

Dynamics of spectral algorithms for distributed routing

by

Petar Maymounkov

B.A. Mathematics, Harvard University (2001)
M.Sc. Computer Science, New York University (2004)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

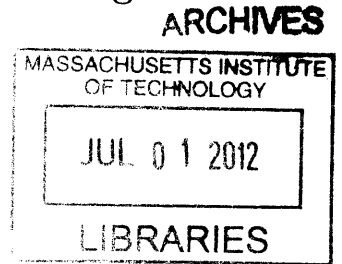
Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

©2012 Massachusetts Institute of Technology. All rights reserved.



DL

Author
Department of Electrical Engineering and Computer Science
March 2, 2012

Certified by
Jonathan Kelner
Assistant Professor of Applied Mathematics
Thesis Supervisor

A

Accepted by
Leslie A. Kolodziejcki
Professor of Electrical Engineering
Chairman, Committee for Graduate Students

Dynamics of spectral algorithms for distributed routing

by

Petar Maymounkov

Submitted to the Department of Electrical Engineering and Computer Science
on March 2, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

In the past few decades distributed systems have evolved from man-made machines to organically changing social, economic and protein networks. This transition has been overwhelming in many ways at once. Dynamic, heterogeneous, irregular topologies have taken the place of static, homogeneous, regular ones. Asynchronous, ad hoc peer-to-peer networks have replaced carefully engineered super-computers, governed by globally synchronized clocks. Modern network scales have demanded distributed data structures in place of traditionally centralized ones.

While the core problems of routing remain mostly unchanged, the sweeping changes of the computing environment invoke an altogether new science of algorithmic and analytic techniques. It is these techniques that are the focus of the present work.

We address the re-design of routing algorithms in three classical domains: multi-commodity routing, broadcast routing and all-pairs route representation. Beyond their practical value, our results make pleasing contributions to Mathematics and Theoretical Computer Science. We exploit surprising connections to NP-hard approximation, and we introduce new techniques in metric embeddings and spectral graph theory.

The distributed computability of “oblivious routes”, a core combinatorial property of every graph and a key ingredient in route engineering, opens interesting questions in the natural and experimental sciences as well. Oblivious routes are “universal” communication pathways in networks which are essentially unique. They are magically robust as their quality degrades smoothly and gracefully with changes in topology or blemishes in the computational processes. While we have only recently learned how to find them algorithmically, their power begs the question whether naturally occurring networks from Biology to Sociology to Economics have their own mechanisms of finding and utilizing these pathways.

Our discoveries constitute a significant progress towards the design of a self-organizing Internet, whose infrastructure is fueled entirely by its participants on an equal citizen basis. This grand engineering challenge is believed to be a potential technological solution to a long line of pressing social and human rights issues in the digital age. Some prominent examples include non-censorship, fair bandwidth allocation, privacy and ownership of social data, the right to copy information, non-discrimination based on identity, and many others.

Thesis Supervisor: Jonathan Kelner

Title: Assistant Professor of Applied Mathematics

Acknowledgments

I was interested in Mathematics at an early age in a kind of manner unconditional on any goal. My interests however took shape and direction when I met teacher Michael Mitzenmacher in college, whose engaging teaching permanently turned me into a Computer Scientist. Soon after I studied at the Courant Institute under the supervision of David Mazières. I continue to be indebted to David for his addictive enthusiasm and energy. The research we did together formed my impressions of what academic work should be like: an exciting social process of collaboration and passion.

To follow I was dropped in the whirlpool of fast-paced science at MIT, where many people were instrumental to my path. I want to thank Frans Kaashoek for providing me with unconditional support and freedom to pursue my ideas.

Perhaps by chance, but probably not just, I have always lived a double-life between the “systems” and “theory” communities. I am very thankful to my systems colleagues – Russ Cox, Michael Freedman, Bryan Ford, Thomer Gil, Szymon Jakubszak, Frans Kaashoek, Maxwell Krohn, Chris Lesniewski-Laas, Robert Morris, Jeremy Stribling, Michael Walfish – for being great company, sporting furiously good debates, being patient with my theory and teaching me the ins and outs of practical science. In my theory life I extend my gratitude to all the remarkable teachers — Richard Cole, David Karger, Piotr Indyk, Michel Goemans, Igor Pak, Joel Spencer, Madhu Sudan, and more — who taught me modern thinking in brilliantly different ways and whose unique personalities have shown me how science is embodied in people.

I especially thank my adviser at MIT, Jonathan Kelner. Jon encouraged me to take on entirely new areas of Mathematics and introduced me to all things spectral. He has been greatly accommodating and helpful, never starved of insightful ideas.

I thank my co-authors – Haim Avron, Keren Censor-Hillel, Matthew Brand, Bernhard Haeupler, Nick Harvey, Desmond Lun, Andreas Molisch, Sivan Toledo – for the exciting times we have shared in making our discoveries and for patiently introducing me to their respective fields of expertise. It has been an honor working with all of you. My work on *Kademlia*, while not my most sophisticated piece, has had the biggest practical impact and has brought me some flattering fame. I would like to thank Jed McCaleb for taking a large part in bringing this theoretical work to practice and the larger audiences.

Throughout my years at MIT I have had numerous stimulating exchanges with the wonderful folk that shared their time with me at MIT. They are Alexandr Andoni, Arnab Bhattacharyya, Elena Grigorescu, Krzysztof Onak, Kevin Matulef and many others. Academics aside, life in Cambridge has been stimulating because I met Oren Weimann and Benjamin Rossman, Grace Loo and Alexis Schulman, Szymon Jakubszak and Katie Lazarowicz, Trisha Shrum and Tara Grillos. I am grateful to my friends for decades, Tasho Statev-Kaletha and Chris Coyne.

Above all, I need to thank my family and relatives for their never-ending support. Despite the ocean that separates us, my parents Boris Maymunkov and Ivet Baeva-Maymunkova have always found ways to be right beside me.

Contents

1	Spectral routing	15
1.1	Routing theory	19
1.2	Demands, flows and routing	19
1.3	Oblivious network design	20
1.3.1	Specializations	21
1.3.2	Lower bounds	22
1.3.3	Bottleneck approximating decompositions	22
1.3.4	Tree-based routing	25
1.4	Linear routing schemes	28
1.4.1	The routing operator	28
1.4.2	Geometry of congestion	29
1.4.3	Routing as a norm minimization problem	31
1.5	Electric routing	33
1.5.1	Electric ℓ_∞ oblivious routing	34
1.5.2	Electric flow, effective resistance and the graph Laplacian	34
1.5.3	Representation	36
1.5.4	Computation	37
1.5.5	Robustness and latency	38
1.5.6	Analysis	39
1.5.7	Worst-case Demands Theorem	41
1.5.8	Unconditional Performance Bound (Proof of Theorem 1.5.2)	43
1.5.9	Laplacian ℓ_1 operator inequalities	44
1.5.10	Latency (Proof of Theorem 1.5.6)	47
1.5.11	Electric walk	48
1.5.12	Electric routing with approximate potentials	49
1.5.13	Robustness (Proof of Theorems 1.5.4 and 1.5.5)	54
1.5.14	Power iterations (Proof of Theorem 1.5.3)	55
1.5.15	Symmetrization of the electric flow algorithm	56
1.5.16	Open questions	57

2	Greedy embeddings	59
2.1	Introduction	61
2.1.1	History of the problem	62
2.1.2	Our results	63
2.1.3	Dimension reduction does not help	63
2.1.4	Road map	64
2.2	Topological embeddings	66
2.2.1	Ordinal embeddings	66
2.2.2	Greedy embeddings vs. monotone maps	66
2.2.3	Greedy embeddings vs. sphericity	67
2.3	Hyperbolic geometry	68
2.3.1	The half-plane model	68
2.3.2	The Klein model	69
2.3.3	Bisecting hyperplanes	70
2.4	Embeddings and tree decompositions	71
2.4.1	Greedy embeddings basics	71
2.4.2	Distance-preserving embeddings	71
2.4.3	Tree decomposition and heavy paths	71
2.5	Lower bounds	73
2.5.1	Graphs with hard crossroads	73
2.5.2	Dimension theorem	73
2.5.3	Complete proof	74
2.5.4	Connection between Euclidean and hyperbolic space bounds	77
2.6	Concise Hyperbolic Embeddings of Trees	78
2.6.1	Construction	78
2.6.2	Correctness argument	79
2.6.3	Canonization	79
2.6.4	Description complexity	80
2.6.5	Remarks	80
2.7	Low dimensional Euclidean embeddings of trees	82
2.8	Open Problems	83
3	Bottleneck-independent computation	85
3.1	Introduction and results	87
3.1.1	Local models of computation	87
3.1.2	Information spreading and model reductions	88
3.1.3	Techniques	88
3.1.4	Related Work	90
3.2	Gossip and conductance	92
3.2.1	The uniform gossip algorithm	92

3.2.2	Conductance	93
3.3	Neighbor Exchange in $O(\log^3 n)$ rounds	95
3.3.1	Conductance decomposition of a graph	95
3.3.2	The Superstep algorithm for the Neighbor Exchange Problem	97
3.4	Neighbor Exchange in hereditary sparse graphs	100
3.5	Simulators and graph spanners	103
3.6	Open problems	107
3.6.1	Simplified Neighbor Exchange	107
3.6.2	Asynchronous algorithm and soft decisions	108

List of Figures

2-1	Shown: (a) an non-rooted tree, (b) its heavy-path decomposition, and (c) the hierarchical path relationship.	72
2-2	Illustrated is the g_f embedding of P_3 from Figure 2-1. On the left is a view from the z -axis looking down towards the origin. On the right is a planar section defined by $g_f(f), g_f(p)$ and the origin.	79
2-3	An illustration of the canonical embedding of a cycle.	81
3-1	Code for Superstep algorithm. It is easily verified that the above algorithm can be implemented in the GOSSIP model of communication.	98
3-2	Code for DirectExchange algorithm.	101
3-3	An example illustrating the behavior of the algorithm choosing a random neighbor whose information is still unknown.	107

List of Tables

Chapter 1

Spectral routing

Our investigation of Spectral Routing arose as an attempt to find a practical solution to a long-standing problem in systems and networking: distributed routing with congestion guarantees. With the advent of real-time, continuously-changing modern networks, with hard-to-predict topologies, this problem has grown in difficulty and application. In this chapter, we describe how recent developments in Spectral Graph Theory have allowed us to devise realistic algorithms for the routing domain. Beyond being of practical value, Spectral Routing stands on its own with the pleasing techniques and connections that it contributes to Theoretical Computer Science and Mathematics.

Routing networks are quite diverse. They range from multicore systems, whose buses form static typically smaller synchronous networks with “nice” and known topologies, to heterogeneous asynchronous real-time peer-to-peer systems with dynamically changing complex topologies. Some aspects of routing are, nevertheless, common to most settings. Connectivity and capacity, at least in a given time instance, are modeled as an edge-weighted graph. The concurrency aspect of communication demands is usually modeled as a multicommodity-type problem over that graph. Whereas, the differences between routing algorithms stem from considerations of computational model, representation, objective and temporal properties of the connectivity graph and the demands.

To distinguish the application area of the present work in the landscape of the larger Routing Theory, we are best suited by a metaphorical division of routing systems into “early” and “modern” networks. The early distributed systems were mostly motivated by multi-core processors and super-computers, where the network topology is synthetic and typically engineered to be “regular” and mathematically convenient. Furthermore, such networks benefit from the convenience of synchronized time. In contrast, modern ad-hoc peer-to-peer systems like social and sensor networks, are generally seen as asynchronous distributed systems, with a potentially changing connectivity graph and communication demands.

The topologies of early networks, for instance those found in multi-core processors, were designed so as to ensure high throughput and easy routing algorithms. One of the most studied

topologies, for example, is the hypercube, which has been the starting point for most modern research on routing [106]. Naturally, other types of constructible expanding graphs have been considered as well [107], since in this setting high connectivity with fewer wires is the main design objective. On the other hand, in modern systems the network topology is typically not designed but rather dictated by external circumstances, which have inspired various topology models and related algorithms. In physical sensor networks, for example, it is customary to assume planarity or other geometrically-inspired graph properties [83], while in social networks it is acceptable to assume at least expansion as a fair mathematical interpretation of Stanley Milgram’s “six degrees of separation phenomenon” [65]. These are just a couple of nice examples, however. Quickly the literature has uncovered a proliferation of network models which are simple and complex, and mostly incomparable. Some such models are scale-free graphs for social and protein networks, disc graphs for cellular networks, etc. The growing variety of network topologies has suggested (for more than a decade now) an obvious general question: Can we architect routing algorithms that work on all topologies? This is one of the questions addressed in the present work.

Another differentiating aspect of modern networks is dynamicity. Graph topologies change over time. For the easier problem of routing for information dissemination (one-to-all communication), some routing works model dynamicity as an adversary [51] and overcome deliberate graph churn via network coding techniques. We are interested in the more subtle multicommodity routing problem. We address dynamicity in two ways. First, our routing algorithms converge quickly, so as to ensure useful routing tables before the graph has had a chance to change. Second, the quality of our routing schemes degrades gradually with the amount of graph change.

In addition to topology changes, another source of variability are the demands. In both early and modern networks, more often than not, one needs to accommodate asynchronously changing demands in real time. This is a challenging task in light of the fact that in general all routes depend on the set of all concurrent demands. While at the same time, it is impractical to recompute a global (essentially offline) multicommodity problem each time a demand changes.

The first glimpse of tackling this hard setting appeared in Valiant’s work [106] on congestion minimizing routing on the hypercube via oblivious routes. Valiant showed that, at least when the graph is expanding and highly symmetric, it is possible to assign universal routes to each source-sink pair so that the resulting routes would be near optimal for all sets of concurrent demands. These routes were dubbed oblivious, in appeal to their demand independence. Various works [6] have highlighted the importance of routing with concurrent and changing demands. Nevertheless, Valiant’s result had not been extended to general graphs for a long while until a groundbreaking line of work on “oblivious network design” [90, 7, 14, 54, 89]. This work was primarily fueled by a different objective: it had been observed that oblivious routes can be used as a general tool for attaining NP-hard approximations to problems like Min Bisection, Sparsest Cut, Max Concurrent Flow-Sparsest Cut Ratio, Minimum k -Multicut, Online Multicut and others. This research culminated in a work of Räcke [89], describing a polynomial-time algorithm for constructing optimal oblivious routes for general graphs. Räcke’s algorithm falls

short of applying to modern networks since it is not distributed. This issue is addressed in the spectral routing algorithm presented here.

In the distributed setting, representation is intimately tied to computation as well as to the mathematical form of the routing scheme. Racke’s scheme is based on distributions over trees, which turn out to be inconvenient in the distributed setting. Spectral routing instead decomposes routes into electrical flows. Since routing in a distributed system occurs via message forwarding, a routing scheme is abstractly represented as a function $\varrho(v, t)$ which given a current vertex v and destination vertex t outputs a vertex $u = \varrho(v, t)$, adjacent to v , where the message is to be forwarded.¹ The obvious representation of the routing function entails storing $\varrho(v, t)$, for all t , at v , and doing so at every v . This requires $\Theta(n)$ space per vertex, which can be reduced significantly in some cases (depending on graph structure, objective function or relaxation). As we see in this chapter, clever encodings of the routing function (quite different from the trivial one) allow for significantly more efficient updates.

The hardness of routing also depends on the objective (or cost) function at hand. Informally, some routing algorithms try to minimize path length, while others try to minimize “congestion” or in other words to maximize a certain measure of global throughput of the network across all active demands. The former has earned the name ℓ_1 -routing whereas the latter ℓ_∞ -routing. This is not coincidental. As we show later, most routing problems amount to the same multicommodity-type convex program with differing norms in the objective and, in fact, ℓ_p -routing, for $1 \leq p \leq \infty$, is defined naturally and often corresponds to a real-world objective. For example, ℓ_2 -routing minimizes latency.

In the static setting, when graphs do not change, ℓ_1 -routing is easy. Computing all pairs shortest paths, even in a distributed manner, is straightforward. This is why research on ℓ_1 -routing has placed more focus on efficient representation, rather than discovery. Notably, Thorup and Zwick [104] construct schemes for general graphs which trade off representation complexity for relaxed guarantees. Their schemes use $O(n^{1/k})$ bits per vertex, while producing paths whose length is within a factor $2k - 1$ of the optimal.

In modern networks, however, ℓ_1 -routing is rarely practical due to its sensitivity to graph changes. Instead, ℓ_∞ -routing is used as an approximate model of the objective: Provide as much bandwidth as possible to all concurrent demands in a fair manner. This objective is significantly harder to optimize for. However, its stability against graph change justifies its popularity. In this chapter we build ℓ_∞ -routing schemes for distributed networks. We leave it open to build such schemes for other ℓ_p norms, by combining with recent techniques on approximating matrix p -norms [13].

The real-time, distributed routing algorithms that we develop here are a step towards a future architecture of the Internet. In this vision, all devices (mobiles, desktops, etc.) can dynamically connect to other “nearby” devices (WiFi networks, Internet Service Providers, next-door neighbors, etc.). And while aware only of their neighbors, devices would be able to

¹In some cases, the routing function may take additional input like e.g. the origin vertex of the message. This information is typically carried inside the message header.

route packets globally to any other destination device. This ambitious goal is motivated by various growing problems with current Internet infrastructure. It aims to reduce the increasing costs of global network administration, which is currently largely manual and centralized. And it addresses the bubbling need of end users for networking independence and non-censorship from corporate and governmental entities.

1.1 Routing theory

In the next few sections, we develop the main tools of routing theory and discuss the historical works building up to distributed oblivious routing.

In Section 1.2 we introduce the key objects at play in routing theory, which constitute a minimal indispensable vocabulary on the subject. The basic concepts in routing, such as demands, flows, routing functions, and so forth, have enjoyed various representations across textbooks and works on the subject. We cast these concepts in a fine-tuned manner in the language of linear algebra. These early definitions of Section 1.2 already start to pave the way to the main unification of routing schemes that is found later in Section 1.4.

Following this, in Section 1.3, we review the line of work that initiated oblivious routing and the main results therein.

1.2 Demands, flows and routing

The object of interest is a graph $G = (V, E)$ (with $V = [n]$ and $|E| = m$) undirected, positively edge-weighted by $w_{u,v} \geq 0$, and not necessarily simple. The intention is that higher $w_{u,v}$ signifies stronger connectivity between u and v ; in particular, $w_{u,v} = 0$ indicates the absence of edge (u, v) . For analysis purposes, we fix an arbitrary orientation “ \rightarrow ” on the edges (u, v) of G , i.e. if (u, v) is an edge then exactly one of $u \rightarrow v$ or $v \rightarrow u$ holds. Two important operators are associated to every G . The *discrete gradient* operator $B \in \mathbf{R}^{E \times V}$ (also written as ∇), sending functions on V to functions on the *undirected* edge set E , is defined as $\chi_{(u,v)}^* B := \chi_u - \chi_v$ if $u \rightarrow v$, and $\chi_{(u,v)}^* B := \chi_v - \chi_u$ otherwise, where χ_y is the Kronecker delta function with mass on y . For $e \in E$, we use the shorthand $B_e := (\chi_e B)^*$. The *discrete divergence* operator is defined as B^* and is also written as ∇^* .

A (*single-commodity*) *demand* of amount $\alpha > 0$ between $s \in V$ and $t \in V$ is defined as the vector $d = \alpha(\chi_s - \chi_t) \in \mathbf{R}^V$. A (*single-commodity*) *flow* on G is defined as a vector $f \in \mathbf{R}^E$, so that $f_{(u,v)}$ equals the flow value from u towards v if $u \rightarrow v$, and the negative of this value otherwise. We also use the notation $f_{u \rightarrow v} := f_{(u,v)}$ if $u \rightarrow v$, and $f_{u \rightarrow v} := -f_{(u,v)}$ otherwise. We say that flow f *routes* demand d if $B^* f = d$. This is a linear algebraic way of encoding the fact that f is an (s, t) -flow of amount α . A *multi-commodity demand*, also called a *demand set*, is a matrix whose columns constitute the individual demands’ vectors. It is given as the direct product $\oplus_\tau d_\tau$ of its columns. A *multi-commodity flow* is represented as a matrix $\oplus_\tau f_\tau$, given as a direct product of its columns, the single-commodity flows. For clarity, we write $f_{\tau,e}$ for $(f_\tau)_e$. The flow $\oplus_\tau f_\tau$ *routes* the demand set $\oplus_\tau d_\tau$ if $B^* f_\tau = d_\tau$, for all τ , or in matrix notation $B^*(\oplus_\tau f_\tau) = \oplus_\tau d_\tau$.

Whenever we want to indicate that a variable d is a multi-demand, we write $d \in (\mathbf{R}^V)^\oplus$. The key notational point here is that the \oplus -superscript notation X^\oplus represents the space of all finite direct products of elements of X . And so, similarly, $(\mathbf{R}^E)^\oplus$ is the space of multi-flows.

1.3 Oblivious network design

Oblivious network design was first introduced as a concept in [50]. It was motivated by applications of multi-commodity flow problems in real-time settings, where it is infeasible to compute commodity flows using classical algorithms due to harsh time constraints and continuously changing demands.

The context is an undirected graph $G = (V, E)$ (with $|V| = n$ and $|E| = m$) which represents a communication network. A *single typed demand* d is 4-tuple (s, t, α, τ) , consisting of a source vertex $s \in V$, a sink vertex $t \in V$, an amount $\alpha \in \mathbf{R}_{\geq 0}$ and an abstract type $\tau \in \mathcal{T}$, which comes from a fixed type set \mathcal{T} with $|\mathcal{T}| = K$. A single typed demand can also be viewed as a pair (d, τ) , where $d \in \mathbf{R}^V$ is a demand vector and τ is the abstract type. In applications, a demand of this kind represents a requirement to send some prespecified amount of a given commodity from the source vertex to the sink vertex. The type represents a characteristic, like quality-of-service level, of the commodity at hand.

Oblivious network design is an optimization problem where, given G , the goal is to output a set of flows $F = \{f_{st}\}_{(s,t) \in \binom{V}{2}}$, one for each pair of distinct vertices $(s, t) \in \binom{V}{2}$, such that f_{st} routes $\chi_s - \chi_t$ (one unit of flow between s and t). The intention here is that after F has been outputted and hence committed to, we would like to route any set of demands using F .

More specifically, let $D = \{(s_i, t_i, \alpha_i, \tau_i)\}_i$ be a given set of typed demands. We would route each d_i using the flow $\alpha_i f_{s_i, t_i}$, where f_{s_i, t_i} comes from F , and we annotate each flow with the respective demand type τ_i . This gives us the *routing* $\{(\alpha_i f_{s_i, t_i}, \tau_i)\}_i$. We abbreviate this routing by $F(D)$.

The objective of oblivious network design is to output an F that is competitive in the worst-case against a routing that is optimal for D in G . In order to formalize this, we need to define a notion of cost that a particular routing $\{(\alpha_i f_{s_i, t_i}, \tau_i)\}_i$ incurs on G .

Assume we are given a routing (set of flows, annotated with type information) $R = \{(f_i, \tau_i)\}_i$. In order to compute the cost of R , an oblivious network design problem specifies two special functions: a load function ℓ_{load} and an aggregation function ℓ_{agg} .

The *load function* ℓ_{load} maps a routing R to a vector of edge loads in $\mathbf{R}_{\geq 0}^E$. The load function has some required structure. For an edge e , we define the *edge traffic* as the vector $t_e = (t_{e,1}, \dots, t_{e,K})$, where $t_{e,\tau} = \sum_{(f_i, \tau_i): \tau_i = \tau} |(f_i)_e|$ is the sum of all flow amounts over e over all flows that are of type τ . We require that $\ell_{\text{load}}(R)_e$ is a function only of t_e . In other words, ℓ_{load} is elementwise composition of m separate functions $(\ell_{\text{load}})_e : \mathbf{R}^K \rightarrow \mathbf{R}$.

The *aggregator function* $\ell_{\text{agg}} : \mathbf{R}_{\geq 0}^E \rightarrow \mathbf{R}_{\geq 0}$ maps the vector of edge loads to a single number that represents the total cost of the routing, which we call *congestion*. In applications, this function is typically a convex norm. Later, we are going to see how the combined choice of ℓ_{load} and ℓ_{agg} encodes various classical combinatorial problems. To summarize so far, the cost of a routing R is given by $\ell_{\text{agg}}(\ell_{\text{load}}(R))$.

We can now return to the objective of the optimization problem. Given G , ℓ_{load} and ℓ_{agg} we aspire to output a set of all-pairs unit flows F , so that for the worst-case demand D , the cost of

$F(D)$ is minimal compared to the cost of $R_{\text{opt}}(D)$ where the latter is the minimum cost routing for D . Formally,

$$F = \arg \min_{F'} \max_D \frac{\ell_{\text{agg}}(\ell_{\text{load}}(F'(D)))}{\ell_{\text{agg}}(\ell_{\text{load}}(R_{\text{opt}}(D)))}$$

The *competitive ratio* of the output is given by

$$\eta = \min_{F'} \max_D \frac{\ell_{\text{agg}}(\ell_{\text{load}}(F'(D)))}{\ell_{\text{agg}}(\ell_{\text{load}}(R_{\text{opt}}(D)))}$$

It is instructive to note that the oblivious design problem has a natural *offline* version. The main difference is that in the offline version the demands D are given as part of the problem, and can therefore be taken into account during the computation of F . Formally, the offline problem is expressed by

$$F = \arg \min_{F'} \ell_{\text{agg}}(\ell_{\text{load}}(F'(D)))$$

The offline problems are clearly strictly simpler than the oblivious ones, since more information is given in the problem input. It is worth noting that all of these problems also possess an *online* version, which in terms of difficulty sits between offline and oblivious. In the online version demands are revealed one at a time, and routing decision have to be made before the next demand is revealed.

1.3.1 Specializations

To demonstrate the generality of the network design problem, we are going to show that different choices of ℓ_{agg} , ℓ_{load} and the size of the type set K specialize to various flavors of classical combinatorial optimization problems.

When $K = 1$, $(\ell_{\text{load}})_e$ are identity, and $\ell_{\text{agg}} = \ell_{\infty}$, the offline problem is the Maximum Concurrent Multi-commodity Flow problem.

When $K = 1$, $(\ell_{\text{load}})_e$ are concave functions, and $\ell_{\text{agg}} = \ell_1$, the offline problem is a buy-at-bulk or rent-or-buy network design problem.

When $K = \binom{V}{2}$, $(\ell_{\text{load}})_e = \ell_{\infty}$, $\ell_{\text{agg}} = \ell_1$ and each (s, t) pair gets a unique type, the offline problem is a fractional Steiner forest problem where one needs to buy fractional edge capacities so that each pair appearing in a request is connected in the resulting graph.

Initially, Gupta et al. [50] have developed algorithms for the oblivious versions of the above problems. More generally, they deal with the cases when the load functions $(\ell_{\text{load}})_e$ are either monotone sub-additive or monotone norms, and the aggregation function ℓ_{agg} is either ℓ_1 or ℓ_{∞} . They obtain poly-logarithmic competitive ratios in these cases.

In a subsequent work, Englert et al. [37] extend these results to the case where $(\ell_{\text{load}})_e$ is a monotone norm and $\ell_{\text{agg}} = \ell_p$, for $1 \leq p \leq \infty$. For these cases, they achieve a $O(\log n)$ competitive ratio. This result also implies a $O(\log^p n)$ competitive ratio for the following oblivious specialization of the network design problem:

When $K = 1$, $(\ell_{\text{load}})_e(t) = t^p$, and $\ell_{\text{agg}} = \ell_1$, the oblivious problem models latency in traffic networks [94, 93, 43]. In this setting every edge possesses a latency function ℓ_{lat} which describes the latency incurred by all flows passing through this edge, as a function of the total edge traffic. The goal is to minimize the average latency over all network links, given by $\propto \sum_e t_e \ell_{\text{lat}}(t_e)$ where t_e is the total edge traffic.

For example, if the latency functions are linear, as is the case in some TCP/IP networks [43], the latency minimization problem can be cast in the oblivious design framework with $K = 1$, $(\ell_{\text{load}})_e = t^2$ and $\ell_{\text{agg}} = \ell_1$.

1.3.2 Lower bounds

Gupta et al. [50], who introduce oblivious network design, show that if $\ell_{\text{agg}} = \ell_\infty$, the load function $(\ell_{\text{load}})_e$ must be a norm since there exist sub-additive load functions for which no oblivious algorithm is $o(n^{1/4})$ -competitive.

The minimum congestion routing version, where $K = 1$, $(\ell_{\text{load}})_e$ is identity and $\ell_{\text{agg}} = \ell_\infty$, has a lower bound of $\Omega(\log n)$ even in the online setting which was shown by Bartal et al. [12] and Maggs et al. [78].

The fraction Steiner tree problem, where $K = \binom{V}{2}$, $(\ell_{\text{load}})_e = \ell_\infty$ and $\ell_{\text{agg}} = \ell_1$, has a lower bound of $\Omega(\log n)$ even if it is not required that the routing scheme be tree-based. This is due to Waxman et al. [56].

Lawler et al. [72] show that it is not possible to design aggregation function oblivious schemes as any fixed scheme will suffer a competitive ratio of $\Omega(\sqrt{n})$ for some p , where $\ell_{\text{agg}} = \ell_p$. They also present a matching competitive ratio upper bound.

Englert et al. [37] show that for the case of latency minimization, where $K = 1$, $(\ell_{\text{load}})_e(t) = t^p$ and $\ell_{\text{agg}} = \ell_1$, there is no oblivious routing scheme that achieves a competitive ratio of $o(\log^p n / (\log \log n)^p)$.

1.3.3 Bottleneck approximating decompositions

Prior to our work on electric routing [62] and the related work [72], all known oblivious routing schemes were based on convex combinations of trees. This line of research culminated in an optimal oblivious routing scheme for congestion (the regime $K = 1$, $\ell_{\text{agg}} = \ell_\infty$, ℓ_{load} identity) that was discovered by Räcke [89]. Notably, Räcke points out that oblivious routing schemes for congestion are, in fact, a form of graph approximation that preserves the cut structure of the graph. This can be juxtaposed to graph metric approximations, like Bartal's probabilistic tree approximations [10], where the approximation preserves distances instead.

It turns out that one can give a general definition of “graph approximation”, which specializes naturally to either of the above cases by changing a certain “cost” function. In this section, we develop the notion of graph approximation formally. Intuitively, a *graph approximation* is a mathematical object that approximates the “bottlenecks” of a graph while at the same time having a computationally convenient form for the purposes of optimization. In other words, the definition is tailored to enable the following use case:

Given a flow-type optimization problem on G , there is a map that converts the problem to one on H (the graph approximation). On H , the respective problem can be solved exactly and efficiently, producing a solution that can be mapped back to G . It is then guaranteed, that this final solution is not too far from the optimal answer on G .

Multicommodity flows

For the purposes of defining graph approximations, we need to recall the classical *Concurrent Multicommodity Flow Problem*. This problem is identical to oblivious network design with the exception that the demands are given ahead of time.

In particular, we are given a set of (untyped) demands (s_i, t_i, α_i) , represented by source and sink vertices and an amount, or by their respective demand vectors $d_i \in D_G$. The goal is to route the set $\oplus_i d_i$ in an underlying weighted undirected graph G , using flows $\oplus_i f_i$ that minimize a cost function. The cost function $c : F_G^\oplus \rightarrow \mathbf{R}$ maps multiflows to a non-negative real. We restrict our interest to cost functions of the form $\ell_{\text{agg}}(\ell_{\text{load}}(\cdot))$, as in our discussion so far.

When the cost function is given by $\ell_{\text{agg}} = \ell_\infty$ and $\ell_{\text{load}} = \ell_1$, the problem is known as *congestion minimization*.

Graph approximations

Intuitively, a graph approximation of a graph G is a mathematical object which is “*nearly equivalent*” to G with respect to Minimum Congestion Multicommodity Problems, but is also “*simpler*” than G in that exact problem solutions can be found by fast (and usually simple) algorithms.

The notion of a graph approximation here is purposefully broad. It generalizes prior work in different domains by varying a certain cost metric while being agnostic to the *implementation* of the graph approximation. E.g. sub-trees, dominating trees, spanners and so forth are all different types of implementations of graph approximations. For its generality, the definition has a slight categorical flavor.

We let G be a graph. This graph has an associated space of flows $F_G \cong \mathbf{R}^E$ and a space of demands $D_G \cong \mathbf{R}^V$, as we have seen earlier. These two spaces are related by an operator $\nabla_G^* : F_G \rightarrow D_G$ which maps every flow to its corresponding demand vector. (This is the divergence operator defined earlier. We subscript it with G to highlight the contextual graph.)

A *graph approximation* H , is an abstract object which, like a graph, has a designated space of flows F_H , a space of demands D_H and a divergence operator $\nabla_H^* : F_H \rightarrow D_H$. Intuitively, in order for H to be an approximation of G , it should fit the following description:

Given an optimization problem specified by a multi-demand on G , if we map the problem to H , solve it there and map the solution multi-flow back to G , the resulting cost should not be much larger than if we had solved the problem directly on G .

We represent the specific minimization problem by a cost function $c_G : F_G^\oplus \rightarrow \mathbf{R}_{\geq 0}$, which maps a multi-flow to a non-negative real cost. Note that the cost function is a property of the problem, not of the graph or the approximation. Consequently we define a “solve” function $S_G : D_G^\oplus \rightarrow F_G^\oplus$ via

$$S_G(d_G) := \arg \min_{f_G \in F_G^\oplus : \nabla_G^*(f_G) = d_G} c_G(f_G),$$

which takes a problem as a multi-demand and outputs the solution as a multi-flow. A similar solve function $S_H : D_H^\oplus \rightarrow F_H^\oplus$ is required on H , and the latter is a property of the graph approximation H . Naturally, S_H is expected to respect the problem constraints which is to say:

- (i) (*Constraint condition*) For all $d_H \in D_H^\oplus$, $\nabla_H^*(S_H(d_H)) = d_H$. In other words, the solution multi-flow actually meets the requested demand.

You will notice, from our informal requirement above, that we need to be able to convert a problem, i.e. a multi-demand, from G to H . We denote the function that achieves this by $\mu_{GH} : D_G \rightarrow D_H$. Additionally, we need to be able to convert a solution, i.e. multi-flow, obtained on H to one on G . We denote this map by $\mu_{HG} : F_H \rightarrow F_G$.

We are now sufficiently equipped to define what it means for H to be a κ -approximation for G in a very general sense. Two conditions must be met:

- (ii) (*Constraint preservation*) For all $f_H \in F_H^\oplus$ and $d_G \in D_G^\oplus$, $\nabla_H^*(f_H) = \mu_{GH}(d_G)$ implies $\nabla_G^*(\mu_{HG}(f_H)) = d_G$.

In other words, given any multi-flow f_H on H , let its respective demand in H be $d_H := \nabla_H^*(f_H)$. Consequently, if d_H happens to be the image $\mu_{GH}(d_G)$ of some demand d_G in G , then it better be the case that if we map f_H to G , via $f_G := \mu_{HG}(f_H)$, then the demand of the image f_G should be the same as d_G .

(As a side remark, notice that this condition implies that if $d' \neq d''$ then $\mu_{GH}(d') \neq \mu_{GH}(d'')$).

This condition ensures that when we start with a problem d_G on G , and map it to a problem $d_H = \mu_{GH}(d_G)$ on H , the eventual solution $f_H = S_H(d_H)$, when pushed to G as $f_G = \mu_{HG}(f_H)$, will actually meet the initial problem constraints $\nabla_G^*(f_G) = d_G$.

(iii) (*Approximation strength*) For all $d_G \in D_G^\oplus$,

$$c_G(\mu_{HG}(S_H(\mu_{GH}(d_G)))) \leq \kappa \cdot c_G(S_G(d_G))$$

This simply says that if we “solve through H ” we shouldn’t do much worse than if we solved directly in G . The one-sided inequality suffices, since we are dealing with a minimization problem.

When conditions (i), (ii) and (iii) are met, the tuple $(H, \mu_{GH}, \mu_{HG}, S_H)$ comprises a κ -graph approximation of G for the cost metric c_G .

Remark 1.3.1. Variations on this definition are possible. E.g. S_H could be removed from the definition and replaced by

$$S_H(d_H) := \arg \min_{f_H \in F_H^\oplus: \nabla_H^*(f_H) = d_H} c_H(f_H),$$

where now the cost c_H must be defined universally so that it does not change much when going through μ_{HG} . These variations are beyond our point of interest.

In the case $c_G = \ell_\infty$, the resulting graph approximations are known as *cut-based decompositions* and are used for approximating graph bottlenecks for the purposes of multicommodity routing [90, 89, 54]. In the other extreme, when $c_G = \ell_1$, the resulting type of graph approximations are widely known as *multiplicative distance-approximation oracles* [10, 11, 38].

1.3.4 Tree-based routing

As we mentioned in the previous section, seminal works in approximating graph bottlenecks [89] and approximating graph metrics [10] use the same underlying object—trees. And more specifically, convex combinations of “decomposition trees”, also previously known as “hierarchical tree decompositions”. We dedicate this section to a precise definition of these and, further, to translating this definition to the language of linear algebra. This translation is an important step on the path of unifying important graph approximation techniques, by casting them in the framework of “linear routing operators” (which are addressed in a later section).

Decomposition trees

Our context will be an undirected graph $G = (V, E)$. A *decomposition tree* for G is a tree $T = (V_T, E_T)$, whose leaf nodes are in one-to-one correspondence with V . Additionally, every non-leaf node of T corresponds to a vertex in G and every edge of T corresponds to a path in G between the respective vertices.

Formally, a *vertex map* $\sigma_{\text{vertex}} : V_T \rightarrow V$ maps the nodes of T onto the vertices of G , such that restricted to the leaf set of T the mapping σ_{vertex} is one-to-one. An *edge map* $\sigma_{\text{edge}} : E_T \rightarrow E^*$ maps the edges of T to paths in G such that (u_T, v_T) is mapped to a path connecting $\sigma_{\text{vertex}}(u_T)$

and $\sigma_{\text{vertex}}(v_T)$. Inverse maps are also included in a decomposition tree. The inverse vertex map $\sigma_{\text{vertex}}^+ : V \rightarrow V_T$ maps each vertex of G to the corresponding leaf node of T . The inverse edge map $\sigma_{\text{edge}}^+ : E \rightarrow E_T^*$ sends each (u, v) of G to the unique shortest path in T connecting $\sigma_{\text{vertex}}^+(u)$ and $\sigma_{\text{vertex}}^+(v)$.

This completes the “traditional” definition. We are now going to extend this definition by replacing each of the four σ -maps with four respective ξ -maps that are linearized versions. Let $D_G \cong \mathbf{R}^V$ and $D_T \cong \mathbf{R}^{V_T}$ be the demand spaces of G and T , respectively. Similarly, let $F_G \cong \mathbf{R}^E$ and $F_T \cong \mathbf{R}^{E_T}$ be the flow spaces of G and T , respectively.

We construct the *demand map* $\xi_{\text{demand}} : D_T \rightarrow D_G$ as a linearization of σ_{vertex} . This is done by defining $\xi_{\text{demand}}(\chi_u) = \chi_{\sigma_{\text{vertex}}(u)}$, for all $u \in V_T$ that are leaf nodes; and $\xi_{\text{demand}}(\chi_u) = 0$ for non-leaf u . The inverse map $\xi_{\text{demand}}^+ : D_G \rightarrow D_T$ is constructed similarly. Using the same idea, we construct the *flow map* $\xi_{\text{flow}} : F_T \rightarrow F_G$ as a linearization of σ_{edge} . For every edge $e_T \in E_T$ of T , let π_{e_T} be the flow vector in F_G of the path $\sigma_{\text{edge}}(e_T)$ in G . We then define $\xi_{\text{flow}}(\chi_{e_T}) := \pi_{e_T}$ for all such e_T . (Since χ_{e_T} form a basis of F_T , ξ_{flow} is fully defined by its values on χ_{e_T} .) In a similar vein, let $\chi_e \in F_G$ be the flow vector of $e \in E$, and let $\tau_e \in F_T$ be the flow vector of the path $\sigma_{\text{edge}}^+(e)$ in T . We then define $\xi_{\text{flow}}^+(\chi_e) := \tau_e$, for all $e \in E$.

In summary, the decomposition tree is now the tuple $(T, \xi_{\text{demand}}, \xi_{\text{demand}}^+, \xi_{\text{flow}}, \xi_{\text{flow}}^+)$.

Cut-based decompositions via trees

It is easy to see that any decomposition tree is a graph approximation. Given a demand $\oplus d$ on G , we can push it to T via ξ_{demand}^+ , solve the problem on T where the solution is always unique and independent of the cost function, and push the answer back to G via ξ_{flow} .

Furthermore, since a tree provides a unique routing for any given demand, it is clear that if $\oplus_i f_i$ is the routing for a multi-demand $\oplus_i d_i$, then each individual flow f_i depends only on d_i and not on the whole multi-demand. (This is not true for graphs in general.) Therefore, tree routing constitutes oblivious routing and, in fact, we could represent it quite concisely in terms of linear algebra as we explain next.

It is easily checked that for a tree $T = (V_T, E_T)$, the function that maps demands to the unique routes in the tree $\varrho_T : \mathbf{R}^{V_T} \rightarrow \mathbf{R}^{E_T}$ is linear. We can thus derive the routing function of a decomposition tree as $\xi_{\text{flow}} \cdot \varrho_T \cdot \xi_{\text{demand}}^+$, which is linear since its components are.

For graph approximation purposes, it turns out that a single decomposition tree is not sufficient for most cost measures. Instead, a *convex combination of decomposition trees* is used. Given decompositions trees T_i with respective routing operators ϱ_i , the routing operator of a convex combination of decomposition trees is simply defined as $\sum_i \lambda_i \varrho_i$, where $\sum_i \lambda_i = 1$ and $\lambda_i \geq 0$, for all i . The power of this type of graph approximation was established by Räcke’s landmark result:

Theorem 1.3.2 (Räcke [89]). *For every weighted, undirected graph G , there exists a polynomially computable convex combination over decomposition trees which $O(\log n)$ -approximates G with respect to congestion cost, given by $\ell_{\text{agg}} = \ell_\infty$ and $(\ell_{\text{load}})_e(x) = x/w_e$, where w_e represents*

the edge weight.

1.4 Linear routing schemes

In this section, we are going to introduce a class of routing schemes, which can be concisely represented by linear operators. This class was first introduced in work of the author [62] as a superclass of electric routing. Independently Lawler et al. [72] consider electric routing and implicitly utilize linear routing schemes. Linear schemes generalize Räcke’s tree-based routing schemes as well as the electric routing schemes of [62, 72].

Linear schemes are interesting for a few reasons. They generalize the key theorems in the analysis of tree routing [72] and electric routing [62]. Roughly speaking, in both cases it is proven that there exists a universal set of worst-case demands that does not depend on the routing operator. The proofs in these works are seemingly very different, however the language of linear routing unifies them in a single conceptually-cleaner proof, given in Section 1.4.2.

Furthermore, since the class of tree-based schemes contains an optimal scheme for every graph, as shown by Räcke [89], it follows that the super-class of linear schemes does as well. Therefore, we can search for an optimal scheme directly in the space of linear schemes. This search, turns out, is a standard convex minimization problem and therefore succumbs to solutions by a long list of well-understood and fine-tuned optimization methods like Interior Point Methods, the Simplex Method, Multiplicative Weight Updates and many others. This is in contrast to Räcke’s algorithm which itself is a special form of a gradient descent and requires a custom implementation.

1.4.1 The routing operator

Recall that a (general) oblivious routing scheme is any function $\varrho : \mathbf{R}^V \rightarrow \mathbf{R}^E$, mapping a demand vector to a flow vector, which has the property that $\varrho(d)$ routes d when d is a valid single commodity demand (i.e. of the form $\chi_s - \chi_t$). Formally, for any $s \neq t \in V$, we require that $\nabla^*(\varrho(\chi_s - \chi_t)) = \chi_s - \chi_t$. A *linear routing scheme* is one where ϱ must also be linear. In other words, we seek a linear operator $\varrho : \mathbf{R}^V \rightarrow \mathbf{R}^E$ such that $\nabla^*(\varrho(d)) = d$ whenever $\langle \mathbf{1}, d \rangle = 0$. We generally do not care how ϱ behaves on $\mathbf{1}$ (since this is not a meaningful demand vector) however for mathematical convenience we throw in the condition that $\varrho(\mathbf{1}) = 0$, which results in the following definition.

Definition 1.4.1. A *linear routing scheme* is a linear operator $\varrho : \mathbf{R}^V \rightarrow \mathbf{R}^E$ with $\nabla^*(\varrho(d)) = \pi_{\perp \mathbf{1}}(d)$, for all $d \in \mathbf{R}^V$, where $\pi_{\perp \mathbf{1}}$ is projection onto the space orthogonal to $\mathbf{1}$. This can be expressed concisely via the matrix identity

$$\nabla^* \varrho = \pi_{\perp \mathbf{1}},$$

which we call the *routing identity*.

We have already seen linear routing. Recall that in Section 1.3.4 we showed that routing via a convex combination of decomposition trees can be represented as a linear operator. We will later see another example—that of electric routing.

Note that ϱ is equivalently a matrix in $\mathbf{R}^{E \times V}$. The routing identity describes the set of valid routing operators as an affine subspace of $\mathbf{R}^{E \times V}$. In particular, for the difference $\Delta = \varrho' - \varrho''$ between two routing operators, we have

$$\nabla^* \Delta = \nabla^*(\varrho' - \varrho'') = \nabla^* \varrho' - \nabla^* \varrho'' = \pi_{\perp \mathbf{1}} - \pi_{\perp \mathbf{1}} = 0$$

Conversely, if Δ is such that $\nabla^* \Delta = 0$ and ϱ is a routing scheme, in that $\nabla^* \varrho = \pi_{\perp \mathbf{1}}$, then

$$\nabla^*(\varrho + \Delta) = \nabla^* \varrho + \nabla^* \Delta = \pi_{\perp \mathbf{1}}$$

And so $\varrho + \Delta$ is a routing operator as well.

The condition $\nabla^* \Delta = 0$ says that every column of Δ is a circulation. And therefore, not surprisingly, the bottom line is that starting from an arbitrary routing operator we can get to any other one by adding circulations to each of its columns.

For notational convenience, we extend ϱ to a function over demand sets by defining $\varrho(\oplus_{\tau} d_{\tau}) = \oplus_{\tau} \varrho(d_{\tau})$. This identity says that each demand in a set is routed independently of the others by its corresponding ϱ -flow. If ϱ is viewed as a matrix, this identity already holds using the convention that \oplus denotes column vector concatenation.

As a reminder, let's rewrite the program for oblivious routing for the case where we insist on finding a linear routing scheme:

$$(1.4.1) \quad \min_{\varrho: \nabla^* \varrho = \pi_{\perp \mathbf{1}}} \max_{\oplus d} \frac{\ell_{\text{agg}}(\ell_{\text{load}}(\varrho(\oplus d)))}{\ell_{\text{agg}}(\ell_{\text{load}}(R_{\text{opt}}(\oplus d)))}$$

The “master theorem” of linear routing that we cover in the next section will allow us to simplify this program significantly.

1.4.2 Geometry of congestion

The oblivious competitive ratio of linear schemes is amenable to theoretical analysis because of a master theorem which says that universal worst-case demands exist independent of the routing operator (this is akin to the Min-Max Theorem for zero-sum games) and there is an easy way to find them. For example, for $\ell_{\text{agg}} = \ell_{\infty}$ the set of unit demands between the endpoints of all graph edges $\oplus_{(u,v) \in E} (\chi_u - \chi_v)$ is the worst-case.

This is what makes their analysis at all possible in both [62] and [72]. Furthermore, it is what helps cast the search for an optimal routing scheme as a standard norm minimization problem. A similar “master theorem” holds for tree-based routing, and it is the starting point of the analyses in [89] and [37].

The following theorem works in the regime of a single commodity type $K = 1$, an orientation-independent and monotone load function $(\ell_{\text{load}})_e$ and a monotone aggregation function ℓ_{agg} . Orientation independence, given by $(\ell_{\text{load}})_e(x) = (\ell_{\text{load}})_e(-x)$, means that the load is independent from which way the flow goes over an edge.

Some notation is due before we proceed. For a set of flows $\oplus f \in (\mathbf{R}^E)^\oplus$, we call the non-negative vector $t \in \mathbf{R}^E$, defined by $t_e = \sum_f |f_e|$, the *traffic* of $\oplus f$.

Theorem 1.4.2. (*Master Theorem*) *Let ϱ be a linear routing scheme, ℓ_{agg} be a monotone aggregation function, and ℓ_{load} be a monotone, orientation-independent load function. If $D = \oplus d$ is a set of demands, $F = \oplus f$ is an optimal routing and t is the traffic vector of F , then*

$$\ell_{\text{agg}}\left(\ell_{\text{load}}\left(\varrho(D)\right)\right) \leq \ell_{\text{agg}}\left(\ell_{\text{load}}\left(\varrho\left(\oplus_{(u,v) \in E} t_{uv}(\chi_u - \chi_v)\right)\right)\right)$$

Proof. Let $D = \oplus d$ be a fixed set of demands and let $F = \oplus f$ be an optimal routing for D . Denote by t_e , for all $e \in E$, the traffic over edge e induced by $\oplus f$. Formulaically:

$$(1.4.2) \quad \sum_{f \in F} |f_e| = t_e$$

We are going to show $|\varrho(D)| \leq |\varrho(\oplus_{(u,v) \in E} t_{uv}(\chi_u - \chi_v))|$ elementwise. Note that generally $|\oplus f| \leq |\oplus g|$ implies $\ell_{\text{load}}(\oplus f) \leq \ell_{\text{load}}(\oplus g)$, since ℓ_{load} is orientation independent and monotone. Thus, it would follow that

$$\ell_{\text{load}}(\varrho(D)) \leq \ell_{\text{load}}\left(\varrho\left(\oplus_{(u,v) \in E} t_{uv}(\chi_u - \chi_v)\right)\right).$$

And since ℓ_{agg} is monotone, the latter would imply that

$$\ell_{\text{agg}}(\ell_{\text{load}}(\varrho(D))) \leq \ell_{\text{agg}}\left(\ell_{\text{load}}\left(\varrho\left(\oplus_{(u,v) \in E} t_{uv}(\chi_u - \chi_v)\right)\right)\right),$$

concluding as desired.

For starters, recall that if f_d in F flows d in D , the routing condition asserts that $d = \sum_{(u,v) \in E} f_{d,uv}(\chi_u - \chi_v)$. This allows us to express the routing of ϱ in terms of F ,

$$\varrho(D) = \varrho(\oplus_{d \in D} d) = \oplus_{d \in D} \varrho\left(\sum_{(u,v) \in E} f_{d,uv}(\chi_u - \chi_v)\right),$$

where f_d is the flow in F that routes demand d and $f_{d,uv}$ is the amount of flow that f_d flows on $(u, v) \in E$.

Next, we separate the contribution of the different $f_{d,uv}$ components using the linearity of ϱ and the triangular inequality for $|\cdot|$

$$\begin{aligned} \left| \oplus_{d \in D} \varrho\left(\sum_{(u,v) \in E} f_{d,uv}(\chi_u - \chi_v)\right) \right| &= \left| \oplus_{d \in D} \sum_{(u,v) \in E} \varrho(f_{d,uv}(\chi_u - \chi_v)) \right| \\ &\leq \left| \oplus_{d \in D} \oplus_{(u,v) \in E} \varrho(f_{d,uv}(\chi_u - \chi_v)) \right| \end{aligned}$$

Having done this, we can switch the order of the \oplus -summation and, using linearity of ϱ ,

combine all component flows that correspond to the same demand

$$\begin{aligned} \left| \bigoplus_{d \in D} \bigoplus_{(u,v) \in E} \varrho(f_{d,uv}(\chi_u - \chi_v)) \right| &= \left| \bigoplus_{(u,v) \in E} \bigoplus_{d \in D} \varrho(f_{d,uv}(\chi_u - \chi_v)) \right| \\ &= \left| \bigoplus_{(u,v) \in E} \varrho \left((\chi_u - \chi_v) \sum_{d \in D} |f_{d,uv}| \right) \right| \end{aligned}$$

Finally, applying (1.4.2) to the expression above,

$$\begin{aligned} \left| \bigoplus_{(u,v) \in E} \varrho \left((\chi_u - \chi_v) \sum_{d \in D} |f_{d,uv}| \right) \right| &= \left| \bigoplus_{(u,v) \in E} \varrho(t_{uv}(\chi_u - \chi_v)) \right| \\ &= \left| \varrho \left(\bigoplus_{(u,v) \in E} t_{uv}(\chi_u - \chi_v) \right) \right|, \end{aligned}$$

concludes the chain of inequalities with the desired result. \square

1.4.3 Routing as a norm minimization problem

The purpose of the Master Theorem is to simplify the convex program for computing an optimal oblivious scheme (1.4.1). Let us start by observing that the cost function is a norm:

Lemma 1.4.3. *The cost function $\ell_{\text{agg}}(\ell_{\text{load}}(\cdot))$ is a convex norm over the vector space of multi-flows $(\mathbf{R}^E)^\oplus$.*

Proof. We demonstrate the triangle inequality

$$\ell_{\text{agg}}(\ell_{\text{load}}(\bigoplus f' + \bigoplus f'')) \leq \ell_{\text{agg}}(\ell_{\text{load}}(\bigoplus f') + \ell_{\text{load}}(\bigoplus f'')) \leq \ell_{\text{agg}}(\ell_{\text{load}}(\bigoplus f')) + \ell_{\text{agg}}(\ell_{\text{load}}(\bigoplus f''))$$

Since ℓ_{load} is monotone and dependent only on the absolute values of the entries in its arguments, it follows that it is sub-additive $\ell_{\text{load}}(\bigoplus f' + \bigoplus f'') \leq \ell_{\text{load}}(\bigoplus f') + \ell_{\text{load}}(\bigoplus f'')$. This fact and the monotonicity of ℓ_{agg} justify the first inequality above. The second inequality follows from ℓ_{agg} being a norm. \square

To simplify notation and emphasize the fact that the cost function is a norm, we henceforth write $\|\cdot\|_{\text{cost}}$ for $\ell_{\text{agg}}(\ell_{\text{load}}(\cdot))$. The key point is that the Master Theorem helps us simplify the expression for the competitive ratio of ϱ ,

$$\max_{\bigoplus d} \frac{\ell_{\text{agg}}(\ell_{\text{load}}(\varrho(\bigoplus d)))}{\ell_{\text{agg}}(\ell_{\text{load}}(R_{\text{opt}}(\bigoplus d)))}$$

in (1.4.1), by replacing the numerator with the upper bound from the theorem. Let $t(\bigoplus f)$ denote

the traffic vector of $\oplus f$ and $D = \oplus d$. We have:

$$\begin{aligned}
\max_{\oplus d} \frac{\ell_{\text{agg}}(\ell_{\text{load}}(\varrho(\oplus d)))}{\ell_{\text{agg}}(\ell_{\text{load}}(R_{\text{opt}}(\oplus d)))} &= \max_D \frac{\|\varrho(D)\|_{\text{cost}}}{\|\varrho_{\text{opt}}(D)\|_{\text{cost}}} \\
&\stackrel{(*)}{=} \max_D \frac{\|\varrho(D)\|_{\text{cost}}}{\|t(\varrho_{\text{opt}}(D))\|_{\text{cost}}} \\
&\stackrel{(**)}{=} \max_t \frac{\|\varrho(\oplus_{(u,v) \in E} t_{uv}(\chi_u - \chi_v))\|_{\text{cost}}}{\|t\|_{\text{cost}}} \\
&= \max_{t: \|t\|_{\text{cost}}=1} \|\varrho(\oplus_{(u,v) \in E} t_{uv}(\chi_u - \chi_v))\|_{\text{cost}}
\end{aligned}$$

The first identity (*) follows from the fact $\|\oplus f\|_{\text{cost}} = \|t(\oplus(f))\|_{\text{cost}}$ (in the regime $K = 1$). The second (**) follows from applying the upper bound in the nominator and noticing that it can be attained.

The upside of this new expression is that for specific cost functions we know the structure of the set of traffic vectors t , for which $\|t\|_{\text{cost}} = 1$. In particular, in the most-widely studied case $\ell_{\text{agg}} = \ell_{\infty}$, the maximum is attained at $t = \mathbf{1}$. This is because $\mathbf{1} \in \mathbf{R}^E$ is maximal in absolute value in the set $\{t : \|t\|_{\text{cost}} = 1\}$ and $\|\varrho(\cdot)\|_{\text{cost}}$ is monotone. This allows us to further simplify the expression for the competitive ratio to $\|\varrho(\oplus_{(u,v)}(\chi_u - \chi_v))\|_{\text{cost}}$. In matrix notation, this latter expression amounts to $\|\varrho \nabla^*\|_{\text{cost}}$, since $\oplus_{(u,v) \in E}(\chi_u - \chi_v)$ is the matrix of the divergence operator ∇^* .

To summarize, the convex program for $\ell_{\text{agg}} = \ell_{\infty}$ oblivious routing can be written as

$$\min_{\varrho: \nabla^* \varrho = \pi_{\perp \mathbf{1}}} \|\varrho \nabla^*\|_{\text{cost}}$$

This represents a standard norm minimization over an affine subspace and can be solved using a range of techniques like Interior Point Methods [57, 92] or Multiplicative Weight Update algorithms.

1.5 Electric routing

Electric routing is a type of linear routing that sits in a sweet spot with respect to multiple computational considerations. Compared to Räcke’s optimal scheme which achieves optimal competitive ratio of $O(\log n)$ for all graphs in the regime $\ell_{\text{agg}} = \ell_{\infty}$, electric routing achieves ratio $O(\log n)$ only for expanders and ratio $O(\sqrt{n})$ for the remaining graphs. On the other hand, unlike Räcke’s scheme, electric routing can be computed efficiently in distributed systems. And in expanders, the computation is extremely efficient, taking only $O(\log n)$ steps.

Considering that many real-world ad-hoc networks like peer-to-peer social systems are expanders, electric routing is the only practical oblivious routing scheme applicable. Beyond its practical appeal, electric routing is of purely mathematical interests. As we show later, it turns out that the competitive ratio of electric routing is in fact equal to the $\|\cdot\|_{1 \rightarrow 1}$ norm of the Laplacian matrix, which is a natural mathematical object. Here we provide tight bounds to this norm, which is a result that seems to be of independent interest.

Related work

Two bodies of prior literature concern themselves with oblivious routing. One focuses on approximating the shortest-path metric [103, 102, 1, 2], the other focuses on approximating the minimal congestion universally across all possible demand sets [89, 55]. The algorithms in these works are essentially best possible in terms of competitive characteristics, however they are not distributed and do not address (competitive) performance in the presence of churn. It is not obvious how to provide efficient distributed variants for these routing schemes that are additionally resistant to churn. The primary reason for this are the algorithmic primitives used. Common techniques are landmark (a.k.a. beacon) selection [103, 102], hierarchical tree-decomposition or tree-packings [89]. These approaches place disproportionately larger importance on “root” nodes, which makes the resulting schemes vulnerable to individual failures. Furthermore, these algorithms require more than (quasi-)linear time (in the centralized model), which translates to prohibitively slow distributed times.

We are aware of one other work in the theoretical literature by Goyal, et al. [47] that relates to efficient and churn-tolerant distributed routing. Motivated by the proliferation of routing schemes for trees, they show that expanders are well-approximated by the union of $O(1)$ spanning trees. However, they do not provide a routing scheme, since routing over unions of trees is not understood.

Concurrently with this paper, Lawler, et al. [72] study just the congestion of electric flow in isolation from other considerations like computation, representation or tolerance to churn. Their main result is a variant of our graph expansion-based bound on $\|L^\dagger\|_{1 \rightarrow 1}$, given by Theorem 1.5.10. Our approaches, however, are different. We use a geometric approach, compared to a less direct probabilistic one. Our proof exposes structural information about the electric flow, which makes the fault-tolerance of electric routing against edge removal an easy consequence. This is not the case for the proofs found in [72].

1.5.1 Electric ℓ_∞ oblivious routing

Our entire discussion on electric routing will focus on the regime $K = 1$, $\ell_{\text{agg}} = \ell_\infty$ and $\ell_{\text{load}} = \ell_1$. In this regime, much of the notation can be simplified from the more general setting discussed so far. We introduce this notation here.

The *congestion* $\|\cdot\|_{\text{cost}}$ of a multi-commodity flow measures the load of the most-loaded edge, relative to its capacity. It is given by

$$(1.5.1) \quad \|\oplus_\tau f_\tau\|_{\text{cost}} := \max_e \sum_\tau |f_{\tau,e}/w_e| = \|(\oplus_\tau f_\tau)^* W^{-1}\|_{1 \rightarrow 1}, \text{ where } \|A\|_{1 \rightarrow 1} := \sup_{x \neq 0} \frac{\|Ax\|_1}{\|x\|_1}.$$

An *oblivious routing scheme* is a (not necessarily linear) function $R : \mathbf{R}^V \rightarrow \mathbf{R}^E$ which has the property that $R(d)$ routes d when d is a valid single-commodity demand (according to our definition). We extend R to a function over demand sets by defining $R(\oplus_\tau d_\tau) := \oplus_\tau R(d_\tau)$. This says that each demand in a set is routed independently of the others by its corresponding R -flow. We measure the “goodness” of an oblivious routing scheme by the maximum traffic that it incurs on an edge (relative to its capacity) compared to that of the optimal (demand-dependent) routing. This is captured by the *competitive ratio* η_R of the routing scheme R , defined

$$(1.5.2) \quad \eta_R := \sup_{\oplus_\tau d_\tau} \sup_{\substack{\oplus_\tau f_\tau \\ B^*(\oplus_\tau f_\tau) = \oplus_\tau d_\tau}} \frac{\|R(\oplus_\tau d_\tau)\|_{\text{cost}}}{\|\oplus_\tau f_\tau\|_{\text{cost}}}.$$

Let \mathcal{E} denote the (yet undefined) function corresponding to electric routing. Our main theorem states:

Theorem 1.5.1. *For every undirected graph G with unit capacity edges, maximum degree d_{max} and vertex expansion $\alpha := \min_{S \subseteq V} \frac{|E(S, S^c)|}{\min\{|S|, |S^c|\}}$, one has $\eta_{\mathcal{E}} \leq \left(4 \ln \frac{n}{2}\right) \cdot \left(\alpha \ln \frac{2d_{\text{max}}}{2d_{\text{max}} - \alpha}\right)^{-1}$. This is tight up to a factor of $O(\ln \ln n)$.*

The competitive ratio in Theorem 1.5.1 is best achievable for any oblivious routing scheme up to a factor of $O(\ln \ln n)$ due to a lower bound for expanders, i.e. the case $\alpha = O(1)$, given in [52]. Theorem 1.5.1 can be extended to other definitions of graph expansion, weighted and unbounded-degree graphs. We omit these extensions for brevity. We also give an unconditional, albeit much worse, bound on $\eta_{\mathcal{E}}$:

Theorem 1.5.2. *For every unweighted graph on m edges, electrical routing has $\eta_{\mathcal{E}} \leq O(m^{1/2})$. Furthermore, there are families of graphs with corresponding demand sets for which $\eta_{\mathcal{E}} = \Omega(m^{1/2})$.*

1.5.2 Electric flow, effective resistance and the graph Laplacian

In this section, we develop a formal definition of electric flow and the electric routing operator \mathcal{E} . Let $W = \text{diag}(\dots, w_e, \dots) \in \mathbf{R}^{E \times E}$ be the edge weights matrix. We appeal to a known

connection between graph Laplacians and electric current [35, 99]. Graph edges are viewed as wires of resistance w_e^{-1} and vertices are viewed as connection points. If $\varphi \in \mathbf{R}^V$ is a vector of vertex potentials then, by Ohm's law, the *electric flow* (over the edge set) is given by $f = WB\varphi$ and the corresponding demand is $B^*f = L\varphi$ where the (*un-normalized*) Laplacian L is defined as $L = B^*WB$. Central to the present work will be the vertex potentials that induce a desired (s, t) -flow, given by $\varphi^{[s,t]} = L^\dagger(\chi_s - \chi_t)$, where L^\dagger is the pseudo-inverse of L . Thus, the electric flow corresponding to the demand pair (s, t) is $WB\varphi^{[s,t]} = WBL^\dagger(\chi_s - \chi_t)$. We define the *electric routing operator* as

$$(1.5.3) \quad \mathcal{E}(d) = WBL^\dagger d$$

The vector $\mathcal{E}(\chi_s - \chi_t) \in \mathbf{R}^E$ encodes a unit flow from s to t supported on G , where the flow along an edge (u, v) is given by $\llbracket st, uv \rrbracket := \mathcal{E}(\chi_s - \chi_t)_{u \rightarrow v} = (\varphi_u^{[s,t]} - \varphi_v^{[s,t]})w_{u,v}$.² (Our convention is that current flows towards lower potential.) When routing an indivisible message (an IP packet e.g.), we can view the unit flow $\mathcal{E}(\chi_s - \chi_t)$ as a distribution over (s, t) -paths defined recursively as follows: Start at s . At any vertex u , *forward* the message along an edge with positive flow, with probability proportional to the edge flow value. Stop when t is reached. This rule defines the *electric walk* from s to t . It is immediate that the flow value over an edge (u, v) equals the probability that the electric walk traverses that edge.

Let “ \sim ” denote the vertex adjacency relation of G . In the distributed setting (which is of key interest for electric routing), in order to make a (divisible or indivisible) forwarding decision, a vertex u must be able to compute $\llbracket st, uv \rrbracket$ for all neighbors $v \sim u$ and all pairs $(s, t) \in \binom{V}{2}$. This issue pertains to the representation of electric routing in a distributed system and is discussed in Section 1.5.3

²The bilinear form $\llbracket st, uv \rrbracket = \chi_{s,t}BL^\dagger B^*\chi_{u,v}$ acts like a “representation” of G , hence the custom bracket notation.

1.5.3 Representation

In order to compute $\llbracket st, uv \rrbracket$ (for all $s, t \in V$ and all $v \sim u$) at u , it suffices that u stores the vector $\varphi^{[w]} := L^\dagger \chi_w$, for all $w \in \{w : w \sim u\} \cup \{u\}$. This is apparent from writing

$$(1.5.4) \quad \llbracket st, uv \rrbracket = (\chi_u - \chi_v)L^\dagger(\chi_s - \chi_t) = (\varphi^{[u]} - \varphi^{[v]})^*(\chi_s - \chi_t),$$

where we have (crucially) used the fact that L^\dagger is symmetric. The vectors $\varphi^{[w]}$ stored at u comprise the (*routing*) *table* of u , which consists of $\deg(u) \cdot n$ real numbers. Thus the per-vertex table sizes of our scheme grow linearly with the vertex degree – a property we call *fair* representation. It seems that fair representation is key for routing in heterogeneous systems consisting of devices with varying capabilities.

Equation (1.5.4), written as $\llbracket st, uv \rrbracket = (\chi_s - \chi_t)^*(\varphi^{[u]} - \varphi^{[v]})$, shows that in order to compute $\llbracket st, uv \rrbracket$ at u , it suffices to know the *indices* of s and t (in the $\varphi^{[w]}$'s). These indices could be represented by $O(\ln n)$ -bit opaque vertex ID's and could be carried in the message headers. Routing schemes that support opaque vertex addressing are called *name-independent*. Name independence allows for vertex name persistence across time (i.e. changing graph topology) and across multiple co-existing routing schemes.

1.5.4 Computation

We use an idealized computational model to facilitate this exposition. The vertices of G are viewed as processors, synchronized by a global step counter. During a time step, pairs of processors can exchange messages of arbitrary (finite) size as long as they are connected by an edge. We describe an algorithm for computing approximations $\tilde{\varphi}^{[v]}$ to all $\varphi^{[v]}$ in $O(\ln n/\lambda)$ steps, where λ is the Fiedler eigenvalue of G (the smallest non-zero eigenvalue of L). If G is an expander, then $\lambda = O(1)$. At every step the algorithm sends messages consisting of $O(n)$ real numbers across every edge and performs $O(\deg(v) \cdot n)$ arithmetic operations on each processor v . Using standard techniques, this algorithm can be converted into a relatively easy-to-implement asynchronous one. (We omit this detail from here.) It is assumed that no graph changes occur during the computation of vertex tables.

A vector $\zeta \in \mathbf{R}^V$ is *distributed* if ζ_v is stored at v , for all v . A matrix $M \in \mathbf{R}^{V \times V}$ is *local* (with respect to G) if $M_{u,v} \neq 0$ implies $u \sim v$ or $u = v$. It is straightforward that if ζ is distributed and M is local, then $M\zeta$ can be computed in a single step, resulting in a new distributed vector. Extending this technique shows that for any polynomial $q(\cdot)$, the vector $q(M)\zeta$ can be computed in $\deg(q)$ steps.

The Power Method gives us a matrix polynomial $q(\cdot)$ of degree $O(\ln n/\lambda)$ such that $q(L)$ is a “good” approximation of L^\dagger . We compute the distributed vectors $\zeta^{[w]} := q(L)\chi_w$, for all w , in parallel. As a result, each vertex u obtains $\tilde{\varphi}^{[u]} = (\zeta_u^{[1]}, \dots, \zeta_u^{[n]})$, which approximates $\varphi^{[u]}$ according to Theorem 1.5.3 and the symmetry of L . In one last step, every processor u sends $\tilde{\varphi}^{[u]}$ to its neighbors. The approximation error n^{-5} is chosen to suffice (in accordance with Corollary 1.5.14) as discussed next.

Theorem 1.5.3. *Let λ be the Fiedler (smallest non-zero) eigenvalue of G 's Laplacian L , and let G be of bounded degree d_{\max} . Then $\|\zeta^{[v]} - \varphi^{[v]}\|_2 \leq n^{-5}$, where $\zeta^{[v]} = (2d_{\max})^{-1} \sum_{\omega=0}^k M^\omega \chi_v$ and $M = I - L/2d_{\max}$, as long as $k \geq \Omega(\lambda^{-1} \cdot \ln n)$.*

Theorem 1.5.3, which analyses the straightforward Power Method approach, implicitly assumes bounded degree in that all vertices must know an upper bound on d_{\max} in order to apply M from Theorem 1.5.3. Using a generous bound, anything $\omega(1)$, on d_{\max} is bad because it slows down the mixing of the power polynomial. To avoid this complication, one must use a symmetrization trick, explained in Section 1.5.15.

1.5.5 Robustness and latency

In order to get a handle on the analysis of routing in an ever-changing network we use a simplifying assumption: the graph does not change during the computation phase while it can change afterwards, during the routing phase. This assumption is informally justified because the computation phase in expander graphs (which we consider to be the typical case) is relatively fast, it takes $O(\ln n)$ steps. The routing phase, on the other hand, should be as “long” as possible before we have to recompute the scheme. Roughly, a routing scheme can be used until the graph changes so much from its shape when the scheme was computed that both the probability of reaching destinations and the congestion properties of the scheme deteriorate with respect to the new shape of the graph. We quantify the robustness of electric routing against edge removals in the following two theorems:

Theorem 1.5.4. *Let G be an unweighted graph with Fiedler eigenvalue $\lambda = \Theta(1)$ and maximum degree d_{\max} , and let $f^{[s,t]}$ denote the unit electric flow between s and t . For any $0 < p \leq 1$, let $Q_p = \{e \in E : |f_e^{[s,t]}| \geq p\}$ be the set of edges carrying more than p flow. Then, $|Q_p| \leq \min\{2/(\lambda p^2), 2d_{\max}\|L^\dagger\|_{1 \rightarrow 1}/p\}$.*

Note that part one of this theorem, i.e. $|Q_p| \leq 2/(\lambda p^2)$, distinguishes electric routing from simple schemes like shortest-path routing. The next theorem studies how edge removals affect demands when “the entire graph is in use:”

Theorem 1.5.5. *Let graph G be unweighted of bounded degree d_{\max} and vertex expansion α . Let f be a routing of the uniform multi-commodity demand set over V (single unit of demand between every pair of vertices), produced by an η -competitive oblivious routing scheme. Then, for any $0 \leq x \leq 1$, removing a x -fraction of edges from G removes at most $x \cdot (\eta \cdot d_{\max} \cdot \ln n \cdot \alpha^{-1})$ -fraction of flow from f .*

The expected number of edges traversed between source and sink reflects the latency of a routing. We establish (Proven in Section 1.5.10):

Theorem 1.5.6. *The latency of every electric walk on an undirected graph of bounded degree d_{\max} and vertex expansion α is at most $O(\min\{m^{1/2}, d_{\max}\alpha^{-2} \ln n\})$.*

1.5.6 Analysis

The main hurdle is Theorem 1.5.1, which we attack in two steps. First, we show that any linear routing scheme R (i.e. scheme for which the operator $R : \mathbf{R}^V \rightarrow \mathbf{R}^E$ is linear) has a distinct worst-case demand set, known as *uniform demands*, consisting of a unit demand between the endpoints of every edge of G . Combining this with the formulaic expression for electric flow (1.5.3) gives us an operator-based geometric bound for $\eta_{\mathcal{E}}$, which in the case of a bounded degree graph is simply $\eta_{\mathcal{E}} \leq O(\|L^\dagger\|_{1 \rightarrow 1})$ where the operator norm $\|\cdot\|_{1 \rightarrow 1}$ is defined by $\|A\|_{1 \rightarrow 1} := \sup_{x \neq 0} \|Ax\|_1 / \|x\|_1$. This is shown in Theorem 1.5.7. Second, we give a rounding-type argument that establishes the desired bound on $\|L^\dagger\|_{1 \rightarrow 1}$. This argument relies on a novel technique we dub *concurrent flow cutting* and is our key technical contribution. This is done in Theorem 1.5.10. This concludes the analysis of the congestion properties of electric flow.

The computational procedure for the vertex potentials $\varphi^{[v]}$'s (above) only affords us approximate versions $\tilde{\varphi}^{[v]}$ with ℓ_2 error guarantees. We need to ensure that, when using these in place of the exact ones, all properties of the exact electric flow are preserved. For this purpose, it is convenient to view the electric flow as a distribution over paths (i.e. the electric walk, defined above) and measure the total variation distance between the walks induced by exact and approximate vertex potentials. This is achieved in Theorem 1.5.13 and Corollary 1.5.14. It is then easy to verify that any two multi-commodity flows, whose respective individual flows have sufficiently small variation distance, have essentially identical congestion and robustness properties.

Organization

We begin our analysis, in Section 1.5.7, with the proof of Theorem 1.5.7 which is the entry point for most other arguments. Following this, in Section 1.5.8, we prove the general bound on the competitive performance of electric routing, which builds on Theorem 1.5.7.

In Section 1.5.9, we develop the self-contained, main technical result which bounds the $\|\cdot\|_{1 \rightarrow 1}$ norm of the graph Laplacian. Further, combining with the result of Section 1.5.7, this produces the conductance-dependent bound on the competitive ratio, Theorem 1.5.1.

Next, in Section 1.5.10, we prove the latency-related Theorem 1.5.6 which rests on the results of Sections 1.5.7 and 1.5.9.

We then switch gears to define the notion of “electric walk” in Section 1.5.11, which gives us a tool for analyzing imperfect electric flows resulting from perturbed potentials or missing edges. In Section 1.5.12, using electric walks we prove Theorem 1.5.13 that quantifies the changes in electric flows resulting from perturbed vertex potentials. This quantification is then used to assert the feasibility of electric routing using approximate computations.

Section 1.5.13 proves Theorems 1.5.4 and 1.5.5, quantifying decrease in electric flow in the presence of missing edges. These proofs build on tools developed in Sections 1.5.9 and 1.5.11.

Sections 1.5.14 and 1.5.15 discuss algorithm designs. The former presents the baseline algorithm for computing electric potentials, whose runtime is sensitive to the maximum degree. The latter improves this algorithm to a “symmetrized” one that does not depend on the maximum

degree.

We conclude with some open questions in Section 2.8.

1.5.7 Worst-case Demands Theorem

Recall that given a multi-commodity demand, electric routing assigns to each demand the corresponding electric flow in G , which we express (1.5.3) in operator form $\mathcal{E}(\oplus_\tau d_\tau) := WBL^\dagger(\oplus_\tau d_\tau)$. Electric routing is oblivious, since $\mathcal{E}(\oplus_\tau d_\tau) = \oplus_\tau \mathcal{E}(d_\tau)$ ensures that individual demands are routed independently from each other. The first key step in our analysis, Theorem 1.5.7, entails bounding $\eta_{\mathcal{E}}$ by the $\|\cdot\|_{1 \rightarrow 1}$ matrix norm of a certain natural graph operator on G . This step hinges on the observation that all linear routing schemes have an easy-to-express worst-case demand set:

Theorem 1.5.7. *For every undirected, weighted graph G , let $\Pi = W^{1/2}BL^\dagger B^*W^{1/2}$, then*

$$(1.5.5) \quad \eta_{\mathcal{E}} \leq \|W^{1/2}\Pi W^{-1/2}\|_{1 \rightarrow 1}.$$

Proof of Theorem 1.5.7. It is sufficient to consider demand sets that can be routed in G with unit congestion, since both electric and optimal routing scale linearly with scaling the entire demand set. Let $\oplus_\tau d_\tau$ be any demand set, which can be (optimally) routed in G with unit congestion via the multi-commodity flow $\oplus_\tau f_\tau$. Thus, $d_\tau = \sum_e f_{\tau,e} B_e$, for all τ .

The proof involves two steps:

$$\|\mathcal{E}(\oplus_\tau d_\tau)\|_{\text{cost}} \stackrel{(i)}{\leq} \|\mathcal{E}(\oplus_e w_e B_e)\|_{\text{cost}} \stackrel{(ii)}{=} \|W^{1/2}\Pi W^{-1/2}\|_{1 \rightarrow 1}$$

Step (i) shows that congestion incurred when routing $\oplus_\tau d_\tau$ is no more than that incurred when routing G 's edges, viewed as demands, through G :

$$\begin{aligned} (i) \quad \|\mathcal{E}(\oplus_\tau d_\tau)\|_{\text{cost}} &= \|\oplus_\tau \mathcal{E}(d_\tau)\|_{\text{cost}} && \\ &= \|\oplus_\tau \mathcal{E}\left(\sum_e f_{\tau,e} B_e\right)\|_{\text{cost}} && \text{use } d_\tau = \sum_e f_{\tau,e} B_e \\ &= \|\oplus_\tau \sum_e \mathcal{E}(f_{\tau,e} B_e)\|_{\text{cost}} && \text{use } \mathcal{E}\left(\sum_j d_j\right) = \sum_j \mathcal{E}(d_j) \\ &\leq \|\oplus_{\tau,e} \mathcal{E}(f_{\tau,e} B_e)\|_{\text{cost}} && \text{use } \left\|\sum_j f_j\right\|_{\text{cost}} \leq \|\oplus_j f_j\|_{\text{cost}} \\ &= \|\oplus_e \mathcal{E}\left(\sum_\tau |f_{\tau,e}| B_e\right)\|_{\text{cost}} && \text{use } \|\oplus_j \alpha_j f\|_{\text{cost}} = \left\|\sum_j |\alpha_j| f\right\|_{\text{cost}} \\ &\leq \|\oplus_e \mathcal{E}(w_e B_e)\|_{\text{cost}} && \text{use } \sum_\tau |f_{\tau,e}| \leq w_e \\ &= \|\mathcal{E}(\oplus_e w_e B_e)\|_{\text{cost}} \end{aligned}$$

$$\begin{aligned}
\text{(ii)} \quad \|\mathcal{E}(\oplus_e w_e B_e)\|_{\text{cost}} &\stackrel{(1.5.1)}{=} \|\mathcal{E}(\oplus_e w_e B_e)^* W^{-1}\|_{1 \rightarrow 1} \\
&\stackrel{(1.5.3)}{=} \|W B L^\dagger B^* W W^{-1}\|_{1 \rightarrow 1} = \|W^{1/2} \Pi W^{-1/2}\|_{1 \rightarrow 1}. \quad \square
\end{aligned}$$

Remark 1.5.8. Note that the proof of step (i) uses only the linearity of \mathcal{E} and so it holds for any linear routing scheme R , i.e. one has $\|R(\oplus_\tau d_\tau)\|_G \leq \|R(\oplus_e w_e B_e)\|_G$.

Using Theorem 1.5.7, the unconditional upper bound in Theorem 1.5.2 is simply a consequence of basic norm inequalities. This argument is given in Section 1.5.8. Theorem 1.5.1 provides a much stronger bound on $\eta_{\mathcal{E}}$ when the underlying graph has high vertex expansion. The lower bound in Theorem 1.5.1 is due to Hajiaghayi, et al. [52]. They show that every oblivious routing scheme is bound to incur congestion of at least $\Omega(\ln n / \ln \ln n)$ on a certain family of expander graphs. The upper bound in Theorem 1.5.1 follows from Theorem 1.5.7, Theorem 1.5.10 and using that $\|\Pi\|_{1 \rightarrow 1} = O(\|L^\dagger\|_{1 \rightarrow 1})$ for unweighted bounded-degree graphs. Thus in the next section we derive a bound on $\|L^\dagger\|_{1 \rightarrow 1}$ in terms of vertex expansion.

1.5.8 Unconditional Performance Bound (Proof of Theorem 1.5.2)

Proof of Theorem 1.5.2. The upper bound follows from:

$$(1.5.6) \quad \eta \stackrel{(1.5.5)}{\leq} \|\Pi\|_{1 \rightarrow 1} \stackrel{(1.5.7)}{\leq} m^{1/2} \cdot \|\Pi\|_{2 \rightarrow 2} \stackrel{(1.5.8)}{=} m^{1/2}$$

The second step is justified as follows

$$(1.5.7) \quad \|\Pi\|_{1 \rightarrow 1} = \max_e \|\Pi \chi_e\|_1 \leq m^{1/2} \cdot \max_e \|\Pi \chi_e\|_2 \leq m^{1/2} \cdot \|\Pi\|_{2 \rightarrow 2}.$$

The third step is the assertion

$$(1.5.8) \quad \|\Pi\|_{2 \rightarrow 2} \leq 1,$$

which follows from the (easy) fact that Π is a projection, shown by Spielman, et al. in Lemma 1.5.9.

The lower bound is achieved by a graph obtained by gluing the endpoints of \sqrt{n} copies of a path of length \sqrt{n} and a single edge. Routing a flow of value \sqrt{n} between these endpoints incurs congestion $\sqrt{n}/2$. \square

Lemma 1.5.9 (Proven in [100]). *Π is a projection; $\text{Im}(\Pi) = \text{Im}(W^{1/2}B)$; The eigenvalues of Π are 1 with multiplicity $n - 1$ and 0 with multiplicity $m - n + 1$; and $\Pi_{e,e} = \|\Pi \chi_e\|^2$.*

1.5.9 Laplacian ℓ_1 operator inequalities

The main results here are an upper and lower bound on $\|L^\dagger\|_{1 \rightarrow 1}$, which match for bounded-degree expander graphs. In this section, we present vertex expansion versions of these bounds that assume bounded-degree.

Theorem 1.5.10. *Let graph $G = (V, E)$ be unweighted, of bounded degree d_{\max} , and vertex expansion*

$$(1.5.9) \quad \alpha = \min_{S \subseteq V} \frac{|E(S, S^c)|}{\min\{|S|, |S^c|\}}, \quad \text{then} \quad \|L^\dagger\|_{1 \rightarrow 1} \leq \left(4 \ln \frac{n}{2}\right) \cdot \left(\alpha \ln \frac{2d_{\max}}{2d_{\max} - \alpha}\right)^{-1}.$$

The proof of this theorem (given in the next Section) boils down to a structural decomposition of unit (s, t) -electric flows in a graph (not necessarily an expander). We believe that this decomposition is of independent interest. In the case of bounded-degree expanders, one can informally say that the electric walk corresponding to the electric flow between s and t takes every path with probability exponentially inversely proportional to its length. We complement Theorem 1.5.10 with a lower bound on $\|L^\dagger\|_{1 \rightarrow 1}$:

Theorem 1.5.11. *Let graph $G = (V, E)$ be unweighted, of bounded degree d_{\max} , with metric diameter D . Then, $\|L^\dagger\|_{1 \rightarrow 1} \geq 2D/d_{\max}$ and, in particular, $\|L^\dagger\|_{1 \rightarrow 1} \geq (2 \ln n) \cdot (d_{\max} \ln d_{\max})^{-1}$ for all bounded-degree, unweighted graphs with vertex expansion $\alpha = O(1)$.*

Proof of Theorem 1.5.11. Let $s \neq t$ be a pair of vertices in G at distance D . We consider the flow $f = BL^\dagger(\chi_s - \chi_t)$. Set $\psi = L^\dagger(\chi_s - \chi_t)$, and note that we can use $\|f\|_1$ as a lower bound on $\|L^\dagger\|_{1 \rightarrow 1}$,

$$\|f\|_1 = \sum_{(u,v)} |\psi_u - \psi_v| \leq d_{\max} \sum_v |\psi_v| \leq d_{\max} \|L^\dagger(\chi_s - \chi_t)\|_1 \leq \frac{d_{\max}}{2} \|L^\dagger\|_{1 \rightarrow 1}.$$

Now, let $\{\pi_i\}_i$ be a path decomposition of f and let $l(\pi_i)$ and $f(\pi_i)$ denote the length and value, respectively, of π_i . Then,

$$\|f\|_1 = \sum_{(u,v)} |\psi_u - \psi_v| = \sum_i l(\pi_i) f(\pi_i) \geq D \sum_i f(\pi_i) = D. \quad \square$$

Proof of upper bound on $\|L^\dagger\|_{1 \rightarrow 1}$ for expanders

Proof of Theorem 1.5.10. Reformulation: We start by transforming the problem in a more manageable form,

$$(1.5.10) \quad \|L^\dagger\|_{1 \rightarrow 1} := \sup_{y \neq 0} \frac{\|L^\dagger y\|_1}{\|y\|_1} = \max_w \|L^\dagger \chi_w\|_1 \stackrel{(*)}{\leq} \frac{n-1}{n} \max_{s \neq t} \|L^\dagger(\chi_s - \chi_t)\|_1,$$

where the latter inequality comes from

$$\|L^\dagger \chi_s\|_1 = \|L^\dagger \pi_{\perp 1} \chi_s\|_1 = \|n^{-1} \sum_{t \neq s} L^\dagger (\chi_s - \chi_t)\|_1 \leq \frac{n-1}{n} \max_t \|L^\dagger (\chi_s - \chi_t)\|_1.$$

Pick any vertices $s \neq t$ and set $\psi = L^\dagger (\chi_s - \chi_t)$. In light of (1.5.10) our goal is to bound $\|\psi\|_1$. We think of ψ as the vertex potentials corresponding to electric flow with imbalance $\chi_s - \chi_t$. By an easy perturbation argument we can assume that no two vertex potentials coincide.

Index the vertices in $[n]$ by increasing potential as $\psi_1 < \dots < \psi_n$. Further, assume that n is even and choose a median c_0 so that $\psi_1 < \dots < \psi_{n/2} < c_0 < \psi_{n/2+1} < \dots < \psi_n$. (If n is odd, then set c_0 to equal the potential of the middle vertex.)

We aim to upper bound $\|\psi\|_1$, given as $\|\psi\|_1 = \sum_v |\psi_v| = \sum_{v: \psi_v > 0} \psi_v - \sum_{u: \psi_u < 0} \psi_u$. Using that $\sum_w \psi_w = 0$, we get $\|\psi\|_1 = 2 \sum_{v: \psi_v > 0} \psi_v = -2 \sum_{u: \psi_u < 0} \psi_u$.

Assume, without loss of generality, that $0 < c_0$, in which case

$$(1.5.11) \quad \|\psi\|_1 = -2 \sum_{u: \psi_u < 0} \psi_u \leq 2 \sum_{i=1}^{n/2} |\psi_i - c_0| =: 2N$$

In what follows we aim to upper-bound N .

Flow cutting: Define a collection of cuts (S_i, S_i^c) of the form $S_i = \{v : \psi_v \leq c_i\}$, for integers $i \geq 0$, where S_i will be the smaller side of the cut by construction. Let k_i be the number of edges cut by (S_i, S_i^c) and p_{ij} be the length of the j^{th} edge across the same cut. The cut points c_i , for $i \geq 1$, are defined according to $c_i = c_{i-1} - \Delta_{i-1}$, where $\Delta_{i-1} := 2 \sum_j \frac{p_{i-1,j}}{k_{i-1}}$. The last cut, (S_{r+1}, S_{r+1}^c) , is the first cut in the sequence c_0, c_1, \dots, c_{r+1} with $k_{r+1} = 0$ or, equivalently, $S_{r+1} = \emptyset$.

Bound on number of cuts: Let $n_i = |S_i|$. The isoperimetric inequality for vertex expansion (1.5.9) applied to (S_i, S_i^c) and the fact that $n_i \leq n/2$, by construction, imply

$$(1.5.12) \quad \frac{k_i}{n_i} \geq \alpha.$$

Let l_i be the number of edges crossing (S_i, S_i^c) that do not extend across c_{i+1} , i.e. edges that are not adjacent to S_{i+1} . The choice $\Delta_i := 2 \sum_j p_{ij}/k_i$ ensures that $l_i \geq k_i/2$. These edges are supported on at least l_i/d_{\max} vertices in S_i , and therefore $n_{i+1} \leq n_i - l_i/d_{\max}$. Thus,

$$(1.5.13) \quad n_{i+1} \leq n_i - \frac{l_i}{d_{\max}} \leq n_i - \frac{k_i}{2d_{\max}} \stackrel{(1.5.12)}{\leq} n_i - \frac{\alpha n_i}{2d_{\max}} = n_i \left(1 - \frac{\alpha}{2d_{\max}}\right),$$

Combining inequality (1.5.13) with $n_0 = n/2$, we get

$$(1.5.14) \quad n_i \leq \frac{n}{2} \left(1 - \frac{\alpha}{2d_{\max}}\right)^i$$

The stopping condition implies $S_r \neq \emptyset$, or $n_r \geq 1$, and together with (1.5.14) this results in

$$(1.5.15) \quad r \leq \log_{1/\theta} \frac{n}{2}, \text{ with } \theta = 1 - \frac{\alpha}{2d_{\max}}.$$

Amortization argument: Continuing from (1.5.11),

$$(1.5.16) \quad N = \sum_{i=1}^{n/2} |\psi_i - c_0| \stackrel{(*)}{\leq} \sum_{i=0}^r (n_i - n_{i+1}) \sum_{j=0}^i \Delta_j,$$

where (*) follows from the fact that for every vertex $v \in S_i - S_{i+1}$ we can write $|\psi_v - c_0| \leq \sum_{j=0}^i \Delta_j$.

Because $BL^\dagger(\chi_s - \chi_t)$ is a unit flow, we have the crucial (and easy to verify) property that, for all i , $\sum_j p_{ij} = 1$. In other words, the total flow of “simultaneous” edges is 1. So,

$$(1.5.17) \quad \Delta_i = 2 \sum_j \frac{p_{ij}}{k_i} = \frac{2}{k_i} \stackrel{(1.5.12)}{\leq} \frac{2}{\alpha n_i}$$

Now we can use this bound on Δ_j in (1.5.16),

$$\sum_{i=0}^r (n_i - n_{i+1}) \sum_{j=0}^i \Delta_j \stackrel{(*)}{=} \sum_{i=1}^r n_i \Delta_i \leq \frac{2}{\alpha} \sum_{i=0}^r 1 = \frac{2}{\alpha} (r+1),$$

where to derive (*) we use $n_{r+1} = 0$. Combining the above inequality with (1.5.15) concludes the proof. \square

1.5.10 Latency (Proof of Theorem 1.5.6)

Proof of Theorem 1.5.6. Let $X_e^{[s,t]}$ be the indicator that edge e participates in the electric walk between s and t . Then the latency can be expressed as

$$\begin{aligned} \max_{s \neq t} \sum_e \mathbf{E} X_e^{[s,t]} &= \max_{s \neq t} \sum_{(u,v)} |(\delta_u - \delta_v) L^\dagger (\delta_s - \delta_t)| \\ &= \max_{s \neq t} \|BL^\dagger (\delta_s - \delta_t)\|_1 = \|BL^\dagger B^*\|_{1 \rightarrow 1} = \|\Pi\|_{1 \rightarrow 1}. \end{aligned}$$

The latter is bounded by Theorem 1.5.10 and $\|\Pi\|_{1 \rightarrow 1} \leq m^{1/2}$, as in (1.5.6) e.g. □

Remark 1.5.12. For expanders, this theorem is not trivial. In fact, there exist path realizations of the electric walk which can traverse up to $O(n)$ edges. Theorem 1.5.6 asserts that this happens with small probability. On the other hand, in a bounded-degree expander, even if s and t are adjacent the walk will still take a $O(\log n)$ -length path with constant probability.

1.5.11 Electric walk

To every unit flow $f \in \mathbf{R}^E$, not necessarily electric, we associate a random walk $W = W_0, W_1, \dots$ called the *flow walk*, defined as follows. Let $\sigma := B^* f$ and so $\sum_v \sigma_v = 0$. The walk starts at W_0 , with

$$\Pr\{W_0 = v\} = \frac{2 \cdot \max\{0, \sigma_v\}}{\sum_w |\sigma_w|} = \frac{\sum_w f_{v \rightarrow w} - \sum_w f_{w \rightarrow v}}{\sum_u (\sum_w f_{u \rightarrow w} - \sum_w f_{w \rightarrow u})}$$

If the walk is currently at W_t , the next vertex is chosen according to $\Pr\{W_{t+1} = v \mid W_t = u\} = \frac{f_{u \rightarrow v}}{\sum_w f_{u \rightarrow w}}$, where

$$(1.5.18) \quad f_{u \rightarrow v} = \begin{cases} |f_{(u,v)}|, & (u, v) \in E \text{ and } f \text{ flows from } u \text{ to } v \\ 0, & \text{otherwise.} \end{cases}$$

When the underlying flow f is an electric flow, i.e. when $f = \mathcal{E}(B^* f)$, the flow walk deserves the specialized name *electric walk*. We study two aspects of electric walks here: (i) stability against perturbations of the vertex potentials, and (ii) robustness against edge removal.

1.5.12 Electric routing with approximate potentials

The set of vertex potential vectors $\varphi^{[v]} = L^\dagger \chi_v$, for all $v \in V$, encodes all electric flows, as argued in (1.5.4). In an algorithmic setting, only approximations $\tilde{\varphi}^{[v]}$ of these vectors are available. We ensure that when these approximations are sufficiently good in an ℓ_2 sense, the path probabilities (and congestion properties) of electric walks are virtually unchanged.

Theorem 1.5.13. *Let $\tilde{\varphi}^{[v]}$ be an approximation of $\varphi^{[v]}$, for all $v \in G$, in the sense that*

$$(1.5.19) \quad \|\varphi^{[v]} - \tilde{\varphi}^{[v]}\|_2 \leq \nu, \text{ for all } v \in V, \text{ with } \nu = n^{-A},$$

where $A > 4$ is a constant. Then for every electric walk, defined by vertex potentials $\varphi = \sum_v \alpha_v \varphi^{[v]}$, the corresponding ‘‘approximate’’ walk, defined by vertex potentials $\tilde{\varphi} = \sum_v \alpha_v \tilde{\varphi}^{[v]}$, induces a distribution over paths γ with

$$(1.5.20) \quad \sum_{\gamma} \left| \Pr_{\varphi}\{W = \gamma\} - \Pr_{\tilde{\varphi}}\{W = \gamma\} \right| \leq O(n^{2-\frac{A}{2}}),$$

where γ ranges over all paths in G , and $\Pr_{\varphi}\{W = \gamma\}$ denotes the probability of γ under φ (respectively for $\Pr_{\tilde{\varphi}}\{W = \gamma\}$).

As shown in Theorem 1.5.16, the Power Method affords us any sufficiently large exponent A , say $A = 5$, without sacrificing efficiency in terms of distributed computation time. In this case, the following corollary asserts that routing with approximate potentials preserves both the congestion properties of the exact electrical flow as well as the probability of reaching the sink.

Corollary 1.5.14. *Under the assumptions of Theorem 1.5.13 and $A = 5$, the electric walk defined by vertex potentials $\tilde{\varphi} = \tilde{\varphi}^{[s]} - \tilde{\varphi}^{[t]}$ reaches t with probability $1 - o_n(1)$. Furthermore, for every edge (u, v) with non-negligible load, i.e. $|\tilde{\varphi}_u - \tilde{\varphi}_v| = \omega(n^{-2})$, we have $|\tilde{\varphi}_u - \tilde{\varphi}_v| \rightarrow_n |\varphi_u - \varphi_v|$, where $\varphi = \varphi^{[s]} - \varphi^{[t]}$.*

Proof of Theorem 1.5.13

Proof of Theorem 1.5.13. Notation: Note that in this proof we use the notation of (1.5.18). So, for a potential vector ψ , we have $\psi_{u \rightarrow v} := \psi_u - \psi_v$ if (u, v) is an edge and $\psi_u \geq \psi_v$, and $\psi_{u \rightarrow v} := 0$ otherwise. So, for example, the potential difference on (u, v) can be written as $\psi_{u \rightarrow v} + \psi_{v \rightarrow u}$. On the other hand, we use the single letter edge notation φ_e to denote the signed (according to B) potential difference on e , so $\varphi_e := \varphi_u - \varphi_v$ if $B_e = \delta_u - \delta_v$. Let D be the maximum degree.

Edge approximation: Fix any unit electric flow, defined by potentials $\varphi := \sum_v \alpha_v \varphi^{[v]}$, and write its approximation as $\tilde{\varphi} := \sum_v \alpha_v \tilde{\varphi}^{[v]}$. All unit flows can be so expressed under the restriction that $\sum_v \alpha_v = 0$ and $\sum_v |\alpha_v| = 2$. The approximation condition (1.5.19) combined with

Lemma 1.5.15 gives us, for every edge $e = (u, v)$,

$$\begin{aligned}
|\varphi_e - \tilde{\varphi}_e| &= \left| \sum_v \alpha_v (\varphi_e^{[v]} - \tilde{\varphi}_e^{[v]}) \right| \\
&\leq \sum_v |\alpha_v| \cdot |\varphi_e^{[v]} - \tilde{\varphi}_e^{[v]}| \\
&\leq \sum_v |\alpha_v| \cdot 2\nu && \text{apply Lemma 1.5.15} \\
&= 4\nu
\end{aligned}$$

We call this the *additive edge approximation condition*

$$(1.5.21) \quad \varphi_e - 4\nu \leq \tilde{\varphi}_e \leq \varphi_e + 4\nu$$

Now, consider a fixed path γ along the electric flow defined by φ , traversing vertices w_0, w_1, \dots, w_k . Let $\mathbf{Pr}_\varphi\{W = \gamma\}$ and $\mathbf{Pr}_{\tilde{\varphi}}\{W = \gamma\}$ denote the probability of this path under the potentials φ and $\tilde{\varphi}$, respectively. In most of what follows, we build machinery to relate one to the other.

Path probabilities: For a general unit flow (not necessarily an (s, t) -flow), defined by vertex potentials ψ , $\mathbf{Pr}_\psi\{W = \gamma\}$ equals

$$(1.5.22) \quad \mathbf{Pr}_\psi\{W_0 = w_0\} \left(\prod_{i=0}^{k-1} \mathbf{Pr}_\psi\{W_{i+1} = w_{i+1} \mid W_i = w_i\} \right) \mathbf{Pr}_\psi\{W_\infty = w_k \mid w_k\},$$

where next we explain each factor in turn.

The first, $\mathbf{Pr}_\psi\{W_0 = w_0\}$, is the probability that the walk starts from w_0 , and is expressed as

$$(1.5.23) \quad \mathbf{Pr}_\psi\{W_0 = w_0\} = \max \left(0, \sum_u \psi_{w_0 \rightarrow u} - \sum_u \psi_{u \rightarrow w_0} \right).$$

The second and trickiest, $\mathbf{Pr}_\psi\{W_{i+1} = w_{i+1} \mid W_i = w_i\}$, is the probability that having reached w_i the walk traverses the edge leading to w_{i+1} , and $\sum_u \psi_{w_i \rightarrow u} \geq \sum_u \psi_{u \rightarrow w_i}$, we write

$$(1.5.24) \quad \mathbf{Pr}_\psi\{W_{i+1} = w_{i+1} \mid W_i = w_i\} = \frac{\psi_{w_i \rightarrow w_{i+1}}}{\max \left(\sum_u \psi_{u \rightarrow w_i}, \sum_u \psi_{w_i \rightarrow u} \right)}.$$

To grasp the meaning of the denominator, note that the quantity $|\sum_u \psi_{u \rightarrow w_i} - \sum_u \psi_{w_i \rightarrow u}|$ is the magnitude of the in or out flow (depending on the case) at w_i .

The third, $\mathbf{Pr}_\psi\{W_\infty = w_k \mid w_k\}$, is the probability that the walk ends (or exits) at w_k

conditioned on having reached w_k , and

$$(1.5.25) \quad \mathbf{Pr}_\psi\{W_\infty = w_k \mid w_k\} = \max\left(0, \sum_u \psi_{u \rightarrow w_k} - \sum_u \psi_{w_k \rightarrow u}\right).$$

Next, we are going to find multiplicative bounds for all three factors by focusing on “dominant” paths, and discarding ones with overall negligible probability.

Dominant paths: It is straightforward to verify (from first principles) that the probability that an edge (u, v) occurs in the electric walk equals $|\varphi_e| = \varphi_{u \rightarrow v} + \varphi_{v \rightarrow u}$. We call an edge *short* if $|\varphi_e| \leq \epsilon$, where the exact asymptotic of $\epsilon > 0$ is determined later, but for the moment $\nu \ll \epsilon \ll 1$. We restrict our attention to *dominant* paths γ that traverse no short edges, and have $\mathbf{Pr}_\varphi\{W_0 = w_0\} \geq \epsilon$ and $\mathbf{Pr}_\varphi\{W_\infty = w_k \mid w_k\} \geq \epsilon$.

Indeed, by a union bound, the probability that the electric walk traverses a non-dominant path is at most $2n\epsilon + n^2\epsilon$. This will be negligible and such paths will be of no interest. In summary,

$$(1.5.26) \quad \mathbf{Pr}_\varphi\{W \text{ dominant}\} \geq 1 - 2n\epsilon - n^2\epsilon$$

We now condition on the event that γ is dominant.

The no short edge condition gives $\epsilon \leq |\varphi_e| \leq 1$, and using (1.5.21) we derive the stronger *multiplicative edge approximation condition*

$$(1.5.27) \quad \frac{1}{\sigma} \leq \frac{\tilde{\varphi}_e}{\varphi_e} \leq \sigma, \text{ where } \sigma = 1 + \frac{8D\nu}{\epsilon},$$

which holds as long as $\epsilon \geq 4\nu$, as guaranteed by the asymptotics of ϵ . Also note that the latter condition ensures that φ_e and $\tilde{\varphi}_e$ have the same sign. An extra factor of $2D$ is included in σ with foresight.

For the first factor (1.5.23), we have

$$(1.5.28) \quad \begin{aligned} \mathbf{Pr}_{\tilde{\varphi}}\{W_0 = w_0\} &= \sum_u \tilde{\varphi}_{w_0 \rightarrow u} - \sum_u \tilde{\varphi}_{u \rightarrow w_0} \\ &\geq \sum_u \varphi_{w_0 \rightarrow u} - \sum_u \varphi_{u \rightarrow w_0} - 4D\nu && \text{use (1.5.21)} \\ &\geq \mathbf{Pr}_\varphi\{W_0 = w_0\} \left(1 - \frac{4D\nu}{\epsilon}\right) && \text{use } \mathbf{Pr}_\varphi\{W_0 = w_0\} \geq \epsilon \\ &\geq \frac{1}{\sigma} \mathbf{Pr}_\varphi\{W_0 = w_0\} && \text{use } \epsilon \leq 1/2. \end{aligned}$$

For the second factor (1.5.24), assume $\sum_u \psi_{u \rightarrow w_i} \geq \sum_u \psi_{w_i \rightarrow u}$. An identical argument holds in the other case. Abbreviate

$$\mathbf{Pr}_{\tilde{\varphi}}\{w_{i+1} \mid w_i\} := \mathbf{Pr}_{\tilde{\varphi}}\{W_{i+1} = w_{i+1} \mid W_i = w_i\}.$$

Path dominance implies $\sum_u \varphi_{u \rightarrow w_i} \geq \epsilon$, and so

$$\begin{aligned}
(1.5.29) \quad \mathbf{Pr}_{\tilde{\varphi}}\{w_{i+1} | w_i\} &= \frac{\tilde{\varphi}_{w_i \rightarrow w_{i+1}}}{\sum_u \tilde{\varphi}_{u \rightarrow w_i}} \\
&\geq \frac{\sigma^{-1} \cdot \varphi_{w_i \rightarrow w_{i+1}}}{\sum_u \varphi_{u \rightarrow w_i} + 4D\nu} && \text{use (1.5.27) and (1.5.21)} \\
&\geq \sigma^{-2} \frac{\varphi_{w_i \rightarrow w_{i+1}}}{\sum_u \varphi_{u \rightarrow w_i}} && \text{use } \epsilon \leq 1/2 \text{ and } \sum_u \varphi_{u \rightarrow w_i} \geq \epsilon \\
&= \frac{1}{\sigma^2} \mathbf{Pr}_{\varphi}\{w_{i+1} | w_i\}.
\end{aligned}$$

For the third factor (1.5.25), similarly to the first, we have

$$\begin{aligned}
(1.5.30) \quad \mathbf{Pr}_{\tilde{\varphi}}\{W_\infty = w_k | w_k\} &= \\
&= \sum_u \tilde{\varphi}_{u \rightarrow w_k} - \sum_u \tilde{\varphi}_{w_k \rightarrow u} \\
&\geq \sum_u \varphi_{u \rightarrow w_k} - \sum_u \varphi_{w_k \rightarrow u} - 4D\nu && \text{use (1.5.21)} \\
&\geq \mathbf{Pr}_{\varphi}\{W_\infty = w_k | w_k\} \left(1 - \frac{4D\nu}{\epsilon}\right) && \text{use } \mathbf{Pr}_{\varphi}\{W_\infty = w_k | w_k\} \geq \epsilon \\
&\geq \frac{1}{\sigma} \mathbf{Pr}_{\varphi}\{W_\infty = w_k | w_k\} && \text{use } \epsilon \leq 1/2.
\end{aligned}$$

Dominant path bound: We now obtain a relation between $\mathbf{Pr}_{\varphi}\{W = \gamma\}$ and $\mathbf{Pr}_{\tilde{\varphi}}\{W = \gamma\}$ by combining the bounds (1.5.28), (1.5.29) and (1.5.30) with (1.5.22):

$$\begin{aligned}
(1.5.31) \quad \frac{\mathbf{Pr}_{\tilde{\varphi}}\{W = \gamma\}}{\mathbf{Pr}_{\varphi}\{W = \gamma\}} &\geq \frac{1}{\sigma^{2n+2}} && \text{apply bounds, and path length } \leq n \\
&\geq \left(1 - \frac{8D\nu}{\epsilon}\right)^{2n+2} && \text{use } \sigma^{-1} \geq 1 - 8D\nu/\epsilon \\
&\geq \exp\left(-\frac{16D\nu}{\epsilon}\right)^{2n+2} && \text{use } 1 - x \geq e^{-2x} \\
&=: \theta
\end{aligned}$$

Statistical difference: Abbreviate $p(\gamma) := \mathbf{Pr}_{\varphi}\{W = \gamma\}$ and $q(\gamma) := \mathbf{Pr}_{\tilde{\varphi}}\{W = \gamma\}$. Below, γ iterates through all paths, ζ iterates through dominant paths and ξ iterates through non-dominant paths. We bound the statistical difference (1.5.20), using (1.5.31) which says $q(\zeta) \geq$

$\theta \cdot p(\zeta)$,

$$\begin{aligned}
& \sum_{\gamma} |p(\gamma) - q(\gamma)| = \\
& = \sum_{\zeta} |p(\zeta) - q(\zeta)| + \sum_{\xi} |p(\xi) - q(\xi)| \\
& \leq \sum_{\zeta} |(1-\theta)p(\zeta) - (q(\zeta) - \theta p(\zeta))| + \sum_{\xi} q(\xi) \quad \text{use } \theta < 1 \\
& \leq \sum_{\zeta} |q(\zeta) - \theta p(\zeta)| + \sum_{\xi} q(\xi) \\
& = 1 - \sum_{\zeta} p(\zeta) \\
(1.5.32) \quad & = (1-\theta) + \theta \sum_{\zeta} p(\zeta)
\end{aligned}$$

In this final step, we pin-point the asymptotics of ϵ that simultaneously minimize the two terms of (1.5.32). In the following, we parameterize $\epsilon = n^{-B}$ and use (1.5.26),

$$\begin{aligned}
(1-\theta) + \theta \sum_{\zeta} p(\zeta) & = \\
& = 1 - \exp\left(-\frac{16D\nu}{\epsilon}\right)^{2n+2} + \exp\left(-\frac{16D\nu}{\epsilon}\right)^{2n+2} (2n\epsilon + n^2\epsilon) \\
& = 1 - \exp O(-Dn^{B-A+1}) + n^{2-B} \cdot \exp O(-Dn^{B-A+1}) \\
& = O(Dn^{B-A+1}) + n^{2-B} \cdot \exp O(-Dn^{B-A+1}), \quad \text{use } 1 - e^{-x} \leq x \\
& = O(n^{B-A+2}) + O(n^{2-B}) \quad \text{use } D \leq n \\
& = O(n^{2-\frac{A}{2}}), \quad \text{set } B = A/2. \quad \square
\end{aligned}$$

Lemma 1.5.15. *If $x, y \in \ell_2$ and $\|x - y\|_2 \leq \nu$, then for all $i \neq j$,*

$$(x_i - x_j) - 2\nu \leq y_i - y_j \leq (x_i - x_j) + 2\nu.$$

Proof. We have $(x_i - y_i)^2 \leq \|x - y\|_2^2 \leq \nu^2$, implying $|x_i - y_i| \leq \nu$. Similarly for j . Combining the two proves the lemma. \square

1.5.13 Robustness (Proof of Theorems 1.5.4 and 1.5.5)

We prove Theorem 1.5.4 and 1.5.5 here. The proof of the former interestingly relies on the flow cutting techniques developed previously in Section 1.5.9.

Proof of Theorem 1.5.4. For the first part, let $\{(u_1, v_1), \dots, (u_k, v_k)\} = Q_p$ and let $p_i = |f_{(u_i, v_i)}^{[s, t]}| = |(\chi_{u_i} - \chi_{v_i})^* L^\dagger (\chi_s - \chi_t)|$. Consider the embedding $\zeta : V \rightarrow \mathbf{R}$, defined by $\zeta(v) = \chi_v^* L^\dagger (\delta_s - \delta_t)$. Assume for convenience that $\zeta(u_i) \leq \zeta(v_i)$ for all i . Let $\zeta_{\min} = \min_v \zeta(v)$ and $\zeta_{\max} = \max_v \zeta(v)$.

Choose c uniformly at random from $[\zeta_{\min}, \zeta_{\max}]$ and let $X_i = p_i \cdot \mathbf{I}\{\zeta(u_i) \leq c \leq \zeta(v_i)\}$, where $\mathbf{I}\{\cdot\}$ is the indicator function. Observe that the random variable $X = \sum_i X_i$ equals the total electric flow of all edges in Q_p cut by c . Since these edges are concurrent (in the electric flow) by construction, we have $X \leq 1$. On the other hand,

$$\mathbf{E}X = \sum_i p_i \cdot \Pr\{\zeta(u_i) \leq c \leq \zeta(v_i)\} \geq \sum_i p_i \frac{p_i}{\zeta_{\max} - \zeta_{\min}} \geq \sum_i \frac{\lambda p_i^2}{2} \geq k \frac{\lambda p^2}{2}$$

Combining this with $\mathbf{E}X \leq 1$ produces $|Q_p| \leq 2/(\lambda p^2)$.

For the second part, $k p \leq \sum_{e \in Q_p} |f_e^{[s, t]}| \leq \sum_{e \in E} |f_e^{[s, t]}| = \|BL^\dagger(\chi_s - \chi_t)\|_1 \leq 2d_{\max} \cdot \|L^\dagger\|_{1 \rightarrow 1}$. This gives $|Q_p| \leq 2d_{\max} \cdot \|L^\dagger\|_{1 \rightarrow 1}/p$. \square

Proof of Theorem 1.5.5. Let f_{opt} be a max-flow routing of the uniform demands and let θ be the fraction of the demand set that is routed by f_{opt} . The Multi-commodity Min-cut Max-flow Gap Theorem (Theorem 2, in [73]) asserts

$$O(\ln n) \cdot \theta \geq \min_{S \subset V} \frac{|E(S, S^c)|}{|S| \cdot |S^c|} \geq \frac{1}{n} \cdot \min_{S \subset V} \frac{|E(S, S^c)|}{\min\{|S|, |S^c|\}} = \frac{\alpha}{n}$$

Thus the total demand flown by f_{opt} is no less than $\theta \binom{n}{2} \geq \Omega(\alpha n / \ln n)$. Normalize f (by scaling) so it routes the same demands as f_{opt} . If k edges are removed, then at most ηk flow is removed from f , which is at most a fraction $\eta k \cdot O(\ln n / \alpha n)$ of the total flow. Substitute $x = k/m$ and use $m \leq d_{\max} n$ to complete the proof. \square

1.5.14 Power iterations (Proof of Theorem 1.5.3)

Theorem 1.5.3 is implied by the following theorem by specializing $\epsilon = O(n^{-5})$:

Theorem 1.5.16. *Let G be a graph, whose Laplacian L has smallest eigenvalue λ and whose maximum degree is D . Then, for every y with $\|y\|_2 = 1$ the vector $x = L^\dagger y$ can be approximated using*

$$\tilde{x} = \frac{1}{2D} \sum_{i=0}^d \left(I - \frac{L}{2D}\right)^i y,$$

so that for every $\epsilon > 0$,

$$\|x - \tilde{x}\|_2 \leq \epsilon, \text{ as long as } d \geq \Omega(1) \cdot \ln \frac{1}{\lambda \epsilon D} \cdot \left(\ln \frac{1}{1 - \lambda}\right)^{-1}.$$

Proof of Theorem 1.5.16. We normalize L via $N = L/\tau$ (and so $L^{-1} = N^{-1}/\tau$), where $\tau = 2D$. Since $\tau = 2D \geq \lambda_{\max}(L)$, the eigenvalues of N are in $[0, 1]$. In this case, the Moore-Penrose inverse of N is given by $N^\dagger = \sum_{i=0}^{\infty} (I - N)^i$. Set $N_0^\dagger = \sum_{i=0}^d (I - N)^i$ and $N_1^\dagger = N^\dagger - N_0^\dagger$. Our aim is to minimize d so that

$$\|x - \tilde{x}\|_2 = \left\| \frac{N_0^\dagger + N_1^\dagger}{\tau} y - \frac{N_0^\dagger}{\tau} y \right\|_2 = \left\| \frac{N_1^\dagger}{\tau} y \right\|_2 \leq \left\| \frac{N_1^\dagger}{\tau} \right\|_{2 \rightarrow 2} \leq \epsilon,$$

where $\|A\|_{2 \rightarrow 2} := \sup_{x \neq 0} \|Ax\|_2 / \|x\|_2$ denotes the matrix spectral norm. Set $\kappa := \tau / \lambda_{\min}$, so that κ^{-1} is the smallest eigenvalue of N ,

$$\begin{aligned} \|N_1^\dagger\|_{2 \rightarrow 2} &= \left\| \sum_{i=d+1}^{\infty} (I - N)^i \right\|_{2 \rightarrow 2} \leq \sum_{i=d+1}^{\infty} \|(I - N)^i\|_{2 \rightarrow 2} \\ (1.5.33) \qquad &\leq \sum_{i=d+1}^{\infty} (1 - \kappa^{-1})^i = (1 - \kappa^{-1})^{d+1} \kappa \end{aligned}$$

Setting (1.5.33) less than $\tau\epsilon$ gives

$$d \geq \frac{\ln \kappa / (\tau\epsilon)}{\ln \kappa / (\kappa - 1)}. \quad \square$$

1.5.15 Symmetrization of the electric flow algorithm

In this section we discuss how to modify the computational procedure, given in Section 1.5.4 and analyzed in Section 1.5.14, in order to apply it to graphs of unbounded degree. The described algorithm for computing $\varphi^{[w]} = L^\dagger \chi_w$ relies on the approximation of L^\dagger via the Taylor series $\frac{1}{1-x} = \sum_{i=0}^{\infty} (1-x)^i$. The series converges only when $\|x\|_2 < 1$, which is ensured by setting $x = \frac{L}{2d_{\max}}$, and using that $\|L\|_{2 \rightarrow 2} < 2d_{\max}$. Thus we arrive at $2d_{\max} \cdot L^\dagger = \sum_{i=0}^{\infty} (I - \frac{L}{2d_{\max}})^i$. This approach continues to work if we replace d_{\max} with any upper bound $h_{\max} \geq d_{\max}$, obtaining $L^\dagger = \frac{1}{2h_{\max}} \sum_{i=0}^{\infty} (I - M)^i$ where $M = \frac{L}{2h_{\max}}$, however this is done at the expense of slower convergence of the series. Since in a distributed setting all vertices must agree on what M is, a worst-case upper bound $h_{\max} = n$ must be used, which results in a prohibitively slow convergence even for expander graphs.

Instead, we pursue a different route. Let $\mathcal{L} = D^{-1/2} L D^{-1/2}$ be the *normalized Laplacian* of G , where $D \in \mathbf{R}^{n \times n}$ is diagonal with $D_{v,v} = \deg(v)$. One always has $\|\mathcal{L}\|_{2 \rightarrow 2} \leq 2$ (Lemma 1.7 in [23]) while at the same time $\lambda_{\min}(\mathcal{L}) \geq \max\{\frac{\beta^2}{2}, \frac{\alpha^2}{4d_{\max} + 2d_{\max}\alpha}\}$ (Theorems 2.2 and 2.6 in [23]), where α and β are the vertex- and edge-expansion of G , respectively. Set $M = \mathcal{L}/3$, so that $\|M\|_{2 \rightarrow 2} < 1$. Recall that the aim of our distributed procedure is to compute $\varphi_u^{[w]}$ at u (for all w). We achieve this using the following:

$$\varphi_u^{[w]} = \chi_u^* L^\dagger \chi_w = \chi_u^* D^{-1/2} \frac{M^\dagger}{3} D^{-1/2} \chi_w = \frac{\chi_u^*}{\sqrt{\deg(u)}} \frac{\sum_{i=0}^{\infty} (I - M)^i}{3} \frac{\chi_w}{\sqrt{\deg(w)}}$$

The key facts about the series in the left-hand side are that (i) it converges quickly when G is an expander and (ii) all vertices can compute M locally, in particular, without requiring any global knowledge like e.g. an upper bound on d_{\max} .

1.5.16 Open questions

We conclude our exposition on electric routing with a couple of open questions.

A central concern, widely-studied in social-networks, are Sybil Attacks [34]. In systems where new members can join via edges to “friend” nodes, Sybil attacks can be modeled as graph-theoretic noise, as defined in [58]. It is interesting to understand how such noise affects electric routing.

Another open problem concerns the space requirements at each vertex for oblivious routing. We suspect that any $O(\ln n)$ -competitive oblivious routing scheme, which outputs its routes in the “next hop” model, must maintain $\Omega(n)$ -size routing tables at every vertex. In the *next hop* model, every vertex v must be able to answer the question “What is the flow of the (s, t) -route in the neighborhood of v ?” in time $O(\text{polylog}(n))$, using its own routing table alone and for every source-sink pair (s, t) .

Chapter 2

Greedy embeddings

In this chapter we concern ourselves with representing routing schemes on trees. While a tree provides an obvious, unique, non-self-intersecting path between every pair of vertices, encoding all such paths in an efficient, distributed manner is not trivial.

Routing on a tree is a key primitive since a number of advanced routing schemes are presented in the form of convex combinations of trees. Bartal’s graph metric approximation via distributions over trees [10], for example, constitutes an ℓ_1 -routing scheme. Räcke’s cut-based graph decompositions in the form of convex combinations of trees [89] constitute an ℓ_∞ -routing scheme. And Thorup and Zwick’s distance oracles for graph metrics [104] constitute another ℓ_1 -routing scheme based on trees, which trades off performance guarantees for a smaller number of trees involved in the construction. In all of the above cases, the vertices of the network need to learn how to route with respect to a collection of graph subtrees, and consequently decide which single tree to use based on some additional rules.

A seminal work of Thorup and Zwick [104] achieves routing on trees with essentially optimal $(1 + o(1)) \log n$ bits per vertex representation. The schemes of [104] are combinatorially complex and quite brittle to changes in the underlying graph. For this reason, a parallel line of research has focused on finding conceptually simpler representations at the cost of small (typically a constant factor) inefficiencies in terms of bit complexity or objective guarantees. In this vein, “greedy routing” has become a prominent approach. The vertices of the tree are embedded into a “nice” metric so that greedy message forwarding with respect to that metric results in routing along desired paths. In this work we develop a number of greedy constructions for trees and provide respective lower bounds on representation complexity. Our key upper bound is an embedding of every tree into 3-dimensional hyperbolic space, which requires $O(\log n)$ -bit coordinates per vertex — a constant factor away from the optimal.

Formally, a *greedy embedding* of an unweighted undirected graph $G = (V, E)$ into a metric space (X, ρ) is a function $f : V \rightarrow X$ such that for every *source-sink* pair of different vertices $s, t \in V$ it is the case that s has a neighbor v in G with $\rho(f(v), f(t)) < \rho(f(s), f(t))$.

Finding greedy embeddings of connectivity graphs helps to build distributed routing schemes

with compact routing tables. In this chapter we take a refined look at greedy embeddings, previously addressed in [66, 82], by examining their description complexity as a key parameter in conjunction with their dimensionality. We give arguments showing that the dimensionality lower-bounds for monotone maps do not extend to greedy embeddings. We prove a *unified* $O(\log n)$ lower-bound on the dimension of no-stretch greedy embeddings when the host metric is Euclidean or Lobachevsky geometry. The essence of the lower bound entails showing that low-dimensional spaces lack the topological capacity to realize the embeddings of certain graphs with “hard crossroads.” This technique might be of independent interest. We develop new methods for building *concise* embeddings of trees (and some other graphs) in 3-dimensional Lobachevsky spaces using recursive applications of hyperbolic isometries guided by caterpillar-type decompositions. Our embeddings improve over prior work [66] by achieving $O(\kappa(T) \cdot \log n)$ description complexity, where $\kappa(T)$ is the caterpillar dimension. We further demonstrate concise $O(\log n)$ -dimensional greedy embeddings of trees into Euclidean space using techniques inspired by [49], thereby strengthening our belief and intuition that all graphs can be embedded with no stretch in $\ell_2^{O(\log n)}$.

2.1 Introduction

A *greedy embedding* of an unweighted undirected graph $G = (V, E)$ into a metric space (X, ρ) is a function $f : V \rightarrow X$ such that for every *source-sink* pair of different vertices $s, t \in V$ it is the case that s has a neighbor v in G with $\rho(f(v), f(t)) < \rho(f(s), f(t))$. From here on $n = |V|$ and the word “embedding” will refer to a greedy one unless otherwise stated. This definition implies that routing greedily (with respect to the host metric) in G always succeeds. In particular, a routing algorithm induced by a given greedy embedding works as follows. To deliver a letter from $s \in V$ to $t^* \in X$, the algorithm recursively forwards the letter to a neighbor of minimal embedding distance to t^* (ties are broken arbitrarily in a deterministic manner which is universally fixed for the purposes of our discussion), provided that such neighbor is closer to t^* than the current vertex. Otherwise, routing halts and it is assumed that the target has been reached. If the embedding is greedy and $t^* = f(t)$ for some $t \in V$, then the letter is guaranteed to reach t .

The notion of a greedy embedding is motivated by its applications to routing-with-local-information in large distributed systems (discussed in more detail later). In this context one is particularly concerned with three properties of the embedding algorithm. From here on we will loosely use the term *embedding* to refer to f itself or to the algorithm that finds an embedding for a given input graph.

- i. For a given embedding algorithm, the maximum (over $v \in G$) number of bits that the algorithm uses to describe $f(v)$ is called the *description complexity* of the embedding (algorithm). Note that in a typical application, the computer at node v stores its own coordinate $f(v)$ in order to be able to perform routing tasks. Embeddings with $\Omega(n)$ description complexity are not interesting in light of application constraints. Our primary interest is in embeddings with $\text{polylog}(n)$ description complexity, heretofore referred to as *concise* embeddings.
- ii. Every embedding f defines a unique path in G between all pairs of unequal vertices (s, t) , which is the path realized by the greedy routing algorithm (when routing from s to $f(t)$ and vice-versa) with respect to f (after resolving ties deterministically, as noted above). The length of this path is denoted by $d_f(s, t)$, which is to be distinguished from the length of the shortest-path between s and t in G denoted by $d_G(s, t)$. With this notation in hand, the *stretch* of a greedy embedding is defined as $D = \max_{s \neq t \in V} \frac{d_f(s, t)}{d_G(s, t)}$. An embedding with stretch $D = 1$ is called a *no-stretch* embedding. Note that no-stretch greedy embeddings are not equivalent to no-stretch distance-preserving embeddings. (Examples are given below.)
- iii. The *congestion* of a greedy embedding is defined as the edge-congestion of the set of routes realized by greedy routing (with respect to the embedding) between all pairs of vertices.

2.1.1 History of the problem

The power of geometric interpretation for routing problems was initially recognized in a sequence of papers [59, 18, 44, 71] from the ad-hoc, wireless and sensor networks communities. These papers consider the problem of routing messages in ad-hoc wireless networks where participating nodes are aware of their physical planar location on Earth; and, additionally, the connectivity graph (induced by the nature of radio communications) is close to planar. The papers describe routing algorithms that make local forwarding decisions based on the geographic location of the target node and the current node's neighbors. The algorithms have a common framework. First, a planarization of the connectivity graph is obtained; consequently, routing consists of greedy approach towards the target, combined with face routing around the perimeter of obstacles (when greedy approach is not possible).

Routing with geographic location, however, has insurmountable shortcomings. In particular, [71] shows that the best possible routing algorithm (based only on geographic location) may result in routing costs that are quadratic in the size of the optimal ones. This negative result is due to the arbitrary geometric complexity that the obstacles can have. Two other shortcomings are the assumption that the connectivity graphs are close to planar and that geographic location information is available. These two assumptions render the routing schemes under consideration useless for more complex networks like the Internet or P2P networks, where the connectivity graphs are significantly more complex: Such graphs are often modeled by scale-free or preferential-attachment random graphs [16, 17].

In light of these shortcomings, the non-strictly-theoretical approaches of [91, 39, 19] consider assigning virtual coordinates in \mathbf{R}^d to network nodes, so that basic greedy routing works with little modification. The assignment methods investigated entail variants of the rubber-band algorithm (applied to the connectivity graphs) and multi-dimensional scaling techniques (applied to the shortest-path metric of the connectivity graphs). The experimental results of these papers are generally promising but far from perfect or well-understood for large and realistic classes of graphs.

The first rigorously theoretic attempt at the problem was made by Papadimitriou et al. in [82], where the notion of greedy embedding was defined. This paper concerns the question of mere existence of greedy embeddings for graphs (irrespective of stretch or congestion). The paper shows that any graph containing a 3-connected planar subgraph has a greedy embedding in \mathbf{R}^3 , and conjectures that every such graph has a greedy embedding in \mathbf{R}^2 as well. This conjecture was proven correct for graphs containing a triangulated planar subgraph [28].

Following the work of [82], and perhaps motivated by the application of greedy embeddings, Kleinberg [66] asks the more general question: *Are there (nice) host metric spaces that accommodate greedy embeddings of all connected graphs?* He answers this question affirmatively using the fact that a greedy embedding of a graph spanning tree is also a greedy embedding of the graph (albeit with possibly arbitrary stretch), and showing that all trees can be embedded (not concisely) into \mathbf{H}^2 , the 2-dimensional Lobachevsky space. Additionally, his paper highlights the importance of the stretch and congestion parameters of greedy embeddings in view of applica-

tions. For embeddings of star graphs on n vertices into \mathbf{R}^d endowed with a Minkowski norm, it is shown that $d = \Omega(\log n)$. Kleinberg concludes his work with a list of open problems regarding existential and algorithmic aspects of greedy embeddings with various levels of stretch.

2.1.2 Our results

This work is a continuation of the study of greedy embeddings. The primary theme in our paper is addressing the existence of no-stretch embeddings for all graphs. Our findings provide evidence that such embeddings may exist. Our emphasis on finding embeddings with no stretch is in line with the fact that in real-world routing applications even small amounts of stretch are prohibitive.

First, in the spirit of keeping the applications in mind, we address the bit complexity of greedy embeddings (defined above). We improve Kleinberg’s result by showing that all trees (as well as some other tree-like graphs) have concise (also defined above) greedy embeddings into \mathbf{H}^3 , the 3-dimensional Lobachevsky space. We complement this result by exhibiting concise low-dimensional greedy embeddings of trees into ℓ_2 . The latter construction sheds some light on the “shape” of possible greedy embeddings of general graphs in Euclidean spaces.

Second, we give arguments and a theorem that strongly suggest that no-stretch embeddings do not require high dimension and, in fact, we believe that all connected graphs have concise no-stretch low-dimensional embeddings in ℓ_2 . Therefore, we begin a systematic attempt to understand the structure of no-stretch embeddings. As a first step, we develop a unified technique with a strong topological flavor that demonstrates that a certain family of graphs with “hard crossroads” requires $\Omega(\log n)$ dimensions to embed into Lobachevsky or Euclidean space. This technique motivates an interesting topological question regarding Minkowski normed spaces and manifolds. Our lower-bound can be interpreted as saying that Lobachevsky geometry is no more powerful than Euclidean geometry when it comes to harder graphs, contrary to intuition. We complement this lower-bound with a theorem stating that every no-stretch greedy embedding into ℓ_2^d can be used to derive a corresponding embedding into \mathbf{H}^{d+1} .

This paper is not concerned with congestion since we believe that this parameter is of secondary importance. This belief is supported by the fact that standard techniques like using a distribution over embeddings or routing through randomly selected intermediaries can be used to reduce congestion.

2.1.3 Dimension reduction does not help

Finding low-dimensional greedy embeddings into ℓ_2 is hard: In this section we explain that standard dimensionality reduction techniques alone are of no use for constructing greedy embeddings. When seeking low-dimensional Euclidean embeddings it is common to use one of the following two approaches:

- *The Bourgain approach:* Find an embedding into an arbitrary metric that realizes the required embedding properties. Then squeeze this embedding into $\ell_2^{O(\log n)}$ using Bour-

gain's $O(\log n)$ -distortion embedding, while “making sure that the necessary embedding properties are preserved.” Alternatively,

- *The Johnson-Lindenstrauss approach:* Find an embedding into ℓ_2 (of arbitrary dimension) that realizes the required embedding properties. Then reduce the dimension using Johnson-Lindenstrauss Flattening Lemma, while again “making sure that the necessary embedding properties are preserved.”

In the case of greedy embeddings “making sure that the necessary embedding properties are preserved” boils down to requiring that pairwise distances whose relative magnitudes must be preserved by the embedding have a sufficiently large margin ϵ . The minimum margins are $\epsilon = O(\log n)$ and $\epsilon = 1/\log^{O(1)} n$, respectively, for the above two approaches.

We are going to show that neither of these requirements are achievable for almost any graph G . Start with two technical observations:

Lemma 2.1.1. *Let $s = v_1, v_2, \dots, v_{k+1} = t$ be the unique shortest path between s and t in a graph G . Let $f : (G, d_G) \rightarrow (X, d_X)$ be a no-stretch greedy embedding with margin ϵ , and also let $x_i = d_X(f(v_i), f(v_{i+1}))$. Then $x_1 + \dots + x_k \geq (1 + \epsilon)^{k-1} \cdot \max \{x_1, \dots, x_k\}$*

Sketch of Proof: The margin requirement says that for any three vertices u, v and w such that u and v are adjacent and v is on the unique shortest path from u to w , it must hold that $d_X(f(u), f(w)) > (1 + \epsilon)d_X(f(v), f(w))$. Observe that for any $1 \leq i < j \leq k$ the unique shortest-path between v_i and v_j is v_i, v_{i+1}, \dots, v_j . Now the lemma follows by induction. \square

Corollary 2.1.2. *Using the setup from the previous lemma, the following must hold $k \geq (1 + \epsilon)^{k-1}$*

Proof. Let $x_{\max} = \max \{x_1, \dots, x_k\}$, then simply: $k \cdot x_{\max} \geq x_1 + \dots + x_k \geq (1 + \epsilon)^{k-1} \cdot x_{\max}$. \square

When we substitute for the margin ϵ in the above corollary we get:

Corollary 2.1.3. *No graph that has a unique shortest path of length $O(1)$, respectively $\log^{O(1)} n$, can be greedily embedded in $\ell_2^{\text{polylog}(n)}$ using the Bourgain approach, respectively the Johnson-Lindenstrauss approach.*

In other words, both approaches are futile for almost every graph, and in particular for nice classes of graphs like trees, cycles, random graphs, etc.

2.1.4 Road map

The paper is organized as follows. Section 2.2 positions our work with respect to the related class of ordinal and proximity embeddings. Sections 2.3 and 2.4 discuss the preliminaries of hyperbolic geometry, greedy embeddings, and tree decompositions. Section 2.5 proves a lower-bound on embedding dimension for a family of graphs with rich combinatorial structure. Sections 2.6

and 2.7 explain our concise embeddings for trees in Lobachevsky and Euclidean space, respectively. Finally, Section 2.8 contains concluding remarks and open problems motivated by our work.

2.2 Topological embeddings

2.2.1 Ordinal embeddings

Greedy embeddings are a type of *ordinal embeddings*, i.e. embeddings that preserve the relative order of pairwise vertex distances. The latter have enjoyed significant attention in the multi-dimensional scaling community in view of their applications to visualization, compression, nearest-neighbor search, etc. (see [3] for details). The strictest kind of ordinal embeddings are monotone maps, which are discussed below. Monotone maps provably require $\Omega(n)$ dimensions to realize almost all distance orders on n -point metrics (see [15]). To address this problem [3] considers *ordinal embeddings of minimum relaxation*, a variant that enforces order preservation of well-separated points only. In this vein, greedy embeddings are a variant of ordinal embeddings that require order preservation only among pairs of points of the form (x, z) and (y, z) where x and y must be neighbors or share a neighbor in the original graph.

2.2.2 Greedy embeddings vs. monotone maps

In [15] Linial and Bilu study embeddings that preserve relative pairwise distances. They define a *monotone map* as function $f : X \rightarrow Y$ mapping a finite metric (X, d_X) into a (usually normed or otherwise nice) host metric (Y, d_Y) such that $\forall a, b, c, d \in X : d_X(a, b) < d_X(c, d) \Leftrightarrow d_Y(f(a), f(b)) < d_Y(f(c), f(d))$. They show that any ordering on $\binom{[n]}{2}$ can be realized as a metric on n points with a matching ordering on the pairwise distances. Furthermore, they show that any such metric can be embedded in ℓ_2^m and almost no such metrics can be embedded in $\ell_2^{o(n)}$.

We prove a theorem showing that the relationship between monotone maps and greedy embeddings is weak at best. The proof of the following theorem is deferred to the full version:

Definition 2.2.1. A *reduction* of the problem of finding a monotone map for a given order $\pi \in S_{\binom{[n]}{2}}$ on the pairwise distances between n points, is a function R from $S_{\binom{[n]}{2}}$ to the set of unweighted undirected graphs. Additionally, there is a subset of vertices $H \subseteq V(R(\pi))$ such that $|H| = n$ and for every no-stretch greedy embedding f of $R(\pi)$ the restriction to H of f is a monotone map for π .

Theorem 2.2.2. *The problem of finding a monotone map for a given order on the pairwise distances between n points cannot be reduced to a problem of finding a no-stretch greedy embedding of a graph on $o(e^n)$ vertices for $1 - o(1)$ fraction of orders.*

It is obvious that a monotone map for a graph metric is also a greedy embedding for this metric. On the other hand, there is an abundance of examples where the greedy embedding of a graph metric is not necessarily a monotone map. Perhaps the best such example is that of $K_{n,n}$. The complete bipartite graph $K_{n,n}$ has no monotone map into $\ell_2^{o(n)}$ while it has a no-stretch greedy embedding into ℓ_2^2 : Position all vertices of $K_{n,n}$ uniformly along the unit circle S^1 while interlacing vertices from opposite sides of $K_{n,n}$. The techniques used in [15] to derive the $\Omega(n)$

bound on the monotone dimension of most graphs were borrowed from [4] and these techniques produce only trivial lower-bounds for the greedy embedding dimension.

2.2.3 Greedy embeddings vs. sphericity

Another related notion is that of a proximity embedding defined in [77]. A *proximity embedding* $f : G \rightarrow \ell_2^d$ of an undirected unweighted graph G is one for which $\forall v, w \in G : \|f(v) - f(w)\| < 1 \Leftrightarrow (v, w) \in G$. The *sphericity* of a graph is the minimum dimension d for which such embedding exists. As pointed out in [15], the sphericity of a graph is a lower-bound on its minimum monotone map dimension. Sphericity also bears only a weak connection to greedy embeddings. On the one hand, $K_{n,n}$ has sphericity $\Omega(n)$ while it embeds greedily into ℓ_2^2 with no stretch. On the other hand, trees and graphs of bounded degree have easy proximity embeddings in $\ell_2^{O(\log n)}$ using standard constructions involving the Johnson-Lindenstrauss lemma and only local graph structure considerations (see [42]). In contrast, Appendix 2.1.3 proves that standard techniques based on flattening or Bourgain's embedding cannot be used to construct greedy embeddings.

2.3 Hyperbolic geometry

Hyperbolic geometry is a vast and complex area with applications in various branches of Mathematics. In this section we give a brief and somewhat self-sufficient introduction to the properties of hyperbolic spaces used in this paper. More comprehensive expositions can be found in the classical texts of Thurston [105] and Do Carmo [29]. A considerably more concise and self-contained introduction is the one by Katok [61], which we recommend for the beginner. Hyperbolic geometry has found little attention in Computer Science, but for a few notable exceptions [98, 66, 68].

Hyperbolic spaces, also known as Lobachevsky spaces (not to be confused with the more general Gromov δ -hyperbolic spaces [5, 48]; of course, Lobachevsky spaces are also Gromov log 3-hyperbolic), can be constructed either axiomatically [25] (much like classical Euclidean geometry) or more explicitly using the language of Differential Geometry [29]. For the benefit of the reader's intuition we give the latter construction. We then state a few simple facts (while omitting proofs) which will enable us to reason about hyperbolic geometry in terms of its model via the more familiar Euclidean space.

2.3.1 The half-plane model

The d -dimensional real hyperbolic space, denoted \mathbf{H}^d , is modeled by the *upper-half plane* $\mathbf{R}^{d+} = \{(x_1, \dots, x_d)^T \in \mathbf{R}^d \mid x_d > 0\}$ in \mathbf{R}^d endowed with the Riemannian metric:

$$ds^2 = \frac{dx_1^2 + \dots + dx_d^2}{x_d^2}$$

By construction \mathbf{H}^d is geodesic. The Euclidean hyperplane $\partial\mathbf{H}^d = \{(x_1, \dots, x_d)^T \in \mathbf{R}^d \mid x_d = 0\}$ plays a special role and is called the *boundary at infinity*. The following few facts establish the basic properties of \mathbf{H}^d .

Theorem 2.3.1 (See Proposition 3.1 in [29]). *Infinite geodesics, also called lines, in \mathbf{H}^d (i.e. isometric maps of the form $g : \mathbf{R} \hookrightarrow \mathbf{H}^d$) correspond to Euclidean circles and lines orthogonal to the boundary at infinity and restricted to the upper half-plane, collectively referred to as generalized circles.*

Fact 2.3.2 (See [29] p.177). Hyperplanes in \mathbf{H}^d (i.e. isometric maps $h : \mathbf{H}^{d-1} \hookrightarrow \mathbf{H}^d$) correspond to $(d-1)$ -dimensional Euclidean spheres and planes orthogonal to the boundary at infinity and restricted to the upper half-plane, collectively called *generalized spheres*.

The isometries of \mathbf{H}^d are modeled by conformal (angle-, but not orientation-, preserving) transformations of \mathbf{R}^d that map the upper half-plane to itself, restricted to the upper half-plane. More notably though:

Theorem 2.3.3 (See Theorem 5.2 and Theorem 5.3 in [29]). *Let $f : \mathbf{H}^d \rightarrow \mathbf{H}^d$ be an isometry. Then, f is the restriction to \mathbf{R}^{d+} of a composition of Euclidean isometries, dilations or inversions that map \mathbf{R}^{d+} onto itself, at most one of each.*

We assume that the reader is already familiar with the isometries and dilations of \mathbf{R}^{d+} . An *inversion about a Euclidean hyperplane* is defined as Euclidean reflection with respect to that hyperplane. An *inversion of a point p about a Euclidean hypersphere* centered at c with radius r is defined as the unique point q on the ray \overrightarrow{cp} for which $|cp| \cdot |cq| = r^2$. For convenience, we define shorthand notation for the Euclidean hemisphere $S_{c,r} = \{u \in \mathbf{R}^{d+} : \|u - c\| = r\}$ and the corresponding half-ball $B_{c,r} = \{u \in \mathbf{R}^{d+} : \|u - c\| < r\}$, where in both cases $c \in \partial\mathbf{H}^d$.

In this paper we use three specific isometries to construct greedy embeddings. Since we need to keep track of the bit complexity of point coordinates after application of isometric transformations, we give explicit formulas for them here:

- An *inversion* about a hyperbolic hyperplane corresponding to a Euclidean hemisphere $S_{c,r}$ is given by $\alpha_{c,r}(v) = (v - c) \cdot r^2 / \|v - c\|^2 - c$
- A *translation* by a vector $w \in \partial\mathbf{H}^d$ is given by $\beta_w(v) = v + w$
- A *dilation* at the origin by a factor $D > 0$ is given by $\gamma_D(v) = D \cdot v$

Finally, we will need an expression for the pairwise distance function of \mathbf{H}^d . We will denote the geodesic segment between points v and w in \mathbf{H}^d by $[v, w]$. The Riemannian metric on \mathbf{H}^d naturally induces a pairwise metric function $\rho(\cdot, \cdot)$. Let $\tilde{h} = \{x \in \mathbf{R}^d \mid x_2 = \dots = x_{d-1} = 0, x_d > 0\}$. We give an expression for $\rho(v, w)$ for the case when $v, w \in \mathbf{H}^d \cap \tilde{h}$. (It should be clear that hyperbolic isometries can position any two points in this manner.) \tilde{h} can be viewed as a copy of \mathbf{H}^2 , and v and w can be located in \tilde{h} using only two coordinates, namely the 1-st and the d -th. We shall now view v and w as complex numbers in the following way $v = v_1 + iv_d$ (similarly for w). With this notation in hand, the following theorem gives the pairwise distance between v and w :

Theorem 2.3.4 (See Theorem 1.2.6 in [61]). *Let $v, w \in \mathbf{H}^2$, then:*

$$\rho(v, w) = \ln \frac{|v - \bar{w}| + |v - w|}{|v - \bar{w}| - |v - w|}$$

2.3.2 The Klein model

The Klein model of hyperbolic space will be instrumental in our lower-bound proof. In the Klein model \mathbf{H}^d is modeled by the d -dimensional Euclidean disc $D^d = \{x \in \mathbf{R}^d \mid \|x\| < 1\}$. In particular, the Klein model can be viewed as a homeomorphism $h : \mathbf{H}^d \rightarrow D^d$. We shall make use of one simple property of this model; for further information, we refer the interested reader to [105]:

Fact 2.3.5. Hyperbolic hyperplanes in the Klein model correspond to Euclidean hyperplanes restricted to the unit disc.

2.3.3 Bisecting hyperplanes

To prove correctness of our constructions, we will use a lemma from [66] for \mathbf{H}^2 whose proof translates to \mathbf{H}^d unchanged:

Lemma 2.3.6. *Let v and w be different points in \mathbf{H}^d , and let b be the hyperbolic hyperplane that bisects the geodesic $[v, w]$, then for all $u \in \mathbf{H}^d$ it holds that $\rho(v, u) < \rho(w, u)$ if and only if v and u are in the same half-space with respect to b .*

In order to apply this lemma in our constructions, we will need to identify the bisecting hyperplane between vertices with equal d -th coordinates:

Lemma 2.3.7. *Let $u, v \in \mathbf{H}^d$ such that $u_d = v_d$. Then the hyperbolic hyperplane bisecting $[u, v]$ coincides with the Euclidean hyperplane bisecting u and v (in Euclidean sense).*

2.4 Embeddings and tree decompositions

2.4.1 Greedy embeddings basics

Here we list a few easy-to-verify facts about greedy embeddings that we use implicitly throughout the paper. More details can be found in [66].

Fact 2.4.1. If $H \subseteq G$ is a subgraph containing all vertices of G , then every greedy embedding of H is also a greedy embedding of G , albeit with a possibly higher stretch.

Fact 2.4.2.

- i. If T is a tree and $T' \subseteq T$ is a subtree, then every greedy embedding of T in a metric space X restricts to a greedy embedding of T' .
- ii. Greedy embeddings of trees always have no stretch 1.
- iii. If X is a normed vector space which admits a greedy embedding of the star graph on n vertices, then $\dim(X) = \Omega(\log n)$.

2.4.2 Distance-preserving embeddings

Definition 2.4.3. A map $f : X \rightarrow Y$, where (X, d_X) and (Y, d_Y) are metric spaces, is a distance-preserving embedding of X into Y with distortion $D > 0$, if there exists a constant $r > 0$ such that:

$$\forall v, w \in X \quad r \cdot d_X(v, w) \leq d_Y(f(v), f(w)) \leq D \cdot r \cdot d_X(v, w)$$

2.4.3 Tree decomposition and heavy paths

This section describes a variant of the well-known caterpillar decomposition of trees [74, 79], also recognized as Tarjan and Harel's [53] heavy-path decomposition. Let T be an arbitrary non-rooted tree on n vertices. A *path decomposition* of T into k paths is a collection of vertex-disjoint line subgraphs of T which covers T 's vertices completely, i.e. $T = P_1 \uplus \dots \uplus P_k$. A *hierarchical path decomposition* is a path decomposition which is additionally endowed with a hierarchical relationship among the paths. In particular, this relationship is represented by a rooted tree H whose vertex set is P_1, \dots, P_k . Furthermore, (P_i, P_j) is an edge in H iff P_i and P_j are connected by an edge in T . A *heavy-path decomposition* of a (rooted/non-rooted) tree is a particular hierarchical path decomposition which has depth at most $2\kappa(T) \leq 2 \log L$, where L is the number of leaves of T . The quantity $\kappa(T)$ is the *caterpillar dimension* of T . It is easily verified that for an non-rooted tree T , a caterpillar decomposition of T using an arbitrary root can be modified to produce a heavy-path decomposition of depth at most $2\kappa(T)$. A heavy-path decomposition of a bounded degree-3 tree is illustrated in Figure 2-1.

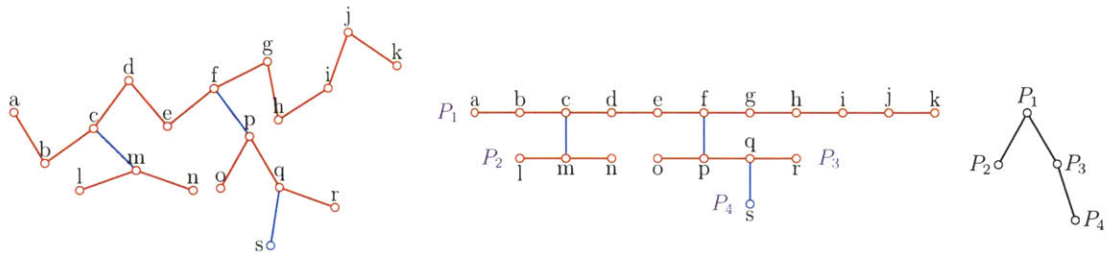


Figure 2-1: Shown: (a) an non-rooted tree, (b) its heavy-path decomposition, and (c) the hierarchical path relationship.

2.5 Lower bounds

In this section we develop a “dual” representation of greedy embeddings in terms of “bisecting sets,” which allows us to prove a unified lower bound on the Euclidean and Lobachevsky dimensionality of a certain family of graphs. The Lobachevsky bound is new, while the Euclidean was already known from [66]. Nevertheless, the unified framework of the proof seems to be of value.

2.5.1 Graphs with hard crossroads

We now define a family of graphs *with hard crossroads* that have rich combinatorial structure on the set of all-pairs shortest paths. Let \mathcal{Q}_d be the graph on $n = d + 3 \cdot 2^d$ vertices and $m = 2^d \cdot (d + 2)$ edges, defined as follows. The vertex set consists of $\{s_i\}_{i \in [2^d]}$, $\{t_j\}_{j \in [d]}$, $\{w_{i,q}\}_{i \in [2^d], q \in \{0,1\}}$. The edge set consist of two types of edges:

- i. For every $i \in [2^d]$, include the edges $(s_i, w_{i,0})$ and $(s_i, w_{i,1})$.
- ii. Let $i = b_1 b_2 \dots b_d$ be the binary representation of i . Then for every $i \in [2^d]$, include the edges $(w_{i,b_1}, t_1), (w_{i,b_2}, t_2), \dots, (w_{i,b_d}, t_d)$.

The main result of this section is the following theorem:

2.5.2 Dimension theorem

Theorem 2.5.1. *Every no-stretch greedy embedding of \mathcal{Q}_d into Euclidean or Lobachevsky space requires d dimensions.*

This theorem implies a $\log(n)$ lower bound on the dimension of no-stretch greedy embeddings of graphs on n vertices.

Sketch of Proof: (See Section 2.5.3 for complete proof.) Let $f : \mathcal{Q}_d \rightarrow (X, d_X)$ be an embedding into a geodesic metric space with a continuous pairwise distance metric. We shall use v and $f(v)$ interchangeably. Let us now cover some topological preliminaries.

Let $a \neq b \in X$ be two different points in X . Define the *bisecting set* of $[a, b]$ to be $\text{Bisect}(a, b) = \{c \in X \mid d_X(c, a) = d_X(c, b)\}$. In the spaces under consideration, every bisecting set is non-trivial and furthermore it separates the space X in at least two disjoint sets, called *chambers*, one for each endpoint of the bisected geodesic segment. We use the notation $(c^a |^b d)$ to indicate that c (respectively d) lies in the chamber of a (respectively b) with respect to the bisecting set of $[a, b]$. We also use that chambers are preserved by homeomorphisms. More generally, let $S \subset X$ separate X into a collection of chambers $\{C_\alpha\}_\alpha$ and let $h : X \rightarrow Y$ be a homeomorphism, then $h(S)$ separates Y exactly into $\{h(C_\alpha)\}_\alpha$.

It is easily seen that the correctness of f (as a no-stretch greedy embedding) can be expressed by a set of inequalities of the form $d_X(f(x), f(z)) < d_X(f(y), f(z))$ where $x, y, z \in V(\mathcal{Q}_d)$. Every

such inequality implies a (weaker) *separation constraint* $(z^x|y)$. The collection of separation constraints partitions X into a set of chambers and establishes combinatorial constraints regarding the position of $f(x)$, for every $x \in V(\mathcal{Q}_d)$, with respect to containment in chambers.

The idea of the proof is to find a homeomorphism h that sends X to \mathbf{R}^d while mapping all bisecting sets (or at least subsets thereof) to hyperplanes. Then using Linear Algebra we establish that the required (by the separation constraints) geometric set system (See [80] for definition), formed by the points $\{(h \circ f)(x)\}_{x \in V(\mathcal{Q}_d)}$ and the hyperplanes $\{h(\text{Bisect}(f(x), f(y)))\}_{x \neq y \in V(\mathcal{Q}_d)}$, cannot be realized in low dimensions.

When X is Lobachevsky, the bisecting sets are hyperbolic hyperplanes, and the Klein model is the required homeomorphism. When X is Euclidean the homeomorphism is simply the identity. \square

It is an interesting (as far as we know, open) question to find homeomorphisms that work for Minkowski normed spaces. Some thought will convince the reader that such homeomorphisms will have to be input-specific, unlike the universal Klein model homeomorphism for the Lobachevsky case. We also believe that this approach can be extended to nice classes of manifolds (but we do not dabble in this here).

2.5.3 Complete proof

Dual Representation

Let $G = (V, E)$ be a graph and let $f : V \rightarrow (X, \|\cdot\|)$ be an embedding, where $(X, \|\cdot\|)$ is one of ℓ_2^d or \mathbf{H}^d and d is referred to as the *dimension of X* . The conditions for f being a no-stretch greedy embedding of G are described by a collection of inequalities, heretofore called *greedy constraints*, of the form:

$$(2.5.1) \quad \|f(x) - f(z)\| < \|f(y) - f(z)\|$$

where $x, y, z \in V$ are pairwise unequal. When the host metric is Euclidean or Lobachevsky, we can rephrase the constraints using the language of hyperplanes. We shall use the notation $(a^c|_d b)$, where a, b, c and d are points in the host metric space, to mean that the bisecting hyperplane of $[c, d]$ separates a, c and b, d . More generally, in this notation we allow arbitrary lists (including the empty list) of points in place of a or b . Furthermore, we abuse notation a little by using v to refer both to a vertex $v \in V$ and its image $f(v)$. It is now easily seen that the constraint (2.5.1) can be rewritten in the form $(z^x|y)$. (The hyperbolic case follows from Lemma 2.3.6.)

Proof Outline

The idea of the proof is to examine the constraints of \mathcal{Q}_d and show that when X is low-dimensional there is no configuration of points and *any* hyperplanes that satisfy the constraints.

What makes this proof manageable is that we seek to realize the greedy constraints using arbitrary hyperplanes, rather than strictly bisecting ones. In order to unify our analysis of the normed and Lobachevsky cases, we make the following observation. In the Klein model of \mathbf{H}^d , the hyperbolic hyperplanes correspond to Euclidean hyperplanes restricted to the unit disc $D^d = \{x \in \mathbf{R}^d : \|x\| < 1\}$ in \mathbf{R}^d . Therefore for both types of geometries it suffices to show that the hyperplane/point configurations required by \mathcal{Q}_d cannot be realized in $\ell_2^{d'}$ with $d' < d$. This will be established using simple Linear Algebra.

Linear Algebra and Point/Hyperplane Configurations

We shall only concern ourselves now with Euclidean geometry. If M is a matrix, we will let m_i , M_j and $M_{i,j}$ refer to the i -th row, j -th column and the (i, j) -th entry of M , respectively.

A *(point/hyperplane) configuration* Ψ is a collection of points $V_1, \dots, V_n \in \mathbf{R}^d$ and hyperplanes $(a_1^T, b_1), \dots, (a_k^T, b_k) \in \mathbf{R}^d \times \mathbf{R}$, where a point V_j is on the “positive” side of (a_i^T, b_i) iff $a_i V_j - b_i > 0$. The left-hand side of the latter inequality is referred to as the *polarity* of V_j with respect to (a_i^T, b_i) .

Let $A \in M_{k,d}(\mathbf{R})$ be the matrix whose rows are a_1, \dots, a_k , $V \in M_{d,n}(\mathbf{R})$ be the matrix whose columns are V_1, \dots, V_n , and $b \in \mathbf{R}^k$ be the vector whose entries are b_1, \dots, b_k . Define the *signature* of Ψ to be the matrix $\chi(\Psi) = AV - b \cdot \mathbf{1}^T$, where $\mathbf{1}^T = (1, \dots, 1) \in \mathbf{R}^k$. Observe that $\chi(\Psi)$ has a natural interpretation; in particular, $\text{sign}(\chi(\Psi)_{i,j})$ indicates the polarity of V_j with respect to (a_i^T, b_i) . Furthermore, $\chi(\Psi)$ can be interpreted as a configuration where the points are represented by the columns of $\chi(\Psi)$ and the hyperplanes are the canonical Euclidean hyperplanes through the origin, orthogonal to the unit vectors e_i in the i -th direction. In that sense, $\chi(\Psi)$ is a “straighten-out” version of Ψ which is more amenable to dimension analysis. In the rest of the proof, we will make use of the following property of $\chi(\Psi)$:

Lemma 2.5.2. *For Ψ as above, $\dim(\text{span}(V_1, \dots, V_n)) \geq \text{rank}(\chi(\Psi)) - 1$.*

Proof of Lemma 2.5.2. Note that $\text{rank}(b \cdot \mathbf{1}^T) \in \{0, 1\}$, then:

$$\begin{aligned} \dim(\text{span}(V_1, \dots, V_n)) &= \text{rank}(V) \\ &\geq \text{rank}(AV) \\ &= \text{rank}(\chi(\Psi) + b \cdot \mathbf{1}^T) \\ &\geq |\text{rank}(\chi(\Psi)) - \text{rank}(b \cdot \mathbf{1}^T)| \\ &\geq \text{rank}(\chi(\Psi)) - 1 \end{aligned}$$

□

For every greedy embedding of a graph G , we can view the image of $V(G)$ and the corresponding bisecting hyperplanes between all pairs of points as a configuration. The greedy constraints will impose certain sign-constraints on the entries of the signature of this configura-

tion. In the case of \mathcal{Q}_d , these sign-constraints will help us derive a lower bound on the rank of the signature and hence on the dimensionality of the embedding.

The Constraints of \mathcal{Q}_d

It is easily checked that the following is a subset of the no-stretch greedy constraints of \mathcal{Q}_n , involved along the routes from s_i , for $i \in [2^d]$, to t_j , for $j \in [d]$. Let $i = b_1 b_2 \cdots b_d$ be the binary representation of i and let $\pi \in S_d$ be a permutation such that $b_{\pi(1)} = \cdots = b_{\pi(q)} = 0$ and $b_{\pi(q+1)} = \cdots = b_{\pi(d)} = 1$ where $0 \leq q \leq d$. The constraints are:

$$(2.5.2) \quad \forall i \in [2^d] \quad (t_{\pi(1)}, \dots, t_{\pi(q)})^{w_{i,0}|w_{i,1}} t_{\pi(q+1)}, \dots, t_{\pi(d)}$$

Let Ψ be the configuration corresponding to a no-stretch greedy embedding of \mathcal{Q}_d . As noted earlier, the rows of $\chi(\Psi)$ correspond to (and are indexed by) the bisecting hyperplanes, and the columns correspond to (and are indexed by) the vertices of \mathcal{Q}_d . Let $C \in M_{2^d, d}(\mathbf{R})$ be the sub-matrix of $\chi(\Psi)$ defined by the hyperplanes (rows) that appear in (2.5.2) and the vertices $\{t_j\}_{j \in [d]}$. It is clear that $\text{rank}(\chi(\Psi)) \geq \text{rank}(C)$. Next, we are going to show that $\text{rank}(C) = d$. This will imply $\text{rank}(\chi(\Psi)) \geq d$, and by Lemma 2.5.2 we will get the desired lower bound $d = \Omega(\log |V(\mathcal{Q}_d)|)$.

Rank of the signature

The constraints of (2.5.2) impose that the set of d -tuples

$$\{(\text{sign}(\sigma \cdot C_{i,1}), \text{sign}(\sigma \cdot C_{i,2}), \dots, \text{sign}(\sigma \cdot C_{i,d})) : i \in [2^d], \sigma \in \{-1, +1\}\}$$

contains all 2^d sign patterns on d slots. Then the following lemma implies that $\text{rank}(C) = d$:

Lemma 2.5.3. *Let $C \in M_{2^d, d}(\mathbf{R})$ be a matrix whose rows realize all 2^d sign patterns over d columns, then $\text{rank}(C) = d$.*

Proof of Lemma 2.5.3. Induct on d . The base case $d = 1$ is straightforward. Without loss of generality let $C \in M_{2^d, d}(\mathbf{R})$ be such that $\text{sign}(C_{i,j}) = \text{sign}(b_{i,j} - 1/2)$, where $b_{i,j}$ is the j -th bit in the binary representation of i . Let U be the sub-matrix of C consisting of the first 2^{d-1} rows. From the induction hypothesis, U has rank $d - 1$. Let $U' \in M_{d-1, d}(\mathbf{R})$ be a diagonalized version of U . In particular:

- i. $U'_{i, i+1} = 1$ for $i \in [d - 1]$,
- ii. $U'_{i, j+1} = 0$ for $i \neq j \in [d - 1]$, and

Pictorially:

$$U' = \begin{pmatrix} U_{1,1} & 1 & 0 & \cdots & 0 \\ U_{2,1} & 0 & 1 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ U_{d-1,1} & 0 & \cdots & 0 & 1 \end{pmatrix}$$

By definition, C must have a row c_l where $l \in [2^d] \setminus [2^{d-1}]$ such that:

- i. $\text{sign}(C_{l,1}) = +$, and
- ii. $\text{sign}(C_{l,j+1}) = \text{sign}(-U_{j,1})$ for all $j \in [d-1]$; if $U_{j,1} = 0$ then $\text{sign}(C_{l,j+1})$ can be arbitrary.

It is now easily verified that c_l is linearly independent from all rows in U' thereby proving the inductive step. \square

2.5.4 Connection between Euclidean and hyperbolic space bounds

The following theorem complements our lower-bound result:

Theorem 2.5.4. *If a graph G has a no-stretch greedy embedding into ℓ_2^d , then it has a no-stretch greedy embedding into \mathbf{H}^{d+1} .*

Sketch of proof of Theorem 2.5.4. Given a greedy embedding f of G into ℓ_2^d , the set $f(G)$ can be embedded onto the d -dimensional unit sphere $S^d \in \mathbf{R}^{d+1}$ such that the relative distance between all pairs of points is preserved. Let $g : V(G) \rightarrow S^d$ be this embedding. The bisecting hyperplanes between all pairs of points in $g(G)$ in \mathbf{R}^{d+1} go through the origin. The disc $2D^d = \{x \in \mathbf{R}^{d+1} : \|x\| < 2\}$ together with $g(G)$ inside it, can be interpreted as an embedding of G into \mathbf{H}^d via the Klein model. This is the required embedding. \square

2.6 Concise Hyperbolic Embeddings of Trees

Theorem 2.6.1. *Every tree T on n vertices has a concise greedy embedding in \mathbf{H}^3 with $O(\kappa(T) \cdot \log n)$ -bit vertex coordinates.*

We prove this theorem in the following few sections.

2.6.1 Construction

It is sufficient to exhibit embeddings for bounded degree-3 trees. This follows from the fact that if $T^* \supseteq T$ is a super-tree, then every greedy embedding of T^* restricts to a greedy embedding of T , and the fact that every tree T on n vertices is found as a subtree of a ternary tree of size no larger than $2n$.

We begin by obtaining a heavy-path decomposition $T = P_1 \uplus \cdots \uplus P_k$ with a hierarchical relationship on the P_j 's represented by a tree H (as in Section 2.4.3 and Figure 2-1).

Some notation is due now. Let P_j be a path in T , viewed as a vertex in H , and let $\text{parent}(P_j)$ denote its parent path in H (if it exists). Denote by $\text{apex}(P_j)$ be the unique vertex in $\text{parent}(P_j)$ that P_j connects to, and by $\text{exit}(P_j)$ the unique vertex in P_j that connects to $\text{apex}(P_j)$. Let $\text{subtree}(\text{apex}(P_j))$ denote the subtree of T consisting of P_j , all of its descendants in H , as well as $\text{apex}(P_j)$. For a vertex v in P_j that is not the apex of any P_i we will let $\text{subtree}(v)$ denote the singleton subtree of T consisting of v itself.

A *canonical embedding* $f_v : \text{subtree}(v) \rightarrow \mathbf{H}^3$ of $\text{subtree}(v)$, where $v \in T$, is one for which all relevant vertices are embedded in the interior of $B_{(0,0)T,1}$, and v is embedded at the unique location inside the ray e_z such that $\rho(f_v(v), S_{(0,0)T,1}) = \alpha$, where α is any fixed positive real number for which $e^\alpha \in \mathbf{Q}$. (Later we will see that we can also use $\alpha = 1$.) We will describe a recursive (on H) procedure that canonically embeds each $\text{subtree}(v)$ until all of T is embedded.

In the base case, canonically embedding a single vertex v is trivial. We simply embed v at the unique point on the ray e_z that has hyperbolic distance to $S_{(0,0)T,1}$ equal to α and is “inside” $S_{(0,0)T,1}$ (i.e. on the same side as the origin). Explicitly $f_v(v) = (0, 0, 1/e^\alpha)^T$. The bit complexity of this embedding is $O(1)$ due to our choice of α . We should note however that since the rest of the embedding will be obtained via isometric transformations, we can view the quantity $1/e^\alpha$ as an irreducible (or free) variable and describe all coordinates as polynomials over it. Either approach works.

Let us now proceed to the recursive step of embedding w where $w = \text{apex}(P_j)$ for some P_j consisting of vertices v_1, \dots, v_k . And let $\text{exit}(P_j) = v_q$ for some $q \in [k]$. From the recursion, we have embeddings f_{v_1}, \dots, f_{v_k} with $f_{v_i} : \text{subtree}(v_i) \rightarrow B_{(0,0)T,1}$. We shall first define an embedding $g_w : \text{subtree}(w) \rightarrow \mathbf{H}^3$ which is not canonical. Later we will transform g_w into a canonical one:

$$(2.6.1) \quad g_w(u) = \begin{cases} (\beta_{(i-q,0)T} \circ f_{v_i})(u), & \text{if } u \in \text{subtree}(v_i) \\ (0, 1, 1/e^\alpha)^T, & \text{otherwise, i.e. if } u = w \end{cases}$$

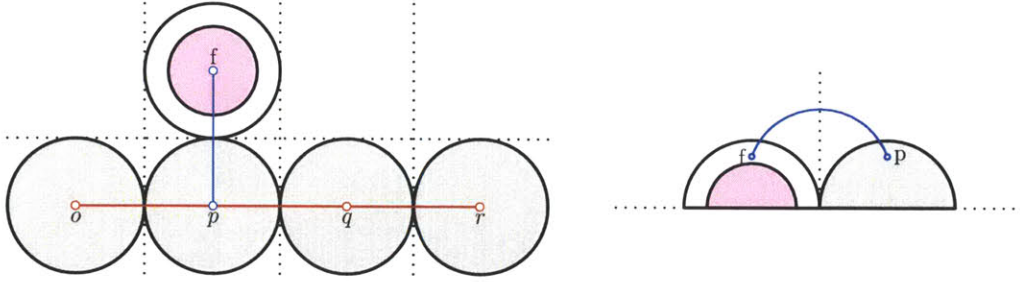


Figure 2-2: Illustrated is the g_f embedding of P_3 from Figure 2-1. On the left is a view from the z -axis looking down towards the origin. On the right is a planar section defined by $g_f(f)$, $g_f(p)$ and the origin.

The embedding $g_w(u)$ is illustrated in Figure 2-2. Our first order of business will be to check that it is correct. Afterwards we will apply the necessary isometric transformations to reshape it into canonical form. Notice that isometric transformations do not violate correctness.

2.6.2 Correctness argument

To check correctness (i.e. that greedy routing works), we have to identify a subregion $R \subset \mathbf{H}^3$ where we plan to position the rest of T , i.e. $T \setminus \text{subtree}(w)$, later in the recursion. We will let $R = B_{(0,1)\tau, 1/e^{2\alpha}}$. Note that R is intentionally chosen so that $\rho(g_w(w), R) = \alpha$.

The inductive (as in recursive) hypothesis is that all points in $\text{subtree}(v_i)$ for $i \in [k]$ are embedded in such a way (by f_{v_i}) that (a) greedy routing works among themselves, and (b) if routing is attempted to any location outside of $B_{(0,0)\tau, 1}$, it will reach v_i . Therefore our task is to check that under g_w :

- i. Routing from any v_i to $u \in \text{subtree}(v_j)$ reaches destination, and
- ii. Routing from any v_i to w or any location inside R reaches destination.

To prove the first part, it is sufficient to show that routing to $u \in \text{subtree}(v_j)$ reaches at least v_j , then by inductive hypothesis we know that u will be reached under f_{v_j} (translated by $\beta_{(i-q,0)\tau}$). Assume without loss of generality that $i < j$. Indeed, since $g_w(v_i)$ and $g_w(v_{i+1})$ have the same z coordinate, their bisecting hyperplane separates $g_w(v_i)$ from all points $g_w(u)$ where $u \in \text{subtree}(v_j)$. Therefore routing will progress to v_{i+1} . The second assertion is also easy to check. In particular, one simply verifies that at a vertex v_i the bisecting plane of the edge that leads to w separates v_i from w and all of R . This is illustrated in Figure 2-2 where bisecting hyperbolic hyperplanes are pictured as dotted lines.

2.6.3 Canonization

The canonical embedding f_w is obtained from g_w by:

- i. Applying a spherical hyperbolic inversion with respect to R . (This transformation takes all of g_w and “squeezes” it inside R .)
- ii. Translating R to R' so that R' is centered at the origin
- iii. Isometrically expanding R' to $R'' = S_{(0,0)T,1}$

Formally, $f_w = (\gamma_{e^{2\alpha}}) \circ (\beta_{(0,-1)T}) \circ (\alpha_{(0,1)T,1/e^{2\alpha}}) \circ (g_w)$.

2.6.4 Description complexity

To calculate the bit-description complexity per vertex, we will trace out what happens to a vertex’s coordinates throughout the recursion. At the lowest level of the recursion, a vertex starts off with $O(1)$ -bit coordinates (namely $(0, 0, 1/e^\alpha)^T$). At each level of the recursion, the vertex is translated by at most n positions along the x axis. This step adds at most $O(\log n)$ bits to its x -coordinate. Observe that the canonization step is a fixed isometric transformation, so it contributes $O(1)$ additional bits. There are $\kappa(T)$ recursive levels, amounting to a total of $O(\kappa(T) \cdot \log n)$ bits per vertex coordinate.

2.6.5 Remarks

We will briefly note (without proof) that since \mathbf{H}^d is Gromov $(\log 3)$ -hyperbolic (for every $d \geq 2$), if we scale our embedding procedure so that the hyperbolic distance between vertices sharing an edge is Δ , then the greedy embedding is also a distance-preserving embedding (in the sense of Definition 2.4.3) with distortion $1 + \log 3/\Delta$.

The techniques described in this section can be used to embed slightly more general classes of graphs. In particular, let G be a graph that can be decomposed into a vertex-disjoint family of subgraphs, i.e. $G = H_1 \uplus \dots \uplus H_k$. Let G^* be a graph with a vertex set $[k]$ where $(i, j) \in E(G^*)$ iff there is an edge in G between H_i and H_j . Then if G^* is a tree and each H_i can be embedded canonically with no stretch, all of G can be embedded canonically with no stretch.

It is easily seen, for example, that graphs that can be decomposed into lines and *cycles* succumb to the same embedding procedure. The canonical embedding of a cycle is illustrated in Figure 2-3. More complicated examples can be derived by using higher hyperbolic dimension and/or a cleverer arrangement of the canonical embeddings from lower levels of the recursion. The limitation of this technique, however, is that it is inherently recursive and therefore it applies to graphs that at large scale look like trees.

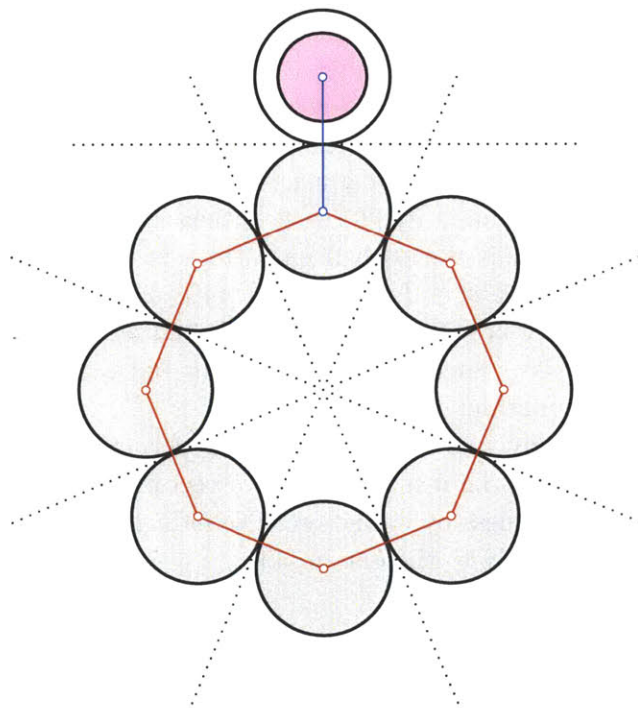


Figure 2-3: An illustration of the canonical embedding of a cycle.

2.7 Low dimensional Euclidean embeddings of trees

Inspired by ideas from [49], in this section we construct low dimensional greedy embeddings of trees into Euclidean spaces. Note that our construction is somewhat different than Gupta’s and surprisingly it does not require the use of a hierarchical path decomposition to accomplish conciseness.

Theorem 2.7.1. *Every tree T on n vertices has a concise greedy embedding in $\ell_2^{O(\log n)}$ with $O(\log^2 n)$ -bit vertex coordinates.*

Sketch of Construction: We begin by picking an arbitrary root v_0 for T . Using the Johnson-Lindenstrauss lemma, or alternatively using sphere packing constructions as in [49, 24], we obtain a bundle \mathcal{B} of $n - 1$ unit vectors such that (a) each vector has positive 1-st coordinate; (b) the angle between any two vectors is a constant slightly larger than $\pi/3$, say $\pi/3 + \pi/180$; and (c) \mathcal{B} is realized in $\ell_2^{O(\log n)}$.

The embedding algorithm assigns a vector $g(v, w) \in \mathcal{B}$ to each edge $(v, w) \in T$ in a manner to be specified shortly. The embedding $f : V(T) \rightarrow \ell_2$ is then defined as $f(v) = g(v_0, v_1) + \dots + g(v_{k-1}, v)$, where $v_0, v_1, \dots, v_{k-1}, v$ is the path from v_0 to v in T . The matching $g : E(T) \rightarrow \mathcal{B}$ is chosen as follows. For vertex $v \in T$ let $\tilde{g}(v) = \{g(u, w) \in \mathcal{B} : (u, w) \in \text{subtree}(v)\}$. Then $g(\cdot, \cdot)$ is such that for every $v \in T$ and all pairs of children v_a and v_b of v it holds that $\text{Cone}(\tilde{g}(v_a)) \cap \text{Cone}(\tilde{g}(v_b)) = \{\mathbf{0}\}$. Such a matching exists and can be found algorithmically using the sweeping-hyperplane method of [49].

Sketch of correctness: For any $v \in \mathcal{B}$ the bisecting hyperplane of v separates v from all other vectors in \mathcal{B} . Using this, and the fact that any two vectors in \mathcal{B} form an angle of roughly $\pi/3 + \pi/180$, one can show correctness by induction. The induction is guided by the “growing” process of creating the tree, similarly to the one in [49]. \square

2.8 Open Problems

An abundance of open problems arises from the notion of greedy embeddings and their applications. We only mention two.

The main open problem is that of finding no-stretch greedy embeddings of any graph into $\ell_2^{O(\log n)}$. We shortly describe a promising strategy for attacking this problem, which we have not yet investigated thoroughly. Let $\mathcal{Y}_{p,q,r}$ be the graph consisting of edge-disjoint copies of L_p, L_q and L_r (where L_l is the undirected line graph on l edges), exactly one of each, where also all three line subgraphs share a starting vertex and a (different) ending vertex. Let a *gadget* be a procedure for embedding $\mathcal{Y}_{p,q,r}$ into $\ell_2^{O(1)}$ for any p, q and r . We believe that using such a gadget in conjunction with dimensionality reduction, can lead to the desired embeddings of arbitrary graphs. Our intuition is based on the following lemma:

Lemma 2.8.1. *Every unweighted undirected graph can be decomposed into a collection of (not necessarily disjoint) sub-trees and (irreducible) sub-cycles such that (i) the shortest paths between vertices on a sub-cycle lie entirely in the sub-cycle, and (ii) the intersection of any two sub-cycles is a connected arc, a vertex, or the empty set.*

In view of applications, one other particularly important question concerns the existence of algorithms for finding greedy embeddings in Peleg's message-passing model of distributed network computation [85], where message-cost (in addition to time) is of central importance. Furthermore, algorithms with good incremental properties and resilience to small changes in the input graph are desired.

Chapter 3

Bottleneck-independent computation

So far we have looked at methods for computing and representing routing schemes aimed at point-to-point communication. At the other extreme are schemes for routing broadcast, or one-to-all, communication. In this chapter, we study the question of how efficiently a collection of interconnected nodes can perform a global computation in the widely studied GOSSIP model of communication. In this model, nodes do not know the global topology of the network, and they may only initiate contact with a single neighbor in each round. This model contrasts with the much less restrictive LOCAL model, where a node may simultaneously communicate with all of its neighbors in a single round. A basic question in this setting is how many rounds of communication are required for the information dissemination problem, in which each node has some piece of information and is required to collect all others.

In the LOCAL model, this is quite simple: each node broadcasts all of its information in each round, and the number of rounds required will be equal to the diameter of the underlying communication graph. In the GOSSIP model, each node must independently choose a single neighbor to contact, and the lack of global information makes it difficult to make any sort of principled choice. As such, researchers have focused on the *uniform gossip algorithm*, in which each node independently selects a neighbor uniformly at random. When the graph is well-connected, this works quite well. In a string of beautiful papers, researchers proved a sequence of successively stronger bounds on the number of rounds required in terms of the conductance ϕ and graph size n , culminating in a bound of $O(\phi^{-1} \log n)$.

In this work, we show that a fairly simple modification of the protocol gives an algorithm that solves the information dissemination problem in at most $O(D + \text{polylog}(n))$ rounds in a network of diameter D , with *no dependence on the conductance*. This is at most an additive polylogarithmic factor from the trivial lower bound of D , which applies even in the LOCAL model.

In fact, we prove that something stronger is true: *any* algorithm that requires T rounds in the LOCAL model can be simulated in $O(T + \text{polylog}(n))$ rounds in the GOSSIP model. We thus prove that these two models of distributed computation are essentially equivalent.

3.1 Introduction and results

3.1.1 Local models of computation

Many distributed applications require nodes of a network to perform a global task using only local knowledge. Typically a node initially only knows the identity of its neighbors and gets to know a wider local neighborhood in the underlying communication graph by repeatedly communicating with its neighbors. Among the most important questions in distributed computing is how certain global computation problems, e.g., computing a maximal independent set [75] or a graph coloring [9], can be performed with such local constraints.

Many upper and lower bounds for distributed tasks are given for the well-known LOCAL model [86, Chapter 2], which operates in synchronized rounds and allows each node in each round to exchange messages of unbounded size with all of its neighbors. It is fair to say that the LOCAL model is essentially the established minimal requirement for a distributed algorithm. Indeed, whenever a distributed algorithm is said to have running time T it is implied that, at the least, there exists a T -round algorithm in the LOCAL model.

In many settings, practical system design or physical constraints do not allow a node to contact all of its (potentially very large number of) neighbors at once. In this work we focus on this case and consider the GOSSIP model, which restricts each node to initiate at most one *bidirectional* communication with one of its neighbors per round. In contrast to computations in the LOCAL model, algorithms for the GOSSIP model have to decide which neighbor to contact in each round. This is particularly challenging when the network topology is unknown. Note that, as in the LOCAL model, messages sizes are unbounded (in fact, $O(n)$ for our purposes, where n is the size of the network) as these models reflect high-latency networks where round reduction is what counts. Furthermore, while in GOSSIP a node may end up communicating with many neighbors in a single step, every node “pays” for only one connection: the one they initiated.

Algorithms with such *gossip constraints* have been intensively studied for the so-called RUMOR problem (also known as the *rumor spreading* or *information dissemination* problem), in which each node has some initial input and is required to collect the information of all other nodes. Most previous papers analyzed the simple `UniformGossip` algorithm, which chooses a random neighbor to contact in each round. The uniform gossip mixes well on well-connected graphs, and good bounds for its convergence in terms of the graph conductance have been given [21, 81, 45]. For regular graphs, bounds in terms of vertex expansion are known as well [95]. In general, the gossip has a tendency to repeatedly communicate between well-connected neighbors while not transmitting information across bottlenecks. Only recently have algorithms been designed that try to avoid this behavior. By alternating between random and deterministic choices, [20] showed that fast convergence can be achieved for a wider family of graphs, namely, those which have large *weak conductance* (a notion defined therein). However, while this outperformed existing techniques in many cases, its running time bound still depended on a notion of the connectivity of the graph.

3.1.2 Information spreading and model reductions

Our contribution in this chapter significantly improves upon previous algorithms by providing the first information spreading algorithm for the GOSSIP model that is fast for *all* graphs, with no dependence on their conductance. Our algorithm requires at most $O(D + \text{polylog}(n))$ rounds in a network of size n and diameter D . This is at most an additive polylogarithmic factor from the trivial lower bound of $\Omega(D)$ rounds even for the LOCAL model. In contrast, there are many graphs with polylogarithmic diameter on which all prior algorithms have $\Omega(n)$ bounds.

In addition, our results apply more generally to any algorithm in the LOCAL model. We show how any algorithm that takes T time in the LOCAL model can be simulated in the GOSSIP model in $O(T + \text{polylog}(n))$ time, thus incurring only an additional cost which is polylogarithmic in the size of the network n . Our main result that leads to this simulation is an algorithm for the GOSSIP model in which each node exchanges information (perhaps indirectly) with each of its neighbors within a polylogarithmic number of rounds. This holds for every graph, despite the possibility of large degrees. A key ingredient in this algorithm is a recursive decomposition of graphs into clusters of sufficiently large conductance, allowing fast (possibly indirect) exchange of information between nodes inside clusters. The decomposition guarantees that the number of edges between pairs of nodes that did not exchange information decreases by a constant fraction. To convert the multiplicative polylogarithmic overhead for each simulated round into the additive overhead in our final simulation result we show connections between sparse graph spanners and algorithms in the GOSSIP model. This allows us to simulate known constructions of nearly-additive sparse spanners [88], which then in turn can be used in our simulations for even more efficient communication.

3.1.3 Techniques

The key step in our approach is to devise a distributed subroutine in the GOSSIP model to efficiently simulate one round of the LOCAL model by a small number of GOSSIP rounds. In particular, the goal is to deliver each node's current messages to all of its neighbors, which we refer to as the NEIGHBOREXCHANGE problem. Indeed, we exhibit such an algorithm, called **Superstep**, which requires at most $O(\log^3 n)$ rounds in the GOSSIP model for *all* graphs:

Theorem 3.1.1. *The Superstep algorithm solves NEIGHBOREXCHANGE in the GOSSIP model in $O(\log^3 n)$ rounds.*

Our design for the **Superstep** algorithm was inspired by ideas from [20] and started with an attempt to analyze the following very natural algorithm for the NEIGHBOREXCHANGE problem: In each round each node contacts a random neighbor whose message is not yet known to it. While this algorithm works well on most graphs, there exist graphs on which it requires a long time to complete due to asymmetric propagation of messages. We give an explicit example and discuss this issue in Section 3.6.

The **Superstep** algorithm is simple and operates by repeatedly performing $\log^3 n$ rounds of the **UniformGossip** algorithm while eliminating some edges after each round. During a round,

UniformGossip has each node choose a random neighbor to contact and exchange messages for a few steps, followed by a reversal of the message exchanges to maintain symmetry. From [21] or its strengthening [45], it is known that all pairs of vertices (and in particular all pairs of neighbors) that lie inside a high-conductance subset of the underlying graph exchange each other's messages within a single iteration. An existential graph decomposition result, given in Corollary 3.3.4, shows that for any graph there is a decomposition into high-conductance clusters with at least a constant fraction of intra-cluster edges. This implies that the number of remaining message exchanges required decreases by a constant factor in each iteration, which results in a logarithmic number of iterations until **NEIGHBOREXCHANGE** is solved.

This gives a simple algorithm for solving the **RUMOR** problem, which requires all nodes to receive the messages of all other nodes: By iterating **Superstep** D times, where D is the diameter of the network, one obtains an $O(D \cdot \log^3 n)$ round algorithm. This is at most an $O(\log^3 n)$ -factor slower than the trivial diameter lower bound and is a drastic improvement compared to prior upper bounds [20, 81, 21, 45], which can be of order $O(n)$ even for networks with constant or logarithmic D .

Beyond the **RUMOR** problem, it is immediate that the **NEIGHBOREXCHANGE** problem bridges the gap between the **LOCAL** and **GOSSIP** models in general. Indeed, we can simply translate a single round of a **LOCAL** algorithm into the **GOSSIP** model by first using any algorithm for **NEIGHBOREXCHANGE** to achieve the local broadcast and then performing the same local computations. We call this a simulation and more generally define an $(\alpha(G), \beta(G))$ -*simulator* as a transformation that takes any algorithm in the **LOCAL** model that runs in $T(G)$ rounds if the underlying topology is G , and outputs an equivalent algorithm in the **GOSSIP** model that runs in $O_n(\alpha(G)) \cdot T(G) + O_n(\beta(G))$ rounds. Thus, the simulation based on the **Superstep** algorithm gives a $(\log^3 n, 0)$ -simulator.

In many natural graph classes, like graphs with bounded genus or excluded minors, one can do better. Indeed we give a simple argument that on any (sparse) graph with *hereditary density* δ there is a schedule of direct message exchanges such that **NEIGHBOREXCHANGE** is achieved in 2δ rounds. Furthermore an order-optimal schedule can be computed in $\delta \log n$ rounds of the **GOSSIP** model even if δ is not known. This leads to a $(\delta, \delta \log n)$ -simulator.

Another way to look at this is that communicating over any hereditary sparse graph remains fast in the **GOSSIP** model. Thus, for a general graph, if one knows a sparse subgraph that has short paths from any node to its neighbors, one can solve the **NEIGHBOREXCHANGE** problem by communicating via these paths. Such graphs have been intensely studied and are known as spanners. We show interesting connections between simulators and spanners. For one, any fast algorithm for the **NEIGHBOREXCHANGE** problem induces a sparse low-stretch spanner. The **Superstep** algorithm can thus be seen as a new spanner construction in the **GOSSIP** model with the interesting property that the total number of messages used is at most $O(n \log^3 n)$. To our knowledge this is the first such construction. This also implies that, in general, **NEIGHBOREXCHANGE** requires a logarithmic number of rounds (up to $\log \log n$ factors perhaps) in the **GOSSIP** model. Considering in the other direction, we show that any fast spanner construc-

tion in the LOCAL model can be used to further decrease the multiplicative overhead of our $(\log^3 n, 0)$ -simulator. Applying this insight to several known spanner constructions [27, 87, 36, 88] leads to our second main theorem:

Theorem 3.1.2. *Every algorithm in the LOCAL model which completes in $T = T(G)$ rounds when run on the topology G can be simulated in the GOSSIP model in*

$$O(1) \cdot \min \left\{ T \cdot \log^3 n, T \cdot 2^{\log^* n} \log n + \log^4 n, T \cdot \log n + 2^{\log^* n} \log^4 n, T + \log^{O(1)} n, T \cdot \delta + \delta \log n, T \cdot \Delta \right\}$$

rounds, where n is the number of nodes, Δ the maximum degree and δ the hereditary density of G .

When we apply this result to the greedy algorithm for the RUMOR problem, where $T = D$, we obtain an algorithm whose $O(D + \text{polylog} n)$ rounds are optimal up to the additive polylogarithmic term, essentially closing the gap to the known trivial lower bound of $\Omega(D)$.

3.1.4 Related Work

The problem of spreading information in a distributed system was introduced by Demers et al. [26] for the purpose of replicated database maintenance, and it has been extensively studied thereafter.

One fundamental property of the distributed system that affects the number of rounds required for information spreading is the communication model. The *random phone call* model was introduced by Karp et al. [60], allowing every node to contact one other node in each round. In our setting, this corresponds to the complete graph. This model alone received much attention, such as in bounding the number of calls [30], bounding the number of random bits used [46], bounding the total number of bits [41], and more.

The number of rounds it takes to spread information for the randomized algorithm `UniformGossip`, in which every node chooses its communication partner for the next round uniformly at random from its set of neighbors, was analyzed in terms of the conductance of the underlying graph by Mosk-Aoyama and Shah [81], by Chierichetti et al. [21], and later by Giakkoupis [45], whose work currently has the best bound in terms of conductance, of $O(\frac{\log n}{\Phi(G)})$ rounds, with high probability.

Apart from the uniform randomized algorithm, additional algorithms were suggested for spreading information. We shortly overview some of these approaches. Doerr et al. [32] introduce *quasi-random* rumor spreading, in which a node chooses its next communication partner by deterministically going over its list of neighbors, but the starting point of the list is chosen at random. Results are $O(\log n)$ rounds for a complete graph and the hypercube, as well as improved complexities for other families of graphs compared to the randomized rumor spreading algorithm with uniform distribution over neighbors. This was followed by further analysis of the quasi-random algorithm [33, 40]. A hybrid algorithm, alternating between deterministic and randomized choices [20], was shown to achieve information spreading in $O(c(\frac{\log n}{\Phi_c(G)} + c))$

round, w.h.p., where $\Phi_c(G)$ is the *weak conductance* of the graph, a measure of connectivity of subsets in the graph. Distance-based bounds were given for nodes placed with uniform density in R^d [63, 64], which also address gossip-based solutions to specific problems such as resource location and minimum spanning tree. Doerr et al. [31] have recently presented an algorithm for fast information spreading in preferential attachment graphs, which model social networks.

The LOCAL model of communication, where each node communicates with each of its neighbors in every round, was formalized by Peleg [86]. Information spreading in this model requires a number of rounds which is equal to the diameter of the communication graph. Many other distributed tasks have been studied in this model, and below we mention a few in order to give a sense of the variety of problems studied. These include computing maximal independent sets [8], graph colorings [9], computing capacitated dominating sets [69], general covering and packing problems [70], and general techniques for distributed symmetry breaking [96].

Our algorithm **Superstep** implicitly constructs a sparsifier (sparse subgraph) of G that itself is a graph that has low vertex degree and hence supports fast **UniformGossip** whose runtime depends mainly on the diameter (module some log-factors), which essentially meets the lower bound. It is worth noting a related, but different, observation of [22] that the Spielman-Teng sparsifier has roughly the same **UniformGossip** runtime as that of the original graph.

3.2 Gossip and conductance

3.2.1 The uniform gossip algorithm

The `UniformGossip` algorithm is a common algorithm for RUMOR. (It is also known as the PUSH-PULL algorithm in some papers, such as [45].) Initially, each vertex u has some message M_u . At each step, every vertex chooses a random incident edge (u, v) at which point u and v exchange all messages currently known to them. The process stops when all vertices know everyone's initial messages. In order to treat this process formally, for any fixed vertex v and its message M_v , we treat the set of vertices that know M_v as a set that evolves probabilistically over time, as we explain next.

We begin by fixing an ambient graph $G = (V, E)$, which is unweighted and directed. The `UniformGossip` process is a Markov chain over 2^V , the set of vertex subsets of G . Given a current state $S \subseteq V$, one transition is defined as follows. Every vertex u picks an incident outgoing edge $a_u = (u, w) \in E$ uniformly at random from all such candidates. Let us call the set of all chosen edges $A = \{a_u : u \in V\}$ an *activated set*. Further let $A^\circ = \{(u, w) : (u, w) \in A \text{ or } (w, u) \in A\}$ be the symmetric closure of A . The new state of the chain is given by $S \cup B$, where by definition a vertex v is in the *boundary set* B if and only if there exists $u \in S$ such that $(u, v) \in A^\circ$. Note that V is the unique absorbing state, assuming a non-empty start.

We say that an edge (u, w) is *activated* if $(u, w) \in A^\circ$. If we let S model the set of nodes in possession of the message M_v of some fixed vertex v and we assume bidirectional message exchange along activated edges, the new state $S \cup B$ (of the Markov process) actually describes the set of nodes in possession of the message M_v after one distributed step of the `UniformGossip` algorithm.

Consider a τ -step Markov process K , whose activated sets at each step are respectively A_1, \dots, A_τ . Let the *reverse* of K , written K^{rev} , be the τ -step process defined by the activated sets A_τ, \dots, A_1 , in this order. For a process K , let $K(S)$ denote the end state when started from S .

Without loss of generality, for our analysis we will assume that only a single “starting” vertex s has an initial message M_s . We will be interested in analyzing the number of rounds of `UniformGossip` that ensure that all other vertices learn M_s , which we call the *broadcast time*. Clearly, when more than one vertex has an initial message, the broadcast time is the same since all messages are exchanged in parallel.

Lemma 3.2.1 (Reversal Lemma). *If $u \in K(\{w\})$, then $w \in K^{\text{rev}}(\{u\})$.*

Proof. The condition $u \in K(\{w\})$ holds if and only if there exists a sequence of edges $(e_{i_1}, \dots, e_{i_r})$ such that $e_{i_j} \in A_{i_j}^\circ$ for all j , the indices are increasing in that $i_1 < \dots < i_r$, and the sequence forms a path from w to u . The presence of the reversed sequence in K^{rev} implies $w \in K^{\text{rev}}(\{u\})$. \square

In communication terms, the lemma says that if u receives a message originating at w after τ

rounds determined by K , then w will receive a message originating at u after τ rounds determined by K^{rev} .

3.2.2 Conductance

The notion of *graph conductance* was introduced by Sinclair [97]. We require a more general version, which we introduce here. We begin with the requisite notation on edge-weighted graphs. We assume that each edge (u, v) has a weight $w_{uv} \in [0, 1]$. For an unweighted graph $G = (V, E)$ and any $u, v \in V$, we define $w_{uv} = 1$ if $(u, v) \in E$ and $w_{uv} = 0$ otherwise. Now we set $w(S, T) = \sum_{u \in S, v \in T} w_{uv}$. Note that in this definition it need not be the case that $S \cap T = \emptyset$, so, e.g., $w(S, S)$, when applied to an unweighted graph, counts every edge in S twice. The volume of a set $S \subseteq V$ with respect to V is written as $\text{vol}(S) = w(S, V)$. Sometimes we will have different graphs defined over the same vertex set. In such cases, we will write the identity of the graph as a subscript, as in $\text{vol}_G(S)$, in order to clarify which is the ambient graph (and hence the ambient edge set). Further, we allow self-loops at the vertices. A single loop at v of weight α is modeled by setting $w_{vv} = 2\alpha$, because both ends of the edge contribute α .

For a graph $G = (V, E)$ and a cut (S, T) where $S, T \subseteq V$ and $S \cap T = \emptyset$ (but where $T \cup S$ does not necessarily equal all of V), the *cut conductance* is given by

$$(3.2.1) \quad \varphi(S, T) = \frac{w(S, T)}{\min \{ \text{vol}_G(S), \text{vol}_G(T) \}}.$$

For a subset $H \subseteq V$ we need to define the *conductance of H (embedded) in V* . We will use this quantity to measure how quickly the `UniformGossip` algorithm proceeds in H , while accounting for the fact that edges in $(H, V - H)$ may slow down the process. The conductance of H in G is defined by

$$(3.2.2) \quad \Phi(H) = \min_{S \subseteq H} \varphi(S, H - S)$$

Note that the classical notion of conductance of G (according to Sinclair [97]) equals $\Phi(V)$ in our notation. When we want to emphasize the ambient graph G within which H resides, we will write $\Phi_G(H)$.

A few arguments in this chapter will benefit from the notion of a “strongly induced” graph of a vertex subset of an ambient graph G .

Definition 3.2.2. Let $U \subseteq V$ be a vertex subset of G . The *strongly induced* graph of U in G is a (new) graph H with vertex set U , whose edge weight function $h : U \times U \rightarrow \mathbf{R}$ is defined by

$$h_{uv} = \begin{cases} w_{uv}, & \text{if } u \neq v, \\ w_{uu} + \sum_{x \in V - U} w_{ux}, & \text{if } u = v. \end{cases}$$

Note that by construction we have $\Phi_H(U) = \Phi_G(U)$. The significance of this notion is the fact that the Markov process, describing the vertex set in possession of some message M_s for a starting vertex $s \in U$ in the `UniformGossip` algorithm executed on the strongly induced H , behaves identically to the respective process in G observed only on U . In particular, this definition allows us to use Theorem 1 of [45] in the following form:

Theorem 3.2.3. *For any graph $G = (V, E)$ and a subgraph $U \subseteq V$ and any start vertex in U , the broadcast time of the `UniformGossip` algorithm on U is $O(\Phi_G(U)^{-1} \log n)$ rounds w.h.p.*

3.3 Neighbor Exchange in $O(\log^3 n)$ rounds

The idea behind our algorithm for solving the NEIGHBOREXCHANGE problem is as follows. For every graph there exists a partition into clusters whose conductance is high, and therefore the UniformGossip algorithm allows information to spread quickly in each cluster. The latter further implies that pairs of neighbors inside a cluster exchange their messages quickly (perhaps indirectly). What remains is to exchange messages across inter-cluster edges. This is done recursively. In the following subsection we describe the conductance decomposition and then in Subsection 3.3.2 we give the details for the algorithm together with the proof of correctness.

3.3.1 Conductance decomposition of a graph

As described, our first goal is to partition the graph into clusters with large conductance. The challenge here is to do so while limiting the number of inter-cluster edges, so that we can efficiently apply this argument recursively. (Otherwise, this could be trivially done in any graph, for example by having each node as a separate cluster.) We are going to achieve this in the following lemma whose proof (found in Appendix ??) is very similar to that of Theorem 7.1 in [101]. Note that for our eventual algorithm, we are only going to need an existential proof of this clustering and not an actual algorithm for finding it.

Lemma 3.3.1. *Let $S \subseteq V$ be of maximum volume such that $\text{vol}(S) \leq \text{vol}(V)/2$ and $\varphi(S, V - S) \leq \xi$, for a fixed parameter $\xi \geq \Phi(G)$. If $\text{vol}(S) \leq \text{vol}(V)/4$, then $\Phi(V - S) \geq \xi/3$.*

Lemma 3.3.1 says that if a graph has no sparse balanced cuts, then it has a large subgraph which has no sparse cuts. The following corollary establishes that Lemma 3.3.1 holds even in the case when the ambient graph is itself a subgraph of a larger graph.

Proof. Assume, towards a contradiction, that $\Phi(V - S) < \xi/3$. Then, there exists a cut (P, Q) of $V - S$ with $\varphi(P, Q) < \xi/3$ and specifically

$$(3.3.1) \quad \max \left\{ \frac{w(P, Q)}{\text{vol}(P)}, \frac{w(Q, P)}{\text{vol}(Q)} \right\} \leq \frac{\xi}{3}$$

Henceforth, let Q be the smaller of the two, i.e. $\text{vol}(Q) \leq \text{vol}(V - S)/2$.

We are going to show that $\varphi(S \cup Q, P) \leq \xi$ and either $S \cup Q$ or P should have been chosen instead of S .

Consider the case $\text{vol}(S \cup Q) \leq \text{vol}(V)/2$. In this case,

$$\begin{aligned} \varphi(S \cup Q, P) &= \frac{w(S, P) + w(Q, P)}{\text{vol}(S \cup Q)} = \frac{w(S, P) + w(Q, P)}{\text{vol}(S) + \text{vol}(Q)} \\ &\leq \max \left\{ \frac{w(S, P)}{\text{vol}(S)}, \frac{w(Q, P)}{\text{vol}(Q)} \right\} \leq \max \left\{ \frac{w(S, P) + w(S, Q)}{\text{vol}(S)}, \frac{w(Q, P)}{\text{vol}(Q)} \right\} \leq \max \{ \xi, \xi/3 \} = \xi \end{aligned}$$

This establishes a contradiction, because $\varphi(S \cup Q, P) \leq \xi$ and $\text{vol}(S) < \text{vol}(S \cup Q) \leq \text{vol}(V)/2$.

Now let's consider the case $\text{vol}(S \cup Q) > \text{vol}(V)/2$. First, we argue that $\text{vol}(S \cup Q)$ cannot be too large. We use that $\text{vol}(Q) \leq \frac{1}{2} \text{vol}(V - S) = \frac{1}{2}(\text{vol}(V) - \text{vol}(S))$.

$$(3.3.2) \quad \text{vol}(S \cup Q) = \text{vol}(S) + \text{vol}(Q) \leq \text{vol}(S) + \frac{\text{vol}(V) - \text{vol}(S)}{2} = \frac{\text{vol}(V) + \text{vol}(S)}{2} \leq \frac{5}{8} \text{vol}(V)$$

Hence, $\text{vol}(P) \geq \frac{3}{8} \text{vol}(V)$. In addition, for the cut size, we have

$$\begin{aligned} w(S \cup Q, P) &= w(S, P) + w(Q, P) \\ &\leq \xi \text{vol}(S) + \frac{\xi}{3} \text{vol}(Q) \\ &\leq \xi \text{vol}(S) + \frac{\xi}{3} \frac{\text{vol}(V) - \text{vol}(S)}{2} \\ &\leq \frac{5}{6} \xi \text{vol}(S) + \frac{1}{6} \xi \text{vol}(V) \\ &= \frac{3}{8} \xi \text{vol}(V) \end{aligned}$$

And now we can bound the cut conductance:

$$(3.3.3) \quad \varphi(S \cup Q, P) = \frac{w(S \cup Q, P)}{\text{vol}(P)} \leq \frac{\frac{3}{8} \xi \text{vol}(V)}{\frac{3}{8} \text{vol}(V)} = \xi$$

This also establishes a contradiction because $\varphi(S \cup Q, P) \leq \xi$ while $\text{vol}(S) \leq \frac{1}{4} \text{vol}(V) < \frac{3}{8} \text{vol}(V) \leq \text{vol}(P) \leq \frac{1}{2} \text{vol}(V)$. \square

Corollary 3.3.2. *Let $U \subseteq V$ and let $S \subseteq U$ be of maximum volume such that $\text{vol}(S) \leq \text{vol}(U)/2$ and $\varphi(S, U - S) \leq \xi$, for a fixed parameter $\xi \geq \Phi(U)$. If $\text{vol}(S) \leq \text{vol}(U)/4$, then $\Phi(U - S) \geq \xi/3$.*

Proof. Observe that the proof of Lemma 3.3.1 holds when the graph has loops, i.e. $w_{uu} \neq 0$ for some u 's. Let H be the strongly induced graph of U . It follows from the definition that for any two disjoint sets $A, B \subseteq U$ we have $\text{vol}_G(A) = \text{vol}_H(A)$ and $w(A, B) = h(A, B)$. We can therefore apply Lemma 3.3.1 to H and deduce that the statement holds for the respective sets in G . \square

We are now ready to state and analyze the strong clustering algorithm. We emphasize that this is not a distributed algorithm, but an algorithm that only serves as a proof of existence of the partition. First, consider the following subroutine:

Cluster(G, U, ξ):

The inputs are a graph $G = (V, E)$, a subset $U \subseteq V$ and a parameter $0 < \xi < 1$.

1. Find a subset $S \subseteq U$ of maximum volume such that $\text{vol}(S) \leq \text{vol}(U)/2$ and $\varphi(S, U - S) \leq \xi$.
2. If no such S exists, then stop and output a single cluster $\{U\}$. Otherwise,
- 3a. If $\text{vol}(S) \leq \text{vol}(U)/4$, output $\{U - S\} \cup \text{Cluster}(G, S, \xi)$.
- 3b. If $\text{vol}(S) > \text{vol}(U)/4$, output $\text{Cluster}(G, S, \xi) \cup \text{Cluster}(G, U - S, \xi)$.

The clustering algorithm for a graph $G = (V, E)$ is simply a call to $\text{Cluster}(G, V, \xi)$. The following theorem analyses it.

Theorem 3.3.3. *For every $0 < \zeta < 1$, every graph $G = (V, E)$ with edge weights $w_{uv} \in \{0\} \cup [1, +\infty)$ has a partition $V = V_1 \cup \dots \cup V_k$ such that $\Phi(V_i) \geq \frac{\zeta}{\log_{4/3} \text{vol}(V)}$, for all i , and $\sum_{i < j} w(V_i, V_j) \leq \frac{3\zeta}{2} \text{vol}(V)$.*

Proof. The depth K of the recursion is, by construction, at most $\log_{4/3} \text{vol}(V)$ assuming that the smallest non-zero weight is 1. Let $\mathcal{R}_i \subseteq 2^V$ be a collection of the U -parameters of invocations of Cluster at depth $0 \leq i \leq K$ of the recursion. (So, for example, $\mathcal{R}_0 = \{V\}$.) For a set U let $S(U)$ be the small side of the cut produced by $\text{Cluster}(G, U, \xi)$, or \emptyset if no eligible cut was found. We can then bound the total weight of cut edges as

$$\begin{aligned} \sum_{0 \leq i \leq K} \sum_{U \in \mathcal{R}_i} w(S(U), U - S(U)) &\leq \sum_{0 \leq i \leq K} \sum_{U \in \mathcal{R}_i} \xi \text{vol}(S(U)) \leq \sum_{0 \leq i \leq K} \sum_{U \in \mathcal{R}_i} \frac{\xi}{2} \text{vol}(U) \\ &\leq \frac{\xi}{2} \sum_{0 \leq i \leq K} \sum_{U \in \mathcal{R}_i} \text{vol}(U) \leq \frac{\xi}{2} \sum_{0 \leq i \leq K} \text{vol}(V) \leq \frac{\xi \log_{4/3} \text{vol}(V)}{2} \text{vol}(V), \end{aligned}$$

Where we use the convention $w(\emptyset, S) = 0$. If we set $\xi = \frac{3\zeta}{\log_{4/3} \text{vol}(V)}$, for some $0 < \zeta < 1$, then Corollary 3.3.2 establishes the theorem. \square

In our exposition, we are going to use the following specialization of this theorem, obtained by plugging in $\zeta = 1/3$:

Corollary 3.3.4. *Every unweighted graph on m edges has a clustering that cuts at most $\frac{m}{2}$ edges and each cluster has conductance at least $\frac{1}{3 \log_{4/3} 2m}$.*

3.3.2 The Superstep algorithm for the Neighbor Exchange Problem

In this section, we will describe the **Superstep** algorithm, which solves the **NEIGHBOR EXCHANGE** problem. Recall that, for this problem, all vertices v are assumed to possess an initial message M_v , and the goal is for every pair of neighbors to know each other's initial messages.

We now describe our communication protocol, which specifies a local, per-vertex rule that tells a node which edge to choose for communication at any given round. It is assumed that

Superstep(G, τ):

The parameter $G = (V, E)$ is an unweighted, undirected graph, and τ is a positive integer. Set $F_0 := \vec{E}$ and $i := 0$. While $F_i \neq \emptyset$, repeat:

1. (First half)
 - 1a. Initialize every vertex v with a new auxiliary message $a(v)$, unique to v . (This message is added to the set of initial messages that v happens to know currently.)
 - 1b. Perform the **UniformGossip** algorithm with respect to F_i for τ rounds. And denote the outcome of the random activated edge choices by K_i
 - 1c. For every vertex u and neighbor w , let X_{uw} be the indicator that u received $a(w)$
2. (Second half)
 - 2a. Initialize every vertex v with a fresh auxiliary message $b(v)$, unique to v
 - 2b. Perform K_i^{rev} , the reverse process of the one realized in Step 1b
 - 2c. For every vertex u and neighbor w , let Y_{uw} be the indicator that u received $b(w)$
3. (Pruning) Compute the set of pruned directed edges $P_i = \{(u, w) : X_{uw} + Y_{uw} > 0\}$
4. Set $F_{i+1} := F_i - P_i$ and $i := i + 1$

Figure 3-1: Code for **Superstep** algorithm. It is easily verified that the above algorithm can be implemented in the GOSSIP model of communication.

the node will greedily transmit all messages known to it whenever an edge is chosen for communication. The protocol described here will employ some auxiliary messages, which are needed exclusively for its internal workings.

The **Superstep** subroutine described in Figure 3-1 is designed to ensure that, after a single invocation, all neighbors (u, w) in an undirected graph G have exchanged each other's initial messages. Clearly then, D invocations of **Superstep**, where D is the diameter of G , ensure that a message starting at vertex v reaches all $u \in V$, and this holds for all messages. D invocations of **Superstep** thus resolve the RUMOR problem.

Theorem 3.3.5. *Let $G = (V, E)$ be an undirected, unweighted graph with $|V| = n$ and $|E| = m$. Then, after one invocation of **Superstep**(G, τ), where $\tau = \Theta(\log^2 m)$, the following hold with probability $1 - 1/n^{\Omega(1)}$:*

- (i) Every pair of neighbors $\{u, w\} \in E$ receive each other's messages.
- (ii) The algorithm performs $\Theta(\log^3 m)$ distributed rounds.

Finally, our main result, Theorem 3.1.1, follows as a corollary of Theorem 3.3.5.

Our proof of Theorem 3.3.5 is structured as follows. If E is a set of undirected edges, let $\vec{E} = \{(u, w) : \{u, w\} \in E\}$ be the corresponding directed graph. Let $\vec{E} = F_0, \dots, F_d = \emptyset$ be the respective edge sets of each iteration in **Superstep**. We are going to show that, with probability $1 - 1/n^{\Omega(1)}$, the following invariants are maintained at each iteration:

- (a) The directed edge set F_i is symmetric in the sense that $(u, w) \in F_i \Rightarrow (w, u) \in F_i$,
- (b) The size of F_i reduces by a constant factor at each iteration. Formally, $\text{vol}(F_{i+1}) \leq \frac{1}{2} \text{vol}(F_i)$, and
- (c) After the i -th iteration, for every $(u, w) \in \vec{E} - F_{i+1}$, vertex u has received the message of vertex w and vice-versa.

Since $F_d = \emptyset$, claim (c) implies part (i) of Theorem 3.3.5. Claim (b) implies that the maximum number of iterations is $\log 2m$. Noting that every iteration entails 2τ distributed rounds, establishes part (ii) of Theorem 3.3.5.

Proof of Claim (a): Initially, F_0 is symmetric by construction. Inductively, assume that F_i is symmetric. The Reversal Lemma applied to K_i and K_i^{rev} implies $X_{uw} = Y_{wu}$, for all $u, w \in V$. This in turn implies that $X_{uw} + Y_{uw} = X_{wu} + Y_{wu}$, so P_i is symmetric. Since F_i is symmetric by hypothesis, we can conclude that $F_{i+1} = F_i - P_i$ is symmetric as well. \square

Proof of Claim (b): Consider the graph $G_i = (V, F_i)$ on the edge set F_i . Since F_i is symmetric, by Claim (a), we can treat G_i as undirected for the purposes of analyzing the **UniformGossip** algorithm. Let $V_1 \cup \dots \cup V_k$ be the decomposition of G_i promised by Corollary 3.3.4. (Note that the corollary holds for disconnected graphs, which may arise.) We thus have $\Phi(V_j) \geq \frac{1}{3 \log_{4/3} 2m}$, for all $1 \leq j \leq k$.

The choice $\tau = O(3 \log_{4/3} 2m \cdot \log m)$ ensures, via Theorem 3.2.3, that the first **UniformGossip** execution in every iteration mixes on all V_j with probability $1 - 1/n^{\Omega(1)}$. Mixing in V_j implies that for every internal edge (u, w) , where $u, w \in V_j$ and $(u, w) \in F_i$, the vertices (u, w) receive each other's auxiliary messages. The latter is summarized as $X_{uw} = X_{wu} = 1$. Applying the Reversal Lemma to the second execution of the **UniformGossip** algorithm, we deduce that $Y_{uw} = Y_{wu} = 1$ as well. These two equalities imply, by the definition of P_i , that P_i is a superset of the edges not cut by the decomposition $V_1 \cup \dots \cup V_k$. Equivalently, F_{i+1} is a subset of the cut edges. Corollary 3.3.4, however, bounds the volume of the cut edges by $\frac{1}{2} \text{vol}(F_i)$, which concludes the proof of Claim (b). \square

Proof of Claim (c): Initially, $\vec{E} - F_0 = \emptyset$ and so the claim holds trivially. By induction, the claim holds for edges in $\vec{E} - F_i$. And so it suffices to establish that u and v exchange their respective payload messages for all $(u, w) \in P_i$. However, this is equivalent to the conditions $X_{uw} + Y_{uw} > 0$, which are enforced by the definition of P_i . \square

3.4 Neighbor Exchange in hereditary sparse graphs

Next, we ask what can be achieved if instead of exchanging information indirectly as done in the Superstep algorithm, we exchange information only directly between neighbors. We will show in this section that this results in very simple deterministic algorithms for an important class of graphs that includes bounded genus graphs and all graphs that can be characterized by excluded minors [76, 67]. The results here will be used for the more general simulators in Section 3.5.

As before we will focus on solving the NEIGHBOREXCHANGE problem. One trivial way to solve this problem is for each node to contact its neighbors directly, e.g., by using a simple round robin method. This takes at most Δ time, where Δ is the maximum-degree of the network. However, in some cases direct message exchanges work better. One graph that exemplifies this is the star graph on n nodes. While it takes $\Delta = n$ time to complete a round robin in the center, after just a single round of message exchanges each leaf has initiated a bidirectional link to the center and thus exchanged its messages. On the other hand, scheduling edges cannot be fast on dense graphs with many more edges than nodes. The following lemma, whose proof appears in Appendix ??, shows that the *hereditary density* captures how efficient direct message exchanges can be on a given graph. Let the hereditary density δ of a graph G be the minimal integer such that for every subset of nodes S the subgraph induced by S has at most density δ , i.e., at most $\delta|S|$ edges.

Lemma 3.4.1. *The following holds for a graph G with hereditary density δ :*

1. *Any schedule of direct message exchanges that solves the NEIGHBOREXCHANGE problem on G takes at least δ rounds.*
2. *There exists a schedule of the edges of G such that each node needs only 2δ direct message exchanges to solve the NEIGHBOREXCHANGE problem.*

Proof. Since the hereditary density of G is δ , there is a subset of nodes $S \subseteq V$ with at least $\delta|S|$ edges between nodes in S . In each round, each of the $|S|$ nodes is allowed to schedule at most one message exchange, so a simple pigeonhole principle argument shows that at least one node needs to initiate at least δ message exchanges.

For the second claim, we are going to show that for any $\epsilon > 0$ there is an $O(\epsilon^{-1} \log n)$ -time deterministic distributed algorithm in the LOCAL model that assigns the edges of G to nodes such that each node is assigned at most $2(1 + \epsilon)\delta$ edges. Then setting $\epsilon < (3\delta)^{-1}$ makes the algorithm inefficient but finishes the existential proof. edge $\{u, v\}$ is assigned to u , we say that it has been *oriented* away from u , as in (u, v) .

The algorithm runs in phases in which, iteratively, a node takes responsibility for some of the remaining edges connected to it. All edges that are assigned are then eliminated and so are nodes that have no unassigned incident edges. In each phase, every node of degree at most $2(1 + \epsilon)\delta$ takes responsibility for all of its incident edges (breaking ties arbitrarily). At least a $1/(1 + \frac{1}{\epsilon})$ fraction of the remaining nodes fall under this category in every phase. This is because otherwise,

Set $\delta' = 1$ and $H = \emptyset$. H is the subset of neighbors in $\Gamma(v)$ that node v has exchanged messages with. Repeat:

```

 $\delta' = (1 + \epsilon)\delta'$ 
for  $O(\frac{1}{\epsilon} \cdot \log n)$  rounds do
  if  $|\Gamma(v) \setminus H| \leq \delta'$ 
    in the next  $\delta'$  rounds exchange messages with all neighbors in  $\Gamma(v) \setminus H$ 
    terminate
  else
    wait for  $\delta'$  rounds
  update  $H$ 

```

Figure 3-2: Code for `DirectExchange` algorithm.

the number of edges in the subgraph would be more than $(|S| - |S|/(1 + \frac{1}{\epsilon}))(2(1 + \epsilon)\delta)/2 = |S|\delta$, which would contradict the fact that the hereditary density of the graph equals δ . What remains after each phase is an induced subgraph which, by definition of the hereditary density, continues to have hereditary density at most δ . The number of remaining nodes thus decreases by a factor of $1 - 1/(1 + \frac{1}{\epsilon})$ in every phase and it takes at most $O(\log_{1+\frac{1}{\epsilon}} n)$ phases until no more nodes remain, at which point all edges have been assigned to a node. \square

We note that the lower bound of Lemma 3.4.1 is tight in all graphs, i.e., the upper bound of 2δ can be improved to δ . Graphs with hereditary density δ , also known as $(0, \delta)$ -sparse graphs, are thus exactly the graphs in which δ is the minimum number such that the edges can be oriented to form a directed graph with out-degree at most δ . This in turn is equivalent to the *pseudoarboricity* of the graph, i.e., the minimum number of pseudoforests needed to cover the graph. Due to the matroid structure of pseudoforests, the pseudoarboricity can be computed in polynomial time. For our purposes the (non-distributed) algorithms to compute these optimal direct message exchange schedule are too slow. Instead, we present a simple and fast algorithm, based on the `LOCAL` algorithm in Lemma 3.4.1, which computes a schedule that is within a factor of $2 + \epsilon$ of the optimal. We note that the `DirectExchange` algorithm presented here works in the `GOSSIP` model and furthermore does not require the hereditary density δ to be known *a priori*. The algorithm for an individual node v is given in Figure 3-2. Its properties are stated in Theorem 3.4.2.

Theorem 3.4.2. *For any constant $\epsilon > 0$, the deterministic algorithm `DirectExchange` solves the `NEIGHBOREXCHANGE` problem in the `GOSSIP` model using $O(\frac{\delta \log n}{\epsilon^2})$ rounds, where δ is the hereditary density of the underlying topology. During the algorithm, each node initiates at most $2(1 + \epsilon)^2\delta$ exchanges.*

Proof. Let δ be the hereditary density of the underlying topology. We know from the proof of Lemma 3.4.1 that the algorithm terminates during the for-loop if δ' is at least $2(1 + \epsilon)\delta$.

Thus, when the algorithm terminates, δ' is at most $2(1 + \epsilon)^2\delta$ which is also an upper bound on the number of neighbors contacted by any node. In the $(i + 1)^{\text{th}}$ -to-last iteration of the outer loop, δ' is at most $2(1 + \epsilon)^2\delta/(1 + \epsilon)^i$, and the running time for this phase is thus at most $2(1 + \epsilon)^2\delta/(1 + \epsilon)^i \cdot O(\frac{1}{\epsilon} \log n)$. Summing up over these powers of $1/(1 + \epsilon)$ results in a total of at most $\delta/((1 + \epsilon) - 1) \cdot O(\frac{1}{\epsilon} \log n) = O(\frac{\delta \log n}{\epsilon^2})$ rounds. \square

3.5 Simulators and graph spanners

In this section we generalize our results to arbitrary simulations of LOCAL algorithms in the GOSSIP model and point out connections to graph spanners, another well-studied subject.

Recall that we defined the NEIGHBOREXCHANGE problem exactly in such a way that it simulates in the GOSSIP model what is done in one round of the LOCAL model. With our solutions, an $O(\delta \log n)$ -round algorithm and an $O(\log^3 n)$ -round algorithm for the NEIGHBOREXCHANGE problem in the GOSSIP model, it is obvious that we can now easily convert any T -round algorithm for the LOCAL model to an algorithm in the GOSSIP model, e.g., by T times applying the Superstep algorithm. In the case of the DirectExchange algorithm we can do even better. While it takes $O(\delta \log n)$ rounds to compute a good scheduling, once it is known it can be reused and each node can simply exchange messages with the same $O(\delta)$ nodes without incurring an additional overhead. Thus, simulating the second and any further rounds can be easily done in $O(\delta)$ rounds in the GOSSIP model. This means that any algorithm that takes $O(T)$ rounds to complete in the LOCAL model can be converted to an algorithm that takes $O(\delta T + \delta \log n)$ rounds in the GOSSIP model. We call this a simulation and define simulators formally as follows.

Definition 3.5.1. An (α, β) -simulator is a way to transform any algorithm A in the LOCAL model to an algorithm A' in the GOSSIP model such that A' computes the same output as A and if A takes $O(T)$ rounds then A' takes at most $O(\alpha T + \beta)$ rounds.

Phrasing our results from Section 3.3.2 and Section 3.4 in terms of simulators we get the following corollary.

Corollary 3.5.2. *For a graph G of n nodes, hereditary density δ , and maximum degree Δ , the following hold: (a) There is a randomized $(\log^3 n, 0)$ -simulator; (b) There is a deterministic $(\Delta, 0)$ -simulator; (c) There is a deterministic $(2(1 + \epsilon)^2 \delta, O(\delta \epsilon^{-2} \log n))$ -simulator for any $\epsilon > 0$ or, simply, there is a $(\delta, \delta \log n)$ -simulator.*

Note that for computations that require many rounds in the LOCAL model the $(2(1 + \epsilon)^2 \delta, O(\delta \epsilon^{-2} \log n))$ -simulator is a $\log n$ -factor faster than repeatedly applying the DirectExchange algorithm. This raises the question whether we can similarly improve our $(\log^3 n, 0)$ -simulator to obtain a smaller multiplicative overhead for the simulation.

What we would need for this is to compute, e.g., using the Superstep algorithm, a schedule that can then be repeated to exchange messages between every node and its neighbors. What we are essentially asking for is a short sequence of neighbors for each node over which each node can indirectly get in contact with all its neighbors. Note that any such schedule of length t must at least fulfill the property that the union of all edges used by any node is connected (if the original graph G is connected) and even more that each node is connected to all its neighbors via a path of length at most t . Subgraphs with this property are called *spanners*. Spanners are well-studied objects, due to their extremely useful property that they approximately preserve distances while potentially being much sparser than the original graph. The quality of a spanner

is described by two parameters, its number of edges and its *stretch*, which measures how well it preserves distances.

Definition 3.5.3 (Spanners). A subgraph $S = (V, E')$ of a graph $G = (V, E)$ is called an (α, β) -*stretch spanner* if any two nodes u, v with distance d in G have distance at most $\alpha d + \beta$ in S .

From the discussion above it is also clear that any solution to the NEIGHBOREXCHANGE problem in the GOSSIP model also computes a spanner as a byproduct.

Lemma 3.5.4. *If A is an algorithm in the GOSSIP model that solves the NEIGHBOREXCHANGE problem in any graph G in T rounds then this algorithm can be used to compute a $(T, 0)$ -stretch spanner with hereditary density T in $O(T)$ rounds in the GOSSIP model.*

While there are spanners with better properties than the $(\log^3 n, 0)$ -stretch and $\log^3 n$ -density implied by Lemma 3.5.4 and Theorem 3.3.5, our construction has the interesting property that the number of messages exchanged during the algorithm is at most $O(n \log^3 n)$, whereas all prior algorithms rely on the broadcast nature of the LOCAL model and therefore use already $O(n^2)$ messages in one round on a dense graph. Lemma 3.5.4 furthermore implies a nearly logarithmic lower bound on the time that is needed in the GOSSIP model to solve the NEIGHBOREXCHANGE problem since a significantly sub-logarithmic simulator would imply the existence of a too good spanner:

Corollary 3.5.5. *For any algorithm in the GOSSIP model that solves the NEIGHBOREXCHANGE problem there is a graph G on n nodes on which this algorithm takes at least $\Omega(\frac{\log n}{\log \log n})$ rounds.*

Proof. Assume an algorithm takes at most $T(n)$ rounds on any graph with n nodes. The edges used by the algorithm form a $T(n)$ -stretch spanner with density $T(n)$, as stated in Lemma 3.5.4. For values of $T(n)$ which are too small it is known that such spanners do not exist [84]. More specifically it is known that there are graphs with n nodes, density at least $1/4n^{1/r}$ and girth r , i.e., the length of the smallest cycle is r . In such a graph any $(r - 2)$ -stretch spanner has to be the original graph itself, since removing a single edge causes its end-points to have distance at least $r - 1$, and thus the spanner also have density $1/4n^{1/r}$. Therefore $T(n) \geq \operatorname{argmin}_r \{r - 2, 1/4n^{1/r}\} = \Omega(\frac{\log n}{\log \log n})$. \square

Interestingly, it is not only the case that efficient simulators imply good spanners but the next theorem shows as a converse that good existing spanner constructions for the LOCAL model can be used to improve the performance of simulators.

Theorem 3.5.6. *If there is an algorithm that computes an (α, β) -stretch spanner with hereditary density δ in $O(T)$ rounds in the LOCAL model then this can be combined with an (α', β') -simulator to an $(\alpha\delta, T\alpha' + \beta' + \delta \log n + \delta\beta)$ -simulator.*

Proof. For simplicity we first assume that $\beta = 0$, i.e., the spanner S computed by the algorithm in the LOCAL model has purely multiplicative stretch α and hereditary density δ . Our strategy

is simple: We are first going to compute the good spanner by simulating the spanner creation algorithm from the LOCAL model using the given simulator. This takes $T\alpha' + \beta'$ rounds in the GOSSIP model. Once this spanner S is computed we are only going to communicate via the edges in this spanner. Note that for any node there is a path of length at most α to any of its neighbors. Thus if we perform α rounds of LOCAL-flooding rounds in which each node forwards all messages it knows of to all its neighbors in S each node obtains the messages of all its neighbors in G . This corresponds exactly to a NEIGHBOREXCHANGE in G . Therefore if we want to simulate T' rounds of an algorithm A in the LOCAL model on G we can alternatively perform $\alpha T'$ LOCAL computation rounds on S while doing the LOCAL computations of A every α rounds. This is a computation in the LOCAL model but on a sparse graph. We are therefore going to use the $(O(\delta), O(\delta \log n))$ -simulator from Corollary 3.5.2 to simulate this computation which takes $O(\delta\alpha T' + \delta \log n)$ rounds in the GOSSIP model. Putting this together with the $T\alpha' + \beta'$ rounds it takes to compute the spanner S we end up with $\delta\alpha T' + \delta \log n + T\alpha' + \beta'$ rounds in total.

In general (i.e., for $\beta > \alpha$) it is not possible (see, e.g., Corollary 3.5.5) to simulate the LOCAL algorithm step by step. Instead we rely on the fact that any LOCAL computation over T rounds can be performed by each node first gathering information of all nodes in a T -neighborhood and then doing LOCAL computations to determine the output. For this all nodes simply include all their initial knowledge (and for a randomized algorithm all the random bits they might use throughout the algorithm) in a message and flood this in T rounds to all node in their T -neighborhood. Because a node now knows all information that can influence its output over a T -round computation it can now locally simulate the algorithm for itself and its neighbors to the extend that its output can be determined. Having this we simulate the transformed algorithm as before: We first precompute S in $T\alpha' + \beta'$ time and then simulate the T' rounds of flooding in G by performing $\alpha T' + \beta$ rounds of LOCAL-flooding in S . Using the $(O(\delta), O(\delta \log n))$ -simulator this takes $O(\delta(\alpha T' + \beta) + \delta \log n)$ rounds in the GOSSIP model. \square

Corollary 3.5.7. *There is a $(2^{\log^* n} \log n, \log^4 n)$ -simulator, a $(\log n, 2^{\log^* n} \log^4 n)$ -simulator and a $(O(1), \text{polylog} n)$ -simulator.*

Proof. We are going to construct the simulators with increasingly better multiplicative overhead by applying Theorem 3.5.6 to existing spanner constructions [27, 87, 36, 88] for the LOCAL model. We first construct a $(\log^2 n, \log^4 n)$ -simulator by combining our new $(\log^3 n, 0)$ -simulator with the deterministic spanner construction in [27]. The construction in [27] takes $O(\log n)$ rounds in the LOCAL model and adds at most one edge to each node per round. Using $\alpha = T = \delta = O(\log n)$, $\alpha' = \log^3 n$ and $\beta = \beta' = 0$ in Theorem 3.5.6 gives the desired $(\log^2 n, \log^4 n)$ -simulator. Having this simulator, we can use [87] to improve the multiplicative overhead while keeping the additive simulation overhead the same. In [87] an $\alpha = (2^{\log^* n} \log n)$ -stretch spanner with constant hereditary density $\delta = O(1)$ is constructed in $T = O(2^{\log^* n} \log n)$ -time in the LOCAL model. Using these parameters and the $(\log^2 n, \log^4 n)$ -simulator in Theorem 3.5.6

leads to the strictly better $(2^{\log^* n} \log n, \log^4 n)$ -simulator claimed here. Having this simulator, we can use it with the randomized spanner construction in [36]. There, an α -stretch spanner, with $\alpha = O(\log n)$, is constructed in $T = O(\log^3 n)$ -time in the LOCAL model by extracting a subgraph with $\Omega(\log n)$ girth. Such a graph has constant hereditary density $\delta = O(1)$, as argued in [84]. Using these parameters and the $(2^{\log^* n} \log n, \log^4 n)$ -simulator in Theorem 3.5.6 leads to the $(\log n, 2^{\log^* n} \log^4 n)$ -simulator. Finally, we can use any of these simulators together with the nearly-additive $(5 + \epsilon, \text{polylog} n)$ -spanner construction from [88] to obtain our last simulator. It is easy to verify that the randomized construction named $AD^{\log \log n}$ in [88] can be computed in a distributed fashion in the LOCAL model in $\text{polylog} n$ time and has hereditary density $\delta = O(1)$. This together with any of the previous simulators and Theorem 3.5.6 results in a $(O(1), \text{polylog} n)$ -simulator. \square

With these various simulators it is possible to simulate a computation in the LOCAL model with very little (polylogarithmic) multiplicative or additive overhead in the GOSSIP model. Note that while the complexity of the presented simulators is incomparable, one can interleave their executions (or the executions of the simulated algorithms) and thus get the best runtime for any instance. This, together with Corollaries 3.5.7 and 3.5.2, proves our main result of Theorem 3.1.2.

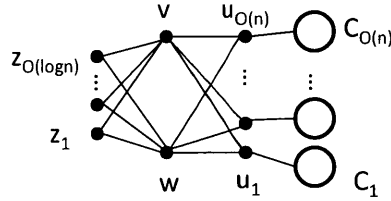


Figure 3-3: An example illustrating the behavior of the algorithm choosing a random neighbor whose information is still unknown.

3.6 Open problems

Our work presents a more efficient alternative to the `UniformGossip` algorithm that allows fast rumor spreading on all graphs, with no dependence on their conductance. We then show how this leads to fast simulation in the `GOSSIP` model of any algorithm designed for the `LOCAL` model by constructing sparse spanners. This work leaves some interesting directions for future work, which we discuss below.

3.6.1 Simplified Neighbor Exchange

First, as mentioned in the introduction, there are cases in which the algorithm where each node chooses a neighbor uniformly at random only from among those it has not yet heard from (directly or indirectly), performs slower than the optimal. This is illustrated in the following example.

Here is an example for a case in which the algorithm where each node chooses a neighbor uniformly at random only from among those it has not yet heard from (directly or indirectly), performs slower than the optimal. The graph appears in Figure 3-3, where C_i stands for a clique of size $O(1)$ in which every node is also connected to the node v_i .

In this example, it takes 2 rounds for the node w to hear about the node v (through nodes in $\{z_1, \dots, z_{O(\log n)}\}$). During these rounds there is a high probability that a constant fraction of the nodes in $\{u_1, \dots, u_{O(n)}\}$ did not yet hear from neither v nor w . With high probability, a constant fraction of these will contact w before contacting v , after which they will not contact v anymore because they will have heard from it through w . This leaves $O(n)$ nodes which v has to contact directly (since nodes in $\{z_1, \dots, z_{O(\log n)}\}$ are no longer active since they already heard from both of their neighbors), resulting in a linear number of rounds for `NEIGHBOREXCHANGE`.

We note, however, that this specific example can be solved by requiring nodes that have heard from all their neighbors to continue the algorithm after resetting their state, in the sense

that they now consider all their neighbors to be such that they have not heard from (this is only for the sake of choosing the next neighbor to contact, the messages they send can include previous information they received). Therefore, we do not rule out the possibility that this algorithm works well, but our example suggests that this may not be trivial to prove.

3.6.2 Asynchronous algorithm and soft decisions

Second, regarding our solution to the RUMOR problem, the `Superstep` algorithm, as presented, can be implemented in synchronous environments in a straightforward manner. To convert our algorithm to the asynchronous setting, one needs to synchronize the reversal step. Synchronization is a heavy-handed approach and not desirable in general.

To alleviate this problem, we believe, it is possible to get rid of the reversal step altogether. The basic idea is to do away with the hard decisions to “remove” edges once a message from a neighbor has been received. And instead to multiplicatively decrease the weight of such edges for the next round. This approach would introduce a slight asymmetry in each edge’s weight in both directions. In order to analyze such an algorithm, it is needed to understand the behavior of `RandomNeighbor` in the general asymmetric setting. In this setting, each vertex uses its own distribution over outgoing links when choosing a communication partner at each step. We believe that understanding the asymmetric `RandomNeighbor` is an open problem of central importance. It should be mentioned, in general, that solving RUMOR (and not just `Superstep`) asynchronously in the LOCAL model is a well-known, major open problem.

Bibliography

- [1] I. Abraham, C. Gavoille, and D. Malkhi. On space-stretch trade-offs: Lower bounds. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 207–216. ACM New York, NY, USA, 2006. 33
- [2] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup. Compact name-independent routing with minimum stretch. 2008. 33
- [3] Alon, Badoiu, Demaine, Farach-Colton, Hajiaghayi, and Sidiropoulos. Ordinal embeddings of minimum relaxation: General properties, trees, and ultrametrics. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 2005. 66
- [4] Noga Alon, Peter Frankl, and Vojtech Rödl. Geometrical realization of set systems and probabilistic communication complexity. In *FOCS*, pages 277–280. IEEE, 1985. 67
- [5] J. M. Alonso and et al. Notes on word hyperbolic groups. In *Group theory from a geometrical viewpoint (Trieste, 1990)*, pages 3–63. World Sci. Publ., River Edge, NJ, 1991. Edited by H. Short. 68
- [6] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '03*, pages 313–324, New York, NY, USA, 2003. ACM. 16
- [7] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, STOC '03*, pages 383–388, New York, NY, USA, 2003. ACM. 16
- [8] Leonid Barenboim and Michael Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. In *Proceedings of the twenty-seventh ACM Symposium on Principles of Distributed Computing (PODC)*, pages 25–34, New York, NY, USA, 2008. ACM. 91
- [9] Leonid Barenboim and Michael Elkin. Distributed $(\delta + 1)$ -coloring in linear (in δ) time. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 111–120, New York, NY, USA, 2009. ACM. 87, 91

- [10] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pages 184–, Washington, DC, USA, 1996. IEEE Computer Society. 22, 25, 59
- [11] Yair Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 161–168, New York, NY, USA, 1998. ACM. 25
- [12] Yair Bartal and Stefano Leonardi. On-line routing in all-optical networks. In *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, ICALP '97, pages 516–526, London, UK, 1997. Springer-Verlag. 22
- [13] Aditya Bhaskara and Aravindan Vijayaraghavan. Computing the matrix p-norm. *CoRR*, abs/1001.2613, 2010. 17
- [14] Marcin Bienkowski, Mirosław Korzeniowski, and Harald Räcke. A practical algorithm for constructing oblivious routing schemes. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, SPAA '03, pages 24–33, New York, NY, USA, 2003. ACM. 16
- [15] Bilu and Linial. Monotone maps, sphericity and bounded second eigenvalue. *JCTB: Journal of Combinatorial Theory, Series B*, 95, 2005. 66, 67
- [16] Bollobas, Riordan, Spencer, and Tusnady. The degree sequence of a scale-free random graph process. *RSA: Random Structures and Algorithms*, 18, 2001. 62
- [17] B. Bollobás and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, 2002. In press. 62
- [18] Prosenjit Bose, Pat Morin, Ivan Stojmenovic, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001. 62
- [19] Antonio Caruso, Stefano Chessa, S. De, and A. Urpi. GPS free coordinate assignment and routing in wireless sensor networks. In *INFOCOM*, pages 150–160. IEEE, 2005. 62
- [20] Keren Censor-Hillel and Hadas Shachnai. Fast information spreading in graphs with large weak conductance. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 440–448, 2011. 87, 88, 89, 90
- [21] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 399–408, 2010. 87, 89, 90
- [22] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumour spreading and graph conductance. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1657–1663, 2010. 91

- [23] F.R.K. Chung. *Spectral graph theory*. American Mathematical Society, 1997. 56
- [24] J. H. Conway and N. J. A. Sloane. *Sphere packings, lattices and groups*, volume 290 of *Grundlehren der math. Wissenschaften*. Springer, 1993. 82
- [25] H. S. M. Coxeter. *Non-Euclidean geometry*. MAA Spectrum. Mathematical Association of America, Washington, DC, sixth edition, 1998. 68
- [26] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, 1987. 90
- [27] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. On the locality of distributed sparse spanner construction. In *Proceedings of the twenty-seventh ACM Symposium on Principles of Distributed Computing (PODC)*, pages 273–282, New York, NY, USA, 2008. ACM. 90, 105
- [28] Raghavan Dhandapani. Greedy drawings of triangulations. *Submitted*, 2006. 62
- [29] Do Carmo, Manfredo. *Riemannian Geometry*. Birkhäuser Verlag, Boston, 1992. 68
- [30] Benjamin Doerr and Mahmoud Fouz. Asymptotically optimal randomized rumor spreading. *CoRR*, abs/1011.1868, 2010. 90
- [31] Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Social networks spread rumors in sublogarithmic time. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2011. 91
- [32] Benjamin Doerr, Tobias Friedrich, and Thomas Sauerwald. Quasirandom rumor spreading. In *Proceedings of the nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 773–781, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. 90
- [33] Benjamin Doerr, Tobias Friedrich, and Thomas Sauerwald. Quasirandom rumor spreading: Expanders, push vs. pull, and robustness. In *36th International Colloquium on Automata, Languages and Programming (ICALP)(1)*, pages 366–377, 2009. 90
- [34] J.R. Douceur. The sybil attack. In *Peer-To-Peer Systems: First International Workshop, Iptps 2002, Cambridge, Ma, USA, March 7-8, 2002, Revised Papers*, page 251. Springer, 2002. 57
- [35] P.G. Doyle and J.L. Snell. Random walks and electric networks. *Arxiv preprint math.PR/0001057*, 2000. 35
- [36] Devdatt Dubhashi, Alessandro Mei, Alessandro Panconesi, Jaikumar Radhakrishnan, and Aravind Srinivasan. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. *J. Comput. Syst. Sci.*, 71:467–479, November 2005. 90, 105, 106

- [37] Matthias Englert and Harald Räcke. Oblivious routing for the lp-norm. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science, FOCS '09*, pages 32–40, Washington, DC, USA, 2009. IEEE Computer Society. 22, 29
- [38] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, STOC '03*, pages 448–455, New York, NY, USA, 2003. ACM. 25
- [39] Rodrigo Fonseca, Sylvia Ratnasamy, David Culler, Scott Shenker, and Ion Stoica. Beacon vector routing: Scalable point-to-point in wireless sensor networks, July 09 2004. 62
- [40] Nikolaos Fountoulakis and Anna Huber. Quasirandom rumor spreading on the complete graph is as fast as randomized rumor spreading. *SIAM Journal on Discrete Mathematics*, 23(4):1964–1991, 2009. 90
- [41] Pierre Fraigniaud and George Giakkoupis. On the bit communication complexity of randomized rumor spreading. In *Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 134–143, New York, NY, USA, 2010. ACM. 90
- [42] Peter Frankl and Hiroshi Maehara. The johnson-lindenstrauss lemma and the sphericity of some graphs. *J. Comb. Theory, Ser. B*, 44(3):355–362, 1988. 67
- [43] Eric J. Friedman. Genericity and congestion control in selfish routing. In *In Proceedings of the 43rd Annual IEEE Conference on Decision and Control (CDC)*, pages 4667–4672, 2004. 22
- [44] Jie Gao, Leonidas J. Guibas, John Hershberger, Li Zhang, and An Zhu. Geometric spanner for routing in mobile networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 45–55, New York, NY, USA, 2001. ACM Press. 62
- [45] George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *28th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 57–68, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 87, 89, 90, 92, 94
- [46] George Giakkoupis and Philipp Woelfel. On the randomness requirements of rumor spreading. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 449–461, 2011. 90
- [47] N. Goyal, L. Rademacher, and S. Vempala. Expanders via random spanning trees. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 576–585. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2009. 33

- [48] M. Gromov. Hyperbolic groups. In *Essays in group theory*, volume 8 of *Math. Sci. Res. Inst. Publ.*, pages 75–263. Springer, New York, 1987. 68
- [49] Anupam Gupta. Embedding tree metrics into low dimensional euclidean spaces. In *STOC*, pages 694–700, 1999. 60, 82
- [50] Anupam Gupta, Mohammad T. Hajiaghayi, and Harald Räcke. Oblivious network design. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, SODA '06*, pages 970–979, New York, NY, USA, 2006. ACM. 20, 21, 22
- [51] Bernhard Haeupler and David Karger. Faster information dissemination in dynamic networks via network coding. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing, PODC '11*, pages 381–390, New York, NY, USA, 2011. ACM. 16
- [52] M.T. Hajiaghayi, R.D. Kleinberg, T. Leighton, and H. Räcke. New lower bounds for oblivious routing in undirected graphs. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 918–927. ACM New York, NY, USA, 2006. 34, 42
- [53] Harel and Tarjan. Fast algorithms for finding nearest common ancestors. *SICOMP: SIAM Journal on Computing*, 13, 1984. 71
- [54] Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures, SPAA '03*, pages 34–43, New York, NY, USA, 2003. ACM. 16, 25
- [55] P. Harsha, T.P. Hayes, H. Narayanan, H. Räcke, and J. Radhakrishnan. Minimizing average latency in oblivious routing. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 200–207. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2008. 33
- [56] Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, pages 369–384, 1991. 22
- [57] James Renegar. *A mathematical view of interior-point methods in convex optimization*. 2001. 32
- [58] S. Kale, Y. Peres, and C. Seshadhri. Noise Tolerance of Expanders and Sublinear Expander Reconstruction. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science-Volume 00*, pages 719–728. IEEE Computer Society Washington, DC, USA, 2008. 57
- [59] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MOBICOM*, pages 243–254, 2000. 62

- [60] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, page 565, Washington, DC, USA, 2000. IEEE Computer Society. 90
- [61] Svetlana Katok. *Fuchsian groups*. Chicago Lectures in Mathematics. University of Chicago Press, Chicago, IL, 1992. 68, 69
- [62] Jonathan Kelner and Petar Maymounkov. Electric routing and concurrent flow cutting. *Theor. Comput. Sci.*, 412:4123–4135, July 2011. 22, 28, 29
- [63] David Kempe, Jon Kleinberg, and Alan Demers. Spatial gossip and resource location protocols. In *Proceedings of the thirty-third Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–172, New York, NY, USA, 2001. ACM. 91
- [64] David Kempe and Jon M. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 471–480, Washington, DC, USA, 2002. IEEE Computer Society. 91
- [65] Jon Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 163–170, New York, NY, USA, 2000. ACM. 16
- [66] Robert Kleinberg. Geographic routing using hyperbolic space. *To appear in Proc. 32nd IEEE INFOCOM (INFOCOM 2007)*, 2007. 60, 62, 68, 70, 71, 73
- [67] A. Kostochka. Lower bound of the hadwiger number of graphs by their average degree. *Combinatorica*, 4:307–316, 1984. 10.1007/BF02579141. 100
- [68] Robert Krauthgamer and James Lee. Algorithms on negatively curved spaces. In *FOCS*, 2006. 68
- [69] Fabian Kuhn and Thomas Moscibroda. Distributed approximation of capacitated dominating sets. *Theor. Comp. Sys.*, 47:811–836, November 2010. 91
- [70] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proceedings of the seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 980–989, New York, NY, USA, 2006. ACM. 91
- [71] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *PODC*, pages 63–72, 2003. 62
- [72] G. Lawler and H. Narayanan. Mixing times and l_p bounds for Oblivious routing. In *Workshop on Analytic Algorithmics and Combinatorics (ANALCO09)*. 22, 28, 29, 33
- [73] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999. 54

- [74] Linial, Magen, and Saks. Trees and euclidean metrics. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1998. 71
- [75] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. 87
- [76] W. Mader. Homomorphieeigenschaften und mittlere kantendichte von graphen. *Mathematische Annalen*, 174:265–268, 1967. 10.1007/BF01364272. 100
- [77] Hiroshi Maehara. Space graphs and sphericity. *Discrete Appl. Math.*, 7:55–64, 1984. 67
- [78] B. Maggs, F. Meyer auf der Heide, B. Voeking, and M. Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 284–, Washington, DC, USA, 1997. IEEE Computer Society. 22
- [79] J. Matousek. On embedding trees into uniformly convex banach spaces, 1999. 71
- [80] Jiri Matousek. Geometric computation and the art of sampling. In *IEEE Symposium on Foundations of Computer Science*, page 2, 1998. 74
- [81] Damon Mosk-Aoyama and Devavrat Shah. Computing separable functions via gossip. In *Proceedings of the twenty-fifth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 113–122, New York, NY, USA, 2006. ACM. 87, 89, 90
- [82] Papadimitriou and Ratajczak. On a conjecture related to geometric routing. *TCS: Theoretical Computer Science*, 345, 2005. 60, 62
- [83] Christos H. Papadimitriou and David Ratajczak. On a conjecture related to geometric routing. *Theor. Comput. Sci.*, 344:3–14, November 2005. 16
- [84] D. Peleg and A.A. Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989. 104, 106
- [85] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. 83
- [86] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. 87, 91
- [87] S. Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Computing*, 22(3):147–166, 2010. 90, 105
- [88] Seth Pettie. Low distortion spanners. *ACM Transactions on Algorithms (TALG)*, 6:7:1–7:22, December 2009. 88, 90, 105, 106
- [89] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 255–264. ACM New York, NY, USA, 2008. 16, 22, 25, 26, 28, 29, 33, 59

- [90] Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science, FOCS '02*, pages 43–52, Washington, DC, USA, 2002. IEEE Computer Society. 16, 25
- [91] Ananth Rao, Christos Papadimitriou, Scott Shenker, and Ion Stoica. Geographic routing without location information. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108, New York, NY, USA, 2003. ACM Press. 62
- [92] Robert Freund. Interior-point theory for convex optimization. 2007. 32
- [93] Tim Roughgarden. The price of anarchy is independent of the network topology. *J. Comput. Syst. Sci.*, 67:341–364, September 2003. 22
- [94] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *J. ACM*, 49:236–259, March 2002. 22
- [95] Thomas Sauerwald and Alexandre Stauffer. Rumor spreading and vertex expansion on regular graphs. In Dana Randall, editor, *SODA*, pages 462–475. SIAM, 2011. 87
- [96] Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In *Proceeding of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 257–266, New York, NY, USA, 2010. ACM. 91
- [97] Alistair Sinclair. *Algorithms for random generation and counting: a Markov chain approach*. Birkhauser Verlag, Basel, Switzerland, Switzerland, 1993. 93
- [98] Sleator, Tarjan, and Thurston. Rotation distance, triangulations, and hyperbolic geometry. *JAMS: Journal of the American Mathematical Society*, 1, 1988. 68
- [99] D.A. Spielman. Graphs and networks, Lecture Notes. 35
- [100] D.A. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 563–568. ACM New York, NY, USA, 2008. 43
- [101] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *CoRR*, abs/0808.4134, 2008. 95
- [102] M. Thorup and U. Zwick. Compact routing schemes. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 1–10. ACM New York, NY, USA, 2001. 33
- [103] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005. 33

- [104] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, SPAA '01, pages 1–10, New York, NY, USA, 2001. ACM. 17, 59
- [105] W. P. Thurston. *Three-Dimensional Geometry and Topology*, volume 35 of *Princeton Mathematical Series*. Princeton University Press, 1997. 68, 69
- [106] L G Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, 1982. 16
- [107] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, STOC '81, pages 263–277, New York, NY, USA, 1981. ACM. 16