lg n)$	1	[Dji96, $r : n^{3/2}$]
$\mathcal{O}(n^{7/4})$	$\mathcal{O}(n^{7/4})$	$\mathcal{O}(n^{1/8} \lg^{3/2} n)$	1	[Cab06, $r : n^{7/4}$]
$\mathcal{O}\left(\frac{n}{\epsilon^2} \lg^3 n\right)$	$\mathcal{O}\left(\frac{n}{\epsilon} \lg n\right)$	$\mathcal{O}(1/\epsilon)$	$1+\epsilon$	[Tho04a, T3.19]
	$\mathcal{O}\left(\frac{n}{\epsilon} \lg n\right)$	$\mathcal{O}(1/\epsilon)$	$1+\epsilon$	[Kle02a]
$\mathcal{O}\left(\frac{n}{\epsilon} \lg^2 n \lg \Delta\right)$	$\mathcal{O}(\frac{n}{\epsilon} \lg n \lg \Delta)$	$\mathcal{O}(\lg n \lg\lg \Delta + \frac{\lg n}{\epsilon})$	$1+\epsilon$	[Tho04a, P3.14]
$\mathcal{O}(\frac{n}{\epsilon} + n \lg^2 n \lg \Delta)$	$\mathcal{O}(\frac{n}{\epsilon} \lg n \lg \Delta)$	$\mathcal{O}(\lg n \lg\lg \Delta + \frac{\lg n}{\epsilon})$	$1+\epsilon$	[Kle05, Sec. 7]
$\mathcal{O}(poly(n))$	$\mathcal{O}\left(\frac{n}{\epsilon} \lg n\right)$	$\mathcal{O}\left(\frac{\lg n}{\epsilon}\right)$	$1+\epsilon$	[AG06]

***Table 3.2**: Time and space complexities of distance oracles for undirected planar graphs (some results extend to planar digraphs and/or minor-free graphs). α denotes the multiplicative stretch; Δ denotes the diameter.*

Approximate shortest path query processing for planar graphs has been investigated before Thorup's and Klein's seminal results. Frederickson and Janardan [FJ89, FJ90] give stretch $(3, 0)$–approximate routing schemes for planar graphs. Klein and Subramanian [KS98] give a data structure that also works in the dynamic case. The stretch is $(1 + \epsilon, 0)$; query and update times are $\mathcal{O}(\epsilon^{-1} n^{2/3} \lg^2 n \lg \Delta)$.

Road Networks – Graphs with Bounded *Highway Dimension*

An important characteristic of road networks appears to be its *highway dimension* [AFGW10]. A graph is said to have small highway dimension if for any $r > 0$, there is a "not-too-large" set of vertices S_r, which all shortest paths of length at least r pass through. S_r is not too large if all balls of radius at most $\mathcal{O}(r)$ contain only few vertices of S_r. Based on this notion, many efficient practical methods (to be discussed in Section 3.2.3) such as contraction hierarchies [GSSD08, BDS$^+$08], highway hierarchies [SS05, SS06, NBB$^+$08], transit-node routing [BFM$^+$07, Sch08b], and SHARC [BD08] have provable performance guarantees. The highway dimension of real road networks has not been investigated yet.

Road networks also share some properties with planar graphs such as small separators [EG08]. The techniques of Thorup [Tho04a] may potentially apply too (experimental results indicate so [MZ07]).

Bounded Tree-Width

For digraphs with tree-width w (Definition 18), Chaudhuri and Zaroliagis [CZ00] give an algorithm, parameterized by an integer $q \in [1, \hat{a}(n)]$, to compute a distance oracle with query time $\mathcal{O}(w^3 q)$ in preprocessing time $\mathcal{O}(w^3 n \lg n)$ for $q = 1$, preprocessing time $\mathcal{O}(w^3 n \lg^* n)$ for $q = 2$, and preprocessing time $\mathcal{O}(w^3 n)$ for $q = \hat{a}(n)$. The tradeoff between preprocessing and query time arises from semigroup computations over trees [AS87, Cha87, Sei06]. $\hat{a}(n)$ denotes the inverse Ackermann function and $\lg^* n$ denotes the iterated logarithm, which is the the number of times the logarithm function has to be applied recursively to reach 1.

Hagerup [Hag00a] obtains the same results (and extends it to dynamic oracles) using monadic second-order logic. Courcelle and Vanicat [CV03b] obtain a related result for graphs with bounded *clique-width* [ER97, CO00]. Their distance oracle can be computed in time $\mathcal{O}(n \lg n)$ and (exact) distance queries can be answered in time $\mathcal{O}(\lg n)$.

k–chordal graphs

A graph is *k–chordal* if there is no cycle with more than k edges and no *chord*. A chord is an edge between two nodes of the cycle that are not neighbors in the cycle. *Chordal* graphs are 3–chordal. Gavoille et al. [GKK$^+$01] give a $(1, \lfloor \frac{k}{2} \rfloor)$–approximate distance labeling scheme with labels of size $\mathcal{O}(\lg n \lg \Delta)$ for k–chordal graphs. They also prove that these label sizes are optimal. Their result implies the existence of a $(1, 1)$–approximate distance oracle for chordal graphs with size $\mathcal{O}(n \lg^2 n)$.

Intersection Graphs and Permutation Graphs

Chen et al. [CLSS98] consider interval and circular-arc graphs. A graph $G = (V, E)$ is an *interval graph* if there exists a set S of intervals on the real line (called *model*) such that there is a bijection between the vertices $v \in V$ and the intervals $I_v \in S$ in such a way that an edge $(u, v) \in E$ if and only if $I_u \cap I_v \neq \emptyset$. Given a model, the input intervals get sorted by their endpoints (this sorting step may take time $\mathcal{O}(n \lg n)$); after this, preprocessing takes time $\mathcal{O}(n)$. The distance oracle has size $\mathcal{O}(n)$ and supports distance queries in time $\mathcal{O}(1)$. The definition of circular-arc graphs is the same as that of interval graphs, with the exception that the set of intervals on the real line is replaced by a set of circular-arcs on a unit circle. Sprague and Takaoka [ST99] give a simpler method with the same performance guarantees. Gavoille and Paul [GP03a] obtain the same performance using distance labelings.

Recall that a *ball graph* is an intersection graph of balls in $\mathbb{R}^{\mathsf{dim}}$. It consists of n balls with centers v_i and radii r_i. Two centers v_i, v_j are connected in the intersection graph iff their balls intersect in $\mathbb{R}^{\mathsf{dim}}$. A *disk graph* is a ball graph with $\mathsf{dim} = 2$. In *unit-disk* and *unit-ball graphs*, all radii are equal. Gao and Zhang [GZ05, Corollary 5.2] consider unit-disk (and unit-ball) graphs. Based on the well-separated pair decomposition by Callahan and Kosaraju [CK95] (Definition 29), they obtain, in preprocessing time $\mathcal{O}(n\sqrt{n \lg n}/\epsilon^3)$, a data structure of size $\mathcal{O}(n \lg n/\epsilon^4)$ that supports $(1+\epsilon, 0)$–approximate distance queries in time $\mathcal{O}(1)$. Fürer and Kasiviswanathan [FK07] give efficient algorithms for disk and ball graphs with general radii. Let $R := \frac{\max r_i}{\min r_i}$ denote the ratio between the largest and the smallest radius in the graph. They construct sparse $(1 + \epsilon, 0)$–spanners (Definition 26), which have a separator of size $S(n, R, \epsilon) = \mathcal{O}(n^{1-1/\mathsf{dim}}\epsilon^{-\mathsf{dim}+1/2} + \epsilon^{-2\mathsf{dim}+1} \lg R)$ that can be found in time $\mathcal{O}(n \lg n)$. After preprocessing of time $\widetilde{\mathcal{O}}(n \cdot S(n, R, \epsilon))$ their distance oracle can answer queries in time $\mathcal{O}(S(n, R, \epsilon))$. Recall that the class of disk graphs contains the class of planar graphs [Koe36]. The oracle of Fürer and Kasiviswanathan is more general than the oracles for planar graphs but not competitive on planar instances.

Let π denote a permutation of $[n] = \{1, 2, \ldots n\}$. The *permutation graph* $G(\pi)$ is defined as the graph with vertex set $[n]$ and edge set $\{\{i, j\} : (i - j) \cdot (\pi^{-1}(i) - \pi^{-1}(j)) < 0\}$. Sprague [Spr07] gives an algorithm that preprocesses a permutation graph in time $\mathcal{O}(n)$ such that distance queries can be answered in time $\mathcal{O}(1)$.

Graphs with Bounded Doubling Dimension

The doubling dimension of a metric space is the minimum dim such that any ball of radius r can be covered by at most 2^{dim} balls of radius $r/2$ (Definition 19).

Talwar [Tal04] provides distance labels of length $\mathcal{O}(\epsilon^{-\mathsf{dim}})$, using which there is a $(1+\epsilon, 0)$–approximate distance oracle with space $\mathcal{O}(n\epsilon^{-\mathsf{dim}})$ and query time $\mathcal{O}(\epsilon^{-\mathsf{dim}})$. Abraham et al. [ABN08] provide, for a parameter $\theta \in (0,1]$, an $(\mathcal{O}(\lg^{1+\theta} n), 0)$–approximate distance oracle with space complexity $\mathcal{O}(n\mathsf{dim}/\theta)$ and query time $\mathcal{O}(\mathsf{dim}/\theta)$. The stretch is worse but the dependencies on the doubling dimension dim are better.

The beacon-based embedding by Kleinberg et al. [KSW09] works as follows. A set of landmarks is selected (for example at random); a constant number of landmarks suffices. Each node stores its distance to all the landmarks. Distances are approximated by triangulation. Distance estimates are unbounded for a σ–fraction of the node pairs. This fraction is referred to as *slack*. For all the remaining node pairs, the triangulation yields a $(1+\epsilon, 0)$–approximate distance oracle with linear space complexity and constant query time.

Graphs with bounded doubling dimension have also been studied in the setting of compact routing [AM05, AGGM06, KRX07] and spanners [CG06]. Some researchers suggest to apply algorithms that work well for graphs with bounded doubling dimension to Internet-like graphs. However, the graph representing the Internet does not appear to have bounded ball growth or bounded doubling dimension. Both measures can be large [FLV08].

Power-law Graphs and Complex Networks

For power-law graphs, compact routing schemes have been studied. Any compact routing scheme may serve as an approximate shortest path oracle. While the time to retrieve the approximate distance may not be competitive, we can retrieve each edge of the path by simulating the decision of a router in a centralized way such that path queries are efficient. The results for distance oracles and compact routing schemes are often strongly related. For a routing scheme, all the information needs to be distributed. This constraint renders the problem intuitively harder than constructing a distance oracle, where information is centralized and the query algorithm is not restricted to local information. We do not attempt to cover results on compact routing in this review. We instead refer to [ANLP90, AP92, Gav01, GP03b, TZ01, AGM$^+$08, CW04] and the references therein. Compact routing schemes for power-law graphs with direct influence on the best results for distance oracles are discussed in the following.

Krioukov et al. [KFY04] evaluate the compact routing scheme by Thorup and Zwick [TZ01] for Internet-like inter-domain topologies and random power-law graphs. They also analyze the stretch-distribution for this routing scheme when run on Erdős-Rényi $G_{n,p}$ random graphs [ER60]. Enachescu et al. [EWG08] also analyze the compact routing scheme by Thorup and Zwick [TZ01] for Erdős-Rényi $G_{n,p}$ random graphs [ER60]. They prove that stretch $(2,0)$ can be achieved with space $\widetilde{\mathcal{O}}(n^{7/4})$ by selecting $\widetilde{\mathcal{O}}(n^{3/4})$ landmarks. They also claim (without proof in the proceedings version) that stretch $(\alpha, 0)$ can be achieved with space $\widetilde{\mathcal{O}}\left(n^{1+\frac{2}{\alpha+1}+\epsilon}\right)$. Recall that the Erdős-Rényi random graphs do not have a power-law degree sequence (Section 2.1.3).

The compact routing scheme by Brady and Cowen [BC06] is evaluated experimentally only. More on their scheme can be found in Section 3.2.4 on practical results.

Geometric Graphs

A *geometric graph* $G = (V, E)$ has vertices corresponding to points in $\mathbb{R}^{\mathsf{dim}}$ and edge weights from a Euclidean metric; G is said to be a (α, β)–spanner for V, if for any two points p and q in V the shortest path metric in G $(\alpha, 0)$–approximates the Euclidean distance in $\mathbb{R}^{\mathsf{dim}}$. Since some distances may be larger than the corresponding Euclidean distance, we can not just apply Johnson-Lindenstrauss [JL84] (Lemma 2) to obtain an efficient distance oracle for G. For geometric graphs with sparse spanners, Gudmundsson et al. [GLNS08] give a $(1+\epsilon, 0)$–approximate distance oracle with preprocessing time $\mathcal{O}(m \lg n)$, space $\mathcal{O}(n \lg n)$, and query time $\mathcal{O}(1)$.

Andersson et al. [AGL07] extend the result to geometric graphs with dense clusters using the well-separated pair decomposition by Callahan and Kosaraju [CK95] (Definition 29) and well-separated clusters by Krznaric and Levcopoulos [KL95]. If G contains N disjoint $(\alpha, 0)$–spanners that are inter-connected with M edges, there is an algorithm that constructs an $(1+\epsilon, 0)$–approximate distance oracle in time $\mathcal{O}((m + M^2) \lg n)$ with space $\mathcal{O}(M^2 + n \lg n)$ and constant query time. Their algorithm chooses a representative point for each cluster, based on which distances are computed. As a potential application they give the following example: in the European railway network, each country has a its own network (a spanner) and the railway networks of the countries are then sparsely interconnected.

Sankaranarayanan and Samet [SS09, SSA09] adapt the well-separated pair decomposition to spatial networks of dim dimensions. They obtain $(1 + \epsilon, 0)$–approximate distance oracles using space $\mathcal{O}(n/\epsilon^{\mathsf{dim}})$ and query time $\mathcal{O}(\lg n)$. With a hash table, the query time can be reduced to $\mathcal{O}(1)$ while the space consumption increases to $\mathcal{O}(n \lg n/\epsilon^{\mathsf{dim}})$. They also evaluate their scheme experimentally on road networks.

3.2 Practical

Efficient practical methods to process shortest path queries are often devised by following a feedback loop that consists of design, analysis, implementation, and experimentation. The approach using this feedback loop is also called *algorithm engineering* [San09, Figure 1]. Since experimentation is an integral part of the feedback loop, the choice of the datasets may highly influence the outcome of the algorithm engineering process. If possible, experiments are run with input graphs that are actually used in practice.

Practical instances for shortest path problems are often sparse. The number of edges is roughly linear in the number of nodes. Besides sparsity, practical networks often have other important properties. A large fraction of the efforts in the field of practical shortest path query processing has been devoted to transportation networks, in particular to road networks. Road networks, for example, share many properties with planar graphs; in particular, road networks also have small separators. In practice, however, approaches directly based on separators are often not the most efficient ones.

Experimental evaluation [Hit68, BH69, Dre69, GW73, Pap74, Gol76] has always been an important part of shortest path research. For the first practical methods devised by the algorithm engineering approach, the feedback loop was rather short. Researchers found that the representation of a graph in memory 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 [DGKK79]. Such a sorting step preprocesses the graph in order to obtain faster query times. It may also be efficient to reorder the vertices such that proximity in the graph is reflected in proximity in memory as well. Such a reordering may have great impact on the running time of the query algorithm due to caching effects [GKW07].

Reordering nodes and edges was just the beginning. If additional structure is computed, or if the network is changed structurally, the investment during the preprocessing phase is higher but so is the payoff at query time. Network decomposition [LS67, FLM67, KY65, Mil66, Hu68, HT69, Yen71, GKN74, LR82] was used to speedup APSP algorithms on sparse networks. Other than the articles on the network decomposition technique, the thesis of Smolleck [Smo75, SC81] and the article by van Vliet [Vli78] appear to be among the first reports on the shortest path query problem with "considerable" preprocessing.[2]

Smolleck models the network by an electric circuit, wherein each edge is mapped to an impedance. According to [DP84], 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 apparently 1,000 times slower than the query time.

Van Vliet compares the running times of Dijkstra's [Dij59], D'Esopo's [PW60, Pap74], and Moore's [Moo59] algorithms on road networks with up to 5,337 nodes and 14,930 edges. Based on his observations, he introduces heuristics termed "spider web techniques" [Vli78, Section 6]. He contracts nodes such that *"groups of 2 or more links from the original network are combined into single links representing minimum distance paths between their end nodes."* For an illustration, see Figure 3.1. He attributes the idea to Hu [Hu69], who termed it "distance equivalent networks";[3] he also relates it to "triple operations" [Flo62, Mur65, Hu68]. 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. Van Vliet combining APSP and SSSP techniques into a query method illustrates the tradeoff that shortest path query methods address. Van Vliet's contraction techniques decrease the CPU time for multiple queries by approximately 25%.

For recent methods, two preprocessing strategies are distinguished. *Hierarchical approaches* compute an additional graph structure to speed up shortest path queries. Approaches based on *graph annotation* attach additional information to each vertex, based on which, at query time, the search tree can be pruned.

3.2.1 Hierarchical Approaches

Hierarchical methods to compute shortest paths in graphs have been proposed by many researchers [KK77, AJ94, SWN92, SFG97, IOAI91, CF94, JHR96, TF97, FS97, CRS98, CTB01, HJR95, CL07, CZ07, AY00, JP02, AY01, HSW08, SS06, BFM$^+$07, Sch08b, KKRS08, GSSD08, Hol08, BDS$^+$08]. 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.

If, for each level, the size of the graph is reduced by a constant factor, the hierarchy contains $\mathcal{O}(\lg n)$ levels. In practice, it may be beneficial to stop the recursive process when only $\mathcal{O}(\sqrt{n})$ nodes are left. For these remaining nodes, a distance table can be computed and stored. This yields more efficient query algorithms at a comparably low preprocessing cost.

[2] An earlier approach by Bazaraa and Langley [BL74] was to preprocess a graph in order to "eliminate" *negative* weights such that Dijktra's algorithm can be applied.

[3] There may be a connection to the "minimum-route transformations" by Akers [Ake60] and William S. Jewell (no reference). These network changes are based on "Wye-Delta" $Y - \Delta$–transformations of electrical networks. However, the transformations appear to be restricted to planar networks and to two or three terminals. Hu and Torres [HT69, p. 390] attribute smaller "flow equivalent networks" to Akers [Ake60].

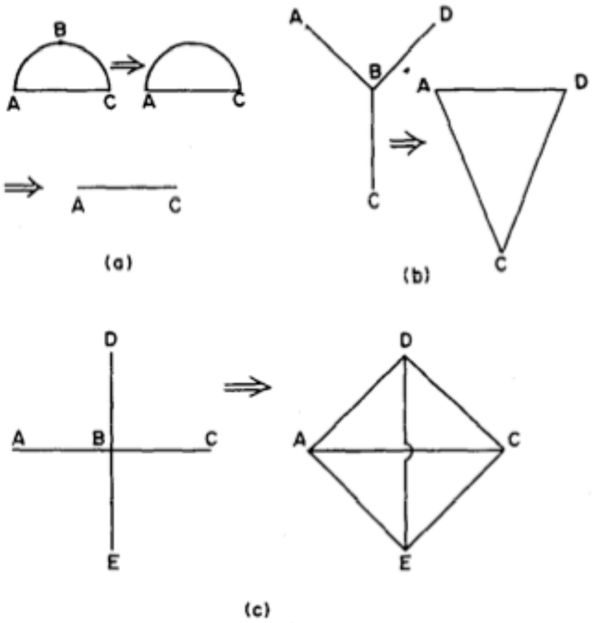

Fig. 5. Spider web transformations for : (a) $e_B = 2$, (b) $e_B = 3$ and (c) $e_B = 4$.

***Figure 3.1**: The contractions for nodes of degrees 2, 3, and 4 (termed "spider web" transformations) by van Vliet [Vli78, Figure 5].*

Concrete approaches exploiting hierarchy are reviewed in the forthcoming section on road networks (Section 3.2.3).

3.2.2 Graph Annotation Approaches

Algorithms exploiting the annotation approach are sometimes also termed *goal-directed search algorithms*. Additional information is attached to all/some vertices or edges of the graph. Based on this information, the search algorithm decides which part of the graph not to search.

A* Search

A* [Gel63, Sam63, KHI$^+$86, HNR68, Dor67, Gel77, Kor85] 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, at each iteration, the node with the shortest distance to the source is taken from the queue. In the original 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* (see for example [Del09, Algorithm 2, p. 22]). 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 the other nodes). In road networks for example, if the coordinates of the target are known, the Euclidean distance provides a good lower bound on the graph distance and thus a good potential function [SV86]. Using the Euclidean distance as a potential function for A* has been exploited and applied successfully. In general, however, the coordinates may not be known. A metric embedding or a drawing [WW05] can provide coordinates for a potential function.

Goldberg and Harrelson [GH05] (see also [GW05, GKW06, GKW07]) propose to use a set of landmarks $S \subseteq V$ and the triangle inequality (their method is sometimes called ALT (A*-Landmarks-Triangle inequality) for this reason). Their potential function is a "beacon-based" triangulation [KSW09]. Analogous to the distance oracle of Thorup and Zwick [TZ05], all nodes $v \in V$ know the distance to all landmarks $\mathcal{L} \in S$. For two nodes $u, v \in V$ and a landmark $\mathcal{L} \in S$, the triangle inequality yields that $d(u, v) \geqslant d(u, \mathcal{L}) - d(v, \mathcal{L})$. Taking the maximum difference over all $\mathcal{L} \in S$ 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 guarantee good coverage, meaning that some nodes are far from all landmarks. Several heuristics have been proposed to improve coverage [GH05, GW05], or to choose 'important' nodes [PBCG09]. The theory on beacon-based triangulations by Kleinberg et al. [KSW09] may help to explain for which graphs ALT works well and how many landmarks to select. For graphs with bounded doubling dimension, triangulations with respect to a constant number of landmarks yields $(1 + \epsilon, 0)$–approximate distances for a $(1 - \sigma)$–fraction of the node pairs (Kleinberg et al. also prove that this *slack* σ is necessary). While A* with landmarks [GH05] works for general graphs, it is thus expected to perform best on graphs with low doubling dimension. Poudel [Pou08] proposes a similar algorithm with a potential function based on an approximate distance oracle. With increasing quality of the distance approximations, fewer nodes are visited at query time.

A* is easy to implement and it yields decent speedups. In external memory setups, there appear to be better practical methods [EM01]. Better speedups can be obtained when combining A* with the bidirectional version of Dijkstra's algorithm. This is however not a 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 function [IHI^{+}94].

Querying using precomputed cluster distances [MSM09] is a somewhat similar approach. The network is partitioned into clusters and distances between all pairs 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

Reach-Based Routing [Gut04] is another modification of Dijkstra's algorithm. Each vertex is assigned a so-called *reach value* that determines whether a particular vertex will be considered during Dijkstra's algorithm. To have a high reach value, a vertex must lie on a shortest path that extends a long distance in both directions from the vertex (similar to Highway Hierarchies [SS05, SS06, NBB^{+}08], to be discussed in the next section). A vertex 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. Gutman [Gut04] reports fast query times with a speed up factor of 10 compared to Dijkstra's algorithm.

Arc Flags

In the preprocessing phase, the Arc Flag method [Lau04, KMS05, MSS$^+$06] partitions the graph into clusters and then, for each cluster, marks all edges where shortest paths towards nodes in the cluster start. At query time, edges that are not marked with the target cluster are ignored. A related approach uses geometric containers [WW03, WWZ05]. On its own, the preprocessing step of the Arc Flag approach is rather expensive. However, when applied within a hierarchy [MSS$^+$06] or when combined with other techniques, it can be very efficient [BD08, BDS$^+$08].

2–Hop Covers and Reachability Queries

For general directed graphs, the answers to shortest path and reachability queries (Definition 31) are harder to compute than for undirected graphs [AF90]. The scheme by Cohen et al. [CHKZ03] actually lies at the boundary between theory and practice. Cohen et al. focus on algorithms for directed graphs that occur in practice. They introduce a new technique, which they call *2–hop covers*: such a cover is a set of shortest paths $\mathcal{P}$ such that for every pair of vertices $(u, v) \in V \times V$ there is a shortest path between u and v that is the concatenation of two paths in $\mathcal{P}$. Based on this cover, they assign labels to vertices. The sizes of the labels thus depend directly on the size of the cover. It is known that graph classes with separators of size $\mathcal{O}(n^{\sigma})$ have 2–hop labels of size at most $\mathcal{O}(n^{1+\sigma} \lg n)$ [Coh96] (optimal labelings may be smaller). Unfortunately, finding a 2–hop cover of minimal size is an $\mathbf{NP}$–hard problem; the optimal cover can be approximated up to a logarithmic factor [CHKZ03, Theorem 4.2]. Fast practical computation of the labels has been proposed [CYL$^+$06, CYT06, CYL$^+$08, CY09]. Still, depending on the optimal cover size, the data structure may have size $\widetilde{\mathcal{O}}(n\sqrt{m})$ and queries may take $\widetilde{\mathcal{O}}(\sqrt{m})$ in the worst case. Wang et al. [WHY$^+$06] propose a different reachability labeling for (very) sparse graphs. They consider a graph as having two components: a spanning tree plus a set of a additional non-tree edges. For query time $\mathcal{O}(1)$ (reachability queries), their approach requires preprocessing time $\mathcal{O}(n + m + a^3)$ and space $\mathcal{O}(n + a^2)$. Note that $a = |E| - n + 1$, which may be $\Theta(n^2)$. Trißl and Leser [TL07] create a graph index called GRIPP. Preprocessing time and space complexities are $\mathcal{O}(m + n)$; the index supports efficient reachability queries in time $\mathcal{O}(m - n)$. They run a depth-first search from several root nodes such that each node obtains pairs of pre- and post-ordering rankings. A tree can then be queried for reachability by looking at the ranking only. Their approach outperforms the labelings by Cohen et al. [CHKZ03] and Wang et al. [WHY$^+$06] in practice (experiments are run for metabolic networks). Extensions to shortest path queries are planned. Jin et al. [JXRW08, JXRF09] propose *PathTree* (for sparse graphs) and 3–hop (for dense graphs) to further improve preprocessing and query times.

3.2.3 Road Networks

Route planning for transportation networks (road networks in particular) has been studied intensively [HP58, But68, EL82, IOAI91, SWN92, SKC93, CF94, LCL$^+$94, ZZ94, Liu95, HJR95, HJR96a, HJR96b, JHR96, AI97, FS97, HJR97a, HJR97b, SL97, SFG97, PS98, JHR98, CRS98, HJR00, AY00, AY01, CTB01, ZC01, JP02, AOPS02, BSWW04, KMS05, HSWW05, GW05, WWZ05, SS06, HLL06, MSS$^+$06, BFM$^+$07, BG07, KKK$^+$07, CL07, Hol08, KKR08, KKK$^+$08, KKRS08, HSW08, GSSD08, Sch08a, Sch08b, BDS$^+$08, DABC08, KH08, SS09, Del09]. The 9th DIMACS Implementation Challenge [DGJ08], which took place in 2006, stimulated a lot of research. For recent results, we refer to the survey on route planning [DSSW09], the survey on A*–based point-to-point shortest path queries [Gol07], the overview on engineering large network applications [Zar08], and the theses by Schultes [Sch08a] and Delling [Del09]. Route planning is also strongly related to efficient path query processing on spatial networks [PZMT03, GKR04, DABC08, SSA08, SSA09, SS09].

Approaches based on graph annotation can be applied as described in Section 3.2.2. Recent approaches exploiting hierarchy are specifically tailored for transportation networks. For an overview of recent hierarchical approaches see also [Del09, pp. 4–6] and the references therein. We give a brief overview in the following.

Highway Hierarchies (HH) [SS05, SS06, NBB$^+$08] are based on the observation that a certain class of edges (the 'highway' edges) tend to have greater representation among the portion of the shortest paths that are not in the vicinity of either the source or target. A recursive computation of these edges, paired with a contraction step, leads to a hierarchy of graphs that enables an impressive speedup at query time. Highway hierarchies were first proposed by Sanders and Schultes for undirected graphs [SS05], and later extended to directed graphs [SS06]. Nannicini et al. [NBB$^+$08] give a different approach for directed highway hierarchies. Their main focus is on time-dependent weights though. *Highway-Node Routing* (HNR) [SS07b] is a variant of HH that supports fast updates by additionally constructing overlay graphs.

The contraction step is an integral ingredient of the HH speedup technique. 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" [Vli78] and Hu's "distance equivalent networks" [Hu69]). This observation can be generalized [GSSD08]: 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. The method called *Contraction Hierarchies* [GSSD08, BDS$^+$08] uses intelligent heuristics to contract nodes in the right order.[4] This order yields a hierarchy, with which the query algorithm can efficiently find shortest paths. Contraction-based techniques perform very well in practice. The preprocessing step is particularly efficient.

Transit-Node Routing (TNR) [BFSS07, BFM$^+$07, Sch08b] is based on the following observation: When driving somewhere far away, drivers usually leave their current location via one of only a few access routes to a relatively small set of transit nodes. These transit nodes are then interconnected by a sparse network relevant for long-distance travel. The TNR method precomputes all shortest paths to transit nodes and all shortest paths among transit nodes. The preprocessing is expensive but the query time is extremely fast.

Combining graph annotation and hierarchical approaches often yields powerful methods. Several combinations have been investigated and evaluated empirically [HSWW05, BDS$^+$08]. Two particularly strong combinations are CHASE [BDS$^+$08], which combines contraction hierarchies and arc flags, and SHARC [BD08], which combines shortcuts and arc flags. Shortcut edges are additional edges that maintain the original distance but decrease the number of hops (the number of edges on the shortest path, Definition 8). The problem of finding the best k shortcuts has been formalized and it is apparently hard (see [BDDW09, Theorems 1 and 2]). Nevertheless, SHARC works very well in practice.

For an overview of the preprocessing time vs. query time tradeoff for some of these methods see Figure 6.5; for experimental results, see Tables 6.4 and 6.1 (all in Chapter 6).

The performance of these hierarchy-based methods is really good in practice, however, complexity results are mostly experimental only.[5] Recently, Abraham et al. [AFGW10] show that

[4] Van Vliet [Vli78] contracts nodes up to degree 4; his contractions of nodes with higher degree did not yield any speedup but a slowdown. The contraction order appears to be very important.

[5] Exactness and correctness are proven.

if a graph has low *highway dimension*, algorithms based on reach [Gut04], contraction hierarchies [GSSD08, BDS$^+$08], highway hierarchies [SS05, SS06, NBB$^+$08], transit nodes [Sch08b], and SHARC [BD08] have provable efficiency guarantees.

Some of the methods that were tailored for road networks also perform well for timetable information systems [Sch05b, MHSWZ04, PSWZ07, Jac08, BDW09] (for details we refer to the thesis by Schulz [Sch05b]) and for grids and unit-disk graphs [Del09, Tables 4.21 & 4.22]. For higher-dimensional graphs, however, the preprocessing time and the space consumption increase significantly [BDS$^+$08, Tables 10 and 11].[6]

3.2.4 Complex Networks

Although very efficient when applied to road networks, the hierarchical techniques outlined in the previous section seem to have problems handling graphs with higher node degrees. For complex networks, other techniques and heuristics have been suggested and evaluated.

Rattigan et al. [RMJ06] approximate distances in graphs using a *structure index*. Their algorithm grows *zones* using random exploration starting from random seeds. When combining their zones with landmarks, the algorithm computes the distances between each node and zone in time $\mathcal{O}(ms)$ for s zones. They obtain approximations of the closeness and betweenness centrality for nodes. The structure indices also help finding a graph clustering [RMJ07].

Potamias et al. [PBCG09] use landmark-based A*. Their strategies for landmark selection outperform random landmark selection strategies. Their experimental evaluation includes social networks. The networks they consider do not seem to have low doubling dimension. To the best of our knowledge, the theoretical performance of beacon-based triangulations [KSW09] for complex networks has not been investigated yet. Metric embeddings of power-law graphs seem to require higher dimensional spaces.

Brady and Cowen [BC06] propose a compact routing scheme for unweighted power-law graphs. Since the scheme is based on tree routing, it can be extended to efficient distance and shortest path queries. Their distance oracle has size $\mathcal{O}(n\mathcal{E}\lg^2 n)$ and it returns $(1,\mathcal{D})$–approximate distances in time $\widetilde{\mathcal{O}}(1)$ for a parameter $\mathcal{D}$ and a value $\mathcal{E}$ that depends on $\mathcal{D}$ and the graph. There are no theoretical bounds on $\mathcal{D}$ and $\mathcal{E}$. The scheme roughly works as follows: we grow a shortest path tree from the node with the highest degree. All nodes up to distance $\mathcal{D}/2$ form the *core* with diameter $\mathcal{D}$. The remaining nodes form the *fringe*. The fringe is claimed to be almost a forest. $\mathcal{E}$ denotes the number of edges we must remove such that the fringe actually becomes a forest. For each of these edges, an additional routing tree is produced. The scheme routes through the node with the highest degree with additive stretch at most $\mathcal{D}$ or optimally using one of the fringe-trees. Intuitively, small values of $\mathcal{D}$ imply large values of $\mathcal{E}$. Experiments using random power-law graphs [ACL00, CL02] indicate that both $\mathcal{D}$ and $\mathcal{E}$ can be chosen to be small simultaneously. $(1,\mathcal{O}(1))$–approximate distance oracles with space $\widetilde{\mathcal{O}}(n)$ may be possible for power-law graphs, although there is no proof.

Das Sarma et al. [SGNP10] provide a practical implementation of Bourgain's embedding (Theorem 3), and they propose an extension of the distance oracle by Thorup and Zwick [TZ05]. In their extension, they omit ball computations. While the asymptotic performance is not affected, their algorithms both for preprocessing and query are simpler and potentially faster in practice than the corresponding original algorithms. The stretch bounds, however, only hold with high probability.

[6] Bauer et al. [BDS$^+$08, p. 22 of TR] observe that *most speed-up techniques have problems when the average node degree becomes too large*.

Xiao et al. [XWP$^+$09] compress graphs by exploiting symmetries. Instead of treating vertices as a single unit, they work on *orbits* of *automorphism groups*. Shortest path queries are answered using *compact BFS-trees*, which are based on these orbits. Symmetries in complex networks seem to be very common. Experiments show that their method may be very efficient. Scalability cannot be assessed appropriately without having the code. The largest graph they consider (the Internet Autonomous System network) has 22,442 nodes and 45,550 edges. For this graph, their preprocessing algorithm runs in roughly 347 seconds [XWP$^+$09, Table 3]. No speedup for queries is reported. The third-largest graph under consideration (a social network of Erdős collaborators) has 6,927 nodes and 11,850 edges. For this graph, preprocessing takes roughly 29 seconds [XWP$^+$09, Table 3]. They report a speedup for shortest path queries of 57.72 [XWP$^+$09, Table 4]. Based on these two graphs, the running time of the preprocessing algorithm appears to be roughly quadratic in the number of nodes. Note, however, that the running time of their algorithm mainly depends on the symmetries, which may be the main cause for the difference in the running times (also for the two graphs considered here).

Goldman et al. [GSVGM98] consider relationships among objects in large databases. Their method processes keyword searches over databases in interactive query sessions. Distances between objects are computed based on a compact index, which consists of local neighborhoods and distances to *hub* vertices (separators). Hubs are chosen as high-degree nodes. They evaluate the performance using the Internet Movie Data Base (IMDB), which arguably has a power-law degree sequence. In database systems, distance estimation is also related to computing the transitive closure of relations [DR94, UY90, Jag90, HWYY05, TL07] and thus related to reachability query processing.

Cheng and Yu [CY09] use 2–hop labels [CHKZ03] to efficiently compute exact distances. For the DBLP graph with 52,682 nodes and 59,395 edges, preprocessing takes 20 seconds [CY09, Table 1]. At query time, their method outperforms Dijkstra's algorithm by two orders of magnitude [CY09, Figure 17 and Section 7.4].

The GRIPP index by Trißl and Leser [TL07] efficiently answers reachability queries for scale-free networks (evaluation on metabolic networks). Distance queries are planned as an extension.

3.3 Summary

Shortest path query processing in graphs has been studied extensively both in theory and in practice. 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 to dynamic scenarios. For complex networks, methods have been proposed only recently; their efficiency and optimality is still under investigation.

Theoretical research on distance oracles for general graphs is 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. Also, distance oracles for directed graphs of restricted classes are mostly unknown territory.

ningen banji saiou ga uma
(often what at first appears to be bad turns out to be good)
Japanese proverb

4 Lower Bounds for Sparse Graphs

In this chapter, we investigate the tradeoff between the space complexity, the query time, and the stretch of approximate distance oracles. The main result is a lower bound on the minimum space consumption of distance oracles with query time $\mathtt{t}$ and stretch $(\alpha, 0)$. This space lower bound holds even for sparse ($polylog(n)$–degree) graphs. The bound is proven using techniques based on Pătraşcu's [Pat08a] recent communication lower bounds on communication protocols for the LopsidedSetDisjointness (LSD) problem and the space lower bounds for data structures obtained by reductions to LSD.

Thorup and Zwick [TZ05] prove that, for some integer values of the stretch parameter $k \geqslant 1$, any distance oracle with multiplicative stretch less than $2k+1$, needs space at least $\Omega(n^{1+1/k})$ bits (Theorem 7, connected to Erdős' girth conjecture [Erd64, ES63]). Their proof holds even if infinite query time is allowed. For sparse graphs, the best bound it proves is that the size of the data structure is at least proportional to the number of edges in the graph. An exact distance oracle with space complexity $\mathcal{O}(m)$ and query time $\mathcal{O}(1)$ would still be possible. For sparse graphs, such a distance oracle would be very useful. Unfortunately, our bound implies that such an oracle does not exist. Even if a small stretch is allowed, linear space and constant query time is impossible.

4.1 Context

We prove a lower bound for approximate distance oracles in the cell-probe model (Definition 25). The main result of this chapter is a three-way tradeoff between space, stretch and query time.

Theorem 8. *There exists an integer n_* such that for all $n \geq n_*$, the following holds. Let $\mathcal{S} = \mathcal{S}(n)$, $\mathtt{t} = \mathtt{t}(n) \leqslant \lg n$, $w = w(n)$, and $\alpha = \alpha(n) = o\left(\frac{\lg n}{\lg(wn)}\right)$ be integers such that there exists an $(\alpha, 0)$–approximate distance oracle with query time $\mathtt{t}$ in the cell-probe model with word-length w for any graph with n vertices and maximum degree at least poly $\left(\frac{\mathtt{t}w\alpha}{\lg n}\right)$. Then, the space complexity of this distance oracle is at least*

$$\mathcal{S} \geqslant n^{1+\Omega\left(\frac{1}{\alpha\mathtt{t}}\right)} / \lg n.$$

In the lower bound by Thorup and Zwick (Theorem 7), dense graphs with large girth (and their subgraphs) are the worst-case instances. In our proof, r–regular graphs with large girth (and their subgraphs) are the worst-case instances. For graphs with high regularity, devising a distance oracle should intuitively be easy. For example, the distance between two nodes in the hypercube is equal to the Hamming distance between the corresponding node labels. Short labelings may be possible for various regular graphs.

For an r–regular graph $G = (V, E)$ with diameter $\Delta = \mathrm{diam}(G) = \mathcal{O}(\lg_r n)$ and for $\alpha \geqslant \Delta/\mathrm{g}(G)$, $(\alpha, 0)$–approximate distance oracles can be devised in a straightforward way. For each node $u \in V$, compute the ball with radius Δ/α, denoted by $B_G^{\Delta/\alpha}(u)$ (which contains $\mathcal{O}(r^{\Delta/\alpha}) \leqslant n^{\mathcal{O}(1/\alpha)}$ nodes). For a distance query $d(u, v)$, if $v \in B_G^{\Delta/\alpha}(u)$, then return the exact distance; otherwise return Δ. This yields total space (and preprocessing time) $n^{1+\mathcal{O}(1/\alpha)}$ for an $(\alpha, 0)$–approximate distance oracle with constant query time.

We prove that this amount of space is necessary (up to constant factors in the exponent) for distance oracles with constant query time that can preprocess r–regular graphs and all their subgraphs.[1] It is impossible to prove a lower bound that holds for a particular graph G. This is due to the fact that the algorithm can hard-code G and its distance table in order to answer queries in constant time (without accessing the data structure). In this proof, we will refer to a "worst-case instance" for a distance oracle as a *base-graph* G, meaning that the distance oracle must accept at least the class $\mathcal{G}$ consisting of G and all its subgraphs. The space lower bound by Thorup and Zwick (Theorem 7) is also proven with respect to a base-graph and all its subgraphs.

4.2 Preliminaries

In the cell-probe model [Yao81, Mil99] (Definition 25), a cell has w bits and the *space* of a data structure is measured as the number of cells it occupies, denoted by $\mathcal{S}$. The query time is measured by the worst-case number of cells that a query reads. All computations based on cells that have been read are free. The most typical values of the cell size (also called word length) are $w = \lg n$ or $w = polylog(n)$, but larger (or smaller) values may be interesting as well.

A class of graphs $G = (V, E)$ is considered *sparse* if $|E| = \widetilde{\mathcal{O}}(|V|)$. We may sometimes deem a graph to be sparse if $|E| \leqslant n \cdot poly(w, \lg n)$.

4.2.1 Communication Complexity

Our proof uses a reduction from a distance oracle to a communication protocol. This proof technique [Ajt88, Mil94, KW90, MNSW98, AIP06, Pat08a, Pat08b] has been used for reductions from various data structures (the cell-probe interactions with the data structure, to be precise) to communication protocols.

Communication complexity [Yao79, AB07] is the problem of two separated parties, Alice and Bob, both holding an input $x \in \{0, 1\}^n$ and $y \in \{0, 1\}^n$, respectively, computing the result $\phi(x, y)$ of a function $\phi : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$. Before Alice and Bob receive their inputs, they agree on a communication protocol using which they will compute $\phi(x, y)$. It is assumed that both Alice and Bob have infinite computing resources. The communication complexity of the protocol is the number of bits sent in total. The communication complexity of computing ϕ is equal to the communication complexity of the best-possible protocol.

For our proof, an intuitive understanding of a communication protocol as a sequence of messages between Alice and Bob should suffice. A more rigorous definition is the following.

Definition 32 ((Symmetric) Communication Protocol [AB07])**.** *A (symmetric)* t–round communication protocol *for a function $\phi : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ is a sequence of function pairs $(S_1, C_1), (S_2, C_2), \ldots, (S_{\mathrm{t}}, C_{\mathrm{t}}), (\phi_1, \phi_2)$. The input of S_i is the communication pattern of the first*

[1] As a comparison: it is known that a metric embedding into ℓ_2 of the shortest path metric on r–regular graphs with girth g requires $(\Omega(\sqrt{g}), 0)$ distortion [LMN02]. This means that, for constant r and girth $\mathcal{O}(\lg_r n)$, an embedding-based distance oracle with space $\widetilde{\mathcal{O}}(n)$ has stretch $(\Omega(\sqrt{\lg n}), 0)$.

$i-1$ rounds and the output is from $\{1,2\}$, indicating which player will communicate in the i–th round. The input of C_i is the input string of this selected player as well as the communication pattern of the first $i-1$ rounds. The output of C_i is the bit that this player will communicate in the ith round. Finally, ϕ_1 and ϕ_2 are $0/1$–valued functions that the players apply at the end of the protocol to their inputs as well as the communication pattern in the t *rounds in order to compute the output. These two outputs must be $\phi(x,y)$. The* communication complexity *of ϕ is*

$$C(\phi) = \min_{\text{protocols } CP} \max_{x,y} \{\text{Number of bits exchanged by } CP \text{ on } x,y\}\,.$$

A trivial protocol is to communicate the entire input of one player, compute $\phi(x,y)$ at the other player, and send back the result. This yields $C(\phi) \leqslant n+1$. The objective is to communicate (significantly) less.

Lopsided Set Disjointness

In communication complexity [Yao79, KN96, AB07], SETDISJOINTNESS is the problem of two separated agents deciding whether two sets are disjoint. In the asymmetric version of the problem, called LOPSIDEDSETDISJOINTNESS (LSD), Alice and Bob receive sets S_{Alice} and S_{Bob}, respectively. Their goal is to determine whether $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} = \emptyset$ using a communication protocol (Figure 4.1). More precisely (as in Definition 32), the function ϕ is 1 if $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} = \emptyset$ and 0 otherwise. LOPSIDEDSETDISJOINTNESS has two parameters, N and B, known to both Alice and Bob. The "universe" has size NB and both Alice and Bob are given a subset of this universe; $S_{\mathsf{Alice}} \subseteq [NB]$ and $S_{\mathsf{Bob}} \subseteq [NB]$. Alice's set has size $|S_{\mathsf{Alice}}| = N$. (Alice is given one of $\binom{NB}{N}$ different sets.) B thus denotes the fraction between N and the size of the universe NB. The size of Bob's set is not fixed; it may be $\Theta(NB)$. Two trivial protocols are the following: (1) Alice communicates her set S_{Alice} with a message of length $\lceil \lg \binom{NB}{N} \rceil = \mathcal{O}(N \lg B)$ bits; Bob can then compute $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}}$ and reply with 1 bit. Alternatively, (2) Bob communicates his set S_{Bob} with a message of length $\lceil \lg |S_{\mathsf{Bob}}| \rceil + \lceil \lg \binom{NB}{|S_{\mathsf{Bob}}|} \rceil$ (encoding $|S_{\mathsf{Bob}}|$ and S_{Bob}); Alice computes $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}}$ and sends back 1 bit. Either Alice or Bob communicates his/her complete set. The question is how much communication Alice and Bob *need* to decide whether $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} = \emptyset$. A trivial *randomized* protocol is the following: Alice tosses a coin; based on the result, she decides whether $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}}$ and sends 1 bit to Bob. Let this protocol be denoted by CP. The probability

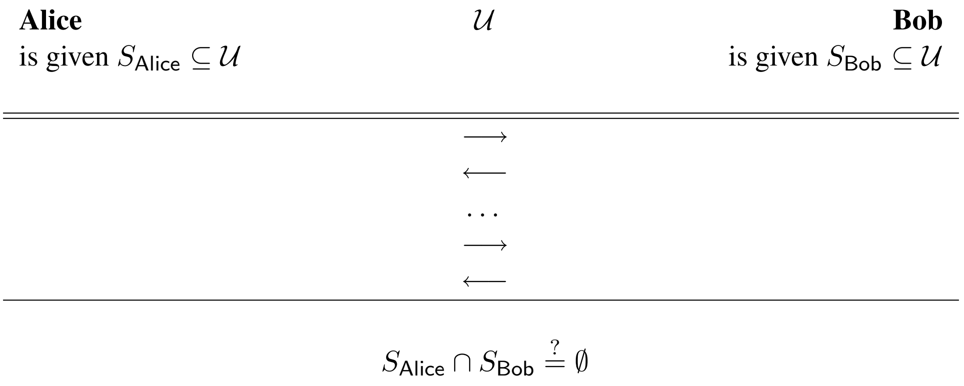
Alice
is given $S_{\mathsf{Alice}} \subseteq \mathcal{U}$
$\mathcal{U}$
Bob
is given $S_{\mathsf{Bob}} \subseteq \mathcal{U}$
$\longrightarrow$
$\longleftarrow$
...
$\longrightarrow$
$\longleftarrow$
$S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} \overset{?}{=} \emptyset$

***Figure 4.1**: The* LOPSIDEDSETDISJOINTNESS *communication problem*

that the result is correct is $\Pr[CP(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) = \phi(S_{\mathsf{Alice}}, S_{\mathsf{Bob}})] = 1/2$. Alice could also send a random sample, based on which Bob must decide whether $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}}$. A randomized protocol CP' is said to have have *two-sided error* if there is a non-zero probability that Alice and Bob err when they output either 0 or 1, and it is said to have *one-sided error* if the probability that Alice and Bob err is zero for at least one of the possible outcomes 0 or 1. For example, CP' has one-sided error if $\phi(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) = 0 \Rightarrow \Pr[CP'(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) = 0] = 1$ [MR95]. A protocol CP' is said to have *bounded error* if there are two constants $\psi, \psi' > 1/2$ such that $\phi(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) = 0 \Rightarrow \Pr[CP'(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) = 0] \geqslant \psi$ and $\phi(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) = 1 \Rightarrow \Pr[CP'(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) = 1] \geqslant \psi'$.

The communication complexity to solve LSD with a one-sided error is known to be bounded from below (Miltersen et al. [MNSW98]). Note that a lower bound on communication protocols with one-sided error is easier than a bound on protocols with two-sided error, since one-sided error protocols are required to have a “better” performance than two-sided error protocols.

Lemma 9 (Miltersen et al. [MNSW98])**.** *There exists some constant $\mathsf{C} > 0$ such that in a one-sided error protocol for LSD, either Alice sends $\mathsf{C}N \lg B$ bits or Bob sends NB^{C} bits.*

Andoni et al. [AIP06] extend the bound to include protocols with two-sided error as well; Pătraşcu [Pat08a, Pat09, Pat08c] proves the following (alternatively, one could use Pătraşcu’s approach with [Pat08b, Theorem 5.15] and [Pat08b, Chapter 5.4.3]).

Lemma 10 (Pătraşcu [Pat09, Theorem 1.4])**.** *There exists some constant $\mathsf{C} > 0$ such that in a bounded-error protocol for LSD, either Alice sends $\mathsf{C}N \lg B$ bits or Bob sends NB^{C} bits.*

This roughly means that, for Alice and Bob to know whether $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}}$ with probability bounded away from $1/2$, either Alice or Bob must send almost their complete set. (For the communication complexity of symmetric set disjointness, see [KS92, Raz92, BYJKS04, HW07].)

Data Structure Lower Bounds based on LSD

One key part in Pătraşcu’s results [Pat08a] is the reduction from LopsidedSetDisjointness (LSD) to reachability oracles. Recall (Definition 31) that the reachability query problem is, given a (sparse) *directed* graph $G = (V, E)$, to construct a data structure using less than n^2 space such that reachability queries (deciding whether there is a directed path from u to v) can be answered efficiently. Reachability oracles for *undirected* graphs are trivial: we compute the connected components and store a component number for each node. Storing reachability information for directed graphs appears to be hard [AF90] and so is the reachability query problem.

Recall Pătraşcu’s theorem on reachability oracles (Theorem 5), which we restate here.

Theorem (Pătraşcu [Pat08a, Theorem 2])**.** *A reachability oracle using space $\mathcal{S}$ in the cell-probe model with w–bit cells, requires query time $\mathsf{t} = \Omega(\lg n / \lg \frac{\mathcal{S}w}{n})$.*

Pătraşcu’s proof is a reduction from a variant of LSD to the problem of reachability queries in a *butterfly graph* and its subgraphs.

Definition 33. *A* butterfly graph Butterfly(ℓ, r) *is a directed graph $F^{\ell,r} = (V, A)$ specified by two parameters: the depth ℓ and the degree r.*

- *$F^{\ell,r}$ has $\ell + 1$ layers $V_0, V_1, \ldots V_\ell$ with r^ℓ vertices each.*
  The nodeset is $[\ell + 1] \times [r]^\ell$.
- *Every vertex $v \in V \setminus V_\ell$ has out-degree r.*
  Every vertex $v \in V \setminus V_0$ has in-degree r.

- *Arcs $a \in A \subseteq \bigcup_{i=0}^{\ell} V_i \times V_{i+1}$ only connect nodes in adjacent levels V_i, V_{i+1}.*
  Two nodes $v \in V_i$ and $v' \in V_{i+1}$ are adjacent if they differ only at coordinate i. Node $(i, c_1, c_2, \ldots c_i, \ldots c_\ell)$ is connected to all nodes $(i+1, c_1, c_2, \ldots c_i', \ldots c_\ell)$ for $c_i' \in [r]$.

Note that paths between any $s \in V_0$ and $t \in V_\ell$ are unique. For an illustration of *undirected* butterfly graphs see Figure 4.2 for 3 layers with degree 2 and Figure 4.3 for 3 layers with degree 3.

The reduction to LSD is based on the following idea: the universe of LSD is mapped to the edgeset of the butterfly graph $G = (V, E)$ using a bijection $f : [NB] \leftrightarrow E$. The input sets of Alice and Bob can be mapped to subsets of the edges $f(S_{\mathsf{Alice}}), f(S_{\mathsf{Bob}}) \subseteq E$, respectively.[2] Let us assume that the set of Alice consists of a set of κ paths of length ℓ between nodes in V_0 and V_ℓ. Let $(e_1, e_2, \ldots e_\ell)$ denote such a path between $u \in V_0$ and $v \in V_\ell$. Bob simulates the reachability data structure for the subgraph of the butterfly graph with "his" edges removed: $G^- = (V, E \setminus f(S_{\mathsf{Bob}}))$. Alice can find out whether at least one of her ℓ edges e_i is in Bob's set $f(S_{\mathsf{Bob}})$ by asking one reachability query $\mathsf{reachable}(u, v)$. Alice does so by sending a message in a communication protocol (technically speaking, she encodes the position of the word in the data structure she wants to read as in the cell-probe model). By asking all queries in parallel [PT06, Pat08a], Alice learns whether at least one of her $\kappa\ell$ edges is in Bob's set $f(S_{\mathsf{Bob}})$ — Alice thus also knows whether $f(S_{\mathsf{Alice}}) \cap f(S_{\mathsf{Bob}}) \Leftrightarrow S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}}$. It turns out that, if the data structure is small, the communication complexity of this protocol is very low, which contradicts the communication lower bound of LSD.

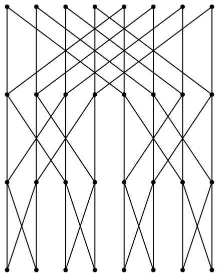

***Figure 4.2**: A drawing of an undirected butterfly graph with degree 2 spanning 3 layers.*

Recall that the lower bound for reachability oracles directly implies a lower bound for distance oracles on directed graphs (Section 3.1.1, Definition 31). In a straightforward way, it also implies a lower bound for distance oracles on undirected graphs. The following direct reduction from reachability oracles for subgraphs of the butterfly graph to distance oracles with less than stretch $(1, 2)$ yields the same space lower bound for the latter. The lemma is a minor result of this chapter; it serves as a warm-up and as an illustration of Pătraşcu's reduction technique.

Lemma 11. *In the cell-probe model with w–bit cells, a distance oracle with* **additive** *stretch less than 2 using space S requires query time $\mathsf{t} = \Omega(\lg n / \lg \frac{Sw}{n})$.*

[2] We abuse notation by simplifying $f(E') := \bigcup_{e \in E'} \{f(e)\}$.

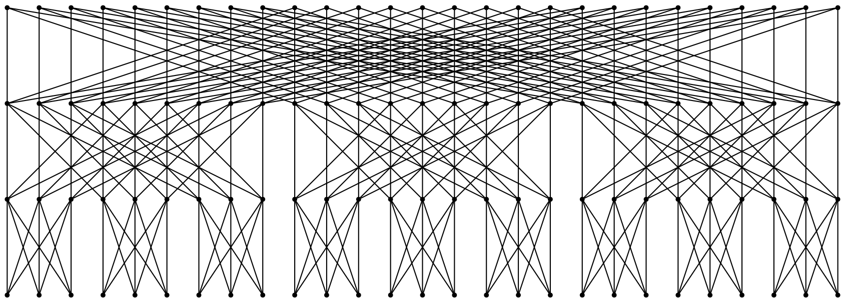

***Figure 4.3**: A drawing of an undirected butterfly graph with degree 3 spanning 3 layers.*

Proof. We give a straightforward reduction from reachability oracles for the directed butterfly graph F in Pătraşcu's reduction [Pat08a, Reduction 13] (see also [Pat08b, Reduction 7.11]) to a distance oracle for the same graph, interpreted as undirected, say F'. If node v is reachable from u in F, the distance in F' is equal to ℓ; if v is not reachable from u in F, the distance in F' must be at least $\ell + 2$ since the butterfly graph F' and its subgraphs are bipartite. Therefore, if there is an algorithm that approximates distances with additive stretch less than 2 using $\mathtt{t}$ probes in a data structure of size $\mathcal{S}$, then there is an algorithm — using the same $\mathtt{t}$ probes in the same data structure of size $\mathcal{S}$ — that answers reachability queries by distinguishing distances ℓ and distances at least $\ell + 2$. □

Corollary 12. *In the cell-probe model with w–bit cells, a distance oracle with* **additive** *stretch less than 2 and query time* $\mathtt{t}$ *requires space* $\mathcal{S}$ *at least*

$$\mathcal{S} \quad \geqslant \quad n^{1+\Omega(1/\mathtt{t})}/w.$$

Note that, for $w = 1$, Corollary 12 yields a lower bound in the bit-probe model [MP69].

The remainder of this chapter is devoted to amplifying the construction to obtain a lower bound for multiplicative stretch α. Instead of the butterfly graph, the worst-case instance is a regular graph with large girth.

4.2.2 Regular Graphs with Large Girth

In our space lower bound, we need graphs that contain many disjoint shortest paths of length ℓ. The following is the definition of the set of sets of vertex-disjoint paths. It is used extensively in the proof.

Definition 34. *For a graph $G = (V, E)$ and two positive integers κ, ℓ, let $\mathcal{P}^{\ell,\kappa}(G)$ denote the set of sets of edges defined as follows. The members of $\mathcal{P}^{\ell,\kappa}(G)$ are all possible sets $P \subseteq E$, where P can be expressed as the union of κ vertex-disjoint paths in G, each of length exactly ℓ.*

The proof also requires that these paths $(v_0, v_1, \ldots, v_\ell)$ are shortest paths chosen such that there is no short alternative path connecting v_0 and v_ℓ. Paths do have this property if the girth of the graph is sufficiently large.

Graph Requirements

Our bounds require reductions from sparse graphs with large $\mathcal{P}^{\ell,\kappa}(G)$. To ensure that $\mathcal{P}^{\ell,\kappa}(G)$ is sufficiently large, we require certain properties of the graph. Based on good *expansion* $\lambda(G)$ of a graph G, one can show that there are many disjoint paths in G [AC07]. Based on this, we prove that $\mathcal{P}^{\ell,\kappa}(G)$ is large (Lemma 21).

Definition 35 ((normalized) second-largest eigenvalue [Alo86, AFWZ95])**.** *Let $G = (V, E)$ be an r–regular graph with n vertices. Let A be an adjacency matrix of G. Let $B := A/r$, that is $b_{i,j} = 1/r$ if $(v_i, v_j) \in E$ and $b_{i,j} = 0$ otherwise. Let $\lambda_0 \geqslant \lambda_1 \geqslant \ldots \geqslant \lambda_{n-1}$ denote the eigenvalues of B. Let $\lambda(G)$ denote the second-largest absolute value of an eigenvalue: $|\lambda_0|, |\lambda_1|, \ldots |\lambda_{n-1}|$. $\lambda(G)$ is called the* (normalized) second-largest eigenvalue *of G. $\lambda(G)$ is also called the* expansion *of G. G is called* Ramanujan *if $\lambda(G) \leqslant \frac{2\sqrt{r-1}}{r}$.*

It is known that each $|\lambda_i|$ is real number in the range $[0, 1]$. It is also known that $\lambda_0 = 1$ and that $\lambda_1 \geqslant 0$. Therefore, $\lambda(G) := \max\{\lambda_1, |\lambda_{n-1}|\}$.

At one point in the proof, we rely on the expansion property of a graph. We use the following theorem. Based on the expansion $\lambda(G)$, Alon et al. [AFWZ95] prove a lower bound on the probability that a random walk of length ℓ stays inside a set of a certain density.

Theorem 13 (Alon et al. [AFWZ95, Theorem 4.2])**.** *For a graph $G = (V, E)$ with $\lambda := \lambda(G)$, the probability that a random walk of ℓ steps from a uniformly random starting vertex stays inside $U \subseteq V$, where $|U| \geqslant \rho n \geqslant 6\lambda n$, is at least $\rho(\rho - 2\lambda)^{\ell}$.*

The theorem implies the following corollary by setting $\rho = 0.9$.

Corollary 14. *For a graph $G = (V, E)$ with $\lambda(G) \leqslant 0.1$, the probability that a random walk of ℓ steps from a uniformly random starting vertex stays inside $U \subseteq V$, where $|U| \geqslant \frac{9}{10}|V|$, is at least $\frac{1}{2^{\ell}}$.*

Deep knowledge of expansion is not necessary to understand our proof in this chapter. For more information on expander graphs, we refer to [Alo86, AFWZ95, HLW06].

Graph Construction

Our proof relies on the existence of graphs with large girth and large $\mathcal{P}^{\ell,\kappa}(G)$; Ramanujan graphs[3] (construction by Lubotzky et al. [LPS88] using the Cayley graph of a projective general linear group) are what we use.[4] For the connection to dense graphs with large girth, see also [EJ08, Construction IV].

Lubotzky et al. [LPS88, pp. 262–263] prove the following.[5] Recall that the Legendre symbol for two unequal primes $p \neq q$ is defined as follows:

$$(p|q) = \begin{cases} +1 & \text{if there is an integer } x \text{ such that } p \equiv_q x^2 \\ -1 & \text{otherwise} \end{cases}$$

[3] Ramanujan graphs are named after the Indian mathematician Srinivasa Iyengar Ramanujan (1887–1920).

[4] In our proof, we apply Theorem 13 by Alon et al. with the condition that $\lambda(G) \leqslant 0.1$. The construction by Lubotzky et al. is not the only one that guarantees regular expanders with large girth. The construction by Morgenstern [Mor94] may potentially work as well. It has been shown that random regular graphs also have large expansion [Fri91, FKS89]. Random regular graphs with large girth may thus potentially work as well.

[5] Note that, in their paper, Lubotzky et al. do not normalize λ. The statement in Theorem 15 uses normalized eigenvalues as defined in Definition 35.

Theorem 15 (Lubotzky et al. [LPS88, pp. 262–263])**.** *Let p and q be unequal primes congruent to $1 \mod 4$. There exists a $(p+1)$–regular graph $X^{p,q}$ with the following properties:*

- *Case* $(p|q) = +1$
  1. *$X^{p,q}$ has $\frac{q(q^2-1)}{2}$ nodes.*
  2. $\mathsf{g}(X^{p,q}) \geqslant 2\lg_p q$
  3. $\operatorname{diam}(X^{p,q}) \leqslant 2\lg_p n + 2\lg_p 2 + 1$
  4. $\lambda(X^{p,q}) \leqslant \frac{2\sqrt{p}}{p+1}$
- *Case* $(p|q) = -1$
  1. *$X^{p,q}$ has $q(q^2-1)$ nodes.*
  2. $\mathsf{g}(X^{p,q}) \geqslant 4\lg_p q - \lg_p 4$
  3. $\operatorname{diam}(X^{p,q}) \leqslant 2\lg_p n + 2\lg_p 2 + 1$
  4. $\lambda(X^{p,q}) \leqslant \frac{2\sqrt{p}}{p+1}$

In the following we prove the existence of Ramanujan graphs for infinitely many and various choices of large r and n. The theorem is mostly implied by the result of Lubotzky et al. [LPS88] stated in Theorem 15.

Lemma 16. *For every sufficiently large n_0, r_0 with $n_0 > 8r_0^3$, there exists a graph $G = (V, E)$ with the following properties:*

1. *$|V| = n$ with $\frac{n_0}{2} \leqslant n \leqslant 9n_0$*
2. *G is r–regular, where $r_0 \leqslant r \leqslant 2r_0$*
3. *The girth of G is at least $\mathsf{g}(G) \geqslant \frac{1}{2}\lg_r n$*
4. $\lambda(G) \leqslant \frac{2\sqrt{r-1}}{r}$

Proof. The graph claimed to exist is a Ramanujan graph $X^{p,q}$ as in Theorem 15. The construction by Lubotzky et al. [LPS88, pp. 262–263] requires unequal primes $p \neq q$, both congruent to $1 \mod 4$. The graph has at least $n = \frac{q(q^2-1)}{2}$ nodes and it is $r = (p+1)$–regular.

In the following, we prove the existence of primes p, q such that r and n lie in the ranges $r \in [r_0, 2r_0]$ and $n \in [\frac{n_0}{2}, 9n_0]$.

The Bertrand-Chebyshev theorem[6] states that for every $x > 1$ there is always at least one prime p such that $x < p < 2x$. This generalizes to certain arithmetic progressions [Bre32, Erd35, Bre64, Mor93]. Let $z \in \mathbb{N}^0$. Breusch [Bre32, p. 505] states

> [...] daß für $x \geqslant 7$ zwischen x und $2x$ stets Primzahlen einer jeden der vier Progressionen $3z+1$, $3z+2$, $4z+1$, $4z+3$ liegen.

[6] Named after Joseph Louis François Bertrand (1822–1900), who conjectured the existence of primes between x and $2x$ in 1845, and Pafnuty Lvovich Chebyshev (1821–1894), who proved the conjecture in 1850. Ramanujan gave a simpler proof in 1919. It's a small world!

that for every $x \geqslant 7$ there is a prime of the form $p = 4z + 1$ in the interval between x and $2x$. Note that the modular congruence is satisfied, $p \equiv_4 1$.

Due to Breusch [Bre32, p. 505], there exists a prime $p \in [r_0 - 1, 2r_0 - 2]$ of the form $p = 4z + 1$. Let p denote this prime.

The Lubotzky et al. construction of the Ramanujan graph requires a different prime $q \neq p$ of the form $q = 4z + 1$. Again, due to Breusch [Bre32, p. 505], there exists such a prime $q \in [x, 2x]$ if the intervals for p and q do not overlap. Let $x := \lceil n_0^{1/3} \rceil + 1 \in [n_0^{1/3} + 1, n_0^{1/3} + 2)$. The imposed condition $n_0 > 8r_0^3$ ensures that the two intervals do not overlap. Thus, there is a different prime in the interval $[x, 2x]$ for every integer $x \geqslant 2r_0 - 1$. Let q denote this prime. The Ramanujan graph has either $q(q^2 - 1) = q^3 - q$ or $\frac{q(q^2-1)}{2} = \frac{q^3-q}{2}$ nodes. For the number of nodes n, we have that $n \in [\frac{x^3-x}{2}, 8x^3 - 2x]$. With $x = \lceil n_0^{1/3} \rceil + 1 \in [n_0^{1/3} + 1, n_0^{1/3} + 2)$, we derive

$$\begin{aligned}
n &\in \left[\frac{n_0 + 3n_0^{2/3} + 3n_0^{1/3} + 1 - n_0^{1/3} - 2}{2}, 8(n_0 + 6n_0^{2/3} + 6n_0^{1/3} + 8) - 2(n_0^{1/3} - 1)\right] \\
n &\in \left[\frac{n_0 + 3n_0^{2/3} + 2n_0^{1/3} - 1}{2}, 8n_0 + 48n_0^{2/3} + 46n_0^{1/3} + 10\right] \\
n &\in \left[\frac{n_0}{2}, 9n_0\right] \quad \text{for } n_0 \text{ sufficiently large (such that } n_0 \geqslant 48n_0^{2/3} + 46n_0^{1/3} + 10).
\end{aligned}$$

The graph is $(p + 1)$–regular. Let $r := p + 1$.

The girth is at least $2 \lg_p q$ (since $4 \lg_p q - \lg_p 4 \geqslant 2 \lg_p q$ for $q \geqslant 2$). In terms of n and r, this yields

$$\mathsf{g}(X^{p,q}) \geqslant 2 \lg_p q = \frac{2}{3} \lg_p q^3 = \frac{2}{3} \frac{\ln q^3}{\ln p} = \frac{2 \ln(p+1)}{3 \ln p} \lg_{p+1} q^3 \geqslant \frac{1}{2} \lg_r n.$$

The expansion is $\lambda(X^{p,q}) \leqslant \frac{2\sqrt{p}}{p+1}$ due to Theorem 15. This concludes the proof. □

4.2.3 Counting Permutations

In the following, we prove the existence of a "not-too-large" set of permutations with certain properties. The lemma is a tailored restatement of Pătraşcu [Pat08a, Lemma 11], proven using the probabilistic method [AS00, Erd63].

Lemma 17. *Let S_ν denote the set of permutations of $[\nu] = \{1, 2 \ldots \nu\}$. For $\rho \in [0, 1]$, let $\mathcal{A}^\rho$ denote the set of all subsets of $[\nu]$ with $\rho \cdot \nu$ elements. Let $\mathsf{a} := |\mathcal{A}^\rho|$. Let $\mathcal{B} \subseteq \mathcal{A}^\rho$ of size at least $|\mathcal{B}| \geqslant \mathsf{b}$. There exists a set $\Pi \subseteq S_\nu$ of $|\Pi| =: \mu > \frac{\mathsf{a}}{\mathsf{b}} \ln \mathsf{a}$ permutations $\{\pi_1, \pi_2, \ldots, \pi_\mu\}$ such that for any set $A \in \mathcal{A}(\rho)$ there exists a permutation π_i with $\bigcup_{\bar{a} \in A} \{\pi_i(\bar{a})\} \in \mathcal{B}$.*

Proof. We denote $\pi_i(A) := \bigcup_{\bar{a} \in A} \{\pi_i(\bar{a})\}$. Let $\tilde{\pi}$ denote a randomly-chosen permutation $\tilde{\pi} : [\nu] \leftrightarrow [\nu]$. Fix $A \in \mathcal{A}^\rho$. The probability that $\tilde{\pi}(A) \in \mathcal{B}$ is at least

$$\mathsf{Pr}[\tilde{\pi}(A) \in \mathcal{B}] = \frac{\mathsf{b}}{\mathsf{a}}.$$

Consider μ permutations $\pi_i : [\nu] \leftrightarrow [\nu]$ selected independently at random, $\pi_1, \pi_2, \ldots, \pi_\mu$. The probability that *none* of the permutations π_i maps A to a set contained in $\mathcal{B}$ is bounded by

$$\mathsf{Pr}[\forall i \in [\mu] : \pi_i(A) \notin \mathcal{B}] = \left(1 - \frac{\mathsf{b}}{\mathsf{a}}\right)^\mu < e^{-\mu \mathsf{b}/\mathsf{a}}.$$

We need that for any $A \in \mathcal{A}(\rho)$ there is at least one permutation out of the μ permutations that maps A to set contained in $\mathcal{B}$. We apply the union bound to obtain the following.

$$q := \Pr\left[\exists A \in \mathcal{A}(\rho) \forall i \in [\mu] : \pi_i(A) \notin \mathcal{B}\right] \leqslant \mathsf{a}\left(1 - \frac{\mathsf{b}}{\mathsf{a}}\right)^{\mu} < \mathsf{a}e^{-\mu\mathsf{b}/\mathsf{a}}.$$

We need $1 - q > 0$ to apply the probabilistic method [AS00, Erd63] to guarantee that there exists a set $\Pi \subseteq S_\nu$ of μ permutations such that for each $A \in \mathcal{A}(\rho)$ there exists a permutation $\pi_i \in \Pi$ with $\pi_i(A) \in \mathcal{B}$. We obtain

$$\begin{aligned} 1 - \mathsf{a}e^{-\mu\mathsf{b}/\mathsf{a}} &> 0 \\ \frac{1}{\mathsf{a}} &> e^{-\mu\mathsf{b}/\mathsf{a}} \\ -\ln\mathsf{a} &> -\frac{\mu\mathsf{b}}{\mathsf{a}} \\ \mu &> \frac{\mathsf{a}}{\mathsf{b}}\ln\mathsf{a}. \end{aligned}$$

□

The following lemma is a direct consequence of Lemma 17.

Lemma 18 (Corollary of Lemma 17)**.** *Let $\mathcal{X}, \mathcal{Y}$ denote two sets of size $\nu := |\mathcal{X}| = |\mathcal{Y}|$. For $\rho \in [0,1]$, let $\mathcal{A}^\rho$ denote the set of all subsets of $\mathcal{X}$ with $\rho \cdot \nu$ elements, $\mathcal{A}^\rho := \binom{\mathcal{X}}{\rho\nu}$. Let $\mathcal{B}$ be a set of subsets of $\mathcal{Y}$ of size at least $|\mathcal{B}| \geqslant \mathsf{b}$ such that each subset has $\rho \cdot \nu$ elements. There exists a set of*

$$\mu = \frac{\rho\nu}{\mathsf{b}}\left(\frac{e}{\rho}\right)^{\rho\nu}\ln\left(\frac{e}{\rho}\right)$$

bijections $f_1, \ldots, f_\mu$ such that for any $A \in \mathcal{A}(\rho)$ there exists a bijection f_i with $\bigcup_{\bar{a} \in A}\{f_i(\bar{a})\} \in \mathcal{B}$.

Proof. We select an arbitrary bijection $f : \mathcal{X} \leftrightarrow \mathcal{Y}$. f maps each $A \in \mathcal{A}^\rho$ to a set $A' \in \mathcal{Y}$. Since f is a bijection, we have that

$$|\{A \in \mathcal{A}^\rho : f(a)\}| = |\mathcal{A}^\rho| =: \mathsf{a}.$$

By Lemma 17, there exists a set of permutations $\{\pi_1, \pi_2, \ldots \pi_\mu\}$ of $\mathcal{Y}$ such that for each A' (as above) there exists a π_i that maps A' to an element of $\mathcal{B}$.

We have[7] that

$$\mathsf{a} = \binom{\nu}{\rho\nu} < \left(\frac{e}{\rho}\right)^{\rho\nu}.$$

The statement of the lemma is immediate. □

[7] We use the following well-known inequality for binomial coefficients:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \cdot (n-1) \ldots (n-k+1)}{k!} \leqslant \frac{n^k}{k!} \leqslant \left(\frac{en}{k}\right)^k,$$

where the last step is due to the Taylor series of $e^x = \sum_{k=1}^{\infty} \frac{x^k}{k!}$, which yields $\forall x, k : e^x \geqslant \frac{x^k}{k!}$, in particular $e^k \geqslant \frac{k^k}{k!}$ and thus $k! \geqslant \left(\frac{k}{e}\right)^k$.

4.3 Reduction from Lopsided Set Disjointness

In this section, we show a lower bound on the space complexity of any approximate distance oracle on *any* base-graph G, based on essentially two parameters of G: the girth of G, and the *path-count* of G, which is the cardinality of the set $\mathcal{P}^{\ell,\kappa}(G)$.

We prove that a graph G with a large path-count and large girth is a "hard" base-graph for approximate distance oracles.

Lemma 19. *Let $G = (V, E)$ be a graph, such that an $(\alpha, 0)$–approximate distance oracle exists for G and all its subgraphs, using query time $\mathtt{t}$ and space $\mathcal{S}$. Let C denote the constant from the LSD communication complexity lower bound in Lemma 10. Let κ, ℓ be two positive integers, such that $\ell < \frac{\mathsf{g}(G)}{\alpha+1}$ and $|E| \geqslant \kappa\ell(2\mathtt{t}w/\ell)^{1/\mathsf{C}}$. Then,*

$$\mathcal{S} \geqslant \frac{\kappa}{e} \cdot \left(\frac{|\mathcal{P}^{\ell,\kappa}(G)|^{1/\kappa\ell}}{e(|E|/\kappa\ell)^{1-\mathsf{C}}} \right)^{\ell/\mathtt{t}} \cdot \left(\frac{1}{e|E|} \right)^{1/\mathtt{t}\kappa}.$$

The proof proceeds as follows. We start by giving some intuition in Section 4.3.1. We prove that if there exists a good data structure then there is a good (short) protocol for LSD (Section 4.3.2). Then, we use the LSD lower bound to show that there cannot be a good data structure (Section 4.3.3).

4.3.1 Intuition

> Was wirklich zählt, ist Intuition.
> (The only really valuable thing is intuition.)
>
> *Albert Einstein (1879–1975)*

We give a rough sketch of the proof of the main theorem, highlighting some technical details. Statements are not necessarily formally correct in this section (for details, see proof of Lemma 20). The proof is a reduction from the communication problem LSD, in which Alice is given a set $S_{\mathsf{Alice}} \subseteq [N \cdot B]$ of cardinality N, Bob is given a set $S_{\mathsf{Bob}} \subseteq [N \cdot B]$, and they must decide whether $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} = \emptyset$. For an illustration of the reduction, see Figure 4.4. There are communication lower bounds for LSD (see Section 4.2.1). We prove that a distance oracle with good parameters implies a good protocol for LSD, and derive our lower bound from the contrapositive of this claim.

A standard way to perform such reductions is to translate one query to the data structure into a $\mathtt{t}$ rounds of a communication protocol. In total, Alice sends Bob $\mathtt{t} \lg \mathcal{S}$ bits and Bob replies with $\mathtt{t}w$ bits. However, lower bounds in communication complexity are usually asymptotic. The standard reduction puts the constant multiplicative factor in the exponent of $\mathcal{S}$; therefore, using the standard reduction, we can only prove lower bounds on the space complexity of the form $\mathcal{S} \geqslant x^{\Omega(1)}$, where x is some expression that depends on the problem parameters. Since any distance oracle must use space n and since there is a trivial distance oracle that requires space $\Theta(n^2)$ (a complete distance table), this reduction can not provide a meaningful lower bound.

Pătraşcu and Thorup [PT06, Pat08a] found a way to prove lower bounds on the space complexity $\mathcal{S}$ of data structures that hold up to a polylogarithmic multiplicative factor. Alice holds κ independent queries to the same database, where κ is large ($\kappa = \mathcal{O}\left(\frac{n}{polylog(n)}\right)$). The κ queries performed in parallel are transformed into a communication protocol. We use the same approach.

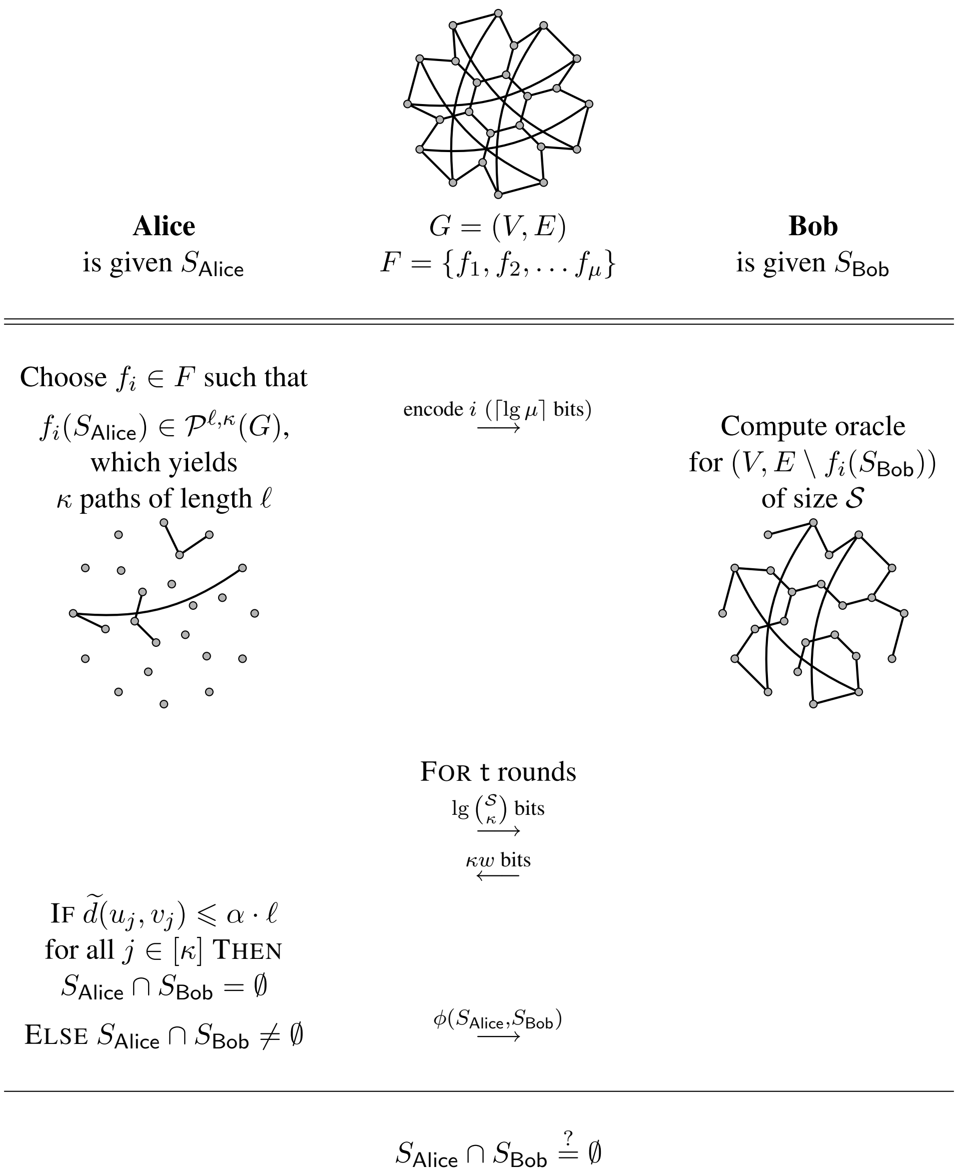
Alice
is given S_{Alice}
$G = (V, E)$
$F = \{f_1, f_2, \ldots f_\mu\}$
Bob
is given S_{Bob}
Choose $f_i \in F$ such that
$f_i(S_{\mathsf{Alice}}) \in \mathcal{P}^{\ell,\kappa}(G)$,
which yields
κ paths of length ℓ
$\xrightarrow{\text{encode } i \ (\lceil \lg \mu \rceil \text{ bits})}$
Compute oracle
for $(V, E \setminus f_i(S_{\mathsf{Bob}}))$
of size $\mathcal{S}$
FOR t rounds
$\xrightarrow{\lg \binom{\mathcal{S}}{\kappa} \text{ bits}}$
$\xleftarrow{\kappa w \text{ bits}}$
IF $\widetilde{d}(u_j, v_j) \leqslant \alpha \cdot \ell$
for all $j \in [\kappa]$ THEN
$S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} = \emptyset$
ELSE $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} \neq \emptyset$
$\xrightarrow{\phi(S_{\mathsf{Alice}}, S_{\mathsf{Bob}})}$
$S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} \overset{?}{=} \emptyset$

***Figure 4.4**: Illustration of a reduction from a distance oracle to a communication protocol for* LOPSIDEDSETDISJOINTNESS. *The protocol computes* $\phi(S_{\mathsf{Alice}}, S_{\mathsf{Bob}}) := \left(S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} \overset{?}{=} \emptyset\right)$. *$G$ and F are known to both Alice and Bob. Details are in the proof of Lemma 20.*

Our first objective is to prove that a good distance oracle implies a good protocol for LSD. We identify the universe of the LSD problem with the edge set of a graph. Let the universe size of the LSD problem be $N \cdot B = |E|$. For the moment, f denotes an arbitrary bijection $f : E \leftrightarrow [NB]$ between the edgeset E and the elements of the universe $[NB]$.

In the reduction, Alice plays the role of the querier, and Bob plays the role of the data structure. Bob transforms his set S_{Bob} into a subgraph of G, $G' = (V, E') = (V, E \setminus f(S_{\mathsf{Bob}}))$. In other words, Bob constructs a subgraph G' of G where the missing edges are the ones that correspond to his input set S_{Bob}. Bob then builds a distance oracle for the graph G'. Alice constructs a set of queries based on her input S_{Alice}, in a way that we shall specify next.

Let the girth $g = \mathrm{g}(G)$ of the graph $G = (V, E)$ be large. Now consider one query to the data structure, which asks for $\widetilde{d}_{G'}(u, v)$ where u and v are close to each other in G, that is, $d_G(u, v) = \ell < \frac{g}{\alpha+1}$. Let $p_{u,v}$ be the path of length ℓ between u and v in G. We ask for the distance in G'. A query to the distance oracle returns the answer $\widetilde{d}_{G'}(u, v)$, which is an $(\alpha, 0)$–approximation for $d_{G'}(u, v)$. We know that the girth of G' is large, $\mathrm{g}(G') \geqslant g$, and that u and v are close in G (at distance $\ell < \frac{g}{\alpha+1}$). The result of the distance query is at most $\alpha\ell$ if and only if all edges on the path $p_{u,v}$ are in E'. Otherwise, the result of the distance query is a number strictly larger than $\alpha\ell$.

Thus, using one query to the distance oracle, we can distinguish the case that all edges of $p_{u,v}$ are in E' from the case where at least one edge is missing. Now, if we perform κ queries of this form, we can check κ paths. Therefore, using κ queries we can check whether $\kappa\ell$ edges are all in the graph, or whether at least one of these $\kappa\ell$ edges is missing. By setting $NB = |E|$ and $N = \kappa\ell$, this is an instance of the LSD problem. By doing a standard transformation from a data structure to a communication protocol, we connect the parameters of the data structure to those of a protocol for LSD: Alice sends roughly $\mathtt{t}\kappa \lg(\mathcal{S}/\kappa)$ bits to Bob, and Bob sends roughly $\mathtt{t}\kappa w$ bits to Alice.

However, the instance described above is a very specific instance of the LSD problem, where Alice's set is restricted to a set that corresponds to paths in G. For a general instance of LSD, Alice's input may not necessarily map to a collection of κ vertex-disjoint paths of length ℓ each. There is a technique of Pătraşcu [Pat08a] to perform a reduction from LSD even when only a non-negligible fraction of Alice's inputs map to a set of vertex-disjoint paths: in a preliminary round of communication, Alice and Bob choose the bijection f from some "not-too-large" set of bijections (see Lemma 18). In order to obtain such a set of bijections, we prove that there is a large set of sets of vertex-disjoint paths in G (we refer to this as the *path-count*, as in Definition 34). If the path-count is sufficiently large, then we obtain a strong lower bound.

For details, we refer to the proof of Lemma 20 in the next section.

4.3.2 Reduction from a data structure to a communication protocol

We prove that the distance oracle data structure can be transformed into a protocol for LSD. For an illustration, see Figure 4.4.

Lemma 20. *Let $G = (V, E)$ be a graph, such that an $(\alpha, 0)$–approximate distance oracle exists for G and all its subgraphs, using query time $\mathtt{t}$ and space $\mathcal{S}$ in the cell-probe model with word size w. Let κ, ℓ be two positive integers, such that $\ell < \frac{\mathrm{g}(G)}{\alpha+1}$. Then, there exists a protocol for LSD with parameters $N = \kappa\ell$ and $B = |E|/N$, where Alice sends*

$$\mathtt{t}\kappa \lg(e\mathcal{S}/\kappa) + N \lg(eB) + \lg(eBN) - \lg|\mathcal{P}^{\ell,\kappa}(G)| \textit{ bits},$$

and Bob sends $\kappa\mathtt{t}w$ bits.

Proof. We begin by defining a bijection between the universe $[NB]$ and E. For now, any bijection will do — additional restrictions will be imposed later. Denote the bijection by $f : [NB] \leftrightarrow E$. Alice and Bob both know G and f.

In the LSD problem, Alice receives a set $S_{\mathsf{Alice}} \subseteq [NB]$ of cardinality $|S_{\mathsf{Alice}}| = N$, and Bob receives a set $S_{\mathsf{Bob}} \subseteq [NB]$. We now derive a protocol for LSD based on the existence of the data structure. Bob uses his set S_{Bob} to construct a set of edges, $E' = E \setminus f(S_{\mathsf{Bob}})$. That is, an edge

$e \in E$ is in E' if and only if its corresponding element is *not* in Bob's set S_{Bob}. Bob preprocesses the graph $G' = (V, E')$ creating the distance oracle data structure, and from now on, Bob "plays" the role of the data structure. Since G' is a subgraph of G, by the condition of Lemma 19, this data structure requires space $\mathcal{S}$, query time t, and it is an $(\alpha, 0)$–approximate distance oracle for G'.

Alice constructs the set $P = f(S_{\mathsf{Alice}})$. For now, assume that $P \in \mathcal{P}^{\ell,\kappa}(G)$. We call this assumption the *perfect bijection scenario*; we shall remove this assumption later. Under this assumption, P can be written as the union of κ vertex-disjoint paths, each of length ℓ. Let $(u_1, v_1), \ldots, (u_\kappa, v_\kappa)$ be the endpoints of these paths.

For every pair (u_i, v_i), we know that $d_G(u_i, v_i) = \ell$ (it cannot be smaller since the graph has large girth and thus there is no cycle of length $< 2\ell$ in G). Denote the path of length ℓ between u_i and v_i by p_i. We see that, on one hand, an $(\alpha, 0)$–approximate distance query on the pair (u_i, v_i) returns $\widetilde{d}(u_i, v_i) \leqslant \alpha\ell$ if all of the edges of the path p_i are in E'. On the other hand, the result of the distance query is $\widetilde{d}(u_i, v_i) \geqslant \mathsf{g}(G) - \ell$ if at least one of the edges of p_i is not in E', since there are no cycles shorter than $\mathsf{g}(G)$, and since an approximate distance oracle never returns an underestimate of the distance, but always an overestimate or the correct value. After querying all κ distances $(u_1, v_1), \ldots, (u_\kappa, v_\kappa)$, if all of the κ queries return distances at most $\widetilde{d}(u_i, v_i) \leqslant \alpha\ell$, then Alice and Bob conclude that $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} = \emptyset$, otherwise they conclude that $S_{\mathsf{Alice}} \cap S_{\mathsf{Bob}} \neq \emptyset$.

Alice and Bob, in order to compute the answer to LSD, simulate κ queries to the data structure by communication [MNSW98, Pat08a].

Bob computes the data structure itself, based on E'. Alice computes the set P and the pairs $(u_1, v_1), \ldots, (u_\kappa, v_\kappa)$. Alice then considers which cells should be probed in the first round of each of the κ queries, and sends the set of probed cells to Bob. This set can be communicated using $\lg \binom{\mathcal{S}}{\kappa}$ bits (it is crucial not to send the queries one by one, which would require $\kappa \lg \mathcal{S}$ bits [PT06, Pat08a]). Bob replies with the contents of these cells, using $w\kappa$ bits. Next, Alice sends the set of cells to be probed in the second round, using another $\lg \binom{\mathcal{S}}{\kappa}$ bits, and Bob replies, using another $w\kappa$ bits. This procedure is repeated for t rounds in total. Overall, Alice sends

$$\mathsf{t} \lg \binom{\mathcal{S}}{\kappa} \leqslant \mathsf{t} \cdot \lg \left(\frac{e\mathcal{S}}{\kappa}\right)^{\kappa} = \mathsf{t}\kappa \lg \left(\frac{e\mathcal{S}}{\kappa}\right) \text{ bits,}$$

and Bob sends $\kappa \mathsf{t} w$ bits.

We eliminate the perfect bijection assumption by including an additional round of communication at the beginning of the protocol. In this round, Alice chooses a particular bijection f to reach the perfect bijection scenario. Instead of having only one bijection $f : [NB] \leftrightarrow E$, Alice and Bob share knowledge of μ bijections $f_1, f_2, \ldots, f_\mu$, all between $[NB]$ and E. This set of bijections must have the property that for any set $S_{\mathsf{Alice}} \subseteq [NB]$ of cardinality N, there exists an i such that choosing $f = f_i$ reaches the perfect bijection scenario, that is, $\exists i \in [\mu] : f_i(S_{\mathsf{Alice}}) = \mathcal{P}^{\ell,\kappa}(G)$. If there is such a set of μ bijections, then Alice and Bob can reach the perfect bijection scenario by having Alice send $\lceil \lg \mu \rceil$ bits (the index of the bijection they use) and then continue as before. By Lemma 18, there is such a set of size

$$\mu = N \ln (eB) \cdot \frac{(eB)^N}{|\mathcal{P}^{\ell,\kappa}(G)|}.$$

Therefore, there is an LSD protocol where Alice first sends

$$\begin{aligned} & \lg N + \lg \ln(eB) + N \lg(eB) - \lg |\mathcal{P}^{\ell,\kappa}(G)| \text{ bits} \\ \leqslant \; & \lg(eBN) + N \lg(eB) - \lg |\mathcal{P}^{\ell,\kappa}(G)| \text{ bits} \end{aligned}$$

in order to reach the perfect bijection scenario, and then she sends another

$$\mathtt{t}\kappa \lg \left(\frac{e\mathcal{S}}{\kappa} \right) \text{ bits}$$

in the perfect bijection scenario. This yields a total of at most

$$\lg(eBN) + N \lg(eB) - \lg |\mathcal{P}^{\ell,\kappa}(G)| + \mathtt{t}\kappa \lg \left(\frac{e\mathcal{S}}{\kappa} \right) \text{ bits.}$$

Bob sends $\kappa \mathtt{t} w$ bits. We ignore the final 1 bit that Alice sends to inform Bob of the result. For an illustration, see Figure 4.4. □

4.3.3 Communication complexity implies space complexity

Conditioned on the existence of a space-efficient distance oracle, there is a communication protocol for LSD with low communication complexity (Lemma 20). In the following, we prove that the lower bound on the communication problem LSD (Lemmas 9 and 10) yields a lower bound on the space complexity of distance oracles.

Recall the statement of Lemma 19: *Let $G = (V, E)$ be a graph, such that an $(\alpha, 0)$–approximate distance oracle exists for G and all its subgraphs, using query time $\mathtt{t}$ and space $\mathcal{S}$. Let C denote the constant from the LSD communication complexity lower bound in Lemma 10. Let κ, ℓ be two positive integers, such that $\ell < \frac{\mathrm{g}(G)}{\alpha+1}$ and $|E| \geqslant \kappa\ell(2\mathtt{t}w/\ell)^{1/\mathsf{C}}$. Then,*

$$\mathcal{S} \geqslant \frac{\kappa}{e} \cdot \left(\frac{|\mathcal{P}^{\ell,\kappa}(G)|^{1/\kappa\ell}}{e(|E|/\kappa\ell)^{1-\mathsf{C}}} \right)^{\ell/\mathtt{t}} \cdot \left(\frac{1}{e|E|} \right)^{1/\mathtt{t}\kappa} .$$

Proof of Lemma 19. If a protocol computes LSD with parameters N and B, then either Alice sends at least $\mathsf{C}N \lg B$ bits or Bob must send at least NB^{C} bits (Lemma 10).

Bob communicates $\kappa \mathtt{t} w$ bits. By the condition of Lemma 19, $B \geqslant (2\mathtt{t}w/\ell)^{1/\mathsf{C}}$. This implies

$$NB^{\mathsf{C}} \geqslant \kappa\ell \cdot 2\mathtt{t}w/\ell = 2\kappa\mathtt{t}w.$$

Bob, by sending $\kappa\mathtt{t}w$ bits, uses strictly less than NB^{C} bits. The lower bound on the communication complexity of LSD implies that Alice must communicate at least $\mathsf{C}N \lg B$ bits. Using the protocol of Lemma 20, we have that

$$\lg(eBN) + N \lg(eB) - \lg |\mathcal{P}^{\ell,\kappa}(G)| + \mathtt{t}\kappa \lg \left(\frac{e\mathcal{S}}{\kappa} \right) \geqslant \mathsf{C}N \lg B.$$

Starting from this inequality, we derive a bound on $\mathcal{S}$. Recall that $N = \kappa\ell$ and recall that the edgeset is identified with the universe of LSD ($|E| = BN$). We get

$$\begin{aligned}
\mathsf{t}\kappa \lg\left(\frac{e\mathcal{S}}{\kappa}\right) &\geqslant \mathsf{C}N\lg B - \lg(eBN) - N\lg(eB) + \lg|\mathcal{P}^{\ell,\kappa}(G)| \\
\mathsf{t}\kappa \lg\left(\frac{e\mathcal{S}}{\kappa}\right) &\geqslant \lg|\mathcal{P}^{\ell,\kappa}(G)| + \mathsf{C}\kappa\ell\lg B - \kappa\ell\lg(eB) - \lg(eBN) \\
\lg\left(\frac{e\mathcal{S}}{\kappa}\right) &\geqslant \frac{1}{\mathsf{t}\kappa}\lg|\mathcal{P}^{\ell,\kappa}(G)| + \frac{\mathsf{C}\ell}{\mathsf{t}}\lg B - \frac{\ell}{\mathsf{t}}\lg(eB) - \frac{1}{\mathsf{t}\kappa}\lg(eBN) \\
\frac{e\mathcal{S}}{\kappa} &\geqslant |\mathcal{P}^{\ell,\kappa}(G)|^{1/\mathsf{t}\kappa}\cdot(B)^{\mathsf{C}\ell/\mathsf{t}}\cdot(eB)^{-\ell/\mathsf{t}}\cdot(eBN)^{-1/\mathsf{t}\kappa} \\
\mathcal{S} &\geqslant \frac{\kappa}{e}\cdot|\mathcal{P}^{\ell,\kappa}(G)|^{1/\mathsf{t}\kappa}\cdot(eB^{1-\mathsf{C}})^{-\ell/\mathsf{t}}\cdot(eBN)^{-1/\mathsf{t}\kappa} \\
\mathcal{S} &\geqslant \left(\frac{|\mathcal{P}^{\ell,\kappa}(G)|^{1/\kappa\ell}}{eB^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}}\cdot\frac{\kappa}{e(e|E|)^{1/\mathsf{t}\kappa}},
\end{aligned}$$

which yields the statement of the theorem. □

4.3.4 Counting Paths

We prove a lower bound on the size of the set of sets of disjoint paths $\mathcal{P}^{\ell,\kappa}(G)$.

Lemma 21. *Let $G = (V, E)$ be an r–regular graph and κ, ℓ be two positive integers, such that the following three conditions hold:*

1. $\lambda(G) \leq 0.1$,
2. $|V| \geqslant 20\kappa\ell$, *and*
3. $\ell < \mathsf{g}(G)$.

Then

$$|\mathcal{P}^{\ell,\kappa}(G)| \geqslant \binom{|V|}{\kappa}\cdot\left(\frac{r}{8}\right)^{\kappa\ell}.$$

Proof. Let $N = \kappa\ell$.

Let us first choose one path. There are $|V|$ vertices to start a path. Since the graph is r–regular, and since $\ell < \mathsf{g}(G)$, we have r choices for the first step and $r - 1$ choices for each subsequent step. This yields

$$|V|\cdot r\cdot(r-1)^{\ell-1} \tag{4.1}$$

possibilities to choose one path. Since the graph is undirected and since we want to reduce LSD to *distance* queries (as opposed to actual *path* queries), an $s - t$ path is equal to an $t - s$ path. We divide (4.1) by 2 to account for this.

Let us now choose κ vertex-disjoint paths of length ℓ — one by one. Recall that, due to Corollary 14 (implied by Theorem 13 of Alon et al. [AFWZ95]), the probability that a random walk of ℓ steps from a uniformly random starting vertex stays inside $U \subseteq V$, where $|U| \geqslant \frac{9}{10}|V|$, is at least $\frac{1}{2^\ell}$. Let $U := V \setminus A$. Since $|V| = 20\kappa\ell$ and $|A| \leqslant \kappa\ell$, we have that $|U| \geqslant \frac{9}{10}|V|$.

According to (4.1) (divided by 2) there are at least

$$\frac{|V|}{2}\cdot r\cdot(r-1)^{\ell-1} \geqslant \frac{|V|}{2}\cdot\left(\frac{r}{4}\right)^{\ell}$$

(simple) paths of length ℓ. According to the corollary, each path has probability at least $\frac{1}{2^\ell}$ to avoid A. Therefore, the number of different paths of length ℓ in G that do not use any vertices of A is at least

$$|V| \cdot \left(\frac{r}{8}\right)^\ell .$$

We apply this argument κ times to generate all sets with κ paths of length ℓ. After each application, we add the vertices of the path to A. We divide by $\kappa!$ to account for the fact that the order in which the paths were chosen does not matter. We obtain

$$|\mathcal{P}^{\ell,\kappa}(G)| \geqslant \frac{\left(|V| \cdot \left(\frac{r}{8}\right)^\ell\right)^\kappa}{\kappa!} \geqslant \frac{|V|^\kappa}{\kappa!} \cdot \left(\frac{r}{8}\right)^{\kappa\ell} \geqslant \binom{|V|}{\kappa} \cdot \left(\frac{r}{8}\right)^{\kappa\ell} .$$

This concludes the proof of Lemma 21. □

4.3.5 Assembly

We now combine Lemma 21 with Lemma 19 to prove a lower bound for any expander graph based on its expansion, degree, and girth. After this, we use the Ramanujan graph from Lemma 16 to derive the main result of this chapter. Both proofs consist of just a sequence of calculations to glue together all the conditions.

Lemma 22. *Let C denote the constant from the LSD communication complexity lower bound in Lemma 10. Let $G = (V, E)$ be an r–regular expander graph on $|V| = n$ vertices, n sufficiently large, with expansion $\lambda(G) \leqslant 0.1$ and girth $g = \mathrm{g}(G)$. In the cell-probe model with word-length at most $w = n^{o(1)}$, for an integer $\alpha \geqslant 1$, any $(\alpha, 0)$–approximate distance oracle with query time $\mathtt{t}$ that works for G and its subgraphs, requires space at least*

$$\mathcal{S} \geqslant \frac{n}{\lg n} \cdot r^{\Omega\left(\frac{g}{\alpha \mathtt{t}}\right)}$$

given that

- *$g = \mathrm{g}(G) \geqslant 2\alpha$ and*
- *$r \geqslant \left(\frac{4\mathtt{t}w\alpha}{g}\right)^{1/\mathsf{C}}$.*

Proof of Lemma 22. Let $\ell = \lfloor \frac{g}{2\alpha} \rfloor \leqslant \lg n$. Let $\kappa = \frac{|V|}{20\ell}$. This yields $\kappa \geqslant \frac{n}{20 \lg n}$. Recall that $N = \kappa\ell$.

The conditions of Lemma 21 are satisfied:

- $\lambda(G) \leq 0.1$ (by the condition of Lemma 22)
- $|V| \geqslant 20\kappa\ell$ (by the definition of ℓ and κ)
- $\ell < \mathrm{g}(G) = g$ since $g \geqslant 2\alpha$ and $\ell = \lfloor \frac{g}{2\alpha} \rfloor$

From Lemma 21, we know that (under the above conditions)

$$|\mathcal{P}^{\ell,\kappa}(G)| \geqslant \binom{|V|}{\kappa} \cdot \left(\frac{r}{8}\right)^{\kappa\ell} .$$

The conditions of Lemma 19 are satisfied.

- $\ell < \frac{\mathrm{g}(G)}{\alpha+1}$
- $|E| = |V| \cdot \frac{r}{2} = 10\kappa\ell r \geqslant 10\kappa\ell \left(\frac{4\mathrm{t}w\alpha}{g}\right)^{1/\mathsf{C}} \geqslant \kappa\ell(2\mathrm{t}w/\ell)^{1/\mathsf{C}}$

From Lemma 19, we know that (under the above conditions)

$$\mathcal{S} \geqslant \kappa \cdot \left(\frac{|\mathcal{P}^{\ell,\kappa}(G)|^{1/\kappa\ell}}{e(|E|/\kappa\ell)^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}} /((e|E|)^{1/\mathsf{t}\kappa}e).$$

Since we are interested in the asymptotic behavior of $\mathcal{S}$, we may ignore constant factors. We claim that $(e|E|)^{1/\mathsf{t}\kappa}e = \Theta(1)$. Since $|E| \geqslant 1$, we have that $(e|E|)^{1/\mathsf{t}\kappa}e = \Omega(1)$. Since $|E| \leqslant V^2 = 400\ell^2\kappa^2 \leqslant 400\kappa^4$,

$$\begin{aligned}
(e|E|)^{1/\mathsf{t}\kappa}e &\leqslant \mathcal{O}(1)\\
(e|E|)^{1/\mathsf{t}\kappa} &\leqslant \mathcal{O}(1)\\
\frac{\lg(e400\ell\kappa)}{\mathsf{t}\kappa} &\leqslant \mathcal{O}(1)\\
\frac{\lg\kappa}{\mathsf{t}\kappa} &\leqslant \mathcal{O}(1).
\end{aligned}$$

We derive (using $|V| = 20\kappa\ell$ and $|E| = |V|\frac{r}{2}$)

$$\begin{aligned}
\mathcal{S} &\geqslant \kappa \cdot \left(\frac{|\mathcal{P}^{\ell,\kappa}(G)|^{1/\kappa\ell}}{e(|E|/\kappa\ell)^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}}\\
&\geqslant \frac{n}{20\lg n} \cdot \left(\frac{\left(\binom{|V|}{\kappa} \cdot \left(\frac{r}{8}\right)^{\kappa\ell}\right)^{1/\kappa\ell}}{e(|E|/\kappa\ell)^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}}\\
&\geqslant \frac{n}{20\lg n} \cdot \left(\frac{\binom{|V|}{\kappa}^{1/\kappa\ell} \cdot r}{8e(|E|/\kappa\ell)^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}}\\
&\geqslant \frac{n}{20\lg n} \cdot \left(\frac{\binom{20\kappa\ell}{\kappa}^{1/\kappa\ell} \cdot r}{8e(10r)^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}}\\
&\geqslant \frac{n}{20\lg n} \cdot \left(\frac{(20\ell)^{1/\ell} \cdot r}{8e(10r)^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}}\\
&\geqslant \frac{n}{20\lg n} \cdot \left(\frac{r^{\mathsf{C}}}{8e10^{1-\mathsf{C}}}\right)^{\ell/\mathsf{t}} \cdot (20\ell)^{1/\mathsf{t}}.
\end{aligned}$$

By $\ell = \lfloor\frac{g}{2\alpha}\rfloor$, we get the statement in the theorem for sufficiently large n. □

Based on Lemma 22, we now use the Ramanujan graph from Lemma 16 to derive the lower bound stated in the main theorem of this chapter (Theorem 8).

Proof of Theorem 8. Let C denote the constant from the LSD communication complexity lower bound in Lemma 10.

Let $n_0 = \Theta(n)$. Let $r_0 := \left(\frac{8\mathsf{t}w\alpha}{\lg n_0}\right)^{2/\mathsf{C}}$. We assume that n_0 is sufficiently large such that $r_0 \geqslant \max\{400, \left(\frac{4}{\mathsf{C}}\right)^{4/\mathsf{C}}\}$.

To use Lemma 16, we need that $n_0 > 8r_0^3$. Since $\mathsf{t}, \alpha \in \mathcal{O}(polylog(n))$ and $w = n^{o(1)}$, we have that

$$8\left(\frac{8\mathsf{t}w\alpha}{\lg n_0}\right)^{6/\mathsf{C}} < 8\left(8\mathsf{t}w\alpha\right)^{6/\mathsf{C}} < n_0,$$

thus there exists a graph $G = (V, E)$ with the following properties:

1. $|V| = n'$ with $\frac{n_0}{2} \leqslant n' \leqslant 9n_0$
2. G is r–regular, where $r_0 \leqslant r \leqslant 2r_0$
3. The girth of G is at least $\mathsf{g}(G) \geqslant \frac{1}{2}\lg_r n$
4. $\lambda(G) \leqslant \frac{2\sqrt{r-1}}{r}$

Since $n_0 = \Theta(n)$, we have that $n' = \Theta(n)$. To simplify notation, we use n instead of n' for the remainder of the proof.

To apply Lemma 22, three conditions must be verified.

- $\lambda(G) \leqslant \frac{2\sqrt{r-1}}{r} \leqslant 0.1$ holds for $r \geqslant 400$.
- $g = \mathsf{g}(G) \geqslant 2\alpha$
  We have that $r_0 = \left(\frac{8\mathsf{t}w\alpha}{\lg n_0}\right)^{2/\mathsf{C}}$ and $r \in [r_0, 2r_0]$. Also, both $\mathsf{t} \leqslant \lg n$ and $\alpha \leqslant \lg n$.

$$\begin{aligned}
\mathsf{g}(G) &\geqslant \frac{1}{2}\lg_r n \\
&= \frac{1}{2}\frac{\lg n}{\lg r} \\
&\geqslant \frac{1}{2}\frac{\lg n}{\lg(2r_0)} \\
&= \frac{1}{2}\frac{\lg n}{\lg\left(2\left(\frac{8\mathsf{t}w\alpha}{\lg n_0}\right)^{2/\mathsf{C}}\right)} \\
&= \frac{\mathsf{C}}{4}\frac{\lg n}{\lg\left(\frac{2^{\mathsf{C}/2}8\mathsf{t}w\alpha}{\lg n_0}\right)} \\
&\geqslant \frac{\mathsf{C}}{4}\frac{\lg n}{\lg\left(\frac{2^{\mathsf{C}/2}8w\lg^2 n}{\lg n_0}\right)} \\
&= \Omega\left(\frac{\lg n}{\lg(w\lg n)}\right)
\end{aligned}$$

Since $\alpha = o\left(\frac{\lg n}{\lg(wn)}\right)$, the condition holds. Note that the necessary condition on α is

$$\begin{aligned}
\frac{\mathsf{C}}{4}\frac{\lg n}{\lg\left(\frac{2^{\mathsf{C}/2}8w\lg^2 n}{\lg n_0}\right)} &\geqslant 2\alpha \\
\frac{\mathsf{C}}{8}\frac{\lg n}{\lg(2^{\mathsf{C}/2}8) + \lg w + 2\lg\lg n - \lg\lg\frac{n}{2}} &\geqslant \alpha \\
\frac{\mathsf{C}}{8}\frac{\lg n}{\frac{\mathsf{C}}{2} + \lg 8 + \lg w + \lg\lg\frac{n}{2}} &\geqslant \alpha \geqslant 1.
\end{aligned}$$

The lower bound thus extends to constant w and, in particular, to the bit-probe model [MP69].

- $\left(\frac{4\mathsf{t}w\alpha}{g}\right)^{1/\mathsf{C}} \leqslant r$

  We need two inequalities on r. Since $r \geqslant r_0 \geqslant \left(\frac{4}{\mathsf{C}}\right)^{4/\mathsf{C}}$ (for sufficiently large n_0), we have that $(\lg r)^{1/\mathsf{C}} \leqslant \sqrt{r}$. Since $r \geqslant r_0 = \left(\frac{8\mathsf{t}w\alpha}{\lg n_0}\right)^{2/\mathsf{C}} \geqslant \left(\frac{8\mathsf{t}w\alpha}{\lg n}\right)^{2/\mathsf{C}}$,

$$\begin{aligned}
\left(\frac{4\mathsf{t}w\alpha}{g}\right)^{1/\mathsf{C}} &\leqslant \left(\frac{8\mathsf{t}w\alpha\lg r}{\lg n}\right)^{1/\mathsf{C}} \\
&\leqslant \left(\frac{8\mathsf{t}w\alpha}{\lg n}\right)^{1/\mathsf{C}}\cdot(\lg r)^{1/\mathsf{C}} \\
&\leqslant \left(\frac{8\mathsf{t}w\alpha}{\lg n}\right)^{1/\mathsf{C}}\cdot\sqrt{r} \\
&\leqslant \sqrt{r}\cdot\sqrt{r} \leqslant r.
\end{aligned}$$

We now apply Lemma 22. Since $g = \mathsf{g}(G) \geqslant \frac{1}{2}\lg_r n$, we have that

$$r^g \geqslant r^{\frac{1}{2}\lg_r n} = n^{1/2},$$

therefore,

$$\mathcal{S} \geqslant \frac{n}{\lg n}\cdot n^{\Omega\left(\frac{1}{\alpha\mathsf{t}}\right)}$$

This concludes the proof. □

4.4 Conclusion and Open Problems

Theorem 8 implies that a distance oracle with query time t and stretch $(\alpha, 0)$ requires space $n^{1+\Omega(1/\alpha\mathsf{t})}$. Since our proof holds even for sparse graphs with $m = \widetilde{\mathcal{O}}(n)$ edges, the space requirement is strictly larger than the original graph size. For sparse graphs, our space lower bound is an improvement over the lower bound by Thorup and Zwick [TZ05], which states that at least space $\Omega(m)$ is required. We prove that $\mathcal{O}(m)$ is not enough.

Our lower bound also indicates that the tradeoff of the distance oracle of Thorup and Zwick can potentially be improved for sparse graphs. Their tradeoff is that for multiplicative stretch

$(\alpha, 0)$ and query time $\mathcal{O}(\alpha)$, the space is roughly $n^{1+\mathcal{O}(1/\alpha)}$. Our lower bound only proves space requirement $n^{1+\Omega(1/\alpha^2)}$. There is a gap between the upper and the lower bound. Mendel and Naor [MN06] improve the query time to $\mathcal{O}(1)$ while maintaining the same amount of space asymptotically. Their tradeoff is tight up to constant factors in the exponent with respect to our lower bound.

Two technical questions remain open.

Linear Number of Edges. The worst-case graphs in our proof have degree $\widetilde{\mathcal{O}}(1)$. It would be interesting to generalize the proof to constant-degree graphs.

Graphs without Large Girth. In our proof, we require that a graph has large girth. It may be possible to remove this requirement. In the reduction, we perform κ distance queries. The corresponding κ paths must be vertex-disjoint and non-bypassable, meaning that any alternative path is long. We ensure that paths do not have a short alternative by using graphs with large girth. It may however be feasbile to use graphs without large girth to prove a lower bound.

Our lower bound applies to general sparse graphs. It however does not apply to specific graph classes such as those with many short cycles; efficient distance oracles may still be possible for specific graph classes.

Unversehens hängt alles ineinander [...]
(everything is connected)
Max Frisch [Fri64, p. 116]

5 Distance Oracles for Power-law Graphs

5.1 Introduction

Although complex networks are very common in practice (Section 1.1.2), there is no distance oracle with provable guarantees better than those of the general distance oracle of Thorup and Zwick [TZ05]. For stretch parameter $k = 2$, the distance oracle of Thorup and Zwick has the following worst-case performance: the size is $\mathcal{O}(n^{3/2})$ and the stretch is $(3, 0)$. Fortunately, the theoretical worst-case stretch bounds of Thorup and Zwick's distance oracle [TZ05] (and, also, of their routing scheme [TZ01]) are not observed in practice[1] [KFY04], even though they are tight.

In this chapter, we make an attempt to bridge the gap between theory and practice. We provide the first theoretical analysis that directly links the power-law exponent τ of a random power-law graph to the bound on distance oracle sizes.

We adapt the distance oracle of Thorup and Zwick [TZ05] to optimize it for unweighted, undirected power-law graphs. The scheme by Thorup and Zwick is based on a set of *landmarks* selected uniformly at random. Instead of sampling landmarks at random, we select the nodes with highest degrees as landmarks.

The use of nodes with high degrees is a heuristic that has been proposed by many researchers. The "high-degree" heuristic is also very common in practice. For power-law graphs it particularly makes sense to leverage the power of high-degree nodes. These nodes are also called *hubs*. These *hubs* "appear in most large complex networks" [Bar03, p. 63].

> Connectors are an extremely important component of our social network. They create trends and fashions, make important deals, spread fads, or help launch a restaurant. They are the thread of society, smoothly bringing together different races, levels of education, and pedigrees. [...] Connectors — nodes with an anomalously large number of links — are present in very diverse complex systems, ranging from the economy to the cell. They are a fundamental property of most networks [...]. [Bar03, p. 56]
>
> Indeed, with links to an unusually large number of nodes, hubs create short paths between any two nodes in the system [Bar03, p. 64]

Intuitively, using these hubs to approximate distances in power-law graphs is a good heuristic. The main result of this chapter is a theoretical proof that may explain why this heuristic performs well in practice.

[1] Krioukov et al. [KFY04, Section IV.B] report routing tables with $\sim$ 52 entries for random power-law graphs [ACL00] with 10,000 nodes. The bound by Thorup and Zwick [TZ01] is $\mathcal{O}(\sqrt{n \lg n})$ entries. For $n = 10000$, $\sqrt{n \lg n}$ is ~ 365.

5.1.1 Overview of the Result

We give an informal statement of our result. The precise statement is deferred to Theorem 30.

For the nodes of a power-law graph, the probability that a node has degree x is proportional to $x^{-\tau}$ for some τ, which is called the *power-law exponent*. For most practical scenarios, the power-law exponent lies in the interval $2 < \tau < 3$. These inequalities are assumed to hold in the following.

The complexity analysis of our distance oracle is based on the random power-law graph model with expected degree sequence proposed by Aiello, Chung and Lu [ACL00, CL02, Lu02, CL06] with some minor simplifications.

Let $\gamma = \frac{\tau-2}{2\tau-3} + \varepsilon$ and $\varepsilon > 0$. For sufficiently large n, we prove that for a random power-law graph (sampled from the modified Chung-Lu model, see Definition 37) with n nodes, with probability at least $1 - 1/n$, our distance oracle of size $\mathcal{O}(n^{1+\gamma} \lg n)$ can be constructed in time $\mathcal{O}(n^{1+\gamma} \lg n)$. With probability 1, the distance oracle has stretch $(3, 0)$. The space requirement of $\mathcal{O}(n^{1+\gamma} \lg n)$ (for a plot of γ with respect to τ, see Figure 5.1) improves upon the general distance oracle of size $\mathcal{O}(n^{3/2})$ by Thorup and Zwick [TZ05].

Our bounds on the space complexity of the distance oracle of Thorup and Zwick [TZ05] extend to the *labeled compact routing scheme*[2] by Thorup and Zwick [TZ01].

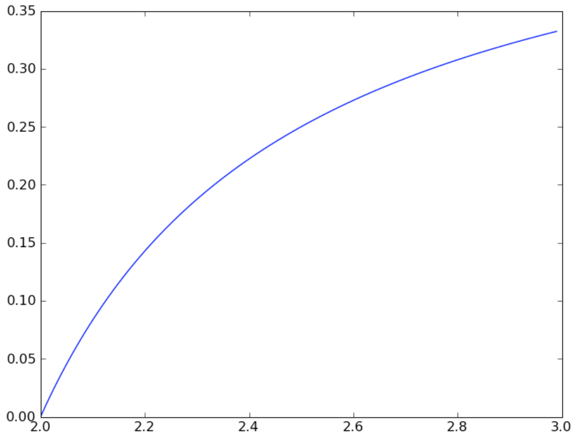

***Figure 5.1**: The figure shows a plot of $f(\tau) = \frac{\tau-2}{2\tau-3}$ for $\tau \in (2, 3)$. For values of τ close to 2, for example for $\tau = 2.1$, which is the exponent that fits the power-law distribution well to the degree distribution of the actual Internet inter-domain graph [FFF99, KFY04], our bound is $\mathcal{O}(n^{13/12+\varepsilon})$, which indicates that the adapted distance oracle (and the adapted routing scheme) could be very effective on Internet-like graphs.*

[2] In this thesis, we focus on distance oracles; we do not define compact routing schemes. Some notes on the routing scheme, without detailed explanation and proof. The routing scheme is a *fixed-port* scheme, meaning that it works for any permutation of port number assignments on any node. The routing scheme requires a stretch–5 handshaking (see [TZ01, Section 4]), and uses addresses and message headers of size $\mathcal{O}(\lg n \lg \lg n)$, with probability at least $1 - o(1)$. Addresses and headers are based on an efficient path encoding scheme using $\mathcal{O}(\lg n \lg \lg n)$ bits per node. The encoding scheme relies on specific distance properties of power-law graphs. For details, see [CSTW09b].

5.1.2 Related Work

Due to their large occurrence in practice, various aspects of power-law graphs and complex networks have been studied.

There is some evidence that, despite their unique features, power-law graphs are actually not "easy" instances for algorithms. Although power-law graphs are sparse, optimization problems remain hard: problems such as COLORING or CLIQUE are **NP**–hard for power-law graphs as well [FPP08].

Power-law graphs have a dense core that consists of nodes with high degrees. Core properties have been investigated for several power-law graph models. Having a small core whose removal substantially changes connectivity, would allow for a scheme that constructs shortest paths through this core based on a separator theorem, as for planar graphs [Tho04a] and for minor-free graphs [AG06]. However, the proportion of nodes that have to be removed to substantially change the connectivity of a power-law graph is linear with respect to the size of the graph [CNSW00, NSW01, NWS02, BR03, FFV05, NR08]. Therefore, a separator-based strategy is not suitable for power-law graphs; different techniques are necessary.

Also, most powerful techniques that work well for graphs with bounded doubling dimension cannot be used. Although sometimes claimed, the Internet does not appear to have bounded ball growth or bounded doubling dimension; both measures can be large [FLV08].

Practical routing schemes (and distance oracles) for power-law graphs have been proposed [BC06, RMJ07, PBCG09, CY09, GSVGM98, XWP$^+$09] (Section 3.2.4). However, there are no theoretical results on the space requirements of routing schemes for power-law graphs.

An approach related to routing is due to Kleinberg [Kle00]. He formally proves that, in the re-wired lattice model [BMST97, NW99, Kle00] (Section 2.1.3), *greedy routing* is a good routing scheme. Greedy routing intuitively means that edges on the path to the target are chosen one after another such that the estimated distance to the target is minimized. Kleinberg proves that paths have length $\mathcal{O}(\lg^2 n)$. For this greedy approach to work, it is however crucial that nodes know their *coordinates* within the lattice.[3] Unfortunately, we cannot transform the greedy routing scheme into a distance oracle, since there is no bound on the stretch for greedy routing; even for two nodes connected by a short path of constant length, the greedy route may have length $\mathcal{O}(\lg^2 n)$. The stretch is thus $(\mathcal{O}(\lg^2 n), 0)$.

Complex networks usually have diameter $\mathcal{O}(\lg n)$. For a graph with diameter Δ, a $(\Delta, 0)$–approximate distance oracle with constant space is trivial (by storing the diameter). If the oracle is required to output actual paths, linear space suffices (the preprocessing algorithm creates a shortest path starting at an arbitrary vertex; the query algorithm outputs paths on this tree).

The objective is to devise a distance oracle (or routing scheme) with constant stretch and good theoretical bounds on the space requirements.

5.2 Preliminaries

5.2.1 Distance Oracle of Thorup and Zwick

The construction algorithm of the distance oracle of Thorup and Zwick [TZ05] is based on the following ideas. Thorup and Zwick use random sampling to select a subset $S \subseteq V$ containing $\mathcal{O}(\sqrt{n})$ vertices. From each vertex of the set S, the algorithm computes and stores the distance to all the vertices in the graph. From all other vertices $v \in V \setminus S$, the algorithm computes the

[3] Without coordinates, paths may have length $\mathcal{O}(n^c)$ [DEH07a, DEH07b].

open ball around v until it "touches" the nearest vertex from S. For each v, the ball has expected size $\mathcal{O}(\sqrt{n})$. The balance between the sample size $|S|$ and the expected ball size is optimal, which yields the desired space complexity of $\mathcal{O}(n\sqrt{n})$. For details, see Algorithms 1 and 2 in Section 3.1.2.

5.2.2 Properties of Random Power-law Graphs

> Essentially, all models are wrong, but some are useful.
>
> *George P. E. Box* [BD86, p. 424]

We adapt the random graph model for fixed expected degree sequence as defined by Aiello, Chung, and Lu [ACL00, CL02, Lu02, CL06] using the definition from [CL02, Section 2]. We refer to the original random graph distribution using the expression Fixed Degree Random Graph (**FDRG**).

Definition 36 (Fixed Degree Random Graph [CL02, Section 2])**.** *In a random graph with a given* expected degree sequence $\vec{w} = \{w_1, w_2, \ldots, w_n\}$ *such that* $\forall i : w_i^2 < \sum_j w_j$, *the edge between* v_i *and* $v_{i'}$ *is present in the random graph with probability*

$$\Pr\left[\{v_i, v_{i'}\} \in E\right] = w_i w_{i'} \rho, \quad \textit{where} \quad \rho = \frac{1}{\sum_j w_j}.$$

In the original **FDRG** model it is assumed that $\forall i, i' : w_i w_{i'} < \sum_j w_j$. We adapt the original model by deterministically inserting edges if $w_i w_{i'} > \sum_j w_j$. Without modification, the original assumption would rule out the values for τ considered in this thesis.

Definition 37. *For a constant* $\tau \in (2, 3)$*, the random power-law graph distribution* **RPLG**(n, τ) *is defined as follows. Let the sequence of generating parameters* $\vec{w} = \{w_1, w_2, \ldots, w_n\}$ *obey a power law:*

$$w_j = \left(\frac{n}{j}\right)^{1/(\tau-1)} \quad \textit{for} \quad j \in \{1, 2, \ldots n\}.$$

The edge between v_i *and* $v_{i'}$ *is present in the random graph with probability*

$$\Pr\left[\{v_i, v_{i'}\} \in E\right] = \min\{w_i w_{i'} \rho, 1\}, \quad \textit{where} \quad \rho = \frac{1}{\sum_j w_j}.$$

Note that, in both models, there is a one-to-one correspondence between a node v_j and its generating parameter w_j. In the **FDRG** model, the value w_j corresponds to the expected degree of vertex v_j, and Chung and Lu refer to $\vec{w}$ as the *expected degree sequence*. In the **RPLG**(n, τ) adaptation, the graph is sampled according to the *generating parameter values* w_j. Let D_j be the random variable denoting the degree of node v_j. In the **RPLG**(n, τ) model, the expected degree $E[D_j]$ of node v_j is less than or equal to the generating parameter w_j. We refer to the edges between two nodes $v_i, v_{i'}$ with $w_i w_{i'} \geqslant \sum_j w_j$ as *deterministic edges*; we refer to the remaining edges as *random edges*.

An important reason to work with this model is that the edges are independent. This independence makes several graph properties easier to analyze. We also (implicitly) rely on a property called *assortativity*. Assortativity is the tendency of nodes with high degree to attach to other highly connected nodes. This tendency is especially high in social networks. The opposite tendency, termed *dissortativity*, is more common in technological and biological networks. Highly connected nodes tend to be connected with low degree nodes. Li et al. [LADW05, Definition 4.1] formalize assortativity as follows. They define the $s(G)$ value of a graph as $s(G) := \sum_{\{v_i,v_{i'}\}\in E} \deg(v_i)\cdot\deg(v_{i'})$. Graphs sampled from the **FDRG** model tend to have a high $s(G)$ value, since high-degree nodes are attached to other highly connected nodes. Li et al. state that $s(G)$ measures to what extent a graph has a "hub-like core."

The *core* of a graph consists of nodes having large degrees. Let $\gamma = \frac{\tau-2}{2\tau-3} + \varepsilon$ for some $\varepsilon > 0$ and $\gamma' = \frac{1-\gamma}{\tau-1}$.

Definition 38. *For a power-law degree sequence $\vec{w}$ and a graph G with n nodes, the* core *with degree threshold $n^{\gamma'}$, $\gamma' \in (0,1)$, is defined as follows.*

$$\begin{aligned} \mathsf{core}_{\gamma'}(\vec{w}) &:= \left\{v_j : w_j > n^{\gamma'}\right\}, \\ \mathsf{core}_{\gamma'}(G) &:= \left\{v_j : \deg_G(v_j) > n^{\gamma'}/4\right\}, \end{aligned}$$

where $\deg_G(v_j)$ is the degree of v_j in G (the subscript G is omitted when the graph is clear from the context).

The $\mathsf{core}_{\gamma'}(\vec{w})$ as defined here is the $n^{\gamma'}$–Core in [Lu02, Chapter 4, Definition 2]. Note that $\mathsf{core}_{\gamma'}(\vec{w})$ and $\mathsf{core}_{\gamma'}(G)$ are not necessarily equivalent. Even if the degree bound in $\mathsf{core}_{\gamma'}(G)$ was set to $n^{\gamma'}$ instead of $n^{\gamma'}/4$, the two cores would not be equal. In Section 5.4.1, we prove that $\mathsf{core}_{\gamma'}(\vec{w}) \subseteq \mathsf{core}_{\gamma'}(G)$ with high probability.

For each vertex u of a graph G, define its ball relative to the core (which is the open metric ball as in Definition 13) as

$$B_{\mathsf{core}}(u) := \left\{ v \in V(G) \,:\, d(u,v) < \min_{v' \in \mathsf{core}_{\gamma'}(G)} d(u,v') \right\}.$$

Note that it is important to use the open ball and not the closed ball.

The *volume* of a set of nodes is an integral notion in the proof. Let G be a random graph sampled from **RPLG**(n,τ). For a set of nodes S, define its *volume* $Vol(S)$ as the sum of all its nodes' w_j, that is, $Vol(S) := \sum_{v_j\in S} w_j$. We simplify notation by $Vol(G) := Vol(V)$. Note that $Vol(G) = 1/\rho$ (Definition 37). Let $vol_G(S)$ denote the sum of the nodes' degrees in the actual graph G, $vol_G(S) := \sum_{v_j\in S} \deg_G(v_j)$.

For our proof, the most important property of the **FDRG** model is captured in the following lemma, which is applied for the core and individual balls. There is an edge between two nodes $v_i, v_{i'}$ with probability proportional to $w_i \cdot w_{i'}$. The statement is extended to sets of nodes $S, T \subseteq V(G)$ in the following. The lemma holds for both **FDRG**$(\vec{w})$ and **RPLG**(n,τ).

Lemma 23 ([Lu02, Lemma 3.3, proof in Lemma 9])**.** *For any two disjoint subsets S and T with $Vol(S)\cdot Vol(T) > c \cdot Vol(G)$, we have*

$$\mathsf{Pr}[d(S,T) > 1] = \prod_{v_i\in S, v_{i'}\in T} \max\{0, (1 - w_i w_{i'}/Vol(G))\} \leqslant e^{-c}.$$

The following lemma proves that $Vol(G)$ is linear in n.

Lemma 24. *Let G be a random graph sampled from* $\mathbf{RPLG}(n, \tau)$*. The volume $\mathit{Vol}(G)$ satisfies*

$$n < \mathit{Vol}(G) \leqslant \frac{\tau-1}{\tau-2} n.$$

Proof.

- Lower bound: it holds that $\sum_j w_j > n$, since $w_j > 1$ for all $j < n$ and $w_n = 1$.
- Upper bound: it holds that

$$\mathit{Vol}(G) \quad = \quad \sum_{j=1}^{n} w_j < w_1 + \int\limits_1^n \left(\frac{n}{x}\right)^{1/(\tau-1)} \mathrm{d}x \leqslant \frac{\tau-1}{\tau-2} n.$$

□

In the remainder of the preliminaries section, we prove certain concentration properties of the adapted random power-law graph model. Since for the $\mathbf{RPLG}(n, \tau)$ model the edge probability is capped, several properties of graphs sampled according to the **FDRG** model do not hold for graphs sampled according to the $\mathbf{RPLG}(n, \tau)$ model.

In the following, we show concentration results for the actual degree of a vertex and for the volume of a set of vertices under the adapted $\mathbf{RPLG}(n, \tau)$ model. We also restate the corresponding results in the original **FDRG** model.

Lemma 25 ([CL06, Lemma 5.6], generalized from [McD98, Theorem 2.7])**.** *For a random graph sampled from* $\mathbf{FDRG}(\vec{w})$*, the random variable D_j measuring the degree of vertex v_j is concentrated around its expectation w_j as follows:*

$$\mathsf{Pr}[D_j > w_j - c\sqrt{w_j}] \quad \geqslant \quad 1 - e^{-c^2/2} \tag{5.1}$$

$$\mathsf{Pr}[D_j < w_j + c\sqrt{w_j}] \quad \geqslant \quad 1 - e^{-\frac{c^2}{2(1+c/(3\sqrt{w_j}))}} \tag{5.2}$$

Lemma 26 ([CL06, Lemma 5.9])**.** *For a random graph sampled from* $\mathbf{FDRG}(\vec{w})$*, for a subset of vertices S and for all $0 < c \leqslant \sqrt{\mathit{Vol}(S)}$,*

$$\mathsf{Pr}\left[|\mathit{vol}(S) - \mathit{Vol}(S)| < c\sqrt{\mathit{Vol}(S)}\right] \geqslant 1 - 2e^{-c^2/6}.$$

Weaker concentration bounds hold for graphs sampled from $\mathbf{RPLG}(n, \tau)$.

Lemma 27. *Let $n \geqslant 4^{\frac{\tau-1}{(\tau-2)^2}}$. For a random graph sampled from* $\mathbf{RPLG}(n, \tau)$*, if $w_j \geqslant 32 \ln n$, for vertex v_j, the degree D_j satisfies the following:*

$$\mathsf{Pr}[w_j/4 \leqslant D_j \leqslant 3w_j] > 1 - 2/n^4.$$

Proof. Recall that $\rho = 1/\mathit{Vol}(G) < 1/n$ (by Lemma 24).

For $1 \leqslant j \leqslant n$, let $h(j) \in \{1, 2, \dots n\}$ denote the smallest integer such that $\rho w_{h(j)} w_j \leqslant 1$.

Consider $h(1)$. Since $\rho w_1 \left(\frac{n}{n^{3-\tau}}\right)^{1/(\tau-1)} \leqslant 1$, we have that

$$h(1) \leqslant \lceil n^{3-\tau} \rceil.$$

Therefore, for all $1 \leqslant j \leqslant n$,

$$h(j) \leqslant h(1) \leqslant \lceil n^{3-\tau} \rceil.$$

We split the degree D_j into two parts: the contribution by edges to nodes $v_{j'}$ with $j' < h(j)$ and the contribution stemming from edges to nodes $v_{j''}$ with $j'' \geqslant h(j)$. When $h(j) \geqslant 1$, there are at least $h(j) - 1$ edges to nodes $v_{j'}$ with $j' < h(j)$. Now consider the edges between v_j and $v_{j'}$ for $j' \geqslant h(j)$. Since the sequence $\vec{w}$ is monotonically decreasing,

$$\begin{aligned}
\sum_{i=h(j)}^{n} w_i &\geqslant \int\limits_{n^{3-\tau}+1}^{n} (n/x)^{1/(\tau-1)} \mathrm{d}x \\
&\geqslant \frac{\tau-1}{\tau-2}\left(n - n^{1/(\tau-1)} 2^{\frac{\tau-2}{\tau-1}} n^{\frac{\tau-2}{\tau-1}(3-\tau)}\right) \quad (\text{since } n^{3-\tau} \geqslant 1) \\
&\geqslant \frac{\tau-1}{2(\tau-2)} n \quad (\text{since } n \geqslant 4^{\frac{\tau-1}{(\tau-2)^2}}).
\end{aligned}$$

Recall that $\rho = 1/\sum_{j=1}^{n} w_j \geqslant \frac{\tau-2}{n(\tau-1)}$ by Lemma 24.

Let D'_j denote the random variable counting the number of edges from v_j to $v_{j'}$ with $j' \geqslant h(j)$ in a random graph. Thus,

$$E[D'_j] = \mu = \rho w_j \sum_{i=h(j)}^{n} w_i \geqslant w_j/2 \geqslant 16 \ln n.$$

Also, $\mu \leqslant w_j$. Since there are no deterministic edges in this case, the random variable D'_j can be bounded using Lemma 25:

$$\begin{aligned}
\mathsf{Pr}[D'_j > \mu/2] &\geqslant 1 - e^{-\mu/4} \geqslant 1 - 1/n^4, \\
\mathsf{Pr}[D'_j < 2\mu] &\geqslant 1 - e^{-3\mu/8} \geqslant 1 - 1/n^4.
\end{aligned}$$

For $h(j) = 1$, the statement of the lemma follows directly.

If $h(j) > 1$, we have $D_j \leqslant D'_j + h(j) - 1$. Notice that $\rho w_j (n/w_j)^{1/(\tau-1)} \leqslant 1$, which implies that $h(j) \leqslant \lceil w_j \rceil \leqslant w_j + 1$. Therefore,

$$\mathsf{Pr}[w_j/4 \leqslant \mu/2 \leqslant D_j \leqslant 3w_j] \leqslant 1 - 2/n^4.$$

□

Lemma 28. *Let G be a random graph sampled from* $\mathbf{RPLG}(n, \tau)$*. For a subset of vertices S satisfying $\mathit{Vol}(S) \geqslant 192 \ln n$, it holds with probability at least $1 - 2/n^3$ that $\mathit{Vol}(S)/8 \leqslant \mathit{vol}(S) \leqslant 4\,\mathit{Vol}(S)$.*

Proof. We split S into two parts $S_1 := \{v_j \in S : w_j < 32 \ln n\}$ and $S_2 := S \setminus S_1$.

By Lemma 27,

$$\mathsf{Pr}[\,\mathit{Vol}(S_2)/4 \leqslant \mathit{vol}(S_2) \leqslant 3\,\mathit{Vol}(S_2)] \geqslant 1 - 2|S_2|/n^4.$$

For each vertex $v_j \in S_1$, $w_j < 32 \ln n$. Since no deterministic edges are attached to S_1, we can apply Lemma 26 to S_1.

Therefore, if $\mathit{Vol}(S_1) \geqslant 96 \ln n$, by Lemma 26,

$$\mathsf{Pr}[\,\mathit{Vol}(S_1)/2 \leqslant \mathit{vol}(S_1) \leqslant 2\,\mathit{Vol}(S_1)/3] \geqslant 1 - 2/n^4.$$

Therefore, the statement holds with probability at least $1 - 2(|S_2| + 1)/n^4 \geqslant 1 - 2/n^3$.

If $Vol(S_1) < 96 \ln n$, we have $Vol(S_2) \geqslant Vol(S)/2 \geqslant 96 \ln n$.
However, since

$$\Pr\left[vol(S_1) < \frac{3}{2} \cdot 96 \ln n \leqslant \frac{3}{4} Vol(S)\right] \geqslant 1 - 2/n^4,$$

we can still apply Lemma 26 to bound $vol(S_1)$ from above.

In this case, since

$$\Pr[Vol(S)/8 \leqslant Vol(S_2)/4 \leqslant vol(S_2) \leqslant 3\, Vol(S_2)] \geqslant 1 - 2|S_2|/n^4,$$

the statement also holds with probability at least $1 - 2/n^3$. □

In Lemma 24, we prove that $Vol(G) = \Theta(n)$. Under the adapted **RPLG**(n, τ) model, compared to the **FDRG** model, the edge probability may only decrease. We immediately have that $vol(G) = \mathcal{O}(n)$ with high probability. With the concentration results of this section, we obtain the following corollary.

Corollary 29. *The number of edges of a random graph sampled from* **RPLG**(n, τ) *is at most* $\frac{vol(G)}{2} \leqslant \frac{4(\tau-1)}{\tau-2} n$ *with probability at least* $1 - n^{-2}$.

5.3 The Adapted Distance Oracle

We propose a modification of the distance oracle of Thorup and Zwick [TZ05, Fig. 5] for stretch parameter $k = 2$, which guarantees stretch $(3, 0)$. The main idea of the scheme by Thorup and Zwick for $k = 2$ is the following (see Figure 5.2): in the preprocessing step, given a graph $G = (V, E)$:

1. Each node $v \in V$ is chosen as a *landmark* independently at random with probability $n^{-1/2}$. The expected number of landmarks is $\sqrt{n}$.

2. For each node $u \in V$, find its nearest landmark $\mathcal{L}(u)$ and compute the distances from u to all landmarks.

3. To guarantee optimal stretch for short distance queries, for every node $u \in V$ a local *ball* $B_G(u) = \{u' \in V(G) : d(u, u') < d(u, \mathcal{L}(u))\}$ is computed, including all nodes with distance strictly less than the distance to the landmarks.

The result of the distance query $d(s, t)$ is exact if $s \in B(t)$ or $t \in B(s)$ and otherwise stretch $(3, 0)$ is guaranteed [Cow01]. Since the set of landmarks consists of a random sample, the expected ball size is $\mathcal{O}(\sqrt{n})$, which is equal to the number of landmarks. This is the optimal balance for general graphs.

For power-law graphs a *better* balance is possible. Using high-degree nodes as landmarks is a natural heuristic. We can select fewer landmarks and obtain smaller sized balls than [TZ05, Fig. 5] at the same time.

It is required that $n = |V(G)|$ is sufficiently large, specifically, that

$$n^{\frac{\varepsilon(2\tau-3)}{\tau-1}} \geqslant \frac{2(\tau-1)}{\tau-2} \ln n. \tag{5.3}$$

The complexity results of this chapter do not have any other implicit dependencies on ε.

The following is the precise version of the main theorem of this chapter.

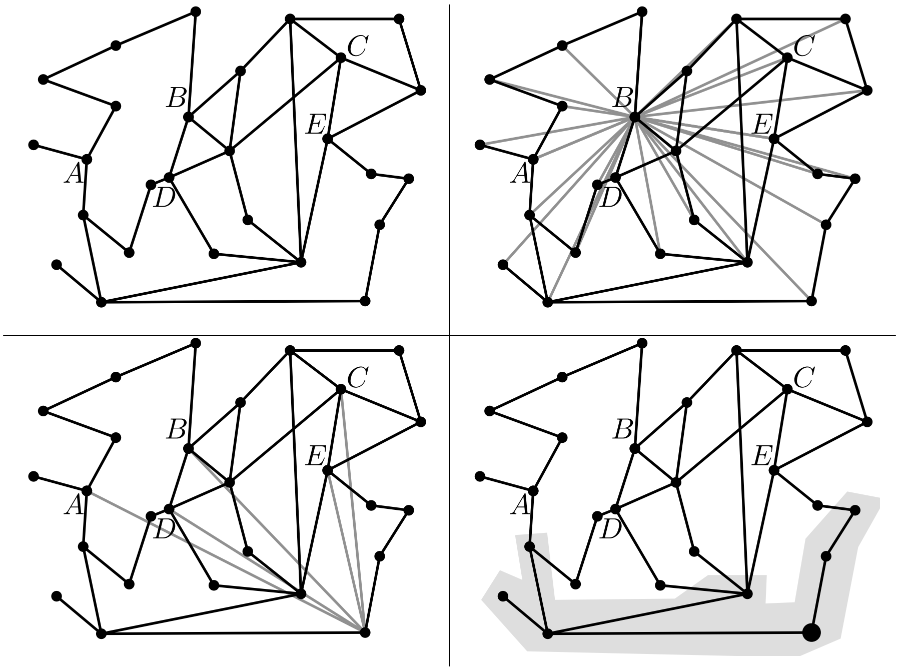

***Figure 5.2**: The distance oracle of Thorup and Zwick [TZ05, Fig. 5] for stretch parameter $k = 2$, which guarantees stretch $(3, 0)$. An illustration of the preprocessing algorithm, from left to right: (1) random sampling of landmarks, (2) SSSP computation for one landmark (node B), (3) knowledge of one node after all SSSP computations (from all landmarks), and (4) ball for the rightmost node in the bottom line*

Theorem 30. *Let $\gamma = \frac{\tau-2}{2\tau-3} + \varepsilon$ be a constant. Assume Equation (5.3) is satisfied. For random power-law graphs from* **RPLG**(n, τ) *(Definition 37), there exists a $(3, 0)$–approximate distance oracle with the following properties. The preprocessing algorithm runs in expected time $\mathcal{O}(n^{1+\gamma} \lg n)$ and creates a distance oracle of expected size $\mathcal{O}(n^{1+\gamma})$. These bounds also hold with probability at least $1 - 1/n$. After preprocessing, approximate distance queries can be answered in $\mathcal{O}(1)$ time with stretch at most $(3, 0)$.*

Since power-law graphs do not have large girth, the lower bound of Chapter 4 does not apply to power-law graphs. However, scale-free networks and expander graphs (which are the worst-case instances in Chapter 4) also share certain important properties [MPS06, Theorem 1, Corollary 4, and p. 247]. It is thus not clear whether space $\mathcal{O}(n^{1+\epsilon})$ is reasonably good or whether space $\widetilde{\mathcal{O}}(n)$ may be sufficient.

Details for the preprocessing step are listed in Algorithm 3. Analogous to the oracle of Thorup and Zwick [TZ05], for efficient query times, preprocessed information is stored in a hash table [FKS84] for each node.

The query algorithm is the same as in [TZ05] for $k = 2$, see Algorithm 4.

Lemma 31. *Algorithm 4 runs in time $\mathcal{O}(1)$ and achieves stretch $(3, 0)$.*

The following proof applies the same stretch and time bounds as [TZ05].

Proof. Time: At each node, all the information is stored in a hash table [FKS84] with constant access time. The number of hash table reads necessary is constant.

Algorithm 3 Preprocess $(G = (V, E), \gamma')$

```
compute core ← $\{v \in V : \deg(v) > n^{\gamma'}/4\}$
for each $v \in$ core do
  run breadth-first search from $v$ in $G$
  for each node $u \neq v$, store $d(u, v)$ and let FirstNode$_u(v)$ be the penultimum node on the shortest path; update $\mathcal{L}(u)$ if $v$ is nearest landmark
end for
for each $u \in V$ do
  compute and store $B_{\mathsf{core}}(u)$ (including distances)
  for each $v \in B_{\mathsf{core}}(u)$ let FirstNode$_u(v)$ be the first node on the shortest path to $v$.
end for
```

Algorithm 4 Distance (s, t)

```
if $s \in B_S(t)$ or $t \in B_S(s)$ then
  return local distance $d(s, t)$ from the information at $s$ or $t$.
else
  return $d(s, \mathcal{L}(t)) + d(\mathcal{L}(t), t)$
end if
```

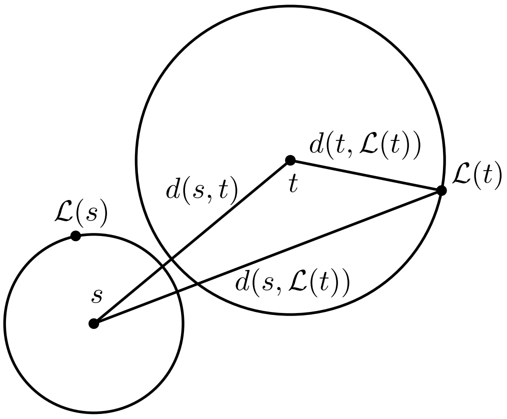

***Figure 5.3**: Illustration of the proof of worst-case stretch $(3, 0)$ using the triangle inequality.*

Stretch $(3, 0)$ is guaranteed by the following observation [Cow01] (for an illustration, see Figure 5.3). For a node $u \in V$, its ball is defined as follows.

$$B_{\mathsf{core}}(u) := \left\{ v \in V(G) \, : \, d(u, v) < \min_{v' \in \mathsf{core}_{\gamma'}(G)} d(u, v') \right\}$$

If neither $s \in B_{\mathsf{core}}(t)$ nor $t \in B_{\mathsf{core}}(s)$, then both

$$\begin{aligned} d(s, t) &\geqslant d(s, \mathsf{core}) = d(s, \mathcal{L}(s)) \text{ and} \\ d(s, t) &\geqslant d(t, \mathsf{core}) = d(t, \mathcal{L}(t)). \end{aligned}$$

Using the second inequality, the triangle inequality

$$d(s,\mathcal{L}(t)) \leqslant d(s,t) + d(t,\mathcal{L}(t)),$$

and $d(t,\mathcal{L}(t)) = d(\mathcal{L}(t),t)$ (since G is undirected), we have

$$d(s,\mathcal{L}(t)) + d(\mathcal{L}(t),t) \leqslant d(s,t) + d(t,\mathcal{L}(t)) + d(\mathcal{L}(t),t) \leqslant 3d(s,t).$$

In practice, the return value

$$\min\left\{d(s,\mathcal{L}(t)) + d(\mathcal{L}(t),t), d(s,\mathcal{L}(s)) + d(\mathcal{L}(s),t)\right\}$$

(or even $\min\limits_{\mathcal{L}\in\mathsf{core}} \{d(s,\mathcal{L}) + d(\mathcal{L},t)\}$) may yield better approximations (this triangulation is related to A* [GH05] and beacon-based embeddings [KSW09], see Section 3.2.2). □

5.4 Time and Space Complexities

The objective of this section is to prove the following lemma.

Lemma 32. *Let $\gamma = \frac{\tau-2}{2\tau-3} + \varepsilon$ be a constant. Assume Equation 5.3 is satisfied. For random power-law graphs* **RPLG**(n,τ)*, Algorithm 3 runs in expected time $\mathcal{O}(n^{1+\gamma}\lg n)$ and creates a distance oracle of expected size $\mathcal{O}(n^{1+\gamma})$. These bounds also hold with probability at least $1-1/n$.*

The main result of this chapter, Theorem 30 is immediate from Lemmas 31 and 32.

5.4.1 Core Size

We prove that the size of the core is $\Theta(n^\gamma)$ in expectation and with high probability. We also prove that it contains the nodes with high degree with high probability.

The core is defined by

$$\begin{aligned}\mathsf{core}_{\gamma'}(\vec{w}) &:= \left\{v_i : w_i > n^{\gamma'}\right\},\\ \mathsf{core}_{\gamma'}(G) &:= \left\{v_i : \deg_G(v_i) > n^{\gamma'}/4\right\}.\end{aligned}$$

To compute the size of $\mathsf{core}_{\gamma'}(\vec{w})$, we solve the inequality $w_k > n^{\gamma'}$ and obtain k.

$$\begin{aligned} w_k = \left(\frac{n}{k}\right)^{\frac{1}{\tau-1}} &> n^{\gamma'}\\ k^{-\frac{1}{\tau-1}} &> n^{\gamma'-\frac{1}{\tau-1}}\\ k &< n^{(1-\tau)(\gamma'-\frac{1}{\tau-1})} = n^{\gamma'(1-\tau)+1}\end{aligned}$$

As $\gamma' = \frac{1-\gamma}{\tau-1}$, we have

$$|\mathsf{core}_{\gamma'}(\vec{w})| = \lceil n^{\gamma'(1-\tau)+1}\rceil - 1 = \lceil n^{\gamma}\rceil - 1.$$

Even if the same degree threshold $n^{\gamma'}$ is used for $\mathsf{core}_{\gamma'}(\vec{w})$ and $\mathsf{core}_{\gamma'}(G)$, the two sets of nodes may differ. For a slightly smaller degree threshold $n^{\gamma'}/4$ (as in Definition 38), the core of the actual graph contains $\mathsf{core}_{\gamma'}(\vec{w})$ with high probability. The proof of the following theorem essentially consists of applying Lemma 27.

Lemma 33. *Let G be a random graph sampled from* $\mathbf{RPLG}(n, \tau)$*. With probability at least* $1 - 1/n^2$ *it holds that*

$$\mathsf{core}_{\gamma'}(\vec{w}) = \left\{v_i : w_i > n^{\gamma'}\right\} \subseteq \left\{v_i : \deg(v_i) > n^{\gamma'}/4\right\} = \mathsf{core}_{\gamma'}(G).$$

Proof. Let v_i be a vertex in $\mathsf{core}_{\gamma'}(\vec{w})$.

By Lemma 27, $D_i \geqslant n^{\gamma'}/4$ with probability at least $1 - 2/n^4$. This holds for all $j \leqslant i$.

Therefore, by applying the union bound, the probability that the core of the actual graph contains the nodes with high potential $\mathsf{core}_{\gamma'}(\vec{w}) \subseteq \left\{v_i : \deg(v_i) > n^{\gamma'}/4\right\}$ is at least $1 - 1/n^2$. □

Lemma 34. *Let G be a random graph sampled from* $\mathbf{RPLG}(n, \tau)$*. With probability at least* $1 - 1/n^2$*,*

$$|\mathsf{core}_{\gamma'}(G)| = \Theta(n^{\gamma}).$$

Proof.

- Lower bound: Since $\mathsf{core}_{\gamma'}(G)$ contains $\mathsf{core}_{\gamma'}(\vec{w})$ with probability at least $1-1/n^2$, its size is at least n^{γ} with at least the same probability.
- Upper bound: Let $i = 144n^{\gamma}$. By Lemma 27, $D_i \leqslant 3w_i < n^{\gamma'}/4$ with probability at least $1 - 2/n^4$. This holds for all $j \in (i, n]$. By union bound, $\mathsf{core}_{\gamma'}(G)$ does not contain any vertex v_j for $i \leqslant j \leqslant n$, with probability at least $1 - 1/n^2$, which implies $|\mathsf{core}_{\gamma'}(G)| \leqslant 144n^{\gamma}$ with probability at least $1 - 1/n^2$.

□

5.4.2 Ball Sizes

We prove that the expected ball size is small. This section contains the main technical idea, which is the application of Lemma 23.

Let G be a random graph sampled from $\mathbf{RPLG}(n, \tau)$. Recall that a ball is defined by

$$B_{\mathsf{core}}(u) = \left\{v \in V(G) \ : \ d(u, v) < \min_{v' \in \mathsf{core}_{\gamma'}(G)} d(u, v')\right\}.$$

Lemma 35. *Let* $\mathsf{b} = \gamma'(\tau - 2) + \frac{(2\tau-3)\varepsilon}{\tau-1}$ *be a constant. Assume Equation (5.3) is satisfied. For a random graph G sampled from* $\mathbf{RPLG}(n, \tau)$*, with probability at least $1 - 3/n^2$, it holds that for all $u \in V(G)$,*

$$\begin{aligned} |B_{\mathsf{core}}(u)| &= |\{u' \in V(G) : d(u, u') < d(u, \mathsf{core}_{\gamma'}(\vec{w}))\}| = \mathcal{O}(n^{\mathsf{b}}), \\ |E(B_{\mathsf{core}}(u))| &= \mathcal{O}(n^{\mathsf{b}} \lg n), \end{aligned}$$

where $E(B_{\mathsf{core}}(u))$ is the set of internal edges among vertices in $B_{\mathsf{core}}(u)$.

Since for $\mathbf{RPLG}(n, \tau)$ the edges are independent, in our analysis, the existence of every edge in random graph G is only determined when it is needed, and before that it is treated as a probability distribution as defined in our random graph model. We call the determination of the existence of an edge according to its probability distribution *revealing* the edge. For a given vertex $u \in V(G)$, we define a *sequence of balls* $(B_0 = \{u\}, B_1, B_2, \ldots)$ as follows:

- Let $V' = V \setminus \mathsf{core}_{\gamma'}(\vec{w})$.
- Now define $B_0 = \{u\}$ and $B_i = \{v \ : \ d_G(u, v) \leqslant i\}$.

- We also define the *circles* $C_i = B_i \setminus B_{i-1}$ for $i \geqslant 0$ with $B_{-1} = \emptyset$. Let E_i denote the random variable counting the number of edges between C_i and $C_i \cup C_{i+1}$.

We first give a concentration result on E_i.

Lemma 36. *For circle C_i, the following holds with probability at least $1 - 2/n^3$:*

$$\begin{aligned} \text{If } Vol(C_i) &< 192\ln n, \text{ then } E_i \leqslant 4 \cdot 192 \ln n, \text{and} \\ \text{if } Vol(C_i) &\geqslant 192\ln n, \text{ then } E_i \leqslant 4\,Vol(C_i). \end{aligned}$$

If $Vol(C_i) < 192 \ln n$, then $E_i \leqslant 4 \cdot 192 \ln n$, and if $Vol(C_i) \geqslant 192 \ln n$, then $E_i \leqslant 4\,Vol(C_i)$.

Proof. For our analysis, we assume that the edges of the random graph are revealed in consecutive steps as follows: in step i with $i \geqslant 0$, edges from C_i to $V' \setminus B_{i-1}$ are revealed and circle C_{i+1} is formed. In other words, when discovering C_i, the edges between C_i and $V'' = V' \setminus B_{i-1}$ have not been revealed yet.

In particular, E_i measures the number of edges between C_i and V'' under the condition that we know all edges adjacent to B_{i-1}. We can define another random graph G' on the vertex set V'', such that the edge between two vertices in V'' is sampled with the same probability as in **RPLG**(n, τ). Now, E_i and $vol_{G'}(C_i)$ have the same distribution, where $vol_{G'}(C_i)$ denotes the number of edges adjacent to C_i in G'.

Let $vol(C_i)$ denote the random variable measuring the number of edges adjacent to C_i for the original model **FDRG**. $vol_{G'}(C_i)$ is *stochastically dominated* by $vol(C_i)$. Hence, the statement of the lemma follows directly, since it applies to $vol(C_i)$ by Lemma 28. □

Since there are at most n circles, Lemma 36 holds for all circles with probability at least $1 - 2/n^2$.

The above arguments are combined to prove Lemma 35.

Proof of Lemma 35. Let k be the smallest integer such that $Vol(B_k) \geqslant n^{\mathsf{b}}$. We have the conditions

- $Vol(B_k) \geqslant n^{\mathsf{b}}$,
- $Vol(\mathsf{core}_{\gamma'}(\vec{w})) \geqslant |\mathsf{core}_{\gamma'}(\vec{w})| n^{\gamma'} = n^{\gamma+\gamma'}$, and
- $Vol(G) \leqslant \frac{\tau-1}{\tau-2} n$ (Lemma 24).

From Equation (5.3),

$$n^{\mathsf{b}-\gamma'(\tau-2)} > 2\frac{\tau-1}{\tau-2}\ln n.$$

Since the edges between B_k and $\mathsf{core}_{\gamma'}(\vec{w})$ have not been revealed yet, Lemma 23 can be applied. Due to Lemma 23, there is an edge between B_k and $\mathsf{core}_{\gamma'}(\vec{w})$ with probability at least $1 - 1/n^2$. By Lemma 33, $\mathsf{core}_{\gamma'}(\vec{w}) \subseteq \mathsf{core}_{\gamma'}(G)$ with probability at least $1 - 1/n^2$. Hence $B_{\mathsf{core}}(u) \subseteq B_k$ with probability at least $1 - 2/n^2$.

In the following, we bound the size of B_k. Lemma 36 holds for all circles with high probability. In our case,

$$Vol(C_{k-1}) \leqslant Vol(B_{k-1}) < n^{\mathsf{b}}.$$

By Lemma 36, $|C_k| \leqslant E_{k-1} \leqslant 4n^{\mathsf{b}}$ with probability at least $1 - 1/n^2$. Then,

$$|B_k| = |B_{k-1}| + |C_k| \leqslant Vol(B_{k-1}) + |C_k| \leqslant 5n^{\mathsf{b}}.$$

Since $B_{\mathsf{core}}(u) \subseteq B_k$ with probability at least $1 - 2/n^2$, we have

$$|E(B_{\mathsf{core}}(u))| = \mathcal{O}(vol(B_{k-1}(u))) = \mathcal{O}\left(\sum_{i=0}^{k-1} E_i\right)$$

with probability at least $1 - 2/n^2$.

By Lemma 36, with probability at least $1 - 1/n^2$, for all i,

$$E_i \leqslant 4 \cdot 192 \ln n + 4\,Vol(C_i).$$

Since $k \leqslant n^{\mathsf{b}}$, with probability at least $1 - 3/n^2$,

$$\begin{aligned} |E(B_{\mathsf{core}}(u))| &= \mathcal{O}\left(\sum_{i=0}^{k-1} E_i\right) \\ &= \mathcal{O}(4 \cdot 192 n^{\mathsf{b}} \ln n + 4\,Vol(B_{k-1})) \\ &= \mathcal{O}(n^{\mathsf{b}} \lg n). \end{aligned}$$

This concludes the proof. □

5.4.3 Assembly

The core $\mathsf{core}_{\gamma'}(G)$ has size $\Theta(n^\gamma)$ with probability at least $1 - 1/n^2$ (Lemma 34) and all balls $B_{\mathsf{core}}(u)$ have size $\mathcal{O}(n^\gamma)$ with probability at least $1 - 3/n^2$ (Lemma 35). Therefore, we have the following result.

Proof of Lemma 32. Our algorithm is deterministic. The expected time (space) complexity is the average running time (space) of our algorithm over all graphs from the random graph distribution $\mathbf{RPLG}(n, \tau)$.

Given a graph G with n nodes and m edges, our algorithm computes the core $\mathsf{core}_{\gamma'}(G)$ of G with time complexity $\mathcal{O}(m + n \lg n)$. It runs a complete breadth-first search for each node of the core in time $\mathcal{O}(m)$. Due to the condition of Lemma 32, Equation (5.3) is satisfied. Let $B_{\mathsf{core}}(u)$ denote the ball computed in our algorithm for vertex u. Let $T(B_{\mathsf{core}}(u))$ denote the time to compute $B_{\mathsf{core}}(u)$. Therefore, the time complexity TC and space complexity SC of our algorithm are at most

$$TC(G) = \mathcal{O}\left(m \cdot |\mathsf{core}_{\gamma'}(G)| + \sum_{v \in V(G)} T(B_{\mathsf{core}}(u))\right), \tag{5.4}$$

$$SC(G) = \mathcal{O}\left(n \cdot |\mathsf{core}_{\gamma'}(G)| + \sum_{v \in V(G)} |B_{\mathsf{core}}(u)|\right). \tag{5.5}$$

Let $\mathsf{b} = \gamma'(\tau - 2) + \frac{(2\tau-3)\varepsilon}{\tau-1}$. By Lemma 35, SC is at most $\mathcal{O}(n^{1+\mathsf{b}})$ with probability at least $1-3/n^2$. The time to compute $B_{\mathsf{core}}(u)$ is linear in the number of internal edges in $B_{\mathsf{core}}(u)$, since the graph is unweighted and the distance from u to the core has been determined before computing $B_{\mathsf{core}}(u)$. By Lemma 35, $TC = \mathcal{O}(n^{1+\mathsf{b}})$ with probability at least $1 - 3/n^2$.

We now know that with probability at least $1 - 5/n^2$, all of the following conditions are true:

1. $m = \Theta(n)$ (Corollary 29);

2. $|\mathsf{core}_{\gamma'}(G)| = \Theta(n^{\gamma})$ (Lemma 34);

3. $|B_{\mathsf{core}}(u)| = \mathcal{O}(n^{\mathsf{b}})$ for all vertices u (Lemma 35);

4. $T(B_{\mathsf{core}}(u)) = \mathcal{O}(n^{\mathsf{b}} \lg n)$ for all vertices u (Lemma 35).

Therefore, from Equations 5.4 and 5.5, we know that with probability at least $1 - 5/n^2$, the space complexity of our algorithm is $\mathcal{O}(n^{1+\gamma} + n^{1+\mathsf{b}})$ and the time complexity is $\mathcal{O}(n^{1+\gamma} + n^{1+\mathsf{b}} \lg n)$.

Finally, we fix the parameters to obtain a balanced scheme. In a balanced scheme, the core size and the expected ball sizes are asymptotically equivalent, that is, $\mathsf{b} = \gamma$. Together with

$$\mathsf{b} = \gamma'(\tau - 2) + \frac{(2\tau - 3)\varepsilon}{\tau - 1}$$

and $\gamma' = \frac{1-\gamma}{\tau-1}$, we have

$$\gamma = \frac{\tau - 2}{2\tau - 3} + \varepsilon.$$

Therefore, assuming that Equation (5.3) is satisfied, the space requirement per node is $\mathcal{O}(n^{\gamma} \lg n)$ bits and the total preprocessing time is bounded by $\mathcal{O}(n^{1+\gamma} \lg n)$, which holds with probability at least $1 - 1/n$. □

5.5 Conclusion and Open Problems

Theorem 30 implies that distances and shortest paths in random power-law graphs can be approximated efficiently. For power-law exponents close to 2, the expected space consumption is close to linear. The extension of the algorithms for distance oracles to compact routing schemes indicates that the routing scheme by Thorup and Zwick [TZ01] may be very efficient on Internet-like network topologies.

Edge-weighted Graphs

The algorithm and the proof currently only apply to unweighted graphs. It seems difficult to extend our distance oracle to graphs with worst-case weights. An adversary could assign all edges within the core and within the fringe (all nodes outside the core) infinitesimal values, and edges between the core and the fringe to large values. With these weight values, all balls of nodes in the fringe span the whole fringe. Since the fringe size is linear in the number of nodes, the distance oracle would have $\mathcal{O}(n^2)$ space. It may be interesting to investigate the case where weights are random.

General Stretch Parameter k

Currently, the adaptation of the distance oracle of Thorup and Zwick [TZ05] works for the stretch parameter $k = 2$ only. An extension to general k seems feasible but with our proof technique, the space requirements for constant k would remain $\mathcal{O}(n^{1+\epsilon})$ for some $\epsilon > 0$.

Stretch

An important open question targets the stretch of the distance oracle. If a distance oracle is used as a component to distinguish between close and far entities in a complex network, the stretch is very important. Since the diameter of random power-law graphs is $\mathcal{O}(\lg n)$, the stretch should be as small as possible to guarantee meaningful estimates. While stretch $(3, 0)$ is best possible for general graphs and distance oracles with $\mathcal{O}(n^{3/2})$ space, we prove (Theorem 30) that it is possible to significantly reduce the space for power-law graphs. For Erdős-Rényi random graphs, stretch $(2, 0)$ has been achieved with space consumption of $\widetilde{\mathcal{O}}(n^{7/4})$ [EWG08] (see Section 3.1.3). Is it possible to reduce both space and stretch?

Different Models

As mentioned in Section 2.1.3, there are many different models for complex networks. Our analysis only works for the adapted random graph model by Aiello, Chung, and Lu [ACL00]. Other models such as the configuration model [BBK72] and the preferential attachment model [BA99] may also have efficient distance oracles.

The best theory is inspired by practice and the best practice is inspired by theory.

Donald Knuth [Knu89]

6 Approximating Shortest Paths Using Voronoi Duals

6.1 Introduction

The main result of this chapter is an approximation method to answer shortest path queries in general, undirected graphs with positive edge weights, based on random sampling and graph Voronoi duals [Meh88, Erw00]. In preprocessing, each node is selected as a Voronoi site independently at random with probability p, and the Voronoi dual is computed for the selected sites (Section 6.3). This preprocessing step is very efficient; it takes time proportional to computing one single source shortest path tree (Section 6.4). For $p < 1$, the resulting dual graph is expected to be smaller than the original graph. At query time, search for the shortest path from source s to target t can potentially be done faster in the Voronoi dual. We let the shortest path in the Voronoi dual guide the search for an approximate shortest path in the original graph. We prove that the expected approximation ratio is at most logarithmic in the number of nodes on the actual shortest path, and that this bound is tight (Section 6.5). Our experimental results show that, in practice, the approximation is much better than the stated theoretical bound and that the preprocessing overhead is indeed extremely low (Section 6.6).

Many practical shortest path query methods are tailored for road networks (Section 3.2.3). There has been considerable recent progress: for the road networks of Europe or the USA, using a high-performance computer, a speedup of several orders of magnitude compared to Dijkstra's algorithm can be achieved with a preprocessing time in the tens of minutes [DSSW09]. Unfortunately, theoretical bounds on both query time and preprocessing time are often difficult to obtain. However, even though road networks constitute the most common and popular application of shortest path query algorithms to date, other challenging applications exist. Computer networks, social networks, protein interaction networks, and the web graph exhibit different degree and structural properties, and may contain hundreds of millions or even billions of nodes. In specific cases, a user might be willing to trade preprocessing time against exactness due to the vast size of the data or due to restricted processing power (Section 1.2.2). These scenarios may require the use of a fast approximation method.

Related methods. Kambara and Ueshima [KU08] independently propose a method (without analysis) that appears to be closely related to the method we present in this chapter. Fang et al. [FGG$^+$05] use graph Voronoi diagrams for routing in sensor networks. Yu et al. [YWD08] use Voronoi paths to bridge communication gaps in sparse sensor networks. Chan and Efrat [CE01] solve the *cheapest path problem* for flight connections in $\mathbb{R}^2$. Their method runs Dijkstra's algorithm on the Delaunay triangulation with respect to a *superquadratic* cost function $\mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}^+$.

6.2 Preliminaries

6.2.1 Graph Voronoi Diagram

The classical Voronoi diagram is a distance-based decomposition of a metric space relative to a discrete set, the Voronoi sites [Dir50, Vor07].[1] For a survey on this fundamental structure, we refer to [Aur91]. Among many applications, the Voronoi diagram is often used to solve facility location problems [Sha75, ACS99, AGK+04, GKP05, Svi08, Sug09]. The Voronoi diagram and the Delaunay triangulation of n points in the plane can be computed in expected time $n \cdot 2^{\mathcal{O}(\sqrt{\lg \lg n})}$ [CP07], which is even faster than $\mathcal{O}(n \lg n)$.

Mehlhorn [Meh88] and Erwig [Erw00] proposed an analogous decomposition, the *Graph Voronoi Diagram*, for undirected and directed graphs respectively. Since the Voronoi diagram for the Euclidean space is used for various applications, its graph counterpart, the graph Voronoi diagram, may be used for these applications if the underlying metric is the shortest path metric of a graph [OSF+08].

Real-world distances or travelling times can be approximated more appropriately using models based on weighted graphs. In general, non-planar networks such as social networks, computer networks, protein interaction networks, and the web graph cannot be embedded into a low-dimensional Euclidean space without significant distortion.

Definition 39 (Graph Voronoi Diagram [Meh88, Erw00])**.** *In a graph $G = (V, E, \mathbf{w})$, the* Voronoi diagram *for a set of nodes $K = \{v_1, \dots, v_k\} \subseteq V$ is a disjoint partition* $\mathsf{Vor}_{(G,K)} := \{V_1, \dots, V_k\}$ *of V such that for each node $u \in V_i$, $d(u, v_i) \leqslant d(u, v_j)$ for all $j \in \{1, \dots, k\}$.*

The V_i are called *Voronoi regions*. The graph Voronoi diagram is not necessarily unique, as a node u may have the same distance to more than one Voronoi node. Let $\mathsf{vor}(u)$ denote the index i of the Voronoi region V_i containing u; that is, $\mathsf{vor}(u) = i \Leftrightarrow u \in V_i$.

Analogously to the Delaunay triangulation dual for classical Voronoi diagrams of point sets, we define the Voronoi dual for graphs.

Definition 40. *Let $G = (V, E, \mathbf{w})$ be an edge-weighted graph and* $\mathsf{Vor}_{G,K}$ *its Voronoi diagram. The* Voronoi dual *is the graph $G^* = (K, E^*, \mathbf{w}^*)$ with edgeset $E^* := \{(v_i, v_j) : v_i, v_j \in K$ and $\exists u \in V_i \wedge \exists w \in V_j : (u, w) \in E\}$, and edge weights*

$$\mathbf{w}^*(v_i, v_j) := \min_{\substack{u \in V_i, w \in V_j \\ (u,w) \in E}} \{d(v_i, u) + \mathbf{w}(u, w) + d(w, v_j)\}.$$

By contracting edges on the shortest paths connecting Voronoi nodes, one can see that G^* is a minor of G (see for example Wolff [Wol08, Lemma 4]; minors are defined in Definition 21).

Figure 6.1 illustrates a Voronoi diagram and a graph Voronoi diagram. Although the classical Voronoi dual of a non-degenerate set of points in the plane is always a triangulation, the graph Voronoi dual is not necessarily a triangulation, even for planar graphs. For example, a graph Voronoi dual may have nodes whose removal would disconnect the graph.

[1] More folklore in the style of Erdős-numbers: according to the *Mathematics Genealogy Project*, available online at `genealogy.math.ndsu.nodak.edu`, there is a tree path in the advisor graph from Dirichlet to my mentor and collaborator on the Voronoi method, Michael E. Houle: Gustav Peter Lejeune Dirichlet – Rudolf Otto Sigismund Lipschitz – Christian Felix Klein – Carl Louis Ferdinand Lindemann – Arnold Johannes Wilhelm Sommerfeld – Ernst Adolph Guillemin – Samuel Jefferson Mason – Robert Wellington Donaldson – Godfried Theodore Patrick Toussaint – Michael Edward Houle.

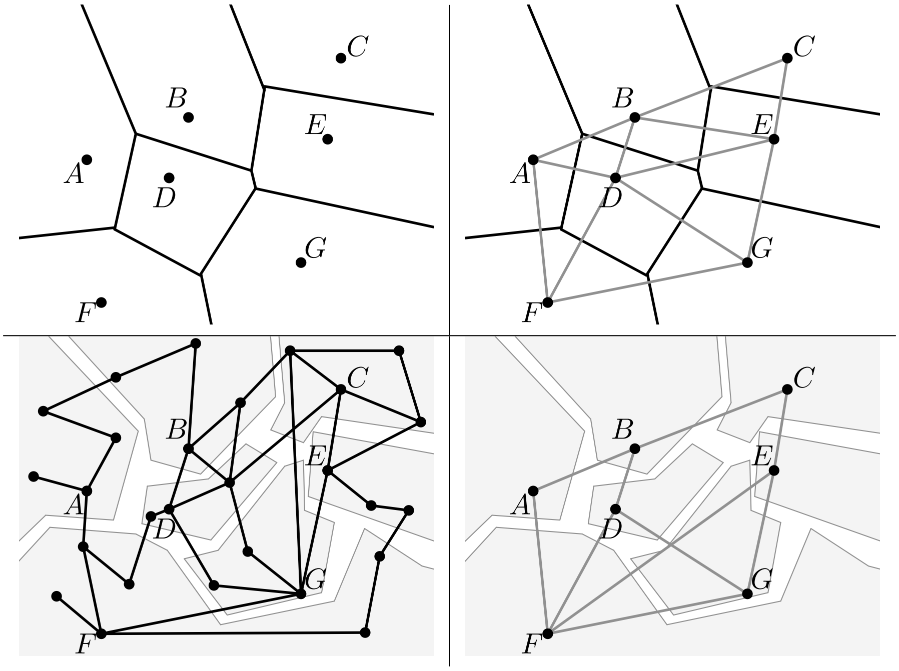

Figure 6.1: The Voronoi diagram and the Delaunay triangulation of the plane for a set of Voronoi sites $\{A, B, \ldots G\}$ and the graph Voronoi diagram and its dual for a set of Voronoi nodes $\{A, B, \ldots G\}$ in an unweighted graph (note that the graph Voronoi dual is not necessarily a triangulation).

Erwig [Erw00, Theorem 2] showed that the graph Voronoi diagram can be constructed with a single Dijkstra search in time $\mathcal{O}(m + n \cdot \lg n)$. A heap is used to store the shortest path distances from nodes to their closest Voronoi node. The heap is initialized to store the Voronoi nodes themselves. Thereafter, as long as there are nodes in the queue, the minimum is extracted from the heap and processed (or 'settled') by assigning to it a Voronoi region, storing the distance to its Voronoi node, and adding to or updating its neighbors in the queue. We slightly modify this construction of the Voronoi *diagram* [Erw00, Section 3.1] to compute the Voronoi *dual* — that is, to also compute E^* and $\mathbf{w}^*$. Whenever a node u is settled in the Dijkstra search, for all its settled neighbors u' of different Voronoi regions, the edge $(v^*_{\mathsf{vor}(u)}, v^*_{\mathsf{vor}(u')})$ with weight $\mathbf{w}_{G^*}(v^*_{\mathsf{vor}(u)}, v^*_{\mathsf{vor}(u')}) = d_G(v_{\mathsf{vor}(u)}, u) + \mathbf{w}_G(u, u') + d_G(u', v_{\mathsf{vor}(u')})$ is added, or its length is decreased if there already is an edge in G^* representing a longer path in G. This modification of Erwig's algorithm is shown as Algorithm 5.

In the analysis to follow (in Section 6.5) we move back and forth between a graph and its dual. For this we need the following definitions.

Definition 41. *Given a path $P = (u_0, u_1, \ldots, u_h)$, the* Voronoi path *of P is the sequence of vertices $P^* = (v_{\mathsf{vor}(u_0)}, v_{\mathsf{vor}(u_1)}, \ldots, v_{\mathsf{vor}(u_h)})$.*

Note that the Voronoi path P^* may not necessarily be simple, as multiple consecutive occurrences of nodes $v_{\mathsf{vor}(u_i)}$ are possible in P^*. They are treated as a single occurrence, and such paths are deemed to be equivalent.

Lemma 37. *For any path $P = (u_0, \ldots, u_h)$ in an undirected graph $G = (V, E, \mathbf{w})$, the corresponding Voronoi path P^* exists and is unique.*

Algorithm 5 ComputeVoronoiDual$(G = (V, E), K \subseteq V)$

```
$V' := V \cup \{v_0\}$, $E' := E$
$i := 1$
for $u \in K$ do
    $v_i := u$
    $\mathsf{vor}(v_i) := i$
    $E' := E' \cup \{\{v_0, u\}\}$
    $i := i + 1$
end for
HEAP.put$(v_0)$
while $\neg$HEAP.empty do
    $u_{\mathsf{cur}}$ := HEAP.extractMin
    for $u \in \Gamma(u_{\mathsf{cur}})$ do
        if $\mathsf{vor}(u) = \mathit{undefined}$ then
            $\mathsf{vor}(u) := \mathsf{vor}(u_{\mathsf{cur}})$
            HEAP.insert$(u, d(v_0, u_{\mathsf{cur}}) + \mathbf{w}(u_{\mathsf{cur}}, u))$
        else if $d(v_0, u_{\mathsf{cur}}) + \mathbf{w}(u_{\mathsf{cur}}, u) < d(v_0, u)$ then
            $\mathsf{vor}(u) := \mathsf{vor}(u_{\mathsf{cur}})$
            HEAP.decreaseKey$(u, d(v_0, u_{\mathsf{cur}}) + \mathbf{w}(u_{\mathsf{cur}}, u))$
        else if $\neg$HEAP.contains$(u)$ and $\mathsf{vor}(u) \neq \mathsf{vor}(u_{\mathsf{cur}})$ then
            if $(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) \notin E^*$ then
                $E^* := E^* \cup \{(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)})\}$
                $\mathbf{w}^*(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) := \infty$
            end if
            if $\mathbf{w}^*(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) > d(v_{\mathsf{vor}(u_{\mathsf{cur}})}, u_{\mathsf{cur}}) + \mathbf{w}(u_{\mathsf{cur}}, u) + d(u, v_{\mathsf{vor}(u)})$ then
                $\mathbf{w}^*(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) := d(v_{\mathsf{vor}(u_{\mathsf{cur}})}, u_{\mathsf{cur}}) + \mathbf{w}(u_{\mathsf{cur}}, u) + d(u, v_{\mathsf{vor}(u)})$
            end if
        end if
    end for
end while
```

Proof. Suppose that there is no such path P^* in G^*. This implies that there exist pairs of nodes u_i, u_{i+1} on the path P for which $v_{\mathsf{vor}(u_i)} \neq v_{\mathsf{vor}(u_{i+1})}$ and $(v_{\mathsf{vor}(u_i)}, v_{\mathsf{vor}(u_{i+1})}) \notin E^*$. As u_i, u_{i+1} are consecutive nodes on the path P, we know that $(u_i, u_{i+1}) \in E$. This contradicts the definition of the Voronoi dual (Def. 40), since $(u_i, u_{i+1}) \in E$ and $v_{\mathsf{vor}(u_i)} \neq v_{\mathsf{vor}(u_{i+1})}$ together imply that $(v_{\mathsf{vor}(u_i)}, v_{\mathsf{vor}(u_{i+1})}) \in E^*$. P^* is unique since each node u_i on the path belongs to exactly one Voronoi region, corresponding to exactly one Voronoi node $v_{\mathsf{vor}(u_i)}$. □

Definition 42. *For a path P^* in the Voronoi dual G^* of a graph G, the* Voronoi sleeve *is the subgraph of G induced by the nodes in the union of all Voronoi regions V_i for which its Voronoi node v_i lies on P^*,*

$$\mathsf{Sleeve}_{(G,G^*)}(P^*) := G\left[\bigcup_{v_i \in P^*} V_i\right].$$

The Voronoi sleeve is related to a subgraph sometimes termed *corridor*.

With the definitions at hand we can now state the approximation method.

6.3 The Voronoi Method

This section describes the preprocessing and query algorithms of the Voronoi method. Both algorithms are conceptually very simple and thus easy to implement.

In *preprocessing*, each node is selected as a Voronoi site independently at random with probability p, and the Voronoi dual is computed for the selected sites (Algorithm 6). For the sake of exposition, we treat the computation of the Voronoi dual as a 'black box', denoted by `ComputeVoronoiDual`.

Algorithm 6 Preprocessing

```
Input: graph $G = (V, E, \mathbf{w})$, sampling rate $p \in [0, 1]$.
Output: Voronoi dual $G^*$ with Voronoi nodes selected independently at random with probability $p$.

1: Random sampling: Generate the set of Voronoi nodes by selecting each node of $V$ independently at random: $\forall v \in V, \mathsf{Pr}[v \in K] = p$.
2: Compute a Voronoi dual $G^* = (K, E^*, \mathbf{w}^*)$ using the modified version of Erwig's algorithm [Erw00, Section 3.1] as shown in Algorithm 5.
   $G^*$:=ComputeVoronoiDual$(G, K)$
3: Return $G^*$.
```

Lemma 38. *For a graph $G = (V, E)$ with $n := |V|$ and $m := |E|$, Algorithm 6 takes time proportional to that of Dijkstra's single source shortest path algorithm.*

Proof. Erwig's variant of Dijkstra's algorithm computes the graph Voronoi diagram in a worst-case time proportional to Dijkstra's algorithm [Erw00, Theorem 2]. The only modification of Algorithm 5 compared to Erwig's variant is the following: for each node, at the time it is settled, all its neighbors are inspected. Therefore, each edge is additionally considered two times in total. This yields the same asymptotic running time. □

The preprocessing time complexity is proportional to the cost of computing one single source shortest path tree. Details are discussed in Section 6.4.

At *query* time, given a graph G and its Voronoi dual G^*, we answer (approximate) shortest path queries between source s and target t, by first searching for a shortest path $SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$ in the smaller Voronoi dual G^*. This path determines the sleeve $\mathcal{S} = \mathsf{Sleeve}(SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}))$, whose shortest path $SP_{\mathcal{S}}(s, t)$ approximates the shortest path $SP_G(s, t)$ in G. The shortest path in the Voronoi dual guides the Dijkstra search in the original graph. For a pseudo-code description, see Algorithm 7; for an illustration, see Figure 6.2.

The running time of Algorithm 7 depends on G and p. Let N^* and M^* denote the random variables measuring the number of nodes and edges of the Voronoi dual. Clearly $E[N^*] = p \cdot n$. The expected query time *without* refinement (computing the shortest path in the Voronoi sleeve) is at most $\mathcal{O}(N^* \lg N^* + M^*)$. The time for the refinement step depends on the size of the Voronoi sleeve. The analysis will show that the refinement step is not necessary for the approximation ratio to hold for long distance queries; however, it makes a practical difference for the quality of paths. For $p = \mathcal{O}(n^{-2/3})$, $E[N^*] = \mathcal{O}(n^{1/3})$, and thus we can afford to compute all-pairs shortest path distances in the Voronoi dual G^* in overall linear expected time. This allows for constant-time approximate distance queries.

Algorithm 7 Query

```
Input: Graph $G$, Voronoi dual $G^*$, Source $s$, Target $t$.
Output: an approximate shortest path $P$ from $s$ to $t$.

Find Voronoi source $v_{\mathsf{vor}(s)}$ from $s$ and Voronoi target $v_{\mathsf{vor}(t)}$ from $t$. If thereby a shortest path $SP_G(s,t)$ has been found, return it.
Compute a shortest path from $v_{\mathsf{vor}(s)}$ to $v_{\mathsf{vor}(t)}$ in the Voronoi dual $G^*$: $SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$.
Compute the Voronoi sleeve
    $$\mathcal{S} := \mathsf{Sleeve}(SP_{G^*}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})).$$
Compute a shortest path from $s$ to $t$ in the Voronoi sleeve, $SP_{\mathcal{S}}(s,t)$.
Return $P = SP_{\mathcal{S}}(s,t)$.
```

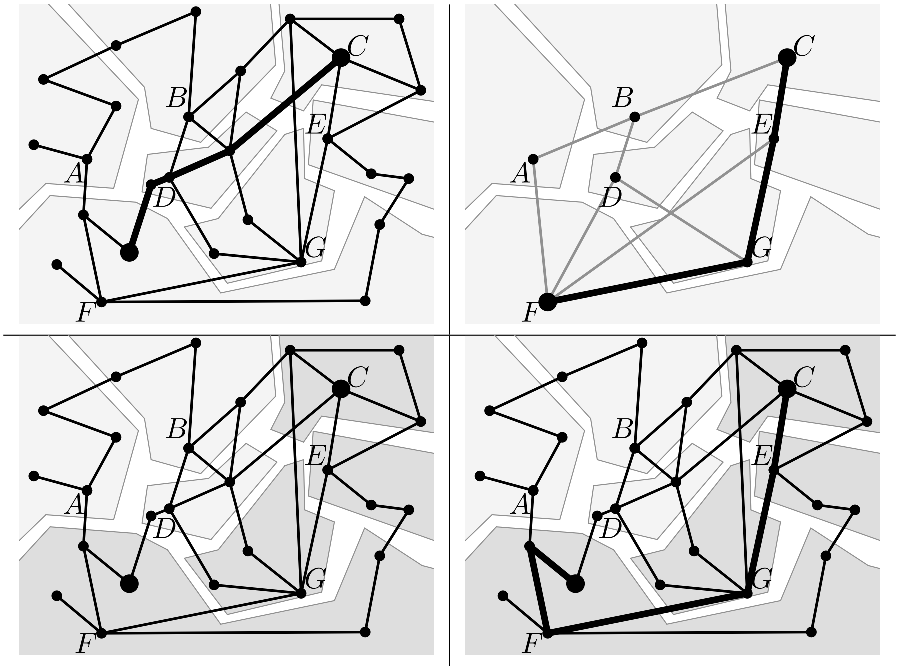

Figure 6.2: Illustration of the query algorithm of the Voronoi method. Left to right, top to bottom: (1) the original shortest path, (2) shortest path in the (weighted) dual, (3) sleeve, and (4) shortest path in the sleeve

6.4 Computational Complexity

In this section we study the cost of computing a Voronoi dual. Recall that in Erwig's algorithm [Erw00, Section 3.1] the graph Voronoi diagram is constructed with a single Dijkstra search. A heap is used to store the shortest path distances from nodes to their closest Voronoi node. Conceptually, a dummy node with a zero-weighted edge to each of the Voronoi nodes is added, the dummy node is inserted into the heap, and the Dijkstra single source shortest path search is executed. The running times of different implementations of Dijkstra's algorithm depend on the priority queue employed (see Table 2.5). Using Fibonacci heaps [FT87], Dijkstra's algorithm takes time $\mathcal{O}(m + n \lg n)$.

Erwig also claims a time lower bound of $\Omega(\max(n, (n-k)\lg k))$ [Erw00, Theorem 1]. The lower bound simplifies to $\Omega(n \lg n)$ when the number of Voronoi nodes is assumed to be $k = n^C$ for a fixed choice of $C \in (0, 1)$. Assuming that all edges must be inspected at construction time, this lower bound would be tight. The bound is information theoretic: for a connected graph, each node $w \in V \setminus K$ is in exactly one of the k regions V_i. Encoding one instance out of these k^{n-k} possibilities requires $\lg k^{n-k} = (n-k)\lg k$ bits.

For some graphs with special properties, Erwig's lower bound may not apply. Eppstein and Goodrich [EG08] presented a linear-time algorithm to compute the Voronoi diagram for road networks satisfying certain geometric properties. Also, the lower bound may not hold under different models of computation, such as the word RAM model. This model assumes that basic operations such as adding two words requires a single time step, and that the time compexity is the number of word operations executed. The space complexity is the number of words of storage required, assuming that any identifier (such as a node label) or value (such as a distance) can be contained in a single word. Under the word RAM model, the implementation of Dijkstra's algorithm by Thorup [Tho04b] requires only $\mathcal{O}(m + n \lg\lg n)$–time.

Corollary 39. *The graph Voronoi dual can be computed in time $\mathcal{O}(m + n \lg\lg n)$ in the word RAM model.*

Note that the time upper bound under the word RAM model does not contradict Erwig's information-theoretic lower bound [Erw00, Theorem 1] of $\Omega(n \lg n)$ bits.

Computing a graph Voronoi dual does not actually require the use of Dijkstra's algorithm — any single source shortest path algorithm (including parallel and distributed algorithms) can be used to compute a graph Voronoi dual as follows. Instead of an adapted Dijkstra search, we may also

1. augment G by introducing a dummy node v_0 connected to each of the Voronoi nodes with an edge of length zero,

2. run any single source shortest path algorithm in the augmented graph G' with v_0 as its source, and

3. explore the search tree rooted at v_0 by following shortest path edges only.

This last step simulates a Dijkstra search by following the single source shortest path tree without using any expensive decrease-key operations (these operations have to be avoided to reduce the worst-case running time [Tho00b, Tho07]); a First-In-First-Out queue with constant time for the enqueue and dequeue operations is sufficient. For a pseudo-code description, see Algorithm 8. Although the construction is mainly of theoretical interest, it may be useful for example for parallel or distributed algorithms and for software that must rely on certain libraries.

Note that, if a single source shortest path algorithm $\mathcal{A}$ works for a special class of graphs $\mathcal{G}$, the augmented graph G' may not necessarily be in $\mathcal{G}$, and thus algorithm $\mathcal{A}$ cannot be used in general. For example, for planar graphs, the $\mathcal{O}(n)$–time algorithm of Henzinger et al. [HKRS97] cannot be applied directly to compute the Voronoi diagram since planarity may be violated by adding a dummy node. In the particular case of the algorithm of Henzinger et al., however, the analysis of the running time depends on separators, which seem to admit the introduction of a dummy node.

Algorithm 8 ComputeVoronoiDual$(G = (V, E), K \subseteq V)$

```
Let $G' := (V', E')$ with $V' = V \cup \{v_0\}$ and $E' = E \cup \{(v_0, v) : v \in K\}$ with $\mathbf{w}'(v_0, v) = \delta$
(one would set $\delta = 0$ if possible; if only positive edge are allowed, other values work as well)

$\mathcal{D}$ := SSSP$(G', v_0)$, where $\mathcal{D}$ is the distance vector storing the distance from $v_0$ to each node $u \in V'$
for $i := 1$ to $k = |K|$ do
  $\mathsf{vor}(v_i) := i$
  FIFO.enqueue$(v_i)$
end for
while $\neg$FIFO.empty do
  $u_{\mathsf{cur}}$ := FIFO.dequeue
  for $u \in \Gamma(u_{\mathsf{cur}})$ do
    if $\mathcal{D}(u) = \mathcal{D}(u_{\mathsf{cur}}) + \mathbf{w}(u, u_{\mathsf{cur}})$ and $\mathsf{vor}(u) = \mathsf{undef}$ then
      $\mathsf{vor}(u) := \mathsf{vor}(u_{\mathsf{cur}})$
      FIFO.enqueue$(u)$
    else if $\mathsf{vor}(u) \neq \mathsf{undef}$ and $\mathsf{vor}(u) \neq \mathsf{vor}(u_{\mathsf{cur}})$ then
      if $(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) \notin E^*$ then
        $E^* := E^* \cup \{(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)})\}$
        $\mathbf{w}^*(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) := \infty$
      end if
      if $\mathbf{w}^*(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) > \mathcal{D}(v_0, u_{\mathsf{cur}}) - \delta + \mathbf{w}(u_{\mathsf{cur}}, u) + \mathcal{D}(u, v_0) - \delta$ then
        $\mathbf{w}^*(v_{\mathsf{vor}(u_{\mathsf{cur}})}, v_{\mathsf{vor}(u)}) := \mathcal{D}(v_0, u_{\mathsf{cur}}) - \delta + \mathbf{w}(u_{\mathsf{cur}}, u) + \mathcal{D}(u, v_0) - \delta$
      end if
    end if
  end for
end while
```

Theorem 40. *Using any single source shortest path algorithm for general graphs with running time $S(n, m, W)$, Algorithm 8 computes a graph Voronoi dual in time $\mathcal{O}(n + m + S(n, m, W))$.*

Proof. After running the SSSP algorithm in time $S(n, m, W)$, Algorithm 8 visits every node exactly once and every edge exactly twice (once for each end point). □

For undirected graphs with integer or floating point weights, we may use the $\mathcal{O}(m)$–time SSSP algorithm of Thorup [Tho99, Tho00a].

Corollary 41. *For undirected graphs with integer or floating point weights, the graph Voronoi dual can be computed in time $\mathcal{O}(m + n)$ in the word RAM model.*

For real weights and undirected graphs, we may use the $\mathcal{O}(m + n \lg \lg n)$–time algorithm of Pettie and Ramachandran [PR02].

Corollary 42. *For undirected graphs with real weights, the graph Voronoi dual can be computed in time $\mathcal{O}(m + n \lg \lg n)$.*

For road networks, we may use the linear-time algorithm of Eppstein and Goodrich [EG08].

Practical considerations. Note that storing each Voronoi node twice, once as a graph node and once as a dual node, causes unnecessary additional space consumption. However, when both the original graph and the dual graph are stored in the same structure, searching the dual could result in a substantial number of cache misses, since Voronoi nodes are roughly $1/p$ positions apart. Adapting the memory organization by reordering the nodes such that the memory locations used for Voronoi nodes are close together may potentially increase the cache efficiency [GKW07].

6.5 Stretch Analysis

In this section, we prove that the expected path length approximation ratio (stretch) is logarithmic in the number of edges of an exact shortest path. The bound on the stretch is the main theoretical result of this chapter.

In this section, to simplify notation, we only consider the multiplicative stretch α. We write stretch α instead of stretch $(\alpha, 0)$ as originally defined in Definition 30.

Theorem 43. *For shortest paths having h edges, Algorithm 7, given a graph and its Voronoi dual with sampling rate p (constructed by Algorithm 6), has expected worst-case stretch $\mathcal{O}(\lg_{1/(1-p)} h)$.*

The path $SP_{\mathcal{S}}(s,t)$ found by the algorithm is an approximation, since it is possible that no actual shortest path $SP_G(s,t)$ lies entirely within the Voronoi sleeve $\mathcal{S}$. We explain how this is possible, and give an upper bound on the expected length $\ell(SP_{\mathcal{S}}(s,t))$. For this purpose, we prove relationships between the lengths of simple paths P and their corresponding Voronoi paths P^*. The stretch of a path P^* depends on the number and distribution of Voronoi nodes on the path P. In particular, the stretch depends linearly on the largest interval between two Voronoi nodes on the path.

Definition 43. *For a path $P = (u_0, u_1, \ldots, u_h)$ in a graph $G = (V, E, \mathbf{w})$, and a set of Voronoi nodes $K \subseteq V$, two Voronoi nodes v_i, v_j on P are called* consecutive *if the subpath between v_i and v_j does not contain another Voronoi node. The* gap *g between two consecutive Voronoi nodes on the path is defined as the number of edges of this subpath. The* largest gap *of a path is the maximum over all gaps between two consecutive Voronoi nodes on the path.*

To simplify the analysis, we initially assume that s and t are Voronoi nodes. Later, we will relax this restriction.

We wish to prove that the stretch is at most the size of the largest gap $\bar{h}$ between two Voronoi nodes on the path $SP_G(s,t)$. For the analysis we fix a shortest path $SP_G(s,t)$ from s to t, say $(s, u_1, u_2, \ldots, u_{h-1}, t)$. If the corresponding Voronoi path $(SP_G(s,t))^*$ is a shortest path from s to t in the Voronoi dual, then the Voronoi sleeve $\mathcal{S}$ also contains $SP_G(s,t)$. Figure 6.3 gives an example for which $(SP_G(s,t))^*$ is not a shortest path in the dual.

In Lemma 44, for any simple path P, we give a worst-case bound on the length of the corresponding Voronoi path. P^* can have maximal stretch if there is no Voronoi node among the intermediate nodes and the corresponding Voronoi nodes have maximal distance (while still satisfying the Voronoi condition).

Lemma 44. *Given a simple path $P = (s, u_1, \ldots, u_{h-1}, t)$ between two Voronoi nodes $s = u_0$ and $t = u_h$ with h edges and length $\ell(P)$, the corresponding Voronoi path P^* in the Voronoi dual G^* has at most length $\ell(P^*) \leqslant h \cdot \ell(P)$. This upper bound is tight.*

Proof. The path contains $h-1$ intermediate nodes and h edges and therefore passes through at most $h+1$ different Voronoi regions. Out of these, at most $h-1$ regions are 'interfering' regions, meaning that the original shortest path does not lead through the corresponding Voronoi nodes but the shortest Voronoi path does. The path length $\ell(P)$ in the original graph is the sum of the edge weights $\ell(P) := d(s,t) = \sum_{k=0}^{h-1} \mathbf{w}(u_k, u_{k+1})$. The length $d^*(v_{\mathsf{vor}(u_k)}, v_{\mathsf{vor}(u_{k+1})})$ of an edge between two Voronoi nodes on the path P^* can be bounded as follows (see Figure 6.4):

$$d^*(v_{\mathsf{vor}(u_k)}, v_{\mathsf{vor}(u_{k+1})}) \leqslant d(v_{\mathsf{vor}(u_k)}, u_k) + \mathbf{w}(u_k, u_{k+1}) + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})})$$

From the Voronoi condition, we observe that $d(u_k, v_{\mathsf{vor}(u_k)}) \leqslant d(u_k, v_{\mathsf{vor}(u_j)})$ for all j. Due to the assumption that s and t are also Voronoi nodes, this also holds for source and target. That is,

$$\begin{aligned} d(u_k, v_{\mathsf{vor}(u_k)}) &\leqslant d(s, u_k) \\ d(u_k, v_{\mathsf{vor}(u_k)}) &\leqslant d(u_k, t) \\ &= d(v_{\mathsf{vor}(u_k)}, u_k) \end{aligned}$$

This yields:

$$\begin{aligned} \ell(P^*) \leqslant d^*(s,t) &= d^*(s, v_{\mathsf{vor}(u_1)}) \\ &\quad + \sum_{k=1}^{h-2} \Big[d(v_{\mathsf{vor}(u_k)}, u_k) + \mathbf{w}(u_k, u_{k+1}) + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})}) \Big] \\ &\quad + d^*(v_{\mathsf{vor}(u_{h-1})}, t) \\ &\leqslant \mathbf{w}(s, u_1) + d(u_1, v_{\mathsf{vor}(u_1)}) \\ &\quad + \sum_{k=1}^{h-2} \Big[d(v_{\mathsf{vor}(u_k)}, u_k) + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})}) \Big] \\ &\quad + \sum_{k=1}^{h-2} \mathbf{w}(u_k, u_{k+1}) \\ &\quad + d(v_{\mathsf{vor}(u_{h-1})}, u_{h-1}) + \mathbf{w}(u_{h-1}, t) \\ &\leqslant d(s,t) + \sum_{k=1}^{h-1} \Big[d(s, u_k) + d(u_k, t) \Big] \\ &= h \cdot \ell(P) \end{aligned}$$

There exist constructions for which the bound can be shown to be tight. For example, for any choice of $a > \epsilon > 0$, the edge weights of G may be chosen such that $d(u_k, v_{\mathsf{vor}(u_k)}) = a - \epsilon$, $\mathbf{w}(u_k, u_{k+1}) = \epsilon$, and $\mathbf{w}(s, u_1) = \mathbf{w}(u_{h-1}, t) = a$. Path P has length $2a + (h-2)\epsilon$, and the Voronoi path P^* has length $2a + (h-2)\epsilon + 2(h-1) \cdot (a - \epsilon)$. The worst case is attained for very small ϵ. As $\epsilon \to 0$, the ratio $\ell(P^*)/\ell(P) \to h$. □

If in addition to the endpoints there are Voronoi nodes on the shortest path, the maximum stretch is guaranteed to be smaller than the number of edges on the shortest path. In the following lemma, we prove that the maximum stretch is proportional to the largest gap between Voronoi nodes on the path. The proof is a simple composition of Lemma 44, and is supported by the illustration in Figure 6.4.

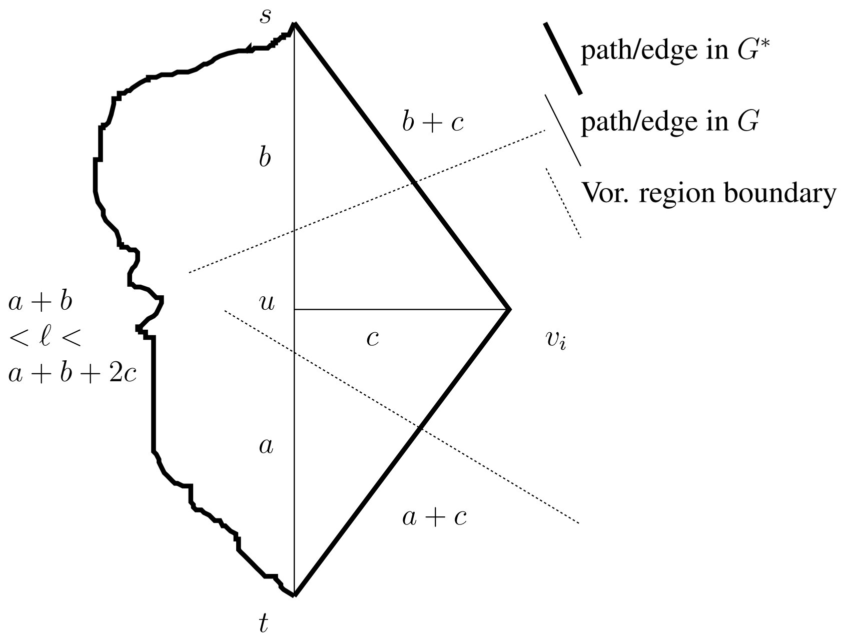

***Figure 6.3**: s, t, and v_i are Voronoi nodes. The shortest path from s to t leads through u, which is in v_i's Voronoi region (if $c < a$ and $c < b$), and paths in the Voronoi dual pass through v_i. If $\ell < a + b + 2c$, the shortest path in the Voronoi dual SP_{G^*} takes the left-hand route, and the Voronoi sleeve $\mathcal{S}$ does not contain u.*

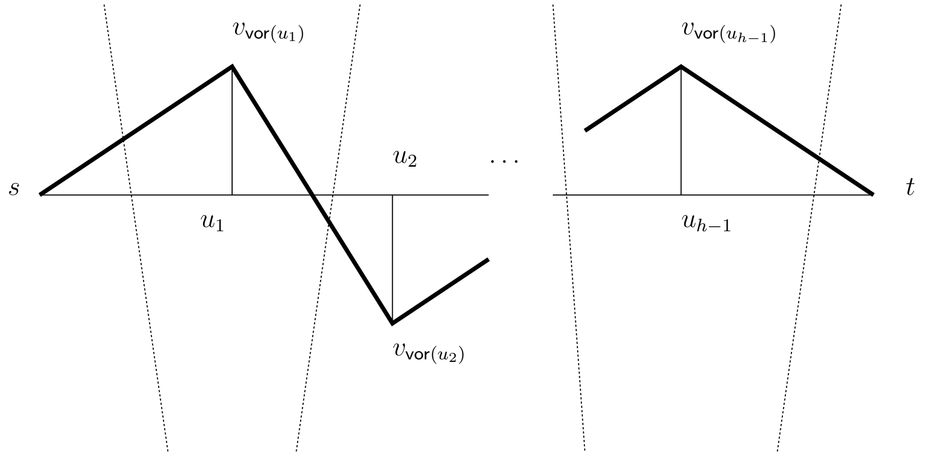

***Figure 6.4**: The shortest path between two Voronoi nodes s and t with $h-1$ intermediate nodes $u_1, \dots, u_{h-1}$. The distance between two Voronoi nodes that are adjacent in the Voronoi dual is at most $\mathbf{w}^*(v_{\mathsf{vor}(u_k)}, v_{\mathsf{vor}(u_{k+1})}) \leqslant d(v_{\mathsf{vor}(u_k)}, u_k) + \mathbf{w}(u_k, u_{k+1}) + d(u_{k+1}, v_{\mathsf{vor}(u_{k+1})})$.*

Lemma 45. *Let $P = (v_i, u_1, \dots, u_{h-1}, v_j)$ be a simple path of length $\ell(P)$ between two Voronoi nodes $v_i = u_0$ and $v_j = u_h$. Let $\bar{h}$ denote the largest gap of P. The corresponding Voronoi path P^* in the Voronoi dual G^* has at most length $\ell(P^*) \leqslant \bar{h} \cdot \ell(P)$. This upper bound is tight.*

Proof. Suppose there are $2+\nu$ Voronoi nodes $u_k = v_{\mathsf{vor}(u_k)}$ on the path. The remaining $h-1-\nu$ nodes are non-Voronoi nodes. We cut the path P into subpaths P_k between Voronoi nodes. Let h_k denote the number of edges between two consecutive Voronoi nodes, which is the number of edges of P_k. The Voronoi path is composed of $1+\nu$ segments P_k between Voronoi nodes ($\sum_{k=0}^{\nu} \ell(P_k) = P$, $\sum_{k=0}^{\nu} h_k = h$, $\forall k : h_k \leqslant \bar{h}$). Composition of Lemma 44 leads to the following bound on the path length:

$$\sum_{k=0}^{\nu} h_k \ell(P_k) \leqslant \sum_{k=0}^{\nu} \max_{\kappa \in \{0,\ldots,\nu\}} h_\kappa \ell(P_k) \leqslant \bar{h} \cdot \ell(P).$$

Tightness can be shown with the same example as in the proof of Lemma 44. □

Lemma 47 gives an upper bound on the expected size of the largest gap. We use the following lemma by Szpankowski and Rego [SR90] concerning the maximum of geometric random variables.

Lemma 46 (Szpankowski and Rego [SR90, eq. (2.6) and (2.12)])**.** *Let X_i, $i = 1, 2, \ldots, n$ be a set of i.i.d. random variables distributed according to the geometric distribution with parameter p. That is, for every $i = 1, 2, \ldots, n$ and $k \in \mathbb{N}^+$,*

$$\begin{aligned} \mathsf{Pr}[X_i = k] &= (1-p)^{k-1}p \\ \mathsf{E}[X_i] &= p^{-1} \\ \mathsf{E}[X_i^2] &= (2-p)p^{-2}. \end{aligned}$$

Let $M_n = \max\{X_1, X_2, \ldots, X_n\}$. The expected value of M_n is

$$\begin{aligned} \mathsf{E}[M_n] &= -\sum_{k=1}^{n} (-1)^k \binom{n}{k} \frac{1}{1-(1-p)^k} \\ &= \lg_{1/(1-p)} n + \mathcal{O}(1). \end{aligned}$$

Lemma 47. *In a path of length $h-1$, where each node has been selected as a Voronoi node independently at random with probability p, the longest sequence of non-Voronoi nodes is of expected length at most $\mathcal{O}(\lg_{1/(1-p)} h)$.*

Proof. The path can be seen as a sequence of coin tosses, for which we want to bound the expected length of the longest sequence of tails. This problem is known as the Longest Success-Run [EMK97, Ch. 8.5]. We wish to bound the expectation of the maximum of N independent geometric random variables with probability p and sum $h-1-N$ (N itself being a random variable).

To derive a bound on the expectation, we observe that by dropping the sum condition, and by taking the maximum over $h \geqslant N$ random variables, the maximum value obtained can only increase.

As of Lemma 46, the expectation of the maximum of h geometric random variables with probability p is known to be at most $\mathcal{O}(\lg_{1/(1-p)} h)$. □

We now combine Lemmas 44, 45, and 47 to prove Theorem 43.

Proof of Theorem 43. Consider first the case where s and t are both Voronoi nodes.

Let $\bar{h}$ denote the largest gap of some shortest path $SP_G(s,t)$. Lemma 45 implies that the corresponding Voronoi path $(SP_G(s,t))^*$ has length at most $\bar{h} \cdot \ell(SP_G(s,t))$. Trivially, the shortest path in the Voronoi dual is of length no more than that of the Voronoi path; that is, $\ell((SP_G(s,t))^*) \geqslant \ell(SP_{G^*}(s,t))$. The path $SP_{G^*}(s,t)$ in the Voronoi dual corresponds to a path P' of the same length in the Voronoi sleeve $\mathsf{Sleeve}(SP_{G^*}(s,t))$. Therefore,

$$\begin{aligned}\ell(SP_{\mathcal{S}}(s,t)) &\leqslant \ell(P') \\ &= \ell(SP_{G^*}(s,t)) \\ &\leqslant \ell((SP_G(s,t))^*) \\ &\leqslant \bar{h} \cdot \ell(SP_G(s,t)).\end{aligned}$$

Recall that nodes are independently selected as Voronoi nodes with sampling rate p. For a shortest path with h edges, the expected largest gap $\bar{h}$ is at most $\mathcal{O}(\lg_{1/(1-p)} h)$ by Lemma 47.

For the case where either s or t (or both) are not Voronoi nodes, if the path returned by Algorithm 7 has been found in Step 1, it is optimal, and the result holds trivially. For the remainder of the proof we assume that the shortest path has not been found in Step 1. In this case, the path returned is at most as long as the shortest path P_{vor} in G from s to t having $SP_{\mathsf{Sleeve}(SP_{G^*}(v_{\mathsf{vor}(s)},v_{\mathsf{vor}(t)}))}(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$ as a subpath. In the following, we derive an upper bound on $\ell(P_{\mathsf{vor}})$ with respect to the number of edges on the shortest path between s and t, denoted by h'. We have that

$$\ell(P_{\mathsf{vor}}) \leqslant d(s, v_{\mathsf{vor}(s)}) + d^*(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}) + d(v_{\mathsf{vor}(t)}, t).$$

Since the shortest path from s to t has not already been found directly in Step 1, it must be true that both $d(s, v_{\mathsf{vor}(s)}) \leqslant d(s,t)$ and $d(s, v_{\mathsf{vor}(s)}) \leqslant d(s,t)$. It remains to bound the distance between $v_{\mathsf{vor}(s)}$ and $v_{\mathsf{vor}(t)}$ in the dual graph.

Observe that augmenting the graph G with one edge $(u, v_{\mathsf{vor}(u)})$ of weight $d(u, v_{\mathsf{vor}(u)})$ for each non-Voronoi node $u \in V \setminus K$ affects neither the Voronoi diagram nor the Voronoi dual, since the nodes on the shortest path from $v_{\mathsf{vor}(u)}$ to u cannot be interfered with by another Voronoi node.

In the augmented primal graph, by the triangle inequality (for details, see Lemma 31), we have that

$$d(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)}) \leqslant d(v_{\mathsf{vor}(s)}, s) + d(s,t) + d(t, v_{\mathsf{vor}(t)}) \leqslant 3d(s,t)$$

using a path with at most $1 + h' + 1$ edges. Therefore, the expected distance $d^*(v_{\mathsf{vor}(s)}, v_{\mathsf{vor}(t)})$ is also bounded by $\mathcal{O}(\lg h') \cdot 3d(s,t)$. The bound for P_{vor} follows directly.

This concludes the proof of Theorem 43. □

6.6 Experiments

In the following, we provide an experimental evaluation for our implementation of the Voronoi shortest path approximation method. The preprocessing and query times are compared with those of Dijkstra's algorithm and with those of related but exact methods.

6.6.1 Algorithms

Benchmarking

As the methods in our study were developed and compiled on different computers and architectures, a direct comparison with reported query times would not be meaningful. We measure the performance of the methods against the bidirectional version of Dijkstra's algorithm, in terms of the ratio of the number of nodes settled by Dijkstra's algorithm over the number of nodes settled by the Voronoi method. This ratio, which we will refer to as the *speedup* of the method, can be used to evaluate the performance of Steps 1, 2, and 4 of Algorithm 7. In addition, we count the number of marked regions to account for Step 3.

The use of the Voronoi sleeve in Steps 3 and 4 of Algorithm 7 leads to practical improvements in accuracy; however, the example in Figure 6.3 shows that for general graphs the worst-case stretch does not improve. For all the experiments, we evaluate the method once using the refinement step and once with these Voronoi sleeve steps omitted. For the second type of queries, the reported distance is the sum of the distances from the query source to the Voronoi source, from the Voronoi source to the Voronoi target, and from the Voronoi target to the query target, as computed in Steps 1 and 2 of Algorithm 7.

Voronoi method

Our method using the Voronoi dual can be parameterized using the sampling probability p, the value of which determines the tradeoff between approximation quality and speedup. For the evaluation, we consider three values of the sampling probability — $p = 1/2$, $p = n^{-1/2}$, and $p = n^{-2/3}$ — that produce Voronoi nodesets of expected sizes $n/2$, $\sqrt{n}$, and $\sqrt[3]{n}$ respectively. The variants are referred to as VORHALF, VORROOT, and VORCUBERT.

Other methods

Sanders and Schultes [SS07a, Table 1] provide a detailed overview of methods for accelerated point-to-point shortest path queries in road networks. Bauer et al. [BDS+08, p. 13] list another set of methods and compare their performance on several transportation networks. We select some of the fastest methods for comparison with our algorithm. Unless stated otherwise, we will use the naming conventions of [SS07a, BDS+08] to refer to these methods. We give a brief description of the methods considered. More background information can be found in Section 3.2.3.

- Highway Hierarchies (HH) [SS06] 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. HH+dist denotes a variant of HH where all higher levels with at most $\mathcal{O}(\sqrt{n})$ nodes are replaced by a single distance table. HH+dist+A* is HH combined with A* search and implemented with distance tables [DSSW06]. Highway Node Routing (HNR) [SS07b] is another variant of the Highway Hierarchies strategy.

- In the same spirit as HH, Transit Node Routing (TNR) [BFM+07] identifies a set of nodes (called 'transit' nodes) that often occur on shortest paths. A table storing the distances between all pairs of these nodes allows any shortest path distance to be computed with a small number of table look-ups. Two variants are listed: TNR-eco with economical space consumption, and TNR-gen with generous space consumption.

- The Arc-Flag method [Lau04] computes a partition of the graph and then, for each component and for each shortest path ending in that component, it labels the first edge. A variant of this method, SHARC [BD08], incorporates techniques developed for Highway Hierarchies.

- Contraction Hierarchies (CHHNR) [GSSD08] is an extension of highway hierarchies in which the graph is further simplified using contraction operations. Many variants have been proposed; we consider only the variant with the fastest preprocessing time, CHHNREDS1235, and the variant with the best speedup, CHHNREVSQWL. The CHASE method [BDS$^+$08] integrates the Contraction Hierarchies and Arc-Flag methods.

- A method based on A* search by Goldberg and Harrelson [GH05], which we will refer to as simply A*, is one of the first methods with reasonable preprocessing time and good speedup.

- ALT-m16 [DW07] is a variant of ALT [GW05], which in turn is a combination of A*, Landmarks, and speedup techniques based on the triangle inequality. CALT-m16 and CALT-a64 [BDS$^+$08] are two variants of a method that combines ALT and Contraction Hierarchies.

6.6.2 Data Sets

For the sake of comparison, we consider transportation networks that were used by Sanders and Schultes [SS07a] and Bauer et al. [BDS$^+$08, BD08] in their evaluations. In addition, to demonstrate that our method is effective for more general graphs, we run experiments with a social network, a citation graph, a router network, and protein interaction networks as data sets. The node degrees of these graphs appear to follow a power-law distribution [Mit03].

Road networks

The road network of Western Europe has been made available for scientific use by the company PTV AG. It covers 14 countries and, with its massive size of 18,010,173 nodes and 42,560,279 directed edges, it serves as an important benchmark for shortest path queries. In order to apply the Voronoi method, we convert the graph into an undirected form. There are two different edge weightings, one representing geographical distances and the other representing driving time. We conduct experiments for both.

Public transportation

We also conduct experiments for three European public transportation networks: (1) long railway connections in Europe, with 1,586,862 nodes and 2,402,352 directed edges, (2) the bus network of the Rhein-Main-Verkehrsverbund RMV, with 2,278,066 nodes and 3,417,084 directed edges, and (3) the bus network of the Verkehrsverbund Berlin Brandenburg VBB, with 2,600,818 nodes and 3,901,212 directed edges. The graphs considered by [BDS$^+$08, BD08] differ slightly from those used for experimentation with the Voronoi method.

The numbers of nodes and edges of the RMV and VBB input graphs are nearly identical; however, the long railway graph used in our experimentation has 33% more nodes and edges than in [BDS$^+$08, BD08]. Again, for the Voronoi experimentation, the graphs were converted into an undirected form.

Social networks

We extracted the DBLP computer science bibliography [Ley02] co-author graph from an official XML version downloaded on 24 August 2008. In the graph, two authors are connected by an edge if they have at least one joint publication. This yielded an undirected graph, from which we selected the largest connected component. The final graph is unweighted and consists of 511,163 nodes and 1,871,070 edges.

Router topology

CAIDA maintains data on the router-level topology of a portion of the Internet [Coo03]. After cleaning we obtained an undirected, unweighted graph with 190,914 nodes and 607,610 edges.

Citation graph

The citations for 27,400 publications in the high energy physics research literature were used as a data set in the KDD Cup 2003 competition [GSDF03]. From these citations, we constructed an undirected, unweighted graph with 352,542 edges.

Protein interactions

The Database of Interacting Proteins [SMS+02] catalogs experimentally determined interactions among proteins. We extracted the largest connected component, consisting of 19,928 nodes and 82,406 edges. BioGRID is a general repository for interaction data sets [SBR+06] from which we extracted the largest connected component, consisting of 4,039 nodes and 43,854 edges.

6.6.3 Experimental Setting

In this section we describe the experimental setting for the Voronoi method. The implementation is written in C++ and executed on one core of a 2x2.66 GHz Dual-Core Intel Xeon Desktop with 6 GB 800 MHz DDR2 FB-DIMM running Mac OS X 10.5.6.

Every graph was preprocessed $1,000$ times using different random seeds (250 times for the European road networks). For these runs we report the mean value and standard deviation of the execution time in seconds. After preprocessing, we performed 100 shortest path queries for random (s, t) pairs. For these queries, we provide the mean values and standard deviations of the speedup relative to the bidirectional version of Dijkstra's algorithm, and of the multiplicative stretch relative to a shortest path.

6.6.4 Results

> Computers are useless. They can only give you answers.
>
> *Pablo Picasso (1881–1973)*

Running times, speedups, and approximation qualities for the Voronoi method are listed in Table 6.2 for transportation data sets and 6.3 for data sets concerning complex networks. The performances of the other methods are listed in Tables 6.1 and 6.4 as originally summarized in [SS07a, GSSD08, BDS+08].

Preprocessing

For the Voronoi method, as Lemma 38 predicts, the preprocessing cost is extremely low for all three values of p. For the non-planar graphs, the greatest preprocessing times were observed for the largest value, $p = 1/2$. This likely reflects the logarithmic cost of the heap operations associated with the computation of Voronoi regions. At the start of the Dijkstra search, the heap is initialized with all neighbors of the graph Voronoi nodes. When p is large, the initial heap size is a large proportion of the total number of nodes, and the cost of the heap operations becomes significant. On the other hand, when p and the average node degree are both small, the heap evolves smoothly with its size remaining small.

Speedup

For road networks VORHALF achieves moderate speedups of approximately 2, which likely reflects the fact that the expected number of nodes of the Voronoi dual is half that of the original graph. For the power-law graphs, probability $p = 1/2$ does not lead to a significant speedup. One reason for this might be that the Voronoi dual for each of these graphs is rather dense and, as a consequence, the Dijkstra search in the dual explores many nodes until it can find the destination. For the smaller probabilities, larger speedups can be observed, but the performance gain is significantly smaller[2] than the speedups obtained for almost planar networks. There, the speedup seems proportional to $1/p$. As expected, if for small values of p the sleeve is used to refine the path, the speedup decreases drastically due to the large size of this subgraph.

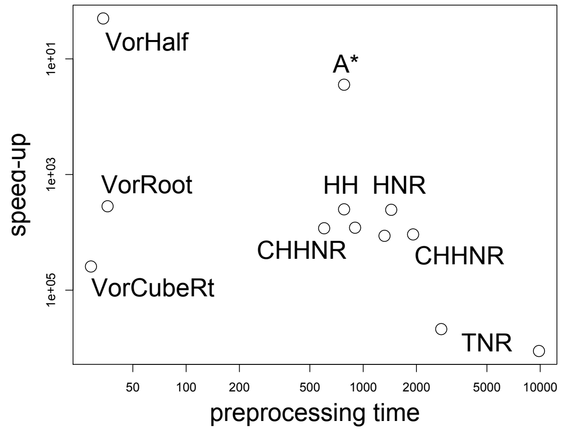

Figure 6.5: *Preprocessing time versus speedup (with respect to Dijkstra's algorithm) tradeoff for the European road network. Plot on a doubly-logarithmic scale, y–axis reversed. Circles stand for variants of the Voronoi method and for the related* exact *methods listed in Table 6.1. Transit-Node Routing (TNR) has the best speedup and the slowest preprocessing. Contraction Hierarchies (CHHNR) and Highway Hierarchies (HH) achieve a very good tradeoff between preprocessing and query times. A* has short preprocessing times but rather low speedup. The Voronoi method (Chapter 6) has the fastest preprocessing times with competitive speedups at the cost of exactness.*

[2] Cheng and Yu [CY09] use 2–hop labels [CHKZ03] to efficiently compute exact distances. They obtain a better speedup at the expense of a significantly longer preprocessing. For a 10 times smaller DBLP graph with 52,682 nodes and 59,395 edges, their preprocessing step takes 20 seconds [CY09, Table 1]. At query time it outperforms Dijkstra's algorithm by two orders of magnitude [CY09, Figure 17 and Section 7.4].

Stretch

The Voronoi method achieved stretch values that were surprisingly consistent among different data sets, with most values under 2 and very close to optimal for the road networks. Figures 6.6 and 6.7 show the approximate path length versus the shortest path length, with and without the sleeve refinement steps, respectively. The theoretical worst-case logarithmic dependency on the number of edges cannot be observed in the experimental results. Refinement using the sleeve substantially improves the stretch in practice, although the theoretical performance is not affected.

PTV European road network, driving time			
		preprocessing $[s]$	speedup
CHHNR$_{\text{EDS1235}}$	[GSSD08]	602	≈8,505
A*	[GH05]	780	28
HH	[SS06]	780	4,002
HH+dist	[SS06]	900	8,320
HH+dist+A*	[DSSW06]	1,320	11,496
HNR	[SS07b]	1,440	4,079
CHHNR$_{\text{EVSQWL}}$	[GSSD08]	1,914	≈10,874
TNR-eco	[BFM$^+$07]	2,760	471,881
TNR-gen	[BFM$^+$07]	9,840	1,129,143

***Table 6.1**: Experimental results of related exact shortest path query methods for road networks. This table is excerpted from Sanders and Schultes [SS07a, Table 1] except for CHHNR values, which are from [GSSD08, Table 1]. Preprocessing times are converted from minutes to seconds to ease comparison with the Voronoi method. Machines used (except for A*): 2.0 or 2.6 GHz processor, 8 or 16 GB RAM,* `C++` *implementation.*

method	preprocessing $[s]$	without sleeve		with sleeve	
		speedup	stretch	speedup	stretch
PTV European road network, driving time, 18,010,173 nodes, 42,560,279 edges					
VorHalf	31.7686±4.4436	2.6061± 0.0734	1.0394±0.0131	2.5878±0.0750	1.0111±0.0062
VorRoot	40.5296±3.6423	3,518.0645± 725.2776	1.6613±0.2078	4.9991±4.9017	1.1291±0.0783
VorCubeRt	31.3372±2.8181	39,918.4988±14,207.5395	1.5544±0.4292	1.5863±1.1123	1.0405±0.0597
PTV European road network, geographical distance, 18,010,173 nodes, 42,560,279 edges					
VorHalf	29.8365±4.3576	2.6266± 0.0558	1.0307±0.0095	2.5800±0.0627	1.0139±0.0057
VorRoot	34.2785±3.0609	3,672.4070± 511.1418	1.1821±0.0960	5.9212±7.9921	1.0390±0.0249
VorCubeRt	22.5531±2.0284	42,266.6442±13,530.5983	1.2882±0.5384	1.6383±1.4232	1.0141±0.0291
Public transportation, long distance railway, 1,586,862 nodes, 2,402,352 edges					
VorHalf	2.0499±0.1998	1.9511± 0.1231	1.0180±0.0227	1.8972±0.1367	1.0080±0.0143
VorRoot	1.9086±0.0946	363.8390± 153.4644	1.3813±0.2848	2.8527±3.3113	1.0829±0.0971
VorCubeRt	1.7633±0.0860	2,116.0373± 1,251.1773	1.5167±0.6610	1.2599±0.5990	1.0247±0.0658
Public transportation, RMV, 2,278,066 nodes, 3,417,084 edges					
VorHalf	3.7714±0.4064	1.9892± 0.1813	1.0290±0.0255	1.9315±0.1766	1.0104±0.0131
VorRoot	3.7455±0.2158	789.2912± 328.2714	1.2972±0.2591	3.1802±5.4237	1.0644±0.0864
VorCubeRt	3.4120±0.1633	5,973.7950± 3,748.1389	1.3522±0.6003	1.3089±0.9703	1.0204±0.0583
Public transportation, VBB, 2,600,818 nodes, 3,901,212 edges					
VorHalf	4.1409±0.4180	1.9881± 0.6476	1.0335±0.0248	1.9313±0.5172	1.0075±0.0097
VorRoot	4.0242±0.2914	866.8917± 405.4821	1.4042±0.2516	3.7864±7.6010	1.0834±0.1000
VorCubeRt	3.7145±0.2333	7,373.2971± 4,742.2783	1.4375±3.3690	1.3427±1.2759	1.0244±0.0660

***Table 6.2**: Experimental results for the Voronoi method on transportation networks.*

method	preprocessing $[s]$	without sleeve		with sleeve	
		speedup	stretch	speedup	stretch
DBLP co-authorship, 511,163 nodes, 1,871,070 edges					
VorHalf	0.9145±0.0431	1.3576± 1.4690	1.2093±0.1805	1.3447± 1.4364	1.1419±0.1468
VorRoot	0.8376±0.0430	37.7082± 53.2992	1.9323±0.3591	11.4432±14.8387	1.3954±0.2850
VorCubeRt	0.6041±0.0312	143.8757±208.7946	2.0033±0.3630	9.9616±12.5412	1.2881±0.2406
CAIDA router topology, 190,914 nodes, 607,610 edges					
VorHalf	0.3050±0.0154	1.3164± 1.1720	1.1810±0.1703	1.2972± 1.1074	1.1283±0.1359
VorRoot	0.1793±0.0092	42.4832± 54.6527	1.7845±0.3533	7.8865± 8.8062	1.2345±0.2175
VorCubeRt	0.1562±0.0081	135.5521±188.9479	1.8314±0.3755	6.0451± 7.1000	1.1621±0.1837
High energy physics citations, 27,400 nodes, 352,542 edges					
VorHalf	0.1764±0.0100	1.6620± 1.2240	1.3179±0.2909	1.6452± 1.1544	1.2107±0.2323
VorRoot	0.0611±0.0043	40.1114± 21.9262	1.9918±0.4695	11.5248± 7.9582	1.3390±0.3286
VorCubeRt	0.0461±0.0032	101.9210± 58.6233	2.0330±0.4852	9.0423± 7.5795	1.2325±0.2750
Database of Interacting Proteins, 19,928 nodes, 82,406 edges					
VorHalf	0.0117±0.0007	2.2044± 1.0637	1.1887±0.2188	2.1248± 1.0093	1.1183±0.1778
VorRoot	0.0108±0.0007	57.7343± 45.7341	1.8214±0.4084	9.1154± 6.0720	1.3216±0.3030
VorCubeRt	0.0096±0.0006	134.4816±106.4737	1.9277±0.4444	6.2541± 3.8117	1.2644±0.2703
BioGRID, 4,039 nodes, 43,854 edges					
VorHalf	0.0035±0.0002	1.5086± 0.8003	1.2581±0.2718	1.3722± 0.6858	1.1334±0.1973
VorRoot	0.0025±0.0001	10.7295± 7.9563	1.8676±0.5737	3.0394± 1.9172	1.2753±0.3354
VorCubeRt	0.0024±0.0001	18.6805± 14.7570	1.9412±0.6250	2.7906± 1.7177	1.2308±0.3137

***Table 6.3**: Experimental results for the Voronoi method on complex networks.*

	long distance rail		RMV		VBB	
$\lvert V\rvert$	1,192,736		2,277,812		2,599,953	
$\lvert E\rvert$	1,789,088		3,416,552		3,899,807	
	preprocessing $[s]$	speedup	preprocessing $[s]$	speedup	preprocessing $[s]$	speedup
CALT-a64 [BDS+08]	87	291.84	191	267.11	123	459.30
CALT-m16 [BDS+08]	158	182.71	377	159.62	174	281.23
ALT-m16 [DW07]	291	20.30	556	18.91	604	23.04
CHHNR [GSSD08]	286	1,620.62	2,584	2,077.69	1,636	3,124.59
CHASE [BDS+08]	536	2,660.93	2,863	4,649.26	2,008	10,398.64
SHARC [BD08]	12,540	81.04			36,120	118.10

***Table 6.4**: Experimental results of related exact shortest path query methods for public transportation networks. This table is excerpted from Bauer et al. [BDS+08, p. 13]. SHARC is evaluated in [BD08, p. 10]. The speedup is computed according to the number of settled nodes. Machines used: 2.0 or 2.6 GHz processor, 8 or 16 GB RAM,* `C++` *implementation.*

6.7 Conclusion and Open Problems

We have presented a simple and general method based on Voronoi duals to efficiently support shortest path queries in undirected graphs with very low preprocessing overheads and competitive query times, at the cost of exactness. The method was shown to be effective on a variety of graph types while remaining a reasonable alternative to existing exact methods specifically designed for transportation networks. The results of our experiments also demonstrate that the approximation ratio in practice is significantly better than the tight theoretical worst-case bound (Theorem 43). The maximal distortion of paths in the graph Voronoi dual depends on the distance between nodes in the original graph, unlike Delaunay triangulations of the Euclidean plane, which have constant distortion [DFS90, KG92].

Although the Voronoi method is intended mainly as a practical one (and thus kept as simple as possible to facilitate efficient implementation), let us conclude with a few remarks concerning the theoretical performance of the Voronoi method. If logarithmic stretch is allowed, the distance oracle of Thorup and Zwick [TZ05] achieves quasi-linear space $\mathcal{O}(n^{1+1/\lg n} \lg n) \leqslant \mathcal{O}(n \lg n)$ and query time $\mathcal{O}(\lg n)$. With the same worst-case stretch, the query time of the Voronoi method is much worse. However, note that the tradeoff between stretch and query time is fundamentally different from Thorup and Zwick's result: in the Voronoi method the long query time helps to compute a better approximation with a better stretch. The quantitative tradeoff between query time and stretch is probably far from optimal but intuitively this is the 'right' relationship. More time should in general yield better results. Note that, in theory, the Voronoi method is also outperformed by a distance oracle based on Bourgain's embedding [Bou85] (Theorem 3) and the distance oracle of Mendel and Naor [MN06]. In practice, however, the Voronoi method provides much better stretch than the worst-case bound predicts.

It remains open as to whether the Voronoi method presented in this paper can be extended to handle directed graphs. The nature of the Voronoi dual within a directed graph is inherently different from the dual within an undirected graph. The need for path connectivity suggests the construction of two Voronoi diagrams, one where reachability paths are oriented outward from Voronoi nodes and another where reachability paths are oriented inward. As the respective Voronoi regions may not coincide [Erw00], it is not straightforward to define a single dual structure whose shortest path lengths approximate those of the original graph.

A natural extension is the computation of a hierarchical structure of Voronoi duals, where the Voronoi nodes are chosen through recursive sampling. At a given level of the hierarchy, shortest path queries within the Voronoi dual would be resolved by a recursive call one level higher in the structure. Note that the union of all regions of different levels do not necessarily form a laminar family [GA06]. The Voronoi sleeves may expand locally compared to the sleeve of a higher level. A node that was not considered on a higher level of the hierarchy may be part of the sleeve in a lower level.

At query time, the practical stretch can be improved; instead of considering the sleeve defined by nodes on the Voronoi path only, one can also allow the query algorithm to search in neighboring cells. Furthermore, instead of searching the path between one Voronoi source and one Voronoi sink, one can compute paths between all Voronoi nodes close to the source and all Voronoi nodes close to the sink. For both heuristics, the theoretical worst-case performance does not seem to improve. However, these practical heuristics give useful parameters that may be tuned depending on the needs of the application.

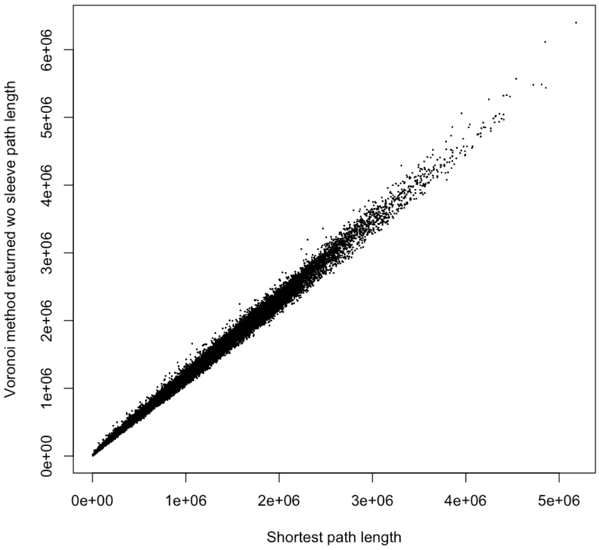

Figure 6.6: *Approximate path length versus actual shortest path length for* VORROOT *with sleeve steps omitted on the European road network, distance metric. The theoretical worst-case logarithmic dependency on the number of edges cannot be observed in the experimental results.*

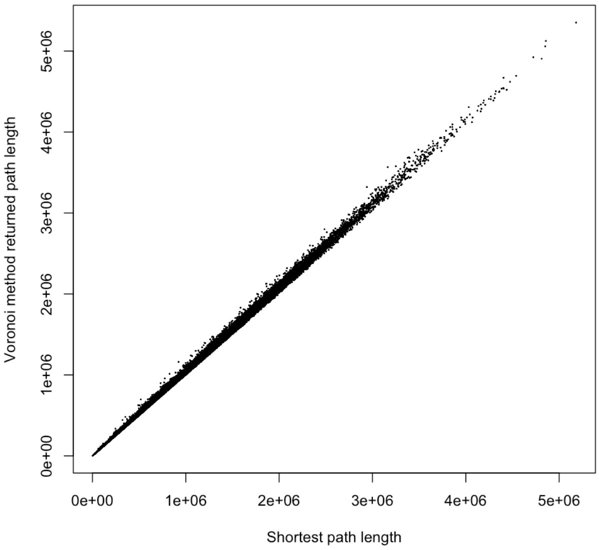

Figure 6.7: *Approximate path length versus actual shortest path length for* VORROOT *using the sleeve on the European road network, distance metric. Refinement using the sleeve substantially improves the stretch in practice (also compare with Figure 6.6), although the theoretical performance is not affected.*

MODUS VIVENDI

7
Conclusion

This thesis investigates shortest path query processing in networks both from a theoretical and a practical point of view.

Theory. We advance the theory of distance oracles in two ways.

Thorup and Zwick [TZ05] left open the important question whether, for graphs with n nodes and m edges, the $\mathcal{O}(m)$ space requirements of SSSP and the $\mathcal{O}(1)$ query time of APSP could be combined to obtain a distance oracle with space $\mathcal{O}(m)$ and query time $\mathcal{O}(1)$. They prove that graphs with large girth cannot be compressed below $\Omega(m)$. One result of this thesis is to prove that there are graphs for which $\mathcal{O}(m)$ space is not sufficient to answer distance queries in constant time. The result is actually stronger: for some graphs, $\mathcal{O}(m)$ space is not sufficient even if the query algorithm is allowed to return approximate distances with a small distortion. To obtain an efficient distance oracle for general sparse graphs, an additional data structure of considerable size is necessary. Our proof implies that a distance oracle with query time $\mathtt{t}$ and stretch $(\alpha, 0)$ requires space roughly $n^{1+\Omega(1/\alpha\mathtt{t})}$.

Thorup and Zwick, prove that, for many integer values of k, $\Omega(n^{1+1/k})$ space is necessary if distances must be approximated by a factor less than $2k+1$. For special classes of sparse graphs, this lower bound can be circumvented [FR06, Cab06, ACC$^+$96, Dji96, DPZ95, DPZ00, GKR04, Tho04a, Kle02a, Kle05, AG06, AFGW10, CZ00, CLSS98, GZ05, FK07, Spr07, Tal04, ABN08, GLNS08, AGL07, SS09, SSA09], usually by using the inherent *structure* of graphs in the class. One common approach is to use separators [LT80, AST90]. Complex networks, in particular power-law graphs, lack structure and separators. A different approach is necessary. Experimental evidence [KFY04], however, shows that the general scheme by Thorup and Zwick with stretch $(3, 0)$ and space requirement $\mathcal{O}(n^{3/2})$ works well for power-law graphs, meaning that the space requirements are significantly lower than the space requirements the tight theoretical bound would predict. We make an attempt to bridge the gap between theory and practice by proving why the space requirements of distance oracles for power-law graphs are indeed smaller. In networking, there is a common and powerful heuristic to use nodes with large degrees for routing purposes. We give a theoretical explanation why this heuristic is good for power-law graphs. Our analysis establishes a direct connection between the exponent of the power-law degree distribution and the space requirements of distance oracles.

Many theoretical questions remain open. Are there non-trivial distance oracles with additive error? What is the optimal tradeoff for dynamic distance oracles? Is linear space, constant stretch, and logarithmic query time achievable for sparse graphs? Is quadratic preprocessing time possible (without increasing the query time)? Is there a linear-space distance oracle for planar graphs? Can we obtain quasi-linear space and stretch less than $(3, 0)$ for power-law graphs? Is there an efficient distance oracle for *certain* sparse directed graphs?

The major open questions for the general shortest path problem may be more difficult. Is there a combinatorial algorithm that computes APSP in truly sub-cubic time? For general weights, can we compute SSSP faster than APSP? Can the SSSP problem be solved in linear time without using specific properties of the word RAM model?

Practice. A third result of this thesis is an efficient and very simple practical method to answer approximate shortest path queries. The Voronoi method works very well on practical network instances. Compared to related practical methods, the preprocessing overheads are lower and the query times remain competitive — at the cost of exactness. The method is effective on a variety of graph types. It is also a reasonable alternative to existing exact methods specifically designed for transportation networks. For example, the Voronoi method has also been applied as a subroutine in large-scale traffic simulations.

Practical challenges are manifold. What if a network changes? How can we take traffic [Ker04] into account? In current nagivation systems for example, *"you can pick the fastest route, the shortest, the one that avoids motorways or a route that passes through or avoids a particular point. Future devices will learn about a driver's preferences and adjust accordingly"* [Eco09]. For this, efficient dynamic methods are necessary. How can we efficiently adapt the data structure?

Dynamic distance oracles are also important for social networking sites. The social graph is changing constantly. At the same time, users and advertisers are interested in information on the social graph.

Euler invented the graph model roughly 250 years ago, the single-source shortest path algorithm of Dijkstra dates back 50 years, and Thorup and Zwick discovered the optimal distance oracle slightly less than 10 years ago. The last a-bit-more-than-2 years of my research life have been devoted to the investigation of shortest path query processing in networks. Some problems have been solved, but the quest must go on.

PS: to finally answer the first question of the introduction: take the express train to Narita airport and board a direct flight to Zurich, from where a train ride via Zurich main station brings you to Affoltern, which is two bus stops away from Ottenbach.

List of Tables

List of Figures

Bibliography

[AB02] Reka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002. 4

[AB07] Sanjeev Arora and Boaz Barak. Computational complexity: A modern approach, 2007. Web draft dated 2007-01-08 22:02. 60, 61

[ABN08] Ittai Abraham, Yair Bartal, and Ofer Neiman. Embedding metric spaces in their intrinsic dimension. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 363–372, 2008. 49, 121

[AC07] Noga Alon and Michael Capalbo. Finding disjoint paths in expanders deterministically and online. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 518–524, 2007. 65

[ACC$^+$96] Srinivasa Rao Arikati, Danny Z. Chen, L. Paul Chew, Gautam Das, Michiel H. M. Smid, and Christos D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *Algorithms - ESA '96, Fourth Annual European Symposium, Barcelona, Spain, September 25-27, 1996, Proceedings*, pages 514–528, 1996. 45, 47, 121

[ACIM99] Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999. Announced at SODA 1996. 28, 37, 38

[ACKM09] Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling: Or, power-law degree distributions in regular graphs. *Journal of the ACM*, 56(4), 2009. 5

[ACL00] William Aiello, Fan Rong King Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 171–180, 2000. 23, 24, 56, 81, 82, 84, 96

[ACS99] Karen Aardal, Fabian A. Chudak, and David B. Shmoys. A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72:161–167, 1999. 98

[ADD$^+$93] Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993. 27

[ADG$^+$06] Lyudmil Aleksandrov, Hristo Djidjev, Hua Guo, Anil Maheshwari, Doron Nussbaum, and Jörg-Rüdiger Sack. Approximate shortest path queries on weighted polyhedral surfaces. In *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, pages 98–109, 2006. 11, 32

[AF90] Miklós Ajtai and Ronald Fagin. Reachability is harder for directed than for undirected finite graphs. *The Journal of Symbolic Logic*, 55(1):113–150, 1990. Announced at FOCS 1988. 41, 54, 62

[AFGW10] Ittai Abraham, Amos Fiat, Andrew V. Goldberg, and Renato Fonseca F. Werneck. Highway dimension, shortest paths, and provably efficient algorithms. In *ACM-*

SIAM Symposium on Discrete Algorithms (SODA'10) January 17-19, 2010, Austin, Texas, 2010. 22, 47, 55, 121

[AFWZ95] Noga Alon, Uriel Feige, Avi Wigderson, and David Zuckerman. Derandomized graph products. *Computational Complexity*, 5:60–75, 1995. 65, 74

[AG06] Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 188–197, 2006. Details in LaBRI Research Report RR-1394-06. 46, 47, 83, 121

[AGGM06] Ittai Abraham, Cyril Gavoille, Andrew V. Goldberg, and Dahlia Malkhi. Routing in networks with low doubling dimension. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), 4-7 July 2006, Lisboa, Portugal*, page 75, 2006. 49

[AGK$^+$04] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004. Announced at STOC 2001. 98

[AGL07] Mattias Andersson, Joachim Gudmundsson, and Christos Levcopoulos. Approximate distance oracles for graphs with dense clusters. *Computational Geometry*, 37(3):142–154, 2007. Announced at ISAAC 2004. 50, 121

[AGM97] Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997. Announced at FOCS 1991. 36

[AGM$^+$08] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Transactions on Algorithms*, 4(3), 2008. 49

[AGMN92] Noga Alon, Zvi Galil, Oded Margalit, and Moni Naor. Witnesses for boolean matrix multiplication and for shortest paths. In *33rd Annual Symposium on Foundations of Computer Science, 24-27 October 1992, Pittsburgh, Pennsylvania, USA*, pages 417–426, 1992. 36

[AHL02] Noga Alon, Shlomo Hoory, and Nathan Linial. The Moore bound for irregular graphs. *Graphs and Combinatorics*, 18(1):53–57, 2002. 28

[AI97] Cavit Aydin and Doug Ierardi. Partitioning algorithms for transportation graphs and their applications to routing. In *Proceedings of the 9th Canadian Conference on Computational Geometry (CCCG)*, 1997. 54

[AI00] Yasuhito Asano and Hiroshi Imai. Practical efficiency of the linear-time algorithm for the single source shortest path problem. *Journal of the Operations Research Society of Japan*, 43(4):431–447, 2000. 33, 34

[AIP06] Alexandr Andoni, Piotr Indyk, and Mihai Patrascu. On the optimality of the dimensionality reduction method. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 449–458, 2006. 60, 62

[AJ89] Rakesh Agrawal and H. V. Jagadish. Materialization and incremental update of path information. In *Proceedings of the Fifth International Conference on Data*

Engineering, February 6-10, 1989, Los Angeles, California, USA, pages 374–383, 1989. 10

[AJ94] Rakesh Agrawal and H. V. Jagadish. Algorithms for searching massive graphs. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):225–238, 1994. 51

[AJB99] Reka Albert, Hawoong Jeong, and Albert-Lászlό Barabási. Diameter of the world-wide web. *Nature*, 401:130, 1999. 5

[Ajt88] Miklós Ajtai. A lower bound for finding predecessors in Yao's call probe model. *Combinatorica*, 8(3):235–247, 1988. 60

[Ake60] Sheldon B. Akers. The use of wye-delta transformations in network simplification. *Operations Research*, 8(3):311–323, 1960. Announced at Rand Symposium on Mathematical Programming 1959. 51

[Akg88] Mustafa Akgül. Shortest paths and the simplex method. 14th International Conference on Mathematical Programming, Tokyo, 1988. 34

[Alo86] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986. 65

[Alo03] Noga Alon. Problems and results in extremal combinatorics–I. *Discrete Mathematics*, 273(1-3):31–53, 2003. 30

[ALZ07] Luca Allulli, Peter Lichodzijewski, and Norbert Zeh. A faster cache-oblivious shortest-path algorithm for undirected graphs with bounded edge lengths. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 910–919, 2007. 35

[AM05] Ittai Abraham and Dahlia Malkhi. Name independent routing for growth bounded networks. In *SPAA 2005: Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms, July 18-20, 2005, Las Vegas, Nevada, USA*, pages 49–55, 2005. 49

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows*. Prentice Hall, 1993. 32

[AMOT90] Ravindra K. Ahuja, Kurt Mehlhorn, James B. Orlin, and Robert Endre Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2):213–223, 1990. 34

[ANLP90] Baruch Awerbuch, Amotz Bar Noy, Nathan Linial, and David Peleg. Improved routing strategies with succinct tables. *Journal of Algorithms*, 11(3):307–341, 1990. 49

[AO92] Ravindra K. Ahuja and James B. Orlin. The scaling network simplex algorithm. *Operations Research*, 40(S1):5–13, 1992. 34

[AOPS02] Ravindra K. Ahuja, James B. Orlin, Stefano Pallottino, and Maria Grazia Scutellà. Minimum time and minimum cost-path problems in street networks with periodic traffic lights. *Transportation Science*, 36(3):326–336, 2002. 54

[AP89] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989. 20, 22

[AP92] Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM Journal on Discrete Mathematics*, 5(2):151–162, 1992. 49

[APF^{+}06] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006. 13

[Ari00] Masanori Arita. Metabolic reconstruction using shortest paths. *Simulation Practice and Theory*, 8(1-2):109 – 125, 2000. 11

[AS87] Noga Alon and Baruch Schieber. Optimal preprocessing for answering on-line product queries. Technical Report 71/87, Tel Aviv University, 1987. 47

[AS00] Noga Alon and Joel Spencer. *The Probabilistic Method (second edition)*. John Wiley Interscience Series in Discrete Mathematics and Optimization, 2000. 67, 68

[ASBS00] Luís A. Nunes Amaral, Antonio Scala, Marc Barthelemy, and H. Eugene Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences USA*, 97(21):11149–11152, 2000. 4, 23, 123

[Ass83] Patrice Assouad. Plongements Lipschitziens dans $\mathbb{R}^n$. *Bulletin de la Société Mathématique de France*, 111:429–448, 1983. 20

[ASS09] Dennis J. Adams-Smith and Douglas R. Shier. Generating random test networks for shortest path algorithms. *Operations Research and Cyber-Infrastructure*, 47:295–308, 2009. 22

[AST90] Noga Alon, Paul D. Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990. Announced at STOC 1990. 31, 46, 121

[AST94] Noga Alon, Paul D. Seymour, and Robin Thomas. Planar separators. *SIAM Journal on Discrete Mathematics*, 7(2):184–193, 1994. 30

[Aur91] Franz Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991. 98

[AV88] Alok Aggarwal and S. Vitter, Jeffrey. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988. 26, 43, 45

[AY00] Muhammad Abaidullah Anwar and Takaichi Yoshida. OORF: An object-oriented route finder. In *SAC '00: Proceedings of the 2000 ACM Symposium on Applied Computing*, pages 301–306, 2000. 51, 54

[AY01] Muhammad Abaidullah Anwar and Takaichi Yoshida. Integrating OO road network database, cases and knowledge for route finding. In *SAC '01: Proceedings of the 2001 ACM Symposium on Applied Computing*, pages 215–219, 2001. 51, 54

[BA99] Albert-László Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. 4, 5, 23, 24, 96

[Bac94] Paul Bachmann. *Die Analytische Zahlentheorie. Zahlentheorie. pt. 2*. 1894. 25

[BAJ00] Albert-László Barabási, Reka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281:69–77, 2000. 4, 5

[Bar03] Albert-László Barabási. *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume, 2003. 81

[Bat08] Michael Batty. The size, scale, and shape of cities. *Science*, 319(5864):769–771, 2008. 10

[Bav48] Alex Bavelas. A mathematical model of group structure. *Human Organizations*, 7:16–30, 1948. 12

[Bav50] Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22:725, 1950. 12

[BBCR03] Béla Bollobás, Christian Borgs, Jennifer T. Chayes, and Oliver Riordan. Directed scale-free graphs. In *Symposium on Discrete Algorithms (SODA)*, pages 132–139, 2003. 5

[BBJ+02] Christopher L. Barrett, Keith R. Bisset, Riko Jacob, Goran Konjevod, and Madhav V. Marathe. Classical and contemporary shortest path problems in road networks: Implementation and experimental analysis of the TRANSIMS router. In *Algorithms - ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17-21, 2002, Proceedings*, pages 126–138, 2002. 10

[BBK72] András Békéssy, P. Bekessy, and János Komlós. Asymptotic enumeration of regular matrices. *Studia Scientiarum Mathematicarum Hungarica*, 7:343–353, 1972. 23, 24, 96

[BC58] Frederick Bock and Scott Cameron. Allocation of network traffic demand by instant determination of optimum paths. *Operations Research*, 6:633–634, 1958. Announced at the 13th National (6th Annual) Meeting of the Operations Research Society of America, 1958. 8

[BC06] Arthur Brady and Lenore Cowen. Compact routing on power law graphs with additive stretch. In *Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments*, pages 119–128, 2006. 49, 56, 83

[BCE03] Béla Bollobás, Don Coppersmith, and Michael L. Elkin. Sparse distance preservers and additive spanners. In *Symposium on Discrete Algorithms (SODA)*, 2003. 27

[BD86] George E. P. Box and Norman R. Draper. *Empirical model-building and response surface*. John Wiley & Sons, Inc., 1986. 84

[BD08] Reinhard Bauer and Daniel Delling. SHARC: Fast and robust unidirectional routing. In *Proceedings of the 10th Workshop on Algorithm Engineering and Experiments (ALENEX'08)*, pages 13–26, 2008. 47, 54, 55, 56, 110, 111, 117, 123

[BDDW09] Reinhard Bauer, Gianlorenzo D'Angelo, Daniel Delling, and Dorothea Wagner. The shortcut problem - complexity and approximation. In *SOFSEM 2009: Theory and Practice of Computer Science, 35th Conference on Current Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 24-30, 2009. Proceedings*, pages 105–116, 2009. 55

[BDS+08] Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm. In *Experimental Algorithms, 7th Inter-*

national Workshop (WEA'08), Provincetown, MA, USA, May 30-June 1, 2008, Proceedings, pages 303–318, 2008. 47, 51, 54, 55, 56, 110, 111, 112, 117, 123

[BDW09] Reinhard Bauer, Daniel Delling, and Dorothea Wagner. Experimental study of speed-up techniques for timetable information systems. *Networks*, 2009. 56

[Bea64] Murray A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10:161–163, 1964. 12

[Bel58] Richard Ernest Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958. 8, 34

[Bel67] Richard Ernest Bellman. *Dynamic Programming*. Princeton University Press, 1967. 34, 35, 37

[Ben66] C. T. Benson. Minimal regular graphs of girth eight and twelve. *Canadian Journal of Mathematics*, 18:1091–1094, 1966. 28

[Ber93] Dimitri P. Bertsekas. A simple and fast label correcting algorithm for shortest paths. *Networks*, 23(7):703–709, 1993. 34

[Ber09] Aaron Bernstein. Fully dynamic $(2+\epsilon)$ approximate all-pairs shortest paths with $\mathcal{O}(\log\log n)$ query and close to linear update time. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 693–702, 2009. 39

[Bes74] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192–236, 1974. 11

[BFM$^+$07] Holger Bast, Stefan Funke, Domagoj Matijevic, Peter Sanders, and Dominik Schultes. In transit to constant time shortest-path queries in road networks. In *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX'07), New Orleans, Louisiana, USA, January 6, 2007*, 2007. 47, 51, 54, 55, 110, 114

[BFMZ04] Gerth Stølting Brodal, Rolf Fagerberg, Ulrich Meyer, and Norbert Zeh. Cache-oblivious data structures and algorithms for undirected breadth-first search and shortest paths. In *Algorithm Theory - SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, Denmark, July 8-10, 2004, Proceedings*, pages 480–492, 2004. 35

[BFSS07] Holger Bast, Stefan Funke, Peter Sanders, and Dominik Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007. 55

[BG07] Zachary K. Baker and Maya Gokhale. On the acceleration of shortest path calculations in transportatoin networks. In *International Symposium on Field-Programmable Custom Computing Machines*, pages 23–32, 2007. 10, 54

[BGS09] Surender Baswana, Vishrut Goyal, and Sandeep Sen. All-pairs nearly 2-approximate shortest paths in $\widetilde{\mathcal{O}}(n^2)$ time. *Theoretical Computer Science*, 410(1):84–93, 2009. Announced at STACS 2005. 37

[BGSU08] Surender Baswana, Akshay Gaur, Sandeep Sen, and Jayant Upadhyay. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I:*

Tack A: Algorithms, Automata, Complexity, and Games, pages 609–621, 2008. 43, 44, 123

[BH69] M. H. Bourgoin and E. M. J. Heurgon. Study and comparison of algorithms of the shortest path through planned experiments. In *Project Planning by Network Analysis*, pages 106–118, 1969. 35, 50

[BHS07] Surender Baswana, Ramesh Hariharan, and Sandeep Sen. Improved decremental algorithms for maintaining transitive closure and all-pairs shortest paths. *Journal of Algorithms*, 62(2):74–92, 2007. 39

[Big98] Norman Biggs. Constructions for cubic graphs with large girth. *The Electronic Journal of Combinatorics*, 5, 1998. 28

[BK06] Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA*, pages 591–602, 2006. 37, 43, 44

[BK07] Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 541–552, 2007. 37

[BKM$^+$00] Andrei Z. Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet L. Wiener. Graph structure in the web. *Computer Networks*, 33(1-6):309–320, 2000. 5

[BKMM07] David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. Approximating betweenness centrality. In *Algorithms and Models for the Web-Graph, 5th International Workshop, WAW 2007, San Diego, CA, USA, December 11-12, 2007, Proceedings*, pages 124–137, 2007. 12

[BKMP05] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α, β)-spanners and purely additive spanners. In *SODA ’05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 672–681, 2005. 28

[BKS01] Ingmar Bitter, Arie E. Kaufman, and Mie Sato. Penalized-distance volumetric skeleton algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):195–206, 2001. 11

[BL74] Mokhtar S. Bazaraa and R. W. Langley. A dual shortest path algorithm. *SIAM Journal on Applied Mathematics*, 26(3):496–501, 1974. 51

[BLM$^+$06] Stefano Boccaletti, Vito Latora, Yamir Moreno, Mario Chavez, and Dong-Uk Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308, 2006. 4, 13

[BLMN05] Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric Ramsey type phenomena. *Annals of Mathematics (2)*, 162(2):643–709, 2005. Announced at STOC 2003. 43

[Blo83] Peter A. Bloniarz. A shortest-path algorithm with expected time $\mathcal{O}(n^2 \log n \log^* n)$. *SIAM Journal on Computing*, 12(3):588–600, 1983. Announced at STOC 1980. 35

[BMST97] Frank Ball, Denis Mollison, and Gianpaolo Scalia-Tomba. Epidemics with two levels of mixing. *Ann. Appl. Probab.*, 7(1):46–89, 1997. 24, 83

[BMW89] Anantaram Balakrishnan, Thomas L. Magnanti, and Richard T. Wong. A dual-ascent procedure for large-scale uncapacitated network design. *Operations Research*, 37(5):716–740, 1989. 8

[Bol01] Béla Bollobás. *Random Graphs*. Cambridge University Press, 2001. 22, 23

[Boo67] John Boothroyd. 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, 1967. In the Algorithms Supplement. 35

[BOR80] John J. Bartholdi, James B. Orlin, and H. Donald Ratliff. Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28(5):1074–1085, 1980. 8

[Bor84] Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321–345, September 1984. 11

[Bou85] Jean Bourgain. On lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. 30, 118

[BR03] Béla Bollobás and Oliver Riordan. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics*, 1, 2003. 83

[BR04] Béla Bollobás and Oliver Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004. 4

[Bra68] Dietrich Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968. 10

[Bra01] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001. 12

[Bre32] Robert Breusch. Zur Verallgemeinerung des Bertrandschen Postulates, daß zwischen x und $2x$ stets Primzahlen liegen. *Mathematische Zeitschrift*, 34(1):505–526, 1932. 66, 67

[Bre64] Robert Breusch. An asymptotic formula for primes of the form $4n + 1$. *The Michigan Mathematical Journal*, 11:311–315, 1964. 66

[Bre66] Melvin A. Breuer. Coding the vertexes of a graph. *IEEE Transactions on Information Theory*, 12(2):148–153, 1966. 31

[Bro66] William G. Brown. On graphs that do not contain a Thomsen graph. *Canadian Mathematical Bulletin*, 9:281–85, 1966. 28

[BRST01] Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor E. Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, 18(3):279–290, 2001. 4

[BS05] Nadine Baumann and Sebastian Stiller. *Network Analysis*, chapter 13. Network Models, pages 341–372. Springer, 2005. 24

[BS06] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $\mathcal{O}(n^2)$ time. *ACM Transactions on Algorithms*, 2(4):557–577, 2006. Announced at SODA 2004. 43, 44

[BS08] Surender Baswana and Sandeep Sen. Algorithms for spanners in weighted graphs. In *Encyclopedia of Algorithms*. 2008. 27

[BSB$^+$00] Ingmar Bitter, Mie Sato, Michael Bender, Kevin T. McDonnell, Arie Kaufman, and Ming Wan. CEASAR: a smooth, accurate and robust centerline extraction algorithm. In *Visualization 2000. Proceedings*, pages 45–52, Oct. 2000. 11

[BSWW04] Ulrik Brandes, Frank Schulz, Dorothea Wagner, and Thomas Willhalm. Generating node coordinates for shortest-path computations in transportation networks. *ACM Journal of Experimental Algorithmics*, 9, 2004. 54

[BTZ98] Gerth Stølting Brodal, Jesper Larsson Träff, and Christos D. Zaroliagis. A parallel priority queue with constant time operations. *Journal of Parallel and Distributed Computing*, 49(1):4–21, 1998. Announced at IPPS 1997. 35

[But68] Larry F. Butas. A directionally oriented shortest path algorithm. *Transportation Research*, 2(3):253–268, 1968. 54

[BVW08] Guy E. Blelloch, Virginia Vassilevska, and Ryan Williams. A new combinatorial approach for sparse graph problems. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 108–120, 2008. 36, 37

[BYJKS04] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004. Special Issue on FOCS 2002. 62

[Cab06] Sergio Cabello. Many distances in planar graphs. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 1213–1220, 2006. A preprint of the journal version is available in the University of Ljubljana preprint series, Vol. 47 (2009), 1089. 37, 45, 46, 47, 121

[Cal61] Tom Caldwell. On finding minimum routes in a network with turn penalties. *Communications of the ACM*, 4(2):107–108, 1961. 8, 39

[Car71] Bernard A. Carré. An algebra for network routing problems. *IMA Journal of Applied Mathematics*, 7(3):273–294, 1971. 36

[CC07] Sergio Cabello and Erin W. Chambers. Multiple source shortest paths in a genus g graph. In *Symposium on Discrete Algorithms (SODA)*, pages 89–97, 2007. 46

[CE91] Marek Chrobak and David Eppstein. Planar orientations with low out-degree and compaction of adjacency matrices. *Theoretical Computer Science*, 86(2):243–266, 1991. 31

[CE01] Timothy M. Chan and Alon Efrat. Fly cheaply: On the minimum fuel consumption problem. *Journal of Algorithms*, 41(2):330–337, 2001. Announced at SOCG 1998 by Alon Efrat and Sariel Har-Peled. 97

[CF94] Adrijana Car and Andrew U. Frank. Modelling a hierarchy of space applied to large road networks. In *IGIS '94: Geographic Information Systems, International Workshop on Advanced Information Systems, Monte Verita, Ascona, Switzerland, February 28 - March 4, 1994, Proceedings*, pages 15–24, 1994. 51, 54

[CF06] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):2, 2006. 24

[CG06] T-H. Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 70–78, 2006. 49

[CGR96] Boris V. Cherkassky, Andrew V. Goldberg, and Tomasz Radzik. Shortest paths algorithms: Theory and experimental evaluation. *Mathematical Programming*, 73:129–174, 1996. 35

[CGS99] Boris V. Cherkassky, Andrew V. Goldberg, and Craig Silverstein. Buckets, heaps, lists, and monotone priority queues. *SIAM Journal on Computing*, 28(4):1326–1346, 1999. 35

[CH66] K. L. Cooke and E. Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, 14:492–498, 1966. 32

[CH78] Louis Caccetta and Roland Häggkvist. On minimal digraphs with given girth. In *Congressus Numerantium XXI — Proceedings of the Ninth Southeastern Conference on Combinatorics, Graph Theory, and Computing*, pages 181–187, 1978. 28

[Cha67] Bruce A. Chartres. Letter concerning Nicholson's paper. *The Computer Journal*, 10(1):118–119, 1967. In Discussion and Correspondence. 35

[Cha87] Bernard Chazelle. Computing on a free tree via complexity-preserving mappings. *Algorithmica*, 2:337–361, 1987. Announced at FOCS 1984. 47

[Cha07] Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC'07)*, pages 590–598, 2007. 36, 37

[CHC$^+$00] Maria C. Costanzo, Jennifer D. Hogan, Michael E. Cusick, Brian P. Davis, Ann M. Fancher, Peter E. Hodges, Pinar Kondu, Carey Lengieza, Jodi E. Lew-Smith, Carol Lingner, Kevin J. Roberg-Perez, Michael Tillberg, Joan E. Brooks, and James I. Garrels. The yeast proteome database (ypd) and caenorhabditis elegans proteome database (wormpd): comprehensive resources for the organization and comparison of model organism protein information. *Nucleic Acids Research*, 28(1):73–76, 2000. 6

[Che96] Danny Z. Chen. Developing algorithms and software for geometric path planning problems. *ACM Computing Surveys*, page 18, 1996. 32

[CHKZ03] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. Reachability and distance queries via 2-hop labels. *SIAM Journal on Computing*, 32(5):1338–1355, 2003. Announced at SODA 2002. 54, 57, 113

[CK95] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995. 31, 48, 50

[CK96] Laurent D. Cohen and Ron Kimmel. Global minimum for active contour models: A minimal path approach. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 666–673, 1996. 8, 10

[CKL97] Jason Cong, Andrew B. Kahng, and Kwok-Shing Leung. Efficient heuristics for the minimum shortest path Steiner arborescence problem with applications to VLSI physical design. In *International Symposium on Physical Design*, pages 88–95, 1997. 8

[CKL$^+$09] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. Models for the compressible web. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 331–340, 2009. 5

[CKR04] Gruia Calinescu, Howard J. Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, 34(2):358–372, 2004. Announced at SODA 2001. 43

[CL02] Fan Rong King Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Internet Mathematics*, 99:15879–15882, 2002. 24, 56, 82, 84

[CL06] Fan Rong King Chung and Linyuan Lu. *Complex Graphs and Networks*. American Mathematical Society, 2006. 82, 84, 86

[CL07] Edward P. Chan and Heechul Lim. Optimization and evaluation of shortest path queries. *The VLDB Journal*, 16(3):343–369, 2007. 51, 54

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001. 8, 19, 25, 27, 32, 34, 35

[CLSS98] Danny Z. Chen, D. T. Lee, R. Sridhar, and Chandra N. Sekharan. Solving the all-pair shortest path query problem on interval and circular-arc graphs. *Networks*, 31(4):249–258, 1998. 48, 121

[CLW08] Danny Z. Chen, Shuang Luan, and Chao Wang. Coupled path planning, region optimization, and applications in intensity-modulated radiation therapy. In *Algorithms - ESA 2008, 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*, pages 271–283, 2008. 11

[CMMS98] Andreas Crauser, Kurt Mehlhorn, Ulrich Meyer, and Peter Sanders. A parallelization of Dijkstra's shortest path algorithm. In *Mathematical Foundations of Computer Science 1998, 23rd International Symposium, MFCS'98, Brno, Czech Republic, August 24-28, 1998, Proceedings*, pages 722–731, 1998. 35

[CNM04] Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(6):066111+, Dec 2004. 13

[CNSW00] Duncan S. Callaway, Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85:5468–5471, 2000. 83

[CO00] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1-3):77 – 114, 2000. 48

[Coh94] Edith Cohen. Polylog-time and linear-work approximation scheme for undirected shortest paths. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 16–26, 1994. 38

[Coh96] Edith Cohen. Efficient parallel shortest-paths in digraphs with a separator decomposition. *Journal of Algorithms*, 21(2):331–357, 1996. Announced at SPAA 1993. 54

[Coh98] Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM Journal on Computing*, 28(1):210–236, 1998. Announced at FOCS 1993. 27

[Coh00] Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM*, 47(1):132–166, 2000. 35

[Coo03] Cooperative Association for Internet Data Analysis. Router-level topology measurements. Online at caida.org/tools/measurement/skitter/router_topology/, file: itdk0304_rlinks_undirected.gz, 2003. 5, 112

[Cow01] Lenore Cowen. Compact routing with minimum stretch. *Journal of Algorithms*, 38(1):170–183, 2001. 88, 90

[CP07] Timothy M. Chan and Mihai Patrascu. Voronoi diagrams in $n \cdot 2^{O(\sqrt{\lg \lg n})}$ time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 31–39, 2007. 98

[CR73] Stephen A. Cook and Robert A. Reckhow. Time bounded random access machines. *Journal of Computer and System Sciences*, 7(4):354–375, 1973. Announced at STOC 1972. 26

[CRS98] Yu-Li Chou, H. Edwin Romeijn, and Robert L. Smith. Approximating shortest paths in large-scale networks with an application to intelligent transportation systems. *INFORMS Journal on Computing*, 10(2):163–179, 1998. 51, 54

[CS83] Vasek Chvátal and Endre Szemerédi. Short cycles in directed graphs. *Journal of Combinatorial Theory, Series B*, 35(3):323–327, 1983. 28

[CSN07] Aaron Clauset, Cosma R. Shalizi, and Mark E. J. Newman. Power-law distributions in empirical data. *SIAM Reviews*, June 2007. 4, 23

[CSTW09a] Wei Chen, Christian Sommer, Shang-Hua Teng, and Yajun Wang. Compact routing in power-law graphs. In *23rd International Symposium on Distributed Computing (DISC)*, pages 379–391, 2009. 16

[CSTW09b] Wei Chen, Christian Sommer, Shang-Hua Teng, and Yajun Wang. A compact routing scheme and approximate distance oracle for power-law graphs. Technical Report MSR-TR-2009-84, Microsoft Research, July 2009. 16, 82

[CTB01] Adrijana Car, George Taylor, and Chris Brunsdon. An analysis of the performance of a hierarchical wayfinding computational model using synthetic graphs. *Computers, Environment and Urban Systems*, 25(1):69–88, 2001. 51, 54

[CV03a] Elizabeth Costenbader and Thomas W. Valente. The stability of centrality measures when networks are sampled. *Social Networks*, 25(4):283 – 307, 2003. 12

[CV03b] Bruno Courcelle and Rémi Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Applied Mathematics*, 131(1):129–150, 2003. 48

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Announced at STOC 1987. 36

[CW04] Lenore J. Cowen and Christopher G. Wagner. Compact roundtrip routing in directed networks. *Journal of Algorithms*, 50(1):79–95, 2004. Announced at PODC 2000. 49

[CWM94] Gordon Cameron, Brian J. N. Wylie, and David McArthur. PARAMICS: moving vehicles on the connection machine. Technical report, Edinburgh Parallel Computing Center, 1994. ISSN 1063-9535, IEEE. 10

[CX00] Danny Z. Chen and Jinhui Xu. Shortest path queries in planar graphs. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 469–478, 2000. 45

[CY09] Jiefeng Cheng and Jeffrey Xu Yu. On-line exact shortest distance query processing. In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings*, pages 481–492, 2009. 54, 57, 83, 113

[CYL$^+$06] Jiefeng Cheng, Jeffrey Xu Yu, Xuemin Lin, Haixun Wang, and Philip S. Yu. Fast computation of reachability labeling for large graphs. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*, pages 961–979, 2006. 54

[CYL$^+$08] Jiefeng Cheng, Jeffrey Xu Yu, Xuemin Lin, Haixun Wang, and Philip S. Yu. Fast computing reachability labelings for large graphs with high compression rate. In *EDBT 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings*, pages 193–204, 2008. 54

[CYT06] Jiefeng Cheng, Jeffrey Xu Yu, and Nan Tang. Fast reachability query processing. In *Database Systems for Advanced Applications, 11th International Conference, DASFAA 2006, Singapore, April 12-15, 2006, Proceedings*, pages 674–688, 2006. 54

[CZ00] Shiva Chaudhuri and Christos D. Zaroliagis. Shortest paths in digraphs of small treewidth. part I: Sequential algorithms. *Algorithmica*, 27(3):212–226, 2000. Announced at ICALP 1995. 47, 121

[CZ01] Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *Journal of Algorithms*, 38(2):335–353, 2001. Announced at SODA 1997. 37, 44

[CZ07] Edward P. F. Chan and Jie Zhang. A fast unified optimal route query evaluation algorithm. In *CIKM ’07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 371–380, 2007. 51

[DABC08] Engin Demir, Cevdet Aykanat, and B. Barla Cambazoglu. Clustering spatial networks for aggregate query processing: A hypergraph approach. *Inf. Syst.*, 33(1):1–17, 2008. 54, 55

[Dan57] George Bernard Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–277, 1957. 8, 9, 125

[Dan60] George Bernard Dantzig. On the shortest route through a network. *Management Science*, 6(2):187–190, 1960. 8, 32, 35

[DBCP97] Mikael Degermark, Andrej Brodnik, Svante Carlsson, and Stephen Pink. Small forwarding tables for fast routing lookups. In *SIGCOMM*, pages 3–14, 1997. 15

[dC83] Dennis de Champeaux. Bidirectional heuristic search again. *Journal of the ACM*, 30(1):22–32, 1983. 10, 35

[DCKM04] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 15–26, 2004. 15

[DEH07a] Philippe Duchon, Nicole Eggemann, and Nicolas Hanusse. Non-searchability of random power-law graphs. In *Principles of Distributed Systems, 11th International Conference, OPODIS 2007, Guadeloupe, French West Indies, December 17-20, 2007. Proceedings*, pages 274–285, 2007. 83

[DEH07b] Philippe Duchon, Nicole Eggemann, and Nicolas Hanusse. Non-searchability of random scale-free graphs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*, pages 380–381, 2007. 83

[Del09] Daniel Delling. *Engineering and Augmenting Route Planning Algorithms*. PhD thesis, Universität Karlsruhe, 2009. 52, 54, 55, 56

[DF79] Eric V. Denardo and Bennett L. Fox. Shortest route methods: Reaching, pruning and buckets. *Operations Research*, 27:161–186, 1979. 33

[DFJ54] George Bernard Dantzig, Delbert Ray Fulkerson, and Selmer M. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954. 8

[DFS90] David P. Dobkin, Steven J. Friedman, and Kenneth J. Supowit. Delaunay graphs are almost as good as complete graphs. *Discrete & Computational Geometry*, 5:399–407, 1990. 118

[DGJ08] Camil Demetrescu, Andrew V. Goldberg, and David S. Johnson. Implementation challenge for shortest paths. In *Encyclopedia of Algorithms*. 2008. 15, 54

[DGKK79] Robert B. Dial, Fred Glover, David Karney, and Darwin Klingman. A computational analysis of alternative algorithms and labeling techniques for finding shortest path trees. *Networks*, 9:215–248, 1979. 33, 50

[DGST88] James R. Driscoll, Harold N. Gabow, Ruth Shrairman, and Robert E. Tarjan. Relaxed heaps: an alternative to Fibonacci heaps with applications to parallel computation. *Communications of the ACM*, 31(11):1343–1354, 1988. 33, 34

[DHZ00] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000. Announced at FOCS 1996. 27, 29, 37, 38

[DI04] Camil Demetrescu and Giuseppe F. Italiano. Engineering shortest path algorithms. In *Experimental and Efficient Algorithms, Third International Workshop, WEA 2004, Angra dos Reis, Brazil, May 25-28, 2004, Proceedings*, pages 191–198, 2004. 35

[DI08] Camil Demetrescu and Giuseppe F. Italiano. Decremental all-pairs shortest paths. In *Encyclopedia of Algorithms*. 2008. 32, 35

[Dia69] Robert B. Dial. Algorithm 360: shortest-path forest with topological ordering [h]. *Communications of the ACM*, 12(11):632–633, 1969. 33

[Dij59] Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 8, 32, 35, 51

[Dir50] Gustav Lejeune Dirichlet. Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal für die Reine und Angewandte Mathematik*, 40:209–227, 1850. 98

[Dji96] Hristo Djidjev. Efficient algorithms for shortest path problems on planar digraphs. In *Graph-Theoretic Concepts in Computer Science, 22nd International Workshop, WG '96, Cadenabbia (Como), Italy, June 12-14, 1996, Proceedings*, pages 151–165, 1996. 45, 47, 121

[Dji06] Hristo Djidjev. A scalable multilevel algorithm for graph clustering and community structure detection. In *Algorithms and Models for the Web-Graph, Fourth International Workshop, WAW 2006, Banff, Canada, November 30 - December 1, 2006. Revised Papers*, pages 117–128, 2006. 13

[Djo73] Dragomir Z. Djokovic. Distance-preserving subgraphs of hypercubes. *Journal of Combinatorial Theory, Series B*, 14(3):263–267, 1973. 30

[DJW07] Jörg Derungs, Riko Jacob, and Peter Widmayer. Approximate shortest paths guided by a small index. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 553–564, 2007. Journal version to appear in Algorithmica. 43

[DLL$^+$06] Debora Donato, Luigi Laura, Stefano Leonardi, Ulrich Meyer, Stefano Millozzi, and Jop F. Sibeyn. Algorithms and experiments for the webgraph. *Journal of Graph Algorithms and Applications*, 10(2):219–236, 2006. Announced at ESA 2003. 5

[DLO03] Erik D. Demaine and Alejandro López-Ortiz. A linear lower bound on index size for text retrieval. *Journal of Algorithms*, 48(1):2–15, 2003. Announced at SODA 2001. 43

[DMS00] Sergey N. Dorogovtsev, José Fernando Ferreira Mendes, and Alexander N. Samukhin. Structure of growing networks with preferential linking. *Physical Review Letters*, 85(21):4633–4636, Nov 2000. 24

[Dor67] Jim E. Doran. An approach to automatic problem-solving. *Machine Intelligence*, 1:105–124, 1967. 35, 52

[DP84] Narsingh Deo and Chi-Yin Pang. Shortest path algorithms: Taxonomy and annotation. *Networks*, 14:257–323, 1984. 32, 51

[DP08] Ran Duan and Seth Pettie. Bounded-leg distance and reachability oracles. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 436–445, 2008. 39

[DPW09] Daniel Delling, Thomas Pajor, and Dorothea Wagner. Accelerating multi-modal route planning by access-nodes. In *Algorithms - ESA 2009, 17th Annual Eu-*

ropean Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings, pages 587–598, 2009. 2

[DPWZ09] Daniel Delling, Thomas Pajor, Dorothea Wagner, and Christos D. Zaroliagis. Efficient route planning in flight networks. In *ATMOS 2009 - 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, IT University of Copenhagen, Denmark, September 10, 2009*, 2009. 2

[DPZ91] Hristo Djidjev, Grammati E. Pantziou, and Christos D. Zaroliagis. Computing shortest paths and distances in planar graphs. In *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, pages 327–338, 1991. 45

[DPZ95] Hristo Djidjev, Grammati E. Pantziou, and Christos D. Zaroliagis. On-line and dynamic algorithms for shorted path problems. In *STACS*, pages 193–204, 1995. 45, 121

[DPZ00] Hristo Djidjev, Grammati E. Pantziou, and Christos D. Zaroliagis. Improved algorithms for dynamic shortest paths. *Algorithmica*, 28(4):367–389, 2000. 45, 121

[DR94] Shaul Dar and Raghu Ramakrishnan. A performance study of transitive closure algorithms. *ACM SIGMOD Record*, 23(2):454–465, 1994. 57

[Dre69] Stuart E. Dreyfus. An appraisal of some shortest-path algorithms. *Operations Research*, 17(3):395–412, 1969. 32, 35, 50

[dSP65] Derek J. de Solla Price. Networks of scientific papers. *Science*, 149(3683):510–515, 1965. 4, 5

[dSP76] Derek J. de Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science*, 27(5):292–306, 1976. 5

[dSPK78] Ithiel de Sola Pool and Manfred Kochen. Contacts and influence. *Social Networks*, 1:5–51, 1978. 6

[DSSW06] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Highway hierarchies star. In *9th DIMACS Implementation Challenge*, 2006. 110, 114

[DSSW09] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks - Design, Analysis, and Simulation [DFG priority program 1126]*, pages 117–139, 2009. 54, 97

[DW07] Daniel Delling and Dorothea Wagner. Landmark-based routing in dynamic graphs. In *Experimental Algorithms, 6th International Workshop (WEA'07), Rome, Italy, June 6-8, 2007, Proceedings*, pages 52–65, 2007. 111, 117

[Eco09] The Economist. Rational consumer: The road ahead. *The Economist (Technology Quarterly)*, 392(8647):17, 2009. 122

[Edm65] Jack Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. 25

[EG60] Paul Erdős and Tibor Gallai. Grafok előirt fokú pontokkal. *Matematikai Lapok*, 11:264–274, 1960. English title: Graphs with points of prescribed degrees. 24

[EG08] David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*, page 16, 2008. 22, 31, 47, 103, 104

[Ege31] Jenő Egerváry. Matrixok kombinatorikus tulajdonságairól. *Középiskolai Matematikai és Fizikai Lapok*, 38:16–27, 1931. Translation by Harold W. Kuhn in Logistics Papers, issue 11, 1955. 8

[EGK$^+$04] Stephen Eubank, Hasan Guclu, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180–184, 2004. 7, 10

[EGS09] David Eppstein, Michael T. Goodrich, and Darren Strash. Linear-time algorithms for geometric graphs with sublinearly many crossings. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 150–159, 2009. 22

[EHP04] Jeff Erickson and Sariel Har-Peled. Optimally cutting a surface into a disk. *Discrete & Computational Geometry*, 31(1):37–59, 2004. Announced at SOCG 2002. 19

[EJ73] Jack Edmonds and Ellis L. Johnson. Matching, euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124, 1973. 8

[EJ08] Geoffrey Exoo and Robert Jajcay. Dynamic cage survey. *The Electronic Journal of Combinatorics*, 15, 2008. 28, 65

[EL82] R. J. Elliott and Michael Lesk. Route finding in street maps by computers and people. In *AAAI*, pages 258–261, 1982. 54

[Elk05] Michael L. Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms*, 1(2):283–323, 2005. Announced at PODC 2001. 38

[Elk08a] Michael L. Elkin. Sparse graph spanners. In *Encyclopedia of Algorithms*. 2008. 27

[Elk08b] Michael L. Elkin. Synchronizers, spanners. In *Encyclopedia of Algorithms*. 2008. 27

[Elm77] Salah E. Elmaghraby. *Activity Networks: Project Planning and Control by Network Models*. Wiley, New York, 1977. 8

[EM01] Stefan Edelkamp and Ulrich Meyer. Theory and practice of time-space tradeoffs in memory limited search. In *KI 2001: Advances in Artificial Intelligence, Joint German/Austrian Conference on AI, Vienna, Austria, September 19-21, 2001, Proceedings*, pages 169–184, 2001. 53

[EMK97] Paul Embrechts, Thomas Mikosch, and Claudia Klüppelberg. *Modelling extremal events: for insurance and finance*. Springer-Verlag, London, UK, 1997. 108

[EP04] Michael L. Elkin and David Peleg. (1+epsilon, beta)-spanner constructions for general graphs. *SIAM Journal on Computing*, 33(3):608–631, 2004. 27

[Epp99] David Eppstein. Subgraph isomorphism in planar graphs and related problems. *Journal of Graph Algorithms and Applications*, 3(3), 1999. Announced at SODA 1995. 45

[ER60] Paul Erdős and Alfréd A. Rényi. On the evolution of random graphs. *Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 5:17–61, 1960. 22, 23, 49

[ER97] Joost Engelfriet and Grzegorz Rozenberg. Node replacement graph grammars. In *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, pages 1–94, 1997. 48

[Erd35] Paul Erdős. Über die Primzahlen gewisser arithmetischer Reihen. *Mathematische Zeitschrift*, 39(1):473–491, 1935. 66

[Erd63] Paul Erdős. On a Combinatorial Problem, I. *Nordisk Matematisk Tidsskrift*, 11:5–10, 1963. 67, 68

[Erd64] Paul Erdős. Extremal problems in graph theory. *Theory Graphs Appl., Proc. Symp. Smolenice*, pages 29–36, 1964. 28, 41, 59

[ERS66] Paul Erdős, Alfréd A. Rényi, and Vera Turan Sos. On a problem of graph theory. *Studia Scientiarum Mathematicarum Hungarica*, 1:215–235, 1966. 28

[Erw00] Martin Erwig. The graph Voronoi diagram with applications. *Networks*, 36(3):156–163, 2000. 97, 98, 99, 101, 102, 103, 118

[ES63] Paul Erdős and Horst Sachs. Reguläre Graphen gegebener Taillenweite mit minimaler Knotenzahl. *Wissenschaftliche Zeitschrift der Martin-Luther-Universität Halle-Wittenberg, Mathematisch-Naturwissenschaftliche Reihe*, pages 251–258, 1963. 28, 41, 59

[EW03] Nils Eissfeldt and Peter Wagner. Effect of anticipatory driving in traffic flow model. *The European Physical Journal B*, 33:121–129, 2003. 10

[EW04] David Eppstein and Joseph Wang. Fast approximation of centrality. *Journal of Graph Algorithms and Applications*, 8:39–45, 2004. Announced at SODA 2001. 12

[EWG08] Mihaela Enachescu, Mei Wang, and Ashish Goel. Reducing maximum stretch in compact routing. In *INFOCOM*, pages 336–340, 2008. 49, 96

[FF58] Lester Randolph Ford and Delbert Ray Fulkerson. Constructing maximal dynamic flows from static flows. *Op. Res.*, 6(3):419–433, 1958. 8, 34

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM: Proceedings of the conference on applications, technologies, architectures, and protocols for computer communication*, pages 251–262, 1999. 5, 82, 125

[FFV05] Abraham D. Flaxman, Alan M. Frieze, and Juan Vera. Adversarial deletion in a scale free random graph process. In *Proceedings of the 16th annual ACM-SIAM symposium on Discrete algorithms*, pages 287–292, 2005. 83

[FFV07] Abraham D. Flaxman, Alan M. Frieze, and Juan Vera. A geometric preferential attachment model of networks ii. In *Algorithms and Models for the Web-Graph, 5th International Workshop, WAW 2007, San Diego, CA, USA, December 11-12, 2007, Proceedings*, pages 41–55, 2007. 4, 5

[FG85] Alan M. Frieze and Geoffrey R. Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10(1):57–77, 1985. 35

[FGG$^+$05] Qing Fang, Jie Gao, Leonidas J. Guibas, V. de Silva, and Li Zhang. GLIDER: gradient landmark-based distributed routing for sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 13-17 March 2005, Miami, FL, USA*, pages 339–350, 2005. 97

[FH04] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 11

[FJ88] Greg N. Frederickson and Ravi Janardan. Designing networks with compact routing tables. *Algorithmica*, 3:171–190, 1988. 45

[FJ89] Greg N. Frederickson and Ravi Janardan. Efficient message routing in planar networks. *SIAM Journal on Computing*, 18(4):843–857, 1989. 47

[FJ90] Greg N. Frederickson and Ravi Janardan. Space-efficient message routing in c-decomposable networks. *SIAM Journal on Computing*, 19(1):164–181, 1990. 47

[FK07] Martin Fürer and Shiva Prasad Kasiviswanathan. Spanners for geometric intersection graphs. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, pages 312–324, 2007. 48, 121

[FKS84] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $\mathcal{O}(1)$ worst case access time. *Journal of the ACM*, 31(3):538–544, 1984. Announced at FOCS 1982. 42, 89

[FKS89] Joel Friedman, Jeff Kahn, and Endre Szemerédi. On the second eigenvalue in random regular graphs. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, 15-17 May 1989, Seattle, Washington, USA*, pages 587–598, 1989. 65

[Fla63] C. Flament. *Applications of Graph Theory to Group Structure*. Prentice-Hall, Englewood Cliffs, 1963. 12

[FLM67] B. A. Farbey, A. H. Land, and J. D. Murchland. The cascade algorithm for finding all shortest distances in a directed graph. *Management Science*, 14(1):19–28, 1967. 35, 51

[Flo62] Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962. 8, 35, 37, 51

[FLV08] Pierre Fraigniaud, Emmanuelle Lebhar, and Laurent Viennot. The inframetric model for the internet. In *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 13-18 April 2008, Phoenix, AZ, USA*, pages 1085–1093, 2008. 49, 83

[FM95] Tomás Feder and Rajeev Motwani. Clique partitions, graph compression and speeding-up algorithms. *Journal of Computer and System Sciences*, 51(2):261–272, 1995. Announced at STOC 1991. 36, 37

[For56] Lester R. Ford. Network flow theory. Report P-923, The Rand Corporation, 1956. 8, 34, 35, 37

[FPP08] Alessandro Ferrante, Gopal Pandurangan, and Kihong Park. On the hardness of optimization in power-law graphs. *Theoretical Computer Science*, 393(1-3):220–230, 2008. 83

[FR06] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *Journal of Computer and System Sciences*, 72(5):868–889, 2006. Announced at FOCS 2001. 32, 44, 47, 121

[FR08] Jittat Fakcharoenphol and Satish Rao. Shortest paths in planar graphs with negative weight edges. In *Encyclopedia of Algorithms*. 2008. 44

[Fra08] Andrew U. Frank. Shortest path in a combined public transportation network. *KI*, 22(3):14–18, 2008. 2

[Fre76] Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5(1):83–89, 1976. 36, 37

[Fre77] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. 12

[Fre87] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16(6):1004–1022, 1987. 35

[Fre91] Greg N. Frederickson. Planar graph decomposition and all pairs shortest paths. *Journal of the ACM*, 38(1):162–204, 1991. 21, 45

[Fre95] Greg N. Frederickson. Using cellular graph embeddings in solving all pairs shortest paths problems. *Journal of Algorithms*, 19(1):45–85, 1995. Announced at FOCS 1989. 21, 45

[Fri64] Max Frisch. *Mein Name sei Gantenbein*. Suhrkamp, 1964. 81

[Fri76] Alan M. Frieze. Shortest path algorithms for knapsack type problems. *Mathematical Programming*, 11(1):150–157, 1976. 8

[Fri91] Joel Friedman. On the second eigenvalue and random walks in random d-regular graphs. *Combinatorica*, 11(4):331–362, 1991. 65

[FS97] Andrew Fetterer and Shashi Shekhar. A performance analysis of hierarchical shortest path algorithms. In *ICTAI*, pages 84–93, 1997. 51, 54

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987. Announced at FOCS 1984. 33, 34, 103

[FTW87] Martin A. Fischler, Jay M. Tenenbaum, and Helen C. Wolf. Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. In *Readings in computer vision: issues, problems, principles, and paradigms*, pages 741–752, 1987. 11

[FUS$^+$98] Alexandre X. Falcao, Jayaram K. Udupa, Supun Samarasekera, Shoba Sharma, Bruce Elliot Hirsch, and Roberto de A. Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical Models and Image Processing*, 60(4):233 – 260, 1998. 8, 10

[FVC07] Alan M. Frieze, Juan Vera, and Soumen Chakrabarti. The influence of search engines on preferential attachment. *Internet Mathematics*, 3(3), 2007. 5

[FW93] Michael L. Fredman and Dan E. Willard. Surpassing the information theoretic bound with fusion trees. *Journal of Computer and System Sciences*, 47(3):424–436, 1993. 34

[FW94] Michael L. Fredman and Dan E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences*, 48(3):533–551, 1994. Announced at FOCS 1990. 34

[GA06] Christopher M. Gold and Paul Angel. Voronoi hierarchies. In *Geographic Information Science, 4th International Conference, GIScience 2006, Münster, Germany, September 20-23, 2006, Proceedings*, pages 99–111, 2006. 118

[Gal58] Tibor Gallai. Maximum-minimum Sätze über Graphen. *Acta Mathematica Academiae Scientiarum Hungaricae*, 9:395–434, 1958. 8

[Gav01] Cyril Gavoille. Routing in distributed networks: overview and open problems. *SIGACT News*, 32(1):36–52, 2001. 49

[GBB$^+$03] Loic Giot, Joel S. Bader, C. Brouwer, Amitabha Chaudhuri, Bing Kuang, Y. Li, YL. Hao, CE. Ooi, Brian Godwin, E. Vitols, G. Vijayadamodar, Philippe Pochart, H. Machineni, M. Welsh, Y. Kong, B. Zerhusen, Robert J. Malcolm, Z. Varrone, A. Collis, M. Minto, S. Burgess, L. McDaniel, E. Stimpson, F. Spriggs, J. Williams, Kathryin Neurath, N. Ioime, M. Agee, E. Voss, K. Furtak, R. Renzulli, N. Aanensen, S. Carrolla, E. Bickelhaupt, Y. Lazovatsky, A. DaSilva, J. Zhong, Clement A. Stanyon, Russell L. Finley Jr, Kevin P. White, Michael S. Braverman, Thomas P. Jarvie, S. Gold, M. Leach, J. Knight, Richard A. Shimkets, Michael P. McKenna, John Chant, and Jonathan M. Rothberg. A protein interaction map of Drosophila melanogaster. *Science*, 302(5651):1727–1736, 2003. 6

[GBL08] Amit Goyal, Francesco Bonchi, and Laks V.S. Lakshmanan. Discovering leaders from community actions. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 499–508, 2008. 7

[GCE$^+$09] Emanuele Galli, Leticia Cuellar, Stephan Eidenbenz, Mary Ewers, Sue Mniszewski, and Christof Teuscher. Activitysim: large-scale agent-based activity generation for infrastructure simulation. In *Proceedings of the 2009 Spring Simulation Multiconference, SpringSim 2009, San Diego, California, USA, March 22-27, 2009*, 2009. 7

[Gel63] Herbert L. Gelernter. Realization of a geometry theorem proving machine. *Computers and Thought*, 1963. 35, 52

[Gel77] David Gelperin. On the optimality of A*. *Artificial Intelligence*, 8(1):69–76, 1977. 35, 52

[GGM06] Jesus Gomez-Gardenes and Yamir Moreno. From scale-free to Erdos-Renyi networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 73(5):056124, 2006. 4

[GH05] Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A* search meets graph theory. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05), Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 156–165, 2005. 53, 91, 111, 114

[GHK90] Donald Goldfarb, Jianxiu Hao, and Sheng-Roan Kai. Efficient shortest path simplex algorithms. *Operations Research*, 38(4):624–628, 1990. 34

[GHT84] John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, 1984. 31

[Gil59] Edgar N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30:1141–1144, 1959. 22, 23

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990. 26

[GJ99] Donald Goldfarb and Zhiying Jin. An $\mathcal{O}(nm)$-time network simplex algorithm for the shortest path problem. *Operations Research*, 47(3):445–448, 1999. 34

[GKK$^+$01] Cyril Gavoille, Michal Katz, Nir A. Katz, Christophe Paul, and David Peleg. Approximate distance labeling schemes. In *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, pages 476–487, 2001. 48

[GKN74] Fred Glover, D. Klingman, and A. Napier. A note on finding all shortest paths. *Transportation Science*, 8:3–12, 1974. 35, 51

[GKP85] Fred Glover, Darwin D. Klingman, and Nancy V. Phillips. A new polynomially bounded shortest path algorithm. *Operations Research*, 33(1):65–73, 1985. 34

[GKP05] Naveen Garg, Rohit Khandekar, and Vinayaka Pandit. Improved approximation for universal facility location. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'05), Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 959–960, 2005. 98

[GKPS85] Fred Glover, Darwin D. Klingman, Nancy V. Phillips, and Robert F. Schneider. New polynomial shortest path algorithms and their computational attributes. *Management Science*, 31(9):1106–1128, 1985. 34

[GKR04] Sandeep Gupta, Swastik Kopparty, and Chinya Ravishankar. Roads, codes, and spatiotemporal queries. In *PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 115–124, 2004. 45, 55, 121

[GKW06] Andrew Goldberg, Haim Kaplan, and Renato Fonseca F. Werneck. Reach for A*: Efficient point-to-point shortest path algorithms. In *Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX06)*, pages 129–143. SIAM, 2006. 53

[GKW07] Andrew V. Goldberg, Haim Kaplan, and Renato Fonseca F. Werneck. Better landmarks within reach. In *Experimental Algorithms, 6th International Workshop, WEA 2007, Rome, Italy, June 6-8, 2007, Proceedings*, pages 38–51, 2007. 51, 53, 105

[GL07] Cyril Gavoille and Arnaud Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 582–593, 2007. 31

[GLNS08] Joachim Gudmundsson, Christos Levcopoulos, Giri Narasimhan, and Michiel H. M. Smid. Approximate distance oracles for geometric spanners. *ACM Transactions on Algorithms*, 4(1), 2008. 50, 121

[GM77] Bruce Golden and Thomas L. Magnanti. Deterministic network optimization: A bibliography. *Networks*, 7:149–183, 1977. 32

[GM93] Zvi Galil and Oded Margalit. Witnesses for boolean matrix multiplication and for transitive closure. *Journal of Complexity*, 9(2):201–221, 1993. 36

[GM97] Zvi Galil and Oded Margalit. All pairs shortest distances for graphs with small integer length edges. *Information and Computation*, 134(2):103–139, 1997. 36

[GMMO00] Sudipto Guha, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 359–366, 2000. 13

[GN64] Allan L. Gutjahr and George L. Nemhauser. An algorithm for the line balancing problem. *Management Science*, 11(2):308–315, 1964. 8

[GN67] A. J. Goldman and G. L. Nemhauser. A transport improvement problem transformable to a best-path problem. *Transportation Science*, 1(4):295–307, 1967. 8

[GN02] M. Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. 13

[Gol76] Bruce Golden. Shortest-path algorithms: A comparison. *Operations Research*, 24(6):1164–1168, 1976. 35, 50

[Gol95] Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, 1995. Announced at SODA 1993. 32

[Gol01] Andrew V. Goldberg. Shortest path algorithms: Engineering aspects. In *Algorithms and Computation, 12th International Symposium, ISAAC 2001, Christchurch, New Zealand, December 19-21, 2001, Proceedings*, pages 502–513, 2001. 35

[Gol07] Andrew V. Goldberg. Point-to-point shortest path algorithms with preprocessing. In *SOFSEM 2007: Theory and Practice of Computer Science, 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 20-26, 2007, Proceedings*, pages 88–102, 2007. 54

[Gol08] Andrew V. Goldberg. A practical shortest path algorithm with linear expected time. *SIAM Journal on Computing*, 37(5):1637–1655, 2008. 35

[GOY76] Satoshi Goto, Tatsuo Ohtsuki, and Takeshi Yoshimura. Sparse matrix techniques for the shortest path problem. *IEEE Transactions on Circuits and Systems*, 23(12):752–758, Dec 1976. 35

[GP72] Ronald L. Graham and Henry O. Pollak. *On embedding graphs in squashed cubes*, volume 303, pages 99–110. 1972. 30

[GP86] Giorgio Gallo and Stefano Pallottino. Shortest path methods: A unifying approach. *Mathematical Programming Studies*, 26:38–64, 1986. 34

[GP03a] Cyril Gavoille and Christophe Paul. Optimal distance labeling for interval and circular-arc graphs. In *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, pages 254–265, 2003. 48

[GP03b] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2-3):111–120, 2003. 29, 49

[GPPR04] Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004. Announced at SODA 2000. 29

[Gra06] Leo Grady. Computing exact discrete minimal surfaces: Extending and solving the shortest path problem in 3d with application to segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 69–78, 2006. 11

[Gra08] Leo Grady. Minimal surfaces extend shortest path segmentation methods to 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1, December 2008. 11

[GSDF03] Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors. *SIGKDD Proceedings*, 2003. 112

[GSSD08] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Experimental Algorithms, 7th International Workshop (WEA'08), Provincetown, MA, USA, May 30-June 1, 2008, Proceedings*, pages 319–333, 2008. 47, 51, 54, 55, 56, 111, 112, 114, 117, 123

[GSVGM98] Roy Goldman, Narayanan Shivakumar, Suresh Venkatasubramanian, and Hector Garcia-Molina. Proximity search in databases. In *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 26–37, 1998. 57, 83

[GT89] Harold N. Gabow and Robert Endre Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989. 32

[Gua93] John Guare. Six degrees of separation, 1993. Movie. 6

[Gut04] Ron Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 100–111. SIAM, 2004. 53, 56

[GW73] David E. Gilsinn and Christoph Witzgall. A performance comparison of labeling algorithms for calculating shortest path trees. Technical Note 772, National Institute of Standards and Technology, 1973. 35, 50

[GW05] Andrew V. Goldberg and Renato Fonseca F. Werneck. Computing point-to-point shortest paths from external memory. In *Proceedings of the Seventh Workshop on Algorithm Engineering and Experiments and the Second Workshop on Analytic Algorithmics and Combinatorics, ALENEX /ANALCO 2005, Vancouver, BC, Canada, 22 January 2005*, pages 26–40, 2005. 53, 54, 111

[GYY80] Ronald L. Graham, Andrew Chi-Chih Yao, and F. Frances Yao. Information bounds are weak in the shortest distance problem. *Journal of the ACM*, 27(3):428–444, 1980. 35

[GZ05] Jie Gao and Li Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing*, 35(1):151–169, 2005. 48, 121

[GZC$^+$09] Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*, pages 53–60, 2009. 13

[Hag00a] Torben Hagerup. Dynamic algorithms for graphs of bounded treewidth. *Algorithmica*, 27(3):292–315, 2000. Announced at ICALP 1997. 48

[Hag00b] Torben Hagerup. Improved shortest paths on the word RAM. In *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000, Geneva, Switzerland, July 9-15, 2000, Proceedings*, pages 61–72, 2000. 33

[Hag06] Torben Hagerup. Simpler computation of single-source shortest paths in linear average time. *Theory of Computing Systems*, 39(1):113–120, 2006. 35

[Hak62] Seifollah Louis Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962. 24

[Hal76] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976. 20

[Han87] Eric N. Hanson. A performance analysis of view materialization strategies. *SIGMOD Rec.*, 16(3):440–453, 1987. 10

[Han04] Yijie Han. Improved algorithm for all pairs shortest paths. *Information Processing Letters*, 91(5):245–250, 2004. 37

[Han08a] Yijie Han. A note of an $\mathcal{O}(n^3/\log n)$ time algorithm for all pairs shortest paths. *Information Processing Letters*, 105(3):114–116, 2008. 37

[Han08b] Yijie Han. An $\mathcal{O}(n^3(\log\log n/\log n)^{5/4})$ time algorithm for all pairs shortest path. *Algorithmica*, 51(4):428–434, 2008. Announced at ESA 2006. 37

[Hay06] Brian Hayes. Connecting the dots. *Computing Science*, 94(5):400, 2006. 4

[Hel53] Isidor Heller. On the problem of shortest path between points, I. *Bulletin of the American Mathematical Society*, 59, 1953. A study of the five-city Traveling Salesman Problem polytope. 8

[HHSW09] Shinichi Honiden, Michael E. Houle, Christian Sommer, and Martin Wolff. Approximate shortest path queries in graphs using Voronoi duals. In *Sixth annual International Symposium on Voronoi Diagrams in Science and Engineering (ISVD)*, pages 53–62, 2009. Invited to special issue of Transactions on Computational Science. 16

[Hit68] Lewis E. Hitchner. A comparative investigation of the computational efficiency of shortest path algorithms. Technical Report ORC 68-17, University of California at Berkeley, 1968. 35, 50

[HJR95] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Hierarchical path views: A model based on fragmentation and transportation road types. In *ACM-GIS*, pages 93–, 1995. 51, 54

[HJR96a] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Effective graph clustering for path queries in digital map databases. In *CIKM '96, Proceedings of the Fifth*

International Conference on Information and Knowledge Management, November 12 - 16, 1996, Rockville, Maryland, USA, pages 215–222, 1996. 54

[HJR96b] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Path queries for transportation networks: Dynamic reordering and sliding window paging techniques. In *GIS '96, Proceedings of the fourth ACM workshop on Advances on Advances in Geographic Information Systems, November 15-16, 1996, Rockville, Maryland, USA*, pages 9–16, 1996. 54

[HJR97a] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. A hierarchical path view model for path finding in intelligent transportation systems. *GeoInformatica*, 1(2):125–159, 1997. 54

[HJR97b] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Integrated query processing strategies for spatial path queries. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K*, pages 477–486, 1997. 54

[HJR00] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Optimizing path query performance: Graph clustering strategies, 2000. 54

[HKRS97] Monika Rauch Henzinger, Philip Nathan Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997. Announced at STOC 1994. 35, 103

[HLL06] Haibo Hu, Dik Lun Lee, and Victor C. S. Lee. Distance indexing on road networks. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 894–905, 2006. 54

[HLW98] Abolhassan Halati, Henry Lieu, and Susan Walker. CORSIM-corridor traffic simulation model. In *Proceedings at the traffic congesion and traffic safety conference*, pages 570–576, 1998. 10

[HLW06] Shlomoh Hoory, Nati Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006. 2, 65

[HMZ03] David A. Hutchinson, Anil Maheshwari, and Norbert Zeh. An external memory data structure for shortest path queries. *Discrete Applied Mathematics*, 126(1):55–82, 2003. 45

[HNR68] Peter E. Hart, Nils J. Nilsson, and Bertram R. Raphael. 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, 1968. 35, 52

[Hol04] Johan Holmgren. Efficient updating shortest path calculations for traffic assignment. Technical Report LITH-MAI-EX-2004-13, Division of Optimization, Department of Mathematics, Linkoping Institute of Technology, Linkoping, October 2004. 10

[Hol08] Martin Holzer. *Engineering Planar-Separator and Shortest-Path Algorithms*. PhD thesis, Universität Karlsruhe, 2008. 51, 54

[Hoo02] Shlomo Hoory. *On graphs of high girth*. PhD thesis, Hebrew University, 2002. 28

[HP58] Walter Hoffman and Richard Pavley. Applications of digital computers to problems in the study of vehicular traffic. In *IRE-ACM-AIEE '58 (Western): Proceedings of the May 6-8, 1958, western joint computer conference: contrasts in computers*, pages 159–161, 1958. 10, 54

[HPCD96] John Joseph Helmsen, Elbridge Gerry Puckett, Phillip Colella, and M. Dorr. Two new methods for simulating photolithography development in 3D. In *Proceedings of the SPIE – The International Society for Optical Engineering Optical Microlithography IX*, pages 253–261, 1996. 11

[HSP92] Jianying Hu, William J. Sakoda, and Theodosios Pavlidis. Interactive road finding for aerial images. In *IEEE Workshop on Applications of Computer Vision*, pages 56–63, 1992. 11

[HST09] Bernhard Haeupler, Siddhartha Sen, and Robert Endre Tarjan. Rank-pairing heaps. In *Algorithms - ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, pages 659–670, 2009. 33

[HSW08] Martin Holzer, Frank Schulz, and Dorothea Wagner. Engineering multilevel overlay graphs for shortest-path queries. *ACM Journal of Experimental Algorithmics*, 13, 2008. 51, 54

[HSWW05] Martin Holzer, Frank Schulz, Dorothea Wagner, and Thomas Willhalm. Combining speed-up techniques for shortest-path computations. *ACM Journal of Experimental Algorithmics*, 10, 2005. 54, 55

[HT69] Te C. Hu and William T. Torres. Shortcut in the decomposition algorithm for shortest paths in a network. *IBM Journal of Research and Development*, 13(4):387–390, 1969. 35, 36, 51, 125

[HT02] Yijie Han and Mikkel Thorup. Integer sorting in $\mathcal{O}(n\sqrt{\log\log n})$ expected time and linear space. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 135–144, 2002. 34

[HTB01] Michelle R. Hribar, Valerie E. Taylor, and David E. Boyce. Implementing parallel shortest path for parallel transportation applications. *Parallel Computing*, 27(12):1537–1568, 2001. 35

[Hu67] Te C. Hu. Revised matrix algorithms for shortest paths. *SIAM Journal on Applied Mathematics*, 15(1):207–218, 1967. 8

[Hu68] Te C. Hu. A decomposition algorithm for shortest paths in a network. *Operations Research*, 16(1):91–102, 1968. 35, 51

[Hu69] Te C. Hu. *Integer Programming and Network Flows*. Addison Wesley, 1969. 51, 55

[Hu71] Te C. Hu. Some problems in discrete optimization. *Mathematical Programming*, 1:102–112, 1971. 9

[Hui00] Christian Huitema. *Routing in the Internet (2nd ed.)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000. 14

[HW07] Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007. 62

[HWYY05] Hao He, Haixun Wang, Jun Yang, and Philip S. Yu. Compact reachability labeling for graph-structured data. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 594–601, 2005. 57

[ICO$^+$01] Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proceedings of the National Academy of Sciences*, 98(8):4569–4574, 2001. 6

[IFB$^+$02] Jan Ihmels, Gilgi Friedlander, Sven Bergmann, Ofer Sarig, Yaniv Ziv, and Naama Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31(4):370–377, 2002. 6

[IHI$^+$94] Takahiro Ikeda, Min-Yao Hsu, Hiroshi Imai, Shigeki Nishimura, Hiroshi Shimoura, Takeo Hashimoto, Kenji Tenmoku, and Kunihiko Mitoh. A fast algorithm for finding better routes by AI search techniques. In *Vehicle Navigation and Information Systems Conference*, pages 291–296, 1994. 35, 53

[II84] Hiroshi Imai and Masao Iri. Practical efficiencies of existing shortest-path algorithms and a new bucket algorithm. *Journal of the Operations Research Society of Japan*, 27(1):43–58, 1984. 35

[II86] Hiroshi Imai and Masao Iri. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, 36(1):31–41, 1986. 8

[IM04] Pitor Indyk and Jirí Matousek. *CRC Handbook of Discrete and Computational Geometry, 2nd edition*, chapter Low-Distortion Embeddings of Finite Metric Spaces, pages 177–196. 2004. 29

[IN72] Masao Iri and Mario Nakamori. Path-sets, operator semigroups and shortest path algorithms on a network. Research Association of Applied Geometry (RAAG) memoirs of the unifying study of basic problems in engineering and physical sciences by means of geometry 185, 3rd series, 1972. 36

[Ind01] Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 10–33, 2001. 29

[IOAI91] Kunihiro Ishikawa, Michima Ogawa, Shigetoshi Azuma, and Tooru Ito. Map navigation software of the electro-multivision of the '91 Toyoto Soarer. In *Vehicle Navigation and Information Systems Conference, 1991*, volume 2, pages 463–473, Oct. 1991. 51, 54

[Iri92] Masao Iri. How to generate realistic sample problems for network optimization. In *ISAAC '92: Proceedings of the Third International Symposium on Algorithms and Computation*, pages 342–350, London, UK, 1992. Springer-Verlag. 22

[Ita08] Giuseppe F. Italiano. Fully dynamic all pairs shortest paths. In *Encyclopedia of Algorithms*. 2008. 32, 35, 39

[Jac08] Riko Jacob. Shortest paths approaches for timetable information. In *Encyclopedia of Algorithms*. 2008. 56

[Jag90] H. V. Jagadish. A compression technique to materialize transitive closure. *ACM Transactions on Database Systems*, 15(4):558–598, 1990. 57

[JBIH05] Maliackal Poulo Joy, Amy Brock, Donald E. Ingber, and Sui Huang. High-betweenness proteins in the yeast protein interaction network. *Journal of Biomedicine and Biotechnology*, 2005(2):96–103, 2005. 12

[JHR96] Ning Jing, Yun-Wu Huang, and Elke A. Rundensteiner. Hierarchical optimization of optimal path finding for transportation applications. In *CIKM '96, Proceedings of the Fifth International Conference on Information and Knowledge Management, November 12 - 16, 1996, Rockville, Maryland, USA*, pages 261–268, 1996. 10, 51, 54

[JHR98] Ning Jing, Yun-Wu Huang, and Elke A. Rundensteiner. 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, 1998. 54

[JL84] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984. 30, 50

[JMBO01] Hawoong Jeong, S. P. Mason, Albert-László Barabási, and Zoltan N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001. 12

[JMN99] Riko Jacob, Madhav V. Marathe, and Kai Nagel. A computational study of routing algorithms for realistic transportation networks. *ACM Journal of Experimental Algorithmics*, 4:6, 1999. 35

[JOB03] Hawoong Jeong, Zoltan N. Oltvai, and Albert-László Barabási. Prediction of protein essentiality based on genomic data. *Complexus*, 1(1):19–28, 2003. 6, 12

[Joh72] Ellis L. Johnson. On shortest paths and sorting. In *ACM'72: Proceedings of the ACM annual conference*, pages 510–517, 1972. 33

[Joh77] Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977. 33, 35

[Joh82] Donald B. Johnson. A priority queue in which initialization and queue operations take $\mathcal{O}(\log\log D)$ time. *Mathematical Systems Theory*, 15(4):295–309, 1982. 34

[Joh97] Peter Johansson. On a weighted distance model for injection moulding. Master's thesis, Linköping University, 1997. 8

[JP02] Sungwon Jung and Sakti Pramanik. An efficient path computation model for hierarchically structured topographical road maps. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1029–1046, 2002. 51, 54

[JW03] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 271–279, 2003. 13

[JXRF09] Ruoming Jin, Yang Xiang, Ning Ruan, and David Fuhry. 3-HOP: a high-compression indexing scheme for reachability query. In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 813–826, 2009. 54

[JXRW08] Ruoming Jin, Yang Xiang, Ning Ruan, and Haixun Wang. Efficiently answering reachability queries on very large directed graphs. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 595–608, 2008. 54

[JZ05] Hema Jampala and Norbert Zeh. Cache-oblivious planar shortest paths. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005*, pages 563–575, 2005. 35

[Kar29] Frigyes Karinthy. *Lancszemek*. 1929. 6

[Ker70] Leslie Robert Kerr. *The effect of algebraic structure on the computational complexity of matrix multiplication*. PhD thesis, Cornell University, Ithaca, NY, USA, 1970. 35

[Ker04] Boris S. Kerner. *The Physics of Traffic*. Springer, 2004. 122

[KFY04] Dmitri V. Krioukov, Kevin R. Fall, and Xiaowei Yang. Compact routing on internet-like graphs. In *INFOCOM*, 2004. 49, 81, 82, 121, 125

[KG92] John Mark Keil and Carl Andrew Gutwin. Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry*, 7:13–28, 1992. 118

[KH05] Sebastian König and Juergen Hesser. 3D live-wires on pre-segmented volume data. *Medical Imaging 2005: Image Processing*, 5747(1):1674–1681, 2005. 11

[KH08] Hitoshi Kanoh and Kenta Hara. 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 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008*, pages 657–664, 2008. 54

[KHI+86] Ru-Mei Kung, Eric N. Hanson, Yannis E. Ioannidis, Timos K. Sellis, Leonard D. Shapiro, and Michael Stonebraker. Heuristic search in database systems. In *Proceedings from the first international workshop on Expert database systems*, pages 537–548, Redwood City, CA, USA, 1986. Benjamin-Cummings Publishing Co., Inc. 35, 52

[KIK+06] Shin'ichi Konomi, Sozo Inoue, Takashi Kobayashi, Masashi Tsuchida, and Masaru Kitsuregawa. Supporting colocated interactions using RFID and social network displays. *IEEE Pervasive Computing*, 5(3):48–56, 2006. 14

[KK77] Leonard Kleinrock and Farouk Kamoun. Hierarchical routing for large networks; performance evaluation and optimization. *Computer Networks*, 1:155–174, 1977. 51

[KK03] Lukasz Kowalik and Maciej Kurowski. Short path queries in planar graphs in constant time. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 143–148, 2003. 45

[KKK+07] Hans-Peter Kriegel, Peer Kröger, Peter Kunath, Matthias Renz, and Tim Schmidt. Proximity queries in large traffic networks. In *15th ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, November 7-9, 2007, Seattle, Washington, USA, Proceedings*, page 21, 2007. 54

[KKK+08] Hans-Peter Kriegel, Peer Kröger, Peter Kunath, Matthias Renz, and Tim Schmidt. Efficient query processing in large traffic networks. In *Proceedings of the 24th*

International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México, pages 1451–1453, 2008. 54

[KKP93] David R. Karger, Daphne Koller, and Steven J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. *SIAM Journal on Computing*, 22(6):1199–1217, 1993. Announced at FOCS 1991. 35

[KKR$^+$99] Jon Michael Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. The web as a graph: Measurements, models, and methods. In *COCOON*, pages 1–17, 1999. 5

[KKR08] Hans-Peter Kriegel, Peer Kröger, and Matthias Renz. Continuous proximity monitoring in road networks. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*, page 12, 2008. 54

[KKRS08] Hans-Peter Kriegel, Peer Kröger, Matthias Renz, and Tim Schmidt. Hierarchical graph embedding for efficient query processing in very large traffic networks. In *Scientific and Statistical Database Management, 20th International Conference, SSDBM 2008, Hong Kong, China, July 9-11, 2008, Proceedings*, pages 150–167, 2008. 51, 54

[KL95] Drago Krznaric and Christos Levcopoulos. Computing hierarchies of clusters from the euclidean minimum spanning tree in linear time. In *Foundations of Software Technology and Theoretical Computer Science, 15th Conference, Bangalore, India, December 18-20, 1995, Proceedings*, pages 443–455, 1995. 50

[KL06] Ravi Kumar and Matthieu Latapy. Preface. *Theoretical Computer Science: special issue on Complex Networks*, 355(1):1–5, 2006. 4

[Kle63] Morton Klein. On assembly line balancing. *Operations Research*, 11(2):274–281, 1963. 8

[Kle64] Victor LaRue Klee. A 'string algorithm' for shortest path in directed networks. *Operations Research*, 12(3):428–432, 1964. 8, 35

[Kle00] Jon Michael Kleinberg. Navigation in a small world. *Nature*, 406(6798):845, 2000. 24, 83

[Kle02a] Philip Nathan Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *Symposium on Discrete Algorithms (SODA)*, pages 820–827, 2002. 46, 47, 121

[Kle02b] Judith S. Kleinfeld. Six degrees of separation: Urban myth? *Psychology Today*, 35(2):74, 2002. 6

[Kle02c] Judith S. Kleinfeld. The small world problem. *Society*, 39(2):61–66, 2002. 6

[Kle05] Philip Nathan Klein. Multiple-source shortest paths in planar graphs. In *Symposium on Discrete Algorithms (SODA)*, pages 146–155, 2005. 44, 46, 47, 121

[KLMR98] Lydia E. Kavraki, Jean-Claude Latombe, Rajeev Motwani, and Prabhakar Raghavan. Randomized query processing in robot path planning. *Journal of Computer and System Sciences*, 57(1):50–66, 1998. 39

[KMS05] Ekkehard Köhler, Rolf H. Möhring, and Heiko Schilling. Acceleration of shortest path and constrained shortest path computation. In *Experimental and Efficient*

Algorithms, 4th InternationalWorkshop, WEA 2005, Santorini Island, Greece, May 10-13, 2005, Proceedings, pages 126–138, 2005. 54

[KN96] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1996. 61

[KNR92] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5(4):596–603, 1992. Announced at STOC 1988. 31

[Knu89] Donald Ervin Knuth. Theory and practice. Available online as P138, August 1989. 15, 97

[Koe36] Paul Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften zu Leipzig, Mathematisch-Physikalische Klasse*, 88:141–164, 1936. contains the circle packing theorem. 22, 48

[Kor85] Richard E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, 1985. 52

[Kor01] Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001. 27

[KP69] Ronald F. Kirby and Renfrey B. Potts. The minimum route problem for networks with turn penalties and prohibitions. *Transportation Research*, 3(3):397–408, 1969. 39

[KPSZ96] Dimitris J. Kavvadias, Grammati E. Pantziou, Paul G. Spirakis, and Christos D. Zaroliagis. Hammock-on-ears decomposition: A technique for the efficient parallel solution of shortest paths and other problems. *Theoretical Computer Science*, 168(1):121–154, 1996. 21

[KRR$^+$00] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D. Sivakumar, Andrew Tomkins, and Eli Upfal. Random graph models for the web graph. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 57–65, 2000. 5, 24

[KRX07] Goran Konjevod, Andréa W. Richa, and Donglin Xia. Optimal scale-free compact routing schemes in networks of low doubling dimension. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 939–948, 2007. 49

[KS92] Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992. Announced at “Structure in Complexity Theory” 1987. 62

[KS93a] David E. Kaufman and Robert L. Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1(1):1–11, 1993. 32

[KS93b] Philip Nathan Klein and Sairam Subramanian. A linear-processor polylog-time algorithm for shortest paths in planar graphs. In *34th Annual Symposium on Foundations of Computer Science, 3-5 November 1993, Palo Alto, California, USA*, pages 259–270, 1993. 35

[KS96] Vijay Kumar and Eric J. Schwabe. Improved algorithms and data structures for solving graph problems in external memory. In *Eighth IEEE Symposium on Parallel and Distributed Processing*, pages 169–176, Oct 1996. 35

[KS98] Philip Nathan Klein and Sairam Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, 1998. 47

[KSS97a] Henry A. Kautz, Bart Selman, and Mehul A. Shah. The hidden web. *AI Magazine*, 18(2):27–36, 1997. 14

[KSS97b] Henry A. Kautz, Bart Selman, and Mehul A. Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997. 14

[KSW09] Jon Michael Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. *Journal of the ACM*, 56(6), 2009. Announced at FOCS 2004. 49, 53, 56, 91

[KU08] Tomoya Kambara and Shinichi Ueshima. Proposal of graph-partitioning tree using network Voronoi diagram and its application to shortest path problems. *Journal of the Database Society of Japan*, 7(1):193–198, 2008. 97

[Kuh55] Harold William Kuhn. On certain convex polyhedra. *Bulletin of the American Mathematical Society*, 61:557–558, 1955. 8

[KvK09] Steffen Klamt and Axel von Kamp. Computing paths and cycles in biological interaction graphs. *BMC Bioinformatics*, 10(1):181, 2009. 11

[KW90] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990. 60

[KY65] M. Kitamura and M. Yamazaki. On the connection of the two shortest route systems. In *Proceedings of the 8th Japanese Road Conference*, pages 66–68, 1965. In Japanese. 35, 51

[LAB$^+$04] Siming Li, Christopher M. Armstrong, Nicolas Bertin, Hui Ge, Stuart Milstein, Mike Boxem, Pierre-Olivier Vidalain, Jing-Dong J. Han, Alban Chesneau, Tong Hao, Debra S. Goldberg, 1 Monica Martinez Ning Li, Jean-Francois Rual, Philippe Lamesch, Lai Xu, Muneesh Tewari, Sharyl L. Wong, Lan V. Zhang, Gabriel F. Berriz, Laurent Jacotot, Philippe Vaglio, Jerome Reboul, Tomoko Hirozane-Kishikawa, Qianru Li, Harrison W. Gabel, Ahmed Elewa, Bridget Baumgartner, Debra J. Rose, Haiyuan Yu, Stephanie Bosak, Reynaldo Sequerra, Andrew Fraser, Susan E. Mango, William M. Saxton, Susan Strome, Sander van den Heuvel, Fabio Piano, Jean Vandenhaute, Claude Sardet, Mark Gerstein, Lynn Doucette-Stamm, Kristin C. Gunsalus, J. Wade Harper, Michael E. Cusick, Frederick P. Roth, David E. Hill, and Marc Vidal. A map of the interactome network of the metazoan C. elegans. *Science*, 303(5657):540–543, 2004. 6

[LADW05] Lun Li, David Alderson, John C. Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005. 85

[Lan09] Edmund Landau. *Handbuch der Lehre von der Verteilung der Primzahlen*. 1909. 25

[Lat91] Jean-Claude Latombe. *Robot Motion Planning*. 1991. 39

[Lau04] Ulrich Lauther. 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*, volume 22, pages 219–230, 2004. 54, 110

[LCL$^+$94] Bing Liu, Siew-Hwee Choo, Shee-Ling Lok, Sing-Meng Leong, Soo-Chee Lee, Foong-Ping Poon, and Hwee-Har Tan. Integrating case-based reasoning, knowledge-based approach and Dijkstra algorithm for route finding. In *Proceedings of the Tenth Conference on Artificial Intelligence for Applications*, pages 149–155, Mar 1994. 54

[LDSS99] Aaron W. F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution mesh morphing. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 343–350. ACM Press/Addison-Wesley Publishing Co., 1999. 11

[LEA$^+$01] Fredrik Liljeros, Christofer R. Edling, Luis A. Nunes Amaral, H. Eugene Stanley, and Yvonne Aberg. The web of human sexual contacts. *Nature*, 411:907–908, 2001. 7

[Ley02] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *String Processing and Information Retrieval, 9th International Symposium (SPIRE'02), Lisbon, Portugal, September 11-13, 2002, Proceedings*, pages 1–10, 2002. 111

[LGJ$^+$57] M. Leyzorek, R. S. Gray, A. A. Johnson, W. C. Ladew, S. R. Meaker Jr, R. M. Petry, and R. N. Seitz. Investigation of model techniques — first annual report — a study of model techniques for communication systems. Case Institute of Technology, Cleveland, Ohio, 1957. 8, 32

[LH08] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 915–924, 2008. 7

[Lin02] Nathan Linial. Finite metric spaces – combinatorics, geometry and algorithms. In *Proceedings of the International Congress of Mathematicians III*, pages 573–586, 2002. 29

[Liu95] Bing Liu. Using knowledge to isolate search in route finding. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 119–124, 1995. 54

[LLR95] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. Announced at FOCS 1994. 30

[LMF$^+$07] Jure Leskovec, Mary McGlohon, Christos Faloutsos, Natalie S. Glance, and Matthew Hurst. Patterns of cascading behavior in large blog graphs. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, 2007. 5

[LMN02] Nathan Linial, Avner Magen, and Assaf Naor. Girth and euclidean distortion. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 705–711, 2002. 60

[LN94] Eugene V. Levner and A. S. Nemirovsky. A network flow algorithm for just-in-time project scheduling. *European Journal of Operational Research*, 79(2):167–175, December 1994. 8

[LNMG09] Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. Topic-link LDA: joint models of topic and author community. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 665–672, 2009. 5

[Lov09] László Lovász. Very large graphs. *CoRR*, math.CO/0902.0132, 2009. 4

[LPA$^+$09] David Lazer, Alex Pentland, Lada Adamic, Sinan Aral, Albert-László Barabási, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, Tony Jebara, Gary King, Michael Macy, Deb Roy, and Marshall Van Alstyne. Computational social science. *Science*, 323:721–723, 2009. 4

[LPS88] Alexander Lubotzky, R. Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. 65, 66

[LR82] Zachary F. Lansdowne and David W. Robinson. Geographic decomposition of the shortest path problem, with an application to the traffic assignment problem. *Management Science*, 28(12):1380–1390, 1982. 35, 51

[LS67] A. H. Land and S. W. Stairs. The extension of the cascade algorithm to large graphs. *Management Science*, 14(1):29–33, 1967. 35, 51

[LS93] Nathan Linial and Michael E. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993. Announced at SODA 1991. 43

[LS09] Silvio Lattanzi and D. Sivakumar. Affiliation networks. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 427–434, 2009. 24

[LT80] Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980. Announced at FOCS 1977. 30, 46, 121

[LT09] Ulf Leser and Silke Trißl. Graph management in the life sciences. In *Encyclopedia of Database Systems*, pages 1266–1271. 2009. 4

[LU93] Felix Lazebnik and Vasiliy A. Ustimenko. New examples of graphs without small cycles and of large size. *European Journal of Combinatorics*, 14(5):445–460, 1993. 28

[Lu02] Linyuan Lu. *Probabilistic methods in massive graphs and Internet computing*. PhD thesis, University of California San Diego, 2002. 82, 84, 85

[LUW95] Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. A new series of dense graphs of high girth. *Bulletin of the American Mathematical Society*, 32(1):73–79, 1995. 28

[LUW96] Felix Lazebnik, Vasiliy A. Ustimenko, and Andrew J. Woldar. A characterization of the components of the graphs $D(k, q)$. *Discrete Mathematics*, 157(1-3):271–283, 1996. 28

[MB98] Eric N. Mortensen and William A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384, 1998. 8, 10, 11

[MB08] Scott Morris and Kobus Barnard. Finding trails. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*, 2008. 11

[MC09] Julian John McAuley and Tibério S. Caetano. An expected-case sub-cubic solution to the all-pairs shortest path problem in $\mathbb{R}$. *CoRR*, abs/0912.0975, 2009. 35

[McD98] Colin McDiarmid. *Probabilistic Methods for Algorithmic Discrete Mathematics*, volume 16 of *Algorithms and Combinatorics*, chapter Concentration, pages 1–46. Springer, 1998. 86

[McG95] Catherine C. McGeoch. All-pairs shortest paths and the essential subgraph. *Algorithmica*, 13(5):426–441, 1995. 35

[MDMCM01] Fabien Mourgues, Frederic Devernay, Grégoire Malandain, and Ève Coste-Manière. $3d + t$ modeling of coronary artery tree from standard non simultaneous angiographic sequences. In *Proceedings of the 4th International Conference on on Medical Image Computing and Computer-Assisted Intervention*, pages 1320–1322, 2001. 8

[Meh88] Kurt Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Information Processing Letters*, 27(3):125–128, 1988. 97, 98

[MEJ$^+$09] Kamesh Madduri, David Ediger, Karl Jiang, David A. Bader, and Daniel G. Chavarría-Miranda. A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In *23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 23-29, 2009*, pages 1–8, 2009. 12

[Met07] Philipp Metzner. *Transition Path Theory for Markov Processes*. PhD thesis, Freie Universität Berlin, 2007. 11

[Mey03] Ulrich Meyer. Average-case complexity of single-source shortest-paths algorithms: lower and upper bounds. *Journal of Algorithms*, 48(1):91–134, 2003. Announced at SODA 2001. 35

[Mey09] Ulrich Meyer. Via detours to I/O-efficient shortest paths. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 219–232, 2009. 35

[MHSWZ04] Matthias Müller-Hannemann, Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis. Timetable information: Models and algorithms. In *Algorithmic Methods for Railway Optimization, International Dagstuhl Workshop, Dagstuhl Castle, Germany, June 20-25, 2004, 4th International Workshop, ATMOS 2004, Bergen, Norway, September 16-17, 2004, Revised Selected Papers*, pages 67–90, 2004. 3, 56

[Mil66] G. Mills. A decomposition algorithm for the shortest route problem. *Operations Research*, 14:279–286, 1966. 35, 51

[Mil67] Stanley Milgram. The small world problem. *Psychology Today*, 1:61–67, 1967. 6

[Mil94] Peter Bro Miltersen. Lower bounds for union-split-find related problems on random access machines. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 625–634, 1994. 60

[Mil99] Peter Bro Miltersen. Cell probe complexity — a survey. In *Advances in Data Structures Workshop, Proc. of 19th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 1999. 26, 60

[Min57] George James Minty. A comment on the shortest-route problem. *Operations Research*, 5(5):724, 1957. 8, 35

[Min58] George James Minty. *Operations Research*, 6:882–883, 1958. 8

[Mit97] Joseph S. B. Mitchell. chapter Shortest Paths and Networks, pages 445–466. CRC Press, 1997. 32, 39

[Mit03] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2), 2003. 4, 23, 24, 111

[MMP87] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. 32

[MN67] M. Mori and T. Nishimura. Solution of the routing problem through a network by a matrix method with auxiliary nodes. *Transportation Research*, 1(2):165–180, 1967. Announced in Memoirs of the Faculty of Engineering, Osaka City University, pp. 149–162, 1963. 8, 39

[MN06] Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 109–118, 2006. 43, 44, 79, 118

[MNSW98] Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998. Announced at STOC 1995. 60, 62, 72

[MO09] Ulrich Meyer and Vitaly Osipov. Design and implementation of a practical I/O-efficient shortest paths algorithm. In *Proceedings of the Workshop on Algorithm Engineering and Experiments, ALENEX 2009, New York, New York, USA, January 3, 2009*, pages 85–96, 2009. 35

[Moo59] Edward F. Moore. The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching, and Annals of the Computation Laboratory of Harvard University*, pages 285–292. Harvard University Press, 1959. Announced at the International Symposium on the Theory of Switching 1957. 8, 34, 51

[Mor93] Pieter Moree. Bertrands postulate for primes in arithmetical progressions. *Computers & Mathematics with Applications*, 26(5):35–43, 1993. 66

[Mor94] Moshe Morgenstern. Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q. *Journal of Combinatorial Theory, Series B*, 62(1):44–62, 1994. 65

[MP69] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, Cambridge, 1969. 64, 78

[MP97] Kurt Mehlhorn and Volker Priebe. On the all-pairs shortest-path algorithm of Moffat and Takaoka. *Random Struct. Algorithms*, 10(1-2):205–220, 1997. Announced at ESA 1995. 35

[MPN$^+$05] Lauren Ancel Meyers, Babak Pourbohloul, Mark E. J. Newman, Danuta M. Skowronski, and Robert C. Brunham. Network theory and SARS: predicting outbreak diversity. *Journal of Theoretical Biology*, 232(1):71–81, 2005. 7, 10

[MPS06] Milena Mihail, Christos H. Papadimitriou, and Amin Saberi. On certain connectivity properties of the internet topology. *Journal of Computer and System Sciences*, 72(2):239–251, 2006. Announced at FOCS 2003. 89

[MPSV02] Yamir Moreno, Romualdo Pastor-Satorras, and Alessandro Vespignani. Epidemic outbreaks in complex heterogeneous networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 26(4), 2002. 7

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995. 62

[MS03] Ulrich Meyer and Peter Sanders. δ-stepping: a parallelizable shortest path algorithm. *Journal of Algorithms*, 49(1):114–152, 2003. Announced at ESA 1998. 35

[MS08a] Jiri Matousek and Anastasios Sidiropoulos. Inapproximability for metric embeddings into $\mathbb{R}^d$. In *49th IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, 2008. 30

[MS08b] Manor Mendel and Chaya Schwob. C-K-R partitions of sparse graphs. *CoRR*, abs/0809.1902, 2008. 43, 44

[MSM09] Jens Maue, Peter Sanders, and Domagoj Matijevic. Goal-directed shortest-path queries using precomputed cluster distances. *ACM Journal of Experimental Algorithmics*, 14:3.2–3.27, 2009. 53

[MSS$^+$06] Rolf H. Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. Partitioning graphs to speedup Dijkstra's algorithm. *ACM Journal of Experimental Algorithmics*, 11, 2006. 54

[MT84] Alistair Moffat and Tadao Takaoka. A priority queue for the all pairs shortest path problem. *Information Processing Letters*, 18(4):189–193, 1984. 36

[MT87] Alistair Moffat and Tadao Takaoka. An all pairs shortest path algorithm with expected time $\mathcal{O}(n^2 \log n)$. *SIAM Journal on Computing*, 16(6):1023–1031, 1987. Announced at FOCS 1985. 35

[MT93] Sean P. Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Springer Verlag, 1993. 11

[Mur65] John David Murchland. A new method for finding all elementary paths in a complete directed graph. Technical Report LBS-TNT-22, London Business School, Transport Network Theory Unit, 1965. 51

[Mur67a] John David Murchland. The effect of increasing or decreasing the length of a single arc on all shortest distances in a graph. Technical Report LBS-TNT-26, London Business School, Transport Network Theory Unit, 1967. 33

[Mur67b] John David Murchland. The “once-through” method of finding all shortest distances in a graph from a single origin. Technical Report LBS-TNT-56, London Business School, Transport Network Theory Unit, 1967. 10, 35

[MV06] Oliver Mason and Mark Verwoerd. Graph theory and networks in biology. *arXiv:q-bio*, q-bio/0604006, 2006. 4, 6, 11, 12

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. Announced at STOC 1987. 19

[MWN09] Shay Mozes and Christian Wulff-Nilsen. Shortest paths in planar graphs with real lengths in $\mathcal{O}(n \log^2 n / \log \log n)$ time. *CoRR*, abs/0911.4963, 2009. 44

[MZ93] Nicolas Merlet and Josiane Zerubia. A curvature-dependent energy function for detecting lines in satellite images. In *The 8th Scandinavian Conference on Image Analysis, University of Tromso, Norway*, pages 25–28, 1993. 11

[MZ03] Ulrich Meyer and Norbert Zeh. I/O-efficient undirected shortest paths. In *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, pages 434–445, 2003. 35

[MZ06] Ulrich Meyer and Norbert Zeh. I/O-efficient undirected shortest paths with unbounded edge lengths. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 540–551, 2006. 35

[MZ07] Laurent Flindt Muller and Martin Zachariasen. Fast and compact oracles for approximate distances in planar graphs. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 657–668, 2007. 46, 47

[MZ08] Anil Maheshwari and Norbert Zeh. I/O-efficient planar separators. *SIAM Journal on Computing*, 38(3):767–801, 2008. Announced at SODA 2002. 35

[Nak72] Mario Nakamori. A note on the optimality of some all-shortest path algorithms. *Journal of the Operations Research Society of Japan*, 15:201–204, 1972. 35

[NBB$^+$08] Giacomo Nannicini, Philippe Baptiste, Gilles Barbier, Daniel Krob, and Leo Liberti. Fast paths in large-scale dynamic road networks. *Computational Optimization and Applications*, 2008. 47, 53, 55, 56

[New00] Mark E. J. Newman. Models of the small world. *Journal of Statistical Physics*, pages 819–841, 2000. 4

[New01] Mark E. J. Newman. Scientific collaboration networks. II. shortest paths, weighted networks, and centrality. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 64, 2001. 14

[New02] Mark E. J. Newman. Spread of epidemic disease on networks. *Physical Review E*, 66(1):016128, Jul 2002. 7

[New03] Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003. 4

[New04] Mark E. J. Newman. Analysis of weighted networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 70(5):056131, Nov 2004. 13

[New05] Mark E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46:323–351, 2005. 4, 23

[NG04] Mark E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 69(026113), 2004. 13

[Nic66] T. Alastair J. Nicholson. Finding the shortest route between two points in a network. *The Computer Journal*, 9(3):275–280, November 1966. 10, 35

[NMM78] Kohei Noshita, Etsuo Masuda, and Hajime Machida. On the expected behaviors of the Dijkstra's shortest path algorithm for complete graphs. *Information Processing Letters*, 7(5):237–243, 1978. 35

[Nos85] Kohei Noshita. A theorem on the expected complexity of Dijkstra's shortest path algorithm. *Journal of Algorithms*, 6(3):400–408, 1985. 35

[NR03] Ilan Newman and Yuri Rabinovich. A lower bound on the distortion of embedding planar metrics into Euclidean space. *Discrete and Computational Geometry*, 29:77–81, 2003. 30

[NR06] Ilkka Norros and Hannu Reittu. On a conditionally Poissonian graph process. *Advances in Applied Probability*, 38(1):59–75, 2006. 24

[NR08] Ilkka Norros and Hannu Reittu. On the robustness of power-law random graphs in the finite mean, infinite variance region. arXiv:0801.1079, 2008. 83

[NSW01] Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 64(2):026118 1–17, Jul 2001. 83

[NW99] Mark E. J. Newman and D. J. Watts. Scaling and percolation in the small-world network model. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 60(6):7332–7342, Dec 1999. 24, 83

[NWS02] Mark E. J. Newman, Duncan J. Watts, and Steven H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99:2566–2572, 2002. 83

[NZ02] T. S. Eugene Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, 2002. 15

[OR90] Ariel Orda and Raphael Rom. Shortest-path and minimum delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3):607–625, 1990. 32, 39

[Orl83] James B. Orlin. On the simplex algorithm for networks and generalized networks. Working papers 1467-83., Massachusetts Institute of Technology (MIT), Sloan School of Management, 1983. 34

[OSF$^+$08] Atsuyuki Okabe, Toshiaki Satoh, Takehiro Furuta, Atsuo Suzuki, and Kyoko Okano. Generalized network Voronoi diagrams: Concepts, computational methods, and applications. *International Journal of Geographical Information Science*, 22:965–994, 2008. 98

[OSH$^+$07] Jukka-Pekka Onnela, Jari Saramäki, Jörkki Hyvönen, Gábor Szabó, M. Argollo de Menezes, Kimmo Kaski, Albert-László Barabási, and János Kertész. Analysis of a large-scale weighted network of one-to-one human communication. *New Journal of Physics*, 9(6):179, 2007. 7

[Pap74] Uwe Pape. Implementation and efficiency of Moore-algorithms for the shortest route problem. *Mathematical Programming*, 7(1):212–222, 1974. 34, 35, 50, 51

[Pap80] Uwe Pape. Algorithm 562: Shortest path lengths [h]. *ACM Transactions on Mathematical Software*, 6(3):450–455, 1980. 34

[Par60] Arvind Chandulai Parikh. *Some Theorems and Algorithms for Finding Optimal Paths Over Graphs with Engineering Applications*. PhD thesis, Purdue University, 1960. 8

[Pat08a] Mihai Patrascu. (Data) STRUCTURES. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 434–443, 2008. 41, 59, 60, 62, 63, 64, 67, 69, 71, 72

[Pat08b] Mihai Patrascu. *Lower Bound Techniques for Data Structures*. PhD thesis, Massachusetts Institute of Technology, 2008. 60, 62, 64

[Pat08c] Mihai Patrascu. Randomized lower bounds for lopsided set disjointness. `http://people.csail.mit.edu/mip/papers/structures/lsd.pdf`, 2008. 62

[Pat09] Mihai Patrascu. Unifying the Landscape of Cell-Probe Lower Bounds . submitted, 2009. 62

[PBCG09] Michalis Potamias, Francesco Bonchi, Carlos Castillo, and Aristides Gionis. Fast shortest path distance estimation in large networks. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 867–876, 2009. 13, 53, 56, 83

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 5

[PC06] Gabriel Peyré and Laurent D. Cohen. Landmark-based geodesic computation for heuristically driven path planning. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2229–2236, 2006. 11

[Pel00] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33:167–176, 2000. See also: Proceedings of the 25th Workshop on Graph-Theoretic Concepts in Computer Science (WG), 1999. 29

[Pet03] Seth Pettie. *On the Shortest Path and Minimum Spanning Tree Problems*. PhD thesis, The University of Texas at Austin, 2003. 32, 34

[Pet04] Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science*, 312(1):47–74, 2004. 34

[Pet07] Seth Pettie. Low distortion spanners. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 78–89, 2007. 27, 28

[Pet08a] Seth Pettie. All pairs shortest paths in sparse graphs. In *Encyclopedia of Algorithms*. 2008. 35

[Pet08b] Seth Pettie. Single-source shortest paths. In *Encyclopedia of Algorithms*. 2008. 32, 34

[PK85] Richard C. Paige and Clyde P. Kruskal. Parallel algorithms for shortest path problems. In *ICPP*, pages 14–20, 1985. 35

[PMT03] Sarah V. Porter, Majid Mirmehdi, and Barry T. Thomas. A shortest path representation for video summarisation. In *12th International Conference on Image Analysis and Processing (ICIAP 2003), 17-19 September 2003, Mantova, Italy*, pages 460–465, 2003. 11

[Poh71] Ira Sheldon Pohl. Bi-directional search. *Machine Intelligence*, 6:127–140, 1971. 10, 35

[Pou08] Pawan Poudel. Computing point-to-point shortest path using an approximate distance oracle. Master's thesis, Miami University, 2008. 53

[PR02] Seth Pettie and Vijaya Ramachandran. Computing shortest paths with comparisons and additions. In *SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 267–276, 2002. 33, 34, 37, 104

[PRB60] Robert M. Peart, Paul H. Randolph, and T. E. Bartlett. The shortest-route problem (letters to the editor). *Operations Research*, 8(6):866–868, 1960. 8

[PRS02] Seth Pettie, Vijaya Ramachandran, and Srinath Sridhar. Experimental evaluation of a new shortest path algorithm. In *Algorithm Engineering and Experiments, 4th International Workshop, ALENEX 2002, San Francicsco, CA, USA, January 4-5, 2002, Revised Papers*, pages 126–142, 2002. 34, 35

[PS85] Franco P. Preparata and Michael Ian Shamos. *Computational geometry: an introduction*. 1985. 39

[PS89] David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. 27

[PS98] Stefano Pallottino and Maria Grazia Scutellà. *Equilibrium and Advanced Transportation Modelling*, chapter Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects, pages 245–281. 1998. 54

[PSV01] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86(14):3200–3203, Apr 2001. 7

[PSWZ07] Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos D. Zaroliagis. Efficient models for timetable information in public transportation systems. *ACM Journal of Experimental Algorithmics*, 12, 2007. 56

[PT06] Mihai Patrascu and Mikkel Thorup. Time-space trade-offs for predecessor search. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 232–240, 2006. 63, 69, 72

[PW60] Maurice Pollack and Walter Wiebenson. Solutions of the shortest-route problem — a review. *Operations Research*, 8(2):224–230, 1960. 8, 51

[PW99] Panos D. Prevedouros and Yuhao Wang. Simulation of large freeway and arterial network with CORSIM, INTEGRATION, and WATSim. *Transportation Research Record: Journal of the Transportation Research Board*, 1678:197–207, 1999. 10

[PZMT03] Dimitris Papadias, Jun Zhang, Nikos Mamoulis, and Yufei Tao. Query processing in spatial network databases. In *VLDB*, pages 802–813, 2003. 55

[RA59] Harold Rapaport and Paul Abramson. An analog computer for finding an optimum route through a communication network. *IRE Transactions on Communications Systems*, 7(1):37–42, May 1959. 8

[Ram96a] Ganesan Ramalingam. *Bounded Incremental Computation*. 1996. 39

[Ram96b] Rajeev Raman. Priority queues: Small, monotone and trans-dichotomous. In *Algorithms - ESA '96, Fourth Annual European Symposium, Barcelona, Spain, September 25-27, 1996, Proceedings*, pages 121–137, 1996. 34

[Ram97] Rajeev Raman. Recent results on the single-source shortest paths problem. *SIGACT News*, 28:81–87, 1997. 34

[Rao99] Satish Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. In *Symposium on Computational Geometry*, pages 300–306, 1999. 30

[Raz92] Alexander A. Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992. 62

[Rei58] I. Reiman. Über ein Problem von K. Zarankiewicz. *Acta Mathematica Academiae Scientiarum Hungaricae*, 9:269–279, 1958. 28

[RMJ06] Matthew J. Rattigan, Marc Maier, and David Jensen. Using structure indices for efficient approximation of network properties. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 357–366, 2006. 13, 56

[RMJ07] Matthew J. Rattigan, Marc Maier, and David Jensen. Graph clustering with network structure indices. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvalis, Oregon, USA, June 20-24, 2007*, pages 783–790, 2007. 13, 56, 83

[RN04a] Bryan Raney and Kai Nagel. Iterative route planning for large-scale modular transportation simulations. *Future Generation Computer Systems*, 20(7):1101–1118, 2004. 10

[RN04b] Hannu Reittu and Ilkka Norros. On the power-law random graph model of massive data networks. *Performance Evaluation*, 55(1-2):3–23, 2004. Announced at Internet performance symposium (IPS 2002). 24

[Rob56] John T. Robacker. Min-max theorems on shortest chains and disjoint cuts of a network. Research Memorandum RM-1660, The Rand Corporation, 1956. 8

[Rom80] Francesco Romani. Shortest-path problem is not harder than matrix multiplication. *Information Processing Letters*, 11(3):134–136, 1980. 36

[Rou86] Dennis H. Rouvray. Predicting chemistry from topology. *Scientific American*, 265:40–47, 1986. 12

[RS83] Neil Robertson and Paul D. Seymour. Graph minors. I. excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983. 20, 21

[RS86] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986. 20

[RS08] Liam Roditty and Asaf Shapira. All-pairs shortest paths with a sublinear additive error. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 622–633, 2008. 37

[RSM⁺02] Erzsébet Ravasz, Anna Lisa Somera, Dale A. Mongru, Zoltán N. Oltvai, and Albert-László Barabási. Hierarchical organization of modularity in metabolic networks. *Science*, 297(5586):1551–1555, 2002. 6

[RSR⁺01] Jean-Christophe Rain, Luc Selig, Hilde De Reuse, Veronique Battaglia, Celine Reverdy, Stephane Simon, Gerlinde Lenzen, Fabien Petel, Jerome Wojcik, Vincent Schächter, Y. Chemama, Agnes Labigne, and Pierre Legrain. The protein-protein interaction map of Helicobacter pylori. *Nature*, 409(6817):211–215, 2001. 6

[RTMS05] Martin Rosvall, Ala Trusina, Peter Minnhagen, and Kim Sneppen. Networks and cities: An information perspective. *Physical Review Letters*, 94(2):028701, Jan 2005. 10

[RTZ05] Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 261–272, 2005. 38, 42, 44

[RTZ08] Liam Roditty, Mikkel Thorup, and Uri Zwick. Roundtrip spanners and roundtrip routing in directed graphs. *ACM Transactions on Algorithms*, 4(3):1–17, 2008. 28

[RZ04] Liam Roditty and Uri Zwick. Dynamic approximate all-pairs shortest paths in undirected graphs. In *Proc. of Symp. on Foundations of Computer Science, Rome, Oct. 2004*, pages 499–508, 2004. 39

[Sab66] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966. 12

[Sam63] A. L. Samuel. Some studies in machine learning using the game of checkers. *Computers and Thought*, 1963. Originally in IBM Journal 3, 211-229 (1959). 35, 52

[San09] Peter Sanders. Algorithm engineering - an attempt at a definition. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, pages 321–340, 2009. 50

[SBA⁺95] LaRon Smith, Richard Beckman, Doug Anson, Kai Nagel, and Michael E Williams. TRANSIMS: Transportation analysis and simulation system. In *Fifth National Conference on Transportation Planning Methods Applications-Volume II: A Compendium of Papers Based on a Conference Held in Seattle, Washington in April 1995*, 1995. 10

[SBR⁺06] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, 34(1):535–539, 2006. 112

[SC81] Howard A. Smolleck and Mo-Shing Chen. A new approach to near-optimal path assignment through electric-circuit modeling. *Networks*, 11:335–349, 1981. 51

[Sch98] Jeanette P. Schmidt. 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, 1998. Announced at ISTCS 1995. 8, 46

[Sch03] Alexander Schrijver. *Combinatorial Optimization — Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003. 32, 35

[Sch05a] Alexander Schrijver. *Handbook of Discrete Optimization*, chapter On the history of combinatorial optimization (till 1960), pages 1–68. 2005. 32

[Sch05b] Frank Schulz. *Timetable Information and Shortest Paths*. PhD thesis, Universität Karlsruhe, 2005. 56

[Sch08a] Dominik Schultes. *Route Planning in Road Networks*. PhD thesis, Universität Karlsruhe, 2008. 54, 55

[Sch08b] Dominik Schultes. Routing in road networks with transit nodes. In *Encyclopedia of Algorithms*. 2008. 47, 51, 54, 55, 56

[SCK$^+$08] Ralf Schenkel, Tom Crecelius, Mouna Kacimi, Thomas Neumann, Josiane Xavier Parreira, Marc Spaniol, and Gerhard Weikum. Social wisdom for search and recommendation, June 2008. 13

[SdC77] Lenie Sint and Dennis de Champeaux. An improved bidirectional heuristic search algorithm. *Journal of the ACM*, 24(2):177–191, 1977. Announced at IJCAI 1975. 10, 35

[Sei95] Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of Computer and System Sciences*, 51(3):400–403, 1995. Announced at STOC 1992. 36

[Sei06] Raimund Seidel. Top-down analysis of path compression: Deriving the inverse-ackermann bound naturally (and easily). In *Algorithm Theory - SWAT 2006, 10th ScandinavianWorkshop on Algorithm Theory, Riga, Latvia, July 6-8, 2006, Proceedings*, page 1, 2006. 47

[Sen09] Sandeep Sen. Approximating shortest paths in graphs. In *WALCOM: Algorithms and Computation, Third International Workshop, WALCOM 2009, Kolkata, India, February 18-20, 2009. Proceedings*, pages 32–43, 2009. 32, 37, 42, 43, 44, 123

[Set96] James Albert Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. 11

[SFG97] Shashi Shekhar, Andrew Fetterer, and Bjajesh Goyal. Materialization trade-offs in hierarchical shortest path algorithms. In *SSD '97: Proceedings of the 5th International Symposium on Advances in Spatial Databases*, pages 94–111, London, UK, 1997. Springer-Verlag. 10, 51, 54

[SG67] Ralph E. Schofer and Franklin F. Goodyear. Electronic computer applications in urban transportation planning. In *Proceedings of the 1967 22nd national conference*, pages 247–253, 1967. 10

[SG05] Alfons Schnitzler and Joachim Gross. Normal and pathological oscillatory communication in the brain. *Nature Reviews Neuroscience*, 6:285–296, 2005. 6

[SGNP10] Atish Das Sarma, Sreenivas Gollapudi, Marc Najork, and Rina Panigrahy. A sketch-based distance oracle for web-scale graphs. In *International Conference on Web Search and Data Mining (WSDM)*, 2010. to appear. 56

[Sha54] Marvin E. Shaw. Group structure and the behavior of individuals in small groups. *Journal of Psychology: Interdisciplinary and Applied*, 38:139–149, 1954. 12

[Sha75] Michael Ian Shamos. Geometric complexity. In *Conference Record of Seventh Annual ACM Symposium on Theory of Computation (STOC'75), 5-7 May 1975, Albuquerque, New Mexico, USA*, pages 224–233, 1975. 98

[Sha97a] Mehul A. Shah. ReferralWeb: A resource location system guided by personal relations. Master's thesis, Massachusetts Institute of Technology, 1997. 14

[Sha97b] Micha Sharir. *Handbook of Discrete and Computational Geometry*, chapter Algorithmic Motion Planning, pages 733–754. CRC Press, 1997. 32, 39

[Shi53] Alfonso Shimbel. Structural parameters of communication networks. *Bulletin of Mathematical Biophysics*, 15:501–507, 1953. 8, 12

[Shi55] Alfonso Shimbel. Structure in communication nets. In *Proceedings of the Symposium on Information Networks (New York, 1954)*, pages 199–203, 1955. 8

[Shi00] Tetsuo Shibuya. Computing the $n \times m$ shortest paths efficiently. *ACM Journal of Experimental Algorithmics*, 5:9, 2000. Announced at ALENEX 1999. TR: IBM TRL Report RT5133 1997. 35

[Shi03] Clay Shirky. Power laws, weblogs, and inequality. "Networks, Economics, and Culture" mailing list, 2003. 5

[SHWH08] Christian Sommer, Michael E. Houle, Martin Wolff, and Shinichi Honiden. Approximate shortest path queries in graphs using Voronoi duals. Technical Report NII-2008-007E, National Institute of Informatics, August 2008. 16

[SKC93] Shashi Shekhar, Ashim Kohli, and Mark Coyle. Path computation algorithms for advanced traveller information system (ATIS). In *Proceedings of the Ninth International Conference on Data Engineering, April 19-23, 1993, Vienna, Austria*, pages 31–39, 1993. 54

[SL67] Ralph E. Schofer and Bernard M. Levin. The urban transportation planning process. *Socio-Economic Planning Sciences*, 1:185–197, 1967. 10

[SL97] Shashi Shekhar and Duen-Ren Liu. CCAM: A connectivity-clustered access method for networks and network computations. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):102–119, 1997. 54

[SL03] Manoj Pratim Samanta and Shuguang Liang. Predicting protein functions from redundancies in large-scale protein interaction networks. *Proceedings of the National Academy of Sciences*, 100(22):12579–12583, 2003. 6

[SLSP03] Thomas Seidenbecher, T. Rao Laxmi, Oliver Stork, and Hans-Christian Pape. Amygdalar and hippocampal theta rhythm synchronization during fear memory retrieval. *Science*, 301(5634):846–850, 2003. 6

[Smo75] Howard A. Smolleck. *Application of Fast Sparse-Matrix Techniques and an Energy Estimation Model for Large Transportation Networks*. PhD thesis, University of Texas at Arlington, 1975. 35, 51

[SMR97] Carsten Steger, Helmut Mayer, and Bernd Radig. The role of grouping for road extraction. In *Automatic Extraction of Man-Made Objects from Aerial and Space Images (II)*, pages 245–256, 1997. 11

[SMS+02] Lukasz Salwinski, Christopher S. Miller, Adam J. Smith, Frank K. Pettit, James U. Bowie, and David Eisenberg. DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Research*, 30(1):303–305, 2002. 112

[SNGM09] Antonio Sedeño-Noda and Carlos González-Martín. New efficient shortest path simplex algorithm: pseudo permanent labels instead of permanent labels. *Computational Optimization and Applications*, 43(3):437–448, 2009. 34

[Spi73] Philip M. Spira. A new algorithm for finding all shortest paths in a graph of positive arcs in average time $\mathcal{O}(n^2 \log^2 n)$. *SIAM Journal on Computing*, 2(1):28–32, 1973. 35

[Spr07] Alan P. Sprague. $\mathcal{O}(1)$ query time algorithm for all pairs shortest distances on permutation graphs. *Discrete Applied Mathematics*, 155(3):365–373, 2007. 48, 121

[SPZ06] Gabriele Di Stefano, Alberto Petricola, and Christos D. Zaroliagis. On the implementation of parallel shortest path algorithms on a supercomputer. In *Parallel and Distributed Processing and Applications, 4th International Symposium, ISPA 2006, Sorrento, Italy, December 4-6, 2006, Proceedings*, pages 406–417, 2006. 35

[SR90] Wojciech Szpankowski and Vernon Rego. Yet another application of a binomial recurrence. Order statistics. *Computing*, 43(4):401–410, 1990. 108

[SR08] Parag Singla and Matthew Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 655–664, 2008. 13

[SS80] Mischa Schwartz and Thomas E. Stern. Routing techniques used in computer communication networks. *IEEE Transactions on Communications*, 28(4):539–552, Apr 1980. 8, 14

[SS99] Hanmao Shi and Thomas H. Spencer. Time—work tradeoffs of the single-source shortest paths problem. *Journal of Algorithms*, 30(1):19–32, 1999. 35

[SS05] Peter Sanders and Dominik Schultes. Highway hierarchies hasten exact shortest path queries. In *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, pages 568–579, 2005. 47, 53, 55, 56

[SS06] Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 804–816, 2006. 47, 51, 53, 54, 55, 56, 110, 114

[SS07a] Peter Sanders and Dominik Schultes. Engineering fast route planning algorithms. In *Experimental Algorithms, 6th International Workshop (WEA'07), Rome, Italy, June 6-8, 2007, Proceedings*, pages 23–36, 2007. 110, 111, 112, 114, 123

[SS07b] Dominik Schultes and Peter Sanders. Dynamic highway-node routing. In *Experimental Algorithms, 6th International Workshop (WEA'07), Rome, Italy, June 6-8, 2007, Proceedings*, pages 66–79, 2007. 55, 110, 114

[SS09] Jagan Sankaranarayanan and Hanan Samet. Distance oracles for spatial networks. In *Proceedings of the 25th International Conference on Data Engineering, ICDE*

2009, March 29 2009 - April 2 2009, Shanghai, China, pages 652–663, 2009. 50, 54, 55, 121

[SSA08] Hanan Samet, Jagan Sankaranarayanan, and Houman Alborzi. Scalable network distance browsing in spatial databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 43–54, 2008. 55

[SSA09] Jagan Sankaranarayanan, Hanan Samet, and Houman Alborzi. Path oracles for spatial networks. *Proceedings of the VLDB Endowment*, 2(1):1210–1221, 2009. 50, 55, 121

[ST99] Alan P. Sprague and Tadao Takaoka. $\mathcal{O}(1)$ query time algorithm for all pairs shortest distances on interval graphs. *International Journal of Foundations of Computer Science*, 10(4):465–472, 1999. 48

[Sto99] Bryan Stout. Smart move: Intelligent path-finding. Online at gamasutra.com/view/feature/3317/smart_move_intelligent_.php, 1999. 8

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969. 36

[Str01] Steven H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2001. 4

[Sug09] Kokichi Sugihara. Voronoi diagrams in facility location. In *Encyclopedia of Optimization, Second Edition*, pages 4040–4045. 2009. 98

[SV86] Robert Sedgewick and Jeffrey Scott Vitter. Shortest paths in Euclidean graphs. *Algorithmica*, 1(1):31–48, 1986. Announced at FOCS 1984. 35, 52

[Svi08] Zoya Svitkina. Lower-bounded facility location. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'08), San Francisco, California, USA, January 20-22, 2008*, pages 1154–1163, 2008. 98

[SVY09] Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 703–712, 2009. 15

[SWN92] Jacob Shapiro, Jerry Waxman, and Danny Nir. Level graphs and approximate shortest path algorithms. *Networks*, 22:691–717, 1992. 51, 54

[SZ99] Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 605–615, 1999. 36

[Tak92] Tadao Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Information Processing Letters*, 43(4):195–199, 1992. Announced at WG 1991. 36, 37

[Tak05] Tadao Takaoka. An $\mathcal{O}(n^3 \log\log n/\log n)$ time algorithm for the all-pairs shortest path problem. *Information Processing Letters*, 96(5):155–161, 2005. 36, 37

[Tak08] Tadao Takaoka. All pairs shortest paths via matrix multiplication. In *Encyclopedia of Algorithms*. 2008. 36

[Tal04] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290, 2004. 49, 121

[TE09] Sunil Thulasidasan and Stephan Eidenbenz. Accelerating traffic microsimulations: A parallel discrete-event queue-based approach for speed and scale. In *The Winter Simulation Conference*, 2009. 10

[TF97] Sabine Timpf and Andrew U. Frank. Using hierarchical spatial data structures for hierarchical spatial reasoning. In *Spatial Information Theory: A Theoretical Basis for GIS, International Conference COSIT '97, Laurel Highlands, Pennsylvania, USA, October 15-18, 1997, Proceedings*, pages 69–83, 1997. 51

[TGJ$^+$02] Hongsuda Tangmunarunkit, Ramesh Govindan, Sugih Jamin, Scott Shenker, and Walter Willinger. Network topology generators: degree-based vs. structural. In *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 19-23, 2002, Pittsburgh, PA, USA*, pages 147–159, 2002. 24

[Tho99] Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46(3):362–394, 1999. Announced at FOCS 1997. 33, 34, 104, 123

[Tho00a] Mikkel Thorup. Floats, integers, and single source shortest paths. *Journal of Algorithms*, 35(2):189–201, 2000. Announced at STACS 1998. 33, 104

[Tho00b] Mikkel Thorup. On RAM priority queues. *SIAM Journal of Computing*, 30(1):86–109, 2000. Announced at SODA 1996. 34, 103

[Tho04a] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004. Announced at FOCS 2001. 31, 46, 47, 83, 121

[Tho04b] Mikkel Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, 69(3):330–353, 2004. Announced at STOC 2003. 34, 103

[Tho07] Mikkel Thorup. Equivalence between priority queues and sorting. *Journal of the ACM*, 54(6), 2007. Announced at FOCS 2002. 33, 103

[Tit59] Jacques Tits. Sur la trialité et certains groupes qui sen déduisent. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 2(1), 1959. 28

[TKE$^+$09] Sunil Thulasidasan, Shiva Kasiviswanathan, Stephan Eidenbenz, Emanuele Galli, Susan Mniszewski, and Philip Romero. Designing systems for large-scale, discrete-event simulations: Experiences with the FastTrans parallel microsimulator. In *HiPC - International Conference on High Performance Computing*, 2009. 10

[TL07] Silke Trißl and Ulf Leser. Fast and practical indexing and querying of very large graphs. In *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 845–856, 2007. 54, 57

[TM69] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32:425–443, 1969. 6

[TM80] Tadao Takaoka and Alistair Moffat. An $\mathcal{O}(n^2 \log\log\log n)$ expected time algorithm for the all shortest distance problem. In *Mathematical Foundations of Computer Science 1980 (MFCS'80), Proceedings of the 9th Symposium, Rydzyna, Poland, September 1-5, 1980*, pages 643–655, 1980. 35

[TM92] Edouard Thiel and Annick Montanvert. Chamfer masks: discrete distance functions, geometrical properties and optimization. In *11th IAPR International Conference on Pattern Recognition, 1992. Vol.III. Conference C: Image, Speech and Signal Analysis, Proceedings.*, pages 244–247, Aug-3 Sep 1992. 11

[Tob70] Waldo R. Tobler. A computer movie simulating urban growth in the Detroit region. *Economic Geography*, 46:234–240, 1970. 7

[Tsi95] John N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, Sep 1995. 11

[Tur37] Alan Mathison Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(1):230–265, 1937. 26

[TZ00] Jesper Larsson Träff and Christos D. Zaroliagis. A simple parallel algorithm for the single-source shortest path problem on planar digraphs. *Journal of Parallel and Distributed Computing*, 60(9):1103–1124, 2000. Announced at IRREGULAR 1996. 35

[TZ01] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *ACM Symposium on Parallelism in Algorithms and Architectures*, pages 1–10, 2001. 49, 81, 82, 95

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005. Announced at STOC 2001. 28, 40, 41, 42, 43, 44, 53, 56, 59, 78, 81, 82, 83, 88, 89, 95, 118, 121, 123, 125

[TZ06] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 802–809, 2006. 27, 28, 29

[UCDG08] Antti Ukkonen, Carlos Castillo, Debora Donato, and Aristides Gionis. Searching the wikipedia with contextual information. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 1351–1352, 2008. 13

[UGC$^+$00] Peter Uetz, Loic Giot, Gerard Cagney, Traci A. Mansfield, Richard S. Judson, James R. Knight, Daniel Lockshon, Vaibhav Narayan, Maithreyan Srinivasan, Pascale Pochart, Alia Qureshi-Emili, Ying Li, Brian Godwin, Diana Conover, Theodore Kalbfleisch, Govindan Vijayadamodar, Meijia Yang, Mark Johnston, Stanley Fields, and Jonathan M. Rothberg. A comprehensive analysis of protein–protein interactions in Saccharomyces cerevisiae. *Nature*, 403:623–627, 2000. 6

[UY90] Jeffrey D. Ullman and Mihalis Yannakakis. The input/output complexity of transitive closure. *ACM SIGMOD Record*, 19(2):44–53, 1990. 57

[vEB90] Peter van Emde Boas. *Handbook of theoretical computer science (vol. A): algorithms and complexity*, chapter Machine models and simulations, pages 1–66. 1990. 26

[vEBKZ77] Peter van Emde Boas, Rob Kaas, and E. Zijlstra. Design and implementation of an efficient priority queue. *Mathematical Systems Theory*, 10:99–127, 1977. 34

[Ves09] Alessandro Vespignani. Predicting the behavior of techno-social systems. *Science*, 325:425–428, 2009. 7, 10

[VFD$^+$07] Monique V. Vieira, Bruno M. Fonseca, Rodrigo Damazio, Paulo B. Golgher, Davi de Castro Reis, and Berthier Ribeiro-Neto. Efficient search ranking in social networks. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 563–572, 2007. 13

[Vli78] Dirck Van Vliet. Improved shortest path algorithms for transport networks. *Transportation Research*, 12(1):7–20, 1978. 51, 52, 55, 125

[VLL00] Bert Vogelstein, David Lane, and Arnold J. Levine. Surfing the p53 network. *Nature*, 408(6810):307–310, 2000. 6, 12

[Vor07] Georgy Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, 133:97–178, 1907. 98

[VPSV02] Alexei Vázquez, Romualdo Pastor-Satorras, and Alessandro Vespignani. Large-scale topological and dynamical properties of the internet. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 65(6):066130, Jun 2002. 5

[VS94] Jeffrey Scott Vitter and Elizabeth A. M. Shriver. Algorithms for parallel memory I: Two-level memories. *Algorithmica*, 12(2/3):110–147, 1994. 26, 43, 45

[VS09] Vishal Verma and Jack Snoeyink. Reducing the memory required to find a geodesic shortest path on a large mesh. In *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*, pages 227–235, 2009. 32

[Wag76] Robert A. Wagner. A shortest path algorithm for edge-sparse graphs. *Journal of the ACM*, 23(1):50–57, 1976. 33, 35

[War62] Stephen Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962. 8, 35, 37

[Wen91] Rephael Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *Journal of Combinatorial Theory, Series B*, 52(1):113–116, 1991. 28

[WH60] P. D. Whiting and J. A. Hillier. A method for finding the shortest route through a road network. *Journal of the Operational Research Society*, 11(1/2):37–40, 1960. 8, 32

[Whi69] Leon S. White. Shortest route models for the allocation of inspection effort on a production line. *Management Science*, 15(5):249–259, 1969. 8

[WHY$^+$06] Haixun Wang, Hao He, Jun Yang, Philip S. Yu, and Jeffrey Xu Yu. Dual labeling: Answering graph reachability queries in constant time. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 75, 2006. 54

[Wie73] Christian Wiener. Ueber eine Aufgabe aus der Geometria situs. *Mathematische Annalen*, 6(1):29–30, 1873. Dated December 1871. 39

[Wie47] Harry Wiener. Structural determination of paraffin boiling points. *Journal of the American Chemical Society*, 69(1):17–20, 1947. 12

[Wil64] J. W. J. Williams. Algorithm 232: Heapsort. *Communications of the ACM*, 7:347–348, 1964. 33, 34

[Win83] Peter M. Winkler. Proof of the squashed cube conjecture. *Combinatorica*, 3(1):135–139, 1983. 30

[WM03] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *ACM SIGKDD Explorations Newsletter*, 5(1):59–68, 2003. 4

[Wol08] Martin Joachim Wolff. Finding important edges for routing in networks. Master's thesis, Universität Karlsruhe, 2008. 98

[Woo06] David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 389–398, 2006. 28, 29

[Wri75] J. W. Wright. Reallocation of housing by use of network analysis. *Operational Research Quarterly*, 26(2):253–258, 1975. 8

[WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, pages 440–442, 1998. 4, 7

[WS03] Stefan Wuchty and Peter F. Stadler. Centers of complex networks. *Journal of Theoretical Biology*, 223(1):45 – 53, 2003. 12

[WU93] Andrew J. Woldar and Vasiliy A. Ustimenko. An application of group theory to extremal graph theory. In *Group Theory: Proceedings of the Biennial Ohio State-Denison Conference*, pages 293–298, 1993. 28

[WW03] Dorothea Wagner and Thomas Willhalm. Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings*, pages 776–787, 2003. 54

[WW05] Dorothea Wagner and Thomas Willhalm. Drawing graphs to speed up shortest-path computations. In *Proceedings of the Seventh Workshop on Algorithm Engineering and Experiments and the Second Workshop on Analytic Algorithmics and Combinatorics, ALENEX /ANALCO 2005, Vancouver, BC, Canada, 22 January 2005*, pages 17–25, 2005. 53

[WWZ05] Dorothea Wagner, Thomas Willhalm, and Christos D. Zaroliagis. Geometric containers for efficient shortest-path computation. *ACM Journal of Experimental Algorithmics*, 10, 2005. 54

[XNR10] Rongjing Xiang, Jennifer Neville, and Monica Rogati. Modeling relationship strength in online social networks. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010*, 2010. to appear. 2

[XWP$^+$09] Yanghua Xiao, Wentao Wu, Jian Pei, Wei Wang, and Zhenying He. Efficiently indexing shortest paths by exploiting symmetry in graphs. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 493–504, 2009. 57, 83

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 209–213, 1979. 60, 61

[Yao81] Andrew Chi-Chih Yao. Should tables be sorted? *Journal of the ACM*, 28(3):615–628, 1981. 26, 60

[Yao90] Andrew Chi-Chih Yao. Coherent functions and program checkers (extended abstract). In *Proceedings of the Twenty Second Annual ACM Symposium on Theory* *of Computing, 14-16 May 1990, Baltimore, Maryland, USA*, pages 84–94, 1990. 43

[YAR77] Andrew Chi-Chih Yao, David Avis, and Ronald L. Rivest. An $\Omega(n^2 \log n)$ lower bound to the shortest paths problem. In *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing, 2-4 May 1977, Boulder, Colorado, USA*, pages 11–17, 1977. 35

[YBLS08] Sihem Amer Yahia, Michael Benedikt, Laks V. S. Lakshmanan, and Julia Stoyanovich. Efficient network aware search in collaborative tagging sites. *Proceedings of the VLDB Endowment*, 1(1):710–721, 2008. 13

[Yen71] Jin Y. Yen. On Hu's decomposition algorithm for shortest paths in a network. *Operations Research*, 19(4):983–985, 1971. 35, 51

[Yen72] Jin Y. Yen. Finding the lengths of all shortest paths in n-node nonnegative-distance complete networks using $\frac{1}{2}n^3$ additions and n^3 comparisons. *Journal of the ACM*, 19(3):423–424, 1972. 36

[Yen73] Jin Y. Yen. Reply to Williams and White's note. *Journal of the ACM*, 20(3):390, 1973. 36

[Yuv76] Gideon Yuval. An algorithm for finding all shortest paths using $n^{2.81}$ infinite-precision multiplications. *Information Processing Letters*, 4(6):155–156, 1976. 36

[YWD08] Ruiyun Yu, Xingwei Wang, and Sajal K. Das. A Voronoi diagram approach for mobile element scheduling in sparse sensor networks. In *FGCN '08: Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking*, pages 62–67, 2008. 97

[YZ05] Raphael Yuster and Uri Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 389–396, 2005. 39

[Zah71] Charles T. Zahn. Graph-theoretical methods for detecting and describing Gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86, 1971. 11

[Zar08] Christos Zaroliagis. Engineering algorithms for large network applications. In *Encyclopedia of Algorithms*. 2008. 10, 55

[ZC01] J. Leon Zhao and Hsing Kenneth Cheng. Graph indexing for spatial data traversal in road map databases. *Computers & OR*, 28(3):223–241, 2001. 54

[ZKM97] Athanasios K. Ziliaskopoulos, Dimitri Kotzinos, and Hani S. Mahmassani. 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, 1997. 10

[ZN98] F. Benjamin Zhan and Charles E. Noon. Shortest path algorithms: An evaluation using real road networks. *Transportation Science*, 32(1):65–73, 1998. 35

[ZN00] F. Benjamin Zhan and Charles E. Noon. A comparison between label-setting and label-correcting algorithms for computing one-to-one shortest paths. *Journal of Geographic Information and Decision Analysis*, 4(2):1–11, 2000. 35

[Zwi98] Uri Zwick. All pairs shortest paths in weighted directed graphs $\frac{3}{4}$ exact and almost exact algorithms. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 310–319, 1998. 36

[Zwi01] Uri Zwick. Exact and approximate distances in graphs — a survey. In *Algorithms - ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, pages 33–48, 2001. 32

[Zwi02] Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3):289–317, 2002. 36, 37

[Zwi06] Uri Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. *Algorithmica*, 46(2):181–192, 2006. Announced at ISAAC 2004. 36, 37

[ZZ94] J. Leon Zhao and Ahmed Zaki. Spatial data traversal in road map databases: A graph indexing approach. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, November 29 - December 2, 1994*, pages 355–362, 1994. 54