Stony Brook University

Exploring Advanced Communication Primitives Using Greedy Routing in Sensor Networks and Other Complex Networks

A Dissertation Presented
by
Xiaomeng Ban

to
The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of
Doctor of Philosophy
in
Computer Science
Stony Brook University

December 2012

Stony Brook University
The Graduate School

Xiaomeng Ban

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Jie Gao – Dissertation Advisor
Associate Professor, Department of Computer Science

Joseph Mitchell – Chairperson of Defense
Professor, Department of Applied Mathematics and Statistics

Xianfeng Gu
Associate Professor, Department of Computer Science

Wei Zeng – External Member
Assistant Professor, School of Computing and Information Sciences
Florida International University

This dissertation is accepted by the Graduate School

Charles Taber
Interim Dean of the Graduate School

Abstract of the Dissertation

Exploring Advanced Communication Primitives Using Greedy Routing in Sensor Networks and Other Complex Networks

by

Xiaomeng Ban

Doctor of Philosophy

in

Computer Science

Stony Brook University

2012

Scalable point-to-point routing on a wireless sensor network has been an active research topic for the past ten years. The major challenge comes from the fundamental resource limitation of sensor nodes, in terms of storage size and communication bandwidth. The solution that requires a node to acquire the entire network topology does not scale well. In the past few years there have been a number of innovative proposals on scalable routing schemes where each node only keeps local information and a routing path can be discovered by iteratively applying greedy routing decisions. Such work has mainly focused on issues such as guaranteed delivery and low path stretch, and has been relatively successful in that regard. The goal of this dissertation is to move on with greedy routing techniques and explore more advanced communication primitives.

The first challenge comes from load balancing in sensor networks. In large scale sensor networks it is critical to balance out work load on different sensors, to prolong network lifetime and prevent immature node failures or network disconnection. We propose two different techniques to balance out traffic load in the case of uniform network traffic pattern. Given a sensor network densely deployed on a simply connected domain Ω, we apply area-preserving map to transform Ω to a disk $\mathcal{D}$, then use load balanced routing on the virtual coordinates of the sensor nodes on the disk $\mathcal{D}$. Another technique we propose applies on a 3-connected sensor network deployed on a domain possibly with holes inside, we use discrete Ricci flow

to compute the circle packing of the spherical embedding of the 3-connected planar subgraph, and apply a heuristic algorithm by Möbius transformation to optimize load balancing across the sensor nodes.

The second issue is exploring the path space in sensor networks. In a sensor network there could exist multiple disjoint paths between a source and a destination, an efficient method to explore and navigate in the 'path space' can help many important routing objectives, e.g., high network throughput, low latency and fast recovery on network failures. We present distributed algorithms based on Möbius transformation on circular domains. The algorithms use local information and limited global information to generate disjoint multi-paths for a given source destination pair or switch to a different path 'on the fly' when transmission failure is encountered. This method compares favorably in terms of performance and cost metrics with centralized solutions of using flow algorithms or random walk based decentralized solutions in generating alternative paths.

Thirdly, greedy routing could suffer from a wormhole attack, in which the adversary places two radio transceivers connected by a high capacity link and retransmits wireless signals from one antenna to the other. This creates a set of shortcut paths in the network, and may attract a lot of traffic to the wormhole link. We introduce a wormhole detection and removal algorithm based on local connectivity tests. The algorithm uses purely local connectivity information, handles multiple wormhole attacks and generalizes to wireless networks deployed in 3D. It does not suffer from typical limitations in previous work such as the requirements of special hardware, communication models, synchronization, node density etc, and guarantees to detect and remove the wormholes.

Last but not the least, greedy routing can be extended to routing on a general graph due to its simplicity and efficiency, especially for navigation in real-world complex networks. We systematically investigate the conjecture made in earlier small world navigation studies that many real-world complex networks are navigable. That is, it is possible to discover a hidden metric space purely from the network connectivity information alone that permits greedy routing on the coordinates in the hidden space to discover extremely short paths for a majority of node pairs. We confirm the conjecture, delivering packages in a majority of cases in each of our five empirical networks, from a diverse set of application scenarios.

Dedicated to my family.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advances in recent technologies have enabled the manufacturing of tiny and inexpensive sensor nodes. Sensor nodes can be densely deployed to monitor real world environments and achieve unprecedented spatial coverage and robustness. By communicating with nearby nodes using wireless transceivers, the sensor nodes are able to form large scale networks to serve in many applications, e.g., industrial monitoring, natural disaster relief and smart transportation. While sensor networks are powerful by monitoring the signal in the physical world, they are subject to a unique set of resource constraints such as limited amount of memory, battery life and network communication bandwidth, this makes efficient routing a big challenge. Routing in a wireless ad hoc network can be largely divided into proactive routing and reactive routing. Reactive routing techniques, e.g., DSR [75] and AODV [120], use on-demand approaches for finding routes. A route is established only when it is required by a source node for transmitting data packets to the destination, usually by issuing a flooding. While reactive routing techniques do not require configuring and maintaining routing tables on sensor nodes, they introduce heavy routing overhead and do not scale well when networks become large. Traditional proactive routing techniques, e.g., link state based (OLSR) or distance vector based (DSDV) routing algorithms, require to pre-configure and maintain routing tables on sensor nodes. While proactive routing techniques have been successful in Internet routing and are heavily used in TCP/IP stack, compared to Internet routers, sensor nodes have quite limited resources and are prone to node and link failures, those traditional proactive routing techniques do not scale well in sensor networks.

In the past few years there have been a number of innovative proposals on geographical based routing schemes where each sensor node only keeps and makes use of geographical information of its neighbors, among those routing schemes greedy routing is a widely used routing strategy. In greedy routing each intermediate node on the routing path would greedily forward the packets by choosing the neighbor closest to the destination as the next hop, and the routing path can be discovered by iteratively applying greedy routing decisions. Since sensor networks are inherently geometric, the number of neighbors of each node usually remains a constant when the size of the network grows, which makes the geographical information storage on each node scales well with the network size. While greedy routing works well in general, it may get stuck in the middle due to a local trap and fail to delivery the data to destination. There has been previous works focused on guaranteed delivery for greedy routing, those works have been relatively successful in that regard. While guaranteeing delivery is important for routing, there are some other communication primitives that are important as well, e.g., load balancing, multi-path routing, securities issues, etc. In our work, we move on with greedy routing techniques and focus on those advanced communication primitives.

In large scale sensor networks it is critical to balance out work load on different sensors, to prolong network lifetime and prevent immature node failures or network disconnection. We examine the problem of scalable routing algorithm design that balances out traffic load. Here, We focus on the load balancing issue in the case of uniform network traffic pattern, i.e., all pairs of nodes have the same amount of data to deliver. In this case, uneven load across the nodes can happen as a result of network geometry and routing schemes. It is easy to see that for sensors uniformly spread on a disk or a square, the nodes near the center carry more traffic than the nodes near the boundary if we use shortest path algorithm or geographical greedy routing, which is called crowded center effect. The network topology and its 'shape' are critically relevant to load balancing of a specific routing scheme, how to design routing strategies to achieve load balancing for sensor networks is a fundamental problem that is both theoretically and practically interesting. The solution for uniform traffic pattern will also hope to provide insights for the general case when traffic pattern is not uniform.

Besides load balancing, routing primitives such as throughput, low latency and

recovery from failures are also important in practice. The exploration of 'path space' turns out to be helpful in those primitives. Between a source and a destination there can be multiple different paths, a single path from source to destination may give limited throughput due to bandwidth constraints, hop length, wireless interference or other transmission failures. If there is a lot of data to be delivered, it is natural to consider using multiple node disjoint paths such that different data segments can be simultaneously delivered to the destination. With multipath routing one obtains higher throughput and lower delay. Such multipath routing can also be used to enhance data security. For example, sensor data can be encoded such that different codewords are sent along different paths. Therefore a single compromised node stays on at most one path and with its captured data segments it is unlikely to reconstruct the original data. Exploring the space of routing paths between two nodes is also helpful for fast recovery from link or node failures. In a large wireless network, there are network changes of different scales. At the node level, wireless links have high link quality variation; nodes may fail; interference with other nodes unpredictable, e.g., as in the hidden terminal problem. At a large scale, communication links in a region can be temporarily disabled by jamming attacks, either imposed by malicious parties [159], or as a result of co-located multiple benign wireless networks interfering with each other. In case of a transmission failure, it would be good to quickly discover an alternative path to the destination. A single isolated link failure can possibly be bypassed by local random de-touring attempts. A large scale node or link failure, in particular one with strong spatial correlations and a geometric pattern, would need some non-trivial exploration of the path space – by making possibly a big de-tour from the planned path, see Figure 1 for an example. To allow such robustness and quick response to network conditions, routing schemes that find a single path are not enough and it is important to understand the 'space' of paths and efficiently navigate within this space. multipath routing is also helpful for load balancing under non-uniform traffic pattern. If there is a lot of data to be exchanged between the source s and destination t, the nodes on the path P are used more often than average. To improve load balancing, traffic needs to be distributed on multiple paths from s to t that are sufficiently far apart from one another to share the traffic load.

Since greedy routing is a geographic based routing paradigm, it may suffer

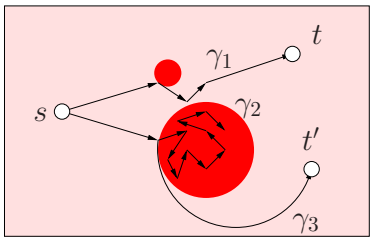


Figure 1: Consider some part of the network experiencing heavy inference (or jamming attacks), shown as the dark colored circles. Links inside these 'failure' regions have much higher loss rate. A route that hits a small failure region might be able to get around by performing some random walks in the neighborhood, as in the case of path γ_1. A route that hits a large failure region has difficulty recovering from it – as simple random walk is likely to wander around for a long time, as shown by the path γ_2. In this case a path that makes big de-tours would perform much better, as shown by the path γ_3.

from attacks making use of location information. In the wormhole attack, the adversary places two radio transceivers in the network, connected by a high capacity out-of-band wireless or wired link, signals captured by one transceiver are "tunneled" through the wormhole link to the other transceiver, and replayed there. If the two transceivers are further away from each other, the wormhole link gains control of a large fraction of network traffic which opens the door for more dangerous attacks afterwards.

Collecting data from sensor networks to a static data sink often suffers from communication bottleneck near the sink, this is called 'funnel effect'. One way to address this is to use a mobile sink, or called a data mule, implemented by a mobile device touring around the network to collect data through direct communication with a sensor in close proximity. A data mule moves along a path, planning the motion of a data mule requires a path that visits the nodes in the network with minimum duplicate visits.

While greedy routing is widely used in sensor networks routing, as a routing paradigm it is not limited to sensor networks. The 'small-world phenomenon' (aka 'six degree of separation') states that there exists a short path between almost any pair of individuals in the world. It was later discovered that many other networks,

in vastly different contexts ranging from power grids, film collaboration networks, and neural networks [48] to email networks [42], food webs [158] and protein interaction networks [72], also exhibit the small-world property. In addition to revealing the existence of short paths in real-world acquaintance networks, the small-world experiments showed that these networks are navigable by greedy routing: in Milgram's small world experiment [151], a short path was discovered through a local algorithm with the participants forwarding to a friend who they believed to be more likely to know the target. According to the homophily theory, each individual can be considered having a vector of attributes, such as gender, age, nationality and geographical location, as its own coordinate, and greedy forwarding can be done by making use of measurements on such coordinate systems. Geographical proximity has been found to be an important forwarding criterion in some cases, other criteria such as profession and popularity may have been used as well. A recent small-world study using email-chains [42] confirms this, finding that at least half of the choices were due to either geographical proximity of the acquaintance to the target or occupational similarity. Thus these experiments hint that perhaps also in other networks, some greedy routing algorithm can successfully deliver messages, provided that nodes are given appropriate coordinates. In our work we consider the conjecture that real-world networks from diverse contexts, social and non-social, can be embedded in a low-dimensional hidden space where the distances between nodes in the hidden space approximate their graph distances in the network, such that some greedy mechanism minimizing the distances to the destination in the latent space is able to find a short path for most pairs of nodes.

1.1 Overview

Chapter 2 is background knowledge on greedy routing, conformal mapping to circular domain and polyhedron routing.

In chapter 3 we address the problem of load balanced greedy routing for wireless sensor networks using area-preserving map. An *area-preserving map* ϕ is able to transform any simply connected domain Ω to a disk $\mathcal{D}$. If the sensor network is uniformly densely deployed within an arbitrary shape region Ω, after applying the area-preserving map on Ω, the original network sources and destinations are still

uniformly distributed in the disk $\mathcal{D}$. There have been techniques achieving good load balancing on sensor networks densely deployed on a disk, e.g., curveball routing. Based on the virtual coordinates of the sensor nodes in $\mathcal{D}$, we can apply those techniques to achieve good load balancing on sensor networks densely deployed on an arbitrary simply connected domain.

In chapter 4 we address the problem of load balanced greedy routing for wireless sensor networks using Ricci flow and Möbius transformation. Motivated by the analog of the continuous setting that geodesic routing on a sphere gives perfect load balancing, we embed sensor nodes on a convex polyhedron in 3D and use greedy routing to deliver messages between any pair of nodes with guaranteed success. This embedding is known to *exist* by the Koebe-Andreev-Thurston Theorem for any 3-connected planar graphs. Here we use Ricci flow to develop a distributed algorithm to compute this embedding. Further, such an embedding is not unique and differ by one another with a Möbius transformation. We employ an optimization routine to look for the Möbius transformation such that the nodes are spread on the polyhedron as uniformly as possible. We evaluated the load balancing property of this greedy routing scheme and showed favorable comparison with previous schemes.

In chapter 5 we investigate the application of 'path space' in greedy routing. In a sensor network there are many paths between a source and a destination. An efficient method to explore and navigate in the 'path space' can help many important routing primitives, in particular, *multipath routing* and *resilient routing* (when nodes or links can fail unexpectedly) as considered. Both problems are challenging for a general graph setting, especially if each node cannot afford to have the global knowledge. In this paper we use a geometric approach to allow efficient exploration of the path space with very little overhead. We are motivated by the recent development on regulating a sensor network geometry using conformal mapping [137,138], in which any sensor network can be embedded to be circular (and any possible hole is made circular as well) and greedy routing guarantees delivery. In this paper we explore the freedom of a Möbius transformation inherent to this conformal mapping. By applying a Möbius transformation we can get an alternative embedding with the same property such that greedy routing generates a different path. We

present distributed algorithms using local information and limited global information (the positions and sizes of the holes) to generate disjoint multi-paths for a given source destination pair or switch to a different path 'on the fly' when transmission failure is encountered. The overhead of applying a Möbius transformation simply boils down to four parameters that could be carried by a packet or determined at need at the source. Demonstrated by simulation results, this method compares favorably in terms of performance and cost metrics with centralized solutions of using flow algorithms or random walk based decentralized solutions in generating alternative paths.

In chapter 6 we propose a distributed wormhole detection and removal algorithm, which uses purely local connectivity information. A wormhole attack places two radio transceivers connected by a high capacity link and retransmits wireless signals from one antenna at the other. This creates a set of shortcut paths in the network, and may attract a lot of traffic to the wormhole link. The link thus gains control of a large fraction of network traffic which opens the door for more dangerous attacks afterwards. The basic idea of our algorithm is that the neighborhood of a wormhole contains two sets of nodes corresponding to two sides of the wormhole. The distance between these two sets is small when using paths that pass through the wormhole link, but is large when only regular network paths are considered, Thus the removal of a small neighborhood containing potential wormhole edges would lead a larger neighborhood to fall apart to multiple connected components. The algorithm uses purely local connectivity information, handles multiple wormhole attacks and generalizes to wireless networks deployed in 3D. It does not suffer from typical limitations in previous work such as the requirements of special hardware, communication models, synchronization, node density etc. Our algorithm guarantees the detection and removal of wormhole edges, handles multiple wormhole attacks and generalizes to wireless networks deployed in 3D. It does not suffer from typical limitations in previous work such as the requirements of special hardware, communication models, synchronization, node density etc. In simulations, our method is seen to beat the state of the art solutions, in particular for cases where previous solutions experience poor performance. In simulations, our method is seen to beat the state of the art solutions, in particular for cases where previous solutions experience poor performance.

In chapter 7 we propose an algorithm to construct a "space filling" curve for a sensor network with holes. Mathematically, for a given domain $\mathcal{R}$ possibly with holes inside, we generate a path $\mathcal{P}$ that is provably aperiodic (i.e., any point is covered at most a constant number of times) and dense (i.e., any point of $\mathcal{R}$ is arbitrarily close to $\mathcal{P}$). Specifically, for a simple domain with no holes, we will first map it one-to-one to a unit square, and then flip the square along the top edge and the right edge to get four copies, creating a torus, then we find the dense curve on the torus. Since any point in the original domain is mapped to four copies on the torus, the curve we find will visit any point for at most four times, the property of being dense still holds. For a domain with holes, we will first double cover it, i.e., creating two copies of the network, the upstairs copy and the downstairs copy. The two copies are glued to each other along the hole boundaries to create a multi-torus, each hole being a handle. In the same way we choose one handle to flatten the torus, and the rest of the handles are mapped to very narrow 'slits'. A line with irrational slope in the covering space, when hitting a slit, bounces back. We could show that the curve will visit each point of the original domain at most twice and is provably dense. In a discrete setting as in a sensor network, the path visits the nodes with progressive density, which can be adaptive to the budget of the path length. Given a longer budget, the path covers the network with higher density. With a lower budget the path becomes proportional sparser. We show how this density-adpative space filling curve can be useful for applications such as serial data fusion, motion planning for data mules, sensor node indexing, double ruling type in-network data storage and retrieval. We show by simulation results the superior performance of using our algorithm vs standard space filling curves and random walks.

In chapter 8 we propose to first embed social networks in latent spaces and apply greedy routing with the coordinates generated, Our experiment is the first to systematically investigate the conjecture made in earlier small world navigation studies that many real-world complex networks are navigable. That is, it is possible to discover a hidden metric space purely from the network connectivity information alone that permits greedy routing on the coordinates in the hidden space to discover extremely short paths for a majority of node pairs.

1.2 References

Chapter 3 is a recent work with Mayank Goswami, Chien-Chun Ni, Xianfeng David Gu and Jie Gao.

Chapter 4 is the work coauthored with Xiaokang Yu, Wei Zeng, Rik Sarkar, Xianfeng David Gu and Jie Gao. A paper with title "Spherical representation and polyhedron routing for load balancing in wireless sensor networks" has been published in the Proceedings of the 30th Annual IEEE Conference on Computer Communications (INFOCOM'11), 612-615, mini-conference, March, 2011. The work is also presented at 20th Fall Workshop on Computational Geometry, Oct 29-30, 2010.

Chapter 5 is the work coauthored with Ruirui Jiang, Xiaomeng Ban, Mayank Goswami, Wei Zeng, Jie Gao, Xianfeng David Gu. A paper with title "Exploration of Path Space using Sensor Network Geometry" has been published in the Proceedings of the 10th International Symposium on Information Processing in Sensor Networks (IPSN'11), 49-60, April, 2011.

Chapter 6 is the work coauthored with Rik Sarkar and Jie Gao. A paper with title "Local Connectivity Tests to Identify Wormholes in Wireless Networks" has been published in the Proceedings of the 12th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'11), May, 2011.

Chapter 7 is the work coauthored with Mayank Goswami, Wei Zeng, Xianfeng David Gu and Jie Gao. A paper with title "Topology Dependent Space Filling Curves for Sensor Networks and Applications" has been accepted by the 32th Annual IEEE Conference on Computer Communications (INFOCOM'13) main conference.

Chapter 8 is the work coauthored with Jie Gao and Arnout van de Rijt. A paper with title "Navigation in Real-World Complex Networks through Embedding in Latent Spaces" has been published in Workshop on Algorithm Engineering and Experiments (ALENEX10), 138-148, January, 2010.

Chapter 2

Background Review

2.1 Geographic Routing

2.1.1 Greedy Routing

Sensor networks are deployed in real world and inherently associated with geometric information. If nodes know their own location information, they can adopt the greedy forwarding strategy for routing. The nodes would maintain the location information of their neighbors. On each routing path, the current node would choose the neighbor node closest to destination based on the location table, and forward the packet to this neighbor. While greedy routing is light weight and scalable by making use of purely local information, it might get stuck in the middle when no neighbor node is closer to the destination than the current node itself. There have been various approaches to deal with this issue [13, 56, 78, 87, 95].

Greedy routing has received a lot of attention since it was proposed for routing in ad hoc wireless networks [19, 78], due to its simplicity and efficiency. As greedy routing may get stuck if all neighbors are further away from the destination, people ask whether one can find a proper embedding of the network such that delivery is guaranteed.

For 2D embedding, Papadimitriou and Ratajczak [118] made the conjecture that any planar 3-connected graph has a greedy embedding in the plane. Dhandapani discovered that any planar triangulation (without holes) admits a greedy embedding in the plane [40]. Recently the 3-connected graph conjecture was proved to

be true by Leighton and Moitra [95], and independently by Angelini *et al.* [5]. Later the algorithm in [95] was improved such that the coordinates use $O(\log n)$ bits for a graph with n vertices [62]. In [137], a given network is embedded in the plane such that all the holes are circular and thus greedy routing guaranteed delivery.

In higher dimensions, the polyhedron routing (in 3D) is first proposed in [118]. Flury *et al.* [56] examined embedding into $O(\log n)$ dimensional Euclidean space and proposed greedy method that gives low stretch routing paths.

Embedding into non-Euclidean spaces has also been proposed. R. Kleinberg [87] shows an embedding of a tree in the hyperbolic space for which greedy routing always works. This scheme was later improved such that the coordinates use logarithmic bits [52].

Of particular relevance to our technique, Ben-Chen *et al.* [13] computed a circle packing of a given 3-connected planar graph and used the power distance for greedy routing. They showed that their embedding as a contained power diagram is another interpretation (and generalization) of the spherical embedding.

The above greedy routing schemes focused on guaranteed delivery and have no consideration on load balancing.

2.1.2 Location Service

To apply greedy routing on sensor networks, the source and intermediate nodes need to know the location information of the target node. In most cases, the source node only knows unique identification UID of the target node instead of the location information. Therefore, a mapping from UID to location information is needed, this is called the location service.

One strategy for location service is to store all the locations in a central repository. Although centralized storage is simple and keeps data integrity, it suffers single point of failure. Besides, this method is not scalable since when the network size is large, communication cost and delay can be significantly large, and funnel effect would happen around the repository. Another strategy is to store all the locations on each sensor node, this would lead to enormous storage and the information updating would be extremely expensive. Flooding-based strategies adopt either the proactive (e.g. SLS, DLS) or the reactive routing protocols (e.g. LAR,

RLS), in proactive routing nodes periodically flood their locations to a range of nodes, while in reactive routing the source node perform a flooding to get the location of the target node. In general flooding is not scalable for large sensor networks. Rendezvous-based strategies set up a mapping from each node to a group of nodes, the mapped-to nodes are the location servers for the node, which would get periodic updated location information from the node, The location information of the target node would be retrieved from the location servers of the target node. The mapping can be either flat or hierarchical, depending on whether recursive hierarchical subareas are used.

2.2 Load Balanced Routing

In the field of networking algorithms, load balanced routing on a graph with given source destination pairs is a long standing problem that has been studied a lot in the literature. One way to formulate this problem is to select routes that minimize congestion (the maximum number of messages that any node/link carries), termed the *unsplittable flow problem*. Finding the optimal solution for this problem is NP-hard even in very simple networks (such as grid). The best approximation algorithm has an approximation factor of $O(\log n/\log\log n)$ [126, 127] in a network of n vertices. It is also shown that getting an approximation within factor $\Omega(\log\log n)$ is NP-hard [4]. Another popular way to formulate the problem is to consider node disjoint or edge disjoint paths that deliver the largest number of given source destination pairs. This is again extensively studied. It is one of the earliest NP-hard problems [79]. The best approximation algorithm has a factor of $O(\sqrt{n})$ approximation [30] and it is NP-hard to approximate within a factor of $\Omega(\log^{1/2-\varepsilon} n)$ [33]. Beside, the approximation algorithms are mostly only of theoretical interest. They require global knowledge and are not suitable for practical settings.

Here we consider the load balancing for sensor networks densely deployed inside a geometric domain Ω, insights on the geometric properties of such networks are thus extremely helpful for interesting solutions. This is an approach that has been adopted in recent years, in particular using geometric maps to generate virtual coordinates that are suitable for greedy routing algorithms [137]. Our objective is along the same direction with load balancing as the main consideration: given

a sensor network deployed inside a geometric domain Ω, can we generate virtual coordinates and a companion greedy routing algorithm that achieves *bounded* maximum congestion?

Existing work mainly focused on simple shapes such as strips, disks or squares; or focused on reducing congestions in certain parts of the network such as interior hole boundaries. Clearly the maximum congestion depends on the traffic pattern, i.e., the distribution of sources and destinations. If all messages are from the same source and/or same destination, there is no way we can reduce the congestion at the sources/destinations and the problem becomes less interesting. Therefore it is a common practice in previous work to assume that sources and destinations are uniformly distributed. This limits the problem domain down to the core problem — under uniform traffic pattern, how the geometric shape affects the congestion and what kind of greedy routing algorithm gives the minimum congestion.

Load balancing routing on simple shapes. In [60], a greedy routing scheme that achieves both constant routing stretch factor (compared with shortest path routing) and constant load balancing ratio (compared with the optimal load balanced routing) is proposed, but only for wireless nodes distributed in a narrow strip.

In a disk, Popa *et al.* [124] pointed out that under uniform traffic patten, the center is more loaded than nodes near the boundary – as more shortest paths go through or near the center. By using numerical solvers, they develop a numerical approximation of the optimal load balanced routing solution. They also proposed a practical algorithm by using stereographic projection to map the network on a hemisphere. Routing is guided by the spherical distance in a greedy manner. Improved load balancing is shown as the routes are made to 'curve' around the network center. The same idea is also used by Li *et al.* [97] to resolve routing congestion at network center.

Mei and Stefa [109] studied load balancing for a regular square shape sensor network and propose to use the 'outer space' by essentially creating four copies of the network, wrapped up as a topological torus. The destination has four images and greedy routing is done by using the coordinates on the torus towards a randomly selected image of the destination. On the original network, it is as if we are reflecting on the boundaries. The idea is that through mapping on a torus the original boundary or center of the square are essentially removed so they should

not present higher congestion. By evaluations we found out that the traffic load on the nodes are indeed evened out, but unfortunately the traffic level at all nodes have been greatly elevated as routing paths through reflections are on average much longer than shortest paths.

Yu *et al.* [161] examined a network inside a simply connected domain and used Ricci flow to generate the Thurston's embedding as the skeleton of a convex polytope. The intuition is to map the network on a sphere (or half sphere) as routing on a sphere has no congestion due to perfect symmetry. The choice of a convex polytope is to ensure that greedy routing has guaranteed delivery [118]. This has good performance as shown in simulations but there is no guarantee on the worst case congestion.

Reducing congestion on centers or hole boundaries. A number of previous work focused on reducing traffic load on hole boundaries. This is because traditional geographical routing [55,78] tends to send messages to nodes near hole boundaries. In [138], a network of multiple holes is converted to the covering space, such that one maps the network to the interior of each hole, filling it up. Again, since the boundaries are removed, greedy routing, when touching a boundary node, does not follow the boundary but get reflected away from the boundary. Simulation results show that the traffic load on boundaries are greatly reduced. But the average traffic load is increased as routing paths are made longer.

In [22], the focus is to reduce traffic load near the medial axis, which is a generalization of the center of a disk in a domain of general shape and is likely to attract traffic load. The proposed routing algorithm will follow a path parallel to the medial axis or orthogonal to the medial axis, minimizing the intersections with the medial axis. Again no theoretical guarantee is given.

2.3 Conformal Mapping to Circular Domain

2.3.1 Conformal Mapping

In the continuous setting for Riemannian surfaces, let (S_1, g_1) and (S_2, g_2) be two surfaces with Riemannian metrics g_1, g_2. A mapping $\phi : S_1 \to S_2$ is called a *conformal mapping* (*angle preserving mapping*), if the intersection angle of any

two curves is preserved.

A planar domain D of connectivity m is called a *circular domain*, if all its m boundaries are circles. It is known from conformal geometry that any genus zero multiply connected planar domain can be mapped to a circular domain by conformal mappings. Such a mapping is not unique: all such mappings differ by Möbius transformations [39, 122].

To compute the conformal mapping from a surface to a circular domain, one can use Ricci flow as introduced in [74, 137]. In the case of sensor network setting, we will use the discrete version, which represents a domain by a discrete triangulation. We first give some references on how to obtain such a triangulation from a sensor network setting and then move on to the algorithm description of the discrete Ricci flow.

2.3.2 Discrete Ricci Flow

In the following we explain Ricci flow in the discrete setting for a triangulation of a domain with m holes. The triangulation is denoted by Σ with vertex set V, edge set E and face set F.

In the discrete setting we define a Riemannian metric by using the edge lengths on Σ:

$$l : E \to \mathbb{R}^+,$$

such that for a triangle face f_{ijk} with vertices v_i, v_j, v_k, the edge lengths satisfy the triangle inequality:

$$l_{ij} + l_{jk} > l_{ki}, \quad \forall i, j, k.$$

The lengths of the edges of the triangulation determine the corner angles of the triangles. For a triangle f_{ijk} with edge lengths $\{l_{ij}, l_{jk}, l_{ki}\}$, and the angles opposite to these edges $\{\theta_k^{ij}, \theta_i^{jk}, \theta_j^{ki}\}$ respectively, we have the following equations by cosine law:

$$l_{ij}^2 = l_{jk}^2 + l_{ki}^2 - 2 l_{jk} l_{ki} \cos \theta_k^{ij}. \tag{1}$$

Now we can define the discrete Gaussian curvature at a vertex v_i as the angle deficit:

$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \text{ is an interior vertex;} \\ \pi - \sum_{f_{ijk} \in F} \theta_i^{jk}, & v_i \text{ is at boundary.} \end{cases} \tag{2}$$

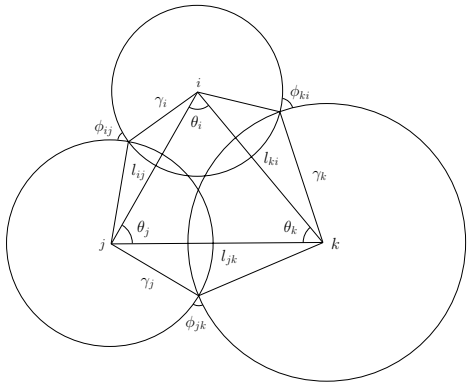


Figure 2: The circle packing metric.

where θ_i^{jk} represents the corner angle at vertex v_i in the triangle f_{ijk}. In other words, the curvature at a vertex v is the difference of 2π or π and the total corner angles at v, for an interior vertex or a vertex on a hole boundary respectively. The curvature is 0 when it is locally flat (for interior vertices) or locally a straight line (for boundary vertices).

Ricci flow uses the circle packing metric in the discrete case, proposed by [146, 150], to approximate the conformal deformation of metrics. See Figure 2. Each vertex v_i has a circle with radius γ_i. On each edge e_{ij}, ϕ_{ij} is defined as the intersection angle of the two circles at v_i and v_j. The pair of vertex radii and intersection angles at the edges on a mesh Σ, (Γ, Φ), is called a *circle packing metric* of Σ. Two circle packing metrics (Γ_1, Φ_1) and (Γ_2, Φ_2) on Σ are *conformal equivalent*, if $\Phi_1 \equiv \Phi_2$. Therefore, a conformal deformation of a circle packing metric only modifies the vertex radii γ_i's and maintains the intersection angles ϕ_{ij}'s to be constant.

For a given mesh, the circle packing metric (Γ, Φ) and the edge lengths can be converted to each other by cosine law as below:

$$l_{ij}^2 = \gamma_i^2 + \gamma_j^2 + 2\gamma_i\gamma_j \cos\phi_{ij}. \tag{3}$$

Thus given a circle packing metric, we can calculate the edge lengths of the triangulation Σ and then the embedding in the plane realizing the given curvatures.

Let u_i to be $\log\gamma_i$ for each vertex. The discrete Ricci flow is defined as the

following differential equation:

$$\frac{du_i(t)}{dt} = (\bar{K}_i - K_i), \tag{4}$$

where K_i is the current curvature at vertex i and $\bar{K}_i$ is the target curvature at i. Define an energy function

$$f(\mathbf{u}) = \int_{\mathbf{u_0}}^{\mathbf{u}} \sum_{i=1}^{n} (\bar{K}_i - K_i) du_i, \tag{5}$$

as the *Ricci energy*, where $\mathbf{u_0}$ is an arbitrary initial metric. It has been proved by Chow and Luo [31] that the discrete Ricci flow will converge to a unique minimum of the Ricci energy. The convergence rate of the discrete Ricci flow using Equation 4 is shown to be exponentially fast, i.e.,

$$|\bar{K}_i - K_i(t)| < c_1 e^{-c_2 t}, \tag{6}$$

where c_1, c_2 are two positive constants.

The Ricci flow algorithm is naturally an iterative algorithm with all vertices adjusting local metrics and local curvatures. All the radii at the vertices are initialized to be $1/2$. That is, the circles at adjacent vertices of Σ are kept to be tangent to each other. We set the target curvature to be 0 at all interior vertices. That is, the network should be embedded to be flat in the domain. We set the target curvature at a boundary vertex to be $-2\pi/k$, if the boundary of the hole has a total number of k vertices. That is, the boundary circle should be perfectly circular. We apply the Ricci flow algorithm by changing the circle packing metric, u_i, by $\delta(\bar{K}_i - K_i)$, where δ is a constant parameter as the step size. The algorithm stops when the current curvature is within an error bound of ε from the target curvature.

Since the curvature error decreases exponentially fast, the number of steps in the Ricci flow algorithm is in $O(\frac{\log(1/\varepsilon)}{\delta})$, where δ is the step size in the Ricci flow algorithm. The total number of messages is thus in $O(\frac{n\log(1/\varepsilon)}{\delta})$, if the algorithm is running on a network of n vertices.

2.3.3 Möbius Transformations

Möbius transformations are rational functions defined on the complex plane $\mathbb{C}$. The general form of a Möbius transformation is

$$f(z) = \frac{az + b}{cz + d}.$$

Here $a, b, c, d \in \mathbb{C}$ and satisfy $ad - bc \neq 0$. If $c \neq 0$ we can extend this mapping to the Riemann Sphere (or the extended complex plane, i.e., with a point of infinity) $\widehat{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ by specifying $f(-d/c) = \infty$ and $f(\infty) = a/c$. In the case when $c = 0$, we specify $f(\infty) = \infty$.

Here are the important properties of Möbius transformations:

1. Möbius transformations are all the bijective holomorphic (differentiable in the complex sense) mappings from $\widehat{\mathbb{C}}$ to itself. This also implies that they are conformal, or angle-preserving.

2. Möbius transformations carry circles and lines (which can be regarded as circles passing through ∞, point of infinity) to circles and lines. Thus, giving a circular domain, any Möbius transformation will map it to another circular domain.

3. To every Möbius transformation one can associate a matrix

$$M_f = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

   Any other matrix which is a (nonzero) scalar multiple of this matrix represents the same Möbius transformation. Composition of two Möbius transformations is equivalent to matrix multiplication, i.e., $M_{f \circ g} = M_f \cdot M_g$.

4. Given distinct $z_1, z_2, z_3 \in \widehat{\mathbb{C}}$ and distinct $w_1, w_2, w_3 \in \widehat{\mathbb{C}}$, there is a ***unique*** Möbius transformation f satisfying $f(z_i) = w_i$, $i = 1, 2, 3$. In other words, as there are unique circles C_1 and C_2, defined by z_1, z_2, z_3 and w_1, w_2, w_3 respectively, the transformation f maps the circle C_1 to C_2 and is unique. Determining f explicitly is equivalent to finding determinants of four 3×3 matrices.

It should be noted that there is a natural way to identify the real plane $\mathbb{R}^2$ with the complex plane $\mathbb{C}$, so for our purposes we can assume that nodes are in the complex plane.

2.4 Polyhedron Routing

2.4.1 Spherical Embedding

A graph G is k-connected if the removal of any $k-1$ vertices will not disconnect the graph. A graph is planar if it can be embedded in the plane. For spherical representation we only require a combinatorial planar graph (that does not have K_5 and $K_{3,3}$ minors).

A *circle packing* is a connected collection of circles on any Riemann surface, whose interiors are disjoint. The intersection graph (the tangency graph or contact graph) of a circle packing is the graph having a vertex for each circle, and an edge for every pair of circles that are tangent.

Suppose G is a 3-connected planar graph, and $\tilde{G}$ is the dual graph[1] of G, the following theorem shows the existence of a pair of circle packings for G and $\tilde{G}$ respectively.

Theorem 1 (The Koebe-Andreev-Thurston Theorem) *There is a pair of circle packing $P, \tilde{P}$, where the intersection graph of P and $\tilde{P}$ are isomorphic to G and $\tilde{G}$ respectively. Furthermore, for any vertex $v \in G$ and an adjacent face f, the vertex circle in P is orthogonal to the face circle in $\tilde{P}$.*

Let e be an edge in G, adjacent to face f_1, f_2. Then the planes of face circles in $\tilde{P}$ corresponding to f_1 and f_2 will intersect at a line tangent to the sphere. All edges through a vertex v will intersect at a common point in $\mathbb{R}^3$. All such kind of line segments form a convex polytope, which is the realization of the graph G induced by the circle packing P.

Corollary 2 (Spherical Representation) *Each 3-connected planar graph can be*

[1]Each face of G is a vertex of $\tilde{G}$. Each vertex of G is a face of $\tilde{G}$. An edge connecting two vertices of $\tilde{G}$ if there is an edge shared by the corresponding faces in G.

realized by a 3-polytope which has all edges tangent to the unit sphere. The realization is unique up to Möbius transformations.

A Möbius transformation is a map that maps a complex plane to itself, $f(z) = \frac{az+b}{cz+d}$, where a, b, c, d are four complex numbers satisfying $ad - bc = 1$. A Möbius transformation is a conformal map and maps circles to circles.

2.4.2 Polyhedron Routing

Given source v_i and destination v_j, polyhedron routing is a greedy routing method with distance function $d(v_i, v_j) = -c(v_i) \cdot c(v_j)$, where $c(v_i)$ is the 3D coordinate of v_i in the spherical representation. v_i delivers the message to the neighbor closer to v_j. To see that this polyhedron routing guarantees delivery, we note that the distance function $d(v_i, v_j)$ is essentially the projection of the vector $\overrightarrow{v}_i$ on the vector $-\overrightarrow{v}_j$. $d(v_i, v_j)$ is clearly a linear function of v_j and achieves the global minimum when $v_i = v_j$. The function can not have a local minimum as for a linear function any local minimum is also the global minimum.

For routing in a sensor network, we first extract a 3-connected planar graph, compute its spherical representation, and obtain a 3D coordinate for each node. Then we apply the greedy routing method on the *original* communication graph. The non-planar edges can also be used to route messages if they happen to be useful by the greedy routing standard.

Chapter 3

Load Balancing using Area-Preserving Map

3.1 Introduction

We study the problem of greedy routing in wireless sensor networks and focus on the issue of load balancing. While load balancing or reducing congestion is a general objective for most networking scenarios, in a battery-powered wireless sensor network, this problem becomes more critical. Any subset of nodes used too much faces the risk of running out of battery prematurely; the functionality of the network may deteriorate dramatically even when many nodes still have ample battery life.

3.1.1 Our Approach

The main idea in this paper is to use an *area-preserving map* ϕ that transforms any simply connected domain Ω to a disk $\mathcal{D}$– the only case in which optimal or near optimal load balanced routing is understood. An area preserving map, intuitively, preserves area – an ε-area ball is mapped to a piece (not necessarily round) with total area ε. One of the area preserving maps is the Lambert azimuthal projection, that projects the sphere to a planar domain in an area preserving way, and is used in printing world maps. In this paper the reason we use area preserving map is to preserve the traffic pattern (distribution of sources and destinations). If on the original

network sources and destinations are uniformly distributed in the domain Ω, then they are still uniformly distributed in the disk $\mathcal{D}$, after an area preserving map. An area preserving map does not necessarily preserves angles or length stretch. But the maximum length stretch d can be computed and is dependent on the specific shape of Ω. Suppose we use the optimal load balanced routing (minimizing maximum congestion) inside the disk $\mathcal{D}$, and pull back the routing method on the original domain Ω, the maximum congestion is bounded by a factor d^2 times the maximum congestion on $\mathcal{D}$, where d is the maximum length stretch of ϕ. In practice, what this means is that we can compute a virtual coordinate for each node x in Ω, as $\phi(x)$, the coordinate inside the disk $\mathcal{D}$. Based on the virtual coordinate we can run good load balanced routing using greedy algorithms. The maximum congestion under uniform traffic pattern is guaranteed to be bounded by only a factor d^2 more. Note that the factor d^2 we compute is via a worst-case analysis.

For load balanced routing on a disk, the most prominent work is done by Popa *et al.* [124]. They studied the numerical solution of the optimal load balanced routing on a disk, though not practical and also proposed a heuristic algorithm, called the *curveball routing*, that routes the message on a sphere using stereographic projection. Curveball routing alleviates congestion near the center but does not have any bound on how much it can reduce maximum congestion. In our paper use area preserving map again to find a good solution for routing in a disk. In particular, by using the Lambert azimuthal projection, we map a disk to a sphere (or part of it). Then we route using greedy routing with spherical coordinates. This algorithm gives not only bounded maximum congestion, but also bounded path stretch. It is the first such algorithm known. Putting two methods together we have an algorithm that computes virtual coordinates for any simply connected domain such that greedy routing using this method has bounded stretch and bounded worst congestion. This is the first such result known for a general network.

In the following we first present the main idea for using area preserving maps in the continuous domain. We also show the bound on routing stretch and maximum load, compared to the optimal solution. The distributed implementation of the idea in a sensor network is presented afterwards together with simulation results.

3.2 Load Balancing Using Area Preserving Maps

3.2.1 Problem Definition

In this section we first describe load balancing in the continuous setting. Throughout the paper we consider a simply connected domain Ω (e.g., no holes). A routing scheme describes how to find a path between two points inside Ω.

Definition 3 (Routing Scheme) *Given a simply connected domain $\Omega \subset \mathbb{R}^2$, a Routing Scheme Γ is a way of specifying, for every pair of points (p, q) inside Ω, a path $\gamma_{p,q} \subset \Omega$ which connects p to q. The set Γ is the collection of all such paths.*

In the discrete graph setting, the traffic load is taken as the number of messages delivered along each edge/through each node. In the continuous setting, however, there could be various definitions of traffic load, e.g. [49, 50, 121, 124]. In [49] load is presented as a flux and load balancing is presented as a $\min\max$ problem. We will use the definition of load presented in [124]. The definition we present below is essentially the same, but described in a way more suited to our purposes. One should note that although these definitions are different, they nevertheless are comparable and the basic idea behind them is the same. Furthermore, one can expect the discrete definitions to agree with the continuous definitions in the case of a dense network.

Definition 4 (Load) *Given a routing scheme Γ for Ω, for any region $A \subset \Omega$, define the load of A (denoted as $\ell_\Gamma(A)$) by the following procedure:*

1. *Choose n source-destination pairs $\{(a_i, b_i)\}_{i=1}^n$ uniformly randomly inside Ω. For all such pairs, find γ_i, the paths as determined by Γ.*
2. *Let $X_n(A)$ be the expected length of intersection of the γ_i with A.*
3. *Define*

$$\ell_\Gamma(A) = \lim_{n\to\infty} \frac{X_n(A)}{\mathit{Area}(A)} \tag{7}$$

Next, define the load at a point p in the domain by:

1. *Choose a nested sequence of neighborhoods A_n whose intersection is p and which satisfy $\mathit{Area}(A_n) \to 0$ as $n \to \infty$;*

2. *Define*

$$\ell_\Gamma(p) = \lim_{n\to\infty} \ell_\Gamma(A_n) \tag{8}$$

The load balancing problem we will attack throughout the remainder of this paper can be formulated now as

Definition 5 (Optimal Load Balanced Routing Problem) *Given a simply connected domain Ω in the plane, find the routing scheme Γ such that*

$$\max_{p\in\Omega} \ell_\Gamma(x) \leq \max_{p\in\Omega} \ell_{\Gamma'}(x)$$

for any other routing scheme Γ' on Ω.

3.2.2 Area Preserving Map to Disk

First we show how we use area preserving maps to reduce the problem on an arbitrary domain to that on a disk. For some domains (e.g. the square) a closed form for an area-preserving mapping to the disk is available. However, this is not generally always possible and numerical methods are required.

We start with a simply connected domain (with finite area) whose boundary is a piece-wise smooth curve. For our purposes, we can assume that the boundary is a polygon with k vertices. This is reasonable because any smooth boundary can be approximated by polygons. Note that k is often much smaller than the number of sensors, n, inside the domain.

Let Ω denote the interior of such a polygonal region, with $\mathcal{P}$ being the boundary polygon. Our goal is to map Ω to a disk D centered at the origin (of area equal to that of Ω) in an area-preserving way. Consider a real valued function f defined on Ω in such a way that the level sets of this function are simple closed curves that fill up Ω. We will call f the *contour generating function*. Given such an f, we will first define an area-preserving mapping from Ω to the disk in terms of f. We will then show how to obtain f for any given Ω in a simple manner.

3.2.2.1 Area-Preserving map using the contour generating function

Let $\mathcal{C}$ be the family of contours of f, i.e. $\mathcal{C}$ is a set containing all simple closed curves $\gamma \subset \Omega$, such that there exists c_γ, $f(x, y) = c_\gamma$ for all points $(x, y) \in \gamma$. Let

$\mathcal{O}$ be a point interior to all the curves in $\mathcal{C}$, and let $\mathcal{L}$ be a path joining $\mathcal{O}$ to some point on $\mathcal{P}$ such that $\mathcal{L}$ intersects each curve in $\mathcal{C}$ once. We will assume that f has continuous first order partial derivatives everywhere except at O, and not both of the partials are zero at any point. See examples of contoured regions in Figure 2.

The area-preserving mapping $\phi : \Omega \to \mathcal{D}$ we use was first described in [21]. It has the following properties:

1. Every curve $\gamma \in \mathcal{C}$ is mapped to a circle inside $\mathcal{D}$ centered at the origin. Hence the contours are mapped to concentric circles.
2. The path $\mathcal{L}$ is mapped to any given radius of $\mathcal{D}$.

To construct ϕ, we first modify our f as follows. Define a function $\tilde{f}$ on Ω such that $[\tilde{f}(x,y)]^2$ at any point $(x,y) \in \Omega$ equals $1/\pi$ times the area inside the contour in $\mathcal{C}$ passing through (x,y), i.e.

$$\tilde{f}(x,y) = \sqrt{\frac{1}{\pi}\text{Area}(\text{ Int } \gamma_{(x,y)})}, \tag{9}$$

where $\gamma_{(x,y)} \in \mathcal{C}$ is the contour passing through (x,y). Choose polar coordinates (R,Θ) in $\mathcal{D}$ where Θ is measured from the given radius to which we want to map $\mathcal{L}$. The map $\phi : (x,y) \to (R,\Theta)$ is now given by :

$$R(x,y) = \tilde{f}(x,y) \tag{10}$$

$$\Theta(x,y) = \frac{1}{\tilde{f}(x,y)} \int_{Y(x,y)}^{x,y} \frac{ds}{(\tilde{f}_x^2 + \tilde{f}_y^2)^{1/2}}, \tag{11}$$

where $Y(x,y)$ is the point of intersection of $\mathcal{L}$ with the curve in $\mathcal{C}$ through (x,y), $\tilde{f}_x$ and $\tilde{f}_y$ are the partial derivatives, and the integral is evaluated along the curve of $\mathcal{C}$ from $Y(x,y)$ to (x,y) in the direction for which the interior is on the left.

For the proof that this map is indeed area-preserving, we refer the reader to [21]. Observe that any contour γ passing through (x,y) gets mapped to a circle of radius $\tilde{f}(x,y)$, and by the definition of $\tilde{f}$, they have the same area. In general, there are many area-preserving mappings from Ω to $\mathcal{D}$; we choose the above one because of its nice property of mapping contours to circles. Furthermore, we will see later that by choosing $\mathcal{O}$ to be the point with the maximum load, we can achieve good results for load balancing. We show a simple example of a triangle mapping to the disk in Figure 3.

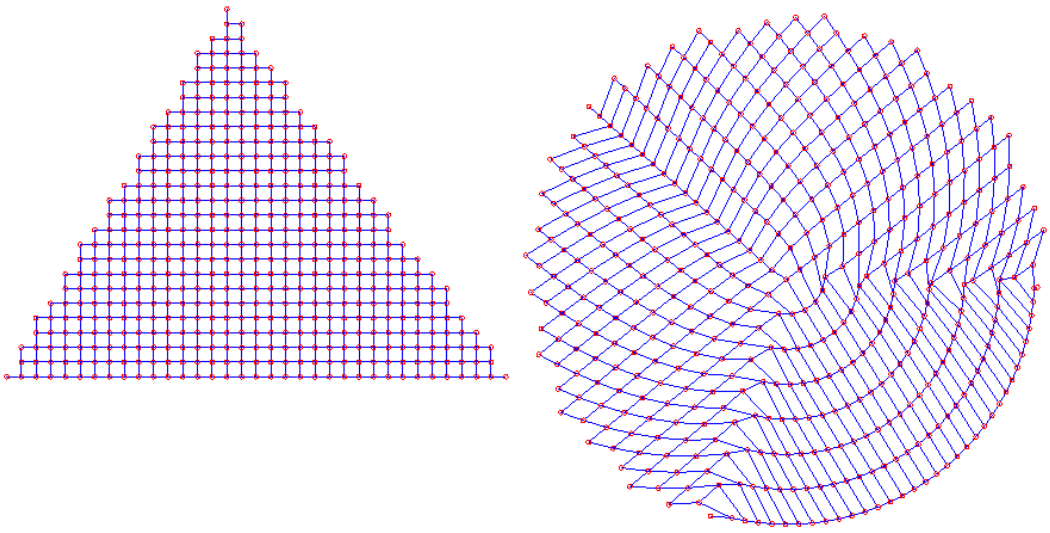


Figure 3: An example of area-preserving map from a triangle to a disk

3.2.2.2 Finding the contour generating function

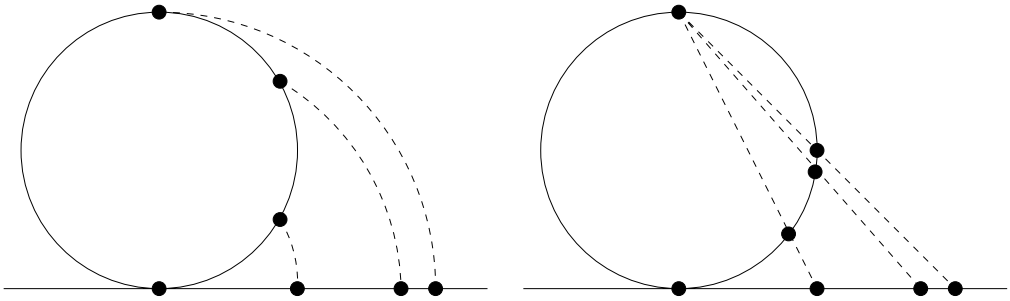
The previous section assumed the knowledge of f - the contour generating function. In this section we describe how we generate contours for an arbitrary domain.

For many shapes, the contours can be generated easily just by shrinking the boundary by appropriate factors. For instance, consider the case of star-shaped polygons $\mathcal{P}$; one can find the center of this polygon p (the point from which the entire polygon is "visible") and find the distance of every point x on the boundary to p. Let L_{px} denote the line segment between p and x and ℓ_{px} denote its length. Define the ith contour γ_i by

$$\gamma_i = \{q_x \in L_{px} : \text{distance}(q_x, p) = \epsilon_i \ell_{px}; x \in \mathcal{P}\}$$

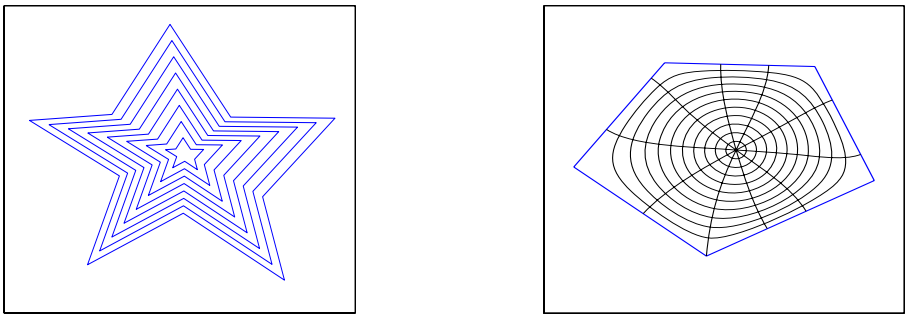
where the $\epsilon_i \in (0, 1)$ are constants. For examples of this, see the contoured domain on the left in Figure 2. Note that the contours attained in this way are non-differentiable, i.e. they will have corners.

We describe next one elegant and simple method to get a (smooth) contour generating function for any domain Ω. Let $g : \Omega \to \mathbb{D}$ be the *conformal mapping* from the polygon to the disk. For conformal mapping between domains whose



(i) (ii)

Figure 4: A cross sectional view of the sphere and a plane tangent to it at south pole. (i) Area preserving map: each point on the sphere (except the northpole) is projected to the plane along a circular arc centered at the point of tangency between the sphere and plane. (ii) Stereographic map: a point p on the plane is mapped to the intersection of the line through p and the north pole with the sphere.



(i) (ii)

Figure 5: Different methods for finding contour generating function. (i) Contoured star-shaped polygon using shrinking boundary method. (ii) Contoured pentagon using conformal mapping method.

boundary is a polygon and the disk, there is a nice formula called the *Schwarz-Christoffel* formula which was developed by the German mathematicians Hermann Amandus Schwarz and Elwin Bruno Christoffel in the mid-19th century. Various softwares implementing this mapping are widely available, including a MATLAB toolbox. The Schwarz-Christoffel formula [46] describes the mapping of the upper half plane $\mathbb{H}$ onto the interior of a simple polygon.It maps the real line (which is the boundary of $\mathbb{H}$) onto the boundary of Ω, which is the the polygon $\mathcal{P}$. This mapping $h : \mathbb{H} \to \Omega$ (with continuous extension to the boundary) is given by:

$$h(z) = A \int^{z} \prod_{k=1}^{n} (\zeta - z_k)^{-\beta_k} d\zeta + B, \tag{12}$$

where the pre-image of w_k (the vertices of the polygon) on the real line is z_k, i.e., $w_k = h(z_k)$; the polygon tangent turning angle at w_k is $\beta_k \pi$; A and B are two constants. The unit disk $\mathbb{D}$ is conformally equivalent to $\mathbb{H}$ via the map $e : \mathbb{D} \to \mathbb{H}$ given by

$$e(z) = i\frac{1+z}{1-z}$$

Using this, we get a map $\tilde{h} : \mathbb{D} \to \Omega$ given by $\tilde{h} = h \circ c$. This is related to our conformal map g from Ω to $\mathbb{D}$ simply by $g = \tilde{h}^{-1}$.

Now we define $f : \Omega \to [0, 1]$ by $f(x, y) = |g(x, y)|$ where $|g(x, y)|$ denotes the distance of $g(x, y)$ from the origin. It is now clear that $f^{-1}(a)$ for some $a \in [0, 1]$ will be a simple closed curve γ inside Ω. Furthermore, the pre-image of the interval $[0, 1]$ can be seen to be a curve joining $\mathcal{O}$ to $g^{-1}(1)$ which intersects every contour only once. We define $\mathcal{L} := g^{-1}([0, 1])$ and then use the mapping described in the previous section. The second figure in Figure 2 illustrates this method for a pentagon. As mentioned, the contours are smoothened out here. Care should be taken when deciding on which point inside the polygon is to be mapped to the origin on the unit disk; essentially the point should not be too close to the boundary of the polygon, since then the contours would no longer be uniformly dense.

3.2.2.3 Approximation load balancing solution

In this section we show why using area-preserving maps guarantees an approximate solution to the load balancing problem on the original domain Ω. Specifically,

we will show that, given a factor c approximation to the solution of the $\min\max$ problem on the disk, we can achieve a factor $cD(\phi_\Omega)$ approximation to the $\min\max$ problem on an arbitrary domain Ω, where $D(\phi_\Omega)$ is a constant depending only on the area-preserving map between Ω and the disk. Later we discuss load balanced routing solutions for a disk with constant approximation factor c.

To describe our bounds, we need to define the Jacobian of our mapping. Set $u := R\cos\Theta$ and $v := R\sin\Theta$ where R and Θ are the same as in Equations 10 and 11, but now we use the conformal mapping g and the contour function f described in the previous section. The new equations for R and Θ now become:

$$R(x,y) = \tilde{f}(x,y) := \sqrt{\frac{1}{\pi}\text{Area (Int }\{f^{-1}(f(x,y))\})} \tag{13}$$

$$\Theta(x,y) = \frac{1}{\tilde{f}(x,y)} \int_{g^{-1}(f(x,y))}^{x,y} \frac{ds}{(\tilde{f}_x^2 + \tilde{f}_y^2)^{1/2}} \tag{14}$$

The area-preserving map can now be written in axis-parallel coordinates as $\phi : (x,y) \to (u,v)$. The Jacobian matrix is a 2×2 matrix of the partials of u and v with respect to x and y. By change of variable formulae, one can see that the following equations hold:

$$\begin{aligned}
\frac{\partial u}{\partial x} &= (\cos\Theta)\frac{\partial R}{\partial x} - (R\sin\Theta)\frac{\partial \Theta}{\partial x} \\
\frac{\partial u}{\partial y} &= (\cos\Theta)\frac{\partial R}{\partial y} - (R\sin\Theta)\frac{\partial \Theta}{\partial y} \\
\frac{\partial v}{\partial x} &= (\sin\Theta)\frac{\partial R}{\partial x} + (R\cos\Theta)\frac{\partial \Theta}{\partial x} \\
\frac{\partial v}{\partial y} &= (\sin\Theta)\frac{\partial R}{\partial y} + (R\cos\Theta)\frac{\partial \Theta}{\partial y}
\end{aligned}$$

The partials of R and Θ with respect to x and y cannot be written down in a nice form, but one can nevertheless form the matrix $J(\phi)$ with the four entries given as above. Now, ϕ is area-preserving, which is equivalent to the determinant of $J(\phi)$ being identically 1 at all points in the domain. Let $\lambda_1(x,y), \lambda_2(x,y)$ denote the two eigenvalues of $J(\phi)$ (not necessarily real). Note that $\lambda_1\lambda_2 = 1$ at every point, and hence

$$|\lambda_1| = \frac{1}{|\lambda_2|}$$

Let $d(x,y) = \max(\lambda_1(x,y), \lambda_2(x,y))$, which is the maximum length distortion at point (x,y). When $p = (x,y)$, we will just write $d(p)$. Let

$$d(\phi_\Omega) = \sup_{p \in \overline{\Omega}} d(p) \tag{15}$$

The following theorems hold for any area preserving map ϕ between two simply connected domains Ω and $\Omega^{'}$ with the same total area. The first theorem tells us how the load function changes after applying the mapping and the subsequent ones then compare the $\min\max$ solutions. Note that in our case, $\Omega^{'} = \mathcal{D}$, the disk with total area equal to that of Ω.

Given $\phi : \Omega \to \Omega^{'}$ area-preserving, define $J(\phi)$ and $d(p)$ analogously, and let $d(\phi_{\Omega,\Omega'}) = \sup_{p \in \overline{\Omega}} d(p)$.

Theorem 6 *Given a routing scheme Γ on Ω and $\phi : \Omega \to \Omega^{'}$ area-preserving, denote by $\Gamma^{'}$ the routing scheme on $\Omega^{'}$ which allots the path $\phi(\gamma)$ to the pair (a,b), where γ is the path joining $\phi^{-1}(a)$ and $\phi^{-1}(b)$ as dictated by Γ. Then*

$$\frac{1}{d(p)} \ell_\Gamma(p) \leq \ell_{\Gamma'}(\phi(p)) \leq d(p) \ell_\Gamma(p) \; \forall p \in \Omega \tag{16}$$

Proof 7 *Fix an $\epsilon > 0$. By continuity of $J_\phi(x,y)$, $\exists \delta > 0$ such that for any $p^{'}$ in the disk of radius δ around p (denoted as $B_\delta(p)$),*

$$d(p^{'}) \;<\; d(p) + \epsilon, \textit{and} \tag{17}$$

$$\frac{1}{d(p^{'})} \;>\; \frac{1}{d(p)} - \epsilon \tag{18}$$

Let $B_\delta(p) \supset B_{\delta_1}(p) \supset B_{\delta 2}(p) \supset \cdots$ be a sequence of nested neighborhoods of p such that $\cap_{i=1}^{\infty} B_{\delta_i}(p) = p$. We will calculate the load at $\phi(p)$ using the sequence $\phi(B_{\delta_i}(p))$. Note that $\phi(B_{\delta_i}(p))$ has the same area as $B_{\delta_i}(p)$.

Pick n source destination pairs uniformly randomly in $\Omega^{'}$. Since ϕ is measure preserving, this amounts to picking n pairs uniformly randomly in Ω and looking at their images under ϕ. Consider any source destination path γ (joining two points a and b) intersecting $B_{\delta_i}(p)$ in Ω. The fact that ϕ is a homeomorphism implies that $\phi(\gamma)$ intersects the neighborhood $\phi(B_{\delta_i}(p))$ of $\phi(p) \in \Omega^{'}$.

Now let $\gamma_i = \gamma \cap B_{\delta_i}(p)$. As a consequence of Equations 17 and 18, the length of $\phi(\gamma_i)$, denoted as $|\phi(\gamma_i)|$ satisfies

$$\left(\frac{1}{d(p)} - \epsilon\right)|\gamma_i| < |\phi(\gamma_i)| < (d(p) + \epsilon)|\gamma_i| \tag{19}$$

as a consequence of which the load of $B_{\delta_i}(p)$ and $\phi(B_{\delta_i}(p))$ (both of which have the same area) satisfy

$$\left(\frac{1}{d(p)} - \epsilon\right)\ell_\Gamma(B_{\delta_i}(p)) < \ell_{\Gamma'}(\phi(B_{\delta_i}(p))) < (d(p) + \epsilon)\ell_\Gamma(B_{\delta_i}(p))$$

Taking limit as $i \to \infty$ gives

$$\left(\frac{1}{d(p)} - \epsilon\right)\ell_\Gamma(p) \leq \ell_{\Gamma'}\phi(p) \leq (d(p) + \epsilon)\ell_\Gamma(p) \;\; \forall p \in \Omega \tag{20}$$

Since $\epsilon > 0$ chosen above was arbitrary, this proves the theorem.

Given a routing scheme Γ, we will denote the maximum load according to this routing scheme as ℓ_Γ. The next theorem shows how the optimal load balanced routing solution to Ω is related to the optimal load balanced routing solution on $\Omega^{'}$.

Theorem 8 *Let Γ^* and Ψ^* be the optimal load balanced routing schemes on domains Ω and $\Omega^{'}$ respectively, i.e.*

$$\ell_{\Gamma^*} \leq \ell_\Gamma \;\; \textit{and} \;\; \ell_{\Psi^*} \leq \ell_\Psi \tag{21}$$

for all routing schemes Γ on Ω and Ψ on $\Omega^{'}$. Let ϕ be an area-preserving map from Ω to $\Omega^{'}$ and define $d(\phi_{\Omega,\Omega'})$ as above. Then

$$\frac{1}{d(\phi_{\Omega,\Omega'})}\ell_{\Gamma^*} \leq \ell_{\Psi^*} \leq d(\phi_{\Omega,\Omega'})\ell_{\Gamma^*} \tag{22}$$

Proof 9 *We first prove $\ell_{\Psi^*} \leq d(\phi_{\Omega,\Omega'})\ell_{\Gamma^*}$. Let $\Gamma^{'}$ denote the push-forward of the routing scheme Γ^* via ϕ, i.e. $\Gamma^{'}$ allots the path $\phi(\gamma)$ to source-destination pair (a, b) inside $\Omega^{'}$, where γ is the path joining $\phi^{-1}(a)$ to $\phi^{-1}(b)$ as dictated by Γ^*.*

Since Ψ^* is optimal, we know that $\ell_{\Psi^*} \leq \ell_{\Gamma'}$. Furthermore, by Theorem 6, we know that $\ell_{\Gamma'} \leq d(\phi_{\Omega,\Omega'})\ell_{\Gamma^*}$, since the two routing schemes arerealted by ϕ. Combining these two inequalities we obtain the claimed result.

Similarly, by interchanging the roles of Ω and $\Omega^{'}$, and by noting that the eigenvalues of $J(\phi^{-1})$ (the Jacobian of ϕ^{-1}) are the inverse of the eigenvalues of $J(\phi)$, we get that $\ell_{\Gamma^*} \leq d(\phi_{\Omega,\Omega'})\ell_{\Psi^*}$, which proves the other inequality.

Remark. The above theorem holds for *any* area-preserving homeomorphism between the two domains. The maximum traffic load in optimal routing solutions on the two domains are bounded from each other by the maximum length stretch of the area preserving map. Different from other approximation algorithms in the literature in which the constant bound is a fixed value, here clearly the length stretch is a property of the shape of Ω – the more similar Ω is to a disk, the smaller the stretch is. Besides, not only does the above theorem help us prove the approximation guarantee of our method, it also gives non-trivial lower bounds for the best achievable load on any given domain. This has not been accomplished before. It can be intuitively understood that any routing algorithms on domains with narrow bridges/cuts necessarily create high traffic load at the bridge area – the shape matters. This theorem makes it more precise. If our area preserving mapping has length distortion at most say δ, then we know that the optimal on the domain is at least $1/\delta$ times the minimum maximum load on the disk, which is shown to be at least[1] 0.45 [124].

The next theorem tells us how to obtain an approximate solution on the original domain by using an approximate (or exact) solution on the target.

Theorem 10 *Let Ψ^c be a factor c approximation of the optimal solution to the* $\min\max$ *problem on $\Omega^{'}$, i.e. $\ell_{\Psi^c} \leq c\ell_{\Psi^*}$, where Ψ^* is as in Theorem 8. Let Γ be the routing scheme on Ω that allots the path $\phi^{-1}(\gamma)$ to the source-destination pair (a, b) inside Ω, where γ is the path between $\phi(a)$ and $\phi(b)$ as dictated by Ψ^c. Let Γ^* be the optimal routing scheme on Ω. Then there exists a constant $D(\phi_{\Omega,\Omega'})$ depending only on the area-preserving mapping between Ω and $\Omega^{'}$ such that*

$$\ell_\Gamma \leq cD(\phi_{\Omega,\Omega'})\ell_{\Gamma^*} \tag{23}$$

Proof 11 *Again, since Γ and Ψ^c are related by ϕ^{-1}, and since the eigenvalues of $J(\phi^{-1})$ are inverses of eigenvalues of $J(\phi)$, we have that $\ell_\Gamma \leq d(\phi_{\Omega,\Omega'})\ell_{\Psi^c}$. We now have the follwing string of inequalities*

[1] 0.45 is the average load of a node in the disk when one uses shortest path routing. Clearly, no other routing scheme can have a maximum load smaller than this, because the maximum of this routing scheme must be larger than its average, which in turn must be larger than the average of shortest path routing since it minimizes the total load. The number 0.45 can be explained in the network setting intuitively as follows. Take any node p in the network and draw a small disk of radius ϵ around it. Consider all shortest paths that go through this disk and find the length of the part lying in this disk. Then the total length of these parts (over all such paths) is $(0.45)\pi\epsilon^2$.

$$\begin{aligned} \ell_\Gamma \leq d(\phi_{\Omega,\Omega'})\ell_{\Psi^c} &\leq cd(\phi_{\Omega,\Omega'})\ell_{\Psi^*} \\ &\leq cd^2(\phi_{\Omega,\Omega'})\ell_{\Gamma^*} \end{aligned} \tag{24}$$

where the second inequality follows by the hypothesis on Ψ^c and the last inequality follows by Theorem 8. Setting $D(\phi_{\Omega,\Omega'}) = d^2(\phi_{\Omega,\Omega'})$, we obtain the claimed result.

This theorem, translated in our setting by putting $\Omega^{'} = \mathcal{D}$, says that if we can find an approximate solution to the load balancing problem on the disk, just pulling it back via our area-preserving mapping described in Section 3.2.2.1 gives us an appoximate solution to the load balancing problem on the domain Ω.

3.2.2.4 Length stretch bound

It is clear that any other routing scheme on the domain Ω will generate longer paths than shortest path routing. In particular, any attempt for load balancing will necessarily make the average path length longer and the total traffic load higher. Now we will show that by using area preserving map ϕ, we also bound the path stretch by a constant factor dependent on $d(\phi)$. Assume that Γ and Ψ are shortest path routing schemes on Ω and $\mathcal{D}$ respectively, and let Ψ^c be a routing scheme on the disk such that the path it generates between a source-destination pair (u, v) is at most c times the length of the shortest path between u and v, for all pairs (u, v). We will write this as

$$|\mathcal{R}_{\Psi^c}(u, v)| \leq c|R_\Psi(u, v)|$$

here $\mathcal{R}_\Psi(u, v)$ denotes the route between u and v under Ψ and $|.|$ denotes its length.

Now let ϕ be an area-preserving map from Ω to $\mathcal{D}$, and define $d(\phi_\Omega)$ as in Equation 15. Let $\Gamma^{'}$ be the pull-back of Ψ^c via ϕ.

Theorem 12 *Under the above hypothesis, there exists a constant $D(\phi_\Omega)$ depending only on the area-preserving map ϕ such that*

$$|\mathcal{R}_{\Gamma'}(a, b)| \leq cD(\phi_\Omega)|\mathcal{R}_\Gamma(a, b)| \;\; \forall a, b \in \Omega \tag{25}$$

Proof 13 *Let Ψ^0 be the push-forward of Γ (the shortest path routing on Ω) via ϕ. From the proof of Theorem 6, one can see that the image of a route $\mathcal{R}$ in Ω (denoted as $\phi(\mathcal{R})$) has length at most $d(\phi_\Omega)$ times that of $\mathcal{R}$. We now have the following string of inequaities:*

$$\begin{aligned} |\mathcal{R}_{\Gamma'}(a,b)| &\leq d(\phi_\Omega)|\mathcal{R}_{\Psi^c}(\phi(a),\phi(b))| \\ &\leq cd(\phi_\Omega)|\mathcal{R}_{\Psi}(\phi(a),\phi(b))| \\ &\leq cd(\phi_\Omega)|\mathcal{R}_{\Gamma^*}(\phi(a),\phi(b))| \\ &\leq cd^2(\phi_\Omega)|\mathcal{R}_{\Gamma}(a,b)| \end{aligned} \tag{26}$$

where the second inequality follows by the hypothesis on Ψ^c, the third by the fact that shortest path routing Ψ will have shorter lengths than Γ^* and the last inequality follows by the same observations as the first (since Γ^* is the image of Γ, the shortestpath routing on Ω).

Setting $D(\phi_\Omega) = d^2(\phi_\Omega)$, the theorem is proved.

Note that the above theorem also holds for any domain Ω' and an area-preserving mapping ϕ from Ω to Ω'. We will show in the simulation section that in all practical scenarios, the stretch is much smaller than the claimed constant and that we always overestimate the maximum stretch in our proof.

3.2.3 Load Balanced Routing in a Disk

Having reduced the problem on an arbitrary domain to that on a disk, in this section we focus solely on the problem of routing in a disk. The problem of finding the optimal load balanced routing inside a disk was initiated in [124]. The authors examined properties of the optimal solution and then developed a numerical solution for the optimum. This solution is not practical. In addition, the authors also used a technique called *Curveball Routing* which first maps the disk to the sphere using *stereographic projection*, an angle-preserving mapping. The nodes in the disk are then assigned virtual coordinates on the sphere, and routing is done using the spherical metric in a greedy fashion.

Although this is an efficient heuristic, there are no results as to how far this solution is from the $\min\max$ solution for the disk (the only available bound is that on the maximum stretch using this routing, published in [96]).

We will now propose a routing scheme on the disk which is both a constant factor from the optimal $\min\max$ solution and also has bounded stretch compared to shortest path routing. The theorems in the last section then help us get the routing scheme on Ω with the same properties.

Instead of stereographic projections, one can also use the *Lambert azimuthal equal-area projection* from the disk to the sphere. Denote the (open) disk of radius r centered at the origin in the plane by $\mathbb{D}_r$, and set $\mathbb{D} := \mathbb{D}_1$. The Lambert azimuthal projection, $g : \mathbb{D}_2 \to S^2 \setminus \{(0,0,1)\}$ is the area preserving map from the disk of radius 2 to the sphere of radius 1 (centered at the origin in 3-D) minus its north pole given by:

$$g(x,y) = \left(x\sqrt{1 - \frac{x^2+y^2}{4}}, y\sqrt{1 - \frac{x^2+y^2}{4}}, \frac{x^2+y^2}{2} - 1 \right) \tag{27}$$

One can see that g maps the disk of radius $\sqrt{2}$ to the lower hemisphere, the circle of radius $\sqrt{2}$ to the equator, and the remainder of the bigger disk with radius 2 to the upper hemisphere. Moreover, it is clear that just by scaling the original network, one can arrange for the map to cover different parts of the sphere. A nice property of this map is that it is area-preserving, i.e. the area of any region in the domain is equal to the area of its image under g. A comparison of the area preserving map and the stereographic projection is shown in Figure 4.

To route on the disk, we give the nodes virtual coordinates on the half sphere (chosen for convenience), and route on the half sphere greedily using the spherical metric. Note that depending on how much we want the final routes to be pushed towards the boundary of the original domain, we increase the part of the sphere we cover. Thus we scale our disk to a radius close to 2 if we want to push routes more towards the boundary and closer to $\sqrt{2}$ if we want them pushed lesser.

Let $\mathcal{S}_H$ denote the lower half sphere, and Γ denote the greedy routing using spherical metric on it. For two nodes $a, b \in \mathcal{D}_{\sqrt{2}}$, we choose the path $g^{-1}(\mathcal{R}_{g(a),g(b)})$, where $\mathcal{R}_{g(a),g(b)}$ is the shortest path between $g(a)$ and $g(b)$ on $\mathcal{S}_H$ using Γ. Denote this routing scheme on $\mathcal{D}_{\sqrt{2}}$ as Ψ. Furthermore, let Δ denote shortest path routing

on the disk. We first show that paths under Ψ are not arbitrarily longer than paths in Δ.

3.2.3.1 Bounded Stretch

Theorem 14 *For any source destination pair (a, b) inside $\mathcal{D}$,*

$$|\mathcal{R}_{\Psi}(a, b)| \leq 4|\mathcal{R}_{\Delta}(a, b)| \tag{28}$$

Proof 15 *First we will need to calculate the maximum stretch in any direction at a point. We will prove that given any curve $\gamma \in \mathcal{D}_{\sqrt{2}}$, its length is stretched by at most a factor of 2 under the Lambert azimuthal map.*

Note that the constant 4 appearing above is a worst-case scenario; which can actually be shown to never occur. In our simulations, the stretch was never larger than 1.9.

3.2.3.2 Approximation of Optimal

Now, let Ψ^* be the optimal routing schemes on $\mathcal{D}_{\sqrt{2}}$. We will now give a bound on the constant c by which Ψ is away from Ψ^*. Let β denote the maximum load (which occurs at the south pole) when one uses greedy routing on the lower half sphere. For the load distribution on the lower half sphere under greedy routing, one can perform a similar integration as in Theorem 1 in [124] and find β exactly.

Theorem 16 *Under the above hypothesis,*

$$\ell_{\Psi} \leq (4.5)\beta\ell_{\Psi^*} \tag{29}$$

Combined with an obvious bound of $\beta < 2\pi$, which is the maximum load at the center of a disc using greedy routing (Theorem 1 in [124]), we see that the constant c mentioned above is at most 9π. Again, we would like to remind the reader that a more detailed analysis might give better bounds. In the simulation section we compare the maximum load of this routing (coming from Lambert azimuthal map) to the approximate optimal routing on the disk given in [124]; they appear to be very close.

3.3 Algorithm

Having laid out the theoretical foundation for our method, we will now proceed to describe how to implement it in a distributed fashion. Assume that the sensors are distributed densely uniformly inside a simply connected domain Ω. Basically, there are four steps in our algorithm:

1. Compute the contour generating function using conformal mapping or the shrinking method.
2. Compute the area-preserving map and give virtual coordinates to the sensor nodes in the disk.
3. Compose with the area-preserving Lambert azimuthal map from the disk to the sphere and give virtual coordinates to the sensor nodes on the sphere.
4. Route greedily on the sphere using these virtual coordinates and the spherical metric and follow the corresponding path in the original domain.

Note that instead of 3, one can also use stereographic projection to map the disk to the sphere, i.e., use curveball routing on the disk. Moreover, after Step 2, one can also use the approximate optimal scheme described in [124] and route directly on the disk using this routing scheme, without going to the sphere. We now proceed to describe the steps in detail.

3.3.1 Compute Contour Generating Function

The goal of this step is for every sensor node z to find its image in the unit disk under the conformal mapping $g(z)$. As mentioned in section 3.2.2.2, the map e given by $e(z) = i(1+z)/(1-z)$ maps the unit disk to the half plane conformally (with inverse given by $e^{-1}(z) = (z-i)/(z+i)$), so it is enough to find the point w such that $h(w) = z$, where h is given by Equation 12. The map h depends only on the polygon $\mathcal{P}$ - boundary of the domain Ω. Another freedom in the Schwarz-Christoffel mapping is that we can choose which node in Ω to be mapped to the origin of disk. For our purposes, we feel it is reasonable to map the node with expected high traffic load (e.g., the centroid of $\mathcal{P}$) under the shortest path routing scheme to the origin. This is because the method that we use to alleviate load in

the disk assume the center to be overloaded and then try to route away from it. However, this node should not be very close to the boundary as stated earlier, for so the contours would not be uniformly dense; if this happens, we choose an arbitrary node closer to this node, but is farther from the boundary.

With all this information, a single node (e.g., the base station) computes the prevertices z_k on the real line, and also the constants A and B. Only the information about the prevertices z_k, the polygon angles β_k, and the constants A and B in the Schwarz-Christoffel mapping h is relayed to all nodes via flooding. Using the functional form of h in Equation 12, the nodes individually compute their own preimages and compose with e^{-1} as above to find their coordinates in the disk. Once every sensor node with original coordinate z has found $g(z)$ (its image under the conformal map to the disk), it calculates $f(z)$ which is just the distance of $g(z)$ from the origin. $f(z)$ is the contour generating function.In this method, the information relayed is only $O(k)$, where $k \ll n$ is the number of vertices of the polygon and is much smaller than n, the number of sensors in the domain. The contour function for the shrinking method for star-shaped polygons is just the distance of every node to the center of the star, which can be computed by each node individually by routing to the center node.

3.3.2 Compute the Area-Preserving Map

Now we describe how to map Ω to $\mathcal{D}$ (the disk of area equal to that of Ω) in a distributed manner. For ease of description we assume first that $\mathcal{D}$ is a unit disk; the method described below can easily be generalized for a larger disk. Recall that the nodes on the same contour (the same value under f) will be mapped to the same circle in $\mathcal{D}$. Thus there are two steps for computing the map. First we need to form the level set of f in a distributed way. Second we will compute the virtual coordinate for each node under the area preserving map.

There has been previous work on finding contours, one of such robust algorithms in a distributed setting is to use the *cut locus* [157, 166]. In fact, in this setting the level sets are simple – f has only a single minimum inside Ω and no saddles. We discretize the interval $[0, 1]$ into $1/\varepsilon$ intervals of width ε each. Here ε depends on the density of the nodes in the network and is in the order of $O(\pi/n)$.

We now find one contour cycle γ_i for each interval $[i\varepsilon, (i+1)\varepsilon]$, $0 \leq i \leq n/\pi$. The set of nodes whose values are inside $[i\varepsilon, (i+1)\varepsilon]$, denoted as C_i, naturally occupies an annulus. Note that we will choose ε according to the network density such that C_i is connected. One node, denoted as the root of this contour, r_i floods within C_i. For the purpose of the computations later, we will choose the roots r_i for different levels along a path that maps to a radius under the conformal map g. That is, the nodes whose projection under g on the positive x-axis (or is the closest to the x-axis among neighbors) will be chosen as the roots for the contours. After the flooding from r_i in C_i, the cut locus is defined as a pair of neighboring nodes whose shortest paths to the root are different and far apart. Then connecting the shortest path for the cut pair will give us a closed cycle $\gamma_i = \{z_{ij}\}$ representing the contour at the range $[i\varepsilon, (i+1)\varepsilon]$. The nodes within the band that are not selected to be on the contour cycle will be rounded to the nearest node on the contour cycle and will be handled later. Remark that the flooding is only restricted to the nodes inside the annulus and flooding for different levels does not overlap. Thus the total communication cost is linear in the number of nodes.

At the end of this procedure, we have a closed contour γ_i with a root r_i. Now we can circulate another message along γ_i to calculate the area inside γ_i. Specifically the area of a non-self-intersecting polygon with vertices $(x_1, y_1), \cdots, (x_n, y_n)$ is

$$\frac{1}{2}\left(\begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + \cdots + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix}\right),$$

where $|M|$ is the determinant of M. Thus the message circulating on γ_i will compute the determinant involving the current node and the previous node on γ_i and summing up along γ_i will give the area inside γ_i when the message returns to r_i. Then r_i forms

$$\tilde{f}_i = \sqrt{\frac{1}{\pi}\text{Area}(\gamma_i)}$$

and sends this value to all nodes in the contour γ_i. This gives the radius of the circle that γ_i is mapped to. In polar coordinates (R, Θ), R for node z_{ij} lying on contour γ_i is just $\tilde{f}_i$, i.e $R(z_{ij}) = \tilde{f}_i, \forall j$. Now we will find the angular coordinate for each node of γ_i by computing a (discrete) integral along γ_i, which approximates the integral in Equation 14.

First we need to calculate the partial derivatives. For every node z_{ij} on the contour γ_i:

1. Locates neighbors x_{ij}^+, x_{ij}^-, y_{ij}^+, and y_{ij}^+, where x_{ij}^+ (y_{ij}^+) is the closest neighbor in the positive direction of the horizontal (vertical) line passing through z_{ij}, x_{ij}^- (y_{ij}^-) is the closest neighbor in the negative direction of the horizontal (vertical) line passing through z_{ij}. Find their values under $\tilde{f}$ through local communications.

2. Computes

$$v_x(z_{ij}) = \frac{(\tilde{f}(x_{ij}^+) - \tilde{f}(x_{ij}^-))}{\text{distance}(x_{ij}^+, x_{ij}^-)}; v_y(z_{ij}) = \frac{(\tilde{f}(y_{ij}^+) - \tilde{f}(y_{ij}^-))}{\text{distance}(y_{ij}^+, y_{ij}^-)}$$

and $v_{ij} = \frac{1}{\sqrt{(v_x(z_{ij}))^2 + (v_y(z_{ij}))^2}}$.

For computing $\Theta(z_{ij})$, we start from the root of the contour $r_i = z_{i0}$, for which $\Theta(r_i) = 0$. Now we just discretize the integral in Equation 14. Let $V_{i0} = 0$. A node z_{ij} gets a value $V_{i(j-1)}$ from node $z_{i(j-1)}$. It then adds the quantity $v_{ij}(\text{distance}(z_{ij}, z_{i(j-1)}))$ to this value and passes it as V_{ij} to node $z_{i(j+1)}$. In other words, the sum

$$V_{ij} = \sum_{k=0}^{j} v_{ij}(\text{distance}(z_{ij}, z_{i(j-1)}))$$

is updated by node z_{ij}. We finally define

$$\Theta(z_{ij}) = \frac{V_{ij}}{\tilde{f}_i}$$

In this way all the nodes on contours find their positions inside the disk in polar coordinates. Setting $u = R\cos\Theta$ and $v = R\sin\Theta$ gets them the cartesian coordinates. Denote this map as $\phi : (x, y) \to (u, v)$.

For nodes that are not on any contour in the above method, a simple "interpolation" function can be used. Pick such a node at original coordinate $z \in \Omega$. By simple local queries, this node can locate a quadrilateral $ABCD$ such that z lies in the interior of this quadrilateral, and all nodes at A,B,C and D lie on some contour. Hence by knowledge of this quadrilateral and the quadrilateral $\phi(ABCD)$, this node can compute $\phi(z)$. If we want coordinates inside the unit disk, a simple scaling by the radius of the disk suffices.

3.3.3 Routing on Virtual Coordinates

Having obtained the coordinates on the disk $\mathcal{D}$, we can follow three methods now:

Using Lambert azimuthal Projection. The coordinates obtained in the disk $\mathcal{D}$ can be lifted to the sphere using the Lambert azimuthal equal area projection as in Equation 27. Note that by scaling the disk $\mathcal{D}$ to disks of any desired radius, we can cover either the half sphere ($R = \sqrt{2}$) or more portions of the sphere. In our simulations we provide different values of R for which we apply this map.

Once the nodes have virtual coordinates on the sphere, routing proceeds greedily using the spherical metric.

Using Approximate Optimal. An approximate optimal solution to the load balancing problem on the disk was described in [124]. However, since the optimal is complicated, one cannot hope to get closed formulae for the route from a source to a destination. We refer the reader to Section 3.4 of [124] for details on how to route using this scheme. In our implementation, we weight each edge by its distance r to the center of the disk using the formula $1.8r^3 - 3.1r^2 + 2.3$. And we compute the shortest path in the weighted graph and compare the performance with our methods.

Using Stereographic Projection. This is similar to the method above, except that the projection used in this case form the disk to the sphere is the conformal stereographic projection. We refer the reader to [124] for details. Again, routing is done greedily using spherical metric on these virtual coordinates.

For an arbitrary domain, greedy routing can get stuck at a local minima. However, because our coordinates are either in the disk or on the sphere (both convex shapes), for a dense network it is very unlikely that greedy routing gets stuck. Moreover, if the density is not high enough and greedy routing does get stuck, we can still ensure delivery by using the contours. The node at which the packet is stuck can always use the "contour" routes to route along its contour until it finds a node that can continue greedy routing. Although this might change the load function, the effect would not be drastic as 1) Such instances would be rare due to convexity of the target domain in which the virtual coordinates lie and 2) The expected detour would be very small. In the next section, we always compare our newly developed methods to shortest path routing on the original domain Ω because greedy routing

can get stuck, in which case packets are not delivered, and our conclusions about the change in load function would be erroneous.

3.4 Simulations/Experimental Evaluations

To evaluate the load reduction and routing stretch of our algorithm, the experiments are performed in two phases; first we demonstrate the performance of our algorithm in the disk case. We then reduce the problem on an arbitrary domain to that on the disk using area-preserving mapping and use the previous algorithm on the disk. Our observations can be summarized as follows:

1. When the original network is a disk, Lambert azimuthal mapping performs at least as well as curveball routing when it comes to reducing load. It also does not suffer from the problem of a heavily loaded boundary, which is very apparent in the approximate optimal routing scheme on the disk. Loads in the region near the disk are slightly higher than those in the approximate optimal [124]. The maximum routing stretch of Lambert azimuthal method is less than 1.9.

2. For a general simply connected domain, applying the area-preserving map to the disk followed by any of the three methods above gives a major reduction in load over shortest path routing on the domain. The reduction is similar for a considerable range of varying shapes.

3.4.1 Routing on Disk

In the first phase of our experiments, we compared the area preserving map algorithm to shortest path routing, the optimum approximation [124], and stereographic projection method curveball [124].

We consider a unit disk network model with nodes on a perturbed grid; all nodes have the same radius of communication, and the communication range is set for each node to have an approximate degree of 8. Nodes are then given different virtual coordinates based on the type of routing algorithm. For example, curveball

routing uses stereographic mapping to sphere, and Lambert's area preserving mapping (Eqn. 27) projects the original coordinates onto a sphere minus the north pole (or a smaller portion of it).

For the case of a disk, the two quantities we are interested in are the average load over a circle of radius r (as a function of r) and the maximum load in the disk. For each routing method, we ran all-pairs shortest path over the network with a corresponding method for weighing edges.

For shortest path routing, the edge weight is simply set to be the Euclidean distance between nodes. For our method and curveball routing, since we are routing on the sphere by the virtual coordinates, the weight is the geodesic distance between these virtual coordinates. Optimum approximation is a little different from previous methods. Based on the optimal cost function in [124], each node is given a score based on its radius on the unit disk according to a polynomial. With this score, the edge between nodes is weighted by the product of the average score of each end-nodes and its Euclidean distance.

To evaluate load distribution on different areas of the unit disk network, we partition the disk into annuli of the same width. Fig 6 on a unit disk with 315 nodes and an average degree 7 insinuates that both Lambert's mapping and curveball reduce loads near the origin compared to shortest path routing; and do not do so at the cost of increasing loads at the boundary like the optimal approximation.

Fig. 7 and Fig. 8 illustrate the results in a unit disk graph with 2000 nodes. Since most of the paths used by shortest path routing pass through or near the center of the disk, both the average and maximum load of nodes are higher when closer to the center. By adjusting the radius of the disk for Lambert's mapping, we can "push out" the load from the center to the boundary. The maximum max load decreases by 37.8% of the shortest path at radius $r = 1.6$, is 19% lower than that of the optimal approximation (which performs well near the origin but increases towards the boundary), and is almost the same as the best achieved by Curveball routing at $r = 1.1$. For average load, the maximum average load drops to 40.2% of the shortest path routing ($r = 1.7$). Compared to curveball and the optimal approximation, the average load is about the same in area near the origin, but lower in the far away from origin areas.

To understand the stretch ratio of Lambert's mapping, we generate 10 random

networks with 300 to 1000 perturbed grid nodes with an average degree around 8, and map them to half sphere ($r = \sqrt{2}$). While Thm. 14 suggests an upper bound of 4, we actually got an average stretch ratio of 1.86, and it was never above 1.89. Note that the stretch factor for curveball is around 1.6.

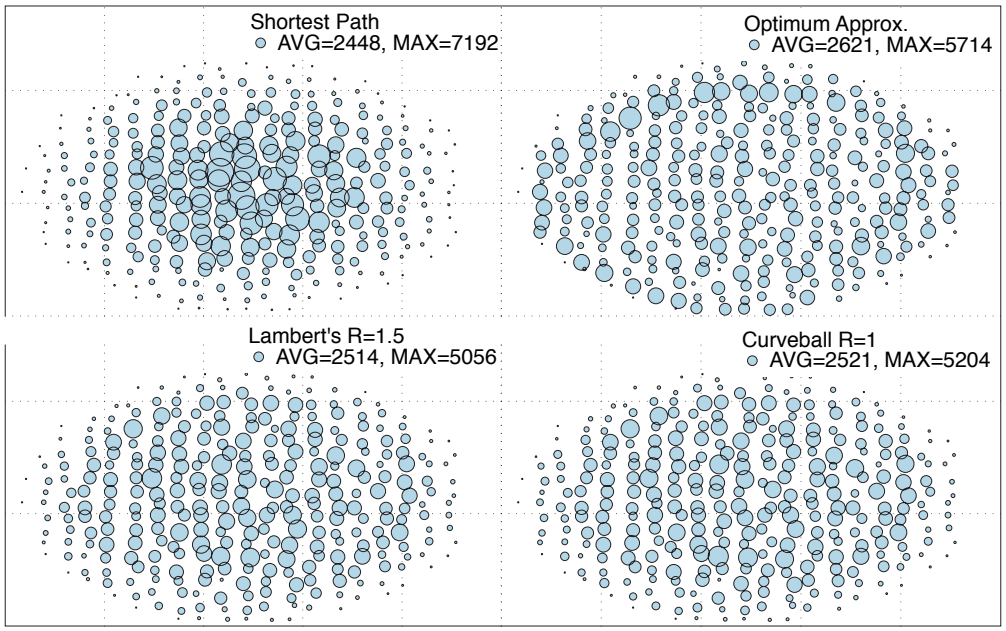


Figure 6: Load distribution of each routing algorithm in the unit disk network. In this figure, each node is represented by a circle, and the diameter of circle is proportional to the traffic load at that node.

3.4.2 Routing on a Simply Connected Domain

We now evaluate our algorithm on differently shaped simply connected domains. As mentioned in Sec. 3.2.2.1, for a given simply connected domain, the area preserving map gives virtual coordinates in the unit disk while maintaining the traffic pattern. In this way, any load balancing algorithm over unit disk can help benefit the process of balancing load on an arbitrary domain.

Fig. 9 shows some examples of area preserving maps over domains. The point

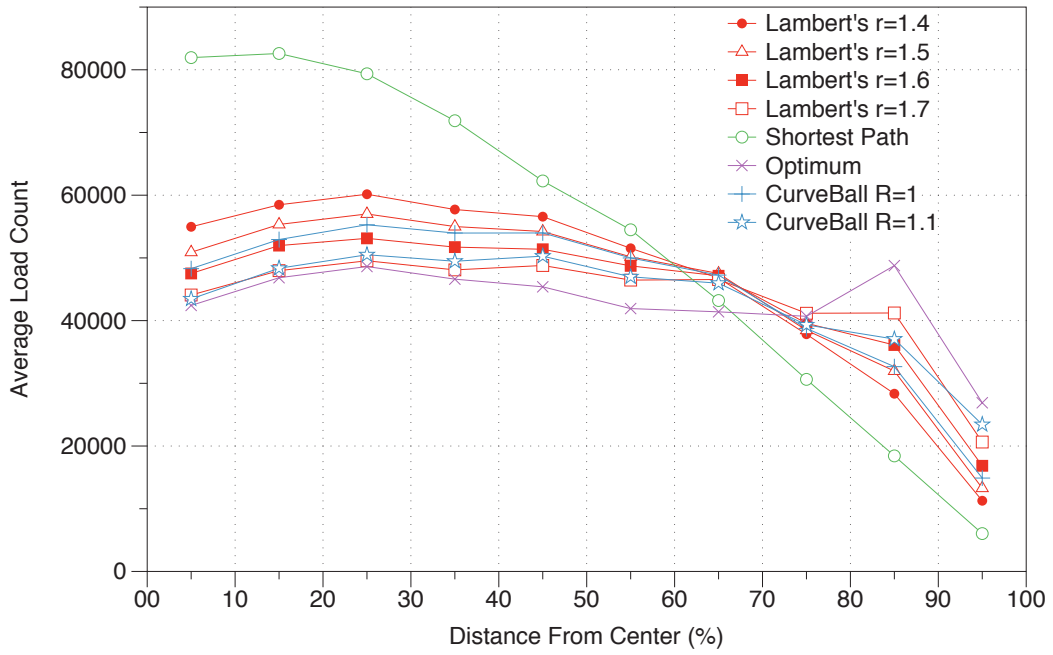


Figure 7: Histogram of the average load for a unit disk network, as a function of distance from the center.

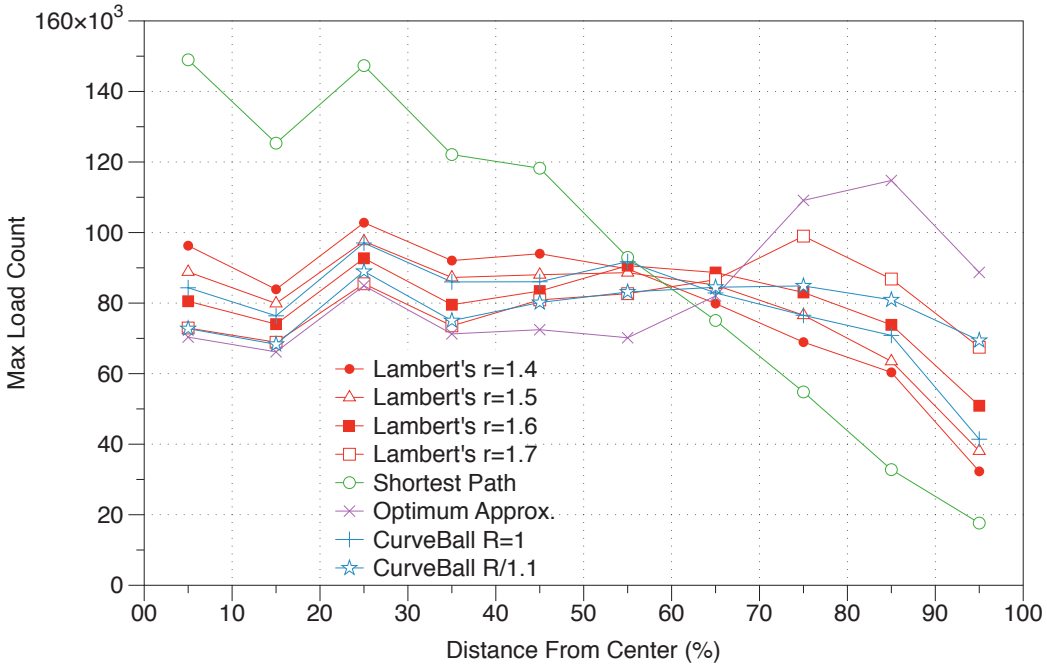


Figure 8: Histogram of the maximum load for a unit disk network.

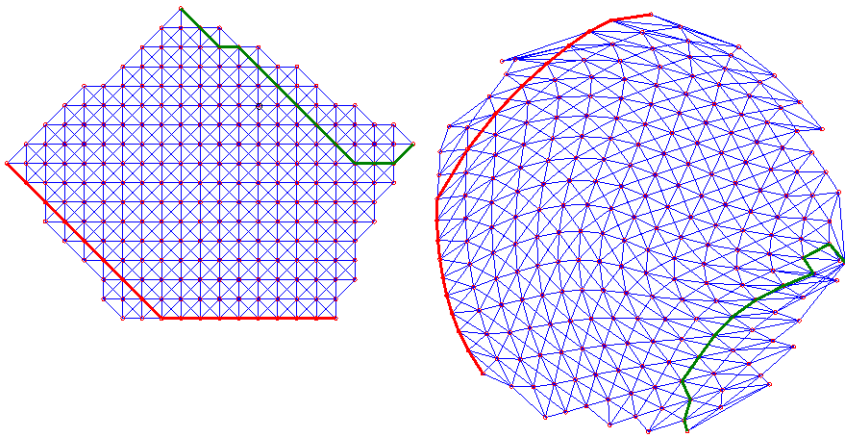
inside the polygon which is mapped to the origin inside the disk is chosen to be the highest loaded point as long as it is not very close to the boundary. If it is close to the boundary then we take a point more in the interior and choose that as the center.

Once the given domain is mapped to a unit disk, we can apply any unit disk load balancing routing algorithm on it. Fig. 9(c) presents the load balancing performance on a cross shape domain. With Lambert's mapping, the maximum load of the domain is reduced by 18%; while Curveball and optimum approximation achieve even better, up to 30%. This clearly shows that area-preserving mapping to a unit disk drastically reducec the load. Note that for "fat" and convex domains, the reduction in load was even more prominent, which is to be expected. We achieve similar figures for many complicated shapes (like the star); for simplicity we just presented a generic case of the cross shaped domain.

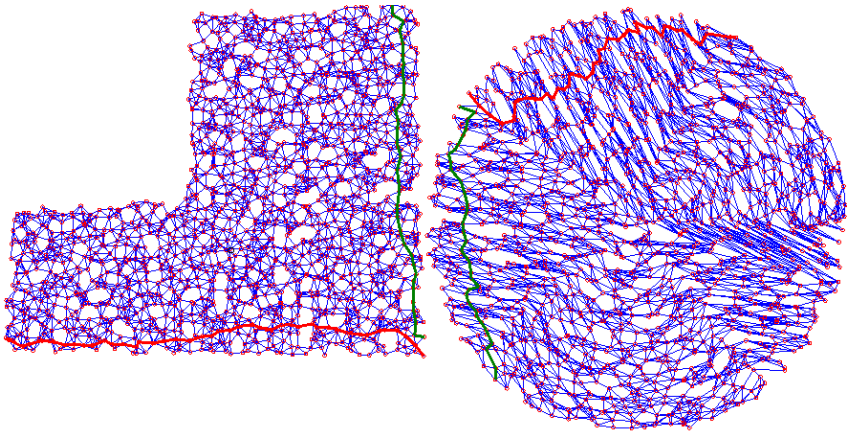
We also test the routing stretch in all domains. The maximum routing stretch factor was 1.96, which we believe is reasonable, considering the reduction in load.

3.5 Discussion

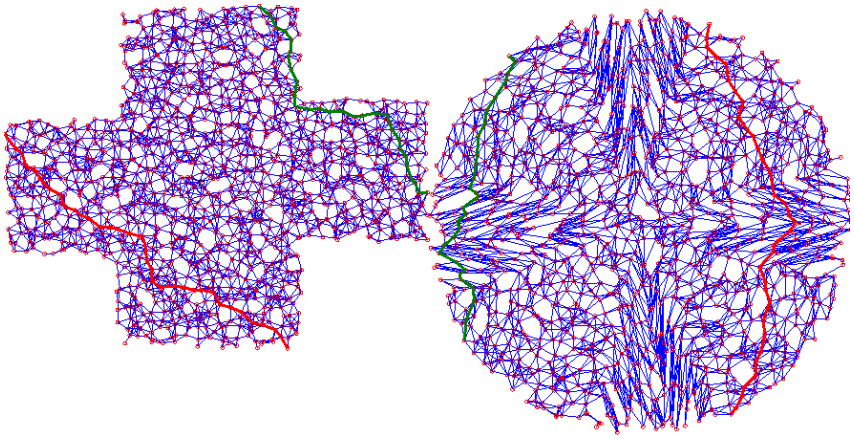
A clear open problem is to consider non-simple domains, i.e., domains with holes. Previous work mainly focused on how to alleviate the heavy traffic along hole boundaries. We remark that bounded load balancing would become much more challenging as one has to also consider paths of different homotopy types (getting around holes in different ways). In particular, an extreme case of the problem on a multi-connected domain could be load balanced routing on a planar graph, which is known to be NP-hard. New ideas are needed to solve that case and this remains one of the most interesting future directions.



(a) A pentagon with grid nodes and area-preserving map to disk using contour,s by conformal mapping method (238 nodes, avg degree 7.24)



(b) L-shape domain with 1441 perturbed grid nodes, avg. degree 7.43



(c) X-shaped domain with 1302 perturbed grid nodes, avg. degree 7.38

Figure 9: Three different domains and their area preserving maps. The red and blue line indicate a sample route in respective domains.

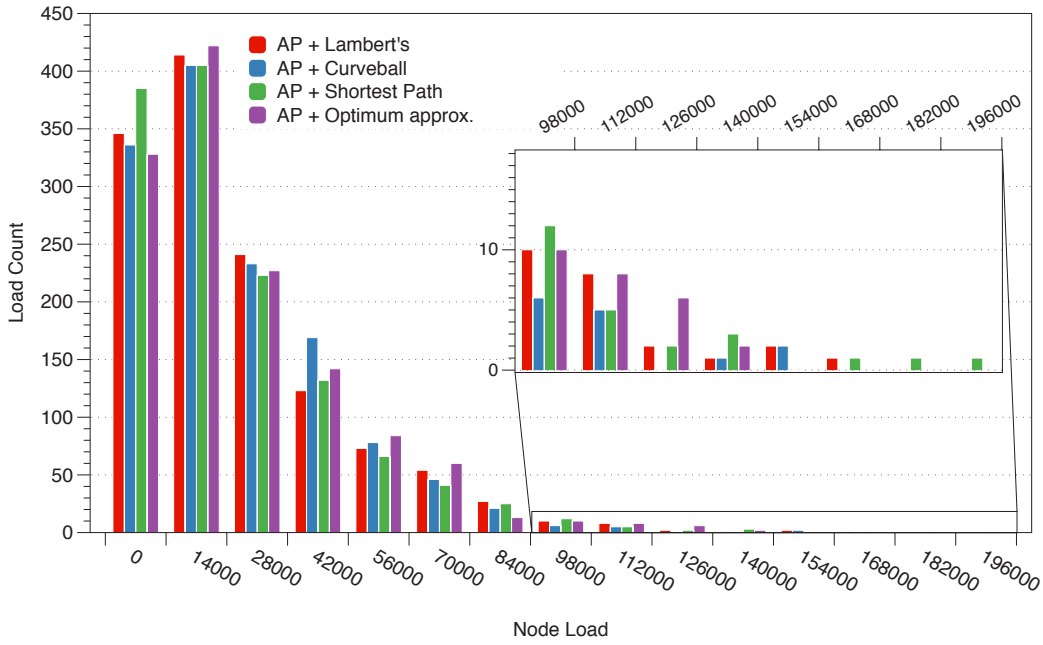


Figure 10: The histogram comparison of load distribution over the cross-shaped domain in fig. 9(c). We first map the cross shape domain to the unit disk, then compare load balancing algorithm on the disk. Notice that all unit disk load balancing routing algorithms now perform better than shortest path routing over cross shape domain.

Chapter 4

Load Balancing using Möbius Transformation

4.1 Introduction

There are two fundamental questions regarding load balancing and routing: (1) From the theoretical perspective, how does the network shape relate to the distribution of shortest paths? (2) From the practical perspective, how to deform a network shape (i.e., designing a new network metric, specifying new weights on the edges) such that some *greedy routing* scheme is able to produce good load balancing? In particular, we look for the embedding of the network in some space with each node given a virtual coordinate such that by some distance function one can route the message to the neighbor closest to the destination, delivery is guaranteed and the routes are load balanced.

Let us first consider a general surface in 3D. Intuitively the points with negative curvatures (the saddle points) attract geodesic (shortest) paths, while points with positive curvatures (peaks and valleys) repel geodesic paths. Thus surface curvature is intrinsically related to geodesic path densities and the load balancing issue for shortest path routing. Unfortunately, to our knowledge, there is little understanding of this relationship. Limited prior work will be reviewed in a later section. Alternatively, we ask whether we can change the surface metric (i.e., deform the surface) such that all points have positive curvature and thus receive uniform traffic

load. A surface with uniform positive curvature everywhere is a sphere. There are tools to deform a given surface to a sphere by using conformal geometry.

4.1.1 Spherical Embedding

For a surface with positive constant curvature everywhere, i.e., a sphere, the shortest paths have uniform distribution and greedy routing (in terms of spherical distance) on the sphere has guaranteed delivery and perfect load balancing. Motivated by this, we would investigate the mapping of a sensor network to a spherical metric. The well-known Koebe-Andreev-Thurston Theorem describes the *spherical embedding* of a 3-connected planar graph[1]: for any such graph G, there *exists* a pair of circle packing on a unit sphere, whose intersection graphs are isomorphic to G and the planar dual of G respectively. This spherical embedding realizing the 3-connected planar graph can be seen as a convex polyhedron with all edges tangent to the unit sphere. Each vertex is associated with a circle on the sphere. Adjacent vertices have their circles tangent to each other. See Figure 15 for an example. With the spherical embedding, it is not hard to see that with the 3D coordinates of the vertices, a greedy method has guaranteed delivery for all pairs of vertices, as mentioned in [118]. This greedy method uses the distance function $d(u, v) = -c(u) \cdot c(v)$, where $c(u)$ is the coordinate of u. This routing scheme is referred to as *polyhedron routing*.

The circle packing can be computed with a number of approaches. The Thurston algorithm sets the boundary radii and iteratively updates the internal radii to satisfy local conditions [34, 149], when G is a triangulation. The algorithm is shown to converge at the limit and in practice is stopped when the error bound is below a given threshold. Springborn and Bobenko [15] proposed a general framework to compute circle patterns for a general 3-connected planar graph, but the algorithm requires a global optimization of certain energy function and is hard to implement in a distributed network. Here we use a different technique by using discrete Ricci flow. We intersect the original planar graph and its dual graph to obtain a planar triangulation and run the Ricci flow algorithm to compute a pair of circle packings, for the original graph and its dual simultaneously. Ricci flow, introduced

[1] A graph is 3-connected if it remains connected after the removal of any 2 nodes.

by Richard Hamilton for Riemannian manifolds [64], modifies the surface metric, in proportional to Gaussian curvatures, such that the curvature evolves in the same manner as heat diffusion. It is a powerful tool for finding a Riemannian metric satisfying the prescribed Gaussian curvature. For discrete triangulated graph, discrete Ricci flow is formulated. Chow and Luo [31] proved a general existence and convergence theorem. Ricci flow is naturally a distributed algorithm in which each node evaluate its local curvature and modify the edge weights accordingly. In this case, we apply Ricci flow to find the spherical embedding of a given graph.

The spherical embedding is not unique and differs from one another by a Möbius transformation. Thus we further investigate the load balancing property of this spherical embedding by choosing a proper Möbius transformation. One heuristic is to makes the circles at the vertices to have a similar size. By simulations we show that this idea works well in practice and compares favorably with previous load balanced routing schemes. We must remark that given a discrete network and uniform traffic, perfect load balancing (i.e., all nodes having the same load) may not be possible. Considering the case when two large sensor fields are connected by a narrow corridor, the nodes on a narrow corridor necessarily carries heavy traffic regardless of routing schemes. Finding the most load balanced routing scheme is an NP-hard problem (modeled as unsplittable flow problem) and the best approximation algorithm has an approximation factor of $O(\log n/\operatorname{loglog} n)$ [126, 127]. These approximation algorithms are centralized and are not practical for large scale low resource networks. For this reason we evaluated our scheme with prior heuristic algorithms using greedy approaches. It remains as our future work to show how our algorithm performs theoretically, compared with the optimal load balanced routing (minimizing the max load).

Last, our scheme falls within the framework of building a virtual coordinate system for greedy routing in a network [129]. For all the virtual coordinates schemes, one needs to have a location service such that one can inquire the virtual coordinate of any other node in the network. Efficient location services for sensor networks have been developed [98, 130]. Such location services can be used in routing with virtual coordinates developed. The virtual coordinate addresses in this case, as in [129], are simple Euclidean coordinates of the form (x, y, z). The location service keeps the mapping from a node's ID and its virtual coordinates.

In the following, we first review related work on the topic of greedy routing and load balanced routing respectively. Then we introduce the theory of spherical representation and polyhedron routing. Simulation results are presented on the performance evaluation.

4.2 Spherical Representation

4.2.1 Spherical Representation Algorithm

According to Koebe-Andreev-Thurston Theorem, spherical embedding of 3-connected planar graph guarantees delivery on greedy routing. Here we present the algorithm to compute a spherical embedding. This algorithm requires a 3-connected planar graph, which is obtained in step 0.

Step 0: Extract a Planar Subgraph. Algorithms for such purpose have been developed in the past literature and are briefly reviewed below. We only require a combinatorial planar graph and does not require a planar embedding.

In [137], a local, distributed algorithm has been developed to obtain a triangulation from the connectivity graph. The idea is to compute the *restricted Delaunay graph (RDG)* [59], i.e., a planar graph containing all Delaunay edges of length no greater than 1. The RDG can be computed by the nodes locally when the communication graph follows a quasi-unit disk graph (q-UDG)[2] of parameter $\alpha \leq \sqrt{2}$. The requirement of a quasi-UDG is to ensure that crossing edges can be detected locally and handled properly. Funke *et al.* [57] developed a location-free triangulation algorithm by using landmarks and combinatorial Delaunay graph. The idea is to select a set of nodes as landmarks. The landmarks flood with a restricted range such that every node identifies the closest landmark a and is grouped to the Voronoi cell of a. A planar graph is computed on the landmarks by connecting the landmarks a, b that have 2-hop wide 'channel' of only nodes within cells of a, b. The authors showed that this graph is planar when the communication model follows a

[2]In a *quasi unit disk graph* with parameter $\alpha \geq 1$, if two nodes are within distance $1/\alpha$, an edge between the two exists, if they are at a distance more than 1, the edge does not exist; while for other distances, the existence of the edge is uncertain.

quasi-UDG of parameter $\alpha \leq \sqrt{2}$. The two algorithms above both require a quasi-UDG model and thus does not work when a sensor network does not follow the quasi-UDG assumption. The following algorithms compute planar graphs without such assumptions.

Kim *et al.* [63, 83] addressed the problem that planarization techniques using relative neighborhood graph or Gabriel graph fail when the communication model does not comply to the unit disk graph assumption. They developed a cross link detection protocol to probe each link, detect and remove possible crossings with other links. The resulted graph is a combinatorial planar graph. Zhang *et al.* [164] developed a location-free algorithm to extract a planar subgraph from the connectivity graph. The main idea is to planarize adjacent layers of a shortest path tree. Again this method does not require a unit disk graph model or quasi-UDG model.

All these algorithms above can be used in our framework. In our implementation, we have used the restricted Delaunay graph approach [137]. In the following we deal with an extracted planar graph and denote it as G.

Step 1: Compute the Dual & Overlap Graph. A planar 3-connected graph G is shown in Figure 11 (a), the k-th vertex is labeled as k, the j-th face is denoted as f_j. Note that, face f_8 represents the infinite face. The overlap graph $D := G \cup \tilde{G}$, where $\tilde{G}$ is the dual graph of G, is shown in Figure 11 (b). On the overlap graph D, there are three types of nodes:

1. Vertex node v_i: corresponding to a vertex in G, is represented as a red dot.
2. Face node f_j: corresponding to a face in G, is represented as a green dot.
3. Edge node e_{ij}^{kl}: corresponding to an intersection of an edge $[v_i, v_j]$ in G, and an edge $[f_k, f_l]$ in the dual graph $\tilde{G}$, where $[v_i, v_j]$ is the common edge of the faces f_k and f_l, represented as a blue square.

Each facet on the overlapped graph D is a topological quadrilateral, with two edge nodes, one vertex node v_i and one face node f_j, we denote the quadrilateral as $\square(v_i, f_j)$.

By adding an edge connecting the vertex node and the face node on each quadrilateral, we get a planar triangulated graph T.

Step 2: Select an Infinity Edge Node. Select one edge node to be mapped to the infinity point. We call it the *infinity edge node* and denote it as e_∞. The choice of

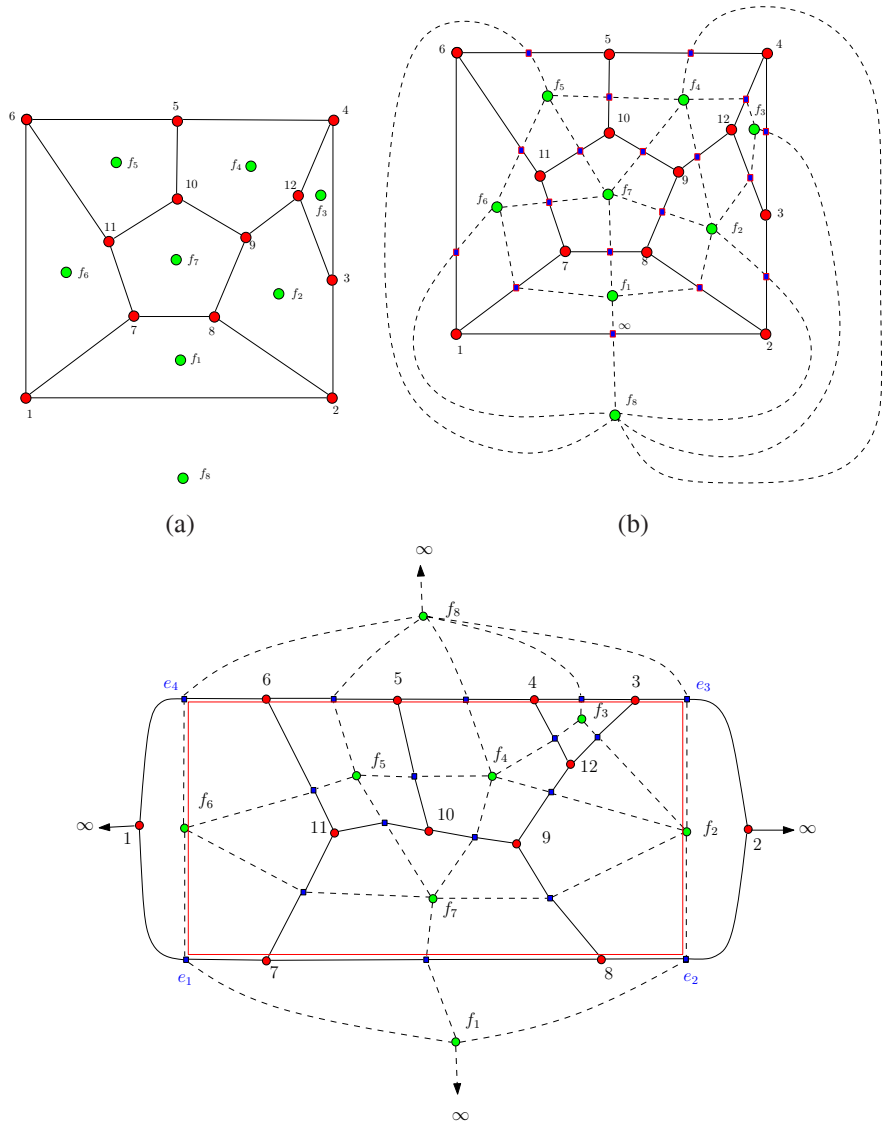
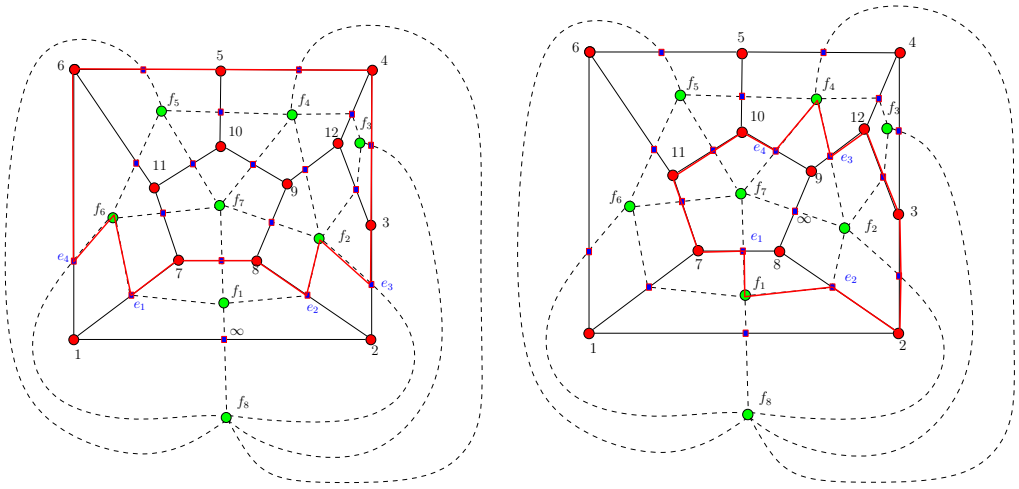


Figure 11: Compute the reduced graph. (a) A 3-connected planar graph G as the input graph. (b) The overlap graph $D = G \cup \tilde{G}$. (c) The reduced graph.



(a) $e_\infty = [v_1, v_2] \cap [f_1, f_8]$ (b) $e_\infty = [v_8, v_9] \cap [f_7, f_2]$

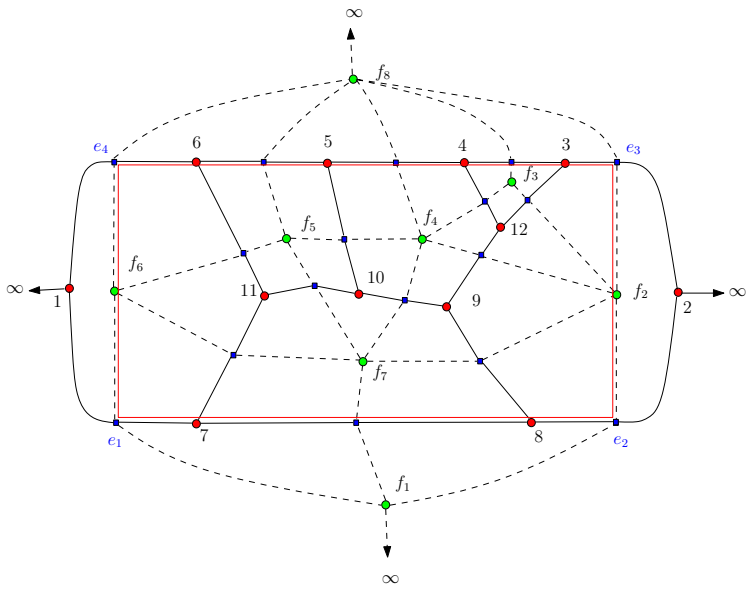
Figure 12: Step 2. Select the infinity edge node e_∞.

the edge could be arbitrary. Figure 11 (b) shows an example when the edge node $e_\infty = [v_1, v_2] \cap [f_1, f_8]$ is selected as the infinity edge node.

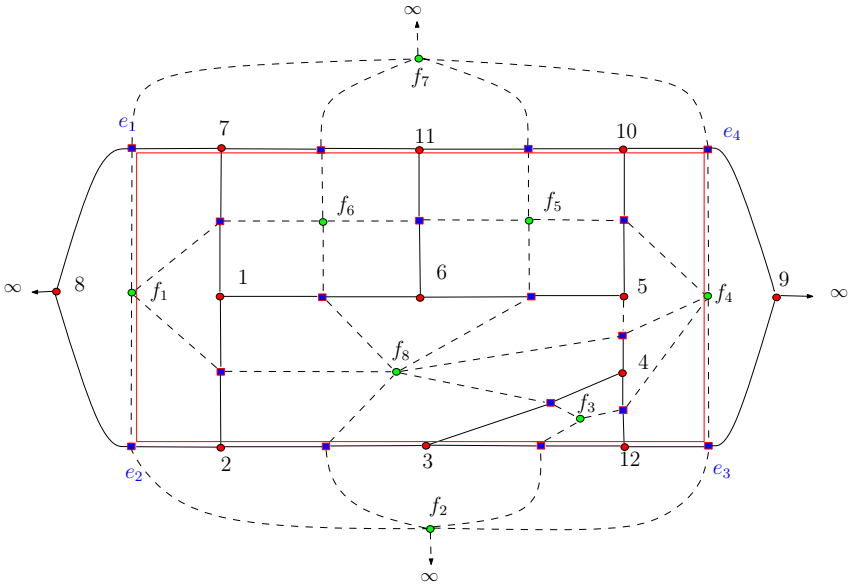
Step 3: Compute the Reduced Graph $\bar{G}$. Suppose the infinity edge node is given by $e_\infty = [v_i, v_j] \cap [f_k, f_l]$, then we remove all the quadrilateral facets adjacent to v_i, v_j or f_k, f_l in the overlap graph D, to get the reduced graph $\bar{G}$. Figure 11 (c) shows the reduced graph for Figure 11 (b). The boundary of the reduced graphs are labeled as the red edges

Figure 13 frame (a) and (b) show the reduced graphs, which are deduced from Figure 12 frame (a) and (b) respectively. The boundary of the reduced graphs are labeled as the red edges in Figure 12.

Step 4: Ricci Flow. On the reduced graph $\bar{G}$, each quadrilateral facet $\square(v_i, f_j)$ is triangulated by adding one virtual edge $[v_i, f_j]$ connecting the vertex node v_i and the face node f_j. Then we run Ricci flow on the triangulated reduced graph in the following way. Each triangle on the triangulated reduced graph has one vertex node v_i, one face node f_j and one edge node e_k, $[v_i, f_j, e_k]$, as shown in Figure 14. In particular, For each vertex node v_i, we associate it with a circle $C(v_i, \gamma_i)$, centered at v_i with radius γ_i; For each face node f_j, we associate it with a circle $C(f_j, \gamma_j)$; For each edge node e_k, we associate it with a circle with zero radius all the time,



(a) reduced graph of Fig 12 (a)



(b) reduced graph of Fig 12 (b)

Figure 13: Step 3. Compute the reduced graph $\bar{G}$

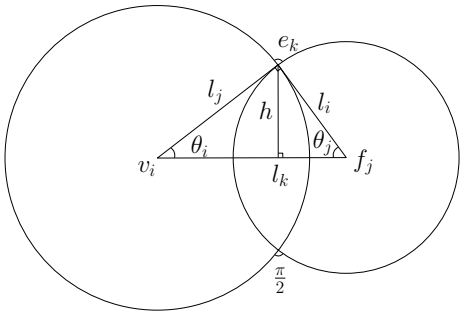


Figure 14: Step 4. Ricci flow

$C(e_k, 0)$; The vertex node circle and the face node circle intersect at a right angle; The intersection angle between the edge node circle with other circles are zeros.

Therefore, from Figure 14,

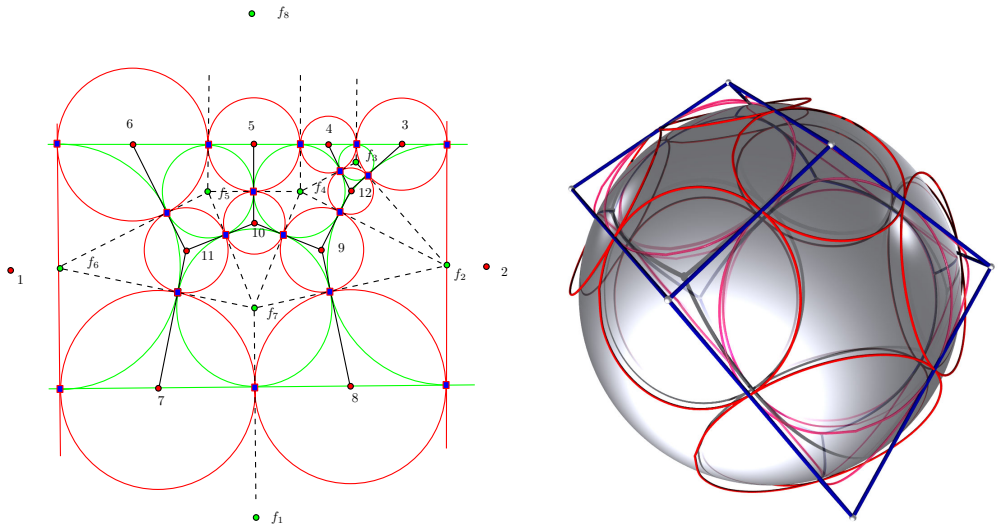
$$l_i = \gamma_j, l_j = \gamma_i, l_k = \sqrt{\gamma_i^2 + \gamma_j^2}, \theta_k = \frac{\pi}{2}.$$

For both vertex nodes and face nodes with circle radius γ, let $u = \log \gamma$, then the Ricci flow is given by the following differential equation:

$$\frac{du_i}{dt} = -K_i, \tag{30}$$

where K_i is the *discrete Gaussian curvature* at a vertex v_i, defined as the angle deficit at v_i, i.e., the difference of 2π and the sum of the corner angles of all triangles adjacent to v_i. Specifically, the Ricci flow algorithm is carried out in a distributed manner. Each vertex v_i modifies its radius γ_i with respect to the current curvature. In an iterative way, the radii are adjusted until the curvature becomes sufficiently close to 0. Recall that we have some virtual nodes such as the face nodes and edge nodes, these are actually represented by some nearby vertex nodes when Ricci flow computation is carried out. To run the algorithm, each sensor node only requires the local information about nodes in adjacent faces.

We then flatten the reduced graph triangle by triangle and compute the circle packing embedded in the plane, as shown in Figure 15. Suppose the infinity edge node is $e_\infty = [v_i, v_j] \cap [f_k, f_l]$, then the vertex node circles of v_i and v_j are mapped to vertical lines, the face node circles of f_k and f_l are mapped to horizontal lines. These four circles intersect at the infinity point.



(i) (ii)

Figure 15: (i) The circle packing embedded in the plane. (ii)The spherical presentation of the graph and convex polytope realization in 3D.

Step 5. Stereo-graphic Projection. Finally, we use stereo-graph projection $\phi : (u, v) \to (x, y, z)$ to map the circle packing on the plane to the sphere,

$$\phi(u, v) = \left(\frac{2u}{1 + u^2 + v^2}, \frac{2v}{1 + u^2 + v^2}, \frac{-1 + u^2 + v^2}{1 + u^2 + v^2} \right).$$

For each face node circle $C(f_i, \gamma_i)$ on the plane, we compute its image of the stereo-graphic projection $\phi(C(f_i, \gamma_i))$, which is a circle in $\mathbb{R}^3$. We can compute the planes through all the spherical face node circles. The convex hull bounded by these planes is the convex polytope P. In more details, if a vertex v_i in the original graph G is adjacent to faces f_k's, then all the planes of spherical face node circles $C(f_k, \gamma_k)$ intersect at one point in $\mathbb{R}^3$. If two faces f_i and f_j sharing an edge e, then the intersection line between the plane of $C(f_i, \gamma_i)$ and $C(f_j, \gamma_j)$ is tangent to the spherical image of e. Figure 15 shows the spherical representation of the graph and the induced convex polytope. The convex polytope will be used for our routing purpose.

Communication Costs. Last we remark that the majority of computation and communication costs is for the Ricci flow algorithm, as all other steps can be handled with only constant cost per node. The curvature error decreases exponentially fast. Therefore, the number of steps to reach the desired curvature error bound ε is given

by $O(-\frac{\log \varepsilon}{\delta})$, where δ is the step size in the Ricci flow algorithm. The total communication cost is thus $O(-\frac{n \log \varepsilon}{\delta})$. In the simulation section we evaluated the cost on different networks.

4.3 Load Balancing

The spherical representation of the 3-connected graph is not unique, two such embeddings differ by a Möbius transformation. All these spherical representations guarantee successful delivery with polyhedron routing. But they give different load balancing properties. Intuitively, when the vertices are more uniformly spread on the sphere, i.e., approximating the spherical metric better, polyhedron routing has better load balancing property. In the following we look for a Möbius transformation that uniformizes the vertex distribution.

For a discrete finite network, the network outer boundary is a large face. We map it to the equator of a unit sphere, such that all the vertices are then mapped to a hemisphere. For that, we apply a circular reflection, as defined below. The *circular reflection* is a special case of Möbius transformation. It maps the points inside a circle C with center $\mathbf{c}$ and radius r to the points outside, and vice versa: $\tau(z) = \mathbf{c} + \frac{\gamma^2}{\bar{z} - \bar{\mathbf{c}}}$.

We first translate and scale the exterior face node circle $C(f_\infty, \gamma_\infty)$ to the unit circle. Then all the other face circles are mapped to the exterior of the unit disk. Then we use circular reflection to map the exterior of the unit circle to the interior. Then we use the Möbius transformation to map the interior of the unit disk to itself. All such kind of maps are parameterized by an interior point of the unit disk z_0,

$$\eta_{\theta, z_0}(z) = e^{i\theta} \frac{z - z_0}{1 - \bar{z}_0 z}.$$

The choice of η is to make the spherical vertex node circle radii as uniform as possible.

Because the graph is 3-connected, each vertex circle C_v goes through at least 3 edge nodes, denoted as $\{e_1, e_2, e_3\}$. The stereo-graphic projection $\phi : \mathbb{C} \to \mathbb{S}^2$ maps them to points on the unit sphere $\{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}$, $\mathbf{s}_k = \phi(e_k)$. These three points determine a plane in $\mathbb{R}^3$, the normal is given by

$$\mathbf{n} = \frac{(\mathbf{s}_1 - \mathbf{s}_3) \times (\mathbf{s}_1 - \mathbf{s}_2)}{|(\mathbf{s}_1 - \mathbf{s}_3) \times (\mathbf{s}_1 - \mathbf{s}_2)|},$$

the plane is given by

$$\pi_v : \langle \mathbf{p} - \mathbf{s}_1, \mathbf{n} \rangle = 0.$$

The distance from the origin to the plane π_v is given by

$$d_v = \langle \mathbf{s}_1, \mathbf{n} \rangle.$$

To find the Möbius transformation such that all the vertex radii are as uniform as possible, we minimize the energy

$$E(\theta, z_0) = \sum_{v \in G} (d_v - \bar{d})^2,$$

where $\bar{d} = \frac{\sum_v dv}{m}$, m is the total number of vertices in G.

The energy $E(\theta, z_0)$ is invariant with respect to angle parameter θ. It is highly non-linear, and has multiple local optima. Direct gradient descend method won't guarantee to reach the global optimum. Therefore, we randomly choose initial seeds, then further use gradient descend method to reach the optimum in the neighborhood of the seed. The seed z_0 is uniformly chosen among the positions of vertex coordinates v_i. The gradient of $E(z_0)$ with respect to the real and imaginary parts of z_0 can be deduced in closed analytic form. This optimization step is a centralized algorithm. Once the desired Möbius transformation is found, it is delivered to all nodes in the network, which then apply the transformation to derive their respective coordinates in 3D.

Figure 16 shows the spherical representation before and after the Möbius optimization.

4.4 Simulations

We conducted extensive simulation tests to evaluate routing properties of our polyhedron routing. It was compared with shortest path routing, and also the previously published methods of Curveball Routing [124] and Outer Space Routing [109]. Finding shortest paths is expensive and requires large routing tables, so is not exactly suitable for distributed efficient routing, but it forms a standard baseline.

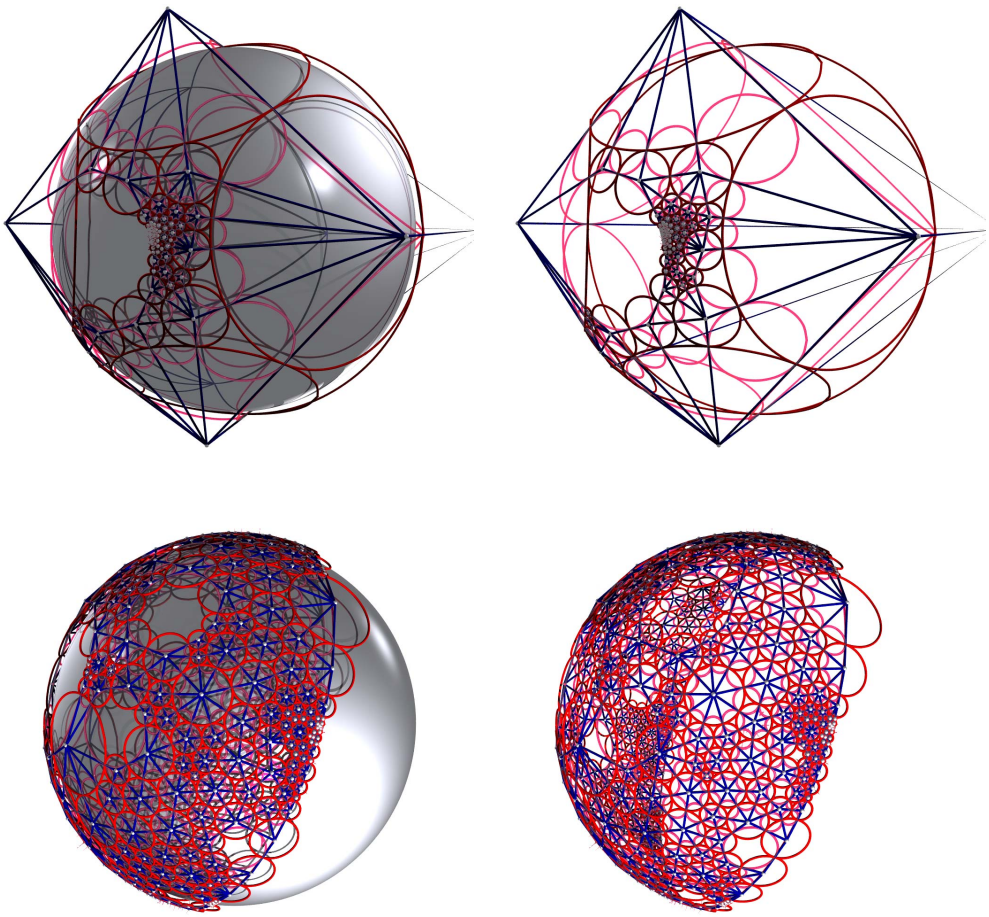
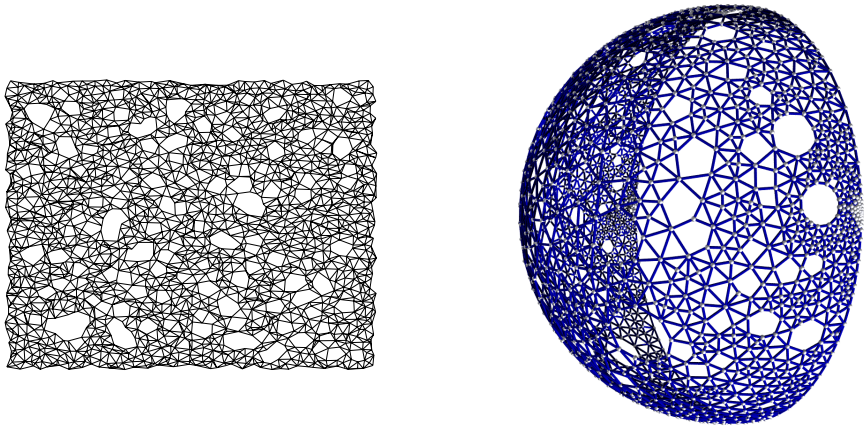


Figure 16: The above two are before Möbius optimization and the below two are after Möbius optimization.



(a) (b)

Figure 17: Sample Embeddings. (a) Original embedding after planarization. (b) Embedding on a hemisphere.

4.4.1 Curveball and Outer Space Routing

In Curveball routing [124], the entire network is mapped to a sphere using stereographic projection. A sphere S is assumed with its center coinciding with the center of the network. Then each node p in the network is projected to the point on S where the straight line joining p with the north pole of the sphere intersects S. The network is assumed to be of the shape of a disk of some radius R, and the authors suggested using a sphere of radius $R/1.2$.

Routing is done as greedy routing based on the geodesic distance between the virtual coordinates on the sphere. The idea behind this method is to avoid the *crowded center* effect. In a usual disk-shaped network, the center receives more load. The greedy routes on the sphere correspond to paths that tend to move away from the center, thereby avoiding the congestion at the network center.

In Outer space routing [109], the network is assumed to be in the shape of a square in the first quadrant. The size of the network is then quadrupled by successive reflections in the X and Y axes. Each node is assigned a virtual coordinate with equal probabilities in the image of the network in one of the 4 quadrants. Note that being created by reflections, the right and left borders of the quadrupled network correspond to the same regions of the original network. Similarly, the top and

bottom borders are identified. The quadrupled network therefore has the topology of a torus.

When routing to any node, the route actually travels to the virtual coordinate selected for the destination. This is achieved by greedy routing in the metric of the torus. Observe that this method spreads the nodes in a region that is 4 times the area. Therefore, it reduces the density of the network to about $1/4$. For networks that are not very dense to start with, this leads to even sparser networks, increasing the chance of a routing attempt failing.

Neither of these methods guarantee delivery. As we see in the simulations below, presence of large holes cause these methods to fail easily. Sparse networks naturally tend to have holes, and therefore cause more frequent failures.

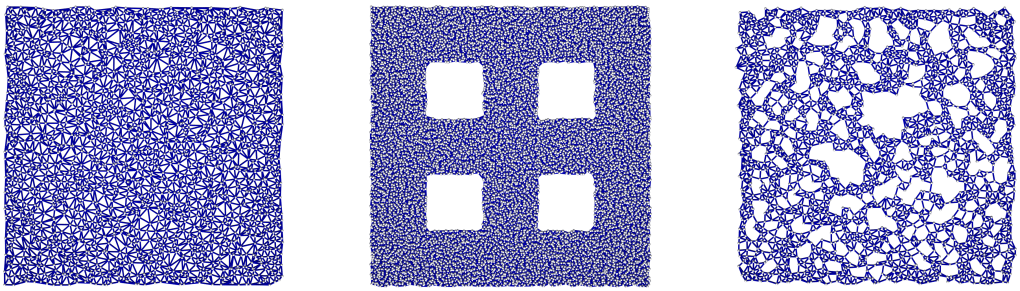
4.4.2 Simulation Results

We varied the density of the random networks and also considered networks with artificially created large holes, such as those representing physical features and obstacles in the domain. We randomly select source and destination pairs and for each experiment we selected 20000 such random routing requests. The main observations of the experiments were the following.

- The polyhedron routing is the only one that has 100% delivery guarantee (other than shortest path routing). Other methods in consideration perform progressively worse as the density decreases.
- Other methods have poor delivery guarantees on networks with large holes.
- The load balancing properties of the methods are comparable.

The details follow. We first look at routing characteristics, followed by the load balancing properties. Finally we present results regarding the computational cost of the embedding algorithm.

Delivery and stretch. To evaluate the routing qualities of our algorithm, we tried it out on networks of various densities and types (as shown in Figure 18). Here we show the extreme cases of a random but dense network, a dense network with large holes and a random sparse network . We compared the delivery guarantees and stretch of different schemes, the results are shown in Tables 1, 2 and 3. Our



(a) Dense Network. (b) Dense network with large holes. (c) Sparse network.

Figure 18: Experimental networks. (a) Dense Network. 1850 nodes, avg. degree 14.88 (b) Dense network with large holes. 2100 nodes, avg degree 12.14 (c) Sparse network. 1774 nodes, avg. degree 3.32.

	Polyhedron	Short path	Curveball	Outer Space
Delivery%	100	100	99.73	70.88
Stretch	1.060	1	1.051	1.216

Table 1: Comparative performance for dense networks (Fig 18(a))

method also has consistently small routing stretch. We also found that the delivery rate of curveball routing is better than that of outer space routing.

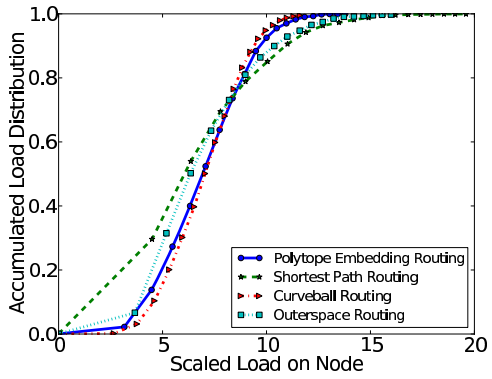
Load balancing. For the same set of routing attempts, we also monitored the traffic load at each node. The cumulative distributions of loads for the networks in Figure 18 are shown in Figure 19. All three schemes, polyhedron routing, the curveball routing and the outer space routing, have similar load balancing properties, which are significantly better than that of shortest path routing. We also remark that for the curveball routing and the outer space routing, since they do not deliver all packets, the load distribution is more favorable for them.

	Polyhedron	Short path	Curveball	Outer Space
Delivery%	100	100	95.55	36.41
Stretch	1.092	1	1.076	1.219

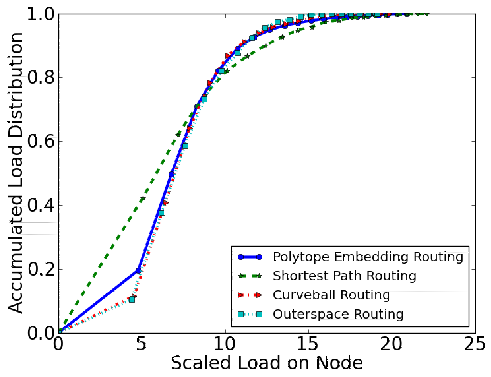
Table 2: Comparative performance for dense network with holes (Figure 18(b))

	Polyhedron	Short path	Curveball	Outer Space
Delivery%	100	100	64.05	1.27
Stretch	1.109	1	1.090	1.071

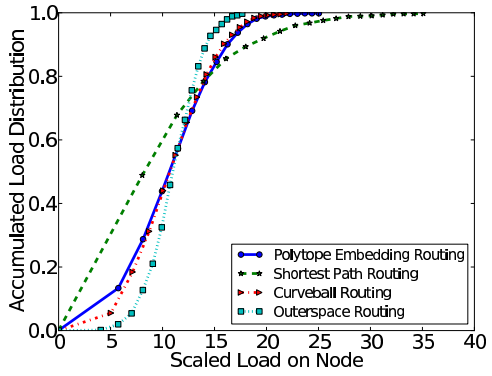
Table 3: Comparative performance for sparse networks (Fig 18(c))



(a)



(b)



(c)

Figure 19: Cumulative distribution of load, for networks respectively corresponding to those in Figure 18. (a) Dense Network. (b) Dense network with large holes. (c) Sparse network.

Chapter 5

Exploring Path Space in Greedy Routing

5.1 Introduction

Finding multiple routings in a network is a challenging problem for a large scale network, in particular if a node does not have the entire topology. On the theoretical side, one can run a flow algorithm to find a maximum number of node disjoint paths between source and destination. But the flow algorithm requires centralized knowledge and also has a high computational cost of $O(n^3)$ if the network size is n. Even if one can afford to pre-compute multiple routing paths, storing these paths at the sensor nodes will be, storage-wise, too overwhelming, which makes the centralized algorithms scale poorly with the size of the network. In literatures on mobile ad hoc networks, there have been a number of heuristic proposals to find multiple paths but the theoretical understanding of these schemes rarely exists. If one wants to use multiple paths to recover from en-route node or link failure, there is not much understanding (globally) on where the second path is going to be.

5.1.1 Our Approach

We approach the problem from a geometric angle. Since nodes are typically densely deployed in a geometric domain, the network topology is not like a general graph. We explore different embeddings of the network such that by controlling a

constant number of parameters we can quickly switch between different network embeddings such that greedy routing in such embeddings generates different routing paths. Thus one can easily come up with multiple node disjoint paths, or even switch to an alternative path in the middle of a routing process, by spontaneously changing to a different embedding.

We are motivated by our recent development of conformal mapping of a sensor network [137, 138]. We first compute an embedding of a sensor network such that all the holes are deformed to be circular. We name this embedding to be a *circular domain*. On a circular domain, greedy routing that always delivers the message to the node closer to the destination using the new coordinates guarantees message delivery. However, the embedding as a circular domain is not unique, and all such embeddings differ by Möbius transformations, which maps a complex plane to itself and can be represented by

$$f(z) = \frac{az + b}{cz + d},$$

where z is a complex variable and a, b, c, d are four complex numbers satisfying $ad - bc = 1$. A Möbius transformation always maps circles to circles. Thus applying a Möbius transformation on a circular domain essentially 're-arranges' the positions and the sizes of the circular holes and the new embedding remains to be a circular domain. Therefore there are actually *infinitely* many circular domains on each of which greedy routing guarantees delivery. The previous work as in [137, 138] only considered one such circular domain by fixing one hole to be at the center of the network. Here we investigate all possible circular domain embeddings and the applications to multipath routing in a sensor network.

The main difficulty for efficient and scalable routing in a sensor network is due to lack of the global knowledge. Embedding the network as a circular domain makes that difficulty go away in some sense. Our routing scheme avoids the requirement for the global network topology, while the geometric information we need — the locations and shapes of the holes and the boundary – are typically of a constant size and usually remain stable. With a circular domain one can predict where the path is (subject to the assumption that the sensor nodes are sufficient dense so that the continuous path is a good approximation of the discrete path by greedy routing) and by applying a Möbius transformation we know what a path we

will get and how different it is from the previous one. Since a Möbius transformation only uses four parameters, we can attach the current Möbius transformation at the packet such that by applying the Möbius transformation a node can compute its coordinates under the transformation on the spot to generate the greedy path under the new embedding. In case of a link failure on the current greedy path, a node can generate a new Möbius transformation and switch to a different path immediately. The new Möbius transformation is simply attached to the packet.

Using a circular domain representation gives us the following advantages that will be proved in the continuous case and evaluated by simulations for the discrete network setting:

- By using different Möbius transformations one generates multiple paths to the destination that are disjoint except at the source and the destination. We present algorithms for networks with or without holes.
- In case of node failures, we present an algorithm that identifies a different path to the destination. The second path takes a big circular arc type of detour that is likely to jump over correlated failure regions.

In the following we first quickly review prior work on multipath routing. We present the theoretical proofs of our method and present simulation results afterwards.

5.2 Related Work

In this section we quickly review prior work on three relevant topics: multipath routing both in theory and in practice; some of them focus on how to recover from node or link failures; and previous greedy routing schemes.

5.2.1 Multipath routing

Multipath routing has been investigated extensively in computer networking in order to improve routing robustness [9, 37], achieve better load balancing [41, 148, 162], reduce network congestion, reduce end-to-end delay [165] and increase network throughput [70, 155]. Between a pair of source and destination,

multipath routing looks for multiple paths that are sufficiently different from each other such that node or link failures will not destroy all of them. One formulation is to look for k node disjoint or edge disjoint paths, which can be computed by flow algorithm [35]. But this is a centralized algorithm and would require the knowledge of the entire network [68]. Distributed algorithms only exist for special case of $k = 2$. In [128] two colored trees were constructed for routing such that the paths in the two trees are link or node disjoint. Relaxation of the node/edge disjointness of the multiple paths leads to the approach of braided multipath [58] in which the multiple paths are only partially disjoint.

In a mobile ad hoc network, multipath routing has also been developed to enhance the performance of on-demand routing protocols such as AODV [3, 28, 92, 106, 132] or DSR [102, 112, 143] as the network topology undergoes constant changes. Prior work in this direction uses extensive message exchange or flooding to discover alternative paths to bypass a broken link. A major problem of these schemes is that they suffer from high recovery delay from node or link failures, which severely affects the performance of end-to-end QoS measurements in the transport or application layer.

5.2.2 Fast recovery from failures

Recently there has been a number of interesting work that studies the problem of fast recovery from link or node failures, even for a centralized situation. When a link or node fails, the goal is to quickly discover an alternative path with nearly no delay, such that the current traffic is not interrupted. For the intra-domain routing protocols on the Internet, the recent IP fast re-routing (IPFRR) schemes (Loop-free alternate (LFA) [7], O2 [133, 134, 142], DIV-R [131], MARA [117] and protection routing [91]) aim to ensure fast re-covergence when node failures are detected. In general this family of work would like to find an alternative next hop when the intended next hop is not reachable. Depending on the detailed implementations, the design often suffers from one or more of the following problems: having possible transient loops, the requirement for a lower bound on node degree, computational intractability (e.g., verifying whether a graph has a protection routing or not turns out to be NP-hard [91]).

Our work is motivated by routing with multiple metrics as introduced in the path splicing idea [111], which is proposed for increasing routing reliability on the Internet. Given a weighted graph, one perturbs the weights of the edges and computes a shortest path tree on each node. These multiple shortest path trees are used in combination to generate a routing path in case of in-transit link failures. Traffic in the network can freely switch between different shortest path trees, which results in a large number of routing paths (these paths are the braided multi-paths). The overhead of switching between different trees is done by just changing a few bits in the packet header. This supports fast recovery from link or node failure and ensures low end-to-end delay. However, this is mainly for interdomain routing on the Internet. The computation of the multiple shortest path trees is too costly for a large scale sensor network. For sensor network setting we need to have a low cost method to generate multiple metrics with great flexibility and path diversity.

5.3 Algorithms

Our routing algorithm tries to find various embeddings, or mappings from the original sensor network to a circle domain, in which all holes are of a circular shape and greedy routing can guarantee delivery. Such mappings are conformal (angle preserving) and computed by discrete Ricci flow. Then Möbius transformations could help us to find more embeddings, or controllable multiple metrics.

5.3.1 Embedding into Circular Domains with Ricci Flow

5.3.2 Multipath Routing

In this section we describe how to generate multiple paths from a given source node s to a given target node t. We will give different embeddings of the domain in such a way that the route used by greedy routing in one embedding is likely to be different from the one used by greedy routing in any other embedding, and these routes are 'well spaced', a notion we will make precise soon. The algorithm we will present generates paths that are provable to be disjoint in the continuous case. In the discrete setting, we evaluate the performance by simulations.

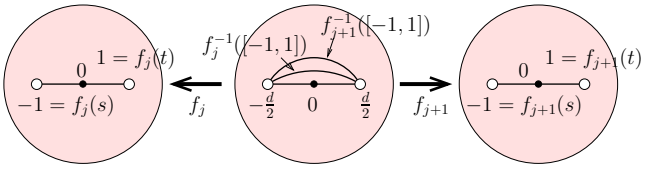


Figure 20: The multiple paths on the domain D (in the middle) are the greedy paths under transformations f_j. The figure shows two transformations f_j and f_{j+1} respectively.

We first present the algorithm for a network without holes. Then we discuss how to find disjoint paths in a network with holes.

5.3.2.1 Network Without Holes

Without loss of generality we can assume that the outer boundary is a circle, and that the coordinates of source s are $(-d/2, 0)$ while those of destination t are $(d/2, 0)$, so that the line segment joining s to t is horizontal and of length d. This can easily be achieved by a rotation and translation. We denote the domain by D.

Consider a continuous domain D we can easily generate many disjoint paths – by essentially applying a different Möbius transformation each time. The greedy path under a Möbius transformation turns out to be a circular arc connecting s and t in the original domain D. See Figure 20. In the discrete case when the domain D is represented by a triangulation, the routing paths are found by using greedy routing in different embeddings, after proper Möbius transformations. We remark that potentially one can design greedy routing to follow any curve, i.e., as in the idea of routing along a curve [113]. But in general routing on a curve does *not* have any guarantee on the delivery. In our case, as we actually perform greedy routing in another circular domain after a proper Möbius transformation, this immediately shows a proof that such a route is guaranteed to reach the destination.

In a continuous domain D, obviously all such circular arcs are disjoint except at source and destination. In the discrete case the greedy paths are guided by the circular arcs but can definitely deviate from them due to discrete node distribution. Since typically sensor networks have upper bounded density, a major constraint on the number of node disjoint paths between s and t is due to the degree at s and t. A

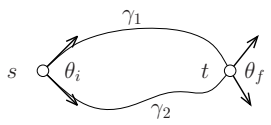


Figure 21: For two curves γ_1 and γ_2 from s to t, the initial directional spread is shown as θ_i and the final directional spread is shown as θ_f.

reasonable heuristic to minimize overlaps of multiple paths is to design paths that are evenly spread out at s and t. We use this heuristic to design our paths in the discrete setting.

Given two curves γ_1 and γ_2 joining s to t, we can define initial and final *directional spread* between γ_1 and γ_2 to be the angle between the tangent vectors of γ_1 and γ_2 at s and t respectively. We denote these by $d_i(\gamma_1, \gamma_2)$ and $d_f(\gamma_1, \gamma_2)$ respectively. See Figure 21 for an example. In the following we use Möbius transformations to generate circular arcs connecting s and t such that their directional spread at source and destination are as evenly spread as possible.

For a given $k \geq 1$, let $\theta = \frac{\pi}{2k}$ and define $\theta_j = \frac{\pi}{2}(1 - \frac{j-1}{k})$ for $1 \leq j \leq 2k+1$. Also, let $\alpha_j = d/2 \tan \theta_j/2$ for $1 \leq j \leq 2k+1$.

We next define a Möbius transformation $f_j(z)$ for $1 \leq j \leq 2k+1$ by

$$f_j(z) = \frac{zd - id\alpha_j}{z(-2i\alpha_j) + d^2/2}.$$

Theorem 17 *f_j has the following properties:*

1. $f_j(s) = -1, \quad \forall 1 \leq j \leq 2k+1.$
2. $f_j(t) = 1, \quad \forall 1 \leq j \leq 2k+1.$
3. *Let $\gamma_j = f_j^{-1}([-1, 1])$. Then γ_j is a curve joining s to t. Moreover, it is the arc of the unique circle passing through s and t, such that the tangent vectors at s and t both make an angle of θ_j with the x-axis.*
4. $d_i(\gamma_j, \gamma_{j+1}) = d_f(\gamma_j, \gamma_{j+1}) = \theta, \quad \forall 1 \leq j \leq 2k+1.$

Proof 18 *(1) and (2) are trivial.*

To prove (3) we use the property that Möbius transformations map circles to circles. Note that the point $(0, \alpha_j)$ is represented by the complex number $z = i\alpha_j$.

One can check that $f_j(i\alpha_j) = 0$. Since $-1, 0$ and 1 lie on a line, it means that s, $(0, \alpha_j)$ and t lie on a circle. Therefore $f_j^{-1}([-1, 1])$ is an arc of the circle passing through these three points. One has to now verify that the tangent to this (unique) circle at s and t makes an angle of θ_j with the horizontal axis.

(4) follows from (3) and the fact that $\theta_{j-1} - \theta_j = \theta$.

What the above calculations mean is that if we define a new embedding of the domain D by mapping a point $z \in D$ to $f_j(z) \in f_j(D)$, then the source maps to -1 and the target maps to 1. In this new embedding, the shortest path from source to target is simply the straight line from -1 to 1. Following this path for some j is equivalent to following the arc of the unique circle passing through s and t whose tangents at s and t make an angle of θ_j with the horizontal axis. Any two such arcs have an initial and final directional spread of at least $\theta = \frac{\pi}{2k}$. Hence we have generated $2k + 1$ node disjoint θ spread paths from s to t. See Figure 20 for an example.

All such paths lie in the circle with the line segment st as its diameter. We can also consider the case $\theta_j = \frac{\pi}{2}(1 + \frac{j-1}{k})$ to get $2k - 2$ more node disjoint paths, with an angle spread of θ, getting $4k - 1$ in total. For example if $k = 3$, we can get a total of 11 paths, such that the directional difference is at least $\frac{\pi}{6}$.

We remark that the above results hold only for source s and target t pairs for which the circle with line segment $\overline{st}$ as diameter, denoted as C_{st}, is contained inside the domain (i.e., in the interior of the outer circle C). When this is not the case, the paths will 'hit' the outer boundary circle. Such paths will merge as they follow along the outer boundary and are hence not disjoint. However, this is actually a case that will be handled by the algorithm in the following section as the outer boundary is also a topological hole.

The above analysis is done in the continuous setting. We will present the multiple path routing results in a discrete setting by simulations.

5.3.2.2 Network With Holes

Consider a circular domain D with k holes (including the outer hole, which can be regarded as a circle centered at ∞). In this case, finding disjoint paths is more complicated. This is precisely because if two paths both hit the same hole,

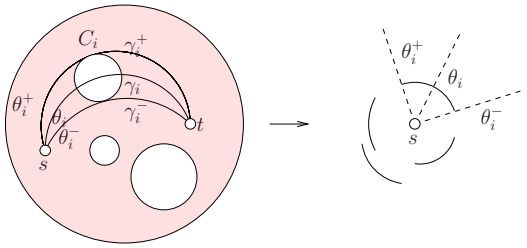


Figure 22: For a pair of source and destination, each hole C_i will produce two intervals θ_i^+ and θ_i^- such that any two paths falling in the same interval will hit the hole and share some segments of the boundary. Thus any set of disjoint paths can only select one path inside each interval.

they will start to follow the boundary of the hole and converge. This will create a long shared sub-path on that boundary. Therefore we would need to find paths such that either (i) they do not hit the same hole, or, (ii) when two paths hit the same hole, they hit the top and bottom of the circular hole respectively so they follow the upper boundary and lower boundary and do not converge. Take a look at Figure 22. For each hole C_i in the domain, we take three circular arcs through s and t, the one that is tangent to C_i internally (i.e., including C_i); the one that goes through the center of C_i; and the one that is tangent to C_i externally (i.e., excluding C_i). These three paths are denoted as γ_i^+, γ_i, γ_i^-, respectively. Now, any two circular arcs through s and t that fall in between γ_i and γ_i^+ (or γ_i and γ_i^-) will definitely merge on the boundary of C_i. Thus we can only allow one path selected within each angular range bounded by $[\gamma_i^+, \gamma_i]$, and $[\gamma_i, \gamma_i^-]$. In the following we present an algorithm that finds a maximum number of hole touching paths satisfying the above constraints.

Given a source s and a destination t, we can first assume without loss of generality that the x coordinate of t is larger than that of s and that there does not exist a circle which passes through s and t which is tangent to more than one of the holes. For a hole C_i with center c_i, there are three arcs of relevance: γ_i, γ_i^+ and γ_i^-, as defined earlier. Let θ_i,θ_i^+ and θ_i^- denote the angles that the initial tangent vectors to γ_i, γ_i^+ and γ_i^- make with the horizontal axis. Note that the convention is that all of them are contained in the interval $[-\pi, \pi]$.

Now we define the following angular intervals $\{T_i\}_{i=1}^{2m}$, two for each of the m

holes: $T_{2h-1} = [\theta_i^-, \theta_i]$ and $T_{2h} = [\theta_i, \theta_i^+]$ ($1 \leq h \leq m$). For the outer hole, arc γ_i is simply the straight line joining s to t; while arcs γ_i^+ and γ_i^- are contained in circles that pass through s and t and are tangent to the outer boundary at the top and bottom.

Now any two circular arcs joining s to t, both of whose initial directions lie in the same interval T_{2h-1} or T_{2h}, will both traverse either the upper or the lower boundary of the hole C_i, and hence cannot be disjoint. We want to find the maximum number of arcs, all of which pass through s and t, and touch at least one hole.

Since all the T_i's lie in $[-\pi, \pi]$, we can think of them as subsets of the unit circle S^1. We can angularly sort the endpoints, to obtain a sequence $s_1, s_2, ..., s_{4k}$ where each s_i is an endpoint of T_j for some j. If some interval $[s_i, s_{i+1}]$ is not contained in any T_j, we are free to use it as there are no constraints associated to such an interval. Assume that we have collapsed all the $[s_i, s_{i+1}]$ that are not contained in any T_j, and now we are left with a sequence $s_1, s_2, ..., s_l$.

Now we want to find a maximum number of intervals such that any two intervals cannot be part of the same T_i for any i. That is, each interval in the solution can be used to generate one circular arc and all such circular arcs do not intersect with each other except at s or t.

The above problem has an optimal solution using a greedy algorithm that we now describe. Each interval $A_i = [s_i, s_{i+1}]$ now is contained in (covered by) some T_j's. We obtain a solution Q_i as follows. Q_i starts with a seed interval A_i. Move clockwise on the circle until the first interval $[s_{j_1}, s_{j_1+1}]$ which is not covered by any T_j that covers A_i. When this happens, include A_{j_1} in the solution Q_i and proceed greedily until we cannot include any more intervals to Q_i.

After this process, we have l solutions $Q_1, Q_2, ..., Q_l$. We choose the one with a maximum number of intervals. This solution is optimal.

Theorem 19 *The number of intervals in the solution chosen above (i.e., the best amongst the $\{Q_i\}_{i=1}^l$) is equal to the number of intervals chosen in the optimal solution.*

Proof 20 *Pick any interval that the optimal solution chose, say A_{j_1}. Consider what the greedy algorithm performed in Q_{j_1}. Let the next interval chosen by the greedy*

algorithm be G_{j_2} while the one chosen by the optimal be O_{j_2}. Assume they are different (if they are the same the argument proceeds). If there does not exist j such that G_{j_2} and O_{j_2} are both contained in T_j, then adding the interval G_{j_2} to the optimal solution increases the number of intervals in the optimal, which is a contradiction. So assume T_j contains both G_{j_2} and O_{j_2}. Therefore by the end of T_j we have not done any worse than the optimal, as both have added one interval each. The argument now proceeds in a similar fashion. G_{j_3} and O_{j_3} would both have to be contained in some T_j again, by the end of which we are again no worse than the optimal and so forth. Inductively we can show that our solution is no worse than the optimal and thus must be optimal.

Using this greedy method we can thus find a maximum number of node disjoint paths in the domain all of which pass through a hole. We can use results of the previous section to generate node disjoint θ spread paths in the intervals $[s_i, s_{i+1}]$ which were not covered by any T_j. Thus putting together, we can find a maximum number of disjoint paths that do touch some hole and depending on the spread we can find disjoint paths that do not touch any hole using previous results.

To summarize, our multipath algorithm will generate k disjoint paths in a network with and without holes by applying different Möbius transformations, with provable results for the continuous case. When the nodes have high density, the greedy paths in the discrete case will better approximate the circular arcs. When the node density drops, the multiple paths may overlap in the middle. We evaluate in the simulation section the dependency of the performance on the network density.

5.3.3 Recovery From Failure

In this section we describe how to deal with en-route node failures or link failures. Recall that our embedding produces a circular domain that guarantees delivery when links are assumed to be reliable. When a link may fail, greedy routing no longer guarantees delivery. For example, a node may discover that all the neighbors that are closer to the destination are not reachable. In this case we aim to find an alternative path. The freedom of applying Möbius transformations on a circular domain provides great flexibility for this task.

Assume s is the source node that wants to transmit a package to t ($s, t \in D$). Let the degree of s be ν. Furthermore, assume that s has sorted its neighbors in increasing order of their distances from t such that

$$||p_1 t|| \leq ... \leq ||p_k t|| \leq ||st|| \leq ||p_{k+1} t|| \leq ... \leq ||p_\nu t||.$$

$||uv||$ is the Euclidean distance between u, v. If the link between s and any of the $\{p_i\}_{i=1}^{k}$ is functional, s routes the message to that neighbor just as in greedy routing. Assume that the only links available to s are those in $\{p_i\}_{i=k+1}^{\nu}$. Then s picks a functional link from this set, say the link to $p = p_{k+1}$. Now the idea is to find a Möbius transformation such that in the new embedding, p is closer to t than s, so that greedy routing would then continue by using p as the next hop. The details are presented below.

As soon as node s finds that greedy routing can no longer continue, it does the following:

1. It finds the coordinates of a neighbor p. Assume that s knows the coordinates of p and the destination t.

2. s finds a Möbius transformation that maps s to -1, p to 0 and t to 1. The explicit formula for f is

$$f(z) = \frac{z(s-t) + p(t-s)}{z(2p - (t+s)) - p(t+s) + 2st}.$$

3. s then sends the package to p along with the information about this Möbius transformation. p calculates new coordinates for all of its neighbors and for t.

4. Greedy routing then continues (since now $f(p)$ is clearly closer to $f(t)$ than $f(s)$), until we get stuck at another node. When this happens, we repeat the entire process, i.e., find another Möbius transformation and compose it with the previous one.

As will be shown later in the simulation section, our failure recovery mechanism is compared with random walk – simply pick a random 'live' link until greedy

routing can be performed again. Basically our scheme makes big jumps and chooses a vastly different alternative path while random walk can only make local adjustments. This benefit of using long de-tours is significant for failures that exhibit spatial patterns.

5.4 Simulations

In the experiments, we perform greedy routing with Möbius transformations to achieve multipath routing and link failure recovery. Our simulations are performed on unit disk graph topologies potentially with holes inside, and in the following are our key observations.

Multipath routing. By using Ricci flow with different Möbius transformations, we can generate a substantial fraction of node disjoint paths. With reasonable sensor density (average degree around 20), the average number of disjoint paths we find using our algorithm is consistently more than 70% of the input parameter m (the desired number of disjoint paths). We can consistently find two node disjoint paths even in very sparse networks. We also observed that when the network is sparse, the bottleneck for finding node disjoint paths is often near the source and destinations.

Recovering route from link failures. Under a spatial failure model in which the nodes in a geometric failure region have a much higher failure rate, our method of using greedy routing on the virtual coordinates in a circular domain with Möbius transformations as the recovery scheme performs consistently better than all other methods (on virtual or original coordinates, using random walk as the recovery scheme). The advantage of using Möbius transformations rather than random walk as a recovery scheme diminishes when the failure pattern is no longer spatially correlated.

5.4.1 Multipath Routing

After we generate the sensor network $G = (V, E)$, we randomly choose two vertices s and t from V as the source and the destination. We then calculate the maximum number of node-disjoint paths between s and t, called the vertex connectivity $\kappa(s, t)$, as a reference for comparison, using the centralized maximum flow

$\kappa(s,t)$	m	Disjoint paths generated	Approximation factor
6	3	2	67.7%
	5	4	
8	3	3	62.5%
	5	4	
	7	5	
11	5	4	72.7%
	9	7	
	11	8	

Table 4: Results of different sources and destinations in a uniform distributed graph with average edge links 20.

algorithm [35]. To test our multipath routing algorithm, we generate m (no larger than $\kappa(s,t)$) paths from the source to the destination, and count how many of them are node-disjoint. We also try different m parameters to further observe the performance of the algorithm.

Figure 23 shows the proposed routing scheme on a sensor network with 1000 vertices and 10006 links. We first apply Ricci flow to embed the network into a circular domain where each node is given a virtual coordinate. We use our multipath routing algorithm to seek $m_1 = 3$, and $m_2 = 5$ paths from s (in yellow) to t (in red) respectively (in this graph $\kappa(s,t) = 6$). Those paths are not necessarily node-disjoint, and all the shared nodes/edges are marked in purple in the figure. We also show the paths on the original network, and a different circular domain obtained by a Möbius transformation as well. In each of the three embeddings, the paths with the same color and number are identical. We can see that in different circular domains, the greedy paths, or the straight lines from s to t are also different, which demonstrates that Möbius transformations together with greedy routing give us flexibility in choosing multiple paths.

More results are shown in Table 4. From the table we can see two facts. First, as a distributed algorithm, the Möbius transformation method gives us a good approximation of the number of node-disjoint paths in a dense graph. Second, the number of disjoint paths we can get is usually smaller than m, or equivalently,

Nodes	Average link number	Input m	Average output paths / m
1000	20.00	3	83.3%
		5	79.0%
		7	71.4%
600	12.02	3	76.7%
		5	64.0%
		7	51.4%
400	5.62	3	63.3%
		5	46.0%
		7	35.7%

Table 5: Results of graphs with different sensor densities.

some paths we generate share common nodes or edges. This is due to the discrete nature of graphs. From s, we can only send packets to its neighbors, which is a restriction in choosing the first few hops of the transmission; the hops near t suffer from the similar problem. But in the middle segments, the paths generally follow the shape of a circle arc connecting s and t, which is desired. When the network becomes denser, the situation becomes more similar to that of the continuous case.

To further explore the differences between the discrete and continuous settings, we also simulate under different graphs. In graphs with different densities (with uniform sensor distribution), we randomly pick 10 pairs of sources and destinations, give the different inputs m as 3, 5 and 7, and calculate the average numbers of disjoint output paths we get. The results are shown in Table 5. From the table, we observe that the algorithm performs better in a denser graph with more links. This is reasonable since the gap between the discrete and continuous settings is smaller with a denser sensor distribution. Moreover, when the input m is smaller, the algorithm gives better approximation. This is simply because given a smaller input, the arcs span further away with each other (this result does not conflict with Table 4 where $\kappa(s,t)$ is known). We also notice that when m exceeds the average node degree, the percentage drops drastically, where the input – the number of disjoint paths we are trying to find – often exceeds the optimal value.

When the sensor distribution is non-uniform, the bottleneck of the performance lies in the sparse regions, especially when those regions cover the neighborhood of the source or the destination.

Figure 24 shows the result of a network region with holes. Some paths go along the inner boundary, but the heavy load along the boundaries is avoided.

5.4.2 Routing with link failures

In sensor networks, links are likely to fail, especially in an adversarial environment. Since greedy routing requires that for each step, there exists one link leading to a node closer to the destination, the performance of greedy routing will drop quickly when a link failure happens. What is more, link failures often have a property of spatial locality, which means that a group of nearby links are likely to fail at the same time. Therefore, when greedy routing hits this region, the message may 'get trapped in the mud'. We use the freedom of Möbius transformations to recover from this situation.

Based on the above observations, we adopt the setting of clustered random link failures. We first test a simple setting in which there is a region with arbitrary size and shape. All the links within this region have a link failure rate p, while all the links outside the region or crossing the boundary of the region will not fail. A message following greedy routing in the circular domain may not have guaranteed delivery as the link to the next hop can suddenly fail. Our strategy is to adopt a different Möbius transformation. We compare it with another simple strategy that recovers from failure by performing a random walk. Although random walk is simple, it is time-consuming for the routing path to jump out of a large link failure region, due to its locality and randomness. In our following experiment, we will compare greedy routing with Möbius transformations with other greedy routing techniques, in terms of routing delivery rate and routing path length. The experiment network size is 1000 nodes with a varying number of links. The link failure region is a rectangle lying in the network.

In the experiments, we compared the following methods.

Greedy routing on the original coordinates. Simple greedy routing on the original coordinates, which fails to route to the destination easily due to link failures.

We call this method *Greedy* in short.

Greedy routing on the virtual coordinates. Greedily route to the destination using coordinates computed by Ricci Flow. We call this method *Ricci* in short.

Greedy routing on the original coordinates with random walk.] Route based on the original coordinate set and perform random walk to recover from failure. We call this method *GreedyRnd* in short.

Greedy routing on the virtual coordinates with random walk. Greedily route using virtual coordinates and perform random walk to recover from link failures. We call this method *RicciRnd* in short.

Greedy routing with Möbius transformations. Our method performs greedy routing based on the virtual coordinates in a circular domain. If the route gets stuck in the middle due to link failures, it performs a Möbius transformation to get a new path towards the destination. We call this method *Möbius* in short.

Various parameters will affect the success rate of routing. In our experiment, we focus on the average degree of the network, the link failure rate and the TTL (time-to-live) of the packet. If the connectivity of networks becomes better as the average degree increases, routing will be easier for all methods. Obviously a higher link failure rate will make all methods suffer. We also include a TTL with each packet to stop a packet from roaming aimlessly in the network, in particular in the random walk method.

Routing result and analysis. In Figure 25, 26 and 27, we show the message delivery rates by varying the network density, the TTL and the link failure rate respectively. In all settings, we can see our method has a significantly higher delivery rate than the other methods, which shows that by using a new Möbius transformation, we can effectively find an appropriate path which leads the route out of the failure region. In general the performance of different methods, in decreasing order, follows the trend of $\textit{Möbius} > \textit{RicciRnd} > \textit{Ricci} \gg \textit{GreedyRnd} \simeq \textit{Greedy}$. Greedy routing using the original coordinates nearly does not work, no matter whether it is augmented with random walk or not.

Figure 25 shows the performance of all methods on networks of different node average degrees. The performance of all methods deteriorates when the network becomes sparse. Still our method is the leader. Note that in all cases we must first

p_1	p_2	Möbius	Ricci	RicciRand	Greedy	GreedyRand
0.3	0.0	0.962	0.573	0.779	0.021	0.028
0.6	0.0	0.912	0.402	0.651	0.014	0.016
0.6	0.3	0.738	0.207	0.472	0.003	0.009
0.9	0.0	0.823	0.271	0.511	0.008	0.013
0.9	0.3	0.632	0.148	0.335	0.002	0.006
0.9	0.6	0.472	0.037	0.193	0.001	0.004
0.6	0.6	0.587	0.094	0.263	0.002	0.004
0.3	0.3	0.802	0.254	0.513	0.007	0.011
0.3	0.6	0.591	0.119	0.285	0.002	0.005
0.3	0.9	0.224	0.017	0.104	0.001	0.002
0.6	0.9	0.152	0.011	0.063	0.001	0.002

Table 6: Comparison of different p_1 and p_2 settings.

make sure that the network is connected and has a triangulation for computing the virtual embedding as a circular domain.

Figure 26 shows the importance to choose an appropriate TTL. As TTL grows, the delivery rate of our routing method grows rapidly and then stays close to 1, while the other methods do not exhibit a growing trend with TTL. This shows that the Möbius transformation scheme does recover from link failures and makes progress towards the destination, while the other methods still get stuck in the middle. Figure 28 shows the distribution of path length under different methods. Indeed our method gradually delivers more messages when TTL is increased.

Another potential factor affecting the routing performance is the distribution of link failure rates. We test the situation where links lying inside and outside the failure regions have failure rates p_1 and p_2, respectively. We evaluate the performance by varying the difference between p_1 and p_2. From the simulation results shown in Table 6, we can see that our method consistently outperforms the other methods in different settings. When p_1 and p_2 are getting close, *RicciRnd* starts to catch up. In these experiments we also vary the shape and positions of failure regions. While the exact values may vary, the trend is clear: our method makes progress towards the destination despite the existence of high link failure rate, and

is unlikely to get stuck in the middle; making long de-tours is generally better than taking local random walks.

5.5 Discussion

In this work we presented a method that uses circular domain embeddings and Möbius transformations to switch between them for multipath routing and improving routing resilience. This shows the power of a geometric transformation that regulates a network shape — difficult routing problems due to lack of global knowledge can benefit significantly from such transformations. We expect to extend the intuition to more problems on distributed network setting in the future.

Our multipath routing algorithm in the discrete case uses a heuristic method that maximizes the angular spread of the paths at source and destination. It would be a very interesting problem to exploit the freedom of using such routing scheme to improve network throughput, or optimize energy usage, etc.

We would also like to mention that the paper borrows heavily intuitions that arise from the continuous domain. Our provable results are in the continuous case and the performance of the algorithm in the discrete case is only evaluated by simulations. The problem of addressing the gap between the continuous space and a discrete graph, theoretically, is yet an open problem. It would be an interesting and challenging problem to come up with a suitable discrete model and derive minimum density bound. We leave this for future work.

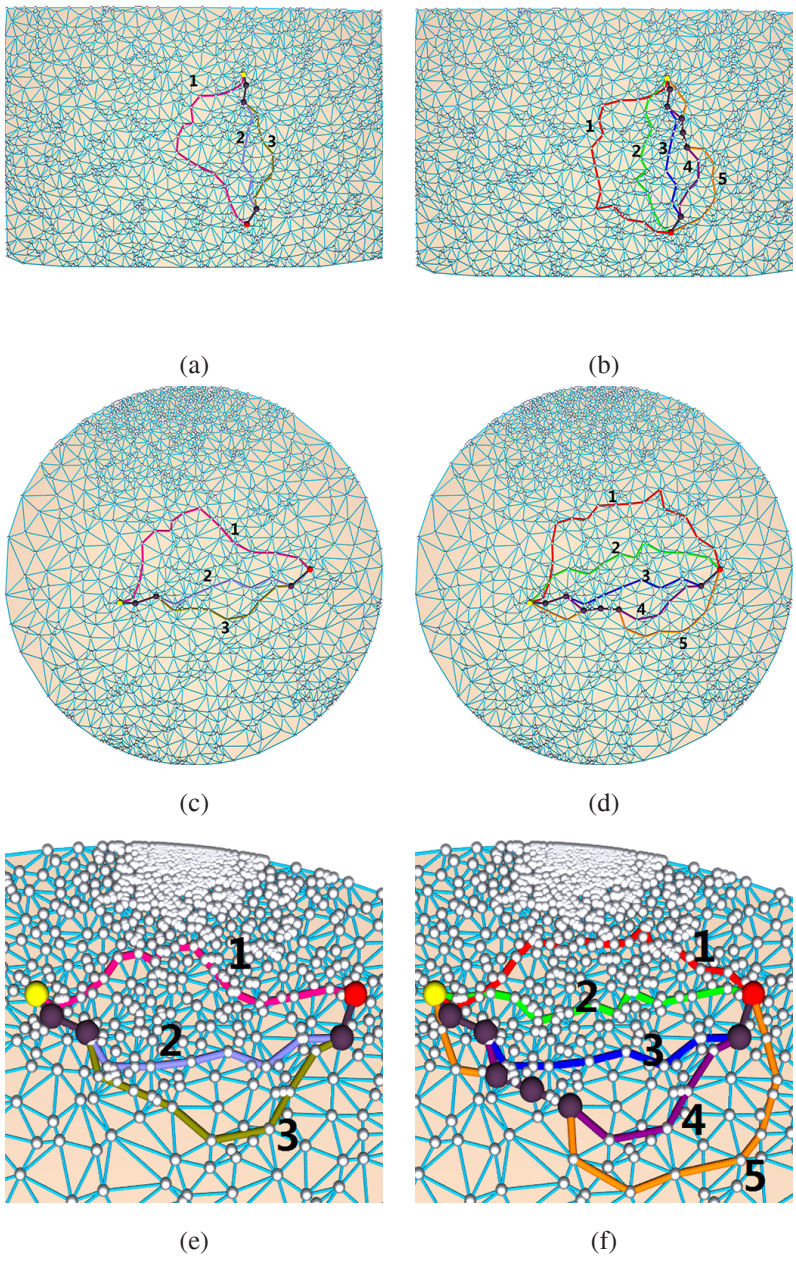


Figure 23: Multipath Routing Algorithm. (a) and (b) are the original networks. (c) and (d) are the networks applying Ricci flow. (e) and (f) are the networks applying Ricci flow and a Möbius transformation (zoomed in). First column: $m = 3$; second column: $m = 5$.

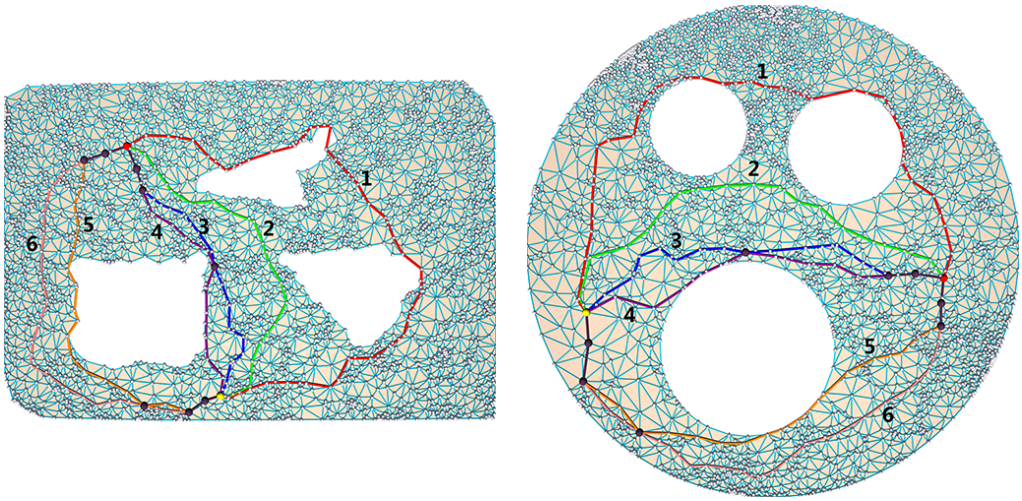


Figure 24: Multipath Routing Algorithm in a region with holes. Up: original network; bottom: network applying ricci flow. Here $\kappa(s,t) = 9$.

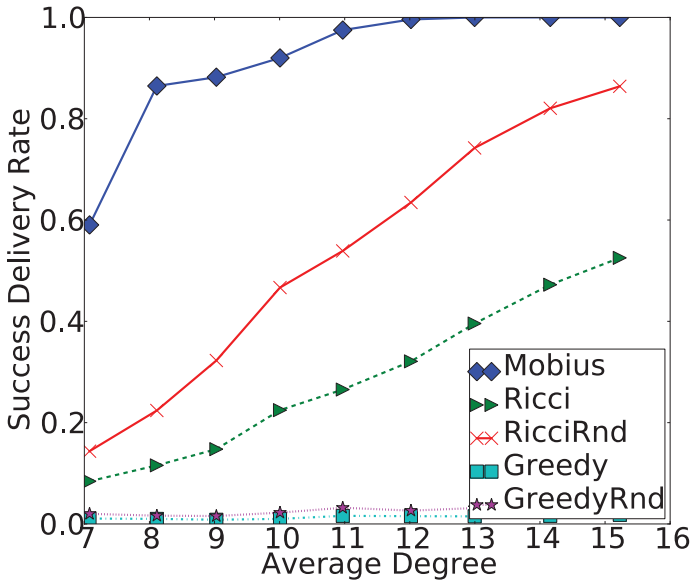


Figure 25: Routing delivery rate versus average degree (TTL = 500; link failure rate = 0.8). *Möbius* is our method. *Greedy* and *Ricci* are greedy routings on the original and Ricci Flow coordinates respectively. *GreedyRand* and *RicciRand* are greedy routings on the original and Ricci Flow coordinates with random walk respectively.

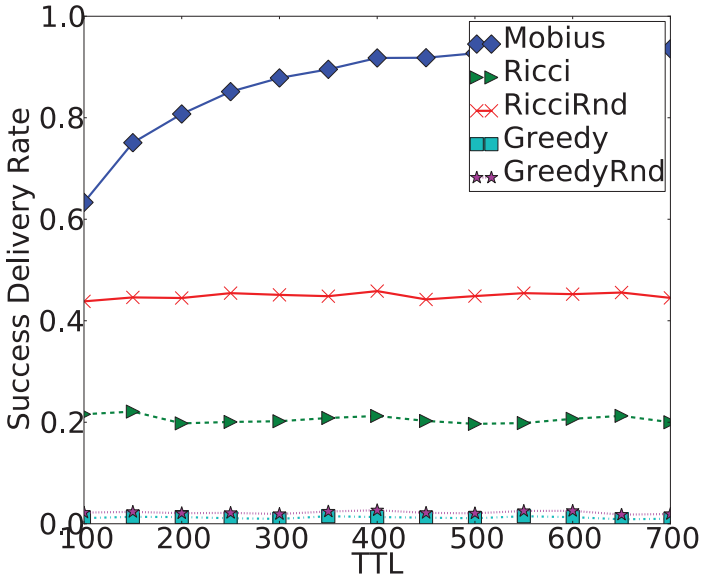


Figure 26: Routing delivery rate versus TTL (time-to-live) of packets (AvgDegree = 10; link failure rate = 0.8).

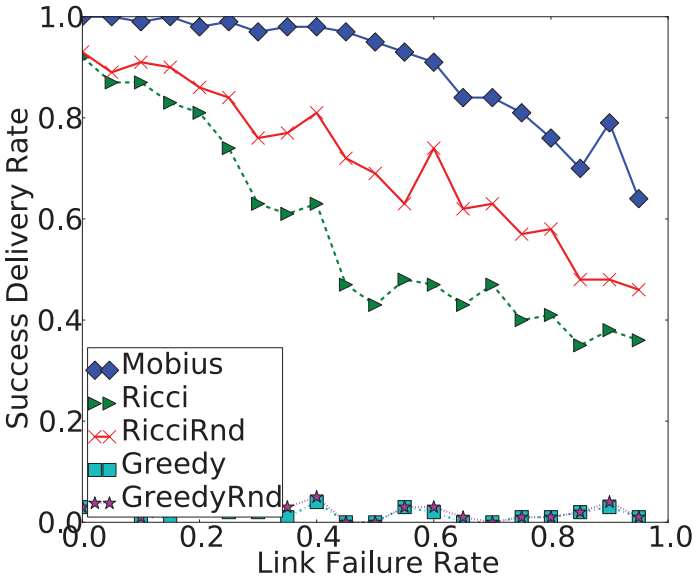


Figure 27: Routing delivery rate versus link failure rate (AvgDegree = 10; TTL = 500).

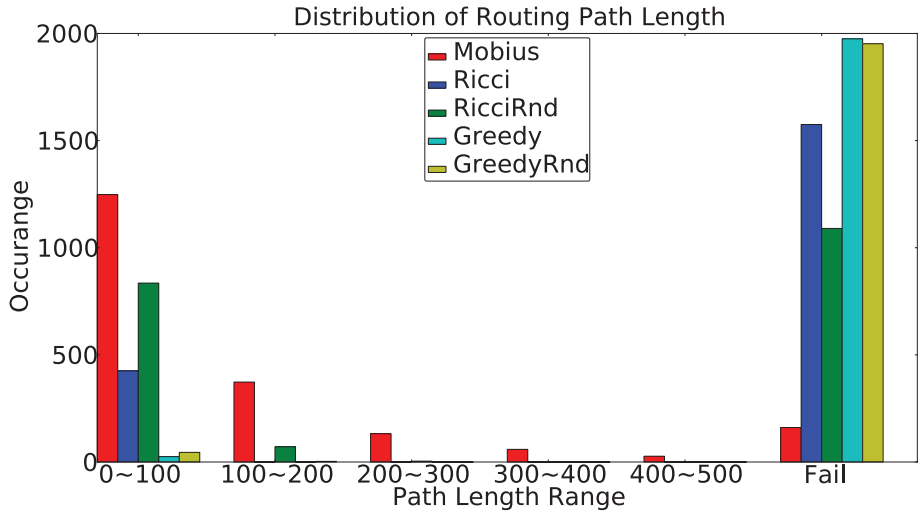


Figure 28: Distribution of routing path lengthes (AvgDegree = 10; TTL = 500; link failure rate = 0.8).

Chapter 6

Wormhole Attack Detection and Removal

6.1 Introduction

A wormhole attack to a wireless network [67] is to place two radio transceivers, connected by high capacity out-of-band wireless or wired links. Signals captured by one antenna are "tunneled" through the wormhole link to the other antenna, and replayed there. In the 'store-and-forward' scheme, the wormhole nodes copy the entire packet before transmittal through the wormhole link. In more sophisticated schemes, the wormhole can be launched *at the bit level* (the replay is done bit-by-bit even before the entire packet is received, similar to cut-through routing [116]) or at the *physical layer* [54] (the actual physical layer signal is replayed, similar to a physical layer relay [141]). Effectively, the wireless nodes near one wormhole antenna find out that they can directly communicate with the wireless nodes near the other antenna and would consider them as immediate neighbors. See Figure 29. A wormhole attack is easy to launch. It is independent of the MAC (medium access control) layer protocols and is also immune to cryptographic techniques. It does not require the adversary to break into the wireless nodes or understand the communication mechanisms employed by the network.

If the adversary only replays the signal faithfully, the presence of wormhole is of no harm or even beneficial as it enhances the network connectivity and creates

short paths between otherwise far off regions. When the tunneled distance is larger than the transmission range in the network, nodes near the wormhole antennas find shorter, faster, and probably more reliable paths by tunneling through the wormhole. Wireless networks running any variations of shortest path routing will discover such paths and eventually make use of them to deliver data. For example, take a simple scenario where nodes are uniformly deployed in the domain with d nodes per unit area on average and the wormhole antennas are placed of distance k apart, roughly at least $\pi dk^2/8$ pairs of nodes will find shorter paths through the wormhole link. In another case when one radio transceiver is placed next to a data sink in a sensor network, the wormhole link provides shortcut paths to the sink for $\pi dk^2/4$ nodes. Therefore, a wormhole attack, in particular one with a long tunneling distance, will be able to attract a lot of traffic through the wormhole link. This puts the wormhole link at a powerful position than other nodes in the network and this allows the adversary to exploit this position in a variety of ways.

Since a wormhole attack fundamentally changes the network connectivity, by turning on and off the signal replay an adversary can suddenly create and destroy a large number of shortest paths in the network and upset most routing protocols. In on-demand routing protocols, a wormhole can attract the route request packet through the tunnel and later play denial of service attack by refusing to forward any packets. In routing protocols that periodically discover neighbors, the adversary can trigger frequent neighbor changes and paths changes, which consumes the node energy and communication bandwidth. Even when the wormhole does not shut down its replay scheme, the wormhole can be used to attract network traffic, and can then eavesdrop, maliciously drop packets, or to perform man-in-the-middle attacks. Traffic gathered this way can also help to break encryption and security mechanisms used in the network. Thus wormhole attack opens the door to many more malicious attacks. We measure the *impact* of a wormhole attack by the number of pairs whose shortest paths are affected by the wormhole attack. In this sense, a wormhole attack has larger impact/potentially more damages when the two antennas are placed relatively far away, as more traffic and more paths in the network are affected by the wormhole link. We call such a wormhole to be a 'long' one and it is of most interest to detect those long wormholes in the network.

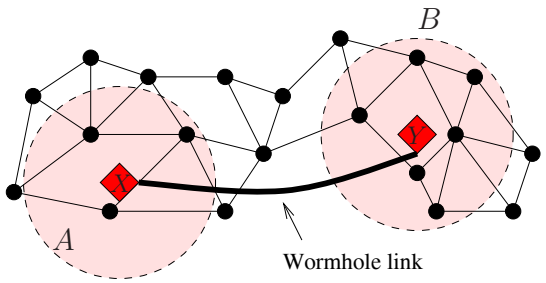


Figure 29: Demonstration of a wormhole attack. X and Y denote the wormhole nodes connected through a long wormhole link. As a result of the attack, nodes in Area A consider nodes in Area B their neighbors and vice versa.

In addition to messing up with the routing protocols, using wormholes an attacker can also break any protocol that directly or indirectly relies on geographic proximity. For example, target tracking applications in sensor networks can be easily confused in the presence of wormholes. Similarly, all localization algorithms that use network connectivity would fail or be confused by the alteration of the network topology due to wormhole links. This can have a major impact as location is a useful service in many protocols and application, and out-of-band location systems such as GPS are not always available.

6.2 Related Work

In the literature a number of techniques have been proposed to detect wormhole attacks. These methods have their respective limitations, e.g., assuming additional hardware or explicit communication models or lacking the ability to single out wormhole links. We first review the prior work and then describe our approach using novel algorithmic techniques.

Methods using distance or timing analysis. Packets going through a wormhole take longer to reach the destination due to the delay in reception, transfer and retransmission at the other end. A number of schemes have tried to detect wormhole attacks by measuring packet traverse distance or time. Such methods are generally called packet leashes [29, 54, 66, 140]. The limitation of this method is that one needs to obtain the node location information using out-of-band mechanism such

as GPS, or, extremely accurate globally synchronized clocks to bound packet propagation time. It is unclear whether the techniques can be carried out in low-cost hardware such as sensors. Even if so, such timing analysis may not be able to detect cut-through or physical layer wormhole attacks, as such replays can happen quite fast and cannot be detected easily.

Methods using special hardware. Using purely physical layer mechanisms one can prevent wormhole attacks such as those involving authentication in packet modulation and demodulation [66]. But such techniques require special RF hardware. Directional antennas can also be used to prevent wormhole attacks [65]. The requirement of special devices limits the use of such protocols.

Methods using special guarding nodes. A few protocols of this type [80, 81, 123] have been proposed that use special-purpose guard nodes with known locations, higher transmit power and different antenna characteristics, to attest the source of each transmission. The use of such special purpose guard nodes makes this approach limited in applicability.

Methods using neighborhood discovery. Since the placement of wormhole increases the local connectivity at the neighborhood of the wormhole nodes, one can use statistical approaches to detect the increase in number of neighbors and the decrease in lengths of shortest paths between all pairs of nodes due to wormhole presence [27]. A similar approach using statistical measurements of multi-path routing is used in [125]. Both schemes assume that the network is free of wormhole to start with and they are vulnerable if the attack is launched prior to such discovery.

A different approach examines the changes in the connectivity graph by the wormhole attacks and look for 'forbidden substructures' in the connectivity graphs that should not be present in a legal connectivity graph [105]. This approach however assumes fairly detailed knowledge of wireless communication model (i.e., a model that describes with some given confidence whether a link between two nodes should exist) and the performance deteriorates if such a model is lacking.

Methods using global network topology. The last family of work examines the global network topology. Essentially the wormhole attack drastically changes the network connectivity by 'gluing' links between the nodes near wormhole nodes. In [156], distance estimates between sensors are used to determine a "network layout" using multi-dimensional scaling (MDS) technique. Without any wormhole the

network layout should be relatively flat. But the layout could be warped in presence of wormholes. Thus detecting whether the network can be embedded on a flat domain can tell whether wormhole attacks are present. This method is centralized and it does not identify nor isolate wormhole attacks.

Dong *et al.* [44] uses the local topological changes around the neighborhood of the wormhole nodes to detect the wormhole links. In particular, one takes a local k-hop neighborhood and see whether the 'boundary' has single or double cycles. Intuitively, the neighborhood that encloses a wormhole link will have two cycles and single cycle otherwise. The limitation of the method is that it requires relatively high node density to ensure that boundary detection algorithm works well, and relies on the local hop count metric being close to the Euclidean metric. They suggest using global topological properties to detect presence of wormholes in [43]. This idea has some merit for certain 2-manifolds, but do not carry over to actual networks, since real world network graphs are not surfaces.

6.3 Overview of Our Approach

We search for a detection method that is not limited to the various constraints as described earlier. The approach we use is to examine graph connectivity, and detect the fundamental connectivity changes a wormhole would introduce. This puts us into the family of protocols that test the network connectivity or global topological changes, such as those described in [43, 44, 105, 156]. Compared with these work, our method makes contributions in the following aspects.

Rigorous Definition of A Wormhole Attack. None of the previous connectivity based detection method has a rigorous definition of what constitutes a wormhole attack in the connectivity graph. Thus there is no provable results on detection ability and the algorithms rely on simulations to evaluate the performance. We introduce a rigorous definition of how a wormhole attack affects the network connectivity. Basically a wormhole would 'shortcut' the paths between two sets of nodes W_0, W_1 that can directly communicate with the two wormhole antennas respectively. Therefore, the wormhole attack introduces links between nodes in W_0 and W_1 and adds the full bipartite graph on W_0, W_1 to the existing topology. The length of the wormhole is dictated by the shortest hop count between nodes in W_0 and nodes in W_1

before the wormhole is introduced.

Guaranteed Detection of Wormhole Sets. All previous algorithms are conservative, in the sense that it is possible to report no wormhole while there is one even in the case of a long wormhole (connecting nodes that are far away in the original network). We consider the false negative to be more dangerous than false positive (that certain legal links are labeled as suspicious). When a false positive link is removed, a valid communication link is lost, but security is not compromised. A false negative, on the other hand, leaves the network insecure. We *prove* that our algorithm *guarantees* to detect all the nodes affected by the wormhole attack. Abstracting away some technical details, in our method we remove a local neighborhood around a node p and check whether a slightly larger neighborhood is connected. If not, p is considered as a suspicious node. We prove for all suitable parameters this simple test is *guaranteed* to identify all the nodes affected by a wormhole. By repeating the test for different sets of parameters we can also substantially reduce the number of false alarms. With the candidate sets, we include additional tests to verify that it is indeed a wormhole structure in our definition. Thus a wormhole set is provably and accurately detected.

Robustness to Different Communication Models and Dimensions. We remark that our detection algorithm looks at network connectivity alone. Thus the method applies to any general network settings. For example, the method does not require any assumption nor knowledge of the wireless communication models (as opposed to the method in [105]). It does not use any geometric intuition that relies on the network being embedded in the plane, as opposed to the methods in [43, 44]. The same algorithm works on networks deployed in 3D.

Scalability and Communication Costs. Our detection algorithm at a node p only uses information of a small bounded neighborhood of p. Thus naturally the algorithm is scalable to networks of large size. The communication cost for the test is low, dependent only on the network degree for each node.

We evaluated the detection performance (in terms of false positive and false negative) with connectivity based methods [43, 44]. The results show that our method has better performance in detecting wormholes. In particular when the network model does not follow unit disk graph model the performance of other

methods deteriorates substantially. Our method has slightly more false alarms but the detection of wormhole attacks is accurate.

In the following we first present the definition of a wormhole set, the threat model, and then describe the algorithm to detect nodes affected by a wormhole attack. We also discuss methods to eliminate false alarms and to detect multiple wormholes. We then present simulation results and comparisons with other connectivity based methods.

6.4 Wormhole and Local Connectivity Tests

Our algorithm is to detect the anomalies in the graph connectivity. To start we first rigorously define what is the connectivity structure of a wormhole and then describe our algorithm.

6.4.1 Assumptions and Threat Model

In a wireless network communication links can possibly be directional. That is, A can send messages to B but not vice versa. We only consider the bidirectional links, as directional links do not support acknowledgement schemes. We assume that the transmission characteristics of the wormhole transceivers are the same as that of the other legal nodes in the network, to enable bidirectional communications.

We assume that the adversary can place wormhole nodes at arbitrary places in the network, and that these nodes are connected through a communication channel that is unobservable by other nodes. The wireless network can adopt efficient symmetric cryptographic schemes (as in [67]) to authenticate communication partners and protect the communication messages. The wormhole attacker simply sniffs traffic on one end and replays on the other end. That is, the attacker does not need to know the cryptographic schemes used in the network to fool the nodes to believe that they have a direct communication link. The wormhole transceivers also do not have identities. In fact, the wireless nodes are not aware of the presence of any special wormhole radios in the neighborhood and just hear about some messages in the air, that are possibly replay messages.

We assume a wireless ad hoc network in which the nodes are not compromised

nor malicious. In particular, there is no Sybil attack [69], where a malicious node behaves as if it was a larger number of nodes, for example by impersonating other nodes or simply by claiming false identities. We will discuss the case of compromised or malicious nodes in the discussion section.

6.4.2 Wormhole Definition

We start with an unweighted communication graph $G = (V, E)$. A wormhole attack captures the signal in the air from one radio transceiver A and then broadcast from another radio transceiver B. As a consequence, all the nodes whose signal reach A and B respectively will think they have direct communication links. This creates a local structure of a full bipartite graph as a subgraph. The damage from a wormhole attack is defined as the number of pairs discovering shorter paths through the wormhole link. Thus, further away the two radio transceivers are, more damage is done by the wormhole attack. On the other hand, very short wormholes do not significantly modify connectivity and are not such a threat. Our wormhole definition captures this parameter by measuring the hop distance k between the nodes connected through a wormhole in the original network in absence of the wormhole. We assume that k is greater than a sufficiently large constant. All our tests will only use a bounded neighborhood of size determined by k around each node.

Definition 21 (k, τ)**-wormhole set.** *A set $W \subset V$ is a (k, τ)-wormhole set if it is a maximal disjoint union $W_0 \cup W_1$ for which the following conditions hold:*

1. *Each edge $(u, v) \in W_0 \times W_1$ is in E. That is, each node in W_0 is a neighbor of each node in W_1. Such edges are called* wormhole edges.
2. *$|W_0|, |W_1| \geq \tau$, that is, there are at least τ nodes whose signals are captured by the wormhole link on either side.*
3. *Removing all wormhole edges $W_0 \times W_1$ increases the distance between W_0 and W_1 to be at least k, but does not disconnect any part of the network.*

The set W is said to *maximal* in the sense that no node can be added to it while keeping true to the conditions above. This definition implies that the diameter of W

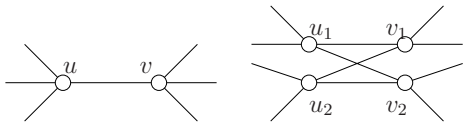


Figure 30: A legal network structure such as a bridge connecting two nodes on the boundary of a hole could also be identified as a 'wormhole' in our definition. However, the same graph structure can be generated by also placing wormhole antennas near u and v. Thus it is impossible to eliminate this case from our definition.

is at most 2. Sometimes we write a wormhole set simply as a k-wormhole to mean that τ is not relevant, or equivalently, $\tau = 1$.

We remark that in certain cases, legal links can be identified as a wormhole set. Consider a network with a 'bridge' connecting two nodes that are otherwise far apart in the network. Such a bridge or bridge like structure falls in our definition. See Figure 30. But such bridges could also be the result from a wormhole attack and there is no way to distinguish them from a real wormhole attack based on graph connectivity only. Thus, our tests will be on the aggressive side and also identify such structures, and report them for further investigation.

Finding a complete bipartite subgraph can be done in the centralized setting when the entire network topology is available. Eppstein [51] shows an algorithm that lists all complete bipartite subgraphs in a network with constant degree. The running time of the algorithm is linear in the size of the graph and exponential in the node degree. We will use this algorithm on local neighborhoods in the final stage of our algorithm to test that the wormholes detected have τ nodes on each side.

6.4.3 Local Connectivity Test

The idea in our test is to observe that a wormhole attack connects two sets of nodes that are otherwise far away in the graph, while the wormhole set itself is contained in a very small neighborhood. As a node near a wormhole expands its neighborhood, the neighborhood grows on two sides of the wormhole edges. Removing a small region around the node removes the wormhole and disconnects the neighborhood into two components.

Thus our local connectivity test is to check whether a neighborhood of a proper size will fall into multiple connected components. Since wireless communication

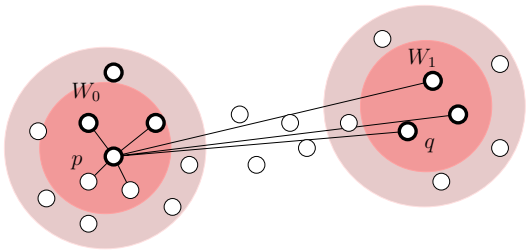


Figure 31: The thick circles represent the nodes within the wormhole range, those on two sides correspond to W_0 and W_1 respectively. The physical wormhole link is not shown since it is not visible in the network connectivity. The darkly shaded region denotes the ball $B_1(p)$, which includes all nodes in W_1. Thus removing $B_1(p)$ also removes all wormhole edges. The lightly shaded region denotes the ring $N_{[1,2]}(p)$. It has two components, one near W_0 and one near W_1.

has a lot of local spatial variations, checking the 1-hop neighborhood does not give reliable results. Thus we consider neighborhoods of different sizes. To be precise, we will introduce the following definitions.

Definition 22 **α-ball and $[\alpha, \beta]$-ring.** *An α-ball centered at node p, written as $B_\alpha(p)$ is the set of all nodes with distance at most α-hops from p. All the nodes that are within β hops from p but are more than α-hops away from p are called the* $[\alpha, \beta]$-ring $N_{[\alpha,\beta]}(p)$. *In symbols :* $N_{[\alpha,\beta]}(p) = B_\beta(p) \setminus B_\alpha(p)$. *$\alpha$, β are integers satisfying $\beta > \alpha \geq 1$.*

To test for a wormhole, we first introduce a basic $[\alpha, \beta]$-ring-connectivity test, where α, β are integers satisfying $\beta > \alpha \geq 1$.

Definition 23 **$[\alpha, \beta]$-ring-connectivity test for node p.** *Consider the set of nodes $N_{[\alpha,\beta]}(p) = B_\beta(p) \setminus B_\alpha(p)$, and the subgraph in G induced by it. If this subgraph contains more than one connected components, the test returns true, and we say p is a k-wormhole candidate for all $k > 2\beta$. See Figure 31 for an example.*

Guaranteed Detection of Wormhole Sets. We show that if there is a wormhole, the $[\alpha, \beta]$-ring-connectivity test always detects it successfully. For now we consider the case that the network has just a single wormhole set. First we show that the connectivity test will surely label the nodes in a wormhole set.

Theorem 24 (Guarantee of detection) *Given a (k, τ)-wormhole set W, all the nodes in W will surely be detected by the $[\alpha, \beta]$-ring-connectivity test, given that $k > 2\beta$, $\beta > \alpha \geq 1$.*

Proof 25 *Consider a (k, τ)-wormhole set W. Without loss of generality, we take representative nodes $p \in W_0$ and argue that it must be labeled as wormhole candidate. Assume otherwise, then the subgraph induced by $N_{[\alpha,\beta]}(p)$ remain as a single connected component, after we remove the α-ball of p. Recall that all the nodes in W_1 are neighbors of p, thus removing α-ball of p with $\alpha \geq 1$ will surely remove all nodes in W_1. Thus all the wormhole edges are removed as a result. Intuitively the nodes in $N_{[\alpha,\beta]}(p)$ were originally reached from p through either the wormhole edges or not using any wormhole edges. After the wormhole edges are removed, these two sets naturally form disconnected components. We make this intuition rigorous in the following.*

Consider the nodes in $N_{[\alpha,\beta]}(p)$. We define the set N_1 to be the nodes whose shortest paths to p go through nodes in W_1, and the set N_0 to be the nodes whose shortest paths to p do not go through nodes in W_1. We argue that the two sets are disjoint, and form disconnected components.

If the subgraph induced by $N_{[\alpha,\beta]}(p)$ has only one connected component, take a node $x \notin W$ in this subgraph. Since $x \in N_{[\alpha,\beta]}(p)$, x is within β hops from p. There are also two shortest paths that connect from p to x, one through the nodes in W_1 (denoted as P_1) and one not through the nodes in W_1 (denoted as P_0). These two paths, concatenated, form a cycle of length at most $2\beta + 1$. See Figure 32 for an example. We now argue that on path P_1 there can only be one node q from W_1, that is, the node immediately after p on P_1. Clearly, if there is another node $q' \in W_1$ further down the path P_1, then one can shortcut the path P_1 as p and q' are also neighbors. This will contradict with the fact that P_1 is a shortest path. Thus, removing the edges between W_0 and W_1 will still leave a path connecting p and q with total length 2β. This contradicts with the definition of a k-wormhole, where $k > 2\beta$.

The parameters α, β can be varied. Our tests are *aggressive*, in the sense that a single wormhole attack will surely be identified for suitably small values of α and β. Thus detection is always guaranteed. Different parameters may introduce different

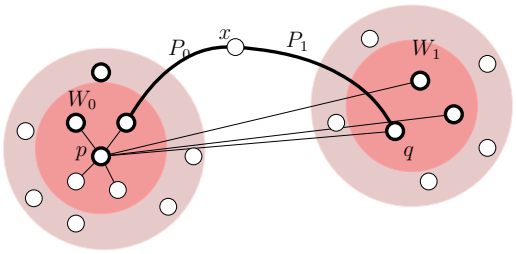


Figure 32: If $N_{[\alpha,\beta]}(p)$ has only one connected component, then there is a path connecting two nodes $p \in W_0$, $q \in W_1$ not using any wormhole edges with total length at most 2β.

type of false positives. For example, a small β is likely to introduce false positives – that is, certain nodes in sparse regions may be wrongly identified as a candidate because their small neighborhoods are naturally disconnected. But using a large β will show that it is actually not a real wormhole node, since the neighborhoods are connected by a slightly longer path. In our final algorithm we run multiple tests with different parameters and output the nodes that are labeled in *all* tests. We start with smaller values of α, β, and perform additional tests with larger values only on the nodes that are labelled as suspicious – as *wormhole candidates* – by the earlier tests. We take to be wormhole set the nodes that that are detected by all the tests up to a suitable value. Once a set of candidates are detected, we can remove the links connecting the candidates.

6.4.4 The Wormhole Algorithm

Based on the ring-connectivity test, we describe a simple distributed algorithm that identifies neighborhoods in a network as wormholes. Our goal is to detect wormholes of length k and greater. Since k must be greater than 2β, and β is at least 2, the minimum permissible value of k is 5.

Let us denote by $C_{[\alpha,\beta]}$ the set of nodes detected to be wormhole candidate by the $[\alpha, \beta]$-ring-connectivity test performed at each node in the network.

Algorithm: Connectivity Metric Test. The algorithm consists of performing the test on increasing values of (α, β) in lexicographic order, and performing subsequent tests only at nodes that are labelled candidates by all previous tests. More

precisely, we select $\alpha = 1, 2, \ldots, \lfloor (k-3)/2 \rfloor$. And for each α, we perform the test for $\beta = \alpha + 1, \alpha + 2, \ldots \lfloor (k-1)/2 \rfloor$. Clearly, the result of the algorithm is a set of candidates

$$\bigcap_{\alpha=1}^{\lfloor (k-3)/2 \rfloor} \left(\bigcap_{\beta=\alpha+1}^{\lfloor (k-1)/2 \rfloor} C_{[\alpha,\beta]} \right).$$

What we have covered until now addresses the detection of some subgraphs whose presence have a large effect on the metric – the basic symptom of a wormhole. Condition 2 in our definition of wormholes requires that each side of a wormhole have a size τ. We now describe how to check for this threshold. For this, we make use of the algorithm in [51] that finds the maximal complete bipartite subgraphs in any graph. Note that this entire phase can be ignored for $\tau = 1$.

Algorithm: Test for τ Partitions. We take connected components of the subgraphs induced by the nodes detected as wormhole candidates after the connectivity metric test above. Let C be one such connected subgraph.

On the subgraph C, we apply the algorithm of [51]. Let B be the set of maximal complete bipartite subgraphs generated by the algorithm. We write as a pair (W_0, W_1) an element in B, where W_0 and W_1 are the two partitions of the bipartite graph.

On each such bipartite subgraph, we perform the following test. We consider a neighboring subgraph N that consists of nodes that are at a distance at most $\lfloor (k-1)/2 \rfloor$ from all nodes in $W = W_0 \cup W_1$, but not the nodes in W itself. Let $N_0, N_1, \ldots$ be the connected components of N.

For any edge $(a, b) \in W_0 \times W_1$, if nodes a and b are neighbors to nodes of N, we check that these are in different components of N. For a graph that satisfies this condition, we check that $|W_0|, |W_1| \geq \tau$. If there is a complete bipartite subgraph that satisfies all these conditions, we have detected a wormhole $W = W_0 \cup W_1$.

Removal of Wormholes. One of the goals of detecting a wormhole is to be able to nullify it unobtrusively. We would like to retain the wireless nodes in action (thus keeping the sensing or computational capabilities of the nodes), but eliminate the high volume of traffic passing through the wormhole link that creates the wormhole effect. We do this by removing the edges $W_0 \times W_1$ in the bipartite graph.

Test for network connectivity. Once a wormhole has been detected and removed,

we flood from any one node in it and ensure that the flood reaches all other nodes. This is to guarantee that the network remains connected as required by our definition.

Provable guarantee. Now we are ready to show our main result. The $[\alpha, \beta]$-ring connectivity test is guaranteed to label all nodes in a real wormhole, but may label some legal nodes incorrectly. Together with γ-partition test, the removal and the connectivity test, the false positives are removed so our detection precisely identifies a wormhole in our definition.

Theorem 26 *Any (k, τ) wormhole $W = W_0 \cup W_1$ is detected by our test. And, our detection is surely a (k, τ) wormhole.*

Proof 27 *To show the first claim that our test is effective, we simply need to show that in each of the succession of tests, a real wormhole set (W_0, W_1) is not eliminated. First, (W_0, W_1) is by definition a maximal bipartite graph. Therefore, it will be one of the graphs detected by [51].*

Next we need to show that if $(a, b) \in W_0 \times W_1$, and a and b are neighbors to the neighbor set N, they are neighbors to different connected components of N. Suppose to the contrary that they are neighbors to the same connected component. Then there is a node $c \in N$ that is at a distance at most $(k-1)/2$ from both a and b. Thus, there is a path of length $k-1$ from a to b not passing through W. This contradicts the definition of a (k, τ) wormhole.

Finally, by definition, $|W_0|, |W_1| \geq \tau$. Thus every legitimate wormhole is detected by the test.

Now we show that our detections follow the wormhole definition. It is clear that our detection generates a bipartite graph (W_0, W_1) satisfying that each side has at least τ nodes. By the test of τ-partition, we see that without edges in the bipartite graph the nodes in W_0 and W_1 can only be connected by paths of length at least k. By the wormhole removal and connectivity test, the removal of the edges in the bipartite subgraph does not disconnect the network. Thus the detected structure precisely follows the definition of a wormhole.

Scalability and Communication Costs. The detection method is naturally local and distributed. It is local in the sense that communication distances are bounded

by a known parameter, and completely independent of the size of the global network. Each node only uses the connectivity information of the nodes within its β neighborhood, whose size just depends on the average network degree and not on any other property of the network. This makes the algorithm scalable to networks of any size.

For the test for τ sized partition, we aggregate the data about the set C and the adjoining components of N to a single node, and conduct the computation at that node. The algorithm from [51] can be computation intensive in a dense network, since its cost is exponential in degree. But note that we do this only at a few small neighborhoods we consider very likely to contain a wormhole. The overall cost for the network is therefore typically not large. Also, this step can be ignored for $\tau = 1$, which is the value we use in simulations and get very good results.

6.4.5 Discussions on Parameters

As shown in the previous section, our $[\alpha, \beta]$-ring connectivity test algorithm surely labels the nodes in a wormhole set. If we use the τ-partition test and the wormhole removal and connectivity test we precisely identify a wormhole. It is nice to have such theoretical guarantee but in practice one suggestion is to use the $[\alpha, \beta]$-ring connectivity test only, for the reason of simplicity and low communication requirement. In this way we do not lose any detection power but may identify some false alarms. In this section we discuss a few interesting cases and in particular how the parameters may influence the performance of the algorithm.

Effect of k. The user supplied parameter k essentially determines the sensitivity of the algorithm. A smaller value of k makes the algorithm more sensitive. It can detect smaller wormholes, but introduces a greater chance of false positives. A larger value of k may miss some smaller wormholes, but provides more reliable detection of the longer wormholes. Longer wormholes are more dangerous, since they introduce larger distortion to the graph metric and attract more traffic. Thus, in a sense the algorithm's accuracy automatically scales with the effect of the wormhole, or the danger posed by it.

The Influence of Parameters α, β. Recall that in our detection algorithm there are parameters α, β satisfying $k > 2\beta$, $\beta > \alpha \geq 1$. α is the size of the neighborhood

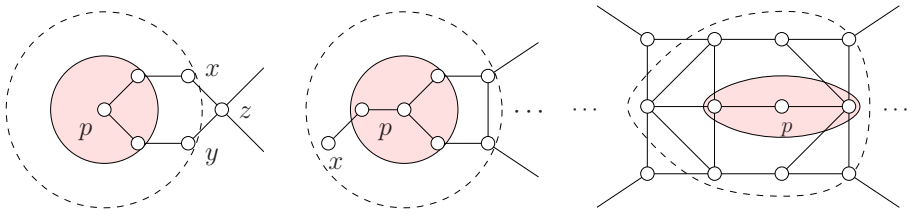


Figure 33: The α-ball is shown as the shaded region and the nodes within β-ball are within the dashed cycle. (i) If we take $\alpha = 1$, $\beta = 2$, p will be identified as a candidate since $x, y \in N_{[1,2]}(p)$ are not directly connected. But if we use $\beta = 2$, $N_{[1,2]}(p)$ has three nodes x, y, z and is connected. This way the false alarm for p is removed. (ii) p has a dangling path of length 2. For $\alpha = 1, \beta = 2$, the dangling node x is not connected with other nodes in the ring. Increasing α to be 2 will remove such dangling paths. (iii) Consider a bridge of 3 hops wide as shown in the figure. Consider a test at p with $\alpha = 1$, $\beta = 2$. The nodes in the ring are connected and thus p is not a candidate in this test. But if we increase $\alpha = 2$, $\beta = 3$, the entire bridge will be removed and the nodes in the ring will be disconnected. Thus large α will not necessarily reduce the number of false positives.

around p to be removed. α is at least one. β must be at least one greater than α to allow a non-empty ring between the α hop and β hop.

While clearly a sufficiently long wormhole will surely be detected for many different combinations of these parameters, an intelligent choice of parameters can lead to fewer false alarms. Our final algorithm tries different sets of parameters and take the intersection of their candidate sets. Notice that our sufficiency proof guarantees that any real wormhole nodes will definitely pass all such tests so we will not miss any real wormholes. We discuss the influence of the parameters in the following.

When β is increased, the $[\alpha, \beta]$-ring has more nodes in it. For an example, take a look at Figure 34 (i). If $\alpha = 1, \beta = 2$, the ring has two nodes that are not connected. But if we increase β to be 3, the ring has three nodes in one connected component. It is also clear that the newly included nodes are always connected to the nodes already in the ring, so there will not be any newly connected emerged components in the ring. Increasing β will always reduce the number of false alarms.

The issue is that β cannot be increased arbitrarily due to upper bound of k and higher communication/computation cost.

The parameter α works in an interesting way regarding false alarms. First, when α is small, there can be many false positives in a network that is not well connected. Take a look at Figure 34 (ii). In particular, when there are small 'dangling' nodes, these nodes may lead to identifications of some false positives. But increasing α can enclose all these dangling nodes inside the α ball and thus remove them. For a 'dangling' component with 'depth' of ℓ, using an $\alpha \geq \ell$ will include all dangling nodes inside the α ball and thus eliminate the false positives created this way. On the other hand, making α too big may remove a 'bridge' in the network and thus create falsely identified candidates. Take a look at Figure 34 (iii). A small α does not disconnect the bridge but a large α can fully remove the bridge and report p as a candidate (false alarm).

Eliminate False Alarms with τ. So far in our discussion we focused on the length of a wormhole, denoted by the parameter k, as the minimum hop distance between nodes in W_0 and W_1 once the wormhole edges are removed. Another parameter in a wormhole definition is the size of W_0 and W_1. A wormhole antenna takes all the signal it hears and broadcasts to the other antenna. Thus all the nodes within direct communication range of a wormhole antenna will be affected by the attack. In a case when the node density has a lower bound τ (i.e., an antenna placed at any location can hear from at least τ nodes), then it is clear that $|W_0|, |W_1| \geq \tau$. We can also use this property to eliminate the false alarms. This avoids identifying isolated edges that act as connection between otherwise distant parts of a sparse network.

6.4.6 Multiple Wormhole Sets

When the network has multiple wormhole sets, our $[\alpha, \beta]$-ring-connectivity test can also detect these wormhole sets if they are far away (and thus 'independently' alter the network connectivity) or too close (thus removing the α neighborhood will remove all related wormhole edges).

Theorem 28 *When there are multiple k-wormhole sets, the nodes in the wormhole sets are surely picked up by our $[\alpha, \beta]$-ring-connectivity test, given that $k > 2\beta$,*

$\beta > \alpha \geq 1$, *and* either *one of the following conditions holds for* each *pair of wormhole sets* W, W'*:*

1. *The minimum hop distance between any two nodes that belong to different wormhole sets* W, W' *is greater than* $\beta + 1$.
2. *There are two nodes* $p \in W$, $p' \in W'$ *such that* p, p' *are within* $\alpha - 1$ *hops of each other.*

Proof 29 *In the first case, the two wormhole sets* W, W' *are far apart. Thus when we run the test at a node* $p \in W$*, all edges involved are within* β *hops from* p*. That means the existence of* W' *does not affect the test we run around* p*. Thus all nodes in* W *are still identified.*

In the second case, the two wormhole sets are 'close'. Basically there is a node $p \in W$ *and* $p' \in W'$*,* p*,* p' *are within* $\alpha - 1$ *hops of each other. Now if we do a test on any node* $x \in W$*, then all the nodes in* W' *are within* α *hops from each other. Thus the wormhole tests running on* p *will remove the wormhole edges of both* W *and* W'*. Thus the test will also turn out to label* p *as a candidate, since there cannot be any edges of* W' *that affect the results (i.e., decrease the number of connected components).*

The test for size τ of wormholes can be carried out as usual. In the second case, the detection of the complete bipartite graphs can help in identifying the fact that there are in fact two wormholes.

The case when our detection algorithm fails with multiple wormholes is when the multiple wormholes are carefully placed at a proper distance from each other such that they interfere. An example is shown in Figure 34. The removal of the α-ball around a node p does not leave the nodes in the ring in different connected components — as they can possibly be connected through another wormhole. In fact, in this case any single wormhole itself does not actually follow our Definition 21. The two wormholes interfere with each other such that the removal of edges from only *one* of them does not leave the nodes with long paths in the network. However, if the wormholes are long, that is, if k is large compared to the separation between W_0 and W_0', then removing a sufficiently large α-ball disconnects both wormholes, and detects a candidate. This property can be used to detect

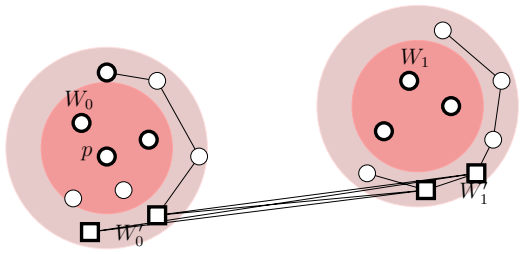


Figure 34: There are two wormhole attacks (W_0, W_1) and (W_0', W_1'), one on top of the other. Nodes in the second set are shown as squares. The edges after the removal of $B_\alpha(p)$ (darkly shaded region) are shown. The second wormhole connects what would have been the two components of $N_{[\alpha,\beta]}(p)$, which now appears to have one component and is not detected in connectivity tests.

potential threats of multiple wormholes though it does not identify the wormholes precisely.

6.5 Simulations

6.5.1 Simulation Setup

We evaluated our algorithm using extensive simulations under various conditions, including different node distributions and density, radio models, positions of wormholes, and different test parameters.

Node Distribution. Two node deployment models are used in our simulations: grid with perturbation and random placement. In the model of grid with perturbation, the wireless nodes are placed on an $m \times n$ grid, each cell in the grid is a square with edge length d. Then each node with coordinate (x, y) will be perturbed around its initial position with displacement parameter p: its coordinate will be uniformly randomly drawn from the region $[x - pd, x + pd] \times [y - pd, y + pd]$. By varying p, we can get various node placements with different levels of regularities. In random placement, each node is assigned a coordinate uniformly randomly drawn from the network field. Random distribution typically has more irregularity than the perturbed grid distribution. In our simulations, we also extend both types of node placement strategies to three dimensional networks.

Radio Models. To determine links between nodes, we adopt both unit disk graph (UDG) and quasi-UDG settings. In the UDG setting, each pair of nodes u and v has an undirected link between them if and only if their distance is no greater than R, where R is the communication radius. Quasi-UDG adopts a more practical link generation model: each pair of nodes u and v will have a link if their distance is no greater than r. Besides, they will have a link with probability q if their distance is within $[r, R]$. In our simulation, we set $r = 0$ for quasi-UDG. By adjusting the parameters in UDG and quasi-UDG, we vary the average degree in the network from 6 to 20.

Wormhole Placement. The location of wormholes is a crucial factor in wormhole detection. The length of a wormhole is important: a wormhole is significant only when it is reasonably long. In previous work [44], the placement of wormhole antennas turns out to be another important factor: for the antennas being placed near the network boundary or sparse regions certain algorithms may experience deteriorating performance. Previous schemes did not tackle the case of multiple wormholes. Multiple wormholes detection is influenced by their relative positions. In our simulations, we vary the length of wormholes, put the antennas at different positions of the network, and change the relative positions of two or more wormholes.

6.5.2 False Positive Rates By Ring Connectivity Tests

Our ring connectivity test guarantees to detect true wormhole nodes, which means that there are no false negatives. Our method may run for multiple rounds using different α, β parameters. For each round, we only test the candidates that have passed all previous rounds. We evaluate the number of false positive nodes in each round, by varying different setup parameters: node distribution, density, α, β, and radio models (UDG or quasi-UDG).

Influence of Node Distributions and Density. Figure 35 shows that in general there are much fewer false positives for networks with perturbed grid distribution than networks of uniform random distribution, since a network of perturbed grid is more regular. Second, with the same node deployment method and the same

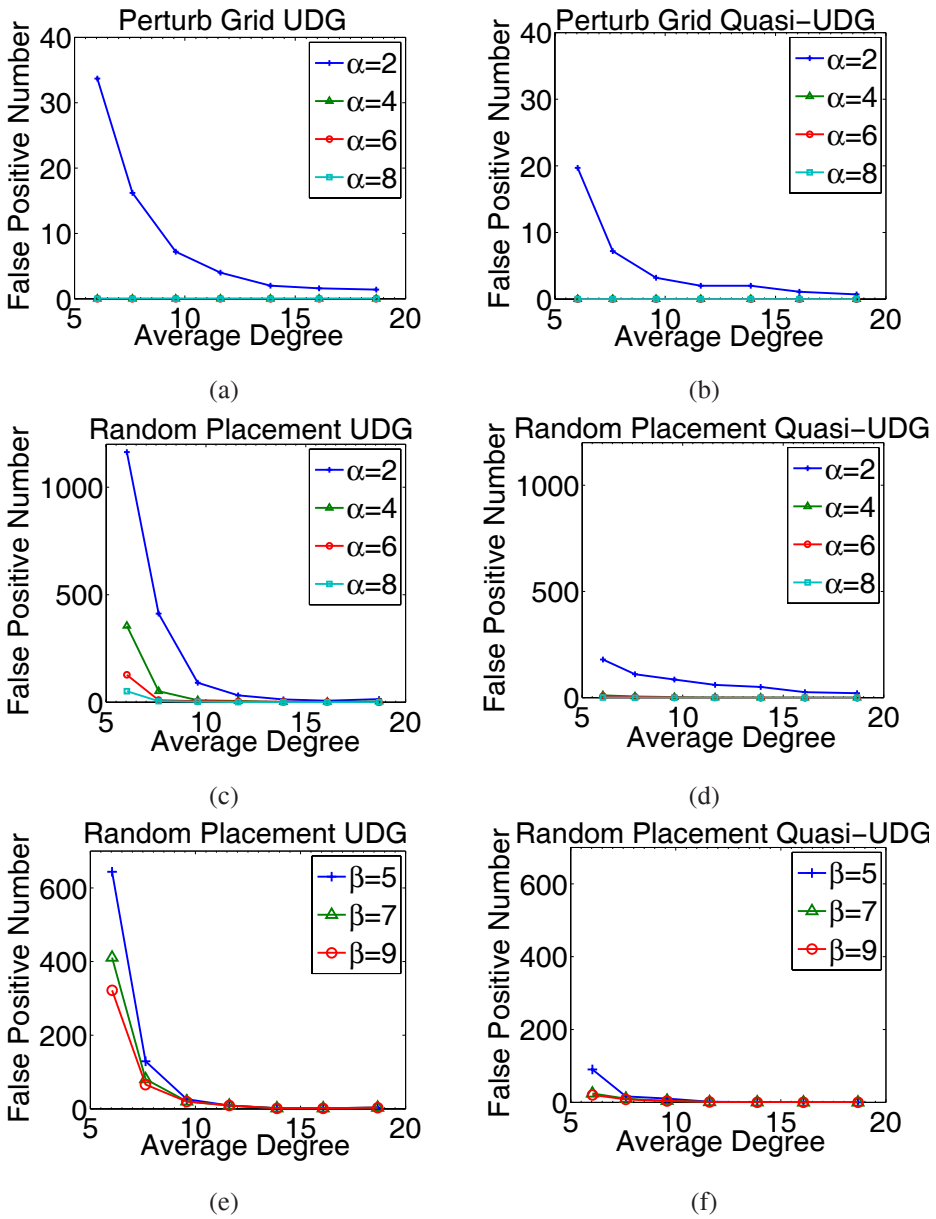


Figure 35: The number of false positive nodes on a network with 5000 nodes. In the first four figures, we vary α to be $2, 4, 6, 8$ and take $\beta = \alpha + 2$. In the last two figures, we take $\alpha = 3$ and take β as $5, 7, 9$ respectively. (a) Perturbed grid with UDG model, perturbation ratio $p = 0.4$. (b) Perturbed grid with quasi-UDG model, $p = 0.4$. quasi-UDG radius $r = 0$,$q = 0.5$. (c) Random distribution with UDG model. (d) Random distribution with quasi-UDG model, $r = 0$,$q = 0.5$. (e) Random distribution with UDG model.(f) Random distribution with quasi-UDG model, $r = 0$,$q = 0.5$.

average degree, our detection methods have fewer false positive nodes on quasi-UDGs than UDGs. This observation is a bit counter-intuitive but confirms that our method does not rely on the communication models. In particular, on quasi-UDGs previous methods typically perform worse, especially for location based techniques. Figure 35 also shows that as the average degree grows, the number of false positive nodes drops very fast.

Effect of α and β. From Figure 35 shows that the increase of α and β reduces the number of false positive nodes. This resonates with our design idea in which we test the (α, β) parameters in lexicographic order, gradually removing false positives. Notice that we take the candidates that pass all tests, the number of false positives is very small.

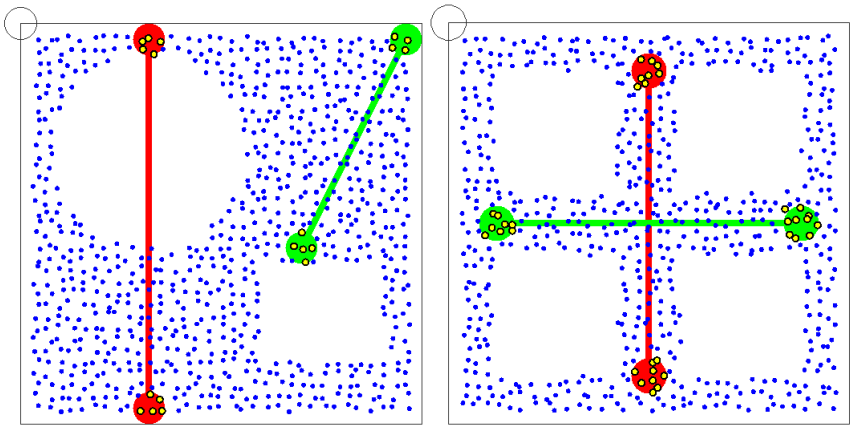


Figure 36: Example of wormhole placement, Network size is 1000, average degree is 6, $\alpha = 1$, $\beta = 3$.

Wormhole Placement. Certain schemes proposed earlier are extremely sensitive to the positions of wormholes. For example, the WormCircle method [44] divides the wormhole positions into different cases and under certain cases the detection rate is high, while in other cases, e.g. placing wormhole antennas on network boundaries, the detection rate is much lower. Our method is not influenced much by the wormhole placement. We show different scenarios in Figure 36. It shows that we can place wormhole antennas near the network outer boundary, or near holes, the detection is always effective and accurate.

3D Wireless Networks. A wireless Network may be deployed in 3D space, say,

under water or in a multi-floor building. Most previous results would fail in 3D networks. The method that uses forbidden substructure in [105] can be extended to 3D, but would need very high node densities and detailed radio models. The WormCircle method [44] strictly assumes the underlying geometry to be two dimensional, and does not generalize to 3D at all.

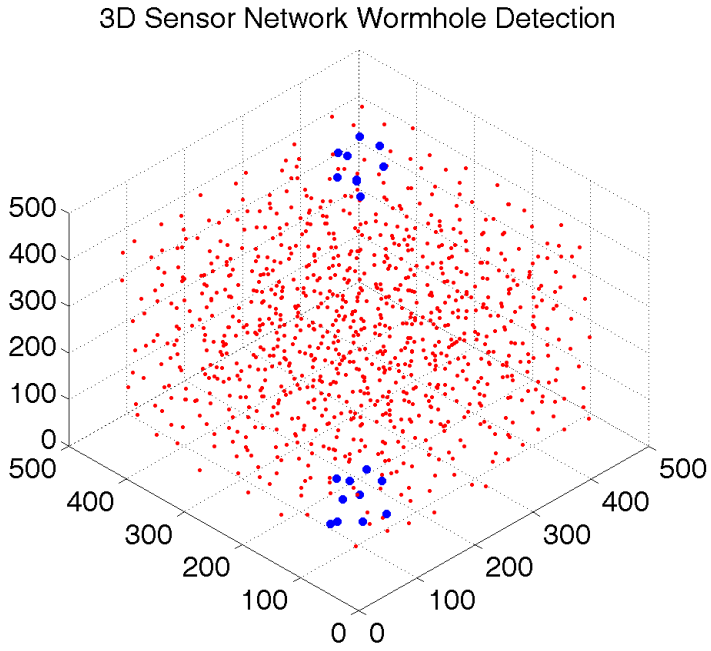


Figure 37: Wormhole detection in a 3D network. Network topology is formed by using a 3D grid with perturbation. The network has 1000 nodes. We use $\alpha = 3$, $\beta = 5$. The wormhole transceivers are located near a pair of diagonal corners and the nodes affected are accurately detected as highlighted in the figure.

Our method operates purely in terms of graph connectivity, without any dependency on the dimension of the network. Therefore it works naturally in 3D. Figure 37 shows an example of wormhole detection on 3-dimensional wireless network. The behavior of our method in a 3D network is similar to that on a 2D network.

6.5.3 Communication Cost

Our detection mechanism requires all nodes to participate initially, and the suspicious nodes participate more rounds using different parameters. For a test using parameter α, β, a node will need to gather the connectivity information for all nodes within β hops. While the nodes participating in more detection rounds will introduce higher communication cost, the number of participants is fairly small compared to the total number of nodes. Figure 38 shows the communication cost in terms of packets transmitted for each node on average for the entire detection process. There are a few interesting observations. First, the communication cost is smaller for networks built by a perturbed grid model than networks of randomly distributed nodes. This is because there are fewer false positives in a perturbed grid. Second, when the network density increases, obviously it would incur a higher cost to collect the connectivity in local neighborhood as there are more nodes. However, when the average degree increases, fewer nodes are marked suspicious in the first round, which leads to a decrease of communication cost in later rounds. The combination of the two factors shows the interesting trend of first increasing and then decreasing for the case of a network of random node distribution.

6.5.4 Multiple Wormholes

When multiple wormholes are placed simultaneously, they may interfere with each other, making the detection harder. The interference of two wormholes depends on the relative positions of their antennas: as long as there exists at least one antenna which is far away from other antennas, those two wormholes will not affect each other in terms of detection. Figure 39 shows three scenarios. From top to bottom, in the first one the antennas of the two wormholes are far away from each other. In the middle, several wormholes share one antenna and the other antennas are far from each other. In both cases the wormhole nodes are well recognized. The last case is an interesting example where the second wormhole reduces the length of a previous existing wormhole. The wormhole nodes are detected for smaller values of α, β (left). But they are not detected when we use a larger set of α, β parameters (right).

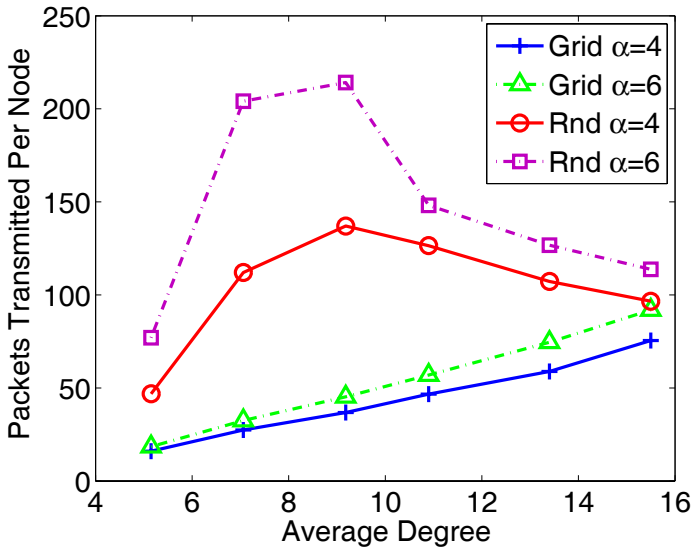


Figure 38: Communication cost in terms of packets transmitted. Network has 5000 nodes, $\beta = \alpha + 2$. Grid is perturbed grid with UDG, perturbation ratio $p = 0.4$. Rnd is node random placement with UDG.

6.5.5 Comparison with Wormcircle

We compared the performance of our method with the Wormcircle algorithm [44].This algorithm is based on the idea that presence of a wormhole changes the geometry and topology of the ring of nodes at k-hops from a root node. Without wormhole, the k-hop ring should have the connectivity as a ring. If one antenna of the wormhole is less than k hops away from the root, then the set of k hop nodes will appear as two rings. The cut locus method of [157] is used to determine the topology of the k-hop band. The paper presents two different algorithms based on this principle.

The *basic Wormcircle* scheme starts with a designated root node in the network, and computes the breadth-first tree from this node. Next, it considers the connected components of nodes at k hops for each k. In a Euclidean or similar domain, each component resembles a circle. However, the connected component induced by the wormhole will have a smaller radius. In particular, the main connected component is expected to have a circumference of $2\pi k$, where the distant wormhole component will have a much smaller circumference. By comparing the

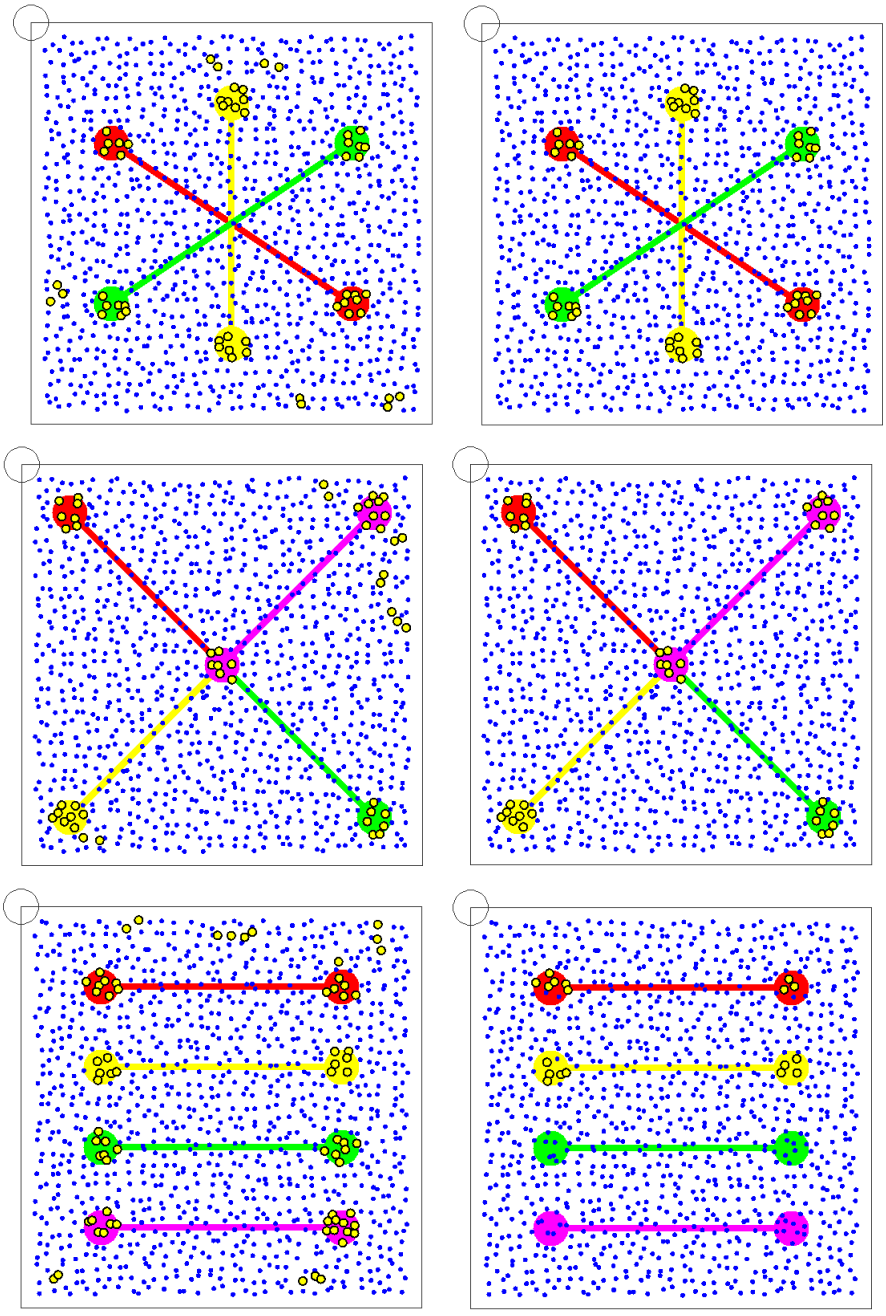


Figure 39: Multiple Wormholes. Left: $\alpha = 1$, $\beta = 3$; Right: $\alpha = 2$, $\beta = 4$.

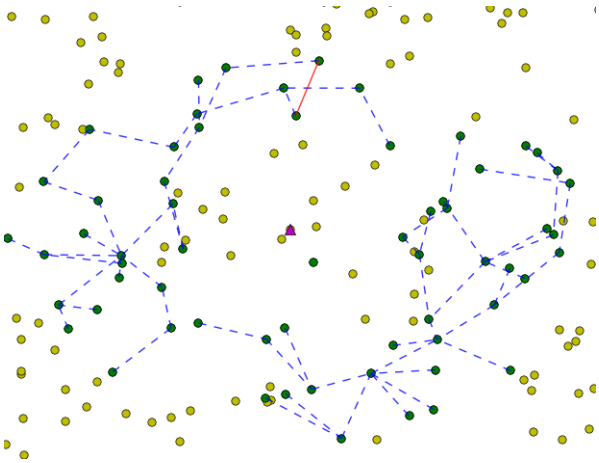


Figure 40: A wormhole detected by localized wormcircle at a regular node, in a quasi unit disk graph. The 3-hop ring has two components. Edges in dashed blue show the breadth first trees in the two cases. The red solid edge is detected as a cut edge, implying a long cycle in one of the trees and a false detection.

circumference to $2\pi k$, a wormhole can be detected.

The *localized wormcircle* scheme takes a more topological approach. It computes a shallow breadth-first tree around every node and considers the k-hop ring. If the k-hop ring has two components and at least one of them resembles a circle, a wormhole is said to be detected. The circumference of the circle is not considered. These methods depend heavily on the geometry of the network resembling a Euclidean plane. On graphs that are more general than that, they can fail frequently. The localized wormhole algorithm, while in some ways similar to ours, is still tied to the Euclidean geometry, and expects a circle as in that case. Our simulations show that if a network has significant *holes* or is not a unit disk graph, both these methods perform poorly.

Figure 40 shows a network constructed as a quasi unit disk graph. In the figure, the edge in red is detected as a cut edge. that is, it connects leaves of the same breadth first tree, such that the leaves are far apart within the tree itself. This method is used to confirm the presence of a circle. As seen in this example, in networks that are less geometric, this strategy can fail by detecting a cycle that does not resemble a circle at all. In our simulations and in [44] the localized method performs better

Avg Degree	False negative	False positive
6.3	60%	00%
7.7	50%	10%
9.0	50%	20%
10.3	40%	20%
11.5	30%	30%
12.8	30%	20%
14.1	30%	30%
15.3	20%	20%
16.8	20%	30%
18.0	20%	30%

Table 7: Wormcircle performance over 20 networks in each degree range. The first column shows the average degree of 20 networks. The false negatives show the percentage of cases that the algorithm failed to detect an actual wormhole, while false positives show the percentage of networks that did not have any wormhole but was erroneously detected to have one.

than basic wormhole. Therefore we only present the results for localized wormhole in the following.

Table 7 shows the performance of localized wormcircle. We created a wormhole with end points 20 units apart in a region of diameter 40 units. Then we added nodes randomly and created networks in quasi unit disk model. We selected networks of different densities, and obtained 20 networks in each range. It is seen that wormcircle makes substantial errors in detecting wormhole. In comparison, our method detected presence or absence of wormhole correctly in all these cases.

In network structures with wormholes placed next to holes such as those in Figure 36, we find that wormcircle performs even more poorly. In these cases, the hole breaks the circular structure of the wormcircle. Thus it fails to detect the actual wormhole in all cases, though sometimes it detects wormhole at incorrect locations. Whereas our method is not affected in any significant way by the presence of holes.

6.5.6 Network Dynamics

In practice, wireless links may experience various types of dynamics, both temporal and spatial. Here we consider the setting that links fail randomly with a probability p. In our method, all nodes participate in the detection of wormhole region, but they may not enter the detection phase at the same time. Therefore, each node may have different view of the network topology due to potential dynamic link failures. When a single transmission fails, we may re-transmit and give up after a maximum K number of trials. In Table 8, we can see that when link failure rate is relatively low, our method still works fine on the tested networks. As failure rate grows, for random placement with UDG, the false positive node number increases dramatically, which makes our identification of wormhole infeasible. This can be understood since the network topology varies significantly and different nodes have very different views.

	0%	1%	5%	10%	15%	20%
Grid	0	0.03	0.05	0.13	0.27	0.46
Q-Grid	0.21	0.24	0.46	0.82	1.40	2.36
UDG	3.40	4.32	20.62	41.67	91.50	180.6
Q-UDG	0.11	0.20	0.32	0.37	5.33	27.17

Table 8: The average number of false positive nodes under random link failure. The network has 2000 nodes and average degree is 8. $\alpha = 5$, $\beta = 7$. The maximum number of retransmissions is 30. Grid is a network with perturbed grid distribution with UDG model, in which the perturbation ratio $p = 0.4$. Q-Grid is a network with perturbed grid distribution with quasi-UDG model, $p = 0.4$. quasi-UDG model uses $r = 0$, $q = 0.5$. UDG is a networrk of node random placement with UDG model. Q-UDG is a network with node random placement with quasi-UDG model, $r = 0$, $q = 0.5$.

6.6 Discussion

6.6.1 Malicious Nodes

Our connectivity tests detect the bipartite subgraph introduced by the presence of wormholes. Notice that such connectivity change does not need the help of any compromised nodes. In the case when some nodes are compromised, a malicious node can choose not to cooperate with the local connectivity tests or report incorrect connectivity information. For example, the nodes that are within communication range of the wormhole antennas can choose not to report the edges faked by the wormhole link. However, not reporting the presence of the link faked by the wormhole attack would be equivalent to not imposing the attack to the network. That is, for the wormhole attack to truly alter the network connectivity and for such connectivity change to be observed and used by the honest nodes – to make any real damage — then the local connectivity tests can be executed to examine such possibilities.

However, a malicious node may impose sybil attacks and fake many node identities or even create phantom subgraphs. This will surely add to the detection difficulty. For example, a node x within the $[\alpha, \beta]$ ring of a node p may wrongly

claim itself to be identical to a node near the other side of the wormhole antenna, thus causing the detection algorithm to fail. Since a sybil attack may create all kinds of incorrect graph structures we remark that the wormhole attack together with carefully positioned sybil attack may change the network topology in such a way that the wormhole links do not follow our definitions. Thus we defer the discussion of such combined, more sophisticated attacks to be the future work.

6.6.2 Open Problems

In this work we examine the network connectivity and propose a local, distributed method to detect suspicious nodes. The method compares favorably with existing connectivity based methods. We believe this strategy can be improved further. For example, the multiple wormholes detection possibly can be improved by a more careful execution of connectivity and bipartite graphs test. The issue of eliminating false positives also remains open for closer investigation.

Chapter 7

Space Filling by Aperiodic Dense Curve

7.1 Introduction

We consider a sensor network that densely covers a planar domain with possibly multiple network holes. Here we develop algorithms to linearize the network, i.e., 'covering' the sensor network by a single path. By enforcing a linear order of the sensor nodes one can carry serial logical definitions and serial operations on both the sensor nodes and the sensor data. We list a number of such applications in the following.

7.1.1 Serial data fusion

When a signal is spread over an area larger than the coverage range of a single sensor, we will need to use multiple sensors to collaboratively detect the distributed signal. One type of data fusion mechanisms, called *serial fusion* [14,154], combines sensor observations in a linear fashion to derive hypothesis. A state is maintained and passed on from sensor to sensor along a serial path, incorporating new observation at each step. This is in contrast with *parallel fusion* mechanism in which sensors independently process their data and pass the output to a centralized fusion center. There are pros and cons for serial fusion v.s. parallel fusion respectively. One particular advantage of serial fusion is that the fusion process can be stopped as

long as there is enough evidence to support or reject the hypothesis, while in parallel fusion all data will be sent to the fusion center nevertheless. The implementation of the serial data fusion in a distributed network requires a path that visits all the nodes in a linear order [119].

7.1.2 Motion planning of data mules

Collecting data from sensor networks to a static data sink often suffers from communication bottleneck near the sink. One way to address this is to use a mobile sink, or called a data mule, implemented by a mobile device touring around the network to collect data through direct communication with a sensor in close proximity. Besides collecting sensor data, a data mule can also be helpful for sensor network maintenance such as battery recharge, beacon-based localization [8,89], etc. A data mule moves along a path. Planning the motion of a data mule requires a path that visits the nodes in the network with minimum duplicate visits. When there are multiple data mules in the network, a flexible set of paths that can be used by the data mules with minimum coordination and minimum interference (e.g duplicate visits by different mules) will be handy.

7.1.3 Sensor node indexing

Another application of representing a sensor network by a linear order is for indexing sensor nodes or sensor data [94]. A number of indexing schemes for multi-dimensional input first take a space filling curve to 'linearize' the input and then apply standard 1D indexing mechanisms.

In the following we first review previous work of linearizing a two dimensional continuous domain or a discrete two dimensional network, before we present our ideas.

7.1.4 Related Work

Space filling curves. In the continuous setting, various space filling curves have been defined for a square region [135]. The narrow definition of space filling curve, in mathematical analysis, refers to a curve whose range contains the entire

2-dimensional unit square (or more generally an N-dimensional hypercube). Space filling curves were initially discovered by Giuseppe Peano and are also called Peano curves. These curves are often recursively constructed. See Figure 41 for an instance of the Hilbert curve. The basic recursive structure is to replace a line segment by a zig-zag pattern. In a recursive step, each segment is replaced by a scaled and rotated version of this pattern. The larger number of recursions used, the denser the curve becomes. Mathematically every point of the unit square is on the curve, given an infinite number of recursions. For a discrete set of points it suffices to take a sufficiently high number of recursions to generate a linear order of the points. Space filling curves in this narrow definition only apply to 2-dimensional (or N-dimensional) unit squares (hyper-cubes). When the domain is irregular and/or has holes the space filling curve will be chopped into many disconnected pieces. Very little work is known about extending the space filling curves to other shapes. The only work known is a heuristic algorithm [76] with a modified Hilbert curve for an ellipse.

Hamiltonian paths. In a discrete setting such as a graph, a natural analog of a space filling curve is a Hamiltonian cycle or a Hamiltonian path, i.e., a cycle or a path that visits each vertex once and only once. Only a subset of graphs has a Hamiltonian path and determining whether a Hamiltonian path or a Hamiltonian cycle exists in a given graph (whether directed or undirected) is NP-complete, even in restricted families such as planar graphs [61].

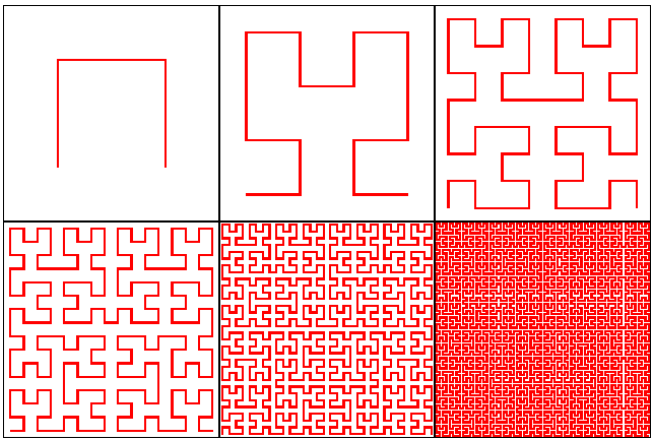


Figure 41: The Hilbert curve (source: Wikipedia).

Traveling salesman tour. When a metric is defined between any two nodes, the traveling salesman problem (TSP) asks for the shortest tour that visits each node once and only once. In our setting the distance between two nodes can be either the graph distance or the Euclidean distance. The latter becomes the Euclidean TSP.

Both the metric TSP and the special case of Euclidean TSP are NP-complete. For the metric TSP, the heuristic of using the Euler tour on the minimum spanning tree gives a two-approximation. With some additional tricks, the Christofides algorithm [32] gives a $3/2$ approximation. For the Euclidean TSP, polynomial approximation schemes (PTAS) are known [6, 110] to find a $(1+\varepsilon)$ approximate solution for any $\varepsilon > 0$. Such algorithms are mostly of theoretical interest. When multiple tours are allowed (e.g., multiple data mules), the problem of minimizing the total travel distance collectively done by all tours becomes the multiple traveling salesman problem (mTSP), which is also NP-complete and does not have any efficient approximation algorithms [12]. Existing solutions for mule planning are all heuristic schemes [71, 77, 101, 144, 153].

Random walk. A practically appealing solution for visiting nodes in a network is by random walk. The downside is that we encounter the coupon collector problem. Initially a random walk visits a new node with high probability. After a random walk has visited a large fraction of nodes, it is highly likely that the next random node encountered has been visited before. Thus it takes a long time to aimlessly walk in the network and hope to find the last few unvisited nodes. Theoretically for a random walk to cover a grid-like network, the number of steps is quadratic in the size of the network [104]. For a random walk of linear number of steps, there are a lot of duplicate visits as well as a large number of nodes unvisited at all. In the case of multiple random walks, since there is little coordination between the random walks, they may visit the same nodes and duplicate their efforts.

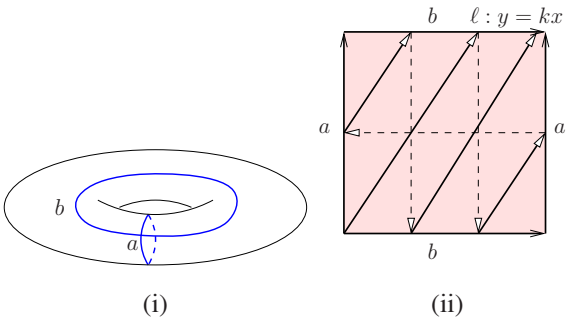


Figure 42: (i) A torus cut open along two curves a, b. (ii) The flattened torus. The line $\ell : y = kx$ is shown on the flattened torus (the top and bottom edges are the cut b, the left and right edges are the cut a). Since the top edge and bottom edge are actually the same, the line will go through the torus as shown by the parallel lines. It will not intersect itself and can be shown to be arbitrarily close to any point on the torus.

A major problem with all the above constructions is that the curve found does not have adaptive density. A space filling curve has a fixed density, determined by the threshold of the recursion. Hamiltonian paths and TSP will generate fixed length paths. In sensor network applications such as serial fusion and data mule planning, the length of a path may be restricted by travel budget or required fusion delay. If we start with a high density curve, we spend a lot of time visiting nodes in one region of the network before we ever get information from another region. Instead, we may want to adopt a visiting scheme such that we quickly tour around the network coarsely, get a rough idea of the sensor data and gradually refine the density when more travel budget is available or a higher delay is allowed. Our construction is one of this type.

7.1.5 Our Contribution

Our main contribution in this paper is to propose a scheme to generate a curve that (i) densely cover any geometric domain with possibly holes; (ii) have a coverage density proportional to its length. To understand the main idea, we first consider a torus. See Figure 42. We cut a torus open with two cuts a, b, and flatten it as a square in the plane with the top edge identified as the bottom edge and the left edge identified as the right edge.

We will consider the universal covering space by packing an infinite number of translated copies of the torus to cover the entire two dimensional plane, with the origin at the bottom left corner of one such copy. Now take a straight line ℓ with a slope k being an irrational number. Mapping back to the original torus, the line becomes a curve that spirals around the torus for infinitely long and never repeats itself. Figure 42 (ii) shows the curve on the torus. We could prove that the curve has no self-intersections and the curve is dense, i.e., any point p of the torus is arbitrarily close to the curve.

With the basic construction for a torus, we will generalize it to any planar domain with holes. Specifically, for a simple domain with no holes, we will first map it one-to-one to a unit square, and then flip the square along the top edge and the right edge to get four copies, creating a torus. Then we find the dense curve on the torus. Since any point in the original domain is mapped to four copies on the

torus, the curve we find will visit any point for at most four times. The property of being dense still holds. For a domain with holes, we will first double cover it, i.e., creating two copies of the network, the upstairs copy and the downstairs copy. The two copies are glued to each other along the hole boundaries to create a multi-torus, each hole being a handle. In the same way we choose one handle to flatten the torus, and the rest of the handles are mapped to very narrow 'slits'. A line with irrational slope in the covering space, when hitting a slit, bounces back. We could show that the curve will visit each point of the original domain at most twice and is provably dense.

The mapping of a general two dimensional domain to a multi-torus is handled by conformal map. Computing a conformal map for deforming the shape of a sensor network has been done by using Ricci flow to change the network curvature, in a number of prior work [73, 137, 138, 163]. We remark that the tools we use in this paper is different. Our current method is based on holomorphic differentials from Riemann surface theory. Imagine an electric field on a surface, then the equipotential lines are orthogonal to the electric field lines everywhere, the pair of electric field lines and the equipotential lines form the holomorphic 1-form. All holomorphic 1-forms on a surface form a group, which is isomorphic to the first homology group of the surface. We select a special holomorphic 1-form, such that the integration of the 1-form gives a special conformal map. Assume the network is a planar domain with multiple holes, then the conformal map transform the domain to an annulus with concentric circular slits. Two boundaries are mapped to the inner and outer circles, the other boundaries are mapped to the slits. This type of maps can not be carried out by Ricci flow method, because Ricci flow requires the target curvature at prior. But in this scenario, neither the position nor the radii of circular slits are known at the beginning. On the other hand, Ricci flow is a non-linear method in nature; whereas holomorphic differential method is a linear one, which is computationally more efficient.

The conformal map is computed for a given network field at the network initialization phase. With the map computed the dense curve can be found and followed locally by simply specifying an irrational slope. This leads to naturally decentralized computations and planning in the network that can benefit data storage and data mule collection.

7.1.6 Planning motion for data mules

The space filling curves have the property that nearby points in the two dimensional space are also nearby on the curve. For example, the Hilbert curve will first visit all the points in the southwest quadrant before it visits the points in the northwest quadrant, then the northeast quadrant, and the southeast quadrant at the last, and do so recursively. This is typically referred to as the locality property of space filling curves. In contrast, the curve we define will grow far apart initially, and the longer it is, the denser it covers the domain. See Figure 42 for an example. Thus our curve does not have the locality property but instead visits the entire domain in a sketchy manner, and gradually refines when the path gets longer. If the budget of a data mule is small, the data mule starts on the curve and simply stops early. This will provide a sampled view of the network and the network data. As more budget is available, the data mule simply prolongs its walk and will gather data with higher resolution. This suggests a progressive and adaptive scheme for data collection, dependent on the urgency of the scenarios.

Our dense curve can also be used for planning motion plans for multiple data mules. If there are m mules starting at different locations in the network, they can simply walk from their respective starting points with the same slope in the virtual coordinate space. Their trajectories will be parallel to each other and *never* intersect as long as they start at integer grid positions. This makes planning motion trajectories for multiple mules and coordinating between them easy. If any one mule fails due to unforeseeable reasons, the other mules can easily take over the task by simply prolonging their paths.

7.1.7 In-network storage and retrieval

Another application of the dense curve is for in-network storage and retrieval. One scheme for storing sensor data in the network, called double rulings, stores the sensor data along a storage curve and retrieves data along a retrieval curve. Data is retrieved when the retrieval curve intersects the storage curve. Previous double rulings schemes are only designed for networks of a regular shape, e.g., the horizontal/vertical lines [103, 147, 160], or proper circles (great circles through a stereographic mapping) [139]. When the network has holes, these curve are fragmented

by the presence of holes. Alternative repairing schemes must be used to reconnect them. We show that by using the space filling curves we can easily generate the storage and retrieval curves, for any two dimensional domain. In particular, for storage curves, we simply use the line $\ell : y = kx$ with slope k. For the retrieval curves, we use the line $\ell^* : y = x/k$, i.e., the line perpendicular to ℓ in the universal covering space. The two curves are guaranteed to intersect and by varying the slope one can get many sets of curves suitable for storing categorized data.

In the following we first present the theory of finding a dense curve in a continuous domain. The algorithmic details follow. We present simulation results and comparisons with space filling curves and random walks at the last.

7.2 Algorithms for Discrete Conformal Mapping

The discrete algorithms for computing conformal mappings for arbitrary 2D domain are explained in details. The pipeline is as follows: 1) Compute cohomology basis; 2) Compute harmonic 1-form basis; 3) Compute holomorphic 1-form basis; and 4) Compute the slit map.

Discrete exterior calculus. Similar to all prior work [73,137,138,163], the network is represented as a discrete triangular mesh $M = (V, E, F)$, with the vertex set V, the edge set E and the face set F. An oriented edge is denoted as $[v_i, v_j]$, an oriented faces is $[v_i, v_j, v_k]$. The boundary operator takes the boundary of a simplex:

$$\partial[v_i, v_j] = v_j - v_i, \partial[v_i, v_j, v_k] = [v_i, v_j] + [v_j, v_k] + [v_k, v_i].$$

A 0-form is a function defined on the vertex set $f : V \to \mathbb{R}$. A 1-form is a linear function defined on the edge set $\omega : E \to \mathbb{R}$. A 2-form is a linear function defined on the face set $\tau : E \to \mathbb{R}$. The discrete exterior differential operator is defined as

$$d\omega(\sigma) := \omega(\partial\sigma).$$

If ω is a closed form, then $d\omega = 0$.

Let $[v_i, v_j]$ is an interior edge, with two adjacent faces $[v_i, v_j, v_k]$ and $[v_j, v_i, v_l]$. Then the edge weight

$$w_{ij} := \cot \theta_{ij}^k + \cot \theta_{ji}^l,$$

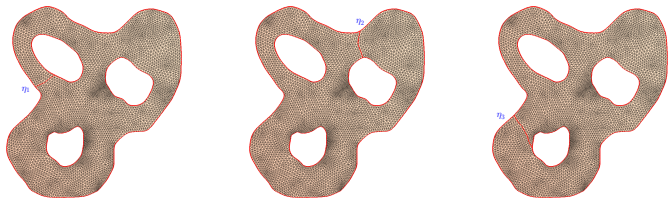


Figure 43: Compute the shortest path η_k connecting γ_0 and γ_k.

where θ_{ij}^k is the corner angle at vertex v_k in the face $[v_i, v_j, v_k]$. If $[v_i, v_j]$ is a boundary edge, adjacent to $[v_i, v_j, v_k]$ only, then $w_{ij} := \cot \theta_{ij}^k$. The discrete co-differential operator is defined as follows. Let ω be a one-form, then $\delta\omega$ is a 0-form,

$$\delta\omega(v_i) = \sum_{[v_i,v_j]\in E} w_{ij}\omega([v_i, v_j].$$

If $f : V \to \mathbb{R}$ is a harmonic function, then

$$\Delta f(v_i) = \delta df(v_i) = \sum_{[v_i,v_j]} w_{ij}(f(v_j) - f(v_i)) = 0, \forall v_i \in V.$$

Step 1: Compute cohomology basis. Suppose the boundary components of the mesh are $\partial M = \gamma_0 - \gamma_1 - \gamma_2 \cdots \gamma_n$, where γ_0 is the exterior boundary. Compute the shortest path from γ_k to γ_0, denoted as η_k as shown in figure 43. Slice the mesh M along η_k to get a mesh M_k, the path η_k on M corresponds to two boundary segments η_k^+ and η_k^- on M_k. Define a function $f_k : M_k \to \mathbb{R}$,

$$f_k(v_i) = \begin{cases} +1 & v_i \in \eta_k^+ \\ -1 & v_i \in \eta_k^- \\ 0 & v_i \notin \eta_k^+ \cup \eta_k^- \end{cases}$$

Assume $e \in \eta_k^+ \cup \eta_k^-$, then $df_k(e) = 0$. Therefore, the exact 1-form df_k on M_k in fact is a closed 1-form on the original mesh M. Let $\rho_k := df_k$ on M, then

$$\{\rho_1, \rho_2, \cdots, \rho_n\}$$

form a basis for the cohomology group $H^1(M, \mathbb{R})$.

Step 2: Compute harmonic 1-form basis. Given a closed 1-form ρ_k, we can find a function $g_k : M \to \mathbb{R}$, such that

$$\delta(\rho_k + dg_k)(v_i) = \sum_{[v_i,v_j]} w_{ij}\{\rho_k([v_i, v_j]) + (g(v_j) - g(v_i))\} = 0,$$

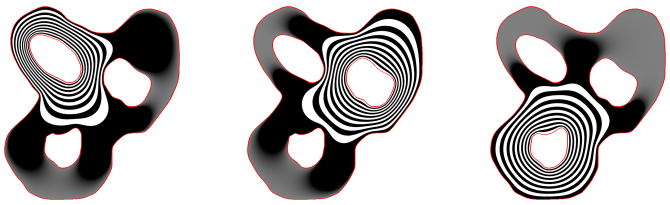


Figure 44: Closed harmonic 1-forms $\{\omega_1, \omega_2, \omega_3\}$.

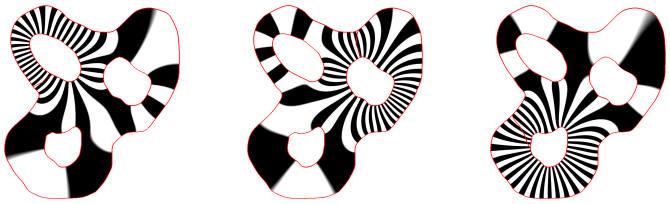


Figure 45: Exact harmonic 1-forms $\{\omega_4, \omega_5, \omega_6\}$.

for all vertex v_i in M. Then $\omega_k := \rho_k + dg_k$ is a harmonic 1-form. Then $\{\omega_1, \omega_2, \cdots, \omega_n\}$ form the basis for the harmonic 1-form basis.

Similarly, we compute n harmonic functions $f_k : V \to \mathbb{R}$, with Dirichlet boundary condition, such that

$$\begin{cases} \Delta f_k(v_i) &= 0 \quad \forall v_i \\ f_k(v_i) &= 1 \quad v_i \in \eta_k \\ f_k(v_i) &= 0 \quad v_i \in \partial M - \eta_k \end{cases}$$

Then let $\omega_{n+k} := df_k$, then $\{\omega_{n+k}\}, k = 1, 2, \cdots, n$ are exact harmonic 1-forms.

Step 3: Holomorphic 1-form basis. For each harmonic 1-form ω_i, we compute its conjugate harmonic 1-form ${}^*\omega_k$.

$${}^*\omega_i = \sum_{j=1}^{2n} \lambda_{ij}\omega_j$$

The unknowns $\{\lambda_{ij}\}$ can be computed by solving the following linear equation system:

$$\int_M \omega_i \wedge {}^*\omega_j = \sum_{k=1}^{2n} \lambda_{jk} \int_M \omega_i \wedge \omega_k.$$

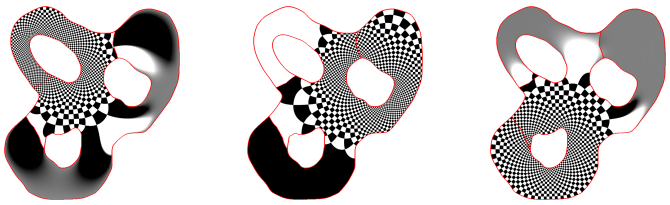


Figure 46: Holomorphic 1-forms basis $\{\tau_1, \tau_2, \tau_3\}$.

The wedge products can be computed as follows. Let $[v_i, v_j, v_k]$ be a triangle face, $e_i = [v_j, v_k]$, $e_j = [v_k, v_i]$ and $e_k = [v_i, v_j]$. Then by direct computation, we get

$$\int_{[v_i,v_j,v_k]} \omega_1 \wedge \omega_2 = \frac{1}{2} \begin{vmatrix} \omega_1(e_i) & \omega_1(e_j) & \omega_1(e_k) \\ \omega_2(e_i) & \omega_2(e_j) & \omega_2(e_k) \\ 1 & 1 & 1 \end{vmatrix}$$

and

$$\int_{[v_i,v_j,v_k]} \omega_1 \wedge {}^*\omega_2 = \tfrac{1}{2}[\omega_1(e_i)\omega_2(e_i)\cot\theta^i_{jk} + \omega_1(e_j)\omega_2(e_j)\cot\theta^j_{ki} + \omega_1(e_k)\omega_2(e_k)\cot\theta^k_{ij}]$$

Let $\tau_k := \omega_k + {}^*\omega_k\sqrt{-1}$, Then $\{\tau_1, \tau_2, \cdots, \tau_n\}$ form the basis for the holomorphic 1-form group. Figure 46 shows the basis for holomorphic 1-forms.

Step 4: Slit conformal mapping. We then search for a special holomorphic 1-form $\tau = \sum_k x_k \tau_k$, such that

$$Img(\int_{\gamma_0} \tau) = \sum_k x_k Img(\int_{\gamma_0} \tau_k) = 2\pi.$$

and

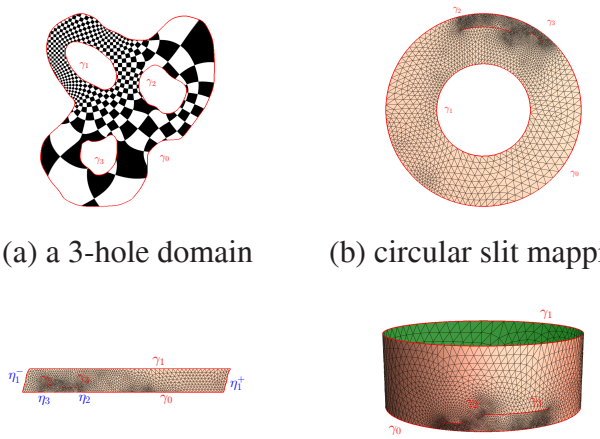
$$\int_{\gamma_i} \tau = \sum_k x_k Img(\int_{\gamma_i} \tau_k) = 0, i = 2, 3, \cdots, n.$$

This implies $\int_{\gamma_1} \tau = -2\pi$. Then the mapping is given by

$$\phi(z) = \exp\{\int_{z_0}^{z} \tau\}.$$

where the integration path is arbitrarily chosen.

Topological torus. A topological torus has the homology basis $\{a, b\}$. The holomorphic 1-form induces the flat metric. The torus is flattened to a periodic rectangle on the Euclidean plane.



(a) a 3-hole domain (b) circular slit mapping

(c) horizontal slit mapping (d) cylinder mapping

Figure 47: Conformal mapping from the domain to the annulus, γ_0 is mapped to the outer circle, γ_1 is mapped to the inner circle.

Simply connected domain. If the input network is a simply connected domain, we select four corner vertices on the boundary $\{v_0, v_1, v_2, v_3\}$ sorted counter-clockwisely. Then we glue two copies of the network along the boundary segments between v_0, v_1 and the boundary segments between v_2, v_3. The result surface is a topological annulus. The above algorithms are able to handle general metric surfaces, then we can map the doubled network to an annulus. By taking the complex logarithm, the original network is mapped to a planar rectangle, such that the four corners are mapped to the corners of the rectangle.

Doubly connected domain. If the input network is a doubly connected domain, the conformal mapping is a canonical annulus. By taking the complex logarithm, it is flattened to a periodic rectangle. Glue one copy along one boundary, the result domain is a topological torus.

Now we summarize the computation and communication steps involved in the pipeline.

First step. Compute cohomology basis. In this step we will find shortest paths connecting the interior holes to the outer boundary. This can be done by a single flooding starting from the nodes at the inner hole boundaries simultaneously.

Second step. Compute harmonic 1-form basis. In this step we compute n harmonic

functions, where n is the number of holes. This uses the Dirichlet boundary condition and an iterative gossip-style algorithm, similar to the distributed algorithm used in [100].

Third step. Compute holomorphic 1-form basis. This involves completely local operations. Each node will solve a linear system only on its neighbors.

Forth step. Slit conformal mapping. This involves only one round of flooding, starting from the outer boundary inward. The nodes compute their virtual coordinates.

The algorithm solves sparse linear systems. Therefore the holomorphic differential method is more efficient compared to the non-linear curvature flow methods.

7.3 Slit Map Algorithm

In order to compute slit maps, one needs to compute the holomorphic forms. In practice, networks are represented as a piecewise linear polyhedral surfaces, namely triangular meshes. All the geometric concepts are generalized to this discrete setting. In the following, we briefly explain the intuitions, details can be found in the appendix B.

Suppose M is a mesh, with vertex, edge and face sets (V, E, F). A function $f : V \to \mathbb{R}$ is defined on vertices, and linearly extended to the whole mesh. A discrete 1-form $\omega : E \to \mathbb{R}$ is a linear function defined on the oriented edges of the mesh. A 2-form $\sigma : F \to \mathbb{R}$ is a function defined on the faces. The boundary operator ∂ takes a simplex to its boundary. The exterior differential operator δ is dual to the boundary operator, e.g. $d\omega = \omega \circ \partial$.

First step. The homology and cohomology groups are computed using algebraic topology methods. If M is a multi-holed disk, then homology group contains loops which can not shrink to a point. The cohomology group is the dual to the homology group. Suppose ω is a 1-form, such that $df \equiv 0$, then ω is a *closed 1-form*. Suppose f is a function, then $df([v_i, v_j]) = f(\partial[v_i, v_j]) = f(v_j) - f(v_i)$ is a 1-form, which is called an *exact 1-form*. Because $d^2 = 0$, all exact 1-forms must be closed. But closed 1-forms may not be exact. If ω_1, ω_2 are closed 1-forms and differ by an exact 1-form $\omega_1 - \omega_2 = df$, then we say ω_1 is cohomological to ω_2. The cohomology

group contains all cohomological equivalence classes of 1-forms.

Second step. Physically, if a surface is deformed, the deformation will introduce the membrane stretching energy, the so-called *harmonic energy*. Harmonic energy can be derived from the derivative of the deformation, which is the 1-form. According to Hodge theory, each cohomological class has a unique 1-form, the so-called *harmonic 1-form*, that minimizes the harmonic energy. The basis of all harmonic 1-forms can be computed using heat diffusion method from the basis of cohomology group.

Third step. Harmonic 1-forms can be treated as piecewise constant vector fields on the mesh. If ω_1 and ω_2 are harmonic 1-forms, such that the vector field of ω_2 can be obtained by rotating that of ω_1 at each point, then we say ω_2 is *conjugate* to ω_1. A pair of conjugate harmonic 1-forms form a *holomorphic 1-form*. In this step, we compute the basis of all holomorphic 1-forms.

Forth step. Finally, we compute a special holomorphic 1-form ω by linear combining the basis of all holomorphic 1-forms, such that the imaginary part of the integration ω along the exterior boundary is 2π, -2π along one inner boundary, and 0's along all other inner boundaries. The slit map $\phi : V \to \mathbb{C}$ is given by this ω. Choose one vertex $v_0 \in M$, for each vertex $v \in M$, find any path γ connecting v_0 and v, then $\phi(v)$ is the complex exponential of the integration of ω along γ.

7.4 Simulations

7.4.1 Experiment Setting

To identify the effectiveness of the aperiodic dense curve in network covering, we choose sensor networks following unit disk graph model. Since the aperiodic dense curve is identified as a continuous line in the universal covering space, to apply it in sensor networks, the curve needs to be mapped to a discrete path. There are various strategies for curve discretization, in our setting we suppose there is a sensor network G deployed on a continuous sensor domain $\mathcal{R}$, then compute the dense curve on $\mathcal{R}$ and expand the width of the curve to get a belt region $\mathcal{B}$. The discrete path starts at an arbitrary node $s \in \mathcal{B}$, then for any node u on the path, we compute its next hop from $C(u) = \{u | u \in N(u) \text{ and } Coor(u) \in \mathcal{R}\}$ based on a

closeness measurement, an infinite discrete path can be generated as the aperiodic dense curve covers $\mathcal{R}$.

7.4.2 Comparison with Various Network Covering Approaches

The Space filling curve is able to cover a specific area, which is quite similar to the aperiodic dense curve. However, there are several different aspects between the two when whey are applied in sensor networks covering. First, starting at a particular node, the space filling curve would visit the nearby nodes before moving to nodes faraway, while the aperiodic dense curve doesn't have this locality. Since in sensor networks nearby nodes tend to contain similar data, the aperiodic dense curve is able to get a better sampling of the whole network than the space filling curve. Second, if the network has holes inside, our method would map the holes to slits, in this case the aperiodic dense curve would still be able to cover the whole network. On the contrary, the space filling curve would get stuck on the boundaries of holes. In our experiment, we compare aperiodic dense curve with a particular space filling curve called the Moore curve.

Besides space filling curves, there are other ways to perform network covering. The Eulerian cycle method builds a spanning tree from the network, duplicates all its edges, then finds an Eulerian cycle in it. Compared to the aperiodic dense curve, Eulerian cycle also exhibits locality issue in network coverage. In our experiment, Eulerian cycles are generated from minimum spanning trees. We also compare with random walk, in which the next hop of the path is chosen uniformly randomly from the neighbors of the current node.

Figure 48(a) shows the network coverage percentage as the paths move forward. The x axis is the length of the path in the number of hops, the y coordinate is the percentage of nodes covered by the path. It's obvious that aperiodic dense curves, Eulerian cycles and Moore curves are much better than random walk in terms of coverage, which is not surprising because these are well-guided curves, while random walk is aimless.

Figure 48(b) shows the average shortest distance from the set of unvisited nodes to the set of visited nodes, this average shortest distance criteria measures the locality property of the paths. If the path visits most of nearby nodes before moving

to nodes faraway, the average shortest distance would remain relatively high as the path visits more nodes, vice versa. Compared to other methods, the average shortest distance of the aperiodic dense curve drops more quickly, which means that the aperiodic dense curve visits the network in a more global way than other methods.

To conclude, the aperiodic dense curve, Moore curve and Eulerian cycle cover the network much faster than random walk. Compared to the Moore curve and Eulerian cycle, the aperiodic dense curve is able to quickly sample the whole network, which gives a good representation of the network in the early stage.

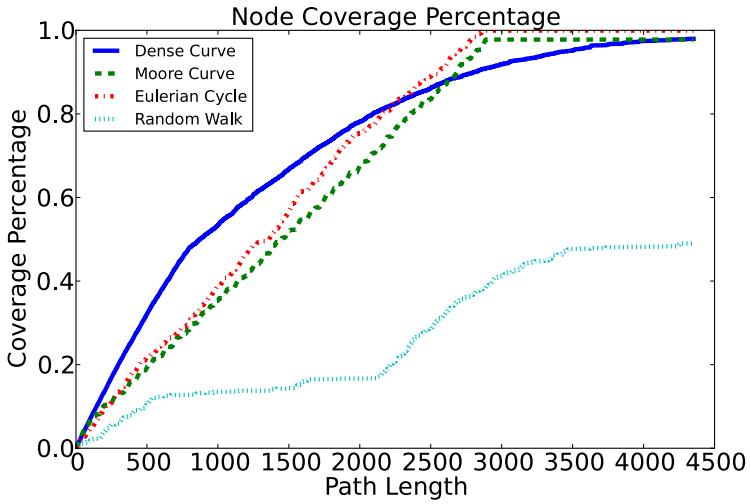
7.4.3 Covering Network with Holes

Sensor networks may have obstacles inside, which lead to holes in sensor domain. Normal space filling curves like Moore curve would fail under such cases, because those curves only cover the unit square, and would become disconnected pieces. By performing conformal mapping to mapped the holes to slits, the aperiodic dense curve can be used to cover the whole sensor domain.

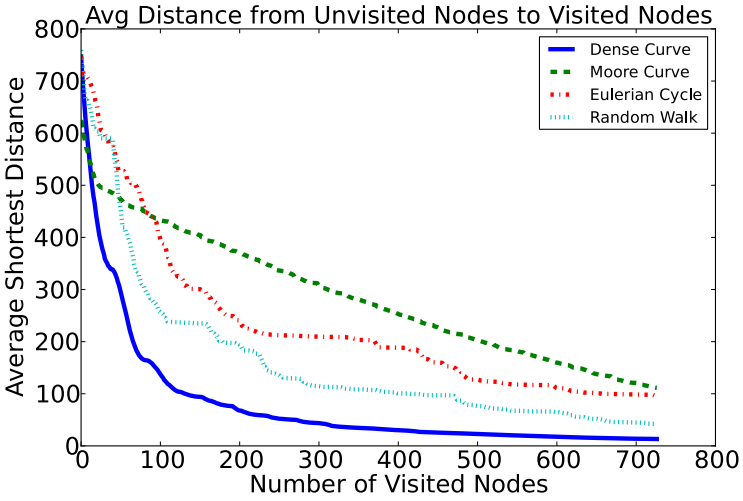
7.4.4 Dense Curve Applications

Double ruling. A pair of non-parallel aperiodic dense curves give two trajectories on the network that intersect with each other. Those two trajectories form a lattice on the network, which is very suitable for double ruling. Double ruling is a framework for information in-network storage and fetching problem, the producer stores data on the nodes of one trajectory, the consumer fetches the data by following another trajectory to a node that stores the data. By using two aperiodic dense curves perpendicular to each other for data storing and fetching, the data producer is given the freedom to store the data on any set of nodes on the producer trajectory, while the data consumer is guaranteed to get the data by following the consumer trajectory.

Multiple paths. It's common to cover the sensor network and fetch data by using not only one, but multiple data mules. To coordinate and collaborate with each other the data mules may need to communicate during the data collection, which can be expensive or even infeasible. Since each aperiodic dense curve would be



(a)



(b)

Figure 48: Comparison Between Dense Curve and Other Network Covering Approaches. (a) Network Coverage. (b) Average Shortest Distance from Unvisited Nodes to Visited Nodes.

able to cover the whole network in a particular pattern, and the visiting pattern is predefined by the slope and starting position, by deliberately assigning slopes and starting positions to multiple aperiodic dense curves, not only they would be able to cover the whole network, but also the paths could be as if they were coordinated so that the coverage converges quickly.

Chapter 8

Navigation in Complex Networks

8.1 Introduction

In the 1960's, Stanley Milgram and his collaborators conducted a series of experiments in which individuals from Nebraska and Kansas were asked to try and get letters delivered to unknown recipients in Boston [151]. A person forwards the letter to a friend who is more likely to know the target. Many letters were discarded by uncooperative intermediaries, but about 20% of the letters arrived at the target, in an average of under six hops. This experiment is the earliest to verify the 'small-world phenomenon' (aka 'six degree of separation') that there *exists* a short path between almost any pair of individuals in the world. It was later discovered that many other networks, in vastly different contexts ranging from power grids, film collaboration networks, and neural networks [48] to email networks [42], food webs [158] and protein interaction networks [72], also exhibit the small-world property.

In addition to revealing the existence of short paths in real-world acquaintance networks, the small-world experiments showed that these networks are *navigable*: A short path was discovered through a *local* algorithm with the participants forwarding to a friend who they believed to be more likely to know the target. Although forwarding decision-making was not systematically recorded, geographical proximity was found to be an important forwarding criterion in some cases. Other criteria such as profession and popularity may have been used as well. A recent small-world study using email-chains [42] confirms this, finding that at least half

of the choices were due to either geographical proximity of the acquaintance to the target or occupational similarity. Thus these experiments hint that perhaps also in other networks, some greedy routing algorithm can successfully deliver messages, provided that nodes are given appropriate coordinates.

We consider the conjecture that real-world networks from diverse contexts, social and non-social, can be embedded in a *low-dimensional hidden space* where the distances between nodes in the hidden space approximate their graph distances in the network, such that some greedy mechanism minimizing the distances to the destination *in the latent space* is able to find a short path for most pairs of nodes. The existence of such a latent space in *social* networks is suggested by the sociological principle of *homophily*, that friends tend to have similar traits or adopt similar behaviors [108]. A set of individuals with a large number of social ties between them may indicate that they have nearby positions in the space of characteristics [16]. This social space may refer to a space of observed or unobserved latent characteristics that represent potential transitive tendencies in network relations. Moreover, networks may have a spatial component, with connectivity decreasing in geographical distance [26, 93]. Nevertheless, we must remark that the methods we consider do not use any social, spatial or any other node-identifying information. Instead, they attempt to *discover* this space from network structure alone. This allows them to be applicable to anonymous social networks as well as non-social networks that must be navigated. We must also emphasize that these methods are rather basic and straight-forward. The methods themselves are not the main contribution here. Rather, we use them to verify the navigability conjecture on five different empirical networks, social and non-social, of very diverse origin and topology. Our results show that they can all be navigated. In each network, a majority of packages is delivered in fewer than six steps.

Our contribution. We propose to first embed the network in a latent space and apply greedy routing with the coordinates generated.

We start with the obvious choice of Euclidean space and use the prominent embedding algorithm of Multi-Dimensional Scaling (MDS) [18]. We measure the distance between any two nodes as the length of the shortest path between them in the social network. With the all pair graph distances MDS produces a coordinate for every node in the Euclidean space with a pre-specified dimensionality.

MDS however requires the all pairs distance matrix of the network and is computationally intensive. To make our method feasible for very large empirical networks that are only partially observed, we adopt landmark-based MDS (LMDS) [38], in which a few nodes are selected as landmarks and embedded first, and the rest of the nodes embed themselves using distances to these landmarks. It is shown below that landmark MDS achieves a comparable performance (in terms of delivery rate and routing path length) to that of MDS, but more than an order of magnitude faster. The use of landmark MDS also implies a distributed implementation of the embedding/routing algorithm. In particular, we can sample a small set of nodes in the social network as landmarks, and embed them. Any individuals who would like to route can embed themselves *on the fly*, by using the distances to the landmarks. In a coauthorship or film collaboration network, the obvious choices for the landmarks are the famous scholars or actors with well-known connections to each other and the distances to others pre-calculated (such as the Erdös number or the Bacon number). The distances from all nodes to the landmarks can be computed in time linear in network size.

As the Euclidean space of dimension d has a geometric growth rate and small-world graphs have been observed to have low diameters, suggesting an exponential growth rate, we also consider embedding into the hyperbolic space. We employ R. Kleinberg's embedding method [88] to embed a tree in hyperbolic space with the induced coordinates used for greedy routing.

We consider five real-world networks from diverse contexts and of varying topology: A peer-to-peer file sharing network, a scientific collaboration network, a movie-actor coappearance network, and an Internet autonomous systems network. Surprisingly, with simply out-of-the-shelf methods one is able to get reasonably high delivery rate, and even more, very small average path length, within *six* steps. Before we conducted these experiments, we expected that possibly some fraction of the messages can reach the destination via the greedy algorithm, as the embedding by MDS preserves the distances to some extent. We have never expected that these messages only use 4 or 5 hops on average! Note that successful delivery does not imply the path is short.

Significance of network navigation. The task of identifying short paths appears

in a wide variety of empirical settings. Short paths allow for speedy package delivery in decentralized file-sharing networks [2], searching for pathways in very large metabolic networks [45], and enable reputation-based trust in exchange [24]. The fact that we are living in a 'small world' suggests that potentially we can consult with any expert in a field of interest, or do business with any individual, with recommendations through a short chain of friends — if only we were able to find such a short path quickly. Short path identification will also facilitate a number of social operations. The occupants 'structural holes', positioned on short paths between otherwise distant others in social networks earn brokerage benefits [23, 152]. The task of ensuring sufficient structural distance between two individuals appears frequently: monopoly mediation, double-targeting in advertisement, avoiding infection, and sharing confidential information that must travel far to reach an unwanted ear [25, 152].

In many of these application scenarios, a central navigation device is often lacking, distributed flooding causes congestion and excessive use of resources. In addition, the nodes may have limited information storage capacity and processing power. The network data may be incomplete, and node attribute information may be absent altogether. For these settings greedy routing is a better choice than centralized short path computations.

8.2 Related Work on Small World Graphs

8.2.1 Navigation in model networks

A number of studies have proposed mathematical models for small world networks and navigation in such networks.

Watts and Strogatz [48] proposed as a 'random rewiring model' in which with some probability the edges on a ring are rewired to random vertices. The rewiring probability can be tuned to generate networks in between the two extremes of perfectly regular and perfectly random networks. It is shown that for most of the parameter space, networks simultaneously exhibit high clustering and low path length. Additionally, three diverse real-world networks are shown to exhibit both properties. They show that short paths often exist but not how they could be found without

global knowledge of the network.

Barabasi *et al.* [10] considered an evolving graph in which each newcomer connects to existing vertices with probability proportional to their current degree (thus the name preferential attachment model). The network constructed is a scale-free graph, i.e., it has a power-law degree distribution, a property of various real-world networks. The graph also has small diameter and in addition, hub nodes that are highly connected to other vertices. For scale-free graphs, a degree-based greedy routing has been investigated [2, 82]. The intuition is to send the message to a neighbor with higher degree as the neighbor is more likely to be a neighbor of the destination.

Another idea to navigate in small world networks is to make use of user identities (geographical location, profession, etc.) and the structure of the 'social space'. Kleinberg [86] considered a lattice network in $\mathcal{R}^d$ and placed additional edges pq with probability proportional to $1/|pq|^\alpha$, where $|pq|$ is the Euclidean distance between p, q and α is a parameter. Then he showed that if $\alpha = d$ the greedy algorithm of delivering the message to the node closest to the destination in *Euclidean distance* is able to find a short path to the destination with polylogarithmic number of hops. If $d \neq \alpha$, the greedy routing takes necessarily polynomial number of hops, i.e., the network is not navigable. Watts *et al.* [47] considered a hierarchical professional organization of individuals and a homophilious network with ties added between two nodes closer in the hierarchy with a higher probability. If each node has a fixed probability of dropping the message, they show a greedy routing algorithm sending packages to the neighbor most similar to the target (called homophily-based routing) successfully deliver a fraction of the messages before they are dropped. Kleinberg [84] also confirmed similar results on a hierarchical network. Şimşek and Jensen [145] evaluated routing schemes on networks with different homophilous level. When the homophily level is low, degree based routing is effective as the hubs connect different part of the network. When the network homophily level is high, hubs are not very useful as they connect to other individuals very similar to themselves. They proposed to use a simple product of the homophily and degree to estimate the neighbor who is most likely to be directly connected to the target.

Boguñá et al. [17] incorporated both the idea of having a social space and the

power law degree distribution. They considered nodes on a ring and assigned target degrees from a power law distribution. An edge is then placed between two nodes with a probability positively dependent on their distance and negatively dependent on their degrees. They investigated greedy routing with the distances on the ring as a means of navigating in the network.

Krioukov et. al [90] considered using a hyperbolic plane as the hidden social space. Nodes are uniformly distributed in a radius R disk in a hyperbolic plane with edges placed in pairs with distance smaller than r. They show that such a graph is naturally scale-free and that greedy routing with hyperbolic distance delivers the packets with high success rate.

8.2.2 Navigation in real-world networks

Although the theoretical models and algorithms above are very inspiring, they require networks to satisfy certain properties, such as a scale-free degree distribution, or they require additional node-identifying information. They may fail in real-world networks that violate these properties, and are impossible to apply in real-world networks that lack node-identifying information, such as anonymous social networks and non-social networks. Even if such information is available, the edges in real-world networks do not necessarily follow the distribution of similarity-dependence specified in these theoretical models. These algorithms have been tested on only a few real-world networks for which node-identifying information was available, and without much success. For example, the hierarchical organization model [47] has been shown to work well only on the HP email network, for which messages were delivered in a median of four hops, but perform poorly for the Club Nexus online social network due to incomplete data or less structured hierarchy [1] (even using extensive profile knowledge the local search has a medium of 21 steps and a mean of 53 steps). Using geographical locations has shown only a delivery rate of 13% to deliver a message to the target city (not the target individual!) on a LiveJournal data set [99]. Kleinberg's small world model [86] is used to fit the 'web of trust' of the email cryptography tool Pretty Good Privacy (PGP) [136]. But the delivery rate is only 32% delivery rate with mean 26 steps.

Compared with prior work, we do not require the network to be scale-free as

in [2, 17, 82]. We do not assume nodes stay in a given space as in [17, 86, 90]. And we do not require any node identities or geographical locations as in [47,84,86,145]. Instead of requiring an embedding of the network in an observed space, our task is to discover the hidden space of real-world networks, which is possibly specific to each network. As it turns out, with this hidden space discovered, greedy routing achieves *much higher* success rates compared with previous experiments with real-world spatial embedding.

8.2.3 Graph embedding and greedy routing

Embedding a graph in Euclidean spaces with small metric distortion has been studied actively in recent years. It is known that any graph of n vertices can be embedded in $\mathcal{R}^d$ with $d = O(\log n)$ such that the graph distance is distorted by a factor of at most $1+\varepsilon$, for any $\varepsilon > 0$, with the Euclidean distance in the embedding (please refer to the book [107] for a large body of work on this topic). It can be shown that the Internet Autonomous Systems network (AS-network) can be embedded in $\mathcal{R}^7$ such that most of the routes can estimate their inter-delay fairly accurately by their Euclidean distances [85,115]. Similar efforts have also been done for transportation networks [20].

The greedy routing algorithm minimizing 'distance' to the destination is used pervasively in ad hoc wireless network routing [19, 78], which is the motivation for the study of graph embeddings that supports greedy routing schemes. Most existing work only considered embedding in low dimensional Euclidean space such as $\mathcal{R}^2$ or $\mathcal{R}^3$ [95, 118].

8.3 Embedding and Greedy Routing

In this section we describe the algorithms we used for embedding a given complex network in Euclidean space and greedy routing with the 'social distances' in that space. We considered embedding in both Euclidean spaces and hyperbolic spaces. We remark that these algorithms are 'off the shelf' techniques. However, even these simple methods produce extremely short paths with greedy routing. We discuss the implication and empirical significance of the results in a later section.

8.3.1 Embedding in Euclidean Spaces

Multi-dimensional scaling (MDS) is a classical method for embedding a set of nodes in $\mathcal{R}^d$. It takes an $n \times n$ distance matrix P as input, outputs an $n \times d$ coordinate matrix such that the ℓ_2 distance between any pair of nodes approximates the corresponding distance in P. MDS is done as follows:

1. Transfer $P = (p_{ij})_{n \times n}$ to $B = (b_{ij})_{n \times n}$ s.t.

$$b_{ij} = -\frac{1}{2}(p_{ij}^2 - \frac{1}{n}\sum_{j=1}^{n} p_{ij}^2 - \frac{1}{n}\sum_{i=1}^{n} p_{ij}^2 + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij}^2).$$

   This step shifts the matrix to the center by subtracting the mean.

2. Perform eigen-decomposition on B s.t.

$$B = VAV^T,$$

   in which matrix A is a diagonal matrix with the eigenvalues ordered from largest to smallest and matrix V contains the corresponding eigenvectors. Set

$$X = VA^{1/2}.$$

   Then X gives the coordinate of the n nodes in n dimensional Euclidean space.

3. To reduce the embedded dimension to d, take the largest d eigenvalues from A and their corresponding eigenvectors in V to form an $n \times d$ coordinate matrix.

MDS has a running time of $O(n^3)$ and requires $O(n^2)$ space, once the distance matrix P is given. In our case, the distance matrix will include the all pairs shortest path distances.

For large data sets, MDS is too slow. V. de Silva and J. Tenenbaum proposed a landmark based multi-dimensional scaling method (LMDS) [38]. LMDS selects a small subset of nodes as landmarks, and performs MDS to embed these landmarks. The other nodes will measure their distances to the landmarks. By performing distance-based triangulation, each node will embed itself. Denoting by n the number of nodes, k the number of landmark nodes, d the embedded dimension, the procedure is as follows:

1. Let P_k denote the squared distance matrix of landmark nodes, compute B_k by adopting step1 in MDS, which shifts P_k to its center.

2. Let λ_i and $\vec{v_i}$ denote the ith largest eigenvalue and its corresponding eigenvector of B_k, respectively. Compute

$$w_i = \vec{v_i}^T / \sqrt{\lambda_i},$$

   then $W = (w_1, w_2, \ldots, w_k)$ is the transformation matrix of triangulation.

3. Let δ_i, $i = 1, \cdots, k$, be the hop distance vector from node i to the k landmarks, taking its mean as

$$\delta_\mu = \sum_{i=1}^{k} \delta_i / k.$$

4. Given a node i, the embedded coordinate

$$\vec{x_i} = -\frac{1}{2} W(\delta_i - \delta_\mu),$$

   which is an affine transformation of the distance vector. The first d elements of $\vec{x_i}$ gives the desired embedding result.

LMDS only requires $O(kn)$ space and its running time is $O(dkn + k^3)$, where k is the number of landmarks, d is the dimensionality to be embedded to. Since $d < k \ll n$, LMDS requires much less space and time than MDS. In our applications k and d are both chosen to be small constants. Thus LMDS has a linear running time.

Another benefit of LMDS is that it does not require the knowledge of the entire network. Note that we only need to know the shortest path distances from the landmarks to all other nodes in the network. This can be achieved by using breadth-first search from the landmarks with a running time of $O(kn)$.

MDS requires the knowledge of the entire network and thus is less desirable for a distributed system, compared with LMDS. To use landmark MDS, each landmark can flood the network so every node knows the distance to the landmarks. One of the landmark nodes performs the classic MDS method on the pairwise distance matrix $M_{k \times k}$ on all landmarks and broadcasts the landmark coordinates to the entire network. Non-landmark nodes then perform distance-based triangulation on their own to derive their coordinates. Since we typically use a constant number of

landmarks. k can be considered as a constant, so MDS on the landmark nodes takes a constant amount of time and space. Alternatively, we may also use the distributed MDS [36] to compute the coordinates of the landmark nodes in the first step.

8.3.2 Embedding in Hyperbolic Spaces

A hyperbolic plane is a 2D Riemannian manifold with negative curvature. A popular model is the Poincaré disk model, in which points are in a unit disk, and the straight lines are segments of circles contained in the disk orthogonal to the boundary of the disk, or else diameters of the disk. If u, v are two vectors with Euclidean norm less than 1, we define an isometric invariant by

$$\delta(u,v) = 2\frac{||u-v||^2}{(1-||u||^2)(1-||v||^2)},$$

where $||\cdot||$ denotes the usual Euclidean norm. Then the distance function is $d(u,v) = \operatorname{arccosh}(1+\delta(u,v))$.

R. Kleinberg [88] proposed a method to embed a graph in hyperbolic plane. First we compute a spanning tree T of the graph. The tree T is then embedded in hyperbolic plane so that each vertex is given a coordinate. Note that although one can route on the tree edges only, the non-tree edges are also used by greedy routing to produce (hopefully) shorter paths.

If the maximum degree of the tree T is d, there will be at most d branches from the root. We take a regular polygon P such that each side corresponds to a branch. Take any side of P, we introduce a hyperbolic isometry τ which maps endpoints of this side to 1 and -1 while mapping the midpoint of its corresponding arc on the boundary circle to $-i$. Define $u = \tau(0)$ as the point of root on the Poincaré disk, then the virtual coordinate of root node r can be computed as $\mu_r^{-1}(u)$. μ_r is a Möbius transformation defined below. Consider two hyperbolic isometries

$$a : z \longmapsto -z,$$

$$b : z \longmapsto \tau(\rho(\tau^{-1}(z))), \rho(z) = e^{2\pi i/d}z,$$

the Möbius transformation is computed from a and b. For each pair of parent and child nodes $(p(w), w)$, if the virtual coordinates of $p(w)$ is known as $f(p(w))$, points

on the Poincaré disk for $p(w)$ and w are u and v, respectively, then $f(w) = \mu_w^{-1}(v)$. This process can be made to work in a distributed manner and it takes $O(n)$ time to find coordinates for all nodes. For the scheme described above the length of the coordinates can be exponential in n in the worst case. There are techniques to reduce the length of the coordinates to be $O(\text{polylog}\, n)$.

8.3.3 Greedy Routing in Latent Spaces

After the construction of the coordinate system, greedy routing will be used to navigate from source to destination, by sending the message to a neighbor closer to the destination. The distance is evaluated by the Euclidean distance or hyperbolic distance, depending on the embedding. When a node does not have a neighbor closer to the destination than itself, greedy routing gets stuck and fails to deliver the message. The performance is evaluated by measuring the success rate of greedy routing and the average routing path length.

Greedy Routing with Degree Information. A small world network typically has a power law degree distribution [11], with many low-degree nodes and a few high degree 'hubs'. As high degree nodes have more neighbors, routing to high degree nodes has a higher chance of encountering a node with the destination as immediate neighbor. This is the idea used in the degree-based greedy routing by Adamic et al. [2]. However, one drawback of pure degree-based routing is that messages arriving at hub nodes have no direction to 'come down' to low degree destinations.

We propose a hybrid scheme by using greedy routing based on both distance and degree. Apart from the reason that high degree nodes are 'more connected' to other nodes, they also tend to be embedded nearer to the 'core' of the network than low degree nodes, as will be shown later with our experiments. We set a degree threshold T. The nodes with degree higher than T are the *hubs* of the network. The rest of the nodes are at the *periphery* of the network. When the message is at a periphery node s, we send the message to its neighboring node with highest degree. If s does not have a neighbor with higher degree or it is already a hub, we go back to greedy routing with the embedded distances.

8.4 Experimental Results

We tested our method on a number of complex networks, including five real-world networks we were able to find from publicly available sources, except some really small size networks. We also tested on three artificially graphs generated from models. The details of these data sets are shown in Table 9.

Data set name	Num(node)	Num(edge)	Avg degree	Num(landmark)
Astrophysics (ASTRO)	14,845	119,652	16.1202	100
Internet AS Network (AS)	31,277	70,527	4.5098	100
Gnutella Network (Gnu)	574	835	2.9178	30
Actor Network (Actor)	171,427	6,984,461	81.4861	100, 300 and 500
NYT Network (NYT)	209,158	666,956	6.378	100 and 500

Table 9: Empirical Data Sets

The astrophysics collaboration network (ASTRO) [114] is the network of coauthorship between scientists posting preprints on the Astrophysics E-Print Archive between January 1, 1995 and December 31, 1999.

The Internet AS network (AS) is a snapshot of autonomous systems generated by The Cooperative Association for Internet Data Analysis (CAIDA). This snapshot is taken on March 11, 2009. We do not specify the type of links, say, as customer-provider links or peer-peer links. So AS is an undirected graph with only connectivity information.

The Gnutella network (Gnu) is a data set used by Adamic in [2]. Gnutella was a popular P2P application, the data set is a small world network with hubs as the high degree nodes.

The actor network (ACT) [10] is a data set extracted from imdb, each line between two nodes means those two actors collaborated in one movie.

The NYT news network (NYT) puts an edge between individuals that appear together in at least two articles(strong juxtapositions) in the New York Times newspaper from 1981 to 2007.

The preferential attachment model (Pre) [10] assumes a dynamic network with nodes coming in one by one. When a new node joins the network, it is connected to

existing nodes with probability proportional to their current degree. The preferential attachment model introduces the 'rich gets richer' hypothesis in the formation of complex networks and has been a popular model in explaining natural systems. In this experiment, we take $14,845$ nodes, each additional node has outdegree 8.

The Boguna-Krioukov-Claffy model (BKC) [17] is a recently proposed model assuming a hidden metric space on which the nodes reside. In particular, the hidden space is assumed to be a ring with the nodes uniformly placed on the ring. Each pair u and v has a distance $|uv|$ on the ring. Each node u is assigned a target degree k_u, drawn from a power law distribution. Each edge (u, v) is added with probability proportional to

$$(1 + |uv|/(k_u k_v))^{-\alpha},$$

where α is a model parameter. Intuitively the probability that two nodes have an edge is inversely dependent on their distance in the hidden space, and positively dependent on their target degrees. In general nodes near to each other in the hidden space are more likely to be connected. If they are hub nodes, they are also likely to be connected even though they are far away in the hidden space. In this experiment, we draw the degree from power law distribution Prob$\{k = i\} \sim 1/i^3$, which is the degree distribution in the preferential attachment model. α is set to be 2.

The Erdos-Renyi model (ER) [53] is the uniform random graph model, in which each pair of nodes is connected uniformly randomly. It has a small diameter. We include this graph as a baseline, as a random graph has no special structure to help with navigation.

For all the theoretical model networks, we vary the average degree to examine the performance dependency on the number of edges.

8.4.1 Embedding by MDS and LMDS

We evaluate the performance of embedding with MDS and landmark MDS. LMDS is computationally more efficient than MDS, yet it gives similar embedding results even when k is small.

Figure 49 shows the comparison between MDS and LMDS. Since the coordinates of landmark nodes by LMDS are consistent with those in MDS [38], LMDS with all nodes as landmarks is exactly MDS. The testing graph is generated by

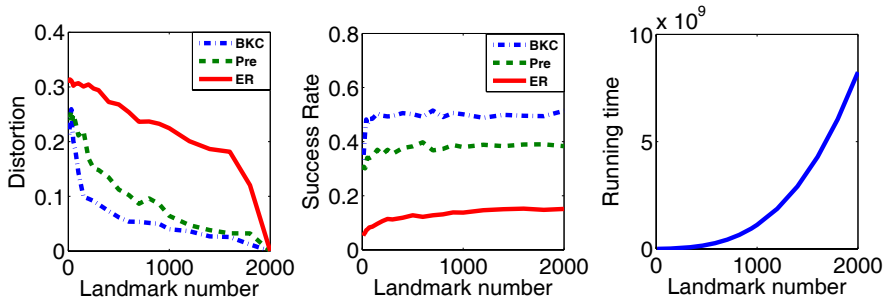


Figure 49: Performance comparison of LMDS and MDS. BKC,Pre and ER stands for Boguna-Krioukov-Claffy model, Preferential Attachment model and Erdos-Renyi model, respectively.

means of the preferential attachment model, Boguna-Krioukov-Claffy model and Erdos-Renyi models with $2,000$ nodes and $8,000$ edges. The embedded dimension is 6. Given a pair of nodes u and v, denote the distance generated by MDS and LMDS as $d_{u,v}$ and $d'_{u,v}$, respectively. The *individual distortion* of LMDS relative to MDS is $\rho'_{u,v} = |d_{u,v} - d'_{u,v}|/d_{u,v}$. The *average distortion* is $\rho' = \sum_{u,v} \rho'_{u,v}/n^2$. The average distortion drops as the number of landmarks decreases. Figure 49 also shows that by selecting only a small number of landmarks, high success rates for greedy routing can nevertheless be achieved.

8.4.2 Greedy routing results on Euclidean space

We evaluated two strategies for routing: greedy routing, and greedy routing with degree information. We also evaluated two different strategies for selecting landmarks, random selection and selection of high degree nodes. The left four plots of Figure 50 shows the success rate for different embedded dimensionality. The Internet autonomous system network has the highest success rate, which reaches nearly 80% with an embedded dimensionality of 50. The preferential attachment graph achieves up to 56% success with hubs selected as landmarks, while the astrophysics collaboration network has a success rate of more than 60% with random landmark selection. The average path length of the AS network and preferential attachment graph stays around 4, and ASTRO network stays around 5. This result

shows that greedy routes are short. The results also show that a random graph does not have good navigability. The graph generated by the recently proposed BKC model, though claimed to be navigable in [17], does not work better than the preferential attachment model or the real networks. Among the model networks, random graph are the most 'unstructured' one. It does not have power-law degree distribution or high clustering coefficient. Our experiments show that random graphs are indeed less navigable than all the other networks.

Figure 50 also shows that for some networks selecting highest degree nodes as landmarks performs better, while for other networks random selection works better. Detailed analysis shows that ASTRO network contains multiple clusters, while most of the highest degree nodes belong to only one cluster. Therefore, selecting highest degree nodes cannot capture the true network structure, which leads to lower success rate than random selection. Figure 51 (i) also shows that random selection works better in the actor network.

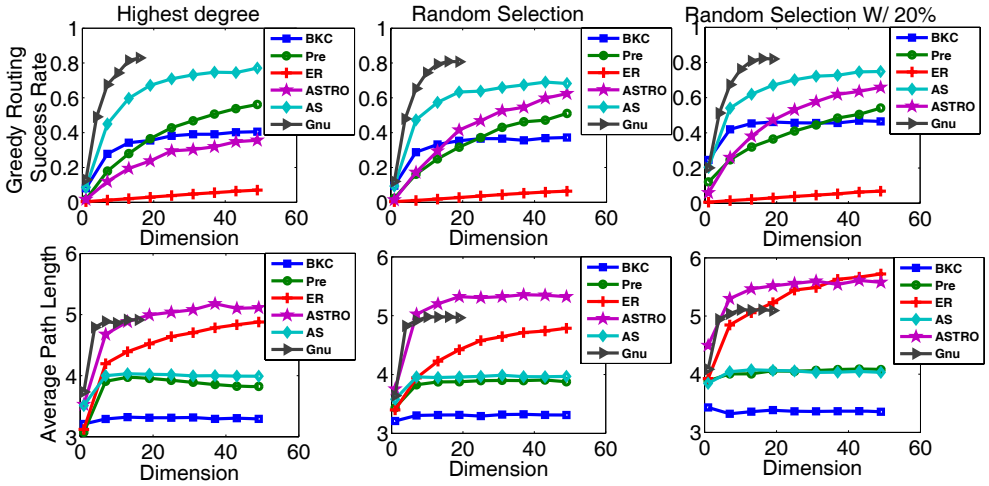
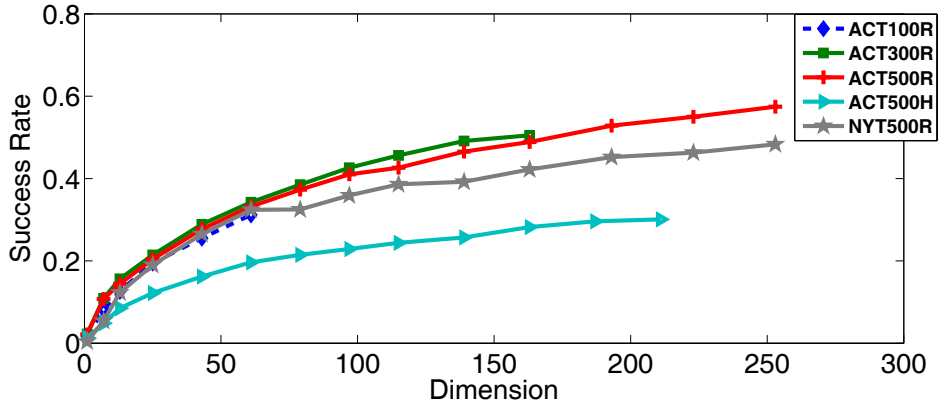
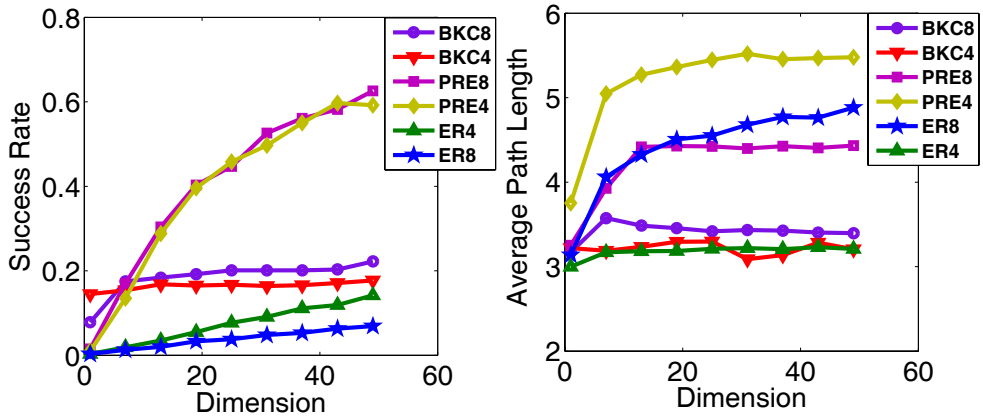


Figure 50: Simple greedy routing. Random Selection W/ 20%: landmark nodes are random selected, 20% highest degree nodes are hub nodes. Node number and average degree of ER, Pre and BKC are the same as ASTRO network. For abbreviations, please refer to Table 9 and Figure 49.

In the right two plots of Figure 50, we set the high degree threshold such that 20% of the nodes are hubs respectively. Contrary to the intuition that greedy



(i)



(ii)

Figure 51: (i) Impact of landmark number and dimension. ACT500H is ACTOR network with 500 highest degree nodes as landmarks. (ii) Impact of average degree. BKC8 means a BKC network with average degree to be 8.

routing with degree information improves the performance, we do not see much improvement. This shows that degree information is not important in our method, but rather that the distances in the discovered space truly help with greedy routing.

Success rate is possibly influenced by landmark number and embedded dimensionality. While Figure 50 shows that the success rate grows with dimension, we take a larger network to see how those two factors influence the success rate.

The actor network is a large network with more than $170,000$ nodes. Figure 51 (i) shows an apparent growth of success rate as the embedded dimensionality grows. Landmark number does not seem to play an important role, since using 100, 300, and 500 landmarks gives similar curves. Remark that the number of dimensions must be smaller than the number of landmarks. Similar results are obtained for NYT network, which contains similar number of nodes with actor network, but with much fewer edges. It suggests that network density does not play an essential role in our greedy routing. Despite that actor network and NYT network are much larger than our other data sets, the average path lengths are below or around 5 for all experiments.

Since the empirical networks have different average degrees, it is essential to justify whether average degree is an important factor in our embedding and greedy routing. Figure 51 (ii) shows the impact of average degree, it takes three model networks preferential attachment model, BKC model and Erdos-Renyi Model. Combining with Figure 50, we show how success rate and average path length changes while the average degree changes from 16 to 8 and 4. The result turns out that average degree does not show an essential impact on our embedding and routing.

8.4.3 Greedy routing results on hyperbolic space

As shown in [88], greedy routing on hyperbolic space is proved to have guaranteed success. Therefore the length of average path length is the crucial performance factor to evaluate. In hyperbolic embedding, different choices of spanning tree and the root of the spanning tree will lead to different routing pathes between source and destination.

We first use the shortest path spanning tree (SPT) rooted as the node with highest degree as the tree in the hyperbolic embedding. This tree is computed by

flooding from the root. In the first phase, an arbitrary node will flood all the other nodes to compute the highest degree node in the network. Ties are broken arbitrarily. In the second phase, The node with maximum degree will perform flooding to get a breadth-first tree.

In the second method, we use random walk to generate a spanning tree (RWT). In particular, we take an arbitrary node u as the starting node, and uniformly randomly select one of its neighbors v. If v has not been visited, edge (u, v) is a tree edge. We move to v and perform the same strategy until all nodes are visited. The running time of this distributed process depends on the cover time of the random walk on the particular graph.and ranges from $O(n \log n)$ to $O(n^3)$. RWT is a spanning tree uniformly randomly selected from the set of all possible spanning trees.

Data Set	SPT			RWT		
	Avg SPL	Avg RL	Avg distortion	Avg SPL	Avg RL	Avg distortion
ASTRO	4.579	5.837	1.459	4.579	23.080	5.770
AS	3.847	4.265	1.422	3.847	9.712	3.237
Gnutella	4.812	5.533	1.383	4.812	9.202	2.301
NYT	5.270	5.971	1.994	5.271	21.846	4.368
Actor	4.185	5.617	1.404	4.182	27.990	6.997
Pre	3.318	4.001	1.334	3.318	18.012	6.004
BKC	3.252	3.927	1.309	3.252	20.425	6.808
ER	4.863	7.978	1.995	4.863	36.270	9.067

Table 10: Greedy routing with embedding in hyperbolic space. For abbreviations, please refer to Table 9 and Figure 49. SPL and RL stands for shortest path length and routing path length, respectively.

Table 10 shows the average distortion, average path length and shortest path length for hyperbolic routing on the empirical networks. It shows that there is a huge difference between different spanning trees. The spanning tree selected by shortest path tree rooted on highest degree node achieves much smaller path distortion. We conjecture that the reason might be that SPT usually divides nodes into more balanced branches than the trees obtained from random walk.

8.5 Discussion

Prior work has demonstrated that under certain special conditions, networks can be navigated in a decentralized fashion with algorithmic methods. How navigable real-world networks that may violate these special conditions are, has remained largely unknown. We have shown here that five actual empirical networks, social and non-social, of very diverse origin and topology, can all be embedded in some hidden space such that effective navigation becomes possible. Using rather elementary methods, we deliver a majority of packages in under six hops in every network. Even more surprisingly, navigation results are generally better, not worse, than in simulated model networks.

Our experimental results yielded a number of interesting findings.

Greedy routing performance. The performance of navigation in real-world networks using a hidden space improves substantially on previous results. Past small-world experiments in which human participants forwarded messages in social networks had much lower delivery rates due to uncooperative participants. Milgram's experiment used node identifying information about the destination such as name, gender, occupation, etc. 64 out of 296 letters arrived at the destination (around 20% success rate), with an average path length of around 5.5. Dodds et. al [42] conducted the same Milgram's experiment with emails instead of snail mails and extended the geographical range to the whole world. They show that only 1.5% messages arrived at the destination, as many participants dropped from the game. Liben-Nowell et. al [99] conducted a greedy routing simulation on a LiveJournal data set, thus eliminating user participation issues. They used geographical information as the greedy routing criterion (a message is delivered to a friend closer to the target geographically) and considered delivery successful if the message arrived at the city where the destination individual resides (not the actual precise location of the individual target). Their simulations show a 13% success rate. We compare our results with Liben-Nowell's results, as also their investigation involves real data sets and users are assumed cooperative. Liben-Nowell's results use only geographical location as the greedy routing criterion, which may only partially capture the attributes attributed to navigability. We use the coordinates extracted from the embedding. Our results confirm that using a proper embedding into latent spaces greatly helps

with navigation in the network. It remains as interesting future work to see whether the embedded space corresponds to a set of real-world attributes or combination of attributes.

	ASTRO	AS	Actor	NYT	Gnutella	Pre	BKC	ER
SR	29.7%	35.7%	74.9%	26.8%	51.8%	99.7%	98.4%	98.9%
APL	55.30	10.96	572.54	341.55	14.02	206.97	338.00	631.58

Table 11: Degree-based routing, SR stands for "success rate", APL stands for "average path length"

Adamic [2] proposed degree routing and performed greedy routing simulations on the same Gnutella data set. The degree routing strategy selects the highest degree node among the neighbors which have not been explored, if all neighbors have been visited, routing process enters a dead end and fails. Results on comparison between degree-based routing strategy and our method is shown in table 11. Although degree routing gives very good success rate on dense model networks, average path length is too high to be considered as practical. Besides that, as average degree of Pre, BKC and ER reduces below 4, success rate drops quickly to below 40%, while average path length is still higher than 100.

Sandberg [136] tried to fit a particular small world model by J. Kleinberg [86] to a real data set, the web of trust, so as to discover the users' locations on a grid using Markov Chain Monte Carlo method. The method has only demonstrated limited success with 32% delivery rate and mean 26 steps. Part of the reason could be that it is not clear whether J. Kleinberg's model truly reflects the structure of the real data set.

There is a tradeoff between the dimensionality of the embedding and the success rate of greedy routing using that embedding. On one hand, the success rates grows as the dimensionality is increased, but the growing speed slows down. Thus increasing the embedding dimensionality has diminishing return. Depending on the network structure and the size of network, some networks, for example AS and BKC, have a tipping point on the growth of success rate. On the other hand, the size of the coordinates for each node grows linearly as the embedding dimensionality.

The embedded dimension should also be strictly smaller than the number of landmarks, hence one may have to use more landmarks with a higher dimensionality.

Real networks are more navigable than models. We did not try to pinpoint the exact properties that make certain networks navigable, like many small-world models that have been proposed were intended to capture. Rather, we show that even with off the shelf techniques one is able to get very reasonable performance on *real-world* networks, something we never expected before we ran these experiments. An interesting additional discovery is that the performance of real-world networks turns out to be better, not worse, than equivalent model networks with the same network size and average degree. That is, real networks are more navigable than existing models! Besides results shown here, we also tested the J. Kleinberg's model [86]. With around $14,000$ nodes and average degree 8, it gives low success rate(around 20%) and average path length is around 20. This is not good compared to greedy routing using Kleinberg's grid coordinates, which guarantees delivery and gives similar average path length. In a sense, this suggests that there is some structural feature to real-world complex networks that has not yet been captured by any single theoretical model. It is also possible that hybrid models that combine the different navigation-facilitating characteristics of various small-world models would do better. How to build such hybrid models is an interesting line of future work.

The choice of latent space and embedding. We experimented with both Euclidean spaces and hyperbolic spaces as the hidden spaces. Embedding in both spaces achieves high success rate and more importantly, very low greedy routing path length. This raises the question whether the topology of the hidden space matters after all, and whether there is some structure of real-world networks that transcends the hidden spaces.

What is the optimal hidden space? – results on the BKC model. The BKC model [17] assumes a hidden space for the nodes in a social network but does not mention how to *discover* this space. The model assumes a ring structure as the hidden space and greedy routing is based on the distance on the ring. With this method, routing on the BKC graph with $14,845$ nodes and $118,325$ edges has a success rate of about 25% with average path length to be 8.07. As Figure 50 shows, our method will give around a 47% success rate with the average path length to be

3.36. Thus, paradoxically, although the BKC model constructed the network with an assumed hidden space, the assumed hidden space is apparently not the optimal one. It remains as interesting future work how to characterize the 'hidden space' from the network structure and how to find the optimal hidden space, if there is one.

Bibliography

[1] L. Adamic and E. Adar. How to search a social network. *Social Networks*, 27:187–203, 2005.

[2] L. A. Adamic, R. M. Lukose, A. R. Puniyani, and B. A. Huberman. Search in power-law networks. *Physics Review E.*, 64:046135, 2001.

[3] G. Alandjani and E. Johnson. Fuzzy routing in ad hoc networks. In *Proceedings of the 2003 IEEE International on Performance, Computing, and Communications*, pages 525 – 530, april 2003.

[4] M. Andrews and L. Zhang. Hardness of the undirected congestion minimization problem. *SIAM Journal on Computing*, 37(1):112–131, 2007.

[5] P. Angelini, F. Frati, and L. Grilli. An algorithm to construct greedy drawings of triangulations. In *Proc. of the 16th International Symposium on Graph Drawing*, pages 26–37, 2008.

[6] S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45:753–782, September 1998.

[7] A. Atlas and A. Zinin. Basic specification for IP fast reroute: Loop-free alternates. IETF RFC 5286, September 2008.

[8] J. M. Bahi, A. Makhoul, and A. Mostefaoui. Localization and coverage for high density sensor networks. *Comput. Commun.*, 31(4):770–781, 2008.

[9] A. Banerjea. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 194–205, New York, NY, USA, 1996. ACM.

[10] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[11] M. Barthelemy and L. Amaral. Small-world networks: Evidence for a crossover picture. 82(15), 1999.

[12] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34:209–219, June 2006.

[13] M. Ben-Chen, C. Gotsman, and C. Wormser. Distributed computation of virtual coordinates. In *Symposium on Computational Geometry*, pages 210–219, 2007.

[14] R. Blum, S. Kassam, and H. Poor. Distributed detection with multiple sensors ii. advanced topics. *Proceedings of the IEEE*, 85(1):64 –79, jan 1997.

[15] A. I. Bobenko and B. A. Springborn. Variational principles for circle patterns and koebe's theorem. *Trans. Amer. Math. Soc*, 356:659–689, 2004.

[16] E. S. Bogardus. Social distance in the city. *Proceedings and Publications of the American Sociological Society*, 20:40–46, 1926.

[17] M. Boguna, D. Krioukov, and K. C. Claffy. Navigability of complex networks. *Nature Physics*, 5:74–80, January 2009.

[18] I. Borg and P. Groenen. *Modern Multidimensional Scaling: theory and applications*. Springer-Verlag, 2nd edition, 2005.

[19] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[20] U. Brandes, F. Schulz, D. Wagner, and T. Willhalm. Generating node coordinates for shortest-path computations in transportation networks. *Journal of Experimental Algorithmics*, 9(1), 2004.

[21] A. B. Brown and M. Halperin. On certain area-preserving maps. *Annals of Mathematics*, 36(4):833–837, Oct. 1935.

[22] J. Bruck, J. Gao, and A. Jiang. MAP: Medial axis based geometric routing in sensor networks. *Wireless Networks*, 13(6):835–853, 2007.

[23] R. S. Burt. *Structural Holes: The Social Structure of Competition*. Cambridge University Press, 1992.

[24] V. Buskens. *Social Networks and Trust*. Kluwer, 2002.

[25] V. Buskens and A. van de Rijt. Dynamics of networks if everyone strives for structural holes. *American Journal of Sociology*, 114:371–407, 2008.

[26] C. T. Butts. Predictability of large-scale spatially embedded networks. In *Dynamic Social Network Modeling and Analysis*, pages 313–323, 2003.

[27] L. Buttyn, L. Dra, and I. Vajda. Statistical wormhole detection in sensor networks. In *Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*, volume 3813, pages 128–141, 2005.

[28] J. Cai and W. Wu. Degraded link-disjoint multipath routing in ad hoc networks. In *ISWPC'09: Proceedings of the 4th international conference on Wireless pervasive computing*, pages 149–153, Piscataway, NJ, USA, 2009. IEEE Press.

[29] S. Capkun, L. Buttyn, and J. P. Hubaux. SECTOR: Secure tracking of node encounters in multi-hop wireless networks. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, October 2003.

[30] C. Chekuri, S. Khanna, and F. B. Shepherd. An $o(\sqrt{n})$-approximation for EDP in undirected graphs and directed acyclic graphs. *Theory of Computing*, 2:137–146, 2006.

[31] B. Chow and F. Luo. Combinatorial ricci flows on surfaces. *Journal Differential Geometry*, 63(1):97–129, 2003.

[32] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. In J. F. Traub, editor, *Sympos. on New Directions and Recent Results in Algorithms and Complexity*, page 441, New York, NY, 1976. Academic Press.

[33] J. Chuzhoy and S. Khanna. New hardness results for undirected edge-disjoint paths. Manuscript, 2005.

[34] C. R. Collins and K. Stephenson. A circle packing algorithm. *Comput. Geom. Theory Appl.*, 25(3):233–256, 2003.

[35] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1994.

[36] J. A. Costa, N. Patwari, and A. O. H. III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Transactions on Sensor Networks*, 2(1):39–64, 2006.

[37] S. De, C. Qiao, and H. Wu. Meshed multipath routing with selective forwarding: an efficient strategy in wireless sensor networks. *Comput. Netw.*, 43(4):481–497, 2003.

[38] V. de Silva and J. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

[39] T. K. Delillo, A. R. Elcrat, and J. A. Pfaltzgraff. Schwarz-christoffel mapping of multiply connected domains. *Journal d'Analyse Mathématique*, 94(1):17–47, 2004.

[40] R. Dhandapani. Greedy drawings of triangulations. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 102–111, 2008.

[41] P. Djukic and S. Valaee. Reliable packet transmissions in multipath routed wireless networks. *IEEE Transactions on Mobile Computing*, 5(5):548–559, 2006.

[42] P. Dodds, M. Roby, and D. Watts. An experimental study of search in global social networks. *Science*, 301:827, 2003.

[43] D. Dong, M. Li, Y. Liu, X.-Y. Li, and X. Liao. Topological detection on wormholes in wireless ad hoc and sensor networks. In *Proceedings of the 17th annual IEEE International Conference on Network Protocols (ICNP'09)*, pages 314–323, 2009.

[44] D. Dong, M. Li, Y. Liu, and X. Liao. Wormcircle: Connectivity-based wormhole detection in wireless ad hoc and sensor networks. In *ICPADS '09: Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems*, pages 72–79, Washington, DC, USA, 2009. IEEE Computer Society.

[45] G. Dooms, Y. Deville, and P. Dupont. Constrained metabolic network analysis: discovering pathways using cp(graph), 2005.

[46] T. A. Driscoll and L. N. Trefethen. *Schwarz-Christoffel Mapping*, volume 8. Cambridge University Press, 2002.

[47] D.Watts, P.Dodds, and M.Newman. Identity and search in social networks. *Science*, (296):1302–1305, 2002.

[48] D.Watts and S.Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):409–410, 1998.

[49] J. V. E. Hyytiä. On load balancing in a dense wireless multihop network, 2006.

[50] J. V. E. Hyytiä. Near optimal load balancing in dense wireless multi-jop networks, 2008.

[51] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Information Processing Letters*, 51(4):207–211, August 1994.

[52] D. Eppstein and M. T. Goodrich. Succinct greedy graph drawing in the hyperbolic plane. In *Proc. of the 16th International Symposium on Graph Drawing*, pages 14–25, 2008.

[53] P. Erdos and A. Renyi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[54] J. Eriksson, S. Krishnamurthy, and M. Faloutsos. Truelink: A practical countermeasure to the wormhole attack. In *ICNP*, 2006.

[55] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *Mobile Networks and Applications*, volume 11, pages 187–200, 2006.

[56] R. Flury, S. Pemmaraju, and R. Wattenhofer. Greedy routing with bounded stretch. In *Proc. of the 28th Annual IEEE Conference on Computer Communications (INFOCOM)*, April 2009.

[57] S. Funke and N. Milosavljević. Network sketching or: "how much geometry hides in connectivity? - part II". In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 958–967, 2007.

[58] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, 2001.

[59] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanners for routing in mobile networks. *IEEE Journal on Selected Areas in Communications Special issue on Wireless Ad Hoc Networks*, 23(1):174–185, 2005.

[60] J. Gao and L. Zhang. Load balanced short path routing in wireless networks. *IEEE Transactions on Parallel and Distributed Systems, Special Issue on Localized Communications*, 17(4):377–388, April 2006.

[61] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[62] M. T. Goodrich and D. Strash. Succinct greedy geometric routing in $\mathbb{R}^2$. Technical report on arXiv:0812.3893, 2008.

[63] Y.-J. K. R. Govindan, B. Karp, and S. Shenker. Lazy cross-link removal for geographic routing. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 112–124, 2006.

[64] R. S. Hamilton. Three manifolds with positive Ricci curvature. *Journal of Differential Geometry.*, 17:255–306, 1982.

[65] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Network and Distributed System Security Symposium (NDSS)*, 2004.

[66] Y. C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM*, volume 3, pages 1976–1986, 2003.

[67] Y.-C. Hu, A. Perrig, and D. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 24:370–380, February 2006.

[68] K. Ishida, Y. Kakuda, and T. Kikuno. A routing protocol for finding two node-disjoint paths in computer networks. In *ICNP '95: Proceedings of the 1995 International Conference on Network Protocols*, page 340, Washington, DC, USA, 1995. IEEE Computer Society.

[69] N. James, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 259–268, New York, NY, USA, 2004. ACM.

[70] F. Javadi and A. Jamalipour. Multi-path routing for a cognitive wireless mesh network. In *RWS'09: Proceedings of the 4th international conference on*

Radio and wireless symposium, pages 223–226, Piscataway, NJ, USA, 2009. IEEE Press.

[71] D. Jea, A. A. Somasundara, and M. B. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *IEEE International Conference on Distributed Computing in Sensor System (DCOSS)*, pages 244–257, 2005.

[72] H. Jeong, S. Mason, A.-L. Barabasi, and Z. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001.

[73] R. Jiang, X. Ban, M. Goswami, W. Z. adn Jie Gao, and X. D. Gu. Exploration of path space using sensor network geometry. In *Proc. of the 10th International Symposium on Information Processing in Sensor Networks (IPSN'11)*, pages 49–60, April 2011.

[74] M. Jin, J. Kim, F. Luo, and X. D. Gu. Discrete surface Ricci flow. *IEEE Transaction on Visualization and computer Graphics*, 14(5):1030–1043, 2008.

[75] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[76] M. Kamat, A. Ismail, and S. Olariu. Modified hilbert space-filling curve for ellipsoidal coverage in wireless ad hoc sensor networks. In *Signal Processing and Communications, 2007. ICSPC 2007. IEEE International Conference on*, pages 1407 –1410, nov. 2007.

[77] A. Kansal, M. Rahimi, W. J. Kaiser, M. B. Srivastava, G. J. Pottie, and D. Estrin. Controlled mobility for sustainable wireless networks. In *IEEE Sensor and Ad Hoc Communications and Networks (SECON'04)*, 2004.

[78] B. Karp and H. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of the ACM/IEEE International Conf. on Mobile Computing and Networking (MobiCom)*, pages 243–254, 2000.

[79] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[80] I. Khalil, S. Bagchi, and N. Shroff. MOBIWORP: Mitigation of the wormhole attack in mobile multihop wireless networks. In *Ad Hoc Networks*, volume 6, pages 344–362, May 2008.

[81] I. Khalil, S. Bagchi, and N. B. Shroff. LITEWORP: A Lightweight Countermeasure for the Wormhole attack in multihop wireless network. In *International Conference on Dependable Systems and Networks (DSN)*, Yokohama, Japan, 2005.

[82] B. Kim, C. Yoon, S. Han, and H. Jeong. Path finding strategies in scale-free networks. *Physical Review E*, 65(027103), 2002.

[83] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *Proceedings of the Second USENIX/ACM Symposium on Networked System Design and Implementation (NSDI 2005)*, May 2005.

[84] J. Kleinberg. Small-world phenomena and the dynamics of information. In *In Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[85] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *Proc. 45th IEEE Symposium on Foundations of Computer Science*, pages 444–453, 2004.

[86] J. M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *Proc. of ACM Symposium on Theory of Computing (STOC)*, pages 163–170, 2000.

[87] R. Kleinberg. Geographic routing using hyperbolic space. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 1902–1909, 2007.

[88] R. Kleinberg. Greedy routing using hyperbolic space. pages 1902–1909, 2007.

[89] D. Koutsonikolas, S. Das, and Y. Hu. Path planning of mobile landmarks for localization inwireless sensor networks. In *Distributed Computing Systems Workshops, 2006. ICDCS Workshops 2006. 26th IEEE International Conference on*, page 86, july 2006.

[90] D. Krioukov, F. Papadopoulos, M. Boguna, and A. Vahdat. Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. In *ACM SIGMETRICS Workshop on Mathematical Performance Modeling and Analysis (MAMA)*, June 2009.

[91] K.-W. Kwong, L. Gao, R. Guerin, and Z.-L. Zhang. On the feasibility and efficacy of protection routing in IP networks. In *INFOCOM'10*, March 2010.

[92] W. K. Lai, S.-Y. Hsiao, and Y.-C. Lin. Adaptive backup routing for ad-hoc networks. *Comput. Commun.*, 30(2):453–464, 2007.

[93] B. Latane, J. H. Liu, A. Nowak, M. Bonevento, and L. Zheng. Distance matters: Physical space and social impact. *Personality and Social Psychology Bulletin*, 21:295–805, 1995.

[94] J. K. Lawder and P. J. H. King. Using space-filling curves for multi-dimensional indexing. In *Proceedings of the 17th British National Conferenc on Databases: Advances in Databases*, BNCOD 17, pages 20–35, London, UK, 2000. Springer-Verlag.

[95] T. Leighton and A. Moitra. Some results on greedy embeddings in metric spaces. In *Proc. of the 49th Annual Symposium on Foundations of Computer Science*, pages 337–346, October 2008.

[96] F. Li, S. Chen, and Y. Wang. Load balancing routing with bounded stretch. *EURASIP J. Wirel. Commun. Netw.*, 2010:10:1–10:16, Apr. 2010.

[97] F. Li and Y. Wang. Circular sailing routing for wireless networks. In *INFOCOM*, pages 1346–1354, 2008.

[98] J. Li, J. Jannotti, D. Decouto, D. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of 6th ACM/IEEE*

International Conference on Mobile Computing and Networking, pages 120–130, 2000.

[99] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. In *Proceedings of the National Academy of Science*, volume 102, pages 11623–11628, 2005.

[100] H. Lin, M. Lu, N. Milosavljević, J. Gao, and L. Guibas. Composable information gradients in wireless sensor networks. In *Proc. of the International Conference on Information Processing in Sensor Networks (IPSN'08)*, pages 121–132, April 2008.

[101] W. Lindner and S. Madden. Data management issues in periodically disconnected sensor networks. In *Proceedings of Workshop on Sensor Networks at Informatik*, 2004.

[102] C. Liu, M. Yarvis, W. S. Conner, and X. Guo. Guaranteed on-demand discovery of node-disjoint paths in ad hoc networks. *Comput. Commun.*, 30(14-15):2917–2930, 2007.

[103] X. Liu, Q. Huang, and Y. Zhang. Combs, needles, haystacks: balancing push and pull for discovery in large-scale sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 122–133, 2004.

[104] L. Lovasz. Random walks on graphs: A survey. *Bolyai Soc. Math. Stud.*, 2, 1996.

[105] R. Maheshwari, J. Gao, and S. R. Das. Detecting wormhole attacks in wireless networks using connectivity information. In *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, pages 107–115, May 2007.

[106] M. K. Marina and S. R. Das. Ad hoc on-demand multipath distance vector routing. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):92–93, 2002.

[107] J. Matoušek. *Lectures on Discrete Geometry*. Springer, 2002.

[108] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415–444, 2001.

[109] A. Mei and J. Stefa. Routing in outer space: fair traffic load in multi-hop wireless networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 23–32, New York, NY, USA, 2008. ACM.

[110] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems. *SIAM J. Comput.*, 28:1298–1309, March 1999.

[111] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path splicing. *SIGCOMM Comput. Commun. Rev.*, 38(4):27–38, 2008.

[112] A. Nasipuri and S. Das. Demand multipath routing for mobile ad hoc networks. In *Proceedings of the 8 th Annual IEEE Internation Conference on Computer Communications and Networks (ICCCN)*, pages 64–70, October 1999.

[113] B. Nath and D. Niculescu. Routing on a curve. *SIGCOMM Comput. Commun. Rev.*, 33(1):155–160, 2003.

[114] M. Newman. Astrophysics collaborations. In *Proc. Natl. Acad. Sci.*, volume 98, pages 404–409, 2001.

[115] E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proc. IEEE INFOCOM*, pages 170–179, 2002.

[116] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[117] Y. Ohara, S. Imahori, and R. V. Meter. Mara: Maximum alternative routing algorithm. In *Proc. IEEE INFOCOM*, 2009.

[118] C. H. Papadimitriou and D. Ratajczak. On a conjecture related to geometric routing. *Theor. Comput. Sci.*, 344(1):3–14, 2005.

[119] S. Patil and S. R. Das. Serial data fusion using space-filling curves in wireless sensor networks. In *Proceedings of IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, pages 182–190, 2004.

[120] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[121] P. P. Pham and S. Perreau. Performance analysis of reactive shortest path and multipath routing mechanism with load balance. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, Vol. 1 (2003), pp. 251-259 vol.1*, 2003.

[122] P.Henrici. *Applied and Computational Complex Analysis*, volume 3. Wiley, New York, 1986.

[123] R. Poovendran and L. Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *ACM Journal of Wireless Networks (WINET)*, 13, January 2005.

[124] L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica. Balancing traffic load in wireless networks with curveball routing. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 170–179, New York, NY, USA, 2007. ACM.

[125] L. Qian, N. Song, and X. Li. Detection of wormhole attacks in multi-path routed wireless ad hoc networks: a statistical analysis approach. *J. Netw. Comput. Appl.*, 30(1):308–330, 2007.

[126] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *J. Comp. and System Sciences*, pages 130–143, 1988.

[127] P. Raghavan and C. D. Thompson. Provably good routing in graphs: regular arrays. In *Proceedings of the 17th annual ACM Symposium on Theory of Computing*, pages 79–87, 1985.

[128] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. Disjoint multipath routing using colored trees. *Comput. Netw.*, 51(8):2163–2180, 2007.

[129] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108, 2003.

[130] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A geographic hash table for data-centric storage in sensornets. In *Proc. 1st ACM Workshop on Wireless Sensor Networks ands Applications*, pages 78–87, 2002.

[131] S. Ray, R. Guérin, K.-W. Kwong, and R. Sofia. Always acyclic distributed path computation. *IEEE/ACM Transactions on Networking*, 2009.

[132] L. Reddeppa Reddy and S. V. Raghavan. Smort: Scalable multipath on-demand routing for mobile ad hoc networks. *Ad Hoc Netw.*, 5(2):162–188, 2007.

[133] C. Reichert, Y. Glickmann, and T. Magedanz. Two routing algorithms for failure protection in IP networks. In *Proc. ISCC*, 2005.

[134] C. Reichert and T. Magedanz. Topology requirements for resilient IP networks. In *Proc. 12th GI/ITG Conf. on Meas., Mod. and Eval. of Comp. and Comm. Sys*, 2004.

[135] H. Sagan. *Space-Filling Curves*. Springer-Verlag, New York, 1994.

[136] O. Sandberg. Distributed routing in small-world networks. In *Proc. of Algorithm Engineering and Experiments (ALENEX)*, 2006.

[137] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu. Greedy routing with guaranteed delivery using ricci flows. In *Proc. of the 8th International Symposium*

on Information Processing in Sensor Networks (IPSN'09), pages 97–108, April 2009.

[138] R. Sarkar, W. Zeng, J. Gao, and X. D. Gu. Covering space for in-network sensor data storage. In *Proc. of the 9th International Symposium on Information Processing in Sensor Networks (IPSN'10)*, pages 232–243, April 2010.

[139] R. Sarkar, X. Zhu, and J. Gao. Double rulings for information brokerage in sensor networks. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 286–297, September 2006.

[140] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *ACM Workshop on Wireless Security (WiSe 2003)*, September 2003.

[141] A. Scaglione and Y. W. Hong. Opportunistic large arrays: Cooperative transmission in wireless multihop ad hoc networks to reach far distances. *IEEE Transactions on Signal Processing*, 51(8), 2003.

[142] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler. Improving the resilience in IP networks. In *Proc. HPSR*, 2003.

[143] C. Sengul and R. Kravets. Bypass routing: An on-demand local recovery protocol for ad hoc networks. *Ad Hoc Netw.*, 4(3):380–397, 2006.

[144] R. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, May 2003.

[145] O. Simsek and D. Jensen. Navigating networks by using homophily and degree. In *Proceedings of the National Academy of Sciences*, pages 12758–12762, September 2008.

[146] K. Stephenson. *Introduction To Circle Packing*. Cambridge University Press, 2005.

[147] I. Stojmenovic. A routing strategy and quorum based location update scheme for ad hoc wireless networks. Technical Report TR-99-09, SITE, University of Ottawa, September, 1999.

[148] H. Suzuki and F. Tobagi. Fast bandwidth reservation scheme with multi-link and multi-path routing in atm networks. In *Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2233 –2240 vol.3, may 1992.

[149] W. Thurston. The finite riemann mapping theorem. In *Invited talk, An International Symposium at Purdue University on the occasion of the proof of the Bieberbach conjecture*, March 1985.

[150] W. P. Thurston. *Geometry and Topology of Three-Manifolds*. Princeton lecture notes, 1976.

[151] J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32:425, 1969.

[152] A. van de Rijt, X. Ban, and R. Sarkar. Effective networking when connections are invisible: Comment on reagans and zuckerman. *Industrial and Corporate Change*, 17:945–952, 2008.

[153] Z. Vincze and R. Vida. Multi-hop wireless sensor networks with mobile sink. In *CoNEXT'05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 302–303, New York, NY, USA, 2005. ACM Press.

[154] R. Viswanathan and P. Varshney. Distributed detection with multiple sensors i. fundamentals. *Proceedings of the IEEE*, 85(1):54 –63, jan 1997.

[155] H. Q. Vo, Y. Y. Yoon, and C. S. Hong. Multi-path routing protocol using cross-layer congestion-awareness in wireless mesh network. In *ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 486–490, New York, NY, USA, 2008. ACM.

[156] W. Wang and B. Bhargava. Visualization of wormholes in sensor networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 51–60, New York, NY, USA, 2004.

[157] Y. Wang, J. Gao, and J. S. B. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 122–133, September 2006.

[158] R. Williams, E. Berlow, J. Dunne, A. Barabasi, and N. Martinez. Two degrees of separation in complex food webs. *Proceedings of the National Academy of Sciences*, 99(20):12913–12916, 2002.

[159] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *Network, IEEE*, 20(3):41–47, May-June 2006.

[160] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 148–159, 2002.

[161] X. Yu, X. Ban, R. Sarkar, W. Zeng, X. D. Gu, and J. Gao. Spherical representation and polyhedron routing for load balancing in wireless sensor networks. In *Proc. of 30th Annual IEEE Conference on Computer Communications (INFOCOM'11)*, April 2011.

[162] D. Zappala. Alternate path routing for multicast. *IEEE/ACM Trans. Netw.*, 12(1):30–43, 2004.

[163] W. Zeng, R. Sarkar, F. Luo, X. D. Gu, and J. Gao. Resilient routing for sensor networks using hyperbolic embedding of universal covering space. In *Proc. of the 29th Annual IEEE Conference on Computer Communications (INFOCOM'10)*, pages 1694–1702, March 2010.

[164] F. Zhang, A. Jiang, and J. Chen. Robust planarization of unlocalized wireless sensor networks. In *Proc. of INFOCOM 2008*, pages 798–806, 2008.

[165] Z. Zhou and J.-H. Cui. Energy efficient multi-path communication for time-critical applications in underwater sensor networks. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 221–230, New York, NY, USA, 2008. ACM.

[166] X. Zhu, R. Sarkar, J. Gao, and J. S. B. Mitchell. Light-weight contour tracking in wireless sensor networks. In *Proceedings of the 27th Annual IEEE Conference on Computer Communications (INFOCOM'08)*, pages 960–967, May 2008.