7-27-2010

Algorithms and Models for Problems in Networking

Michael Dinitz
Carnegie Mellon University

Algorithms and Models for Problems in Networking

Michael Dinitz

CMU-CS-10-136

July 27, 2010

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Anupam Gupta, Chair
Avrim Blum
Bruce Maggs
Matthew Andrews

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Partially supported by an NSF Graduate Research Fellowship, an ARCS Foundation scholarship, and NSF awards CCF-0448095 and CCF-079022

Keywords: Algorithms, Networking, Spanners, Routing, Wireless, Capacity, BGP

This thesis is dedicated to my family. Mom, Dad, Amy, and Tom: I couldn't have done it without your help, support, and encouragement.

Abstract

Many interesting theoretical problems arise from computer networks. In this thesis we will consider three of them: algorithms and data structures for problems involving distances in networks (in particular compact routing schemes, distance labels, and distance oracles), algorithms for wireless capacity and scheduling problems, and algorithms for optimizing iBGP overlays in autonomous systems on the Internet. While at first glance these problems may seem extremely different, they are similar in that they all attempt to look at a previously studied networking problem in new, more realistic frameworks. In other words, they are all as much about new *models* for old problems as they are about new *algorithms*. In this thesis we will define these models, design algorithms for them, and prove hardness and impossibility results for these three types of problems.

Acknowledgments

This thesis would have been impossible without the guidance of my advisor, Anupam Gupta. While we may not have written many papers together, he has been an invaluable mentor who always has good ideas and interesting thoughts. I was also fortunate to have the opportunity to work with the great theory group at Bell Labs, especially Matthew Andrews and Gordon Wilfong. My interactions with them have helped shape my research interests, and papers with them form a considerable portion of this thesis.

Contents

List of Figures

Chapter 1

Introduction

There is no question that much of the algorithmic work in the theoretical computer science community has been at least indirectly motivated by problems arising from computer networks. After all, one of the standard uses of graphs is to represent such networks. However, this motivation has in many cases been one-way: theoreticians get motivation from networking, but after the normal theoretical abstraction the problems they end up working on become so theoretical that the original networking becomes almost unrecognizable. This can happen in multiple ways: for example, perhaps in the abstract theoretical problem it can be shown that basically nothing can be done (the problem is hard to approximate, or certain necessary structures do not exist, or some other negative result) but in practice many things seem to work fine. This is the standard argument for analyses like *average-case* analysis, in which we stay in the same model but try to prove results other than worst case bounds. Or perhaps the model of the network used is fundamentally flawed, so any result we prove is a statement only about the model, not about reality. In this thesis we examine networking problems where the previous theoretical work falls into these categories and we attempt to make the theory more realistic. We also consider some networking problems where there is no previous theoretical work.

1.1 Algorithms and Data Structures for Network Distances

For problems from the first category, in which the theoretical lower bounds seem too strong compared to what is possible in practice, we consider a variety of problems having to do with distances in computer networks. There are many notions of "distance" in networks, for example the latency between points, the bandwidth, the hop count, the IGP distance, etc. For some applications these distances are crucial (e.g. routing), and for others they are nice to know for optimization reasons (e.g. content-distribution networks). Some of these distance notions are truly metric spaces (IGP distances), but for those that do violate triangle inequality there is experimental evidence to suggest that these violations are relatively rare [75], so we will assume throughout this work that whatever notion of distance we are working with obeys the triangle inequality. In this thesis we will examine four distance-based problems: network spanners, distance oracles, distance labels, and compact routing.

Given a metric $M = (V, d)$ (presumably arising from some network distances), an α-spanner

of M is a weighted graph $G = (V, E, w)$ in which $d(u, v) \leq d_G(u, v) \leq \alpha d(u, v)$ for all $u, v \in V$ (where d_G is the distance in G according to the weights w). Less formally, G is an α-spanner if it preserves all distances in the metric up to a factor of α. This is obviously achievable with $\alpha = 1$ by simply making G a complete graph with $w(\{u, v\}) = d(u, v)$, but in many situations what we want is a *sparse* graph that preserves distances well. So the study of spanners is the study of the tradeoff between α (the *stretch* or *distortion* of the spanner) and $|E|$ (the *size* or *sparsity*).

There is a large body of work on spanners dating back to the late 1980's [8, 10, 14, 29, 36, 73, 74, 82] and still going strong [16, 17, 22, 38, 84]; see, e.g., [71] for many of the results. Spanners were initially studied due to applications in network synchronization, but since then they have found myriad uses in network design and routing, as well as in many places where it is advisable to compactly store a graph without changing the distances much, such as in speeding up shortest path computations. Apart from the literature on finding spanners of general graphs, there has also been a large body of work on Euclidean spanners (see, e.g., [10, 29, 49]), as well as work on spanners for doubling metrics [26, 52].

Similar problems that have a more data structure-like flavor include constructing good *distance oracles* and *distance labels*. A distance oracle is an algorithm that preprocesses a metric space $M = (V, d)$, stores some data structure, and then answers pairwise distance queries. The relevant parameters are the amount of preprocessing time, the space used to store the data structure, the time necessary to answer a query, and the accuracy of the answer. Most previous work on distance oracles, including the work in this thesis, focuses mainly on two of these parameters: the size of the data structure and the accuracy of the response. It is obvious that exact distance oracles require either large data structures (e.g. the entire distance matrix) or large query times (e.g. a shortest path computation), but fast and compact oracles with bounded stretch have been given for both general graphs [82] and for special classes of graphs [30, 50, 51, 81].

A distance labeling scheme can be thought of as "distributed" distance oracle: a distance labeling scheme is an algorithm that, given a metric space, assigns a label to every point in the metric such that the distance between two points can be approximated by a computation that takes as input only the two labels. The most important parameters for these schemes are the size of the labels and the accuracy of the response, although the time necessary for the computation is also interesting. There is significant previous work on distance labels in the theory community (e.g. [44, 72, 79, 87]) as well as in the practical networking community, where they usually go by the name "network coordinate systems" and are generally restricted to labels corresponding to the coordinates in some low-dimensional embedding of the metric into real space [32, 88]. The practical uses of distance labels are obvious: any network algorithm that uses overlays or makes connections between nodes that are not neighbors in the routing graph can take advantage of the distance information provided by labels to optimize the overlay. In fact, content distribution networks that use overlays were the original motivation for the work on network coordinate systems.

Finally we consider the problem of compact routing. Of all of the tasks that must be performed by a distributed system, routing messages is among the most fundamental. Whether the system is the Internet, a LAN, or even a large multiprocessor computer, messages have to be routed from sources to destinations. A natural model for this problem is a weighted graph in which the cost of routing a message equals the total cost of the path taken. In this model the best that we can hope to do would be to route along shortest paths between points. This naturally

leads to the main way of measuring routing efficiency: the *stretch* (also called the *distortion*) is the maximum over all source-destination pairs of the ratio of the cost of routing from the source to the sink and the length of the actual shortest path between the source and the sink. The obvious solution is to keep at each node a routing table with entries for all of the other $n-1$ nodes. But this solution takes $\Omega(n \log n)$ space, which for large n is impractical. Thus we have the fundamental tradeoff between stretch and space that compact routing schemes try to address. This has been a very active area of research in the distributed computing and theoretical networking community, and previous work includes results for general graphs by Thorup and Zwick [83] and Abraham, Gavoille, and Malkhi [6] and results on special classes of graphs such as trees [42, 43, 65] and doubling metrics [4, 26, 59, 60].

For all of these problems strong lower bounds are known (assuming the well-known girth conjecture of Erdős). But these lower bounds are on the worst case, and are of the form "there exist graphs (or metrics) for which any spanner/oracle/labeling/routing scheme does poorly for at least one pair of vertices". In practice, though, this type of worst case behavior does not seem to arise. For example, there are many distance labeling schemes that have good empirical performance [32, 88]. We attempt to be slightly more realistic not by changing the network model, but by changing the type of result we prove. In this thesis we will give constructions and results that have some *slack*, in that we will prove statements of the form "on any graph or metric, there is a construction that does well on all but an ϵ fraction of pairs of vertices". These types of results will then let us prove average case bounds of the form "on any graph or metric there is a construction such that the average performance (over pairs of vertices) is good". These are clearly more realistic results, and are at least an initial attempt at explaining why certain algorithms do well in practice despite the theoretical lower bounds.

Slightly more formally, note that all of the problems we consider have a notion of stretch or distortion and a notion of size. We say that they have ϵ-slack and stretch α if they have stretch α on all but an ϵ fraction of pairs. The hope is that allowing some slack also allows us to give better size bounds in term of the slack ϵ. We will also consider constructions that are *gracefully-degrading* in the sense that a single construction has ϵ-slack and $\alpha(\epsilon)$ stretch for all ϵ *simultaneously*. This is a much stronger condition than merely requiring ϵ-slack, since it forces one construction for all ϵ rather than allowing different constructions for different ϵ. However, it will allow us to prove stronger statements not involving slack, such as a construction having low *average* distortion.

This thesis includes the work initially presented in [27] on spanners, distance oracles, and distance labels, and the work in [34] on compact routing schemes. For example, we prove the following theorems about spanners:

Theorem 1.1 *For any metric on n points, for any $0 < \epsilon < 1$, for any integer $k > 0$, there exists a graph $H(\epsilon)$ that is a $(12k-1)$-spanner with ϵ-slack of size $n + O((\frac{1}{\epsilon})^{1+1/k})$. Furthermore, there is a graph H with $O(n)$ edges that is a $O(\log \frac{1}{\epsilon})$-stretch gracefully degrading spanner with $O(1)$ average stretch and $O(\log n)$ worst case stretch.*

Analogous theorems can be proved for distance oracles and distance labels. Note that this construction allows us to essentially bypass known lower bounds: assuming a girth conjecture of Erdős, there are graphs for which it is not possible to construct a spanner with stretch less than $2k-1$ and at most $n^{1+1/k}$ edges. This obviously implies that we cannot construct spanners with

a linear number of edges unless we have logarithmic stretch. But Theorem 1.1 bypasses this and gives us linear size spanners with constant stretch as long as we only make the stretch guarantee on a $1 - \epsilon$ fraction of the pairs.

Our results for compact routing are a little more complicated, as they depend on the details of the model. More specifically, one issue that influences the quality of routing schemes is the power they have to rename nodes. There is the *name-independent* model, in which schemes have to route without changing the names of any nodes, and the *name-dependent* model, in which schemes are allowed to assign labels to vertices and can route using information in the labels. Somewhat surprisingly, for general graphs there does not seem to be a large difference between these two models. In particular, the best general name-dependent scheme is the one devised by Thorup and Zwick in [83] which uses $O(kn^{1/k})$ space and $o(k \log^2 n)$ size labels to get stretch $4k - 5$, while the best name-independent scheme is due to Abraham, Gavoille, and Malkhi [6], who give a scheme that uses $O(k^2 n^{1/k} \log^3 n)$ space and has stretch $O(k)$. There is also a difference between *designer-port* schemes, which can assign the ports that connect a vertex to its neighbors to be any permutation, and *fixed-port* schemes which cannot renumber the ports and thus must assume that they have been assigned adversarially. While there does not appear to be a large amount of literature on this difference, it is one that proves crucial when we start allowing slack.

Note that these two distinctions (name-dependent vs. name-independent, designer port vs. fixed port) give rise to four distinct models, each of which allows the routing scheme to have a different amount of power. In the name-dependent model (in either port model) we can construct routing schemes with slack that give results analogous to those on spanners, i.e. we can construct extremely small routing tables and still get constant stretch on all but an ϵ fraction of pairs. In the name-independent model this is possible for designer ports (although the bounds are somewhat weaker), but we prove that allowing even a constant amount of slack does not help in the name-independent fixed-port model.

1.2 Wireless Network Capacity

A different problem that we consider in the realm of theoretical networking is maximizing the transmission capacity of wireless networks. In the basic model we are given a collection of transmitter/receiver pairs in the Euclidean plane, and the goal is to maximize the number of successful instantaneous transmissions. Maximizing transmission capacity has been studied in many contexts, and while many variants have been considered, there are two axes along which much of the work can be partitioned. The first axis is random vs. arbitrary networks. If we consider random networks, then the goal is typically to give bounds on the expected capacity, and study how this changes with the density of the network (or with some other interesting parameter). Another option, which is what we consider in this work, is to study arbitrary or worst-case topologies. In this setting it makes no sense to study the "average" capacity, since that could depend heavily on the actual structure of the network. Instead, the goal is to study the problem of maximizing capacity as an optimization problem, and give hardness results, centralized algorithms, and distributed protocols given an arbitrary network as input.

The second axis is the protocol model vs. the physical model, and is concerned with how

we model interference and define a successful transmission. In the *protocol model* there is some interference graph on the transmitters, and a transmission is successful if and only if none of the neighbors of the transmitter in this graph also chose to transmit. It is obvious from this definition that maximizing network capacity is the same problem as finding a maximum independent set in the interference graph, which is a famous and well-studied problem in its own right. In the context of this problem, further assumptions are usually made about the structure of the interference graph, since physical constraints make it unlikely that this graph is totally arbitrary. One typical assumption is that it is a *unit disk graph* (abbreviated UDG), which means that two transmitters interfere if and only if they are at distance at most 1 in the Euclidean plane. There has also been a considerable line of work on weakening this assumption or on variants of it, including the Tx model of [91] and the growth-bounded model of [77] and [64].

In the *physical model*, on the other hand, we do not assume the existence of an interference graph. Instead we let every transmitter choose a power to broadcast at, give a rule for how that power fades with distance, and say that a transmission has been successful if and only if the received signal divided by the sum of the interference and background noise is at least some threshold. This model is significantly more complicated than the protocol model, for a variety of reasons. In the protocol model the success of a transmission depends only on the OR of its neighbors; if any of its neighbors transmit then it fails, irrespective of whether one or ten of them transmitted, and any number of transmitters outside of its neighborhood can transmit without affecting its success. But in the physical model interference accumulates and normally spreads out to infinity, so not only is the decision function more complicated than an OR of neighbors it actually depends on every transmitter in the entire network. While not all of the assumptions in the physical model are absolutely true, it is commonly thought to be a more accurate model of reality than the protocol model.

Furthermore, there is a difference between *centralized* and *distributed* algorithms. While studying the fundamental computational problem is interesting, in many (perhaps most) real world situations there is no central authority to run the algorithm and tell all of the transmitters what to do. Ideally each transmitter would make its own decisions about whether to broadcast (and in the physical model, how much power to use). In the protocol model, since we have an interference graph we can simply abstract out to the graph and run a normal distributed protocol on this graph, and indeed this problem is usually classified under "distributed maximum independent set". In the physical model, however, there is no underlying communication or interference graph so coordination, even among very close transmitters, is more complicated. And even in the protocol model, using standard models for distributed algorithms are problematic: do transmitters really know their neighbors? Can they really send different messages to different transmitters? Can a transmitter really receive multiple messages at the same time? In this thesis we will examine both the centralized and the distributed problem.

There has been very little study of the complexity of calculating the maximum possible capacity in arbitrary networks in the physical model. We believe that addressing this question is important for two reasons. First, although analyzing capacity in random networks is important for determining what level of transmissions will be possible in completely unstructured networks, there are many situations where the network will have some sort of structure and the transmission capacity may be very different than what is possible in random networks. In these cases we believe that knowing the complexity of calculating the maximum number of transmissions is

important. Second, as is well-known the Unit Disk Graph model does not capture many features of wireless networks. One reason for this is that receivers will hear interference from all other transmitters, even if they are far away. A more important reason is that interference at a receiver is a *cumulative* effect of multiple transmitters whereas in the Unit Disk Graph model interference is simply a local binary property.

This thesis includes the results of [9] on the centralized problem and the results of [35] on the distributed version. In particular, we show that it is NP-hard to maximize the number of simultaneous successful connections in the physical mode, but on the other hand there is a polynomial time algorithm that gives an $O(\log d_{\max})$-approximation, where $d_{\max}$ is the largest distance of any transmitter-receiver pair. We also design a game played by the transmitters with the property that any Nash equilibrium has an expected number of successful transmissions that is within $O(d_{\max}^{2\alpha})$ of the optimum, where α is a parameter of the physical model known as the path-loss exponent. Finally, we show that this game also has the property that if every transmitter uses an algorithm with the *no-regret* property then the average number of successful transmissions is also within $O(d_{\max}^{2\alpha})$ of the optimum. We will define no-regret algorithms in Section 3.5, but it is known that no-regret algorithms exist for every game. So in other words, we design a distributed algorithm (every transmitter uses no-regret algorithms) that is based on a *game-theoretic* analysis rather than a purely *algorithmic* analysis. We believe that this style of analysis is one of the main contributions of this work, and we hope that this technique for designing distributed algorithms will prove useful for other problems.

1.3 Constrained Connectivity and iBGP

The final problem that we consider is back in the realm of wired networks, but is based on real life protocols rather than the idealized settings of compact routing, spanners, and the other problems involving network distances. In particular, we will be looking at interdomain routing on the Internet and how routes are internally distributed using the *interior Border Gateway Protocol* (iBGP). This is the version of the interdomain routing protocol BGP [78] used by routers within a subnetwork to announce routes to each other that have been learned from outside the subnetwork.

1.3.1 iBGP Problem Definition

The Internet consists of a number of interconnected subnetworks called Autonomous Systems (ASes). As described in [15], the way that routes to a given destination are chosen by routers within an AS can be viewed as follows. Routers have a ranking of routes based on economic considerations of the AS. Without loss of generality, in what follows we assume that all routes are equally ranked. Thus routers must use some tie-breaking scheme in order to choose a route from amongst the equally ranked routes. Tie-breaking is based on traffic engineering considerations and in particular, the goal is to get packets out of the AS as quickly as possible (called *hot-potato routing*).

An AS attempts to achieve hot-potato routing as follows. The routers that initially know of a route are called *border routers*. (These initial routes are those learned by the border routers from routers outside the AS.) The border router that initially knows of a route is said to be the *egress*

router of that route. Each border router knows of at most one route per destination. Thus an initial set of routes F defines a set of egress routers X_F where there is a one-to-one relationship between routes in F and routers in X_F. The AS has an underlying physical network with edge weights (e.g., IGP distances or OSPF weights). The *distance* between two routers is then defined to be the length of the shortest path (according to the edge weights) between them. Given a set of routes, a router will rank highest the one whose egress router is closest according to this definition of distance. The *signaling graph* H is an overlay network whose nodes represent routers and whose edges represent the fact that the two routers at its endpoints use iBGP to inform one another of their current chosen route. The endpoints of an edge in H are called *iBGP neighbors*. A path in H is called a *signaling path*. Finally, iBGP can be thought of as working as follows. In an asynchronous fashion, each router considers all the latest routes it has heard about from its iBGP neighbors, chooses the one with the closest egress router and tells its iBGP neighbors about the route it has chosen. This continues until no router learns of a route whose egress router is closer than that of its currently chosen route. When this process ends the route chosen by router r is denoted by $R(r)$. Let $P(r)$ be the shortest path from r to $E(r)$, the egress router of $R(r)$. When a packet arrives at r, it sends it to the next router r' on $P(r)$, r' in turn sends the packet to the next router on $P(r')$ and so on. Thus if $P(r')$ is not the subpath of $P(r)$ starting at r' then the packet will not get routed as r expected.

Thus there are two properties that a signaling graph H should have if the AS is to achieve hot-potato routing:

1. **complete visibility**: each router r hears about (and hence chooses as $R(r)$) the route in F whose egress router $E(r)$ is closest to r from amongst all routers in X_F and
2. **no deflections**: for each router r, all routers r' along $P(r)$ have $E(r') = E(r)$.

For a given X_F, we say that a signaling graph is *correct for* X_F if it satisfies the goals of complete visibility and no deflections. If it satisfies these goals for all possible X_F then we say that the signaling graph is *correct*.

Clearly if H is the complete graph then H is correct. However the complete graph is not practical and so network managers have adopted various configuration techniques to reduce the size of the signaling graph [18, 85]. Unfortunately these methods do not guarantee correct signaling graphs [15, 47]. Thus our goal is to determine correct signaling graphs with fewer edges than the complete graph.

It is easy to check that complete visibility implies no deflections. Therefore a signaling graph is correct if and only if it satisfies complete visibility. A natural question is to minimize the number of edges in the signaling graph or to minimize the maximum number of iBGP neighbors for any router while guaranteeing correctness. We define IBGP-SUM to be the problem of finding a correct signaling graph with the fewest edges. Similarly we define IBGP-DEGREE to be the problem of finding a correct signaling graph with the minimum possible maximum degree.

1.3.2 Safe Sets and Constrained Connectivity

The definitions of IBGP-SUM and IBGP-DEGREE presented above are the ones that we would like to use, but they are somewhat difficult to work with. This is true for two reasons: first, even if we are given a signaling graph H and an initial set of routes F, how do we check for complete

visibility? The obvious way is to simulate iBGP on that H and F, but since we want to examine this problem from an optimization standpoint we would like to have a more structural definition of correctness that does not require protocol simulation. Second, note that the definition of a correct signaling graph is that it satisfies complete visibility for every possibly X_F. So the corresponding decision problem would be to decide if there exists a graph H such that H has at most k edges and has complete visibility for all possible X_F. But this is a Σ_2 statement, where we have an existential quantifier followed by a universal quantifier followed by a polynomial time evaluable predicate. Unless the polynomial hierarchy collapses, Σ_2 is a harder class even than NP. So *a priori* the iBGP problems might be even more difficult than the NP-complete problems.

Fortunately this turns out not to be the case: the decision versions of the iBGP problems are in NP. Moreover, in the process of proving this we also handle the first problem and give a structural definition of correctness. Given the underlying network (or at least the IGP distances), we can associate a special *safe set* of vertices $S(x, y) \subseteq V$ with every pair of vertices (x, y). The precise definitions of the safe sets will be presented in Section 4.1.1, but they are based only on the network distances. It turns out that H is correct if and only if for all $(x, y) \in V \times V$ there is a path between x and y in H that is completely contained in $S(x, y)$. We call this the *safe set definition* of iBGP, and because of the if and only if statement of the theorem we know that it is completely equivalent to the more normal and application-centered definition. In this thesis we will use the safe set definition to prove that both iBGP-SUM and iBGP-DEGREE cannot be approximated better than $\Omega(\log n)$ (unless P=NP), and we will give two different but related algorithms based on a natural LP relaxation that are $\tilde{O}(n^{2/3})$-approximations.

While for our purposes the safe sets $S(x, y)$ were defined in such a way as to make the safe set definition equivalent to the original iBGP definition, the safe set definition itself suggests a few generalizations which are a bit more theoretical, although still have their own applications. The two natural relaxations are to allow arbitrary safe sets as part of the input to the problem (instead of safe sets based on the network distances), and to allow an input graph G that the output signaling graph must be a subgraph of. When we make both of these relaxations we get a new network design problem that we call *Constrained Connectivity*: given a graph G and a set $S(x, y) \subseteq V$ for all pairs of vertices in V, find a subgraph H of G of minimum size (or minimum max degree) such that every pair of nodes x, y has a path between them in H that is completely contained in $S(x, y)$. If we only allow the first relaxation, then we simple have constrained connectivity on the input graph G is the complete graph K_n; we will refer to this version of the problem as *Constrained Connectivity on* K_n.

While the iBGP problem is not nearly as general as constrained connectivity, there is an obvious security application that, to the best of our knowledge, has not previously been considered. Suppose we have n players who wish to communicate with each other but they do not all trust one another with messages they send to others. That is, when u wishes to send a message to v there is a subset $S(u, v)$ of players that it trusts to see the messages that it sends to v. Of course, if for every pair of players there were direct communication channels between the two players, then there would be no problem. But suppose there is a cost to protect communication channels from eavesdropping or other such attacks. Then a goal would be to have a network of fewer than $O(n^2)$ communication channels that would still allow a route from each u to each v with the route completely contained within $S(u, v)$. Thus this problem defines a Constrained Connectivity problem.

In this thesis we show that both the sum and the degree versions of the general Constrained Connectivity problem are extremely difficult to approximate: under plausible complexity assumptions it is impossible to approximate either one to better than $2^{\log^{1-\epsilon} n}$ for any constant $\epsilon > 0$. Moreover, the natural LP relaxation (which is simply the obvious generalization of the LP for the iBGP problem) has a polynomial integrality gap. On the other hand, the $\tilde{O}(n^{2/3})$-approximation algorithm that we originally designed for iBGP generalizes to Constrained Connectivity on K_n. We will also show some easier settings (for example, if all of the safe sets are hierarchical) in which Constrained Connectivity can actually be solved in polynomial time.

Chapter 2

Network Distances

2.1 Introduction

In this chapter we discuss four data structures related to distances in computer networks: network/graph spanners, distance oracles, distance labels, and compact routing schemes. Recall that given a metric $M = (V, d)$ (presumably arising from some network distances), an α-spanner of M is a weighted graph $G = (V, E, w)$ in which $d(u, v) \leq d_G(u, v) \leq \alpha d(u, v)$ for all $u, v \in V$ (where d_G is the distance in G according to the weights w). Less formally, G is an α-spanner if it preserves all distances in the metric up to a factor of α (which is called the *stretch* of the spanner). Our goal is to construct spanners with both low stretch and few edges. Similarly, a *distance oracle* for M is an algorithm that preprocesses M, stores some data structure, and then approximately answers pairwise distance queries. The relevant parameters are the amount of preprocessing time, the space used to store the data structure, the time necessary to answer a query, and the accuracy of the answer. We are mainly be concerned with the size of the data structure and the accuracy of the response. We define the stretch of a distance oracle in the obvious way analogous to the stretch of a spanner: an oracle has stretch α if on every pair of vertices it returns a value that is at least as large as the true distance and at most α times the true distance. A distance labeling scheme is in many way a distributed distance oracle: a labeling assigns a label to every vertex so that an approximation of the distance between two points can be computed just from the labels of the two points. The most important parameters are the size of the labels and the accuracy of the approximation, where we measure the accuracy using the stretch, defined in the obvious way similar to the stretch of oracles and spanners.

Compact routing schemes are an abstraction of the normal packet-switched network routing schemes that are commonly used in networks today. They assign some data structure to each vertex (e.g. the normal routing tables so heavily used in networking) and when a packet arrives they use this data together with other available information (such as the packet header or the incoming port) to figure out what outgoing link the packet should be forwarded on. The stretch of a compact routing scheme is defined in the obvious way as the maximum over pairs of nodes of the distance taken by a packet going from one node to the other divided by the shortest possible distance. As always, we want to study the tradeoff between the size of the scheme (i.e. the size of the data structures stored at each node) and the stretch it incurs.

There has been a significant amount of work on all of these problems. Our main contribution is introducing the notion of *slack* to these problems, which allows us to ignore an ϵ fraction of the pairs in the hope of getting much stronger guarantees for the remaining $1 - \epsilon$ fraction of the pairs. This notion has its roots in the study of *metric embeddings*, which has been a central pursuit in algorithms research in the past decade. Loosely speaking, an embedding is a map from a metric space into a "simpler" metric space so that distances between points are changed by at most a small factor. More formally, given a *target class* $\mathcal{C}$ of metrics, an *embedding* of a finite metric space $M = (V, d)$ into the class $\mathcal{C}$ is a new metric space $M' = (V, d')$ such that $M' \in \mathcal{C}$. Most of the work on embeddings has used *distortion* as the fundamental measure of quality; the distortion of an embedding is the worst multiplicative factor by which distances are increased by the embedding. This is equivalent to the notion of stretch in our problems, and we will use the terms "distortion" and "stretch" interchangeably. Given the metric $M = (V, d)$ and the class $\mathcal{C}$, one natural goal is to find an embedding $\varphi((V, d)) = (V, d') \in \mathcal{C}$ such that the distortion of the map φ is minimized. Note that this notion of embedding includes concepts such as spanners, in which the class $\mathcal{C}$ is the class of metrics generated by sparse graphs.

In the theoretical computer science community the popularity of the notion of distortion/stretch has been driven by its applicability to approximation algorithms: if the embedding $\varphi : (V, d) \to (V, d')$ has a distortion of D, then the cost of solutions to some optimization problems on (V, d) and on (V, d') can only differ by some function of D; this idea has led to numerous approximation algorithms [54]. Seminal results in embeddings include the $O(\log n)$ distortion embeddings of arbitrary metrics into ℓ_p spaces [23], the fact that any metric admits an $O(\log n)$ stretch spanner with $O(n)$ edges [8], and that any metric can be embedded into a distribution of trees with distortion $O(\log n)$ [41]. These three results are known to be tight.

In parallel to this theoretical work, more applied communities have had much recent interest in embeddings (and more generally, but also somewhat vaguely, on problems of finding "simpler representations" of distance spaces). One example is the networking community, where there is much interest in taking the point-to-point latencies between nodes in a network, treating it as a metric space $M = (V, d)$ satisfying the triangle inequality, and finding some simpler representation $M' = (V, d')$ of this resulting metric so that distances between nodes can be quickly and accurately computed in this "simpler" metric M'. Despite this similarity of interest, many of the theoretical results mentioned above have not been used widely in these applications; the logarithmic guarantees on the distortion are often deemed unacceptable. Indeed, the notion of distortion turns out to be a demanding and inflexible objective function, and the empirical works are often happy with guarantees of the following form: they allow some small fraction of the distances to be distorted by *arbitrary* amounts, but then seek very strong guarantees on the distortion incurred by the remaining large fraction of the distances. E.g., in the networking application above, we would be happy if *most* inter-node distances were correct and only a small fraction of distances would be estimated poorly.

To remedy the situation, Kleinberg, Slivkins, and Wexler [58] defined the notion of *embeddings with slack*: in addition to the metric $M = (V, d)$ and the class $\mathcal{C}$ in the initial formulation above, we are also given a *slack parameter* ϵ. We now want to find a map $\varphi(M) = (V, d') \in \mathcal{C}$ whose distortion is bounded by some quantity $D(\epsilon)$ on all but an ϵ fraction of the pairs of points in $V \times V$. Note that we allow the distortion on the remaining ϵn^2 pairs of points to be arbitrarily large. The line of work starting with their paper, and furthered by Abraham et al. [2] and [3]

showed that very strong results were indeed possible: in fact, when allowed constant slack, one could get constant-distortion constant-dimensional embeddings. Given these results for embeddings into normed spaces, it is natural to ask whether one can obtain similar results for spanners and related constructs such as distance oracles, distance labelings, and compact routing schemes. In particular, all of these have a notion of stretch or distortion and a notion of size; we say that they have ϵ-slack and stretch α if they have stretch α on all but an ϵ fraction of pairs. This should presumably let us give better size bounds in term of the slack parameter ϵ, and that is exactly what we do in this chapter.

We also discuss a different way of bypassing the lower bounds on spanners, by designing approximation algorithms or per-instance guarantees. The lower bound on spanners is based on the existence of graphs that do not have good spanners, so to get around them we make guarantees of the form "this algorithm returns a spanner that is close in size to the best possible spanner". Our algorithms are the best known for the directed spanner problem, and are detailed in Section 2.6

2.2 Definitions and Slack Basics

All metric spaces we consider are finite and the graphs we consider are undirected. Let (V, d) be a metric space, where $n = |V|$. The *ball* $B(x, r) = \{y \in V \mid d(x, y) \leq r\}$ is the set of points at distance at most r from x. For $0 < \epsilon < 1$, let $R(x, \epsilon)$ be the minimum distance r such that $|B(x, r)| \geq \epsilon n$. The point y is *ϵ-far* away from point x if $d(x, y) \geq R(x, \epsilon)$. We begin by defining slack spanners, and then similarly define slack distance oracles, distance labelings, and compact routing scheme.

Definition 2.1 ((Uniform) Slack Spanner) *Given a metric (V, d) and $0 < \epsilon < 1$, a weighted graph $H = (V, E)$ with each edge $(u, v) \in E$ having weight $d(u, v)$ is an* α-spanner with ϵ-uniform slack *if for all $x, y \in V$ such that y is ϵ-far away from x,*

$$d(x, y) \leq d_H(x, y) \leq \alpha \cdot d(x, y)$$

In general, α can be a function of ϵ and $|V|$. If the metric (V, d) is induced by some weighted graph G, we say that H is a subgraph spanner *if H is a subgraph of G.*

In other words, an ϵ-uniform slack spanner is a graph with the property that for each point x, apart from the ϵn points closest to x, the distances from x to the rest of the points are well approximated. We call this concept "*uniform* slack" to be consistent with previous notation; all references to "ϵ-slack" in this thesis mean "ϵ-uniform slack". For the record, there *is* a non-uniform notion of slack in which the only restriction is that at most an ϵ fraction of the edges can be ignored (see [2, Defn. 1.1] or [58] for details). But we achieve positive results even in the more restrictive uniform model. Also, readers of [3] should note that ϵ-uniform slack embeddings are called "coarsely $(1 - \epsilon)$ partial embeddings" in that paper.

Definition 2.2 (Gracefully degrading spanner) *A weighted graph H is an $\alpha(\frac{1}{\epsilon})$-*gracefully degrading spanner *for the metric (V, d) if for each $0 < \epsilon < 1$, H is an $\alpha(\frac{1}{\epsilon})$-spanner with ϵ-slack. The notion of subgraph spanner also applies analogously.*

We also consider two incomparable notions of "average" distortion; both have been considered previously in the literature, and we will construct spanners that are simultaneously good with respect to both these notions.

Definition 2.3 (Average Distortion) *The* average distortion *of a spanner H for a metric space (V, d) is*

$$\frac{1}{\binom{n}{2}} \sum_{\{x,y\} \in \binom{V}{2}} \frac{d_H(x,y)}{d(x,y)}.$$

Definition 2.4 (Distortion of Averages) *The* distortion of averages *of a spanner H for a metric space (V, d) is*

$$\frac{\sum_{\{x,y\} \in \binom{V}{2}} d_H(x,y)}{\sum_{\{x,y\} \in \binom{V}{2}} d(x,y)}.$$

The corresponding slack definitions of distance oracles, distance labelings, and compact routing schemes are obvious. Recall that a distance oracle is a small data structure which allows fast queries for approximate distances, a distance labeling is an assignment of labels to vertices so that the approximate distance between two points can be computed just from their two labels, and a compact routing scheme consists of a data structure at each vertex (e.g. a routing table) and a forwarding rule that outputs the outgoing port number given a packet's header and the node's routing table (as well as possibly other information like the incoming port number, depending on the model).

Definition 2.5 (Slack Distance Oracle) *A distance oracle O has stretch α with ϵ-slack if $d(x, y) \leq d_O(x, y) \leq \alpha d(x, y)$ for all $x, y \in V$ such that y is ϵ-far from x. Oracle O is* α-gracefully degrading *if it has stretch α with ϵ-slack for all $0 < \epsilon < 1$.*

Definition 2.6 (Slack Distance Labeling) *A distance labeling (L, f) (where L maps vertices to labels and f maps pairs of labels to distances) has stretch α with ϵ-slack if $d(x, y) \leq f(L(x), L(y)) \leq \alpha d(x, y)$ for all $x, y \in V$ such that y is ϵ-far from x. A labeling is* α-gracefully-degrading *if it has stretch α with ϵ-slack for all $0 < \epsilon < 1$.*

Definition 2.7 (Slack Compact Routing Schemes) *For a compact routing scheme R, let $d_R(x, y)$ denote the total distance traveled by a packet starting from x that is destined for y. Then R has stretch α with ϵ-slack if $d(x, y) \leq d_R(x, y) \leq \alpha d(x, y)$ for all $x, y \in V$ such that y is ϵ-far from x. A scheme is* α-gracefully-degrading *if it has stretch α with ϵ-slack for all $0 < \epsilon < 1$.*

Now that the basic definitions are clear, we can start actually building these structures. The basic building block that we will use in all of them is a small sample of points from the metric space V such that each point is "close" to some sample point:

Definition 2.8 (Density Net) *Given a metric space (V, d) with $n = |V|$, and $0 < \epsilon < 1$, an* ϵ-density net *is a set $N \subseteq V$ such that **(1)** for all $x \in V$, there exists $y \in N$ such that $d(x, y) \leq 2R(x, \epsilon)$, **(2)** $|N| \leq \frac{1}{\epsilon}$, and **(3)** $B(x, R(x, \epsilon)) \cap B(y, R(y, \epsilon)) = \emptyset$ for all $x, y \in N$.*

We will often refer to the nodes in N as *centers*. Note that the difference between an ϵ-net and an ϵ-density net is in the notion of "closeness": here the allowed distance from x to its closest center depends on the density of points around x.

Lemma 2.9 *Given a metric space (V, d) and $0 < \epsilon < 1$, an ϵ-density net N can be found in polynomial time.*

Proof: For brevity, let us denote the ball $B(x, R(x,\epsilon))$ by B_x for any point $x \in V$. We begin by ordering the vertices in a list L by nondecreasing value of $R(\cdot,\epsilon)$, breaking ties arbitrarily, and initializing the set N to be empty. We remove the first vertex v from list L. If there exists $u \in N$ such that B_v intersects B_u, then we just discard v; otherwise, we add v to N and remove all vertices in the ball B_v from the list L. We repeat this process until the list L becomes empty and return N as our ϵ-density net.

We first prove the third property. Let $x, y \in N$, and without loss of generality suppose that x was before y on the list L. Then when y was considered, since we added it to N we know that B_y did not intersect B_z for any other $z \in N$ (if it did intersect then the algorithm would have discarded it). So in particular $B_x \cap B_y = \emptyset$.

We next show that the subset N returned satisfies the three properties given in Definition 2.8. Consider any point $x \in V$. We show that there is a point $y \in N$ within distance $2R(x,\epsilon)$ of x. If x is included in N, this is trivially true. Otherwise, either x was at some point the first vertex in list L and get discarded, or x was in some ball B_v and removed from list L. In the former case there is some point $u \in N$ such that B_u intersects B_x. Since u appears before x in the initial list, $R(u,\epsilon) \leq R(x,\epsilon)$ and hence the distance between x and the density-net point u is $d(u,x) \leq R(u,\epsilon) + R(x,\epsilon) \leq 2R(x,\epsilon)$. In the latter case, as v appear before x in the initial list, we also have $R(v,\epsilon) \leq R(x,\epsilon)$ and so $d(x,v) \leq R(v,\epsilon) \leq R(x,\epsilon) \leq 2R(x,\epsilon)$.

To show that $|N| \leq \frac{1}{\epsilon}$, note that by the third property the intersection of B_x and B_y is empty for any two distinct points $x, y \in N$. Since for each $x \in N$, the ball B_x contains at least ϵn points, we conclude that $|N| \leq \frac{1}{\epsilon}$. ■

The following lemma, which we will use regularly, is one of the most useful facts about density nets:

Lemma 2.10 *Let N be an ϵ-density net, let $u, v \in V$ such that v is ϵ-far from u, and let u' and v' be the closest nodes in N to u and v respectively. Then $d(u,u') \leq 2d(u,v)$ and $d(v,v') \leq 3d(u,v)$.*

Proof: By property (1) of density nets, $d(u,u') \leq 2R(u,\epsilon)$. Since v is ϵ-far from u, by definition $R(u,\epsilon) \leq d(u,v)$, and thus $d(u,u') \leq 2d(u,v)$. By the choice of v' and the triangle inequality we know that $d(v,v') \leq d(v,u') \leq d(v,u) + d(u,u') \leq 3d(u,v)$. ■

2.3 Slack Spanners

We will now use the ability to find density nets to construct good slack spanners. We give a general transformation technique to convert $\alpha(n)$-spanners with $T(n)$ edges into ϵ-slack spanners with stretch $(5 + 6\alpha(\frac{1}{\epsilon}))$ and $n + T(\frac{1}{\epsilon})$ edges. Our construction is very simple. We first construct an ϵ-density net N as given in Lemma 2.9. Since $|N| \leq \frac{1}{\epsilon}$, we can construct an $\alpha(\frac{1}{\epsilon})$-spanner $\widehat{H}$ for the set of centers N. Then, for each point $x \in X \setminus N$, we add an edge between x and its closest point in N to $\widehat{H}$; this gives us a spanner H for (V,d).

Theorem 2.11 *The spanner H has $n + T(\frac{1}{\epsilon})$ edges, and is a $(5 + 6\alpha(\frac{1}{\epsilon}))$-spanner with ϵ-uniform slack.*

Proof: First we bound the size of H. Since N has at most $\frac{1}{\epsilon}$ points, the spanner $\widehat{H}$ has at most $T(\frac{1}{\epsilon})$ edges. Moreover, for each point $x \in V \setminus N$, one extra edge is added. Hence, H has at most

$n + T(\frac{1}{\epsilon})$ edges.

Next, we bound the stretch of H. Consider two points u and v such that v is ϵ-far away from u, i.e., $d(u,v) \geq R(u,\epsilon)$. Let u' be a closest point in N to which u is connected to in H (or set $u' = u$ if u is in N), and define v' similarly with respect to v. By Lemma 2.10, $d(u,u') \leq 2d(u,v)$ and $d(v,v') \leq 3d(u,v)$. Also, $d(u',v') \leq d(u',u) + d(u,v) + d(v,v') \leq 6d(u,v)$. This implies that

$$\begin{aligned} d_H(u,v) &\leq d(u,u') + d_H(u',v') + d(v',v) \\ &\leq 5d(u,v) + d_H(u',v') \\ &\leq 5d(u,v) + \alpha(\tfrac{1}{\epsilon})d(u',v') \\ &\leq 5d(u,v) + \alpha(\tfrac{1}{\epsilon})(6d(u,v)) \end{aligned}$$

as claimed. ∎

As an example of how we apply Theorem 2.11, let us recall a well-known result about spanners for general metrics.

Theorem 2.12 (Spanners for general metrics [8, 73]) *For any metric of size n, there exists a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges.*

Applying Theorem 2.11 to Theorem 2.12 yields the following corollary.

Corollary 2.13 (Uniform slack spanners for general metrics) *For any metric, for any $0 < \epsilon < 1$, for any integer $k > 0$, there exists a $(12k-1)$-spanner with ϵ-uniform slack of size $n + O((\frac{1}{\epsilon})^{1+1/k})$.*

Note that if the metric (V,d) was generated by a graph $G = (V,E)$, our previous construction may result in a spanner that is not a subgraph of the original graph G. We now give an alternative construction to obtain a subgraph spanner.

Let us first recall a fact about shortest paths in weighted graphs, whose proof can be found in the journal version of [31]. Suppose $G = (V,E)$ is a weighted graph and P is a set of pairs of vertices.

Fact 2.14 *We can assign a shortest path in the graph G to each pair in P such that the intersection of any two such shortest paths is either empty or also a path in G. If H is the subgraph obtained by the union of all such shortest paths and B is the set of vertices in H with degree at least 3, then $\sum_{v \in B} deg_H(v) \leq O(\sqrt{|B|} \cdot |P|)$.*

Using this fact we can now construct subgraph spanners. As before, let N be an ϵ-density net, which we know has at most $\frac{1}{\epsilon}$ elements. We construct an $\alpha(\frac{1}{\epsilon})$-spanner H' of size $T(\frac{1}{\epsilon})$ on N, which we convert to a subgraph in the following manner. We take P to be the set of distinct pairs $\{u,v\}$ that are edges in H'. We take $\widehat{H}$ to be the union of the shortest paths in G between all pairs in P in the manner as stated in Fact 2.14. Finally, assuming that each node in V has a unique closest point in N (by resolving ties according to some fixed permutation of V), points in V are connected to N by shortest path trees rooted at the points in N, using edges in the given graph G.

The following theorem shows that the resulting subgraph spanner H contains a small number of edges and has small stretch.

Theorem 2.15 *The subgraph H is a $(5 + 6\alpha(\frac{1}{\epsilon}))$-spanner with ϵ-uniform slack and has $O(n + T(\frac{1}{\epsilon})^2)$ edges.*

Proof: Note that the set P contains $T(\frac{1}{\epsilon})$ pairs. Moreover, because the intersection of any two shortest paths between pairs in P is either empty or a path in G, any two such shortest paths can lead to at most 2 vertices in $\widehat{H}$ with degrees at least 3 and so it follows that the set B of vertices in $\widehat{H}$ having degree at least 3 has size bounded by $O(T(\frac{1}{\epsilon})^2)$. Using the Fact 2.14 and the observation that the vertices having degree at most 2 in $\widehat{H}$ must be trivially bounded by n, it follows that $\widehat{H}$ has at most $O(n + T(\frac{1}{\epsilon})^2)$ edges. Connecting points in V to their closest points in N using shortest path trees adds an extra $O(n)$ edges.

To bound the stretch, observe that $\widehat{H}$ is an $\alpha(\frac{1}{\epsilon})$-spanner for points in N. Moreover, each point $x \in V$ has a shortest path to a closest point in N. Hence, it follows immediately as in the proof of Theorem 2.11 that for any two points $u, v \in V$ such that u is ϵ-far away from v, $d_H(u, v) \leq (5 + 6\alpha(\frac{1}{\epsilon}))d(u, v)$. ■

Applying Theorem 2.15 to Theorem 2.12 gives the following corollary.

Corollary 2.16 (Subgraph uniform slack spanners for general metrics) *For any metric, for any $0 < \epsilon < 1$, for any integer $k > 0$, there exists a subgraph $(12k - 1)$-spanner with ϵ-uniform slack of size $O(n + (\frac{1}{\epsilon})^{2+2/k})$.*

2.3.1 Low Weight Spanners

Ideally the slack spanners we create would have low weight as well as low size (where weight is the sum of the distances on the included edges, and size is the number of edges). Clearly distances can be scaled arbitrarily, so by low weight we mean relative to an MST. Chandra et al. [29] give a general transformational method for t-spanners that are not necessarily subgraphs. They assume that t is a constant, though, which will not necessarily be true for our purposes. In particular, we will want to let t be $O(\log n)$. A slight reworking of their algorithm and analysis yields the following result:

Lemma 2.17 *Suppose that there exists an $\alpha(n)$-spanner construction with $O(f(n))$ edges, where $f(n)/2 \geq f(\lfloor n/2 \rfloor)$, that can be constructed in polynomial time. Then for every $\epsilon > 0$ there is a poly-time constructible $(\alpha(n)(1+\epsilon)+\epsilon)$-spanner with $O(f(\frac{n}{\epsilon}))$ edges and weight $O(\frac{1}{n} f(\frac{n}{\epsilon})(1+\epsilon)\alpha(n) \log n) \cdot wt(MST)$.*

This lemma allows us to build a low-weight spanner on the ϵ-density net N_ϵ, but in order to make the entire spanner low-weight we also need to be able to connect the rest of the nodes to the centers via short edges. Fortunately, this is easy to do thanks to a result of Khuller, Raghavachari, and Young [57]. They defined and gave a construction for LASTs (Light Approximate Shortest-path Trees) which we will use as a black box.

Definition 2.18 ((α, β)-LAST) *Let G be an arbitrary graph with non-negative edge weights and a root vertex r. A tree T rooted at r is called an (α, β)-LAST if the following conditions are satisfied:*

1. *The distance of every vertex v from r in T is at most α times the distance between v and r in G.*
2. *The weight of T is at most β times the weight of an MST of G.*

Khuller, Raghavachari, and Young give an algorithm for constructing good LASTs, and prove the following theorem about it.

Theorem 2.19 *The algorithm finds a* $(1+\sqrt{2}\gamma, 1+\frac{\sqrt{2}}{\gamma})$*-LAST for any* $\gamma > 0$.

Using this together with Lemma 2.17, the following theorem is fairly simple.

Theorem 2.20 *Given an* $\alpha(n)$*-spanner with* ϵ*-slack and* $f(n)$ *edges, we can create a* $(O(1)\alpha(\frac{1}{\epsilon}))$*-spanner with* ϵ*-slack,* $O(n + f(\frac{1}{\epsilon}))$ *edges, and* $O(\epsilon f(\frac{1}{\epsilon})\alpha(\frac{1}{\epsilon}) \log \frac{1}{\epsilon}) \cdot wt(MST(N_\epsilon)) + O(1) \cdot wt(MST(V)))$ *weight.*

Proof: Our construction from Lemma 2.9 finds a set of $\frac{1}{\epsilon}$ centers, builds an $\alpha(\frac{1}{\epsilon})$-spanner on those, and then connects every node to its closest center. We use the same set of centers, but instead of the default spanner we put down the low-weight transformation of it, and instead of simply connecting every vertex to the closest center we grow a $(1+\sqrt{2}, 1+\sqrt{2})$-LAST out of the set of centers. Then the total number of edges will be at most $O(n + f(\frac{1}{\epsilon}))$, and the total weight will be $O(\epsilon f(\frac{1}{\epsilon})\alpha(\frac{1}{\epsilon}) \log \frac{1}{\epsilon}) \cdot wt(MST_{centers}) + (1+\sqrt{2}) \cdot wt(MST))$ (where the first term is from the low weight spanner on the centers, and the second is from the LAST). Following the analysis of Theorem 2.11, but with an extra $1+\sqrt{2}$ factor for the distances to the centers and a $2\alpha(\frac{1}{\epsilon}) + 1$ factor instead of an $\alpha(\frac{1}{\epsilon})$ factor for the distance between the centers, we get that the stretch is $11 + 5\sqrt{2} + 12\alpha(\frac{1}{\epsilon}) = O(1)\alpha(\frac{1}{\epsilon})$ as claimed. ■

Applying Theorem 2.20 to Theorem 2.12 with $k = O(\log n)$ gives the following corollary.

Corollary 2.21 (Low weight uniform slack spanner for general metrics) *For any metric of size* n*, there exists an* $O(\log \frac{1}{\epsilon})$*-spanner with* ϵ*-uniform slack of size* $O(n + \frac{1}{\epsilon})$ *and weight* $O(\log^2 \frac{1}{\epsilon})$ *times that of an MST.*

2.3.2 Gracefully Degrading Spanners and Notions of Average Distortion

In this section, we give general procedures to convert ordinary spanners into gracefully degrading spanners. We present two constructions, a simpler one, followed by a more sophisticated one that works under weaker assumptions.

The Simpler Construction

Suppose we know how to construct ordinary $\alpha(n)$-spanners of size $T(n)$ for finite metrics of size n. Observe that typically $\alpha(\cdot)$ is a sublinear function, such as $O(\log n)$. It is often the case that there exists $C, c > 1$ such that $\alpha(n) \leq C\alpha(n^{1/c})$. Then, one can take $\epsilon_0 = n^{-1/c}$ and construct a 1-spanner H_0 for some ϵ_0-density net N_0, having small size. In particular, H_0 can just be a complete graph on N_0, which will have size at most $(\frac{1}{\epsilon_0})^2 = n^{2/c}$. We also make use of the $\alpha(n)$-spanner $\widehat{H}$ for the entire metric V. The gracefully degrading spanner consists of the union of $\widehat{H}$ and H_0, together with edges that connect each point in V to its closest point in N_0. So as long as $c \geq 2$ the size of the spanner is at most $T(n) + n^{2/c} + n \leq T(n) + 2n$.

Observe that if $\epsilon < \epsilon_0$, and y is ϵ-far away from x, then we can use the spanner $\widehat{H}$ to bound the stretched distance, which is at most $C\alpha(\frac{1}{\epsilon})d(x,y)$, because $\alpha(n) \leq C\alpha(n^{1/c}) \leq C\alpha(\frac{1}{\epsilon})$. If $\epsilon \geq \epsilon_0$, then we can use the spanner H_0 as in the slack spanner to conclude that the multiplicative stretch is at most 11. Note that for interesting function α, we have $11 \leq C\alpha(\frac{1}{\epsilon})$. Hence, this simple construction gives us the following theorem on gracefully degrading spanners.

Theorem 2.22 *Suppose there exists an $\alpha(n)$-spanner of size $T(n)$ for any metric of size n, where $\alpha(\cdot)$ is a non-decreasing function such that there exists exists $C > 1$ such that $\alpha(n) \leq C\alpha(n^{1/2})$. Then, for any finite metric (V, d) of size n, there exists an $C\alpha(\frac{1}{\epsilon})$-gracefully degrading spanner of size at most $T(n) + 2n$.*

If we have the stronger assumption that $\alpha(n) \leq C\alpha(n^{1/4})$, then the gracefully degrading spanner can be made to be a subgraph of the weighted graph that induces the metric (V, d).

Applying Theorem 2.22 to Theorem 2.12 with $k = O(\log n)$, we obtain the following corollary.

Corollary 2.23 (Gracefully degrading spanner for general metrics) *Any metric of size n has a $O(\log \frac{1}{\epsilon})$-gracefully degrading spanner H of size $O(n)$. If the metric is induced by some weighted graph G, then H can be made to be a subgraph of G.*

We can now show that this construction actually gives a spanner that has $O(1)$ "average distortion" for both notions of average distortion given in Definitions 2.3 and 2.4.

Theorem 2.24 ("Average Distortion") *For any metric (V, d), there exists a spanner H with size $O(n)$ that has both constant average distortion and constant distortion of the average, and moreover has $O(\log n)$ stretch in the worst case. If the metric (V, d) is induced by some graph G, then H can be made to be a subgraph of G.*

Proof: We use the spanner of Corollary 2.23. Recall that if y is ϵ_0-far away from x, then $d_H(x, y) \leq 11d(x, y)$, otherwise $d_H(x, y) \leq O(\log n)d(x, y)$.

We first bound the average distortion.

$$\begin{aligned}\frac{1}{\binom{n}{2}} \sum_{\{x,y\} \in \binom{V}{2}} \frac{d_H(x,y)}{d(x,y)} &= \frac{1}{n} \sum_{x \in V} \frac{1}{n-1} \sum_{y \neq x} \frac{d_H(x,y)}{d(x,y)} \\ &\leq \frac{1}{n} \sum_{x \in V} \left(\frac{1}{n^{1/4}} O(\log n) + (1 - \frac{1}{n^{1/4}}) \cdot 11 \right) = O(1)\end{aligned}$$

We next bound the distortion of average.

$$\begin{aligned}\frac{\sum_{\{x,y\} \in \binom{V}{2}} d_H(x,y)}{\sum_{\{x,y\} \in \binom{V}{2}} d(x,y)} &= \frac{\sum_{x \in V} \sum_{y \neq x} d_H(x,y)}{\sum_{x \in V} \sum_{y \neq x} d(x,y)} \\ &\leq \max_{x \in V} \frac{\sum_{y \neq x} d_H(x,y)}{\sum_{y \neq x} d(x,y)} \\ &\leq \max_{x \in V} \{ \frac{1}{n^{1/4}} O(\log n) + 11 \} = O(1),\end{aligned}$$

the last inequality following from the following argument. For fixed x, let A be the set of points y that are ϵ_0-far away from x and $\overline{A}$ be the rest of the points. Note that for any $y \in A$ and $\overline{y} \in \overline{A}$, $d(x, \overline{y}) \leq d(x, y)$. Hence, from $|\overline{A}| \leq \epsilon_0 n$, we have

$$\frac{\sum_{y \in \overline{A}} d(x,y)}{\sum_{y \in V} d(x,y)} \leq \epsilon_0 = n^{-1/4}.$$

■

The Sophisticated Construction

Our second construction makes use of density nets of different scales and exhibits gracefully degrading behavior with only weaker assumption on the function α.

Suppose there exists an $\alpha(n)$-spanner of size $T(n)$ for any metric of size n. The assumption that we make is α is bounded by a linear function. This is perfectly reasonable as Kruskal's MST algorithm immediately gives an n-spanner. Moreover, note that $T(n)$ has to be at least $n-1$ in order to connect every pair of points. Hence, we have the following smoothness assumptions.

Assumption 2.25 (Smoothness) *We assume that*

1. $\alpha(2n) \leq 2\alpha(n)$, *and*
2. $T(\frac{n}{2}) \leq \frac{1}{2}T(n)$.

Theorem 2.26 *Suppose there exists a $\alpha(n)$-spanner of size $T(n)$ for every metric of size n. Then, for any finite metric of size n, there exists an $(152\alpha(1/\epsilon)+150)$-gracefully degrading spanner of size $2T(n)+O(n\log^* n)$.*

Construction outline. Let $I := \{0,1,2,\ldots,\lceil \log_2 n \rceil\}$, and for each $i \in I$, let $\epsilon_i = 2^i/n$. For each $i \in I$, we construct an ϵ_i-density net N_i for the metric (V,d) and also a corresponding $\alpha(1/\epsilon_i)$-spanner H_i of size $T(1/\epsilon_i)$ for N_i. Note that $N_0 = V$. The union of the H_i's would be part of our gracefully degrading spanner, which so far contains at most $\sum_{i\in I} T(n/2^i) \leq 2T(n)$, by the smoothness assumption. The crux of the construction is how to add a few number of edges between net points from different scales and maintain small gracefully degrading stretch.

Recall that from the first property of density net, each point x is within distance at most $2R(x,\epsilon_i)$ from N_i. The next lemma shows that if we go from x to N_i via net points from smaller scales, the distance travelled would not increase too much.

Lemma 2.27 *Suppose $0 < i(1) < i(2) < \cdots < i(s)$ and $z_0 \in V$. Suppose for $1 \leq l \leq s$, the point z_l is a closest point in $N_{i(l)}$ to z_{l-1}, and z_l^* is a closest point in $N_{i(l)}$ to z_0. Then,*

$$\sum_{l=1}^{s} d(z_{l-1}, z_l) \leq \sum_{l=1}^{s} 2^{s-l} d(z_0, z_l^*).$$

Proof: We show the result by induction on s. For the base case $s=1$, the result is trivial, because $z_1 = z_1^*$. Assume the result holds for some $s > 0$. Consider the point $z_{s+1} \in N_{i(s+1)}$ closest to z_s. Hence, $d(z_s, z_{s+1}) \leq d(z_s, z_{s+1}^*)$. Now, by the triangle inequality,

$$d(z_s, z_{s+1}) \leq d(z_s, z_{s+1}^*) \leq d(z_0, z_s) + d(z_0, z_{s+1}^*).$$

Using the triangle inequality again,

$$d(z_0, z_s) \leq \textstyle\sum_{l=1}^{s} d(z_{l-1}, z_l).$$

By the induction hypothesis,

$$\textstyle\sum_{l=1}^{s} d(z_{l-1}, z_l) \leq \sum_{l=1}^{s} 2^{s-l} d(z_0, z_l^*).$$

Finally, combining the three inequalities, we have

$$\sum_{l=1}^{s+1} d(z_{l-1}, z_l) \leq 2\sum_{l=1}^{s} 2^{s-l} d(z_0, z_l^*) + d(z_0, z_{s+1}^*) = \sum_{l=1}^{s+1} 2^{(s+1)-l} d(z_0, z_l^*),$$

as required. ∎

Observing that for $1 \leq l \leq s$, $d(z_0, z_l^*) \leq 2R(z_0, \epsilon_{i(l)}) \leq 2R(z_0, \epsilon_{i(s)})$, we have the following corollary.

Corollary 2.28 *Let $0 < i(1) < \cdots < i(s)$ and $z_0, z_1, \ldots, z_s$ be as before. Then,*

$$\sum_{l=1}^{s} d(z_{l-1}, z_l) \leq 2(2^s - 1)R(z_0, \epsilon_{i(s)}).$$

In view of Lemma 2.27 and Corollary 2.28, we can bound the distance between x and y in the gracefully degrading spanner if y is $\epsilon_{i(s)}$-far away from x.

Lemma 2.29 *Suppose $0 = i(0) < i(1) < \cdots < i(s)$ and for $1 \leq l \leq s$, every point in $N_{i(l-1)}$ is connected directly with its closest point in $N_{i(l)}$ in the gracefully degrading spanner H. Suppose y is $\epsilon_{i(s)}$-far away from x. Then, the distance between x and y in H is at most*

$$((5 \cdot 2^s - 4)\alpha(\tfrac{1}{\epsilon_{i(s)}}) + 5 \cdot (2^s - 1))d(x, y).$$

Proof: Note that from the hypothesis, we have $R(x, \epsilon_{i(s)}) \leq d(x, y)$. We apply Lemma 2.27 with $z_0 = x$ and go through the various net points $x_1, x_2, \ldots$ to reach x_s. By Corollary 2.28, the distance travelled from x to x_s is $d_H(x, x_s) \leq 2(2^s - 1)R(x, \epsilon_{i(s)}) \leq 2(2^s - 1)d(x, y)$.

Now we apply Lemma 2.27 again with $z_0 = y = y_0$. Using the notation that for $1 \leq l \leq s$, y_l and y_l^* are closest points in $N_{i(l)}$ to y_{l-1} and y respectively, the distance travelled from y to y_s is $d_H(y, y_s) \leq \sum_{l=1}^{s} 2^{s-l} d(y, y_l^*)$.

However, observing that $d(y, y_l^*) \leq d(y, x_l^*) \leq d(x, y) + d(x, x_l^*)$. We conclude that $d_H(y, y_s) \leq 3 \cdot (2^s - 1)d(x, y)$. Combining the two bounds on $d_H(x, x_s)$ and $d_H(y, y_s)$ and using the triangle inequality, $d(x_s, y_s) \leq d(x_s, x) + d(x, y) + d(y, y_s) \leq (5 \cdot 2^s - 4)d(x, y)$.

Now, the distance between x_s and y_s in the spanner $H_{i(s)}$ is at most

$$\alpha(\tfrac{1}{\epsilon_{i(s)}})d(x_s, y_s) \leq (5 \cdot 2^s - 4)\alpha(\tfrac{1}{\epsilon_{i(s)}})d(x, y).$$

Hence, the distance $d_H(x, y)$ is at most

$$((5 \cdot 2^s - 4)\alpha(\tfrac{1}{\epsilon_{i(s)}}) + 5 \cdot (2^s - 1))d(x, y).$$

■

From the above Lemma 2.29, one can see why our spanner H (which consists of the union of H_i's and extra edges between density nets of different scales) is gracefully degrading. Indeed, given $0 < \epsilon < 1$, we pick the largest i such that $\epsilon_i \leq \epsilon$. Observe that $\epsilon \leq 2\epsilon_i$ and hence $\alpha(1/\epsilon_i) \leq 2\alpha(1/\epsilon)$, by the smoothness assumption. Suppose that for any level i, any point $x \in V$ can reach a close net point in N_i using at most s hops, in the manner described in Lemma 2.29. If y is ϵ-far away from x, then we can conclude that $d_H(x, y) \leq O(2^s)\alpha(1/\epsilon)d(x, y)$.

Our problem reduces to how to add a small number of edges between density net points so that the number of hops to reach a close net point in any level is small. For example, if we add an edge from each point in V to a closest net point in each level, then we can ensure that the hop number to reach any level to be 1, but we could have added $\Omega(n \log n)$ edges. Fortunately, there is a technique introduced by Yao [90] and independently by Alon and Schieber [7] that provides a nice tradeoff between the number of edges added and the hop count, which is also used in the construction of low hop diameter spanner in [28].

The construction makes use of the Ackermann's function, whose definition is recalled below.

Definition 2.30 (Ackermann's function [80]) *Let $A(s,t)$ be a function defined for integers $s,t \geq 0$ as the following.*

$$\begin{aligned} A(0,t) &= 2t && \text{for } t \geq 0 \\ A(s,0) &= 0, A(s,1) = 2 && \text{for } s \geq 1 \\ A(s,t) &= A(s-1, A(s,t-1)) && \text{for } s \geq 1, t \geq 2 \end{aligned}$$

Using the construction as described in [90] and [28], one can get the following result in a straight forward manner.

Lemma 2.31 *Suppose h, s, t are non-negative integers such that $2^h \leq 4A(s,t)$. Then, it is possible to add $2(t+1)n$ edges between density net points of different scales such that for any point $x \in V$ and any $i \leq h$, a net point in N_i can be reached from x via net points in smaller scales, in the manner described in Lemma 2.27, in $s+1$ hops.*

Proof of Theorem 2.26: We use Lemma 2.31. Note that the number of levels is $h \leq \log_2 n$. Hence, by putting $s = 3$ and $t = O(\log^* n)$, we conclude that the number of hops is at most 4. Hence, using Lemma 2.29, we can obtain a $(152\alpha(1/\epsilon) + 150)$-gracefully degrading spanner, as required. ■

2.4 Distance Oracles and Labelings

The techniques that we developed for slack spanners also turn out to be useful for developing slack distance oracles and distance labelings. Distance oracles and labelings have been widely studied, perhaps even more so than spanners, and distance labelings were in fact one of the original motivations for the definition of slack embeddings by Kleinberg, Slivkins, and Wexler [58]. We give the first slack labelings that do not use an embedding into ℓ_p, allowing us to bypass a lower bound from Abraham et al. [2].

2.4.1 Distance oracles

Thorup and Zwick [82] studied the problem of creating distance oracles for metric spaces. A distance oracle is a small data structure which allows fast queries for approximate distances. Distance oracles are supposed to capture the heart of the all-pairs shortest paths problem; in many applications we are not actually interested in every single one of the distances, but instead just need to quickly find any distance if it is needed. They give an oracle that, for any integer $k \geq 1$, takes $O(kn^{1+1/k})$ space, has $O(k)$ query time, and has stretch of $2k-1$. It is natural to introduce slack to distance oracles, especially since the main method of constructing distance oracles is via sparse spanners. This was first done by Abraham, Bartal, and Neiman [3], who give a result for gracefully degrading distance oracles but not for slack distance oracles. We describe constructions for both, giving the first slack distance oracles and a simpler gracefully degrading oracle that achieves constant average distortion, matching the result of [3]. We assume a word of memory can store a distance or a node identifier (which would be true in most practical implementations, as well as in the theoretical cell-probe model). We first give a general transformational theorem.

Theorem 2.32 *Suppose that there exists some distance oracle with $\alpha(n)$ stretch and $O(q(n))$ query time that uses $O(f(n))$ space. Then there exists a distance oracle with ϵ-uniform slack, $5 + 6\alpha(\frac{1}{\epsilon})$ stretch, and $O(q(\frac{1}{\epsilon}))$ query time that uses $O(n + f(\frac{1}{\epsilon}))$ space.*

Proof: As usual we first create the set of centers by using the ϵ-density net of Lemma 2.9. For each vertex that is not a center, store the edge to the closest center. Then use the given distance oracle on the set of centers. The claimed slack and stretch bounds are directly from the analysis of Theorem 2.11. On a query of vertices u, v, we simply look up the center u' closest to u and v' closest to v (which can be done in constant time since we only store that one edge) and then use the given oracle on the centers, which takes $O(q(\frac{1}{\epsilon}))$ time. Similarly, we need $O(n)$ space to store the one edge incident on each non-center and $O(f(\frac{1}{\epsilon}))$ space to store the oracle for the centers, giving a total of $O(n + f(\frac{1}{\epsilon}))$ space. ■

By using this transformation on the distance oracle of Thorup and Zwick [82, Theorem 3.1], we get the following corollary.

Corollary 2.33 (Uniform slack distance oracle) *For every integer $k \geq 1$, there is a distance oracle with ϵ-uniform slack, $O(k)$ query time, and $12k - 1$ stretch that uses $O(n + k(\frac{1}{\epsilon})^{1+1/k})$ space.*

Proof: For any integer $k \geq 1$, Thorup and Zwick give a distance oracle with $O(k)$ query time and $2k - 1$ stretch that uses $kn^{1+1/k}$ space. The corollary then follows immediately from Theorem 2.32. ■

Gracefully degrading distance oracles

We can also use the ideas for gracefully degrading spanners from Section 2.3.2 to create gracefully degrading distance oracles. Recall that Theorem 2.22 used a $\alpha(n)$-spanner of the entire metric together with an 1-spanner for an ϵ_0-density net. Instead of using two levels of spanners, we just use two levels of distance oracles, where the oracle on the ϵ_0-density net is exact. So if $\epsilon < \epsilon_0$ then we use the distance oracle on the entire metric which for any points x, y in the space gives a stretch of $\alpha(n) \leq C\alpha(n^{1/c}) \leq C\alpha(\frac{1}{\epsilon})$. If $\epsilon \geq \epsilon_0$ then we use the exact distance oracle on the density net to get a multiplicative stretch of 11. By letting $c = 2$ we know that the 1-oracle will not take more than n space, and storing the closest center for each node only takes another n space, so the total space used is only $f(n) + 2n$. The query time in the 1-oracle is constant, so the total query time is $O(q(n))$. This gives the following theorem on transforming distance oracles into gracefully degrading distance oracles.

Theorem 2.34 *Suppose that there exists some distance oracle with $\alpha(n)$ stretch and $O(q(n))$ query time that uses $O(f(n))$ space, where $\alpha(\cdot)$ is a nondecreasing function for which there exists a constant $C > 0$ such that $\alpha(n) \leq C\alpha(n^{1/2})$. Then there exists a gracefully degrading distance oracle with $C\alpha(\frac{1}{\epsilon})$ stretch and $O(q(n))$ query time that uses $O(f(n)) + 2n$ space. Furthermore, the average distortion and distortion of average of this distance oracle is $O(1)$.*

The stretch of the distance oracle of Thorup and Zwick [82, Theorem 3.1] satisfies the requirement on $\alpha(\cdot)$, so by applying Theorem 2.34 to that oracle and using the average case analysis of Theorem 2.24 we get the following corollary.

Corollary 2.35 (Gracefully degrading distance oracle) *For any integer k with $1 \leq k \leq O(\log n)$, there is a distance oracle with worst case stretch of $2k-1$ and $O(k)$ query time that uses $O(kn^{1+1/k})$ space such that the average distortion and the distortion of average is $O(1)$.*

The gracefully degrading distance oracle of Abraham, Bartal, and Neiman [3, Theorem 14] gives the same query time, worst case stretch, average distortion, and distortion of average. Their oracle uses $O(n^{1+1/k} \log n)$ space, though, which is more than we use if $k = o(\log n)$ and the same if $k = \Theta(\log n)$.

2.4.2 Distance labels

A distance labeling is an assignment of labels to the vertices so that the approximate distance between any two vertices can be computed simply by looking at the two labels. The goals are to minimize the stretch, the size of the label, and the time needed to compute the distance given the two labels. It is natural to extend this definition to slack labelings in the obvious way. We give the first slack distance labeling that uses space independent of n. Note that any embedding of a metric into ℓ_p gives a distance labeling where the size of a label is the dimension of the embedding. Embeddings of this form were considered by Abraham et al. [2], who proved that the dimension must depend on $\log n$. Thus any distance labeling that uses a slack embedding into ℓ_p must use space that depends on $\log n$, whereas our labeling is independent of n.

As with distance oracles, we begin by giving a general transformation theorem. Note that as before all space claims assume that it takes only a constant amount of space to store a distance or an identifier of a point, or equivalently the space bounds can be viewed as bounds on the number of words rather than the number of bits.

Theorem 2.36 *Let (V, d) be a metric space with n points. Suppose that there exists a distance labeling where each label has size $O(f(n))$ and for any two points u, v it is possible to compute, in $O(q(n))$ time, an approximation to the distance between u and v with a stretch of at most $\alpha(n)$. Then there exists a distance labeling with ϵ-uniform slack such that every label has size $O(f(\frac{1}{\epsilon}))$, and computing distances up to a stretch of $5 + 6\alpha(\frac{1}{\epsilon})$ can be done in $O(q(\frac{1}{\epsilon}))$ time.*

Proof: Create a set of centers using Lemma 2.9. Apply the given labeling to the set of centers, and for each non-center let its label be the distance to the center closest to it together with the label for that center. The claimed slack and stretch bounds are clear, and the size bound is immediate since each label is just a distance and a center label. The computation time is just the time to add two distances (which we assume takes constant time) plus the time to find the distance between the two centers. ■

We get the following corollary by simply applying Theorem 2.36 to the distance labeling of Thorup and Zwick [82, Theorem 3.4]. Note that the size of the labels is independent of n.

Corollary 2.37 (Uniform lack distance labeling) *Let (V, d) be a metric space on n points. Let $0 < \epsilon < 1$, and let k be an integer with $1 \leq k \leq \log \frac{1}{\epsilon}$. Then it is possible to assign each point a label that uses $O((\frac{1}{\epsilon})^{1/k} \log^{1-1/k} \frac{1}{\epsilon})$ space such that, given the labels of vertices u, v where v is ϵ-far from u, the distance $d(u, v)$ can be computed up to a stretch of $12k-1$ in $O(k)$ time.*

Gracefully degrading distance labelings

As with distance oracles, we can use the ideas from Section 2.3.2 to create gracefully degrading labelings.

Theorem 2.38 *Suppose that there exists some distance labeling scheme for any metric space in which each label has size $O(f(n))$ and for any two points $u, v \in V$ it is possible to compute, in $O(q(n))$ time, an approximation to $d(u, v)$ with $\alpha(n)$ stretch. Furthermore, suppose that $\alpha(\cdot)$ is a nondecreasing function such that for every integer $c > 0$ there exists a constant $C > 0$ such that $\alpha(n) \leq C\alpha(n^{1/c})$. Then for every metric space (V, d) on n points and every integer $c > 0$ there exists a gracefully degrading distance labeling of V where each label has size $O(f(n)+n^{1/c})$, the stretch is $C\alpha(\frac{1}{\epsilon})$, and the time to compute an approximate distance given two labels is $O(q(n))$.*

Proof: Let $\epsilon_0 = n^{-1/c}$, and create an ϵ_0-density net of V. For each point v in the density net, let $label'(v)$ be the distances to every other point in the net, which takes at most $O(n^{1/c})$ space. For every point $v \in V$, let $label''(v)$ be the label assigned to v by simply using the given scheme on the entire space, giving labels of size $O(f(n))$. Then for every point $v \in V$ we let $label(v)$ be $label''(v)$ together with the distance from v to the closest point u in the ϵ_0-density net and $label'(u)$. Then clearly the required space for a single label is $O(f(n) + n^{1/c})$, as claimed. If $\epsilon < \epsilon_0$ then we can compute the distance between two nodes u and v by simply comparing $label''(u)$ and $label''(v)$ in $O(q(n))$ time, giving a stretch of at most $\alpha(n) \leq C\alpha(n^{1/c}) \leq C\alpha(\frac{1}{\epsilon})$. If $\epsilon \geq \epsilon_0$ then we get a multiplicative stretch of 11 by adding the distance from v to its closest center point to the distance from u to its closest center point and then adding the distance between the two center points (which can be obtained from either label). This only takes constant time, and thus the computation time is $O(q(n))$. ■

Applying Theorem 2.38 with $c = k$ to the distance labeling of Thorup and Zwick [82, Theorem 3.4] and then using the average case analysis of Theorem 2.24 gives the following corollary, which gives a distance labeling with bounds that essentially match those of Abraham, Bartal, and Neiman [3, Theorem 10] when $k = O(\log n)$.

Corollary 2.39 (Gracefully degrading distance labeling) *For any integer k with $1 \leq k \leq O(\log n)$, there is a distance labeling of any n point metric such that each label has size at most $O(n^{1/k} \log^{1-1/k} n)$, and given two labels it is possible to compute the distance between the two points up to a worst case stretch of $2k-1$ in $O(k)$ time. Furthermore, the average distortion and the distortion of average are $O(1)$.*

2.5 Compact Routing

As in spanners, oracles, and labelings, the main tradeoff in compact routing is between the *stretch* (the distance traveled by a packet divided by the shortest path distance) and the *space*. But in the compact routing setting the notion of space is a bit more complicated. There is the *table size*, which is just the amount of space used at each node to store routing table information or anything else used by the routing protocol, but there is also the *header size*, which is the amount of space used in each packet to store information used by the routing scheme. Furthermore, there are two main models of compact routing: the *name-dependent* model (in which schemes are allowed to assign labels to vertices and can route using information in the labels) and the *name-independent*

model (in which schemes have to route without changing the name of any node). In the name-dependent model an additional space parameter is the *label size*, which is the size of an assigned name. These models turn out to be quite different when our goal is slack constructions. We begin with constructing schemes for the name-dependent model, and then construct schemes and prove impossibility results for the name-independent model.

2.5.1 Name Dependent Model

Our name-dependent schemes are adaptations of the spanners in Section 2.3. In particular, they follow the general formula of building a density net and remembering how to route to, on, and from it. Given ϵ, let N be an ϵ-density net. For each $u \in V$, let $Net(u)$ be the closest point in N to u. For each $v \in N$, connect all points $u \in V$ such that $Net(u) = v$ to v by a shortest path tree T_v rooted at v. We begin by giving a general conversion theorem from tree routing to general slack routing. Note that the real power of this method is reducing general routing to tree routing, which works well when tree routing is much easier than general routing but does not obviously help us if tree routing is hard; we will discuss this more in Section 2.5.2.

Theorem 2.40 *Suppose that any tree with n vertices admits a routing scheme R' with routing tables of size $s(n)$, labels of size $\ell(n)$, headers of size $h(n)$, and stretch $\alpha(n)$. Then given any weighted graph $G = (V, E)$ and any $0 < \epsilon < 1$, there is an ϵ-slack routing scheme R in the name-dependent model with routing tables of size $\frac{1}{\epsilon}\log n + s(n)$, labels of size $\log\frac{1}{\epsilon} + \ell(n)$, headers of size $\log\frac{1}{\epsilon} + h(n)$, and stretch $3\alpha(n) + 4$*

Proof: Arbitrarily assign each node in the net a unique ID in $\{1, \ldots, |N|\}$. The ID of a vertex v will be referred to as $ID(v)$. The routing table of each node contains $|N| \leq \frac{1}{\epsilon}$ entries, one for each net node, and thus every node can route directly to any net node. The label of a node v is the ID of $Net(v)$ together with the label assigned to v by R' applied to $T_{Net(v)}$. Routing from u to v is done by routing directly to $Net(v)$ (which can be done using the routing tables at each node and the ID of $Net(v)$, which is contained in the label of v) and then using R' to get from $Net(v)$ to v. The table size is then at most $\frac{1}{\epsilon}\log n + s(n)$, since it takes at most $\log n$ bits to represent a port. Since each ID of a net node is some integer in $\{1, \ldots, \frac{1}{\epsilon}\}$, the total label size is at most $\log\frac{1}{\epsilon} + \ell(n)$ and the total header size is at most $\log\frac{1}{\epsilon} + h(n)$.

We use Lemma 2.10 and the triangle inequality to bound the stretch from u to v when v is ϵ-far from u, getting that $d_R(u, v) = d(u, Net(v)) + d_{R'}(Net(v), v) \leq d(u, v) + d(v, Net(v)) + \alpha(n)d(Net(v), v) \leq 4d(u, v) + 3\alpha(n)d(u, v)$. ∎

Note that since the ports only have to be labeled according to the one tree routing scheme R', this conversion gives a scheme in the designer-port model if R' is in the designer-port model and a scheme in the fixed-port model otherwise.

To get an actual routing scheme, we apply this conversion theorem to the following tree routing schemes of Thorup and Zwick:

Theorem 2.41 (Thorup and Zwick [83]) *Given a tree T, it is possible to route exactly on T (stretch 1) in the fixed port model using no routing tables and labels and headers of size $o(\log^2 n)$. In the designer port model, it is possible to route on T with stretch 1 using no routing tables and labels and headers of size $(1 + o(1))\log n$.*

Using Theorems 2.40 and 2.41, we can give our first name-dependent slack routing schemes:

Theorem 2.42 *Let $G = (V, E)$ be a weighted graph. Then for any $0 < \epsilon < 1$, there is a routing scheme in the fixed-port model with ϵ-slack, routing tables of size $O(\frac{1}{\epsilon} \log n)$, headers and labels of size $o(\log^2 n)$, and stretch 7. In the designer port model, there is a scheme with the same parameters except with headers and labels of size $(1 + o(1)) \log n + \log \frac{1}{\epsilon}$.*

One downside of this scheme is the dependence on $\frac{1}{\epsilon}$ in the table size. We would like to decrease this, since if ϵ is extremely small then this becomes large. Our next scheme reduces this dependence, but at the cost of slightly increased stretch. We again give a conversion theorem, this time from tree and general routing schemes to slack routing schemes:

Theorem 2.43 *Suppose that there is a general routing scheme $\bar{R}$ with routing tables of size $s(n)$, labels of size $\ell(n)$, headers of size $h(n)$, and stretch $\alpha(n)$. Furthermore, suppose that there is a tree routing scheme R' with routing tables of size $s'(n)$, labels of size $\ell'(n)$, headers of size $h'(n)$, and stretch $\alpha'(n)$. Then given a weighted graph $G = (V, E)$ and a parameter $0 < \epsilon < 1$, there is a ϵ-slack routing scheme R for G with routing tables of size $s(\frac{1}{\epsilon^4} + \frac{1}{\epsilon}) + s'(n)$, labels of size $\ell(\frac{1}{\epsilon^4} + \frac{1}{\epsilon}) + \ell'(n)$, headers of size $h(\frac{1}{\epsilon^4} + \frac{1}{\epsilon}) + h'(n)$, and stretch $2 + 3\alpha'(n) + 6\alpha(\frac{1}{\epsilon^4} + \frac{1}{\epsilon})$*

Proof: The basic idea of this scheme is to route inside of the net, so a packet going from u to v will go from u to $Net(u)$ to $Net(v)$ to v instead of going from u to $Net(v)$ to v as in Theorem 2.40. Unfortunately we cannot just apply $\bar{R}$ to the net since the net nodes themselves do not necessarily induce a connected subgraph. Indeed, since the net nodes are all fairly far away from each other they are likely connected only through many intermediate nodes.

Fortunately we can use techniques similar to those used by Coppersmith and Elkin to construct pair-wise distance preservers [31]. In particular, we can use Fact 2.14 on the graph G with $P = \binom{N}{2}$. The resulting subgraph G' is a shortest path graph of N. Considering just the edges used in two paths, there are at most two vertices of degree at least 3 (since they either intersect at a single path or not at all). Since there are at most $\frac{1}{\epsilon^2}$ paths, this implies that there are at most $\frac{1}{\epsilon^4}$ vertices in the shortest path graph with degree at least 3, and thus at most $\frac{1}{\epsilon^4} + \frac{1}{\epsilon}$ vertices with degree either 1 or at least 3. Each node of degree 2 is on a unique path between these nodes, so we get a new graph $\widehat{G}$ by removing all of the nodes of degree 2 in G' and replacing each of the removed paths by an edge of the same total length. $\widehat{G}$ is connected and has the property that the shortest path between every pair of net nodes is the same as the shortest path in the original graph, so we apply $\bar{R}$ to this graph.

Routing from u to v takes place as follows. First u routes up to $Net(u)$, which takes only constant space if we have every node remember the port used to send up to its net node. In the next phase we route to $Net(v)$ using $\bar{R}$ on $\widehat{G}$. Since this graph replaced paths with edges, all of the degree 2 nodes that were removed need to know what to do. But this is easy since they only have degree 2 in G', so they can just remember the two ports used for net routing and automatically forward any packet received on one port in this phase to the other port. All of the other nodes that might be encountered in this phase were not removed, so they know how to route. When the packet reaches $Net(v)$, we route down $T_{Net(v)}$ to v by using R'.

So in this scheme the routing table at a node might have to include the routing table of $\bar{R}$ on a graph with $\frac{1}{\epsilon^4} + \frac{1}{\epsilon}$ nodes and the table of R' on a graph with n nodes, for a total size of $s(\frac{1}{\epsilon^4} + \frac{1}{\epsilon}) + s'(n)$. A label for v consists of a tree routing label from R' together with the label of $Net(v)$ in $\bar{R}$ applied to $\widehat{G}$, for a total size of $\ell(\frac{1}{\epsilon^4} + \frac{1}{\epsilon}) + \ell'(n)$. A header has to include a tree routing header and a general header for $\widehat{G}$, for a total size of $h(\frac{1}{\epsilon^4} + \frac{1}{\epsilon}) + h'(n)$.

It remains to prove the stretch bound. Let $u, v \in V$ such that v is ϵ-far from u. Using Lemma 2.10 and the triangle inequality, we have that

$$\begin{aligned} d_R(u,v) &\leq d(u, Net(u)) + \alpha(\frac{1}{\epsilon^4} + \frac{1}{\epsilon})d(Net(u), Net(v)) + \alpha'(n)d(Net(v), v) \\ &\leq 2d(u,v) + 3\alpha'(n)d(u,v) + \alpha(\frac{1}{\epsilon^4} + \frac{1}{\epsilon})(d(Net(u), u) + d(u,v) + d(Net(v), v)) \\ &\leq 2d(u,v) + 3\alpha'(n)d(u,v) + \alpha(\frac{1}{\epsilon^4} + \frac{1}{\epsilon})6d(u,v)) \end{aligned}$$

as claimed. ∎

We now use another result of Thorup and Zwick, this time on general compact routing. Note that since the fixed-port model is a restricted version of the designer-port model, the following results (which hold for fixed ports) also hold for designer ports.

Theorem 2.44 (Thorup and Zwick [83]) *Let $k > 1$ be an integer, and let $G = (V, E)$ be a weighted graph. Then there is a compact routing scheme in the fixed-port model with routing tables of size $o(n^{1/k} \log^{3-1/k} n)$, labels of size $o(k \log^2 n)$, and headers of size $o(\log^2 n)$, and stretch 4k-3. In addition, after one round of handshaking in which $o(\log^2 n)$ bits are exchanged they route with stretch $2k - 1$.*

Now we can apply Theorem 2.43 to the Thorup and Zwick schemes to get out main result on name-dependent compact routing with slack:

Theorem 2.45 *Let $k > 1$ be an integer, and let $G = (V, E)$ be a weighted graph. Then there is a compact routing scheme in the fixed-port model with ϵ-slack, routing tables of size $o(\frac{1}{\epsilon^{4/k}} \log^{3-1/k} \frac{1}{\epsilon})$, labels of size $o(\log^2 n) + o(k \log^2 \frac{1}{\epsilon})$, headers of size $o(\log^2 n)$, and stretch $24k - 13$. After a round of handshaking in which $o(\log^2 \frac{1}{\epsilon})$ bits are exchanged, this stretch can be reduced to $12k - 1$.*

One important corollary of this is the scheme obtained by setting $k = \log \frac{1}{\epsilon}$ in the above scheme. This results in a name-dependent compact routing scheme with $O(\log \frac{1}{\epsilon})$ stretch and all of the space bounds at most polylogarithmic in n and $\frac{1}{\epsilon}$. We will use this scheme in the next section.

Gracefully Degrading Routing

As with spanners and embeddings, we would like to create a slack routing scheme that works not just for a fixed ϵ but for all ϵ simultaneously. Such a scheme would have polylogarithmic (in n) routing tables, headers, and labels, and for any ϵ would guarantee $O(\log \frac{1}{\epsilon})$ stretch on all but an ϵ fraction of the pairs. We almost give such a scheme, using our slack routing scheme from Theorem 2.45 as a basic building block but also losing a $\log \Delta$ factor in the label size (where Δ is the diameter of the graph). However, this loss of $\log \Delta$ is due exclusively to having to store some distances, so if it takes constant space to store a distance (as would be the case in any actual implementation or in the cell-probe model), then we achieve polylog(n) size labels.

For integer i with $0 \leq i \leq \log n$, let $\epsilon_i = \frac{2^i}{n}$. We will refer to each i as a *level*. For each ϵ_i, create a ϵ_i-slack routing scheme according to Theorem 2.45 with $k = \log \frac{1}{\epsilon_i}$. Let $0 < \epsilon < 1$, and let i be the largest i such that $\epsilon_i \leq \epsilon$ (such an i always exists since without loss of generality

$\epsilon \geq \frac{1}{n}$). Then the routing scheme for ϵ_i suffices for ϵ, since it ignores $\epsilon_i < \epsilon$ fraction of the pairs, has stretch $O(\log \frac{1}{\epsilon_i}) = O(\log \frac{2}{\epsilon}) = O(\log \frac{1}{\epsilon})$, and has tables, headers, and labels of size still polylogarithmic in n and $\frac{1}{\epsilon}$. Unfortunately the same routing scheme needs to work for all ϵ, so we can't simply choose an i and use that level. However, note that the route between u and v in the desired level is no shorter than the shortest route between u and v over all $\log n$ levels. So if we could remember for each pair of nodes which level gives the shortest path, then we could just use that level whenever we're asked to route between that pair.

Obviously we can't actually remember the best level for each pair, since that would take $\Omega(n)$ space at each node. But we can do something just as good: allow the source to do some computation and figure out which level would be best given the destination. And thanks to another result of Thorup and Zwick [82] we can do this using only polylogathmic space at each node. In particular, Thorup and Zwick proved the following result about distance labels:

Theorem 2.46 (Thorup and Zwick [82]) *Let (V, δ) be a metric space on n points with integral distances with diameter Δ. Let $1 \leq k \leq \log n$ be an integer. Then it is possible to assign to each point $v \in V$ an $O(n^{1/k} \log^{1-1/k} n \log(n\Delta))$-bit label, denoted label(v), such that for any two points $u, v \in V$ it is possible to compute, in $O(k)$ time, an approximation to the distances $\delta(u, v)$ with a stretch of at most $2k - 1$.*

The restriction to integral distances is required only so that a distance can be stored in $\log \Delta$ space. We remove this assumption, and just let $\log \Delta$ be the space necessary to store a distance.

Now we can define our gracefully degrading routing scheme. For each level i with $0 \leq i \leq \log n$, let R_i be the routing scheme from Theorem 2.45 with parameters ϵ_i and $k = \log \frac{1}{\epsilon_i}$. For each $v \in V$, let $Net_i(v)$ be the closest net point to v in the density net used by R_i. The routing table of v consists of the union over all levels i of the routing table of R_i at v. The label of v is the union over levels i of the label assigned to v by R_i together with the distance to $Net_i(v)$ and the distance label given by Theorem 2.46 to $Net_i(v)$ in the shortest path graph for the net at level i.

When routing from u to v we compute the distance from u to v in each level i by computing $d(u, Net_i(u)) + d(Net_i(v), v) + d'(Net_i(u), Net_i(v))$, where d' is the distance given by comparing the distance labels for the two net points. We then route along the level that minimizes this quantity. Note that all of the information needed to compute the correct level is in the labels of u and v, and that after the correct level is determined routing can be carried out as in Theorem 2.45 by using an extra $O(\log \log n)$ bits in the header to tell intermediate nodes what the correct level is.

Theorem 2.47 *The above scheme is a gracefully degrading compact routing scheme with routing tables of size $o(\log^4 n)$, labels of size $O(\log^4 n + \log^2 n \log \Delta)$, headers of size $o(\log^2 n)$, and stretch $O(\log \frac{1}{\epsilon})$.*

Proof: Since each level uses $k = \log \frac{1}{\epsilon_i}$, the total size of the tables is at most $\sum_{i=0}^{\log n} o(\log^3 \frac{n}{2^i}) \leq o(\log^4 n)$. Each label contains for each level a normal slack label and a distance label for its net node, so the size of a label is at most $\sum_{i=0}^{\log n} (o(\log^2 n) + o(\log^3 \frac{1}{\epsilon}) + O(\log^2 n + \log n \log \Delta)) \leq O(\log^4 n + \log^2 n \log \Delta)$ since $\frac{1}{\epsilon} \leq n$. Each header is just the header for a given level together with the number of that level, so has size at most $o(\log^2 n) + \log \log n = o(\log^2 n)$.

For the stretch bound, it is important to note that we chose to use results of Thorup and Zwick for both the net routing and the distance labels. This was not coincidence. The Thorup and Zwick

routing scheme we use (Theorem 2.44) is based on the techniques that they pioneered for distance oracles and labels in [82], and in particular the $2k-1$ stretch (after handshaking) in the routing scheme is the exact same $2k-1$ as the stretch of their distance labels. Furthermore, using the same labels we can compute the same $4k-3$ as in the routing scheme without handshaking. It is not just that the bounds are the same, but that they actually compute the exact same path, so in fact the distance label gives precisely the distance that the routing scheme will take inside the net. Thus our calculation of the distance travelled between u and v at each level exactly equals the actual distance it would take if we routed at that level using our scheme, and so by choosing the smallest such distance we choose the best level.

Since we are routing along the best level, we are in particular routing along a path that is no longer than the path we would have taken by routing along the level just below ϵ, which we showed earlier would be a sufficient level to route on. So the stretch of the path that we take is no more than the stretch of that level, which by construction is $O(\log \frac{1}{\epsilon})$. ■

One thing to note about the above theorem is that while it is not scale-free due to the dependence on $\log \Delta$ in the labels, it is close to being scale-free. In particular, the $\log \Delta$ comes from the need to store distances [82, Thm 3.4], not from any inherent dependence on the aspect ratio due to a distance-based decomposition like the sparse covers of [12, 13]. So if distances can be stored in a constant amount of space (which is a reasonable approximation, since presumably any implementation would use a fixed size floating point method of storing distances) then the dependence on the diameter disappears.

An important corollary to the above theorem is the existence of a routing scheme (without any kind of slack) with polylogarithmic size labels, tables, and headers, and stretch $O(\log n)$ in the worst case but $O(1)$ on average. In fact, the above scheme satisfies these requirements:

Corollary 2.48 *The above gracefully degrading routing scheme has $O(\log n)$ worst case stretch, but $O(1)$ average distortion and $O(1)$ distortion of average.*

Proof: It clearly has $O(\log n)$ worst case stretch since if $\epsilon = \frac{1}{n}$ the fact that it has ϵ-(uniform) slack guarantees that it does not ignore any pair (since each vertex v is only allowed to ignore the closest node to v, which would be itself), and on everything else it has stretch $O(\log n)$. To bound the average distortion, again let $\epsilon_i = \frac{2^i}{n}$, and let $u \in V$ be an arbitrary vertex. For each ϵ_i the routing scheme ignores the closest 2^i nodes, and has stretch $O(\log \frac{n}{2^i})$ on the rest. Let $h(v)$ denote the largest i such that u does not ignore v, i.e. $h(v) = \max\{i : v \notin B_{\epsilon_i}(u)\}$. For each $0 \le i \le \log n$, let $F(i) = \{v \in V : h(v) = i\}$. Note that $|F(i)| \le 2^i$, and that there is some constant c such that for each $v \in F(i)$ the stretch of the route from u to v is at most $c \log \frac{n}{2^i}$. Then the average stretch of routes with source u is at most

$$\frac{1}{n-1}\sum_{v \in V} \frac{d'(u,v)}{d(u,v)} \le \frac{1}{n-1}\sum_{i=0}^{\log n} |F(i)| c \log \frac{n}{2^i} \le \frac{c}{n-1}\sum_{i=0}^{\log n} 2^i \log \frac{n}{2^i}$$
$$\le \frac{c}{n-1}(2n - \log n - 2) = O(1)$$

Since this is true for all $u \in V$, the total average distortion is $O(1)$.

To bound the distortion of average, first note that

$$\frac{\sum_{x,y \in V} d_R(x,y)}{\sum_{x,y \in V} d(x,y)} = \frac{\sum_x \sum_y d_R(x,y)}{\sum_x \sum_y d(x,y)} \le \max_{x \in V} \frac{\sum_y d_R(x,y)}{\sum_y d(x,y)}$$

Let x be the vertex that maximizes this ratio. Let $Z = \sum_y d(x,y)$, and let $Z_i = \sum_{y \in F(i)} d(x,y)$. Then for some constant c, the distortion of average is $(1/Z)\sum_{i=0}^{\log n} Z_i c \log \frac{n}{2^i}$. But $Z_i/Z \leq \frac{2^{i+1}}{n}$ since every vertex in $F(i)$ is one of the closest 2^{i+1} nodes to x. Hence the total distortion of average is at most $c\sum_{i=0}^{\log n} \frac{2^{i+1}}{n} \log \frac{n}{2^i} = 2c(2n - \log n - 2)/n = O(1)$. ■

2.5.2 Name-Independent Model

The name-independent model is significantly more difficult to work in than the name dependent model. As an example, note that the above schemes depend on being to extract $Net(v)$ from the label of v. This is impossible in the name-independent model, since the names are arbitrarily (or adversarially) assigned and thus contain no information about their location in the network. In this section we give a good compact routing scheme with slack for the designer port model and give a simple modification of a lower bound of Abraham, Gavoille, and Malkhi [5] to prove that no such scheme exists in the fixed port model. The difference between the models is that in the designer port model the scheme is allowed to number the ports of a vertex in any way it wants (so long as it only uses the values from 0 up to the degree), while in the fixed port model it is not allowed to renumber the ports.

Designer ports

Our name-independent designer port scheme is based off of the basic name-dependent scheme of Theorem 2.42. The main problem is that without the power to assign labels we do not know $Net(v)$, so we do not know which net node to route to. We overcome this by using hashing and a distributed data structure that stores all of the necessary label information somewhere close to the source node.

There is also the added difficulty of tree routing. In the name-dependent model we could route in trees very efficiently, but in the name-independent model the best known single source routing algorithm, due to Laing [65], has stretch $2k-1$ and uses $\tilde{O}(n^{1/k})$ space at each node, and the best all-pairs scheme [6] has $O(k)$ distortion and uses $\tilde{O}(n^{1/k})$ space. In the fixed port model Laing's single-source scheme is optimal [5], and although this lower bound does not work in the designer port model there is no known algorithm that uses the ability to assign ports to beat the scheme of [65]. So reducing the problem to tree routing, as we did in Theorem 2.42, is not particularly helpful.

There is one situation where name-independent tree routing can be done, though: Abraham, Gavoille, and Malkhi showed in [1] that tree routing can be done with polylog space and constant stretch if the tree is unweighted. This result holds for both the fixed port and designer port models.

Let $u \in V$, and let $v \in V$ be ϵ-far from u. Let $B'_u = B(Net(u), R(Net(u), \epsilon))$ be the ϵn closest points to $Net(u)$, and let T'_u be the shortest path tree rooted at $Net(u)$ on B'_u. The following lemma gives the essential property of these trees that we will be using.

Lemma 2.49 *If v is ϵ-far from u and $w \in B'_u$, then $d_{T'_u}(Net(u), w) \leq 4d(u,v)$.*

Proof: Since $w \in B'_u$, by definition $d(Net(u), w) \leq R(Net(u), \epsilon)$. If $Net(u)$ is the first net point to cover u in the greedy density net construction algorithm then we know that $R(Net(u), \epsilon) \leq$

$R(u,\epsilon)$, and thus $d_{T'_u}(Net(u),w) \leq R(Net(u),\epsilon) \leq R(u,\epsilon) \leq d(u,v)$ by the definition of ϵ-far. Otherwise let $u' \in N$ be the first point in the greedy algorithm to cover u. Then not only is $d(u,Net(u)) \leq 2R(u,\epsilon)$, but also $d(u,u') \leq 2R(u,\epsilon)$. Since u' is in N, by property (3) of ϵ-density nets we know that $u' \notin B_\epsilon(Net(u))$, and thus

$$\begin{aligned} d(Net(u),w) &\leq R(Net(u),\epsilon) \leq d(Net(u),u') \\ &\leq d(Net(u),u) + d(u',u) \leq 2d(u,v) + 2d(u,v) = 4d(u,v) \end{aligned}$$

as claimed. ■

Since we are aiming for constant stretch, this lemma implies that we can play around for a while inside B'_u. Let $h : V \to [\epsilon n]$ be any balanced hash function, e.g. computing $ID(v) \mod \epsilon n$. Note that since h is balanced only $O(\frac{1}{\epsilon})$ nodes map onto a single value. For each center $u \in N$ assign each of the ϵn elements of $B'_u = B_\epsilon(u)$ a different color in $[\epsilon n]$, and let the color assigned to a node v be denoted by $color(v)$. Each node receives at most one color since $B_\epsilon(x) \cap B_\epsilon(y) = \emptyset$ for all $x, y \in N$ by property (3) of density nets. If a node v has color a then it stores the label assigned to y by Theorem 2.42 (in the fixed-port model) for all $y \in V$ such that $h(y) = a$, which uses a total of at most $o(\frac{1}{\epsilon}\log^2 n)$ space.

Once we have found the label of the destination then we are done, since we can simply have every node remember the routing table of Theorem 2.42 and pretend that we are in the name dependent setting. Thus the problem is reduced to finding the node $w \in B'_u$ such that $color(w) = h(v)$. Unfortunately as previously mentioned doing name-independent compact routing in trees is hard, but Lemma 2.49 tells us that we can get around this. In particular, stretch is not the right measure: everything in $B_{u'}$ is close to u relative to v, so we can route on shortest paths to a constant number of destinations in B'_u before routing to w (and then to v). These other destinations might be very far away from w relative to u or $Net(u)$ and so would incur large stretch, but since our end goal is not w but v this is acceptable.

Intuitively this is reminiscent of unweighted routing, since we do not care about the weights of the edges in T'_u as long as we are routing along shortest paths to only a constant number of destinations. Indeed, we are able to use a result of Abraham, Gavoille, and Malkhi to do this:

Theorem 2.50 (Abraham, Gavoille, Malkhi [1]) *Every unweighted rooted tree with n nodes has a single-source name-independent routing scheme (in the designer port model) such that the distance traveled between the root r and a destination v is at most $d(r,v) + 2d(T)$, where $d(T)$ is the depth of the tree. Moreover, only $O(\log^4 n/(\log\log n)^2 + \log^3 n/\log\log n)$ bits are needed per node and headers have size $o(\log^2 n)$*

We cannot use this result as a black box since even though it makes a good guarantee about distances it does not make the necessary guarantee about the number of intermediate nodes routed to, so it could require routing directly to $\omega(1)$ intermediate nodes and thus incur too much weighted distance. But an examination of the proof of Theorem 2.50 reveals that it does in fact route to only one other intermediate node x, and routes along the shortest path from r to x, then back up to r, and then directly from r to w. They use another hash function H which maps the nodes of the tree onto a key space, and then assign these keys in a very careful manner so that interval routing on the key space can be done efficiently. This lets them find the node that owns $H(w)$, which will then contain the necessary information for routing to w. One fact to note about their theorem is that it uses the power of designer ports to rearrange the ports so that they

are in order of the size of the subtree rooted at each child. Putting this all together, we get the following theorem:

Theorem 2.51 *Let $G = (V, E)$ be a weighted graph. Then for any $0 < \epsilon < 1$ there is a name-independent designer-port routing scheme with ϵ-slack for G with routing tables of size $O(\frac{1}{\epsilon}\log^2 n + \log^4 n)$, headers of size $o(\log^2 n)$, and stretch 27.*

Proof: Each node has to remember the routing tables of Theorem 2.42, the routing tables and information for Theorem 2.50, and the name-dependent labels for up to $\frac{1}{\epsilon}$ other nodes. This totals $O(\frac{1}{\epsilon}\log n) + O(\log^4 n) + O(\frac{1}{\epsilon}\log^2 n) = O(\frac{1}{\epsilon}\log^2 n + \log^4 n)$ bits. The header contains at most a header from Theorem 2.42 and a header from Theorem 2.50, using at most $o(\log^2 n)$ bits.

Routing in this scheme will take place as follows. First, u will route up to $Net(u)$. Then using the scheme from Theorem 2.50 we route to the node which owns $H(h(v))$, where the name of a node is now its color. Then at $H(h(v))$ we find out how to get to the node colored $h(v)$ in B'_u, so we go there. Then at that node we find out the label from Theorem 2.42 for v, so we route to $Net(v)$ and then down to v. We can use Lemmas 2.49 and 2.10 to give a bound on the distances to and from $H(h(v))$ and $h(v)$, giving us

$$\begin{aligned} d'(u,v) &\leq d(u, Net(u)) + 2d(Net(u), H(h(v))) + \\ &\quad 2d(Net(u), h(v)) + d(Net(u), Net(v)) + d(Net(v), v) \\ &\leq 2d(u,v) + 8d(u,v) + 8d(u,v) + d(Net(u), u) + d(u,v) + d(v, Net(v)) + 3d(u,v) \\ &\leq 18d(u,v) + 2d(u,v) + d(u,v) + 3d(u,v) + 3d(u,v) \leq 27d(u,v) \end{aligned}$$

as claimed. ■

We can create a slightly different tradeoff between space and stretch by making one easy change: instead of having $H(h(v))$ remember how to get to $h(v)$, we just have it remember all of the name-dependent labels that $h(v)$ is supposed to remember. One of the lemmas proved in [1] about Theorem 2.50 is that each node remembers at most $O(\log n/\log\log n)$ different hash values, and at most $O(\log n)$ nodes hash to the same value. So the node that owns $H(h(v))$ might have to store the name-dependent labels originally stored by $O(\log^2 n/\log\log n)$ different nodes, for a total extra space of $O(\frac{1}{\epsilon}\log^4 n/\log\log n)$. The stretch is then reduced from 27 to 19, since we don't waste a trip from $H(h(v))$ to $Net(u)$ to $h(v)$. So we have as an easy corollary:

Corollary 2.52 *Let $G = (V, E)$ be a weighted graph. Then for any $0 < \epsilon < 1$ there is a name-independent designer-port routing scheme with ϵ-slack for G with routing tables of size $O(\frac{1}{\epsilon}\log^4 n/\log\log n + \log^4 n)$, headers of size $o(\log^2 n)$, and stretch 19.*

Fixed ports

Abraham, Gavoille, and Malkhi recently showed [5] that for every integer $k \geq 1$ there is a graph (in fact a star) such that any name-independent fixed-port routing scheme with stretch $2k-1$ requires $\Omega(n^{1/k})$ space. They also proved a polynomial lower bound on the space when only the *average* stretch is constant, which eliminates the possibility of good gracefully degrading routing schemes in this model (since any such gracefully degrading scheme has constant average stretch, following the proof of Corollary 2.48). We extend their basic argument to prove that for any $0 < \epsilon < 1/2$ with ϵ constant, no ϵ-slack scheme exists with constant stretch and space polynomial in $\frac{1}{\epsilon}$ and polylogarithmic in n.

The proof from [5] uses a distributed Kolmogorov complexity argument, where they fix some of the ports to a sequence with high Kolmogorov complexity and show that any routing scheme that does not use $\Omega(n^{1/k})$ space allows us to compute the sequence without using much space, giving a contradiction. In particular, they fix a sequence L of size $\lfloor n/2 \rfloor$ with the property that any subsequence or subset of L has large Kolmogorov complexity relative to its size, and then create a star network where the first $\lfloor n/2 \rfloor$ nodes are at distance 1 from the root and are on ports in L and the other $\lceil n/2 \rceil$ nodes are at distance k from the root. Let C be the set of nodes at distance 1 from the root, and let F be the others. In order to reach the nodes in C from the root with stretch at most $2k-1$, the routing scheme cannot route to any of the nodes in F. They then just consider destinations in C and show that being forced to never leave C except to go to the root forces the scheme to use polynomial space.

However, this is not true anymore when we allow constant slack. In particular, some of the destinations in C will be ignored by the root and so the routes to those nodes could go out to some nodes in F to get extra information. However, we can fix this by just considering the subset of destinations that we do not ignore, which will still have large Kolmogorov complexity since it has size at least a constant fraction of $|C|$. We also need the property that we can easily generate the initial outgoing headers for this set, which we can maintain by slightly perturbing the first $(1-\epsilon)n$ points to have distance $1+\delta$ from the root for some small constant $\delta > 0$. Then since the root is forced to ignore the closest ϵn nodes it cannot ignore any of these nodes, and we easily generate the outgoing headers since they are just all elements of $\{1, \ldots, (1-2\epsilon)n/2\}$. We omit the details since they are just a rehashing of [5]; it is straightforward to modify the proof of the lower bound in [5] using this new subset of C instead of all of C. This gives the following lower bound.

Theorem 2.53 *For each integer $k \geq 1$ and constant $0 < \epsilon < 1/2$, there is a weighted n-node star for which every name-independent fixed-port routing scheme with ϵ-slack and stretch $2k-1$ uses at least $\Omega(n^{1/k})$ bits of memory at some node.*

2.6 Approximation Algorithms for Spanner Problems

We now examine a slightly different topic: approximation algorithms for spanner problems. The motivation for this is simple, and corresponds with the motivation for looking at slack problems in the first place: the strong lower bounds are not realistic. Slack was one way around them, but looking at per-instance guarantees rather than global guarantees is another way. The lower bounds implied by Erdős's girth conjecture are of the form "there exist a family of graphs for which no good spanner exists". But many graphs are not like this, so we instead turn our attention to approximation algorithms, or per-instance guarantees, that are of the form "we can build a spanner that is close to the size of the best spanner, whatever that happens to be".

We will consider a few different spanner variants. In general, we are given a graph $G = (V, E)$ (possibly directed), a *length function* $\ell : E \to \mathbb{R}^+$, and a *weight function* $w : E \to \mathbb{R}^+$. The distance between any two vertices is defined to be the shortest path distance between them according to ℓ. We will want to find a subgraph H of G of minimum total weight such that the stretch between any two points is at most some parameter k (in this section we define a spanner to be what was called a subgraph spanner in Section 2.3). This is obviously equivalent

to finding a subgraph H such that the stretch of any edge in E is at most k. In the *basic* k-spanner problem, the graph G is undirected and both the length and the weight functions are always equal to 1, i.e. distances are the number of hops and the goal is to minimize the number of edges. In the *unit-length* k-spanner problem the length function is always 1 and the weight function is arbitrary, while in the *unit-weight* k-spanner problem the weight function is always 1 and the length function is arbitrary (note that this is the version of spanners discussed in Section 2.3). All of these versions can clearly be defined for both undirected and directed graphs.

The basic tool that we will use to design approximation algorithms is an LP relaxation of the k-spanner problem. For each edge $(x, y) \in E$, let $\mathcal{P}_{x,y}$ denote the set of paths in G from x to y with stretch at most k (i.e. the sum of the lengths of edges in P is at most $k \times \ell(u, v)$). For each edge $e \in E$ we will have a variable x_e, and for each path $P \in \mathcal{P}_{x,y}$ we will have a variable f_P. This gives the following linear program:

$$\begin{aligned}
\min \quad & \sum_{e \in E} w(e) x_e & & \\
\text{s.t.} \quad & \sum_{P \in \mathcal{P}_{x,y} : e \in P} f_P \leq x_e & & \forall (x, y) \in E, e \in E \\
& \sum_{P \in \mathcal{P}_{x,y}} f_P \geq 1 & & \forall (x, y) \in E \\
& x_e \geq 0 & & \forall e \in E \\
& f_P \geq 0 & & \forall (x, y) \in E, P \in \mathcal{P}_{x,y}
\end{aligned} \tag{2.1}$$

This LP is obviously a valid relaxation of the k-spanner problems. To see this, suppose that we are given a spanner H. Then for every edge (x, y) in the original graph there is a path in H of stretch at most k, i.e. there is some path from $\mathcal{P}_{x,y}$ in H. So we can set x_e to 1 for all edges e in H, and we will set f_P to 1 if P is in H. Since H is a valid spanner all the constraints are satisfied.

This LP has polynomial size for certain versions of the problem, for example the unit-length version in which k is a constant. In general, it will have polynomial size whenever the number of stretch k paths between two adjacent points is at most a constant, for example if k is a constant and the ratio between the length of the maximum edge and the length of the minimum edge is a constant. In these cases this LP can be solved in polynomial time using any polynomial time linear programming algorithm. In some cases, though, this LP has exponential size, and it is not obvious how to solve it in those cases. In fact, we will not solve the LP in these instances. We will instead *approximately* solve the LP.

We do this by using a different LP that is an edge based formulation instead of a path based formulation. We will again have a variable x_e for every edge, but we will change the flow variables from f_P to $f_{(u,v)}^{(x,y)}$, i.e. we will have a variable for every edge indicating how much flow is going from one endpoint to the other for every commodity (in the undirected case this will actually mean two variables per edge, one for each direction). The intuition behind this LP is simple: the first constraint forces every flow to obey the edge capacities, the second, third, and fourth constraints force a unit flow out of the source and into the sink (with flow conserved everywhere else) for every commodity, and the fifth constraint forces the "length" of each flow to have stretch at most k.

$$
\begin{aligned}
\min \quad & \sum_{e \in E} w(e) x_e & & \\
\text{s.t.} \quad & f_e^{(x,y)} \leq x_e & & \forall e \in E, (x,y) \in E \\
& \sum_{z:(u,z)\in E} f_{(u,z)}^{(u,v)} - \sum_{z:(z,u)\in E} f_{(z,u)}^{(u,v)} = 1 & & \forall (u,v) \in E \\
& \sum_{z:(z,v)\in E} f_{(z,v)}^{(u,v)} - \sum_{z:(v,z)\in E} f_{(v,z)}^{(u,v)} = 1 & & \forall (u,v) \in E \\
& \sum_{v:(u,v)\in E} f_{(u,v)}^{(x,y)} - \sum_{v:(v,u)\in E} f_{(v,u)}^{(x,y)} = 0 & & \forall (x,y) \in E, u \neq x, y \\
& \sum_{(u,v)\in E} \ell((u,v)) f_{(u,v)}^{(x,y)} \leq k\ell((x,y)) & & \forall (x,y) \in E, (u,v) \in E \\
& x_e \geq 0 & & \forall e \in E \\
& f_{(u,v)}^{(x,y)} \geq 0 & & \forall (x,y) \in E, (u,v) \in E
\end{aligned}
\tag{2.2}
$$

This LP clearly has polynomial size, and is also a relaxation of the k-spanner problems. It is a relaxation basically for the same reason that LP 2.1 is a valid relaxation: given a k-spanner H, we will set x_e to 1 for all edges in H, and for all edges $(x, y) \in E$ we will route one unit of flow from x to y through a path in $\mathcal{P}_{x,y}$ that is also in H, at least one of which must exist since H is a valid k-spanner. So we can solve this LP in polynomial size using any polynomial time linear programming algorithm and get a fractional solution. We can change this into a solution with a polynomial number of path-based flows rather than edge-based flows using standard techniques; for example, for every commodity we can find an arbitrary path from the source to the sink on which every edge has nonzero flow, assign the amount of flow on the bottleneck edge to that path, and then remove that flow from the original. In each iteration some edge gets pushed to have 0 flow, so this only runs a polynomial number of iterations so we end up with a path based solution that has polynomial support.

However, this still might not be a valid solution to LP 2.1. This is because the only guarantee we have on the path lengths is that for every edge $(x, y) \in E$, the sum of the lengths of the paths times the flow on the path is at most $k\ell((x, y))$. More formally, let $\mathcal{P}'_{x,y}$ be the set of *all* paths between x and y, not just the paths with stretch at most k. Then after converting our solution for LP 2.2 to a path-based solution we will have the property that $\sum_{P \in \mathcal{P}'_{x,y}} \ell(P) f_P \leq k\ell((u, v))$, where $\ell(P)$ is defined to be the sum of the lengths of the edges in P. This is weaker than the stretch condition of LP 2.1, since it just says that the average path used must have stretch at most k, not that all paths used must have stretch at most k. On the other hand, if the average stretch is at most k, then by Markov's inequality we know that at least $1/2$ of the flow is on paths with stretch at most $2k$. More generally, we know that at least $\frac{\epsilon}{1+\epsilon}$ flow is on paths with stretch at most $1 + \epsilon$. So if we increase all x values by a factor of $\frac{1+\epsilon}{\epsilon}$ and remove the flow on all paths of length more than $(1 + \epsilon)k$, we will have a solution to LP 2.1 for the $(1 + \epsilon)k$-spanner problem. This naturally gives rise to *bicriteria* approximation algorithms, where we will give a $(1 + \epsilon)k$-spanner that has weight at most an α factor larger than the optimum k-spanner. We will call such algorithms $(1 + \epsilon, \alpha)$-approximations. Slightly more formally, this analysis yields the following theorem:

Theorem 2.54 *If there is an algorithm that is an α-approximation for the k-spanner problem assuming it has a valid fraction solution for LP 2.1, then it can be modified to yield a polynomial time $(1+\epsilon, \frac{1+\epsilon}{\epsilon}\alpha)$-approximation algorithm.*

In this section we will explore these relaxations by using them to design approximation algorithms and analyzing their integrality gaps.

2.6.1 Approximation Algorithm for Unit-Weight k-Spanner

In this section we design a $\tilde{O}(n^{2/3})$-approximation algorithm for the unit-weight k-spanner problem (directed and undirected) as long as we can solve LP 2.1. If we cannot solve it we will give a bicriteria approximation by applying Theorem 2.54. For the undirected unit-weight k-spanner problem it is well known that there is always a k-spanner with at most $n^{1+\frac{2}{k+1}}$ edges, which is obviously a $n^{\frac{2}{k+1}}$ approximation since any spanner needs at least $n-1$ edges just to be connected. So our new algorithm is not an improvement over this existing algorithm. For the directed version, though, the best known algorithm is a $\tilde{O}(n^{1-1/k})$-approximation due to Bhattacharyya et al. [19] that works only for the basic directed k-spanner problem (unit lengths and weights). So for $k > 3$ our algorithm is better for the basic directed problem, and to the best of our knowledge it is the only known algorithm for the unit-weight directed k-spanner problem. For $k = 3$ there is also a previous $\tilde{O}(n^{2/3})$-approximation due to Elkin and Peleg [37] that does not use LP-based techniques, but their algorithm and analysis are significantly more complicated than ours.

We begin by assuming a fractional solution (x, f) for LP 2.1. For every (u, v) edge in E, let $N_{u,v} \subseteq V$ be the set of vertices that lie on a path of stretch at most k from u to v (i.e. the set of vertices that are used by at least one path in $\mathcal{P}_{u,v}$). The following lemma bounds how small the capacities in the solution can be relative to the size of these sets:

Lemma 2.55 *For any $(u, v) \in E$ there is a path $P \in \mathcal{P}_{u,v}$ with the property that every edge in e has $x_e \geq \frac{1}{|N_{u,v}|^2}$*

Proof: Suppose this is false for some (u, v). Let $B \subseteq N_{u,v} \times N_{u,v}$ be the set of edges with $x_e < 1/|N_{u,v}|^2$. Then every path $P \in \mathcal{P}_{u,v}$ goes through at least one edge in B, so these edges form a cut between u and v relative to the paths in $\mathcal{P}_{u,v}$. Since we have a valid LP solution, we know that at least one unit of flow is sent from u to v using paths in $\mathcal{P}_{u,v}$. This means that the number of edges in B must be at least $|N_{u,v}|^2$. But this is a contradiction: every edge in B has both endpoints in $N_{u,v}$, so there are at most $\binom{|N_{u,v}|}{2} < |N_{u,v}|^2$ of them. ■

So if $|N_{u,v}|$ is small, Lemma 2.55 implies that there is some stretch k path with the property that every edge is assigned a large capacity. This is good news for rounding, since it means that we will not have to round the capacities up by very much. But what if $|N_{u,v}|$ is large? Then there are many nodes that are on stretch k paths, so we should be able to find such a path by picking nodes randomly. This is formalized in the following lemma:

Lemma 2.56 *If we sample at least $\frac{3n \ln n}{|N_{u,v}|}$ vertices independently and uniformly at random, then with probability at least $1 - 1/n^3$ at least one sampled vertex will be in $N_{u,v}$*

Proof: The probability that no sampled vertex is in $N_{u,v}$ is at most

$$\left(1 - \frac{|N_{u,v}|}{n}\right)^{\frac{3n \ln n}{|N_{u,v}|}} \leq e^{-3 \ln n} = 1/n^3$$

and thus the probability that at least one sampled vertex is in $N_{u,v}$ is at least $1 - 1/n^3$ ∎

Our algorithm is quite simple, and is based on these two lemmas. We first do a threshold LP rounding: any edge e with $x_e \geq 1/(3n \ln n)^{2/3}$ is included in our spanner. We then randomly sample $(3n \ln n)^{2/3}$ vertices, and for each sampled vertex v we built a shortest path in-arborescence and a shortest path out-arborescence rooted at v (in the undirected case these will simply be the same shortest path tree).

Theorem 2.57 *This algorithm returns a valid spanner with probability at least $1 - 1/n$, and if it does return a valid spanner then it is a $O((n \ln n)^{2/3})$-approximation.*

Proof: We first prove that it results in a valid spanner. Consider some edge $(u, v) \in E$. If $|N_{u,v}| \leq (3n \ln n)^{1/3}$, then Lemma 2.55 implies that there is some stretch k path in which every edge e has $x_e \geq 1/(3n \ln n)^{2/3}$. So the threshold rounding step will add all of the edges of this path to our spanner, so there will be a path from u to v in our spanner with length at most $k\ell((u, v))$. On the other hand, if $|N_{u,v}| > (3n \ln n)^{1/3}$ then Lemma 2.56 implies that with probability at least $1 - 1/n^3$ we will have sampled some vertex in $N_{u,v}$. Suppose we sample $x \in N_{u,v}$. By the definition of $N_{u,v}$ we know that x is on some path from u to v with stretch at most k, and thus the length of the shortest path from u to x plus the length of the shortest path from x to v is at most $k\ell((u, v))$. Since we included both a shortest path in-arborescence and a shortest path out-arborescence rooted at x our spanner will include both of these shortest paths, and thus will include a path from u to v with stretch at most k. Now we can take a union bound over all such edges, and get that with probability at least $1 - 1/n$ *every* edge (u, v) with $|N_{u,v}| > (3n \ln n)^{1/3}$ has at least one vertex from $N_{u,v}$ in the sample set. So for any edge (u, v) either the LP rounding or the random sampling will guarantee a valid path of stretch at most k, and thus the graph we return is a valid spanner (with probability at least $1 - 1/n$).

To prove that it is a $O((n \ln n)^{2/3})$-approximation we will show that each step costs at most $O((n \ln n)^{2/3}) \times OPT$. This is obvious for the LP rounding step: every x_e is increased by at most a factor of $O((n \ln n)^{2/3})$ so the rounding costs at most $O((n \ln n)^{2/3})$ times the LP cost. Since the LP is a valid relaxation, this implies that the rounding costs at most $O((n \ln n)^{2/3}) \times OPT$. To show that the second step does not add many edges, consider some sampled vertex x. Suppose that in the original graph R_x nodes are reachable from x and S_x nodes have paths to x. Then obviously any spanner must have size at least $\max\{R_x - 1, S_x - 1\}$ just to maintain connectivity between nodes that should be connected. But adding a shortest path in-arborescence adds at most $S_x - 1$ edges, and adding a shortest path out-arborescence adds at most $R_x - 1$ edges. So the number of edges we added by sampling x is at most $S_x - 1 + R_x - 1 \leq 2 \times OPT$. Thus the total cost for the second step is at most $2(3n \ln n)^{2/3} \times OPT$ ∎

This gives a $\tilde{O}(n^{2/3})$ approximation to the directed unit-weight k-spanner problem as long as we can solve LP 2.1. If we cannot solve it, then combining Theorem 2.54 with Theorem 2.57 gives a bicriteria $(1 + \epsilon, \frac{1+\epsilon}{\epsilon} O((n \ln n)^{2/3}))$-approximation.

2.6.2 Unit-length 2-Spanner

The unit-length 2-spanner problem is qualitatively and quantitatively different from k-spanner with $k > 2$: it is known that it can be approximated to $O(\log n)$ and that this is tight (assuming $P \neq NP$). As part of our goal of understanding the spanner problem through LP relaxations,

we show that LP 2.1 has integrality gap of $O(\log n)$, and that there exist instances of unit-length 2-spanner on which the integrality gap is also $\Omega(\log n)$. In fact, these instances show large gap even when weights are also unit, i.e. when we are actually in the basic 2-spanner problem. For this section we will assume that we are in the undirected version, but the same techniques work in the directed version as well.

Lower Bound

We first show that the integrality gap is $\Omega(\log n)$ on certain instances. The intuition is that we will apply the hardness reduction from set cover to 2-spanner to an instance of set cover that has a large integrality gap. We first describe the generic reduction, then the particular set cover instance that we apply it to.

Suppose we have a (unweighted) set cover instance with elements U and sets $\mathcal{S}$, where $|U| = N$ and $|\mathcal{S}| = M$. We create a graph G with vertex set $U \cup \mathcal{S} \cup \{x_i : i \in [k]\}$, where $k = M^2$. In other words, there is a vertex for every element, a vertex for every set, and k new vertices $x_1, \ldots, x_k$. Clearly the number of vertices is polynomial in the size of the set cover instance (it is in fact $n = M^2 + M + N$). There is an edge from every x_i to every set node and to every element node, an edge between every two set nodes, and an edge between a set node $S \in \mathcal{S}$ and every $e \in U : e \in S$. More formally, the edge set is $\{\{x_i, S\} : i \in [k], S \in \mathcal{S}\} \cup \{\{x_i, e\} : i \in [k], e \in U\} \cup \{\{S, S'\} : S, S' \in \mathcal{S}\} \cup \{\{S, e\} : s \in \mathcal{S}, e \in U, e \in S\}$.

The set cover instance that we use has element set $\mathbb{F}_2^q \setminus \{\vec{0}\}$, so there are $2^q - 1$ elements. There is a set S_α for every $\alpha \in \mathbb{F}_2^q$ (so there are 2^q sets), where $S_\alpha = \{e \in \mathbb{F}_2^q \setminus \{\vec{0}\} : \alpha \cdot e = 1\}$. We are using the normal notion of dot product over $\mathbb{F}_2^q$ here, i.e. $\alpha \cdot e = \alpha_1 e_1 + \cdots + \alpha_q e_q \pmod 2$. It is easy to see that every element is in exactly half of the sets. This large amount of overlap intuitively allows the linear program to "cheat".

To see that the LP has a small solution, we will set the capacity of the edges between set vertices to 1, the edges between set vertices and element vertices to 1, and the edges between x_i vertices and element vertices to 0. We will also set the capacity of edges between x_i vertices and set vertices to $2/M$. Obviously this solution has cost at most $kM\frac{2}{M} + M^2 + MN = O(M^2) = O(n)$, so it remains to show that it is a feasible solution. To show this, for every edge in the original graph we need to find a way to route at least one unit of flow subject to our capacities from one endpoint to the other along paths of length at most 2. This is trivial for every edge that we set to have capacity 1, so we just need to worry about edges incident on x_i nodes. For edges of the form $\{x_i, S\}$ with $S \in \mathcal{S}$, we can send $1/M$ flow on every edge from x_i to $\mathcal{S}$ (including the edge from x_i to S, and then flow that was set to sets $S' \neq S$ can be forwarded along the $\{S', S\}$ edge. For edges of the form $\{x_i, e\}$ with $e \in U$, we can send $2/M$ flow from x_i to every set that contains e. Since exactly half of the sets contain e this adds up to a total flow of 1. This flow can then be forwarded directly to e, since there is an edge of capacity 1 between e and every set containing e. Thus this if a feasible solution to the flow LP of cost $O(n)$.

Now we want to show than any integral solution has cost at least $\Omega(n \log n)$. Consider some arbitrary integral solution (i.e. a setting of 0/1 capacities to every edge such that one unit of flow can be sent between the endpoints of any original edge using paths of length at most 2). Consider an edge $\{x_i, e\}$ with $e \in U$. Either this edge has capacity 1, or there is some $S \in \mathcal{S}$ with $e \in S$ such that the edges $\{x_i, S\}$ and $\{S, e\}$ both have capacity 1. This is because the only paths of

length at most 2 between x_i and e are paths of this form and the one direct edge. Since this is true for every e, the vertices adjacent to x_i must form a valid set cover of this original instance (where an edge directly to an element e is equivalent to adding the set $\{e\}$). Thus the degree of any x_i node must be at least the size of the smallest valid set cover. For our set cover instance, it is easy to see that the size of the smallest cover is at least q. To see this, suppose otherwise, i.e. assume there is some collection of sets $S_{\alpha_1}, \ldots, S_{\alpha_{q-1}}$ that covers the elements. Then $\cap_{i=1}^{q-1} \overline{S_{\alpha_i}} = \emptyset$, so $\cap_{i=1}^{q-1} \{e \in \mathbb{F}_2^q : \alpha_i \cdot e = 0\} = \{\vec{0}\}$. But this is a contradiction, since the intersection of $q-1$ hyperplanes in the q-dimensional vector space over $\mathbb{F}_2$ cannot be just a single point (that would require at least q hyperplanes). So any valid set cover has size at least $q \geq \log N$.

So now we know that any integral solution to the flow LP has cost at least $kq \geq M^2 \log N = \Omega(n \log n)$, thus proving that the integrality gap of the flow LP is at least $\Omega(\log n)$.

Upper Bound

To see that the LP has an integrality gap of $O(\log n)$ we will essentially do the reverse and reduce 2-spanner to set cover. First, some notation: for each vertex $v \in V$, let $N(v) \subseteq V$ be the set of neighbors of v in G. For each $v \in V$ and $U \subseteq N(v)$, let $S_v^U = \{\{v,u\} \in E : u \in U\} \cup \{\{w,z\} \in E : w,z \in U\}$, i.e. S_v^U consists of edges in which both endpoints are contained in U or one endpoint is v and the other is in U. The intuition is that if we include all edges from v to nodes in U in our spanner, then S_v^U are the edges that are now stretched by at most 2.

This leads to the obvious set cover formulation: the elements are edges in E, the sets are $\{S_v^U\}_{v \in V, U \subseteq N(v)}$, and the cost of a set S_v^U is $c(v,U) = \sum_{u \in U} w(\{u,v\})$ (i.e. the cost of adding all the edges between v and U to the spanner). As a side note, the existing $O(\log n)$-approximation algorithms for 2-spanner [61, 63] are basically just the basic greedy algorithm for this set cover instance, although they don't phrase it that way. The only difference is that in their algorithms the cost of a set changes throughout the algorithm, since if an edge has already been included then it's cost should not be factored into the cost of any set.

This set cover formulation leads to the obvious related IP and LP relaxation:

$$\begin{aligned} \min \quad & \sum_{v \in V} \sum_{U \subseteq N(v)} c(v,U) y_v^U \\ \text{s.t.} \quad & \sum_{v \in V} \sum_{U \subseteq N(v): e \in S_v^U} y_v^U \geq 1 \quad \forall e \in E \end{aligned} \tag{2.3}$$

Since this is a set cover LP, we know that it has integrality gap of $O(\log n)$ (note that this is irrespective of whether or not it is an exact formulation of 2-spanner). Let LP_{flow} and IP_{flow} denote the optimum fractional and integer values respectively LP 2.1, and let LP_{SC} and IP_{SC} denote the optimum fractional and integer values of this second LP. We want to show that $IP_{flow} \leq O(\log n) LP_{flow}$, and we know that $IP_{SC} \leq O(\log n) LP_{SC}$. So to finish off we will show that $IP_{flow} \leq O(IP_{SC})$ and that $LP_{SC} \leq O(LP_{flow})$.

Lemma 2.58 $IP_{flow} \leq IP_{SC}$

Proof: Since these are the optimum values of minimization problems, in order to show this we just need to show how to transform any integer set cover solution into an integer flow solution of no larger cost. Let y be a solution to the set cover IP. To build a flow solution, set $x_{\{u,v\}} = 1$ if

there is some $U \subseteq N(v)$ such that $u \in U$ and $y_v^U = 1$ or if there is some $U \subseteq N(u)$ such that $v \in U$ and $y_u^U = 1$. We first claim that the cost of this flow solution has not gone up, which is easy to see since any edge for which we set $x_e = 1$ (and thus cost us $w(e)$) contributes $w(e)$ to the cost of at least one set S_v^U with $y_v^U = 1$. Now we claim that this is a feasible integer flow solution, which means that we need to find a way to route every edge in E along a path of length at most 2 using only edges with $x_e = 1$. Let $\{u, v\} \in E$ be an arbitrary edge. Since it is covered by the set cover solution, there is some v and $U \subseteq N(v)$ with $y_v^U = 1$ and $\{u, v\} \in S_v^U$. This means that there is either a length 1 path or a length 2 path (through v) between u and v consisting of edges that were set to have $x_e = 1$, and thus we can route a feasible flow. ■

Lemma 2.59 $LP_{SC} \leq O(LP_{flow})$

Proof: To prove this lemma we show how to convert a fractional flow into a fractional set cover that costs at most 8 times as much. Let x, f be a solution to LP 2.1. The cost of this solution is $\sum_{e \in E} w(e)x_e$. We first increase all x and f values up to the next power of 2, giving us a new feasible flow solution x', f' with $\sum_e w(e)x'_e \leq 2\sum_e w(e)x_e$. For all $v \in V$ and $i \in \mathbb{Z}_{\geq 0}$, let $U(v, i) = \{u \in N(v) : x'_{\{u,v\}} \geq 1/2^i\}$. Now to create a fractional set cover we will just set $y_v^{U(v,i)} = 1/2^i$ for all $v \in V$ and nonnegative integers i.

We first claim that this is a feasible fractional set cover. To see this, consider some edge $\{u, v\} \in E$. In the fractional flow solution x', f' at least one unit of flow is sent on paths of length at most 2 between u and v. So some amount of flow is sent directly along edge $\{u, v\}$, some is sent from u to some intermediate vertex z_1 and then to v, some through intermediate vertex z_2, etc. The flow sent along the path $u - z_k - v$ is at most $\min\{x'_{\{u,z_k\}}, x'_{\{v,z_k\}}\}$. Suppose this value is $1/2^i$. Then note that both u and v are in $U(z_k, i)$, so the variable $y_v^{U(v,i)}$ fractionally covers the edge $\{u, v\}$ by $1/2^i$. So any path carrying flow in the flow solution has a corresponding set that covers to the same amount.

Slightly more formally, for any three vertices $z, u, v \in V$ with $u, v \in N(z)$, let $I(z, u, v) = \log \max\{\frac{1}{x'_{\{u,z\}}}, \frac{1}{x'_{\{v,z\}}}\}$. Note that the maximum flow sent between u and v via z is at most $1/2^{I(z,u,v)}$. Then

$$\begin{aligned}\sum_{z \in V} \sum_{U \subseteq N(z): \{u,v\} \in S_z^U} y_z^S &\geq y_u^{U(u, \log(1/x'_{\{u,v\}}))} + y_v^{U(v, \log(1/x'_{\{u,v\}}))} + \sum_{z \in V: u,v \in N(z)} y_z^{U(z, I(z,u,v))} \\ &\geq x'_{\{u,v\}} + x'_{\{u,v\}} + \sum_{z \in V: u,v \in N(z)} 1/2^{I(z,u,v)} \\ &\geq \sum_{P \in \mathcal{P}_{u,v}} f'_P \geq 1\end{aligned}$$

and thus edge $\{x, y\}$ is covered, so the y values we set do indeed form a feasible fractional set cover.

Now we need to argue that the cost of this fractional set cover is close to the cost of the fractional flow. Note that by setting $y_v^{U(v,i)}$ to $1/2^i$ we have incurred cost $\sum_{u \in U(v,i)} c_{\{u,v\}}/2^i$. So

the total cost of our fractional set cover is

$$\begin{aligned}
\sum_{v \in V} \sum_{i} c(v, U(v,i)) y_v^{U(v,i)} &= \sum_{v \in V} \sum_{i} \sum_{u \in U(v,i)} w(\{u,v\})/2^i \\
&= \sum_{v \in V} \sum_{i} \sum_{u \in N(v): x'_{\{u,v\}} \geq 1/2^i} w(\{u,v\})/2^i \\
&\leq \sum_{v \in V} \sum_{u \in N(v)} 2w(\{u,v\}) x'_{\{u,v\}} \\
&= 4 \sum_{e \in E} w(e) x'_e \\
&\leq 8 \sum_{e} w(e) x_e
\end{aligned}$$

Thus the cost of our partial set cover is at most 8 times the cost of the original flow solution, so $LP_{SC} \leq O(LP_{flow})$ as claimed. ■

Chapter 3

Wireless Network Capacity and Scheduling

In this chapter we switch our attention to wireless networks, which have their own unique characteristics including broadcast rather than point-to-point communication, the ability of multiple transmissions to use the same channel, and somewhat more complex requirements for a transmission to be "successful". Because of these differences, there are various different models of wireless communications, all of which are different than the standard models of wired communications. One fundamental problem in which the specifics of the model make a large difference (and which demonstrates how different the wireless settting is from the wired setting) is the problem of *network capacity* or *one-shot scheduling*. In this problem we are given a wireless network and a collection of transmission requests and simply want to maximize the number of simultaneous successful transmissions. Obviously when considering this question the major modeling issue is how we model interference, or equivalently how we determine what sets of transmitters can successfully simultaneously transmit. In this thesis we will consider two basic models: the *protocol model* and the *physical model* (although most of the results will be in the more interesting physical model).

In the protocol model there is some interference graph on the desired transmissions, and a transmission is successful if and only if none of the neighbors of the transmission in this graph also chose to transmit. It is obvious from this definition that maximizing network capacity is the same problem as finding a maximum independent set in the interference graph, which is a famous and well-studied problem in its own right. In the context of this problem, further assumptions are usually made about the structure of the interference graph, since physical constraints make it unlikely that this graph is totally arbitrary. One typical assumption is that it is a *unit disk graph* (UDG), which basically means that transmitters interfere if they are too close to each other. But even in this setting finding the maximum independent set is NP-hard [24], although there are simple polynomial time approximation schemes [39, 53, 70]. There has also been a considerable line of work on weakening this assumption or on variants of it, including the Tx model of [91] and the growth-bounded model of [77] and [64].

In the physical model, on the other hand, we do not assume the existence of an interference graph. Instead we let every transmitter choose a power to broadcast at, give a rule for how that power fades with distance, and say that a transmission is successful if and only if the received signal divided by the sum of the interference and background noise is at least some threshold. This model is significantly more complicated than the protocol model, for a variety of reasons. In

the protocol model the success of a transmission depends only on the OR of its neighbors; if any of its neighbors transmit then it fails, no matter whether one or 10 transmitted, and any number of transmitters outside of its neighborhood can transmit without affecting its success. But in the physical model interference accumulates and spreads out to infinity, so not only is the decision function more complicated than an OR of neighbors it actually depends on every transmitter in the entire network. While not all of the assumptions in the physical model are absolutely true, it is commonly thought to be a more accurate model of reality than the protocol model.

Furthermore, there is a difference between *centralized* and *distributed* algorithms. While studying the fundamental computational problem is interesting, in many (perhaps most) real world situations there is no central authority to run the algorithm and tell all of the transmitters what to do. Ideally each transmitter would make its own decisions about whether to broadcast (and in the physical model, how much power to use). In the protocol model, since we have an interference graph we can simply abstract out to the graph and run a normal distributed protocol on this graph, and indeed this problem is usually classified under "distributed maximum independent set". In the physical model, however, there is no underlying communication or interference graph so coordination is more complicated. And even in the protocol model, using standard models for distributed algorithms are problematic: do transmitters really know their neighbors? Can they send different messages to different transmitters? Can a transmitter receive multiple messages at the same time?

In this thesis we try to work in a more general model that assumes less about inter-node communication. We consider arbitrary networks in both the physical and protocol models, where every transmitter knows only what happens to its transmissions. We do not even assume that neighbors in the protocol model can communicate (although our assumption about reception knowledge is equivalent to every node knowing whether or not at least one of its neighbors attempted to transmit), and in fact we design algorithms assuming that they cannot. We show that even in this extremely general model, there are simple algorithms that can guarantee that the average number of successful transmissions is a good approximation to the optimal solution. While distributed approximation algorithms for maximum independent set are well studied (e.g. [77]), this is, to the best of our knowledge, the first result that does not include communication among nodes, just information about the result of a transmission. This is also the first decentralized algorithm with provable approximation guarantees in the physical model, which is perhaps a more interesting result as until recently we did not even know of a good centralized algorithm in this model [9, 45, 46].

Moreover, the inspiration and techniques we use come not from the distributed computing literature, but instead from the algorithmic game theory and learning theory literature. In particular, we study a notion that generalizes the well-known price of anarchy: the *price of total anarchy*, originally defined by [20] as a way of weakening the rationality assumption behind the price of anarchy. But by definition there are algorithms that do almost as well as the price of total anarchy, unlike the price of anarchy. So by proving that the price of total anarchy is small we have actually proved that if every transmitter runs a so-called *no-regret* algorithm then the average performance will be good. This is a powerful tool when designing distributed algorithms since it allows the algorithm designer to prove approximation guarantees for distributed algorithms simply by proving no-regret for a centralized algorithm. We hope that this technique for designing distributed algorithms will prove useful for other problems, and believe that maximiz-

ing network capacity is simply one of a number of problems in which the price of total anarchy can be bounded.

We will begin by proving that maximizing wireless network capacity is NP-hard in the physical model. We call this the MAX-CONNECTIONS problem. We will then describe two relatively simple approximation algorithms. Since in reality we want distributed protocols and algorithms in wireless networks, we then prove a bound on the price of anarchy of a natural game. Finally, we will extend the analysis used to prove the price of anarchy to give a distributed algorithm based on no-regret algorithms and the price of total anarchy which works in both the protocol and the physical models.

3.1 Wireless Models

In the protocol model every transmitter can either transmit or not transmit, and a particular transmitter is successful if and only if it chooses to transmit and none of the neighbors of its connection in the interference graph choose to transmit. We will briefly consider general graphs, but will spend most of our time on *locally growth-bounded* graphs, which are a generalization of the *growth-bounded* graphs of [64]:

Definition 3.1 *A graph $G = (V, E)$ is* locally growth-bounded *if there is some constant k such that for every node $v \in V$ the size of a maximum independent set in $N(v) \cup \{v\}$ is at most k, where $N(v)$ denotes the set of neighbors of v in G.*

We note that, as pointed out by [77], growth-bounded graphs generalize unit disk graphs, quasi-unit disk graphs, unit ball graphs, and other popular generalizations of UDGs, and therefore locally growth-bounded graphs also generalize these models.

For the physical model we consider a set of n connections in the plane, where each connection has a transmitter t_i and a receiver r_i. For two points u and v in the plane, let $d(u, v)$ be the normal Euclidean distance between them. Suppose that u is broadcasting with power p. Following the model from [76] and [9], the signal strength at v is $P_r(u, v) = p \cdot \min\{(d_0/d(u, v))^\alpha, 1\}$, where α and d_0 are some parameters that we assume are constants. We will also make the standard assumption that $\alpha > 2$ (this assumption was used in [9, 45, 46], among others). Note that this model allows nodes to be arbitrarily close together, and just caps the received power by what happens at distance d_0. This model generalizes the model from much of the previous work in which $d_0 = 1$ and all distances are at least 1 [45, 68, 69].

A transmission from t_u to r_u is successful if the ratio of the received signal strength to the interference is at least some threshold τ; that is, if

$$\frac{P_r(t_u, r_u)}{\sum_{v \neq u} P_r(t_v, r_u)} \geq \tau$$

We will sometimes call this an *SINR constraint*.

For ease of presentation, we will assume throughout this paper that the maximum power of a transmitter is 1. All of our results hold for an arbitrary maximum power. We will sometime make the simplifying assumption that there is no background noise, i.e. the only causes of interference at a receiver are the signals of other transmitters. Most of our results still hold with nonzero

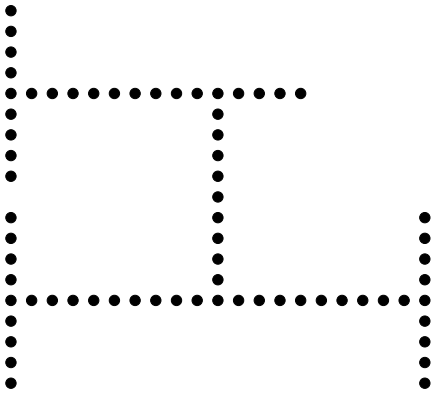

Figure 3.1: The graph that forms the basis of our NP-hardness reduction.

background noise as long as we are guaranteed that every transmitter-receiver pair is at a distance bounded away from their absolute physical limit, i.e. there is some constant $\delta > 0$ such that $d(t_i, r_i) \leq (1-\delta)(\frac{1}{\tau W})^{1/\alpha}$ where W is the background noise. We will let $d_{\max} = \max_i d(t_i, r_i)$ be the maximum distance between any transmitter-receiver pair.

3.2 NP-hardness

In this section we show that the MAX-CONNECTIONS problem in arbitrary networks under the physical model is NP-hard. Our reduction follows the basic strategy of the NP-hardness reduction for Maximum Independent Set (MIS) in unit disk graphs. However, the reduction is somewhat more complicated since we have to deal with the fact that interference comes from arbitrary distances. The reduction starts from the NP-hardness of MIS in planar cubic graphs. Specifically it is known (see e.g. [24]) that MIS is NP-hard in graphs where all nodes are on the edges of a grid with squares of size M, edges are of size 1, all nodes have degree at most 3 and each degree 3 node is incident to linear arrays of size at least $M/4$ (see Figure 3.1). Note that any maximum independent set will include at most every other node along an edge of the grid. The proof becomes somewhat complex since we need to show that all power levels will lead to an infeasible solution for any non-independent set.

We now describe a gadget that will be used in the eventual hardness proof. The purpose of the gadget is to represent a degree-3 node in our grid. We consider three linear arrays of nodes. (See Figure 3.2.) Each node serves as both the transmitter and receiver for a single connection. The first linear array is at positions $(0, 1.2), (0, 2.2), (0, 3.2), \ldots$. The second linear array is at positions $(1.2, 0), (2.2, 0), (3.2, 0), \ldots$. The third linear array is at positions $(0, -1.2), (0, -2.2), (0, -3.2), \ldots$. Lastly we have a single node at $(0, 0)$. We let the path-loss exponent $\alpha = 2.05$, the signal-to-noise ratio threshold $\tau = 1.00001$ and the maximum power $p_{\max} = 1$. We also suppose that each linear array has a at least ℓ nodes for some parameter ℓ. The first result about this gadget follows directly from the chosen value of τ.

Lemma 3.2 *There is no feasible solution that contains adjacent nodes from one of the linear arrays.*

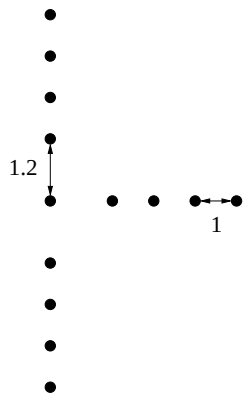

Figure 3.2: The gadget.

Proof: Consider two adjacent nodes from a linear array. The distance between them equals 1. Consider the transmission with the smallest power. The SINR for that transmission will be at most 1. Hence the SINR constraint is not satisfied. ■

Hence it remains to see what configurations are feasible that only use alternating members of a linear array. The following facts can be verified numerically.

Lemma 3.3 *The following configurations are feasible for arbitrarily large ℓ, even when there is a background noise level of $\varepsilon = 0.01$.*

- $(0,0), (0,2.2), (0,4.2), \ldots, (2.2,0), (4.2,0), \ldots, (0,-2.2), (0,-4.2), \ldots$ *(See Figure 3.3 (left).)*
- $(0,1.2), (0,3.2), \ldots (1.2,0), (3.2,0), \ldots, (0,-1.2), (0,-3.2), \ldots$. *(See Figure 3.3 (right).)*

For sufficiently large ℓ the following configurations are not feasible, even if there is no background noise level.

- $(0,0), (0,1.2), (0,3.2), \ldots, (2.2,0), (4.2,0), \ldots, (0,-2.2), (0,-4.2), \ldots$ *(See Figure 3.4 (left).)*
- $(0,0), (0,2.2), (0,4.2), \ldots, (1.2,0), (3.2,0), \ldots, (0,-2.2), (0,-4.2), \ldots$ *(See Figure 3.4 (middle).)*
- $(0,0), (0,2.2), (0,4.2), \ldots, (2.2,0), (4.2,0), \ldots, (0,-1.2), (0,-3.2), \ldots$ *(See Figure 3.4 (right).)*

It is easy to see that by making M sufficiently large we can guarantee that for any node a the interference caused to a by nodes at distance at least $M/4$ from a is at most ε. Note that M will depend only on ε. It is also easy to see from Lemma 3.3 that in a single linear array it is feasible for every other node to transmit at $p_{max} = 1$ even with background noise of 0.01.

We can use the above gadget to show NP-hardness in the following manner. First we can make sure that in the grid example where MIS is hard every node on the corners of the grid have degree 3 and every other node has degree 1 or 2. (See Figure 3.1.) We then place a copy of the gadget around every degree-3 node so that the linear arrays correspond to degree 1 or 2 nodes.

For the first direction of the reduction we would like to show that for any MIS in the original graph, the corresponding nodes can transmit in our wireless instance. This is easy to see by using Lemma 3.3, since close to the center of each gadget we know that the interference from outside the gadget is at most ε, so it is still feasible. The only non-obvious case is when two

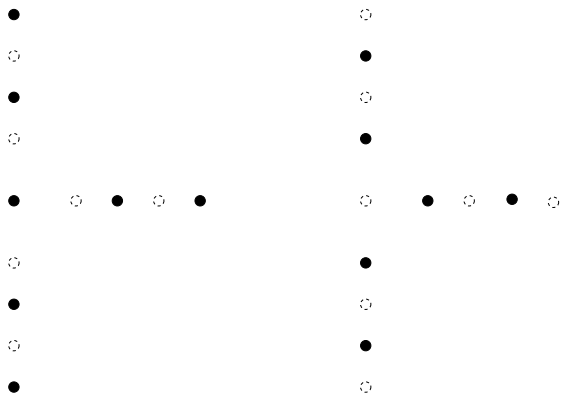

Figure 3.3: The feasible configurations.

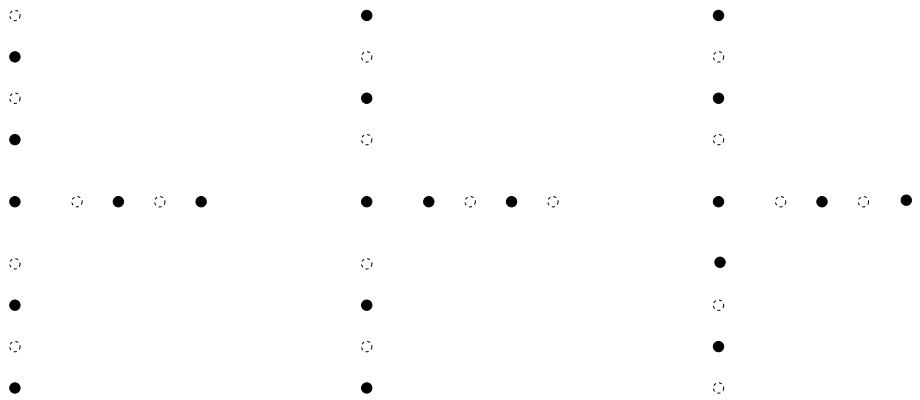

Figure 3.4: The infeasible configurations.

gadgets meet at the center of a chain, but this is clearly still feasible since at the center of the chain everything within distance $M/4$ is just part of the chain, so is still feasible by broadcasting at power 1 (which is consistent with the feasible gadget solution).

Now we need to show that any maximum feasible solution forms an independent set in the original graph. An important observation is that we can without loss of generality assume that in any maximum feasible solution every other node in a linear array is transmitting. If not, then we could always add to the number of nodes transmitting in the linear array by turning off one of the degree 3 nodes. We can repeat this process until every linear array has half its nodes transmitting. Lemma 3.3 then implies that we cannot have a degree 3 node transmitting together with one of its neighbors, and Lemma 3.2 implies that no other adjacent nodes are transmitting. This any maximum feasible set also forms an independent set, completing the reduction.

3.3 Approximation algorithms

Due to the NP-hardness of our problem we now turn our attention to approximation algorithms. Ideally we would like to adapt one of the polynomial time approximation schemes (that give a $(1+\varepsilon)$-approximation for any ε) for MIS on UDGs to the physical model. Unfortunately we are unable to do that, mainly because the analyses of these algorithms make critical use of the fact that two transmissions only interfere if the transmitters are close to each other. However, in the physical model interference can occur at arbitrary distances which makes it difficult to directly adapt these algorithms. However, in this section we show that if $d_{\max}$ is constant then we can obtain constant approximation algorithms in polynomial time. More generally, we present an $O(\log d_{\max})$-approximation that runs in polynomial time, and for the case in which the background noise $W = 0$ we give an $O(1)$-approximation that runs in time $O(n^{d_{\max}^2})$.

Before we present these algorithms we start with a *density lemma* that we shall use both for these results and for our game-theoretic results in Sections 3.4 and 3.5. This lemma states that any feasible solution can only have a limited number of receivers in any fixed area.

Lemma 3.4 *Consider a square S with side-length d_0. In any feasible solution the maximum number of connections with a receiver in square S is $3^\alpha/\tau$.*

Proof: Without loss of generality we assume that the background noise is 0. Having a non-zero background noise can only reduce the number of connections that can be supported.

Suppose that all nodes in the feasible solution transmit at a power such that the received signal is a constant $\bar{p}$, i.e. $p_i \min\{1, (d_0/d(t_i, r_i))^\alpha\} = \bar{p}$. Let i and i' be two connections such that both r_i and $r_{i'}$ lie in S.

The interference caused by connection i at receiver $r_{i'}$ is at least $p_i \cdot \min\{1, (d_0/d(t_i, r_{i'}))^\alpha\} \geq p_i \min\{1, (d_0/(d(r_i, r_{i'}) + d(t_i, r_i)))^\alpha\}$. By the geometry of the square S we know that $d(r_i, r_{i'}) \leq 2d_0$, which implies that $p_i \min\{1, (d_0/(d(r_i, r_{i'}) + d(t_i, r_i)))^\alpha\} \geq \frac{1}{3^\alpha} p_i \min\{1, (d_0/d(t_i, r_i))^\alpha\} \geq \frac{\bar{p}}{3^\alpha}$. Since the actual received signal strength of a connection is $\bar{p}$, if there are more than $3^\alpha/\tau$ such connections the interference experienced by all of them would be enough to prevent the SINR constraint being satisfied for *all* connections.

We now remove the condition that the received powers for every connection are the same. However, in this case the SINR value for some connection must be worse than it was when the received signal powers were the same. This implies that if there are more than $3^\alpha/\tau$ connections,

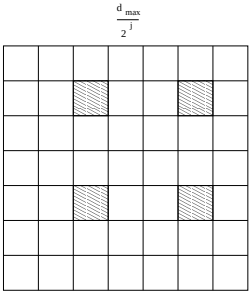

Figure 3.5: We form our solution using 1 out of every k^2 squares. Here $k = 3$.

then for any set of transmission powers there will be *some* connection whose SINR constraint is not satisfied. ■

Corollary 3.5 *Suppose now that square S has side-length d. In any feasible solution the maximum number of connections with a receiver in square S is $3^\alpha d^2/\tau(d_0)^2$.*

Proof: Divide square S up into subsquares of size d_0 and then apply Lemma 3.4. ■

Lemma 3.6 *Now consider a ball B of radius d. In any feasible solution the maximum number of connections with a receiver in ball B is $3^\alpha \cdot 4d^2/\tau(d_0)^2$.*

Proof: Follows immediately from the fact that any circle with radius d is contained in a square with side-length $2d$. ■

The following extension of Lemma 3.4 will also be useful.

Lemma 3.7 *Consider a square S with side-length d. In any feasible solution the maximum number of connections such that $d(t_i, r_i) \geq d$ and r_i is in square S is $3^\alpha/\tau$.*

Proof: The analysis is almost identical to that of Lemma 3.4 once we note that in this case $d(r_i, r_{i'}) \leq 2d \leq 2d(t_i, r_i)$ for all i, i'. ■

In the next two theorems we present our approximation algorithms for the MAX-CONNECTIONS problem in the SINR model.

Theorem 3.8 *There exists a polynomial time algorithm that always finds a solution to* MAX-CONNECTIONS *that is within a factor $O(\log d_{\max})$ of optimal.*

Proof: We divide all connections into classes based on distance. Class F_j contains all connections i such that $d_{\max}/2^{j-1} \geq d(t_i, r_i) \geq d_{\max}/2^j$. Note that in the optimal solution there must exist a j such that F_j contains $\text{OPT}/\log d_{\max}$ connections. In the following we will consider each j in turn and obtain a constant approximation for the connections in F_j only. We focus on a j for which $d_{\max}/2^j \geq d_{\min}$. The connections for which $d_{t_i,r_i} = 0$ can be handled similarly.

We now divide the problem into squares of side $d_{\max}/2^j$. (See Figure 3.5.) We refer to these squares as *j-squares*. From each j-square S, if there is at least one receiver in S then we choose one arbitrarily, and restrict ourselves to the problem on these connections. Note that Lemma 3.7 implies that each j-square only contains at most $3^\alpha/\tau$ receivers from the optimal solution on F_j, so as long as we can support at least a constant fraction of our chosen connections we are still within a constant of the optimal solution on F_j.

We now restrict our attention to 1 out of every k^2 j-squares in an evenly spaced pattern for some parameter k, i.e. squares located at the same coordinates mod k. (See Figure 3.5). We can

partition the plane into k^2 such sets of squares. We show that in each set we can support one connection in each square, so by taking the best set we are only losing another k^2 factor.

Consider some j-square S, and consider the set I of j-squares in the same pattern set that are offset from S by exactly ik in one coordinate and at most ik in the other coordinate (i.e. the set of j-squares that are on the border of the ℓ_∞ ball of radius ik around S). Since the ℓ_∞ distance is at most the normal ℓ_2 distance, it is not hard to verify that the maximum interference caused by the connection in I to the connection in S is at most $p_{\max}(d_0 2^j/(ik-3)d_{\max})^\alpha \leq p_{\max}(d_0 2^j/(i(k-3))d_{\max})^\alpha$. It is also easy to see that there are at most $8i$ squares in I. This implies that the total interference suffered by the connection in S is at most

$$\sum_{i=1}^{\infty} 8ip_{\max}\left(\frac{d_0 2^j}{(i(k-3))d_{\max}}\right)^\alpha = 8p_{\max}\left(\frac{d_0 2^j}{(k-3)d_{\max}}\right)^\alpha \zeta(\alpha-1)$$

where $\zeta(\alpha-1)$ is the Riemann zeta function, which is constant for constant $\alpha > 2$.

The connection in S can use power $p_{\max}$, so if $W = 0$ then the connection in S can be supported as long as

$$\frac{p_{\max}/(d_{\max}/2^{j-1})}{8p_{\max}\left(\frac{d_0 2^j}{(k-3)d_{\max}}\right)^\alpha \zeta(\alpha-1)} \geq \tau.$$

And thus by the same argument, so can all of the rest of the connections in I. So it suffices to choose k such that $(k-3)^\alpha \geq 8d_0 2^\alpha \zeta(\alpha-1)$, and thus k is some constant. If $W \neq 0$ then we have to increase k by a constant factor depending only on δ (recall that δ is a measure of how far $d_{\max}$ is from the physical limit). The approximation factor that we lose for class F_j due to all the connections that have been removed is $\frac{3^\alpha}{\tau}k^2$ which is a constant for fixed α and τ. As already mentioned, our overall approximation ratio is therefore $O(\log d_{\max})$. ■

If there is no background noise we can obtain another algorithm whose approximation ratio has a better dependence on $d_{\max}$ at the expense of a worse dependence in the running time. The algorithm is extremely similar to the proof of Theorem 3.8 except for using squares of size $d_{\max}$, scaling so total power from a square is $p_{\max}$, and dropping half of the connections in each square.

Theorem 3.9 *For the case with no background noise (i.e. $W = 0$) we can find an $O(1)$ approximate solution in time $n^{O((d_{\max}/d_0)^2)}$.*

Proof: We divide the plane into squares of size $d_{\max}$. Consider one such square S. We then divide this square into subsquares of size d_0. By Lemma 3.4, the number of connections that have a receiver in a square of size d_0 in any feasible solution is at most $3^\alpha/\tau$. Therefore the number of connections that have a receiver in square S in any feasible solution is $(d_{\max}/d_0)^2 3^\alpha/\tau$. Hence we can find the optimum solution for connections with a receiver in S in time $n^{(d_{\max}/d_0)^2 3^\alpha/\tau}$ by trying all possible subsets. Let OPT_S denote this set of connections for S and their power assignment.

Now for every square S we scale the powers of the transmitters in OPT_S so that the sum of their powers is 1. Note that any receiver in S that was part of a connection from OPT_S is still feasible relative to the other connections in OPT_S. This is because there is no background noise, so scaling powers up or down by the same factor does not affect feasibility. Obviously combining all of these solutions for every square S might result in an infeasible solution, but certainly $OPT \leq \sum_S OPT_S$.

We now let k be a number such that

$$\sum_{u=-\infty \wedge u \neq 0}^{\infty} \sum_{v=-\infty \wedge u \neq 0}^{\infty} \frac{1}{((k-3)(u+v)d_{\max}))^{\alpha}} < \frac{1}{(6d_{\max})^{\alpha}}.$$

Note that it suffices for k to be a constant. We can now say that if we consider sets of squares in which we include one out of every kth square in the horizontal direction and one out of every kth square in the vertical direction, and limit the total transmission power to 1 in each such square, the total interference experienced at any receiver in $\bar{S}$ is at most $\frac{1}{(6d_{\max})^{\alpha}}$. Now we consider what happens if we remove half of the transmissions from OPT_S. If we remove the half of the transmissions with the maximum transmission power, the reduction in the interference at the remaining transmissions is at least $\frac{1}{(6d_{\max})^{\alpha}}$. Hence the increase in interference from the additional squares is compensated for by the reduction in interference due to connections with receivers in S. By choosing the best of the k^2 square classes, and then losing at most half of the transmissions in any square, we get a $1/(2k^2) = O(1)$-approximation. ■

3.4 Game Theory

As discussed, the approximation algorithms described previously are centralized. We would also like to examine highly distributed algorithms that allow each transmitter to make its own decision based on limited local information. One extreme version of this is the setting in which transmitters are not allowed to exchange any information between themselves, and instead must make a decision on broadcast power based only on knowledge of the signal and noise at their receivers (we assume that receivers periodically provide this information to their transmitters). A natural way of viewing this setting is as a game where the transmitters are the players and the pure strategies are power settings. In this section we will define such a game and show that every Nash equilibrium in this game results in an expected number of successful transmissions that is close to optimal if there is no background noise.

For simplicity of notation we will without loss of generality rescale powers and W so that $p_{\max} = 1$. The game that the transmitters will be playing is simple. Each transmitter is a player, whose pure strategies are the reals in $[0, 1]$, with a nonzero value representing broadcasting at that power and 0 representing not broadcasting. So a mixed strategy is a probability distribution over $[0, 1]$. A transmitter gets payoff 0 if it does not broadcast (i.e. has power 0), payoff 1 if it broadcasts and its receiver has signal to noise ratio at least τ, and -1 if it broadcasts but its receiver has SINR less that τ. We note that it is easy to see that the same game without the -1 penalty can have bad Nash equilibria (in particular, everyone broadcasting). We first discuss pure Nash equilibria in this game, and then examine the more general mixed Nash case.

3.4.1 Pure Nash Equilibria

A pure Nash equilibrium is a very natural solution concept, since it would guarantee that everyone broadcasting is doing so successfully while no one not broadcasting could succeed even if they went at maximum power. Unfortunately a simple example shows that pure Nash equilibria do not always exist in our game.

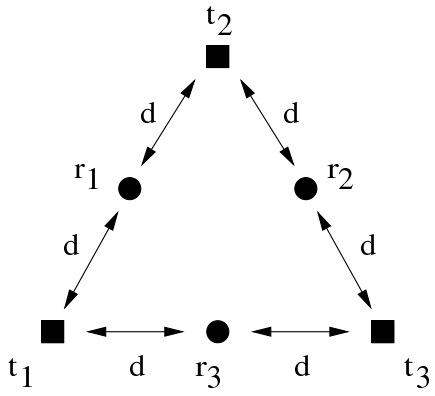

Figure 3.6: No pure Nash exists

The bad example is as follows, and is given in Figure 3.6. There are three transmitters t_1, t_2, t_3 in an equilateral triangle with side length $2d$ for some arbitrary $d \gg d_0$. The receiver for t_1 (i.e. r_1) is located halfway between t_1 and t_2 (so at distance d from each). Similarly, r_2 is located halfway between t_2 and t_3 and r_3 is located halfway between t_3 and t_1. We will set $\tau = 2$ and $\alpha = 2.5$, and will assume no background noise so $W = 0$.

We first claim that $|OPT| = 1$. To see this, suppose that there are at least two successful broadcasts. Without loss of generality we will assume that connections 1 and 2 are successful, and are broadcasting at powers p_1 and p_2 respectively. Then since connection 1 is successful we know that $(p_1/d^\alpha)/(p_2/d^\alpha) \geq 2$, and thus that $p_1 \geq 2p_2$. Now we note by simple geometry that the distance from t_1 to r_2 is exactly $d\sqrt{3}$, so the SINR of connection 2 at r_2 is $(p_2/d^\alpha)/(p_1/(d\sqrt{3})^\alpha) = (p_2 3^{1.25})/p_1 \leq (p_2 3^{1.25})/(2p_2) = 3^{1.25}/2 < 2 = \tau$, which is a contradiction since we assumed that connection 2 was successful.

Now since $|OPT| = 1$, any pure Nash equilibrium must have exactly one successful transmission (since obviously it must have more than 0 and at most $|OPT|$). Without loss of generality we will assume that connection 1 is successful with power p_1, so by the definition of a pure Nash it must be the case that neither connection 2 nor 3 are broadcasting with power greater than 0. But then the interference at r_2 is just $p_1/(d\sqrt{3})^\alpha$, so if t_2 broadcasted at power p_1 then the SINR at r_2 would be $(p_1/d^\alpha)/(p_1/(d\sqrt{3})^\alpha) = (\sqrt{3})^\alpha > 3 > \tau$. So t_2 would be successful if it transmitted at power at least p_1, and thus t_1 broadcasting by itself is not a pure Nash equilibrium.

While pure Nash equilibria do not always exist, when they do exist they have value close to OPT. This is formalized in the next subsection when we prove the same statement about mixed Nash equilibria, but we will provide the intuition for the pure Nash special case. Fix some pure Nash. We will try to find a receiver whose associated transmitter is not broadcasting in the Nash but has "small" interference, where our notion of small is something that increases as the value of the Nash gets closer to OPT. Since this receiver's transmitter is not broadcasting, the interference must be overcoming any possible signal and thus must actually be quite large, implying that the Nash must actually have value close to OPT.

3.4.2 Mixed Nash Equilibria

While pure Nash equilibria do not always exist, obviously a mixed Nash does (assuming a discretization of the action space). We now show that any mixed Nash (and thus any pure Nash, if one does exist) has value close to OPT. This result is actually implied by the no-regret result that we will discuss later (Theorem 3.15), but we feel it is useful to first consider the more intuitive case of mixed Nash. Recall that a mixed strategy is a probability distribution over the possible powers (i.e. over $[0, 1]$), and in a mixed Nash there is no incentive for any transmitter to change its distribution. For our purposes, for a transmitter t_i we will only need to consider the probability q_i that t_i broadcasts with non-zero power. We begin with a few useful lemmas. Fix some Nash equilibrium. For each transmitter t_i, let $p_{good}(i)$ be the probability (over the randomness in the strategies of the other transmitters) that t_i would be successful if it were to broadcast at power 1. Let $p_{bad}(i) = 1 - p_{good}(i)$ be the probability that t_i would not be successful. Note that if t_i has non-zero probability of broadcasting at some power greater than 0 but less than 1 then the probability of it succeeding at that power must be equal to the probability of it succeeding at power 1, since otherwise it could just switch to power 1 and strictly increase its expected payoff. So $S = \sum_i q_i p_{good}(i)$ is the expected number of successful transmissions (i.e. the value of the equilibrium). Let $T = \sum_i q_i$ be the expected number of transmissions.

Lemma 3.10 *For any Nash equilibrium, for any transmitter t_i, if $q_i < 1$ then $p_{bad}(i) \geq 1/2$ and if $q_i > 0$ then $p_{bad}(i) \leq 1/2$*

Proof: Suppose that $q_i < 1$ and that $p_{bad}(i) < 1/2$, so $p_{good}(i) > 1/2$. Then by broadcasting at power 1 with probability q_i, the expected payoff to t_i would be $q_i(p_{good}(i) - (1 - p_{good}(i))) = q_i(2p_{good}(i) - 1)$. Since $2p_{good}(i) - 1 > 0$, this is maximized by setting $q_i = 1$, contradicting our choice of i and our assumption that this is an equilibrium.

Similarly, suppose that $q_i > 0$ and that $p_{bad}(i) > 1/2$. Then when t_i broadcasts it will fail more than $1/2$ the time, giving negative expected payoff, so t_i would just never broadcast (i.e. set q_i to 0), contradicting our choice of i. ■

Lemma 3.11 *For any Nash equilibrium, $S \leq T \leq 2S$*

Proof: The first inequality is obvious from the definitions, and the second immediately follows from the second part of Lemma 3.10, since $T = \sum_i q_i = 2\sum_i \frac{1}{2} q_i \leq 2\sum_i p_{good}(i) q_i = 2S$. ■

Let OPT be the set of receivers that achieve their SINR requirement in the optimal solution. We can now prove the main theorem of this section:

Theorem 3.12 *Any Nash equilibrium has an expected number of successful transmissions at least $\Omega(|OPT|/d_{\max}^{2\alpha})$, where we assume that α and τ are constants.*

Proof: Fix a Nash equilibrium. Let $L = \{i : q_i = 1\}$ be the set of connections with transmitters that broadcast at power greater than 0 with probability 1. Consider the following procedure (only for analysis, obviously). For each receiver x in $OPT \setminus L$ we will keep track of how much it is "bought" with a variable $b(x)$, initially all set to 0. Now we order all transmitters in the instance (or just all transmitters with non-zero q_i in the Nash) arbitrarily. We examine the transmitters one by one in this order. Say we are on transmitter t_i. Let $R(i)$ be the $\lfloor \frac{|OPT \setminus L| - k}{kT} \rfloor$ closest receivers in $OPT \setminus L$ to i (for some parameter k to be defined later) that are currently bought to less than 1, i.e. have $b(x) < 1$. We now increase their b values by q_i, so $b(x) := b(x) + q_i$.

Since each transmitter increases the sum of the b values by $q_i\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$, at the end of this process we know that $\sum_x b(x) = \sum_i q_i\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor \leq \frac{|OPT\setminus L|-k}{k} < \frac{|OPT\ L|}{k}$, since by definition $T = \sum_i q_i$. This means that there is some receiver $a \in OPT \setminus L$ that has $b(a) < 1/k$.

Let M' be the set of transmitters that contributed to $b(a)$ during the above process. Note that since $b(a) \leq 1/k$ we know that $\sum_{x\in M'} q_x \leq 1/k$; we will use this later. Let M be all other transmitters, and for every distance d let $z(d) = \sum_{x\in M:d(a,x)\leq d} q_x$ be the probability mass from M located inside $B(a,d)$. Consider some transmitter $x \in M$. Since $a \notin R(x)$ and $b(a) < 1$, any receiver $y \in R(x)$ must have $d(x,y) \leq d(x,a)$, or else a would be in $R(x)$. So by the triangle inequality we know that $d(a,y) \leq 2d(a,x)$, and thus that any transmitter x at distance at most d from a must have its entire $R(x)$ at distance at most $2d$ from a.

We will now give an upper bound for $z(d)$. Since every transmitter x in $M \cap B(a,d)$ contributes $q_x\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$ to the sum of the b values, and each receiver that it contributes to must be in $B(a,2d)$, the sum of the b values of receivers in $B(a,2d)$ is at least $z(d)\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$. Since a receiver's b value only increases if it is less than 1, and then only increases by at most 1, we know that the b value of any receiver is at most 2. Thus the number of receivers from OPT in $B(a,2d)$ is at least $\frac{z(d)}{2}\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$. By Lemma 3.6, this implies that $cd^2 \geq \frac{z(d)}{2}\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$ and thus that $z(d) \leq 2cd^2/\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$ for some constant c depending only on α, τ, and d_0.

Now that we have an upper bound on the probability mass inside a ball around a, we want to upper bound the probability mass in an annulus of thickness 1 around a. To do this, we note that the interference at a is maximized if every ball around a actually meets the above bound. Since in the end we will care about upper bounding the interference, we can say without loss of generality that every ball meets the above bound, implying that the sum of the probabilities of transmitters between distance d and $d+1$ is at most

$$\frac{2c}{\left\lfloor\frac{|OPT\setminus L|-k}{kT}\right\rfloor}((d+1)^2 - d^2) \leq \frac{6cd}{\left\lfloor\frac{|OPT\setminus L|-k}{kT}\right\rfloor}$$

when $d \geq 1$, and is at most $2c/\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$ when $d = 0$. Since the expected interference from a transmitter at distance d from a is at most its probability of broadcasting times $1/d^\alpha$, this means that the expected interference at a caused by transmitters at distance between d and $d+1$ from a is at most $(6c/\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor)\cdot\frac{1}{d^{\alpha-1}}$ for $d \geq 1$. For $d = 0$, since the interferenced caused by a transmitter is at most 1, the expected interference from transmitters between distances 0 and 1 from a is at most $2c/\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor$. Using linearity of expectations, we can sum over the annuli to get that the expected interference at a is at most

$$\frac{2c}{\left\lfloor\frac{|OPT\setminus L|-k}{kT}\right\rfloor} + \frac{6c}{\left\lfloor\frac{|OPT\setminus L|-k}{kT}\right\rfloor}\sum_{d=1}^{\infty}\frac{1}{d^{\alpha-1}} \leq \frac{8c\zeta(\alpha-1)}{\left\lfloor\frac{|OPT\setminus L|-k}{kT}\right\rfloor}$$

where $\zeta(\alpha-1)$ is the Riemann zeta function (which will be constant for $\alpha > 2$).

This gives us an upper bound on the expected interference at a caused by transmitters in M. What about the transmitters in M'? Since we know that $\sum_{x\in M'} q_x \leq \frac{1}{k}$, we get that they cause at most $\frac{1}{k}$ expected interference (which is what would happen if they were all at distance 1 from a). Thus the total expected interference is at most $(8c\zeta(\alpha-1)/\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor) + \frac{1}{k}$

So now we have an upper bound on the expected interference. Let us assume (for now) that $W = 0$. By using Markov's inequality, we get that the probability that a hears interference at least twice the expected interference is at most $1/2$. But since we know from how we selected a that the probability that its transmitter tries to transmit is less than 1, Lemma 3.10 implies that $p_{bad}(a) \geq 1/2$. Thus $(16c\zeta(\alpha-1)/\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor) + \frac{2}{k}$ must be enough interference to kill the transmission to a; in particular, it must be the case that $16c\zeta(\alpha-1)/\lfloor\frac{|OPT\setminus L|-k}{kT}\rfloor \geq \frac{1}{\tau d_{\max}^{\alpha}} - \frac{2}{k}$. We will now finally set k, to $4\tau d_{\max}^{\alpha}$, giving us that $16c\zeta(\alpha-1)/\lfloor\frac{|OPT\setminus L|-4\tau d_{\max}^{\alpha}}{4\tau d_{\max}^{\alpha} T}\rfloor \geq \frac{1}{2\tau d_{\max}^{\alpha}}$.

Solving for T in this equation, and assuming constant α and τ, implies that $T \geq \Omega(|OPT \setminus L|/d_{\max}^{2\alpha})$, and thus by Lemma 3.11 we have that $S \geq \Omega(|OPT \setminus L|/d_{\max}^{2\alpha})$. If $|OPT \setminus L| = o(|OPT|)$ then a superconstant fraction of transmitters are broadcasting with probability 1 in the Nash, which by Lemma 3.10 and Lemma 3.11 means that the expected number of successful transmissions in the Nash is at least $\Omega(|OPT|)$, which would prove the theorem. On the other hand, if $|OPT \setminus L| = \Omega(|OPT|)$ then the above equation implies that $S \geq \Omega(|OPT|/d_{\max}^{2\alpha})$, thus proving the theorem.

If $W \neq 0$ the theorem is still true but the details are slightly more complicated, so we give only a brief sketch. With background noise, instead of twice the expected interference being enough to kill the signal it must be that twice the expected interference plus the background noise must be enough to kill the signal. But this only causes us to lose another constant, since we assumed from the beginning that the distance from any receiver to its transmitter (and thus from a to its transmitter) is bounded away from the absolute limit by a constant. ■

3.5 Distributed Algorithms and No-Regret

We first need a few basic definitions about games. In this thesis the only games we will care about will be games with n players in which every player has exactly two possible actions. Let $\mathcal{A} = \{0,1\}^n$ be the space of possible strategy profiles for the game, i.e. given a point $A \in \mathcal{A}$, the ith coordinate a_i represents the action used by player i in profile A. Each player i will have a function $\alpha_i : \mathcal{A} \to \mathbb{R}$ that assigns a utility to each strategy profile. We will want to consider modifications of strategy profiles: given $A \in \mathcal{A}$, let $A \oplus a_i'$ be the strategy set obtained if player i changed its action from a_i to a_i'. We will use superscripts to denote time, so A^t will be the strategy profile at time t and a_i^t will be the action taken by player i at time t.

The following definition will play a central role in this section:

Definition 3.13 *The* regret *of player i at time T given strategy profiles $A^1, A^2, \ldots, A^T$ is*

$$\max_{a_i \in \{0,1\}} \frac{1}{T}\sum_{t=1}^{T} \alpha_i(A^t \oplus a_i) - \frac{1}{T}\sum_{t=1}^{T} \alpha_i(A^t)$$

Intuitively, having low regret means that you do almost as well as on average as the best single action would have done. This notion of regret has been studied extensively, especially in two different models: the *experts* model and the *bandit* model. The difference between the two models lies in the knowledge gained by a player after each round: in the bandit model a player only finds out the utility that it gained, while in the experts model players also find out the utility they would have gained if they had played the other action. Since the results that we care about

are similar in both models, we will choose the more general one and be in the bandit model. In the wireless setting, this means that if a transmitter chooses to transmit then it will find out whether or not it succeeded, but if it chooses not to transmit then it gains no information.

The *price of total anarchy* was introduced by Blum et al. [20] as a way of generalizing the *price of anarchy*. The price of anarchy of a game is the ratio of the value of the social optimum to the value of the worst Nash equilibrium. For example, in Section 3.4 we proved that the price of anarchy of the wireless game is $O(d_{\max}^{2\alpha})$. It is, as the name suggests, supposed to quantify the "price" that is being paid by allowing each player to be a separate rational agent rather than simply being controlled by a centralized authority. Unfortunately there are various problems with this definition, one of which is that, since finding a Nash equilibrium is PPAD-complete [33], it is not clear that rational agents will actually play a Nash equilibrium. In particular, if they always do then we could find a Nash equilibrium simply by letting rational agents play. Blum et al. [20] proposed weakening this rationality assumption by assuming only that the agents use strategies with regret tending to 0 as time goes to infinity (called *no-regret algorithms*). They chose this assumption because it generalizes the Nash assumption (playing a Nash equilibrium is a no-regret algorithm), and is plausible since such algorithms actually do exist (e.g. [11]) so rational players should do at least as well. They call the ratio of the optimum social welfare to the average social welfare obtained by players using no-regret algorithms the price of total anarchy. We will use this not as a tool for weakening rationality assumptions, but rather as a tool for designing distributed algorithms, since by definition we are guaranteed the existence of algorithms that achieve the price of total anarchy (any no-regret algorithm).

Some of the basic game theory underlying our results is the same in both the protocol model and the physical model. In particular, the basic game is the same. Each transmitter is a player, with two possible strategies: broadcast at power 0 (i.e. do not broadcast) or broadcast at power 1 (full power). Note that in the physical model we are competing with the optimum solution that can use any power between 0 and 1, but we will only be using powers 0 and 1. A transmitter has utility 1 if it broadcasts successfully, i.e. meets its SINR requirement in the physical model or has no neighbors broadcasting in the protocol model. It has utility -1 if it broadcasts unsuccessfully, and utility 0 if it does not broadcast at all. This is the same game that we considered in Section 3.4 except we are restricting the transmitters to two strategies, 0 and 1.

Let T be some time at which all transmitters have regret at most ϵ. Our goal is to prove that the average number of successful connections up to time T has been close to $|OPT|$. For each transmitter t_i, let q_i be the fraction of times at which t_i chose to transmit (i.e. played action 1), and let s_i be the fraction of times at which t_i transmitted successfully. Then $Q = \sum_i q_i$ is the average number of attempted transmissions and $S = \sum_i s_i$ is the average number of successful transmissions, so we are trying to prove that S is close to $|OPT|$. The following lemma shows that S can be bounded by Q, and thus will allow us to only look at attempted broadcasts rather than successful broadcasts:

Lemma 3.14 $S \leq Q \leq 2S + \epsilon n$

Proof: The first inequality is obvious from the definitions, since the average number of successful transmissions is clearly at most the average number of attempted transmissions. For the second inequality, it is sufficient to show that $s_i \geq \frac{1}{2}(q_i - \epsilon)$ for all transmitters t_i. Suppose that $s_i < \frac{1}{2}(q_i - \epsilon)$ for some i. Then t_i's average utility is $s_i - (q_i - s_i) = 2s_i - q_i < -\epsilon$. But t_i could

have had average utility of 0 by never broadcasting, which is a contradiction since i has regret at most ϵ. ■

3.5.1 Physical Model

We first consider the physical model. Our main theorem is that the price of total anarchy is small; in particular, we show that if all transmitters have low regret then the average number of successful transmissions is close to optimal:

Theorem 3.15 *Suppose that at time T every sender has regret at most ϵ. Then the average number of successful transmissions is $\Omega(|OPT|/d_{\max}^{2\alpha}) - \epsilon n$.*

Proof: The proof of Theorem 3.15 is extremely similar to the proof of Theorem 3.12. In fact, since in a Nash equilibrium every player has regret 0 (by the definition of Nash), Theorem 3.12 is actually a corollary of Theorem 3.15. The intuition behind the proof is that we can treat the fraction of time that a transmitter chose to broadcast using a no-regret algorithm in a similar way to how we treated the probability that a transmitter broadcasted in a mixed Nash. Of course, they are not identical since one is a statement about what has empirically happened in the past while the other is a probabilistic statement that holds for the past, present and future, but they are similar enough that the proof basically goes through. We include the modified proof for completeness.

Let $L = \{i : q_i \geq 1/2 - \epsilon\}$ be the set of connections with transmitters that broadcast at least $1/2 - \epsilon$ fraction of the time. Consider the following procedure. For each receiver x in $OPT \setminus L$ we will keep track of how much it is "bought" with a variable $b(x)$, initially set to 0. Now we order all transmitters in the instance (or just all transmitters with non-zero q_i) arbitrarily. We examine the transmitters one by one in this order. Say we are on transmitter t_i. Let

$$\Phi = \left\lfloor \frac{|OPT \setminus L| - k}{kQ} \right\rfloor$$

for some parameter k to be defined later, and let $R(i)$ be the Φ closest receivers in $OPT \setminus L$ to t_i that are currently bought to less than 1, i.e. have $b(x) < 1$. We now increase their b values by q_i, so $b(x) := b(x) + q_i$.

Since each transmitter i increases the sum of the b values by $q_i\Phi$, at the end of this process we know that

$$\sum_x b(x) = \sum_i q_i \Phi \leq \frac{|OPT \setminus L| - k}{k} < \frac{|OPT \setminus L|}{k}$$

since by definition $Q = \sum_i q_i$. This means that there is some receiver a that is in OPT but whose transmitter is not in L that has $b(a) < 1/k$.

Let M' be the set of transmitters that contributed to $b(a)$ during the above process. Note that since $b(a) \leq 1/k$ we know that $\sum_{x \in M'} q_x \leq 1/k$; we will use this later. Let M be all other transmitters, and for every distance d let $z(d) = \sum_{x \in M : d(a,x) \leq d} q_x$ be the average number of transmissions from transmitters in M located inside $B(a, d)$. Consider some transmitter $x \in M$. Since $a \notin R(x)$ and $b(a) < 1$, any receiver $y \in R(x)$ must have $d(x, y) \leq d(x, a)$, or else a would be in $R(x)$. So by the triangle inequality we know that $d(a, y) \leq 2d(a, x)$, and thus that

any transmitter x at distance at most d from a must have its entire $R(x)$ at distance at most $2d$ from a.

We will now bound $z(d)$. Since every transmitter x in $M \cap B(a,d)$ contributes $q_x\Phi$ to the sum of the b values, and each receiver that it contributes to must be in $B(a,2d)$, the sum of the b values of receivers in $B(a,2d)$ is at least $z(d)\Phi$. Since a receiver's b value only increases if it is less than 1, and then only increases by at most 1, we know that the b value of any receiver is at most 2. Thus the number of receivers from $OPT \setminus L$ in $B(a,2d)$ is at least $\frac{z(d)}{2}\Phi$. By Lemma 3.6, this implies that $cd^2 \geq \frac{z(d)}{2}\Phi$ and thus that

$$z(d) \leq \frac{2cd^2}{\Phi} \tag{3.1}$$

for some constant c depending only on α, τ, and d_0.

Now that we have a bound on the average number of transmissions inside a ball around a, we want to bound the average interference at a. To do this, we will first bound the average number of transmissions in an annulus of radius d_0, i.e. $z(d+d_0) - z(d)$. We first note that the interference at a is at most the interference caused if every ball around a actually meets the bound given by (3.1). This is easily proved: let d be the first ball that doesn't meet the bound of (3.1). If there are no transmitters at distance greater than d from a, then clearly the average interference could be increased by adding more transmitters to every annulus past d so that the bound of (3.1) is met. If there are transmitters at distance greater than d from a, then clearly the average interference would be increased by moving enough of them into $B(a,d)$ to meet the bound. Now we can just keep repeating this process until there are no more transmitters past the first distance that fails to meet (3.1), reducing to the first case.

This now implies that we can treat inequality (3.1) as a lower bound as well as an upper bound, and thus the average number of transmissions coming from senders between distance d and $d+d_0$ is at most

$$\frac{2c}{\Phi}((d+d_0)^2 - d^2) \leq \frac{6cd_0d}{\Phi}$$

when $d \geq d_0$, and is at most $2c/\Phi$ when $d = 0$. Since the interference from a transmitter at distance d from a is at most $\min\{1, (\frac{d_0}{d})^\alpha\}$, this means that the average interference at a caused by transmitters at distance between d and $d+d_0$ from a is at most $(6cd_0^{\alpha+1}/\Phi) \cdot \frac{1}{d^{\alpha-1}}$ for $d \geq d_0$. For $d = 0$, since the interference caused by a transmitter is at most 1 the average interference from transmitters between distances 0 and d_0 from a is at most $2c/\Phi$. Using linearity of expectations, we can sum over the annuli to get that the expected interference at a is at most

$$\begin{aligned}&\frac{2c}{\Phi} + \frac{6cd_0^{\alpha+1}}{\Phi}\sum_{i=1}^{\infty}\frac{1}{(id_0)^{\alpha-1}} \\ &= \frac{2c(1+3d_0^2\zeta(\alpha-1))}{\Phi} \leq \frac{8c\zeta(\alpha-1)}{\Phi}\end{aligned}$$

where $\zeta(\alpha-1)$ is the Riemann zeta function (which will be constant for $\alpha > 2$) and we are assuming $d_0 \leq 1$. If $d_0 > 1$ then we will simply have d_0^2 as a constant to carry through the rest of the calculations, which will not matter since we are not attempting to optimize constants anyway.

This gives us a bound on the average interference at a caused by transmitters in M. What about the transmitters in M'? Since we know that $\sum_{x \in M'} q_x \leq \frac{1}{k}$, it is obvious that they cause at most $\frac{1}{k}$ average interference (which is what would happen if they were all at distance d_0 or less from a). Thus the total expected interference is at most

$$\frac{8c\zeta(\alpha-1)}{\Phi} + \frac{1}{k}$$

So now we have a bound on the average interference. Let $p_{\text{bad}}(a)$ denote the fraction of times in which a's transmitter could not succeed in transmitting, whether it tried or not. If $p_{\text{bad}}(a) < \frac{1}{4}$, then sending at every time would give average utility greater than $\frac{3}{4} - \frac{1}{4} = \frac{1}{2}$. But when we chose a we made sure that its transmitter (call it t) was from $OPT \setminus L$, so we know that $q_t < \frac{1}{2} - \epsilon$. Thus t only tries to transmit less than $\frac{1}{2} - \epsilon$ the time, and hence it has an average utility of less than $\frac{1}{2} - \epsilon$. This is a contradiction since we are assuming that t has regret at most ϵ, and thus $p_{\text{bad}}(a) \geq \frac{1}{4}$.

So a's transmitter would fail at least $\frac{1}{4}$ of the time if it tried to send every time, and the average interference is at most $(8c\zeta(\alpha-1)/\Phi) + \frac{1}{k}$. By Markov's inequality we know that the fraction of times at which a hears interference at least four times the average interference is at most $\frac{1}{4}$, so the interference at a is at least $(32c\zeta(\alpha-1)/\Phi) + \frac{4}{k}$ at most $\frac{1}{4}$ of the time. So this amount of interference must be enough to make it impossible for a to successfully receive (i.e. the SINR constraint would be violated), or else $p_{\text{bad}}(a)$ would be less than $\frac{1}{4}$. Since a is at distance at most $d_{\max}$ from its transmitter, the strength of its signal is at least $\frac{d_0^\alpha}{d_{\max}^\alpha}$. Thus we get that $32c\zeta(\alpha-1)/\Phi \geq \frac{d_0^\alpha}{\tau d_{\max}^\alpha} - \frac{4}{k}$. We will now finally set k, to $8d_{\max}^\alpha/d_0^\alpha$, giving us that

$$\frac{32c\zeta(\alpha-1)}{\left\lfloor \frac{|OPT \setminus L| - 8\tau d_{\max}^\alpha}{8\tau d_{\max}^\alpha Q} \right\rfloor} \geq \frac{d_0^\alpha}{2\tau d_{\max}^\alpha}. \tag{3.2}$$

Solving for Q in this equation, and assuming constant α, τ, and d_0, gives us that $Q \geq \Omega(|OPT \setminus L|/d_{\max}^{2\alpha})$. To compare Q to $|OPT|$ instead of $|OPT \setminus L|$, we note that if $|OPT \setminus L| < \frac{1}{2}|OPT|$) then at least half of the transmitters in OPT are broadcasting at least $\frac{1}{2} - \epsilon$ of the time, and thus $Q \geq \frac{1}{2}|OPT|(\frac{1}{2} - \epsilon) = \Omega(|OPT|)$. On the other hand, if $|OPT \setminus L| \geq \frac{1}{2}|OPT|$ then we get that $Q \geq \Omega(|OPT|/d_{\max}^{2\alpha})$. Now we can simply apply Lemma 3.14 to prove that the average number of successful connections is at least $\Omega(|OPT|/d_{\max}^{2\alpha} - \epsilon n)$, thus proving the theorem. ■

Other Metrics

The physical model assumes that the fundamental underlying metric is the Euclidean plane. However, we only used this assumption in one place: the proof of Lemma 3.6, the main density lemma, which proved that the number of receivers from any feasible set of transmissions contained in a ball of radius d is at most $O(d^2)$. We then used this lemma to bound the average interference, which will work whenever the exponent in the density lemma is strictly less than the path-loss exponent α. So actually our proof will work in any metric in which the number of receivers from a feasible solution contained in a ball of radius d is $O(d^{\alpha-\epsilon})$ for some $\epsilon > 0$.

One example of this is true three-dimensional space with omnidirectional antennas. In this case, it makes sense to assume that $\alpha > 3$, since power is being dissipated in three dimensions

(so $\alpha \geq 3$) and some is probably being lost due to being absorbed by objects (or just the air). On the other hand, a sphere of radius d can clearly be covered with $O((d/d_0)^3)$ spheres of radius d_0, and thus we can immediately derive the appropriate density lemma from Lemma 3.4 (which still holds as stated).

But even making only the weakest standard assumption, that $\alpha > 2$, we can still handle extra metrics; for example, any metric with doubling dimension 2. The *doubling dimension* of a metric is defined to be the smallest number k for which for all distances $d \in \mathbb{R}_{\geq 0}$, any ball of radius d can be covered by 2^k balls of radius $d/2$. Suppose we have a metric with doubling dimension k. Then by recursive applications of the definition of doubling dimension we get that any ball of radius d can be covered by $2^{k \log(d/d_0)} = (d/d_0)^k$ balls of radius d_0, and thus we can once again apply Lemma 3.4 to get Lemma 3.6, and as long as $\alpha > k$ the rest of the proof will go through as before.

Finally, we consider one class of metrics for which a good density lemma does *not* hold, but for intuitively unrealistic reasons: the *wireless manifold*, introduced by Kanade and Vempala [55]. Intuitively, they define the class of wireless manifolds as the class of distorted two-dimensional grids. In particular, consider a $k \times k$ grid, with an arbitrary nonnegative length assigned to every grid edge. Now let the distance between two points be the length of the shortest path between them in this weighted graph. In their paper [55], Kanade and Vempala give heuristics for finding the best such manifold given signal strength data, and show that for existing data sets the best wireless manifold is significantly more accurate than the best embedding into the Euclidean plane. Thus it is natural to ask whether our techniques extend to these manifolds. Unfortunately they do not: the following theorem shows that the density lemma we require for our proof to work is not true.

Theorem 3.16 *For any $d > 0$ there is a wireless manifold and a set of $\Omega(d^\alpha)$ feasible transmissions on this manifold such that all receivers are in a ball of radius d.*

Proof: We first note that we can embed uniformly weighted complete graphs into a wireless manifold. To see this, suppose that we want to embed a complete graph with k vertices and weight d on every edge. We first create a square S with $\lceil k/4 \rceil$ vertices on each side, and set the length of all edges with both endpoints in S to 0. We can then take k edges with one endpoint in S and the other outside of S and set their lengths to $d/2$. All other edges will have weight M for some $M \gg d$. The resulting metric is clearly the required complete graph, where the non-S endpoints of the k edges we chose are the vertices.

Now that we can embed complete graphs, consider the same complete graph on k vertices with distance d between them. Put a transmitter/receiver pair on each vertex of the complete graph (so t_i and r_i are co-located). If all transmitters broadcast at power 1, the signal at r_i is 1 and the interference is $(k-1)\frac{1}{d^\alpha}$. So as long as $k \leq \frac{d^\alpha}{\tau} + 1$ the SINR at receiver r_i is at least τ, and thus the connection is supported. Thus in a ball of radius d we can support $\Omega(d^\alpha)$ connections. ■

Byzantine Transmitters

In many cases it is not realistic to assume that every single transmitter will be running a no-regret algorithm, so when designing a distributed algorithm we would like to be robust to some fraction

of the transmitters behaving in arbitrary ways. We manage to achieve this, but only if the number of transmitters that are Byzantine is a fraction of $|OPT|$, not if it depends on n.

We generalize the proof of Theorem 3.15 in a straightforward way. First, let B be the set of Byzantine transmitters, and let OPT be optimal relative to whatever the Byzantine transmitters do (but OPT is still a fixed set that would be feasible at every time point if not for B). Let Q be defined as before, but let $\bar{Q}$ be the part of the sum that comes only from transmitters not in B. Note that $Q \leq \bar{Q} + |B|$. Also note that Lemma 3.14 still holds, except with $\bar{Q}$ instead of Q and where S is only summed over transmitters not in B.

It is easy to see that the analysis of Theorem 3.15 holds as stated for Q, since the node a we find will be in OPT and thus non-Byzantine and the packing argument works as before since OPT is feasible on its own. So $Q = \Omega(|OPT|/d_{\max}^{2\alpha})$. If $|B| \leq (1-\delta)Q$ then we know that $\bar{Q} \geq \delta Q \geq \Omega(\delta|OPT|/d_{\max}^{2\alpha})$, and can apply Lemma 3.14 to finish the proof. In particular, by setting δ to $1/2$ we see that there is some constant k such that if at most $k|OPT|/d_{\max}^{2\alpha}$ transmitters are Byzantine then we still get a $O(d_{\max}^{2\alpha})$ approximation to $|OPT|$.

3.5.2 Protocol Model

As discussed, in the *protocol model* each connection is a node in an interference graph, and a transmission is successful if none of its neighbors are also transmitting. Clearly maximizing capacity is just the same problem as finding a maximum independent set in the interference graph. The classic theoretical model used for these graphs are unit disk graphs, but we will generalize to all locally growth bounded graphs. We show how to use the same basic technique as in the physical model, i.e. proving that the price of total anarchy is small for a particular game, to give a distributed algorithm that has good average performance. While we will not obtain either as good an approximation or as small a running time as [77], our algorithm is totally distributed in the sense that the only information each node gets is whether or not any of its neighbors tried to join the independent set in the last round.

We first show that in general graphs the average number of successful transmissions when every transmitter uses a no-regret algorithm can be arbitrarily far from the size of even the smallest *maximal* independent set. We then show that for growth bounded graphs, after a sufficient number of rounds, the average number of nodes that broadcasted successfully in a round is within a constant of the size of the maximum independent set.

General Graphs

Consider the following interference graph: there are two special nodes u and v that are adjacent. The node u is also adjacent to $(n-2)/2$ nodes $x_1, x_2, \ldots, x_{(n-2)/2}$, none of which are adjacent to v, and v is also adjacent to $(n-2)/2$ nodes $y_1, y_2, \ldots, y_{(n-2)/2}$, none of which are adjacent to u (see Figure 3.7). Clearly the smallest maximal independent set has size $\frac{n-2}{2} + 1 = n/2$, as it consists of either u with all the y's or v with all the x's. On the other hand, suppose that u and v each choose to broadcast independently with probability $1/2$ and all of the x_i's and y_i's never broadcast. Obviously the expected number of successful transmissions given these strategies is $1/2$, so it is only an $\Omega(n)$-approximation. We claim that in this case, every transmitter is using a no-regret algorithm.

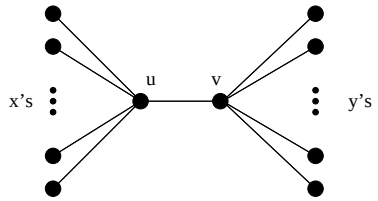

Figure 3.7: Bad interference graph

To see this, first consider some x_i. Since u broadcasts with probability $1/2$, the expected average utility of x_i if it chose to broadcast every time is 0, and obviously never broadcasting has average payoff 0. Thus never broadcasting is a no-regret algorithm. The same argument can be used for any y_i, with v taking the place of u. Now consider u. It too is adjacent to one node v that broadcasts with probability $1/2$, so both of its actions would give average utility 0. Clearly randomizing independently over these two actions also has an average utility of 0, and thus is a no-regret strategy. The same argument obviously works for v as well, finishing the proof.

Growth-Bounded Graphs

When we restrict to growth bounded graphs, no-regret algorithms actually do result in good behavior. This is essentially because the bad example we designed for general graphs can't occur, since in growth bounded graphs the number of neighbors of a node that are in an independent set is at most a constant. In other words, the definition of growth bounded graphs get us the equivalent of the density lemmas we used in the physical model (i.e. Lemmas 3.4 and 3.6).

Theorem 3.17 *In growth-bounded graphs in which at most k neighbors of any node are in an independent set, if all transmitters have regret at most ϵ then the average number of successful connections is at least $\Omega(|OPT|) - \epsilon n$.*

Proof: Let x_0 be some node, with neighbors $x_1, \ldots, x_m$. q, s, Q, and S are as defined in the physical model. We first claim that $\sum_{i=0}^{m} q_{x_i} \geq \frac{1}{3} - \frac{\epsilon}{2}$. The first case is if $q_{x_0} \geq 1/3$, in which case the claim is trivially true. So suppose that $q_{x_0} < 1/3$. Let p_{bad} be the fraction of times that x_0 would be unsuccessful if it chose to broadcast, i.e. the fraction of times at which at least one of its neighbors broadcasts. Then the average utility of always broadcasting is $1 - p_{\text{bad}} - p_{\text{bad}} = 1 - 2p_{\text{bad}}$. If $p_{\text{bad}} \leq 1/3 - \epsilon/2$, then the average utility of always broadcasting is at least $1/3 + \epsilon$. But this is a contradiction since x_0 has average utility at most $q_0 < 1/3$ but also has ϵ-regret. Thus we know that $p_{\text{bad}} > 1/3 - \epsilon/2$. But clearly $p_{\text{bad}} \leq \sum_{i=1}^{m} q_{x_i}$ by definition, since in order for a time to contribute to p_{bad} at least one of the x_i's need to be broadcasting. Hence we have proved that $\sum_{i=0}^{m} q_{x_i} \geq \frac{1}{3} - \frac{\epsilon}{2}$.

Now we relate $|OPT|$ to Q. For nodes $x \in OPT$, let $b(x) = \sum_{y \in N(x) \cup \{x\}} q_y$. By the above claim we know that $b(x) \geq \frac{1}{3} - \frac{\epsilon}{2}$, and thus $\sum_{x \in OPT} b(x) \geq |OPT|(\frac{1}{3} - \frac{\epsilon}{2})$. But every node is adjacent to at most k nodes from OPT by the growth-boundedness of the graph, and thus $\sum_{x \in OPT} b(x) \leq k \sum_u q_u = kQ$. Putting these together, we get that $Q \geq \frac{1}{k}(\frac{1}{3} - \frac{\epsilon}{2})|OPT|$. We

now apply Lemma 3.14 to get that

$$\begin{aligned} S &\geq \frac{1}{2}(Q - \epsilon n) \geq \frac{1}{2}\left(\left(\frac{1}{3k} - \frac{\epsilon}{2k}\right)|OPT| - \epsilon n\right) \\ &\geq \Omega(|OPT| - \epsilon n) \end{aligned}$$

as claimed. ■

3.6 Simulations

While the main contributions of this thesis are theoretical, we also performed simulations to show how no-regret algorithms do in practice. There are two aspects that we wanted to test: the quality of the algorithm (i.e. the average number of successful transmissions) and the speed at which the algorithm converges on that average. Our simulations will be in the vanilla physical model, where transmitters and receiver are points in the Euclidean plane. We will be using random topologies, in which n transmitter-receiver pairs are placed uniformly at random in a square of size of size 100×100 in the Euclidean plane, subject to each receiver being at distance at most $d_{\max}$ from its transmitter. Throughout these simulations we will set $\alpha = 2.1$ and $\tau = 0.5$, since it turns out that changing these parameters does not change the trends by very much.

For the simulations to test the quality of the algorithm we will compare our algorithm to the average performance achieved if every transmitter uses a best-response strategy instead of a no-regret strategy. In particular, we will compare what happens when every transmitter uses Best Response (the trivial algorithm in which each transmitter transmits if it would have succeeded last round and does not transmit if it would not have, i.e. each transmitter simply does the best thing relative to what happened last) to what happens when every transmitter uses the classic no-regret algorithm *Randomized Weighted Majority (WMR)* of Littlestone and Warmuch [67].

Our quality simulation shows the relationship between the number of nodes and the average number of successful transmissions per round after simulating for 100 rounds. We did this on 100 instances for each value of n and averaged the results. Figure 3.8 shows that as n gets larger our algorithm does better, while Best Response does about the same (or slightly worse). Note that large n is the only interesting regime, since only when n is large is there a lot of interference from other transmitters. This figure also shows that the approximation bound we proved, $O(d_{\max}^{2\alpha})$, is overly pessimistic, since for all three values of $d_{\max}$ that we tested the actual performance is significantly better than $n/d_{\max}^{2\alpha}$, and clearly n is an upper bound on $|OPT|$. For example, when $d_{\max} = 8$ and $n = 1000$, we observed an average of 138.861 successful transmissions, while $1000/8^{2.1}$ is only 12.69.

For the convergence speed simulations, instead of comparing to the Best Response algorithm we just test the average number of successful transmissions after various iterations. Our analysis requires $\Omega(n^2 \log n)$ iterations before the approximation guarantee can be made, but our simulations show that in practice this number of iterations is not necessary. As shown in Figure 3.9, substantially fewer than n iterations are required before the average is basically stable. The time to stability is not constant, as it does seem to grow with n, but it is certainly substantially smaller than the $\Omega(n^2 \log n)$ bound required by the analysis.

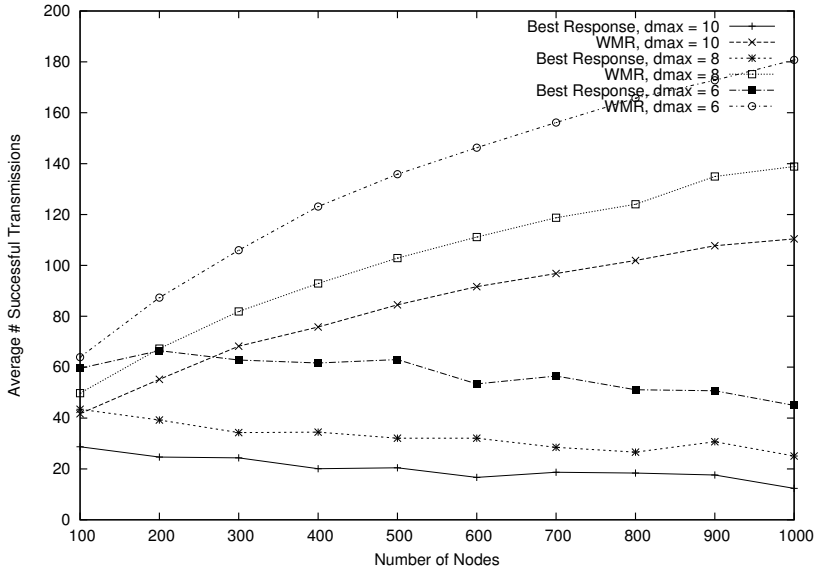

Figure 3.8: Random topology, 100 iterations

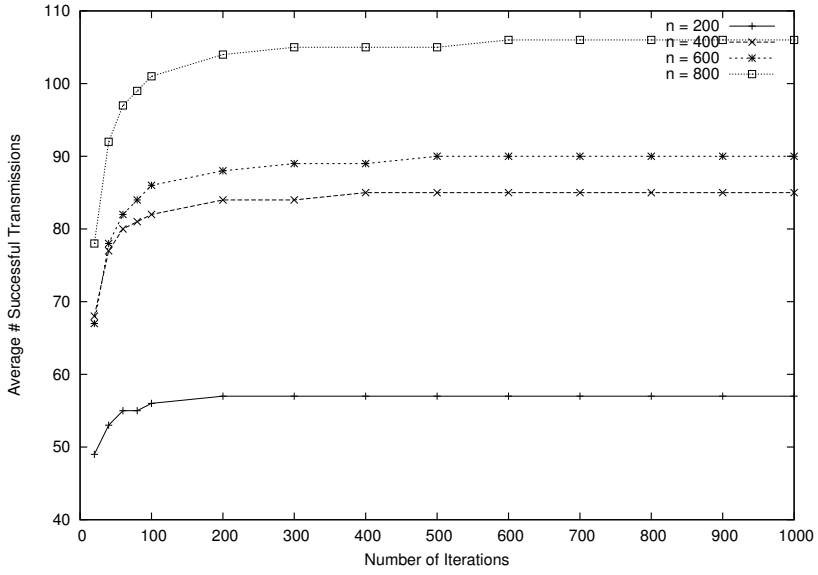

Figure 3.9: Random topology, $d_{max} = 10$

3.7 Conclusions

In this chapter we examined the complexity of maximizing the number of supported connections in the physical wireless model and we studied the performance that can be achieved by completely distributed algorithms operating under an appropriate incentive structure. In particular, we proved NP-hardness and gave two approximation algorithms for the centralized case. In the distributed setting our main techniques were game theoretic, namely proving that the price of total anarchy of an appropriately defined game is small. We hope that this technique will prove fruitful when considering other distributed problems, especially when only extremely limited feedback is allowed. We also showed by simulation that low-regret algorithms do even better in practice than the theoretical worst case, both in terms of their approximation to optimal and the time it takes to achieve this approximation.

Chapter 4

Constrained Connectivity and iBGP

In this chapter we will turn our attention back to wired networks and discuss the iBGP and Constrained Connectivity problems that we introduced in Section 1.3. We first review the basics of iBGP that we introduced there. In this setting there is some set of distances in the network, presumably defined by the IGP distances or OSPF weights or by some similar networking protocol. There is an initial set of routes F with corresponding egress routers X_F, where the egress router for some route is simply the router in the AS that initially heard about the route over eBGP from some other autonomous system. Given a set of routes, a router will rank highest the one whose egress router is closest according to this definition of distance. The *signaling graph* H is an overlay network whose nodes represent routers and where an edge represents the fact that the two routers at its endpoints use iBGP to inform one another of their current chosen route. The endpoints of an edge in H are called *iBGP neighbors*. A path in H is called a *signaling path*.

iBGP can be thought of as working as follows. In an asynchronous fashion, each router considers all the latest routes it has heard about from its iBGP neighbors, chooses the one with the closest egress router and tells its iBGP neighbors about the route it has chosen. This continues until no router learns of a route whose egress router is closer than that of its currently chosen route. When this process ends the route chosen by router r is denoted by $R(r)$. Let $P(r)$ be the shortest path from r to $E(r)$, the egress router of $R(r)$. When a packet arrives at r, it sends it to the next router r' on $P(r)$, r' in turn sends the packet to the next router on $P(r')$ and so on. Thus if $P(r')$ is not the subpath of $P(r)$ starting at r' then the packet will not get routed as r expected. An important property for a signaling graph to have is *complete visibility*, in which each router r hears about (and hence chooses as $R(r)$) the route in F whose egress router $E(r)$ is closest to r from amongst all routers in X_F. It is easy to see that if H has the complete visibility property for F then it will correctly implement the desired routing, so we say that H is *correct* if it has complete visibility for all possible external routes F, or equivalently for all possible subsets X_F of the routers that might announce external routes.

In Section 1.3 we claimed that the problems of minimizing the number of edges in a correct signaling graph (the IBGP-SUM problem) and the minimizing the maximum degree in a correct signaling graph (the IBGP-DEGREE problem) are special cases of a new network design problem that we call *Constrained Connectivity*. In Constrained Connectivity we are given a graph $G = (V, E)$ and for each pair of nodes $(u, v) \in V \times V$ we are given a set $S(u, v) \subseteq V$. Each such $S(u, v)$ is called a *safe set* and it is assumed that $u, v \in S(u, v)$. We say that a subgraph

$H = (V, F)$ of G is *safely connected* if for every pair of nodes (u, v) there is a path in H from u to v in which every node in the path is in $S(u, v)$. We are interested in two versions this problem:

1. CONSTRAINED CONNECTIVITY-SUM: compute a safely connected subgraph H with the minimum number of edges, and
2. CONSTRAINED CONNECTIVITY-DEGREE: compute a safely connected subgraph H that minimizes the maximum degree over all nodes.

We will begin this chapter by proving that the iBGP problems are in fact a special case of Constrained Connectivity, and then we will use this characterization to prove that they are hard to approximate to better than $\Omega(\log n)$. We will then generalize them to Constrained Connectivity on K_n, for which we demonstrate two approximation algorithms based on obvious LP relaxations. Generalizing further, we show that the Constrained Connectivity problems are as hard to approximate as the well-known Label Cover problem, and that the natural LP relaxations have at least a polynomial integrality gap. Finally, we will consider Constrained Connectivity in some simpler settings and show that in these settings it can actually be solved optimally in polynomial time.

4.0.1 Related Work

Issues involving eBGP, the version of BGP that routers in different ASes use to announce routes to one another, have recently received significant attention from the theoretical computer science community, especially stability and game-theoretic issues (e.g., [40, 48, 66]). However, not nearly as much work has been done on problems related to iBGP, which distributes routes internally in an AS. There has been some work on the problem of guaranteeing hot-potato routing in any AS that uses a *route reflector architecture* [18], which is the most commonly used type of iBGP signaling graph. These earlier papers did not consider the issue of finding small signaling graphs that achieved the hot-potato goal. Instead they either provided sufficient conditions for correctness relating the underlying physical network with the route reflector configuration [47] or they showed that by allowing some specific extra routes to be announced (rather than just the one chosen route) one could guarantee a version of hot-potato routing [15]. The first people to consider the problem of designing small iBGP overlays subject to achieving hot-potato correctness were Vutukuru et al. [86], who used graph partitioning schemes to give such configurations. But while they proved that their algorithm gave correct configurations, they only gave simulated evidence that the configurations it produced were small. Buob et al. [25] considered the problem of designing small correct solutions (along with other constraints that we do not concern ourselves with) but went in the opposite direction, giving a mathematical programming formulation but then simply solving the integer program using super-polynomial time algorithms. Xiao et al. [89] were also concerned with constructing small iBGP overlays, but instead of guaranteeing *correctness* they guarantee *reliability*.

4.1 iBGP Problems

4.1.1 iBGP and Constrained Connectivity

We want to show that the iBGP problems are a special case of the Constrained Connectivity problems. This involves defining a safe set for every pair of vertices, and proving that a signaling graph H is correct if and only if it is safely connected according to these safe sets. We will assume that there are no ties, i.e. all distances are distinct. For two routers x and y, let $D(x,y) = \{w : d(x,w) > d(x,y)\}$ be the set of routers that are farther from x than y is. Let $S(x,y) = \{w : d(w,y) < d(w,D(x,y))\} \cup \{y\}$ be the set of routers that are closer to y than to any router not in the ball around x of radius $d(x,y)$. We will refer to $S(x,y)$ as "safe" routers for the pair (x,y), and they form the safe sets for the Constrained Connectivity instance. A path between x and y in a signaling graph is said to be a *safe signaling path* for (x,y) if it is contained in $S(x,y)$.

Theorem 4.1 *An iBGP signaling graph H is correct if and only if for every pair $(x,y) \in V \times V$ there is a signaling path from y to x that uses only routers in $S(x,y)$.*

Proof: We first show that if every pair has a safe signaling path then every node hears about the route that has the closest egress router no matter what the set of egress routers X_F is. This is simple: let x be a router, and let $y \in X_F$ be its closest egress router. Let r be the route whose egress router is y. By assumption there is a signaling path from y to x that uses only routers in $S(x,y)$. By definition, every one of these routers is closer to y than to any router farther from x than y is. Since y is the closest egress to x, this means that for all of the routers in $S(x,y)$, y will be the closest egress router. A simple induction then shows that the routers in a safe signaling path will each choose r and hence tell their iBGP neighbor in the path about r. That is, x hears about r.

For the other direction we need to show that if a signaling graph is correct then every pair has a safe signaling path. For contradiction, suppose that there is no safe signaling path from y to x. Let X_F, the set of egress routers, be $D(x,y) \cup \{y\}$. Let r be the route whose egress router is y. Since every router in $D(x,y)$ is farther from x than y is, this means that for this set of egress routers x is closer to y than any other egress. By correctness we know that x does hear about y. Let $y = a_1, a_2, \ldots, a_k = x$ be the (or at least a) signaling path from y to x through which x hears about r. Since there are no safe signaling paths from y to x, we know that there exists some i such that $a_i \notin S(x,y)$. This means that there is some $w \in D(x,y)$ such that $d(a_i,w) < d(a_i,y)$. Since we assumed correctness we know that a_i heard about the route with the closest egress router z to a_i, and $z \neq y$ (since w in particular is closer). So a_i will not tell its iBGP neighbors about r, which is a contradiction since a_i is on the signaling path from which x heard about r. Thus a safe signaling path must exist. ∎

Note that this condition is easy to check in polynomial time, so we have shown membership in NP. This characterization also shows that the problems iBGP-SUM and iBGP-DEGREE are Constrained Connectivity problems where the underlying graph G is K_n the complete graph on $n = |V|$ nodes and the safe sets are defined by certain geometric properties.

4.1.2 iBGP Hardness

In this section we will show that the iBGP problems are $\Omega(\log n)$-hard to approximate by a reduction from HITTING SET (or equivalently from SET COVER). This is a much weaker hardness than the $2^{\log^{1-\epsilon} n}$ hardness that we will prove for the general Constrained Connectivity problems in Section 4.3, but the iBGP problems are much more restrictive. We begin by giving a useful gadget that encodes a HITTING SET instance as an instance of an iBGP problem in which all we care about is minimizing the degree of a particular vertex. We will then show how a simple combination of these gadgets can be used to prove that iBGP-DEGREE is hard to approximate, and how more complicated modifications to the gadget can be used to prove that iBGP-SUM is hard to approximate.

Suppose we are given an instance of hitting set with elements $1, 2, \ldots, n$ (note that we are overloading these as both integers and elements) and sets $T_1, T_2, \ldots, T_m$. Our gadget will contain a node x whose degree we want to minimize, a node a_i for all elements $i \in \{1, \ldots, n\}$, and a node b_{T_i} for each set T_i in the instance. We will also have four extra "dummy" nodes: z, y, u, and h. The following table specifies some of the distances between points. All other distances are the shortest path graph distances given these. Let M be some large value (e.g. 20), and let ϵ be some extremely small value larger than 0.

	x	z	y	a_i	b_{T_j}	u	h
x		M			$M + 1.4 + j\epsilon$		
z	M		1.5	$1 + i\epsilon$		2	
y		1.5					
a_i		$1 + i\epsilon$			$1 + (i + j)\epsilon$ (if $i \in T_j$)	1.1	
b_{T_j}	$M + 1.4 + j\epsilon$			$1 + (i + j)\epsilon$ (if $i \in T_j$)			$1 + j\epsilon$
u		2		1.1			
h					$1 + j\epsilon$		

It is easy to check that this is indeed a metric space. Informally, we want to claim that any solution to the iBGP problems on this instance must have an edge from x to a_i nodes such that the associated elements i form a hitting set. Here y, u, and h are nodes that force the safe sets into the form we want, and z is used to guarantee the existence of a small solution.

Lemma 4.2 *Let E be any feasible solution to the above iBGP instance. For every vertex b_{T_j} there is either an edge $\{x, b_{T_j}\} \in E$ or an edge $\{x, a_i\} \in E$ where $i \in T_j$.*

Proof: We will prove this by analyzing $S(x, b_{T_j})$. If we can show that $S(x, b_{T_j}) = \{x, b_{T_j}\} \cup \{a_i : i \in T_j\}$ then we will be finished. Note that $d(x, b_{T_j}) = M + 1.4 + j\epsilon$, so the vertices outside $B(x, d(x, b_{T_j}))$ are y (distance $M + 1.5$ from x), u (distance $M + 2$ from x), h (distance at least $M + 2.4$ from x), and b_{T_k} with $k > j$ (distance $M + 1.4 + k\epsilon$ from x). The vertices inside the ball are x, z, all a_i nodes, and b_{T_k} with $k \leq j$.

Obviously x and b_{T_j} are in $S(x, b_{T_j})$ by definition. Let a_i be a vertex with $i \in T_j$. It is easy to verify that a_i is closer to b_{T_j} than to any vertex outside of the ball: it has distance $1 + (i + j)\epsilon$ from b_{T_j}, distance $1 + (i + k)\epsilon$ from b_{T_k} with $k > j$, distance $2.5 + i\epsilon$ from y, distance 1.1 from u, and distance greater than 2 from h. So $a_i \in S(x, b_{T_j})$ as required. On the other hand, suppose $i \notin T_j$. Then $d(a_i, b_{T_j}) > 2$, while $d(a_i, u) = 1.1$, so $a_i \notin S(x, b_{T_j})$. Similarly, any vertex b_{T_k}

with $k < j$ is closer to h (distance $1 + j\epsilon$) than to b_{T_j} (distance at least 2) and z is closer to y (distance 1.5) than to b_{T_j} (distance at least 2). Thus $S(x, b_{T_j}) = \{x, b_{T_j}\} \cup \{a_i : i \in T_j\}$, so E must include an edge from x to either b_{T_j} or an a_i with $i \in T_j$. ■

We now want to use this gadget to prove logarithmic hardness for IBGP-SUM. We will use the basic gadget but will duplicate x. So there will be ℓ copies of x, which we will call $x_1, x_2, \ldots, x_\ell$, and their distances are defined to be $d(x_i, z) = M + i\epsilon$ and $d(x_i, b_{T_j}) = M + 1.4 + (i + j)\epsilon$ with all other distances defined to be the shortest path. Note that all we did was modify the gadget to "break ties" between the x_i's. Also note that the shortest path between x_i and x_j is through z, for a total distance of $2M + (i+j)\epsilon$. As before, let H be the smallest hitting set.

Lemma 4.3 *Any feasible* IBGP-SUM *solution has at least $\ell|H|$ edges.*

Proof: It is easy to see that Lemma 4.2 still holds, i.e. that $S(x_i, b_{T_j}) = \{x_i, b_{T_j}\} \cup \{a_k : k \in T_j\}$. Intuitively this is because all other x nodes are outside of $B(x_i, d(x_i b_{T_j}))$ and all distances from x to the gadget are the same as before except with an additional $i\epsilon$. This implies that the number of a_k and b_{T_j} nodes adjacent to x_i in any feasible solution must be at least $|H|$, since if there were fewer such adjacent nodes it would imply the existence of a smaller hitting set (any b_{T_j} nodes adjacent to x_i could just be covered using an arbitrary element in T_j at the same cost as using the set itself). Thus the total number of edges must be at least $\ell|H|$. ■

Lemma 4.4 *There is a feasible* IBGP-SUM *solution with at most $\ell|H| + \ell + (m+n+4)^2$ edges.*

Proof: The solution is simple: create a clique on the a_i, b_{T_j}, z, u, y, h nodes (which obviously has size at most $(m + n + 4)^2$), include an edge from every x_i to z (another ℓ edges) and include an edge from every x_i to every a_k with $k \in H$ (another $\ell|H|$ edges). Obviously there are the right number of edges in this solution, so it remains to prove that it is feasible. To show this we partition the pairs into types and show that every pair in every type is satisfied. The types are *1)* $x_i - b_{T_j}$, *2)* $x_i - h$, *3)* $x_i - x_j$, *4)* $x_i - \alpha$ (where α is any other node in the gadget not included in a previous type), and *5)* $\alpha - x_i$ This is clearly an exhaustive partitioning, so we can just demonstrate that each type is satisfied in turn.

For the first type we already showed that $S(x_i, b_{T_j})$ includes all a_k where $k \in T_j$. Since H is a valid hitting set x_i must be adjacent to one such a_k, which in turn is adjacent to b_{T_j}, forming a valid safe path. For the second type the only vertices outside $B(x_i, d(x_i, h))$ are x_j with $j \neq i$, and z is closer to h than to any such x_j. Thus $z \in S(x_i, h)$ so the path $x_i - z - h$ in our solution is a valid safe path. For the third type the vertices outside $B(x_i, d(x_i, x_j))$ are $\{x_k : k > j \text{ and } k \neq i\}$. Because of the tie-breaking we introduced, $d(z, x_j) = M + j\epsilon$ while $d(z, x_k) = M + k\epsilon > M + j\epsilon$, and thus $z \in S(x_i, x_j)$ and so the path $x_i - z - x_j$ in our solution is a valid safe path. The fourth type is even simpler, since α must be either z, u, y, or an a_k node and the shortest path from x_i to any of these is through z. So $z \in S(x_i, \alpha)$ and $x_i - z - \alpha$ is a valid safe path. Finally, for the last type the vertices outside $B(\alpha, d(\alpha, x_i))$ are $\{x_k : k > i\}$, and z is closer to x_i (distance $M + i\epsilon$) than any such x_k (distance $M + k\epsilon$). So again $z \in S(\alpha, x_i)$ and thus $\alpha - z - x_i$ is a valid safe path. ■

Theorem 4.5 *It is NP-hard to approximate* IBGP-SUM *to a factor better than $\Omega(\log N)$, where N is the number of vertices in the metric.*

Proof: It is known that there is some β for which it is NP-hard to distinguish hitting set instances with a hitting set of size at most β from instances in which all hitting sets have size at least $\beta \ln m$.

In the first case we know from Lemma 4.4 that there is a valid IBGP-SUM solution of size at most $\ell\beta + \ell + (m+n+4)^2$. In the second case we know from Lemma 4.3 that any valid IBGP-SUM solution must have size at least $\ell\beta \ln m$. If we set $\ell = (m+n+4)^2$ this gives a gap of $\ell\beta \ln m/\ell(\beta+2) = \beta \ln m/\beta + 2 = \Omega(\log m)$. The number of vertices N in the IBGP-SUM instance is $O((m+n+4)^2)$ so $\log m = \Omega(\log N)$, and thus we get $\Omega(\log n)$ hardness of approximation. ■

It is also fairly simple to modify the basic gadget to prove the same logarithmic hardness for IBGP-DEGREE. We do this by duplicating everything *other* than x, instead of duplicating x. This will force x to have the largest degree.

Theorem 4.6 *It is NP-hard to approximate* IBGP-DEGREE *to a factor better than* $\Omega(\log N)$*, where* N *is the number of vertices in the metric.*

Proof: We will use multiple copies of the above gadget. Let α be some large integer that we will define later. We create α copies of the gadget but identify all of the x vertices, so there is still a unique x but for all other nodes v in the original there are now α copies $v^1, v^2, \ldots, v^\alpha$. The distance between two nodes in the same copy is exactly as in the original gadget, and the distance between two nodes in different copies (say s^i and t^j) is the distance implied by forcing them to go through x (i.e. $d(s^i, t^j) = d(s,x) + d(x,t)$). Call this metric $M = (V, d)$. Every vertex in copy i is closer to the rest of copy i than to any vertex in copy j, so Lemma 4.2 holds for every copy. Thus if the smallest hitting set is H the degree of x in any feasible solution to IBGP-DEGREE on M must be at least $\alpha|H|$.

Conversely, we claim that there is a feasible solution to IBGP-DEGREE in which every vertex has degree at most $\alpha(|H|+1)$. Consider the solution in which x is adjacent to z^j and to a_i^j for all $j \in [\alpha]$ and $i \in H$, and all nodes (other than x) in copy j are adjacent to all other nodes (other than x) in copy j for all $j \in [\alpha]$. By the above analysis of $S(x, b^i_{T_j})$ we know that this solution satisfies these safe sets (via the safe path $x - a_i - b_{T_j}$ where $i \in H$ is an element in T_j). It also obviously satisfies pairs not involving x in the same copy, since there is an edge directly between them. It remains to show that pairs involving x are satisfied and that pairs involving two different copies are satisfied.

For the first of these we will show that z is in all safe sets of the form $S(x, w^i)$ where w is not a b node. This is easy to verify exhaustively. It is also true that z is in all safe sets of the form $S(w^i, x)$ even when w is a b node, since all vertices outside the ball $B(w^i, d(w^i, x))$ are in different copies and the shortest path from z to any node in a different copy must go through x. Thus the path $x - z - w^i$ in our solution satisfies both of these safe sets. Finally, it is again easy to verify that pairs in different copies are also satisfied.

Now by setting α appropriately we are finished. Each copy has $n+m+4$ nodes, so in the feasible solution we have constructed the degree of any node other than x is at most $(n+m+4)^2+1$. If we set α to some value larger than this, say $(n+m+4)^3$, we know that the degree of x has to be at least $(n+m+4)^3|H|$. It is known that it is hard to distinguish between hitting set instances with hitting sets of size at most β and those in which every hitting set has size at least $\beta \ln m$ for some value β. Suppose that we are in the first case, where there is a hitting set of size at most β. Then we constructed a feasible solution to the IBGP-DEGREE problem with maximum degree at most $(n+m+4)^3(\beta+1)$. In the second case, where every hitting set has size at least $\beta \ln m$, we showed that the degree of x (and thus the maximum degree) must

be at least $(n+m+4)^3\beta \ln n$. This gives a gap of $\beta \ln m/(\beta+1)$, which is clearly $\Omega(\log m)$. Since the number of vertices in the iBGP-DEGREE instance is polynomial in m, this implies $\Omega(\log N)$-hardness. ■

4.2 Constrained Connectivity on K_n

In this section we show that there is a $\tilde{O}(n^{2/3})$-approximation algorithm for CONSTRAINED CONNECTIVITY-SUM and CONSTRAINED CONNECTIVITY-DEGREE as long as the underlying graph is the complete graph K_n. This obviously implies the same approximation ratio for the iBGP problems, since they are special cases. Our algorithm has two components, an LP rounding algorithm and a random sampling step. So we first discuss a simple LP relaxation: the *flow LP*. This is actually a relaxation of the general Constrained Connectivity problems, but for the purposes of this section we will just assume that the underlying graph G is K_n.

For every pair $(u,v) \in V \times V$ let $\mathcal{P}_{uv}$ be the collection of $u-v$ paths that are contained in $S(u,v)$. The flow LP has a variable c_e for every edge $e \in E$ (called the *capacity* of edge e) and a variable $f(P)$ for every $u-v$ path in $\mathcal{P}_{uv}$ for every $(u,v) \in V \times V$ (called the *flow* assigned to path P). The flow LP simply requires that at least one unit of flow is sent between all pairs while obeying capacity constraints:

$$\begin{aligned}
&\min \textstyle\sum_e c_e \\
&\text{s.t} \textstyle\sum_{P \in \mathcal{P}_{uv}} f(P) \geq 1 && \forall (u,v) \in V \times V \\
&\textstyle\sum_{P \in \mathcal{P}_{uv}: e \in P} f(P) \leq c_e && \forall e \in E, (u,v) \in V \times V \\
&0 \leq c_e \leq 1 && \forall e \in E \\
&0 \leq f(P) \leq 1 && \forall (u,v) \in V \times V, P \in \mathcal{P}_{uv}
\end{aligned}$$

This is obviously a valid relaxation of CONSTRAINED CONNECTIVITY-SUM: given a valid solution to CONSTRAINED CONNECTIVITY-SUM, let P_{uv} denote the required safe $u-v$ path for every $(u,v) \in V \times V$. For every edge e in some P_{uv} set c_e to 1, and set $f(P_{uv})$ to 1 for every $(u,v) \in V \times V$. This is clearly a valid solution to the linear program with the exact same value. To change the LP for CONSTRAINED CONNECTIVITY-DEGREE we can just introduce a new variable λ, change the objective function to $\min \lambda$, and add the extra constraints $\sum_{v:\{u,v\} \in E} c_{\{u,v\}} \leq \lambda$ for all $u \in V$. And while this LP can be exponential in size (since there is a variable for every path), it is also easy to design a compact representation that has only $O(n^4)$ variables and constraints. This compact representation has variables $f_{(u,v)}^{(x,y)}$ instead of $f(P)$, where $f_{(u,v)}^{(x,y)}$ represents the amount of flow from u to v along edge $\{u,v\}$ for the demand (x,y). Then we can write the normal flow conservation and capacity constraints for every demand (x,y) independently, restricted to $S(x,y)$.

Our rounding will make use of the following simple lemma.

Lemma 4.7 *In any fractional solution to the flow LP, for every pair $(x,y) \in V \times V$ there is a path between x and y completely contained in $S(x,y)$ in which every edge is assigned capacity at least $1/|S(x,y)|^2$.*

Proof: Since the fractional solution is feasible, it sends one unit of flow from x to y using only edges that have both endpoints in $S(x,y)$. Suppose for contradiction that the lemma is false for some (x,y) pair. Then *every* safe path from x to y uses at least one edge with capacity less than $1/|S(x,y)|^2$. Let B be the set of these edges, i.e. $B = \{\{u,v\} : u,v \in S(x,y)$ and $c_{\{u,v\}} < 1/|S(x,y)|^2\}$. Since every safe $x-y$ path must go through at least one edge in B, the edges of B must form an $x-y$ cut in the graph induced on $S(x,y)$. However, $|B| < |S(x,y)|^2$, since every edge in B must have both endpoints in $S(x,y)$. This is a contradiction, since we cannot send one unit of flow across a cut with less than $|S(x,y)|^2$ edges in which every edge has capacity at most $1/|S(x,y)|^2$. ■

Another important part of our algorithm will be random sampling. We will use two different types of sampling: star sampling for the sum version and edge sampling for the degree version. First we consider star sampling, in which we independently sample nodes with probability p, and every sampled node becomes the center of a star that spans the vertex set.

Lemma 4.8 *All pairs with safe sets of size at least s will be satisfied by random star sampling with high probability if $p = 3\ln n/s$.*

Proof: Consider some pair (x,y) with $|S(x,y)| \geq s$. If some node (say z) from $S(x,y)$ is sampled then the pair is satisfied, since the creation of a star at z would create a path $x-z-y$ that would satisfy (x,y). The probability that no node from $S(x,y)$ is sampled is

$$(1-p)^{|S(x,y)|} \leq (1-p)^s \leq e^{-ps} = e^{-3\ln n} = 1/n^3$$

Since there are less than n^2 pairs, we can take a union bound over all pairs (x,y) with $|S(x,y)| \geq s$, giving us that all such pairs are satisfied with probability at least $1-1/n$. ■

For edge sampling, we essentially consider the Erdős-Rényi graph $G_{n,p}$, i.e. we just sample every edge independently with probability p. We will actually consider the union of $3\log n$ independent $G_{n,p}$ graphs, where $p = \frac{(1+\epsilon)\log s}{s}$ for some small $\epsilon > 0$. Let H be this random graph.

Lemma 4.9 *With probability at least $1-1/n$, all pairs with safe sets of size at least s will be connected by a safe path in H.*

Proof: Let (x,y) be a pair with $|S(x,y)| \geq s$. Obviously (x,y) is satisfied if the graph induced on $S(x,y)$ is connected. It is known [21] that there is some small ϵ with $0 < \epsilon < 1$ so that $G_{s,p}$ is connected with probability at least $1/2$. Since H is the union of $3\log n$ instantiations of $G_{n,p}$, we know that the probability that the subgraph of H induced on $S(x,y)$ is not connected is at most $1/n^3$. We can now take a union bound over all such (x,y) pairs, giving us that the probability that there is some unsatisfied (x,y) pairs with $|S(x,y)| \geq s$ is at most $1/n$. ■

We will now combine the random sampling and the threshold-based LP rounding into a single approximation algorithm. Our algorithm is divided into two phases: first, we solve the LP and round up any edge with capacity at least $1/n^{2/3}$. This takes care of safe sets of size at most $n^{1/3}$. Second, if the objective is to minimize the number of edges we do star sampling with probability $3\ln n/n^{1/3}$, and if the objective is to minimize the maximum degree we do edge sampling using the construction of Lemma 4.9 with $s = n^{1/3}$.

Theorem 4.10 *This algorithm is a $\tilde{O}(n^{2/3})$-approximation to both* CONSTRAINED CONNECTIVITY-SUM *and* CONSTRAINED CONNECTIVITY-DEGREE *on K_n.*

Proof: We first argue that the algorithm does indeed give a valid solution to the problem. Let (x, y) be an arbitrary pair. If $|S(x, y)| \leq n^{1/3}$, then Lemma 4.7 implies that the first phase of the algorithm results in a safe path. If $|S(x, y)| \geq n^{1/3}$, then Lemma 4.8 or Lemma 4.9 imply that the second phase of the algorithm results in a safe path. So every pair has a safe path, and thus the solution is valid.

We now show that the cost of this algorithm is at most $\tilde{O}(n^{2/3}) \times OPT$. We first consider the objective function of minimizing the number of edges. In phase 1 we only increase capacities by at most a factor of $n^{2/3}$, so since the LP is a relaxation of the problem we know that the total cost of phase 1 is at most $n^{2/3} \times OPT$. For phase 2, in expectation we chose $3n^{2/3} \ln n$ stars, for a total of at most $3n^{5/3} \ln n$ edges. But since there is a demand for every pair we know that $OPT \geq n - 1$, so phase 2 has total cost at most $\tilde{O}(n^{2/3}) \times OPT$.

If instead our objective function is to minimize the maximum degree, then since phase 1 only increases capacities by $n^{2/3}$ we know that after phase 1 the maximum degree is at most $n^{2/3} \times OPT$. In phase 2, a simple Chernoff bound implies that with high probability every node gets $\tilde{O}(n^{2/3})$ new edges, and thus the node with maximum degree still has degree at most $\tilde{O}(n^{2/3}) \times OPT$. ■

We also have a primal-dual algorithm that gives the same basic results as the threshold rounding for the CONSTRAINED CONNECTIVITY-SUM problem. While this algorithm and its analysis is slightly more complicated and only works for the Sum version, by not solving the linear program we get a faster algorithm. In particular, the best known algorithms for solving linear programs with Nn variables take $\Omega(Nn^{3.5})$ time on general LPs, so since there are $N = n^4$ variables in the compact version of the flow LP this takes $\Omega(n^{12.5})$ time. The primal-dual algorithm, on the other hand, is significantly faster: a naïve analysis shows that it takes $\tilde{O}(n^6)$ time.

In this algorithm we use the cut LP, which is a different relaxation than the flow LP. In fact, the algorithm is quite similar to the primal-dual algorithm for Steiner Forest, which uses a similar cut LP but doesn't have to deal with safe sets. Given a pair $(u, v) \in V \times V$, let $\mathcal{S}(u, v) = \{S \subset S(u, v) : u \in S \wedge v \notin S\}$ be the collection of safe set cuts that separate u and v. Furthermore, given a set $S \in \mathcal{S}(u, v)$ let $\delta_{uv}(S) = \{e \in \binom{V}{2} : e \in (S, S(u, v) \setminus S)\}$ be the set of safe edges that cross S. The cut LP has a variable x_e for every edge e, and is quite simple:

$$\begin{aligned} \min & \sum_e x_e \\ \text{s.t.} & \sum_{e \in \delta_{uv}(S)} x_e \geq 1 \qquad \forall u, v \in V, S \in \mathcal{S}(u, v) \\ & x_e \geq 0 \qquad\qquad \forall e \in \binom{V}{2} \end{aligned}$$

This LP simply minimizes the sum of the edge variables subject to the constraint that for every cut between two nodes there must be at least one safe edge crossing it. Since this is a primal-dual algorithm, instead of solving and rounding this LP we also consider the dual, which has a variable y_S^{uv} for every pair (u, v) and $S \in \mathcal{S}(u, v)$. We say that an edge $e \in S(u, v)$ if both

endpoints of e are in $S(u,v)$.

$$\begin{aligned}
\max \sum_{u,v \in V} \sum_{S \in \mathcal{S}(u,v)} & y_S^{uv} & \\
\text{s.t.} \sum_{u,v \in V : e \in S(u,v)} \sum_{S \in \mathcal{S}(u,v) : e \in \delta_{uv}(S)} & y_S^{uv} \leq 1 & \forall e \in \binom{V}{2} \\
& y_S^{uv} \geq 0 & \forall u, v \in V, S \in \mathcal{S}_{uv}
\end{aligned}$$

Unfortunately we will not be able to use a pure primal-dual approximation, but will have to trade off with a random sampling scheme as in the rounding algorithm. So instead of this primal, we will only have constraints for $u, v \in V$ with $|S(u,v)| \leq t$ for some parameter t that we will set later. Thus in the dual we will only have variables y_S^{uv} for (u,v) with $|S(u,v)| \leq t$. This clearly preserves the property that the primal is a valid relaxation of the actual problem. Let $D = \{(u,v) : |S(u,v)| \leq t\}$.

Our primal-dual algorithm, like most primal-dual algorithms, maintains a set of *active* dual variables that it increases until some dual constraint becomes tight. Once that happens we buy an edge (i.e. set some x_e to 1 in the primal), change the set of active dual variables, and repeat. We do this until we have a feasible primal.

Initially our primal solution H is empty and the active dual variables are $y_{\{u\}}^{uv}$ for every $(u,v) \in D$, i.e. every node u has an active dual variable for every other v that it has a demand with corresponding to the cut in $S(u,v)$ that is the singleton $\{u\}$. We raise these variables uniformly until some constraint (say the one for $e = \{w,z\}$) becomes tight. At this point we add e to our current primal solution H. We now change the active dual variables by "merging" moats that cross e. In particular, there are some active variables $\{y_S^{uv}\}$ where $e \in \delta_{uv}(S)$ (which implies that $w, z \in S(u,v)$ as well). Let $H|_{S(u,v)}$ denote the subgraph of H induced on $S(u,v)$. Without loss of generality we can assume that $w \in S$ and $z \notin S$. Let $T \subset S(u,v)$ be the connected component of $H|_{S(u,v)}$ containing z. We now make y_S^{uv} inactive, and make $y_{S \cup T}^{uv}$ active. We do this for all such active variables, and then repeat this process (incrementing all dual variables until some dual constraint becomes tight, adding that edge to H, and then merging moats that cross it) until all pairs $(u,v) \in D$ have a safe path in H.

Lemma 4.11 *This algorithm always maintains a feasible dual solution and an active set that does not contribute to any tight constraint.*

Proof: We will show this by induction, where the inductive hypothesis is that the dual solution is feasible and that no dual variables that contribute to a tight constraint are active. Initially all dual variable are 0, so it is obviously a feasible solution and no constraints are tight. Now suppose this is true after we add some edge e'. We need to show that it is also true after we add the next edge $e = \{w,z\}$. By induction the dual solution after we added e' is feasible and none of the active dual variables contribute to any tight constraints. Thus raising the active dual variables until some constraint becomes tight maintains dual feasibility.

To prove that no active variables contribute to a tight constraint, note that the only new tight constraint is the one corresponding to e. The only variables contributing to that constraint are of the form y_S^{uv} where $e \in \delta_{uv}(S)$. But our algorithm made all of these variables inactive, and only added new active variables for sets S' that contain both w and z and thus do not contribute to the

newly tight constraint. Furthermore, these sets S' are formed by the union of S and the connected component in $H|_{S(u,v)}$ containing the other endpoint, so no newly active variable contributes to a constraint that became tight previously (since they correspond to edges in H). ■

Theorem 4.12 *The primal-dual algorithm returns a graph H with at most $O(t^2) \times OPT$ edges in which every pair (u, v) with $|S(u, v)| \leq t$ has a safe path.*

Proof: After every iteration of the algorithm all of the tight constraints are added to H, which together with Lemma 4.11 implies that the algorithm never gets stuck. Thus it will run until every pair u, v with $|S(u, v)| \leq t$ has a safe path. It just remains to show that the total number of edges returned is at most $O(t^2) \times OPT$. To see this, note that every edge in H corresponds to a tight constraint in the feasible dual solution we constructed, so if $e \in H$ then $\sum_{u,v:e\in S(u,v)} \sum_{S\in\mathcal{S}(u,v):e\in\delta_{uv}(S)} y_S^{uv} = 1$. Thus we have that

$$\begin{aligned}
|H| &= \sum_{e\in H} 1 = \sum_{e\in H} \sum_{(u,v)\in D:e\in S(u,v)} \sum_{S\in\mathcal{S}(u,v):e\in\delta_{uv}(S)} y_S^{uv} \\
&= \sum_{(u,v)\in D} \sum_{S\in\mathcal{S}(u,v)} \sum_{e\in\delta_{uv}(S)\cap H} y_S^{uv} \\
&= \sum_{(u,v)\in D} \sum_{S\in\mathcal{S}(u,v)} |H \cap \delta_{uv}(S)| y_S^{uv} \\
&\leq t^2 \sum_{(u,v)\in D} \sum_{S\in\mathcal{S}(u,v)} y_S^{uv} \\
&\leq t^2 \times OPT
\end{aligned}$$

where the last inequality is by duality, and the next to last inequality is because $|H \cap \delta_{uv}(S)| \leq \binom{|\delta_{uv}(S)|}{2} \leq t^2$ (since $(u, v) \in D$). ■

Lemma 4.13 *The primal-dual algorithm takes at most $\tilde{O}(n^6)$ time.*

Proof: The primal-dual algorithm adds at least one new edge per iteration, so there can be at most n^2 iterations. In each iteration we have to figure out the current value of every dual constraint and the number of active variables in each constraint, which together will imply what the next tight constraint is and how much to raise the y variables. We then need to raise the active variables by that amount and merge moats. Note that for every demand there are at most two active moats, so the total number of active variables is at most $O(n^2)$. Thus each iteration can be done in time $O(n^4)$, where the dominant term is the time taken to calculate the value of each dual constraint. So the total time is $\tilde{O}(n^6)$, where there are extra poylogarithmic terms due to data structure overhead. ■

Now we can trade this off with the random sampling solution for large safe sets to get an actual approximation algorithm:

Theorem 4.14 There is a $\tilde{O}(n^{2/3})$ approximation algorithm for the CONSTRAINED CONNECTIVITY-SUM problem that runs in time $\tilde{O}(n^6)$

Proof: Omitted – basically like the proof of Theorem 4.10 except that instead of trading off the LP rounding and the random sampling we trade off the primal-dual algorithm and random sampling. ■

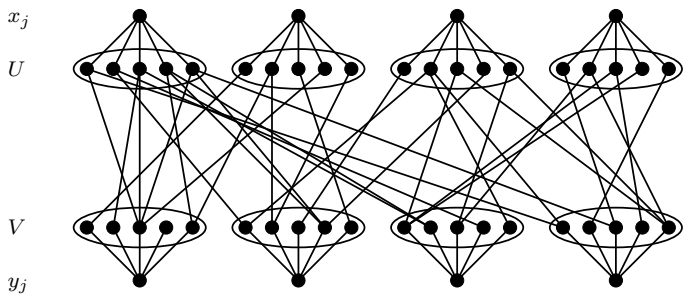

Figure 4.1: Basic hardness construction.

4.3 Constrained Connectivity

In this section we consider the hardness of the Constrained Connectivity problems and the integrality gaps of the natural LP relaxations.

4.3.1 Hardness

We now show that the CONSTRAINED CONNECTIVITY-SUM and CONSTRAINED CONNECTIVITY-DEGREE problems are both hard to approximate to better than $2^{\log^{1-\epsilon} n}$ for any constant $\epsilon > 0$. We do this via a reduction from MIN-REP, a problem that is known to be impossible to approximate to better than $2^{\log^{1-\epsilon} n}$ unless NP $\subseteq$ DTIME($n^{\text{polylog}(n)}$) [62]. An instance of MIN-REP is a bipartite graph $G = (U, V, E)$ in which U is partitioned into groups $U_1, U_2, \ldots, U_m$ and V is partitioned into groups $V_1, V_2, \ldots, V_m$. There is a *super-edge* between U_i and V_j if there is an edge $\{u, v\} \in E$ such that $u \in U_i$ and $v \in V_j$. The goal is to find a minimum set S of vertices such that for all super-edges $\{U_i, V_j\}$ there is some edge $\{u, v\} \in E$ with $u \in U_i$ and $v \in V_j$ and $u, v \in S$. Vertices from a group that are in S are called the *representatives* of the group. It is easy to prove by a reduction from LABEL COVER that MIN-REP is hard to approximate to better than $2^{\log^{1-\epsilon} n}$, and in particular it is hard to distinguish the case when $2m$ vertices are enough (one from each group for each side of the graph) from the case when $2m \times 2^{\log^{1-\epsilon} n}$ vertices are necessary [62].

Given an instance of MIN-REP, we want to convert it into an instance of CONSTRAINED CONNECTIVITY-SUM. We will create a graph with five types of vertices: x_j^i for $j \in [m]$ and $i \in [d]$; U; V; y_j^i for $j \in [m]$ and $i \in [d]$; and z. Here the x nodes represent d copies of the groups of U and the y nodes represent d copies of the groups of V, where d is some parameter that we will define later. z is a dummy node that we will use to connect pairs that are not crucial to the analysis. Given this vertex set, there will be four types of edges: $\{x_j^i, u\}$ for all $j \in [m]$ and $i \in [d]$ and $u \in U_j$; $\{u, v\}$ for all edges $\{u, v\}$ in the original MIN-REP instance; $\{v, y_j^i\}$ for all $j \in [m]$ and $i \in [d]$ and $v \in V_j$; and $\{w, z\}$ for all vertices w.

This construction is shown in Figure 4.1, except in the actual construction there are d copies of each node in the top and bottom layer and there is a z node that is adjacent to all other nodes. In Figure 4.1 the middle two layers are identical to the original MIN-REP problem, and the large ellipses represent the groups. In the figure we have simply added a new vertex for each group, and in the construction there are d such new vertices per group as well as a z vertex.

Now that we have described the constrained connectivity graph, we need to define the safe sets. There are two types of safe sets: if in the original instance there is a super-edge between U_i and V_j then $S(x_i^k, y_j^k) = S(y_j^k, x_i^k) = \{x_i^k, y_j^k\} \cup U_i \cup V_j$ for all $k \in [d]$. All other safe sets consist of the two endpoints and z. Let e_{MR} denote the number of super-edges in the MIN-REP instance, and let n_{MR} denote the number of vertices.

The intuition behind the following theorem and proof is that a safe path between an x node and a y node corresponds to using the intermediate nodes in the path as the representatives of the groups corresponding to the x and y nodes, so minimizing the number of labels is like minimizing the number of edges incident on x and y nodes.

Theorem 4.15 *The original* MIN-REP *instance has a solution of size at most K if and only if there is a solution to the reduced Constrained Connectivity problem of size at most $Kd + e_{MR} + 2md + n_{MR}$.*

Proof: We first prove the only if direction by showing that if there is a MIN-REP solution of size K then there is a Constrained Connectivity solution of size $Kd + e_{MR} + 2md + n_{MR}$. Let OPT_{MR} be the set of vertices in a Min-Rep solution of size K. Our constrained connectivity solution includes all edges of type 4, i.e. we include a star centered at z. For each $i \in [d]$ and $j \in [m]$ we also include all edges of the form $\{x_j^i, u\}$ where $u \in U_j \cap OPT_{MR}$ and all edges of the form $\{y_j^i, v\}$ where $v \in V_j \cap OPT_{MR}$. Finally, for each super-edge in the Min-Rep instance we include the edge between the pair from OPT_{MR} that satisfies it (if there is more than one such pair we choose one arbitrarily). The star clearly has $2md + n_{MR}$ edges, there are Kd edges from x and y nodes to nodes in OPT_{MR}, and there are clearly e_{MR} of the third type of edges, so the total number of edges in our solution is $Kd + e_{MR} + 2md + n_{MR}$ as required. To prove that it is a valid solution, we first note that for all pairs except those of the form (x_i^k, y_j^k) or (y_j^k, x_i^k) where $\{U_i, V_j\}$ is a super-edge are satisfied via the star centered at z. For pairs (x_i^k, y_j^k) and (y_j^k, x_i^k) with an associated super-edge, since OPT_{MR} is a valid solution there must be some $u \in U_i \cap OPT_{MR}$ and $v \in V_j \cap OPT_{MR}$ that have an edge between them, and the above solution would include that edge as well as the edge from x_i^k to u and from y_j^k to v, thus forming a safe path of length 3.

For the if direction we need to show that if there is a Constrained Connectivity solution of size $Kd + e_{MR} + 2md + n_{MR}$ then there is a Min-Rep solution of size at most K. Let OPT_{CC} be a constrained connectivity solution with $Kd + e_{MR} + 2md + n_{MR}$ edges. Since $S(w, z) = \{w, z\}$ for all vertices w, $2md + n_{MR}$ of those edges must be a star centered at z, so only $Kd + e_{MR}$ edges are between other vertices. Obviously there need to be at least e_{MR} edges between U and V, since otherwise it would be impossible to satisfy all of the demands between x and y nodes corresponding to super-edges. Thus there are at most Kd edges incident on either x or y nodes. We can partition these edges into d parts, where the edges in the ith part are those incident on an x^i or y^i node. So there must be one part of size at most K; let i be this part. But since this is a valid constrained connectivity solution there is a safe path between x_j^i and y_ℓ^i for all j, ℓ such that there is a super-edge between U_j and y_ℓ, and thus the nodes in U and V that are incident to edges in this ith part must form a valid Min-Rep solution of size at most K. ■

We can now set $d = n_{MR}^2$, which gives the following theorem

Theorem 4.16 CONSTRAINED CONNECTIVITY-SUM *cannot be approximated better than $2^{\log^{1-\epsilon} n}$ for any $\epsilon > 0$ unless NP $\subseteq$ DTIME$(n^{\text{polylog}(n)})$*

Proof: We know that it is hard to distinguish between an instance of MIN-REP with a solution of size at most $2m$ and an instance in which every solution is of size at least $2m \times 2^{\log^{1-\epsilon} n}$. Let $d = n_{MR}^2$. Then Theorem 4.15 implies that it is hard to distinguish between an instance of constrained connectivity with a solution of size at most $2mn_{MR}^2 + e_{MR} + 2mn_{MR}^2 + n_{MR} = O(mn_{MR}^2)$ and an instance in which every solution has size at least $2m2^{\log^{1-\epsilon} n_{MR}} n_{MR}^2 + e_{MR} + 2mn_{MR}^2 + n_{MR} = \Omega(mn_{MR}^2 2^{\log^{1-\epsilon} n_{MR}})$. This gives an inapproximability gap of $\Omega(2^{\log^{1-\epsilon} n_{MR}})$. Since $d = n_{MR}^2$ the number of vertices n in our constrained connectivity instances is $n_{MR} + 2mn_{MR}^2 \leq O(n_{MR}^2)$, and thus $\Omega(2^{\log^{1-\epsilon} n_{MR}}) = 2^{\Omega(\log^{1-\epsilon} n)}$. To get this to $2^{\log^{1-\epsilon} n}$ we can simply use a smaller ϵ'. ■

We will now prove that CONSTRAINED CONNECTIVITY-DEGREE has the same hardness of approximation of CONSTRAINED CONNECTIVITY-SUM. The reduction from MIN-REP to the degree problem is basically the same as the reduction to the sum problem, except there are also d^2 additional copies of the gadget other than the x and y nodes. More formally, now the nodes are x_j^i and y_j^i for $j \in [m]$ and $i \in [d]$, u^{ij} for $u \in U$ and $i, j \in [d]$, v^{ij} for $v \in V$ and $i, j \in [d]$, and z^{ij} for $i, j \in [d]$. Now intuitively each copy ij of the original U, V, and z is hooked together exactly like in the original construction, and is hooked up to the nodes $\{x_k^i\}_{k \in [m]}$ and $\{y_k^j\}_{k \in [m]}$ exactly as if they were one copy of the outer x and y nodes of the original construction.

More formally, the edges are the same as before, except now each of the d^2 new copies is independent. In other words, there is an edge between x_j^i and u^{ik} for all $i, k \in [d]$ and $j \in [m]$ and $u \in U_j$, an edge between y_j^i and v^{ki} for all $i, k \in [d]$ and $j \in [m]$ and $v \in V_j$, an edge between u^{ij} and v^{ij} for all $i, j \in [d]$ and edges $\{u, v\}$ in the original MIN-REP instance, an edge between x_j^i and z^{ik} for all $i, k \in [d]$ and $j \in [m]$, an edge between y_j^i and z^{ki} for all $i, k \in [d]$ and $j \in [m]$, an edge between u^{ik} and z^{ik} for all $i, k \in [d]$ and $u \in U$, and an edge between v^{ik} and z^{ik} for all $i, k \in [d]$ and $v \in V$. Similarly, the safe sets are as before but defined by the copies. That is, $S(x_i^k, y_j^\ell) = S(y_j^\ell, x_i^k) = \{x_i^k, y_j^\ell\} \cup U_i^k \cup V_j^\ell$. All safe sets between nodes in the same copy ij are the two endpoints together with z^{ij}, and the safe set of vertices in different copies is just all vertices.

Theorem 4.17 CONSTRAINED CONNECTIVITY-DEGREE *cannot be approximated better than* $2^{\log^{1-\epsilon} n}$ *for any constant* $\epsilon > 0$ *unless NP* $\subseteq$ *DTIME*$(n^{\text{polylog}(n)})$

Proof: Every vertex in U^{ij} or V^{ij} can have degree at most $n_{MR} + 1$, since there are only $n_{MR} - 1$ other nodes in its copy, and it can in addition be adjacent to z^{ij} and the node x_k^i or y_k^j corresponding to its group U_k and V_k respectively. Every node z^{ij} has degree at most $n_{MR} + 2m < 3n_{MR}$, since it can be adjacent to n_{MR} nodes in U^{ij} and V^{ij} as well as m nodes from X^i and m nodes from Y^j. On the other hand, every x_k^i node and every y_k^j node must be adjacent to at least 1 $U^{i\ell}$ or $V^{\ell j}$ node respectively for all d possibilities for ℓ. So every such x or y node has degree at least d, so if we set $d = 3n_{MR}$ we know that the node with maximum degree must be an x or a y node.

Recall that it is hard to distinguish MIN-REP instances with solutions of size at most $2m$ from those in which all solutions have size at least $2m2^{\log^{1-\epsilon} n_{MR}}$. Suppose that there is a solution of size $2m$, i.e. there is a solution with one representative from each group. Then there is a solution to the corresponding CONSTRAINED CONNECTIVITY-DEGREE instance with max degree at most d: every x_j^i and y_j^i is connected to its corresponding representative in each of the d copies corresponding to it as well as to the z node for that copy, and in each copy ij we include all edges between U^{ij} and V^{ij} and all edges between those nodes and z^{ij}. It is easy to see that this is a

valid solution: by the analysis of Theorem 4.15 we know that it is valid inside of each copy, and to get between copies nodes s^{ij} and $t^{k\ell}$ can use the safe path $s^{ij} - z^{ij} - x^i_h - z^{i\ell} - y^{k\ell}_h - z^{k\ell} - t^{k\ell}$, where s and t are arbitrary nodes in the copy ij, and h is an arbitrary index in $[m]$.

On the other hand, suppose that every solution to the MIN-REP instance has size at least $2m2^{\log^{1-\epsilon} n_{MR}}$. Then as in the analysis of Theorem 4.15 for every copy ij there must be at least $2m2^{\log^{1-\epsilon} n_{MR}}$ edges that are either between X^i and U^{ij} or between Y^j and V^{ij}. Thus there are at least $d^2 2m2^{\log^{1-\epsilon} n_{MR}}$ such edges. Since there are only $2md$ vertices in $X \cup Y$, at least one such vertex must have degree at least $2^{\log^{1-\epsilon} n_{MR}} d$.

This shows that it is hard to approximate CONSTRAINED CONNECTIVITY-DEGREE to better than $2^{\log^{1-\epsilon} n_{MR}} d/d = 2^{\log^{1-\epsilon} n_{MR}}$. Since the number of vertices n in our instance is polynomial in n_{MR}, this means that it is hard to approximate to better than $2^{\Omega(\log^{1-\epsilon} n)}$. We can then get this to $2^{\log^{1-\epsilon} n}$ by just using a smaller ϵ'. ■

4.3.2 Linear Programming Integrality Gap

We will now show that the integrality gap of the flow LP for CONSTRAINED CONNECTIVITY-SUM is large. The instances for which we will show a large integrality gap are derived from instances of the *Unique Games problem*, in which we are given a graph $G = (V, E)$ and a set of permutations π_{uv} on some alphabet Σ (one constraint for every edge $(u, v) \in E$) and are asked to assign a value x_u from Σ to each vertex u so as to satisfy the maximum number of constraints of the form $\pi_{uv}(x_u) = x_v$. This problem was first considered by Khot [56], who conjectured that it was NP-hard to distinguish instances on which $1 - \delta$ fraction of the constraints can be satisfied from instances on which at most ϵ fraction of the constraints can be satisfied (for sufficiently small ϵ and δ). For our purposes we will consider a minimization version of the Unique Games problem in which we can assign multiple labels to vertices and the goal is to assign as few labels as possible so that for every edge (u, v) there is some label x_u assigned to u with $\pi_{uv}(x_u)$ assigned to v. We first show that there exist instances that require many labels:

Lemma 4.18 *For any constant $\epsilon < 1$, there are instances of Unique Games with alphabet size $O\left(n^{\frac{2(1+\epsilon)}{1-3\epsilon}}\right)$ and $\Theta(n^2)$ edges that require ϵn^2 labels for any valid solution.*

Proof: We will prove this by the probabilistic method, i.e. we will analyze a *random* Unique Games instance with the given parameters and show that the probability that it has a solution of size at most $O(n^2)$ is strictly less than 1. This then implies the existence of such an instance. For our random instance, the underlying graph will be K_n, so there is a permutation constraint on every pair of vertices. Let $k = |\Sigma|$ be the size of the alphabet (we will later set this to the value claimed in the lemma, but for now we will leave it as a parameter). For each pair of vertices we will then select a permutation uniformly at random from S_k.

Now consider some fixed set S of αn labels (so the average number of labels per node is α). What is the probability that S is a valid solution? By Markov's inequality, we know that at most $n/2$ vertices have more than 2α labels, so there are at least $n/2$ vertices with at most 2α labels. Call these vertices *light*, and call an edge *light* if both of its endpoints are light. Let $\{u, v\}$ be a light edge. We claim that the probability that S satisfies $\{u, v\}$ is at most $\frac{4\alpha^2}{k}$. To see this, let $\ell \in \Sigma$ be one of the labels assigned to u by S. Since the permutation for $\{u, v\}$ was chosen uniformly at random, the probability that ℓ is matched to one of the labels assigned to v by S is at

most $2\alpha/k$. Now we can do a union bound over all such labels ℓ, of which there are at most 2α, to get that the probability that edge $\{u, v\}$ is satisfied by S is at most $\frac{4\alpha^2}{k}$. Since the permutations for each edge are chosen independently, the event that edge e is satisfied is independent of the event that edge e' is satisfied for all $e' \neq e$. Thus the probability that S satisfies *every* edge is at most the product of the probabilities that it satisfies each fixed edge, i.e. the probability that S is a valid solution is at most $\left(\frac{4\alpha^2}{k}\right)^{\binom{n}{2}} < \left(\frac{4\alpha^2}{k}\right)^{\frac{1-\epsilon}{2}n^2}$ (for sufficiently large n).

By the trivial union bound, we know that the probability that there is *some* valid solution of size αn for our random instance is at most the sum over all possible solutions of size αn of the probability that the solution is valid, which by the above analysis we know is at most $|\{S : |S| = \alpha n\}| \times \left(\frac{4\alpha^2}{k}\right)^{\binom{n}{2}}$. So we will now bound $N = |\{S : |S| = \alpha n\}|$, which is easy to do by a simple counting argument. In particular, it is obvious that $N = \binom{kn}{\alpha n}$, since there are exactly kn total labels and we are just choosing αn of them. Now standard bounds for binomial coefficients imply that $N \leq \left(\frac{kne}{\alpha n}\right)^{\alpha n} = \left(\frac{ke}{\alpha}\right)^{\alpha n}$. Combining this with the previous analysis and setting $\alpha = \epsilon n$, we get that the probability that there is some valid solution of size αn is at most

$$\begin{aligned}
\left(\frac{ke}{\alpha}\right)^{\alpha n} \times \left(\frac{4\alpha^2}{k}\right)^{\frac{1-\epsilon}{2}n^2} &= \frac{4^{\frac{1-\epsilon}{2}n^2} e^{\epsilon n^2} \alpha^{n^2}}{k^{\frac{1-3\epsilon}{2}n^2}} \\
&= \frac{4^{\frac{1-\epsilon}{2}n^2} e^{\epsilon n^2} \epsilon^{n^2} n^{n^2}}{k^{\frac{1-3\epsilon}{2}n^2}} \\
&< \frac{n^{(1+\epsilon)n^2}}{k^{\frac{1-3\epsilon}{2}n^2}}
\end{aligned}$$

The final inequality is true as long as n is sufficiently large. If we set $k = n^{\frac{2(1+\epsilon)}{1-3\epsilon}}$ then this expression is less than 1. Since this is the probability that the random Unique Games instance we selected has a satisfying solution of size αn, this implies that for the given parameters there is *some* unique games instance that requires more than $\alpha n = \epsilon n^2$ labels. ■

Now that we have found a Unique Games instance that requires many labels we would like to use it to construct a CONSTRAINED CONNECTIVITY-SUM instance on which the flow LP has large integrality gap. We will basically use the same transformation that we used in the reduction of MIN-REP to CONSTRAINED CONNECTIVITY-SUM. Let V_{UG} be the vertex set of the above Unique Games instance, and let Σ be the alphabet. Then our CONSTRAINED CONNECTIVITY-SUM instance will have vertex set V equal to the disjoint union of $V_{UG} \times [d]$, $V_{UG} \times \Sigma$, and a special node z, where d is a duplication parameter that we will set later. For ease of notation, we will let x_i denote the i'th copy of vertex x in $V_{UG} \times [d]$, i.e. $x_i = (x, i)$. For all $x \in V_{UG}$ and $i \in [d]$ there is an edge from x_i to every vertex in $x \times \Sigma$. For every $x, y \in V_{UG}$ and $\alpha, \beta \in \Sigma$ there is an edge between (x, α) and (y, β) if and only if assigning α to x and β to y is sufficient to satisfy the $\{x, y\}$ edge in the Unique Games instance (i.e. the permutation for that edge matches them up). There is also an edge between every vertex and z. For $x, y \in V_{UG}$ and $i \in [d]$ we set $S(x_i, y_i) = S(y_i, x_i) = \{x, y\} \cup (x \times \Sigma) \cup (y \times \Sigma)$, and we set all other safe sets to the two endpoints and z.

Lemma 4.19 *The value of the flow LP on the above* CONSTRAINED CONNECTIVITY-SUM *instance is at least* $2d|V_{UG}| + |\Sigma||V_{UG}| + \binom{|V_{UG}|}{2}$.

Proof: We prove this by constructing an LP solution of the required size. We first set the capacity of every edge incident on z to 1, for a total cost of $|\Sigma||V_{UG}| + d|V_{UG}|$. This is enough capacity to satisfy all pairs other than those of the form (x_i, y_i) or (y_i, x_i), since for any other pair z is in the safe set so we can send one unit of flow on the edge from one endpoint to z and then one unit of flow on the edge from z to the other endpoint.

Now we set the capacity of every other edge to $1/|\Sigma|$. Since the number of other edges is $d|V_{UG}||\Sigma| + \binom{|V_{UG}|}{2}|\Sigma|$ this costs us $d|V_{UG}| + \binom{|V_{UG}|}{2}$ more, which when added to the cost of the edges to z gives us the claimed total LP value. So we just need to prove that this is enough capacity to satisfy demands between x_i and y_i for all $x, y \in V$ and $i \in [d]$. But this is easy to see: x_i can send $1/|\Sigma|$ flow to every node in $x \times \Sigma$ (for a total flow of 1), and each of these nodes will forward its incoming flow to its neighbor in $y \times \Sigma$. Since this is a Unique Games instance this neighbor will be unique, and each node in $y \times \Sigma$ will have exactly $1/|\Sigma|$ incoming flow, which it can then forward along its edge to y_i. Thus we have enough capacity to send one unit of flow from x_i to y_i. And y_i can send flow to x_i the same way, just in reverse. ■

Lemma 4.20 *Any integral solution to the above* CONSTRAINED CONNECTIVITY-SUM *instance must have size at most* $(d \times OPT_{UG}) + \binom{|V_{UG}|}{2} + d|V_{UG}| + |\Sigma||V_{UG}|$ *where* OPT_{UG} *is the minimum number of labels needed to satisfy the original Unique Games instance.*

Proof: The safe set of any node and z is only that node and z, so all edges incident to z need to be present in any integral solution for a cost of $d|V_{UG}| + |\Sigma||V_{UG}|$. Furthermore, for every pair $u, v \in V_{UG}$ at least one edge must be present from $(u \times \Sigma)$ to $(v \times \Sigma)$ since if no such edge existed there would be no way of connecting u_i and v_i through $S(u_i, v_i)$ for any $i \in [d]$. This adds $\binom{|V_{UG}|}{2}$ to the total cost, so now we just need to prove that there must be at least $dOPT_{UG}$ edges between $(V_{UG} \times [d])$ and $(V_{UG} \times \Sigma)$.

To show this, we will consider some arbitrary integral solution and partition the edges between $(V_{UG} \times [d])$ and $(V_{UG} \times \Sigma)$ into d parts where the ith part consists of those edges incident on nodes $\{x_i : x \in V_{UG}\}$. If every part has size at least OPT_{UG} then we are finished. To prove that this is indeed the case, we will prove that for every part, the endpoints that are in $V_{UG} \times \Sigma$ actually form a valid solution to the Unique Games instance. So consider the ith part of the partition. Suppose that the associated label assignment does not form a valid solution to the Unique Games instance. Then there is some pair $u, v \in V$ such that none of the labels assigned to u and none of the labels assigned to v are matched to each other in the permutation corresponding to edge $\{u, v\}$. But this clearly implies that there is no safe path from u_i to v_i, as any such path must be of length 3 and pass through a label for u and a label for v that are matched to each in the permutation corresponding to edge $\{u, v\}$. This is a contradiction since the integral solution must be a valid solution. ■

Now we can finally prove Theorem 4.21:

Theorem 4.21 *The flow LP for* CONSTRAINED CONNECTIVITY-SUM *has an integrality gap of* $\Omega(n^{\frac{1}{3}-\epsilon})$ *for any constant* $\epsilon > 0$.

Proof: We will use the Unique Games instance of Lemma 4.18 in the above reduction. Lemma 4.19 implies that the flow LP has value at most $O(d|V_{UG}| + |V_{UG}|^{\frac{3-\epsilon}{1-3\epsilon}})$ and Lemma 4.20 implies

that any integral solution has size at least $\Omega(d\epsilon|V_{UG}|^2) + |V_{UG}|^{\frac{3-\epsilon}{1-3\epsilon}})$. If we let $d = |\Sigma| = |V_{UG}|^{\frac{2(1+\epsilon)}{1-3\epsilon}}$ then this gives us an integrality gap of

$$\Omega\left(\frac{\epsilon|V_{UG}|^{\frac{4-4\epsilon}{1-3\epsilon}}}{|V_{UG}|^{\frac{3-\epsilon}{1-3\epsilon}}}\right) = \Omega\left(\epsilon|V_{UG}|\right).$$

It is easy to see that the number of nodes n in our reduction equals $d|V_{UG}| + |\Sigma||V_{UG}| + 1$ which in this case is $\Theta(|V_{UG}|^{\frac{3-\epsilon}{1-3\epsilon}})$. Thus the integrality gap is $\Omega(n^{\frac{1-3\epsilon}{3-\epsilon}})$, which is sufficient since we can set ϵ to be arbitrarily small. ■

We can modify this construction to show a polynomial integrality gap for the flow LP for CONSTRAINED CONNECTIVITY-DEGREE also. We will need Unique Games instances with the same parameters as in Lemma 4.18 but on the complete bipartite graph rather than the complete graph. It is easy to see that Lemma 4.18 can be modified to prove the existence of these instance. Now the modification is basically the same as the modification we made to show hardness: we just make d^2 copies of the inner Unique Games instance and connect them up to the d copies of the outer x_i and y_i nodes in the obvious way. This is the proof of Theorem 4.22.

Theorem 4.22 *The flow LP for* CONSTRAINED CONNECTIVITY-DEGREE *has an integrality gap of* $\Omega(n^{\frac{1}{9}-\epsilon})$ *for any constant* $\epsilon > 0$.

Proof: We only give a sketch of the proof here. The maximum degree of any node other than the outer d copies of the x and y nodes is at most $2|V_{UG}|^{\frac{3-\epsilon}{1-3\epsilon}}$, so if we set d equal to that value we know that the maximum degree must be achieved by some copy of an x_i or y_i. By splitting up the flow equally as in the proof of Lemma 4.19 we know that there is an LP solution in which the maximum degree is at most $d|\Sigma|/|\Sigma| + d = 2d$ (where the extra d factor is due to being adjacent to all associated z copies). On the other hand, we know that any valid integer solution must use at least $\epsilon|V_{UG}|^2$ edges incident on copies of x_i or y_i nodes for each of the d^2 instances. Thus there are at least $d^2\epsilon|V_{UG}|^2$ edges incident on these nodes in total, and since there are $d|V_{UG}|$ such nodes there must be at least one with degree at least $\epsilon d|V_{UG}|$. Thus the integrality gap is at least $\epsilon d|V_{UG}|/d = \epsilon|V_{UG}|$. The total number of nodes in our CONSTRAINED CONNECTIVITY-DEGREE instance is $O(|V_{UG}||\Sigma|d^2) = O(|V_{UG}|^{\frac{9-3\epsilon}{1-3\epsilon}})$, so this means the integrality gap is $\Omega(n^{\frac{1-3\epsilon}{9-3\epsilon}})$. By setting ϵ small enough this gives us the claimed gap of $\Omega(n^{\frac{1}{9}-\epsilon})$. ■

4.4 Hierarchical and Symmetric Safe Sets

In the hierarchical and symmetric safe set version of Constrained Connectivity $S(x, y) = S(y, x)$ for all $x, y \in V$ and if some node $z \in S(x, y)$ then $S(x, z) \subseteq S(x, y)$ and $S(z, y) \subseteq S(x, y)$. We show that a simple greedy algorithm solves this version optimally.

We say that a pair $\{x, y\}$ is an *easy* pair if there is some node $z \in S(x, y)$ such that $S(x, z) \subset S(x, y)$ and $S(y, z) \subset S(x, y)$. The pair $\{x, y\}$ is *hard* otherwise. Note that in a hard pair $\{x, y\}$, every node z in $S(x, y)$ has either $S(x, z) = S(x, y)$ or $S(y, x) = S(x, y)$ by the hierarchy property.

Lemma 4.23 *Let G be a graph that has a safe path for all hard pairs. Then all easy pairs also have a safe path in G, i.e. G is a feasible solution.*

Proof: We prove that every pair $\{x, y\}$ has a safe path in G by induction on the size of safe sets. For the base case, all pairs $\{x, y\}$ with $|S(x, y)| = 2$ are hard, so by assumption they have a safe path in G. For the inductive step, suppose that there are safe paths for all pairs $\{u, v\}$ with $|S(u, v)| < k$, and let $\{x, y\}$ be a pair with $|S(x, y)| = k$. If $\{x, y\}$ is hard then by assumption there is a safe path. If it is easy, then there is some node $z \in S(x, y)$ such that $S(x, z) \subset S(x, y)$ and $S(y, z) \subset S(x, y)$. Since these two subsets are strictly smaller, by induction there is an $x - z$ path contained in $S(x, z) \subset S(x, y)$ and there is a $z - y$ path contained in $S(y, z) \subset S(x, y)$. Concatenating these paths give an $x - y$ path contained in $S(x, y)$. ■

This lemma means that we don't have to worry about satisfying easy pairs, just hard ones. We now prove a few useful and easy lemmas.

Lemma 4.24 *Let $\{x, y\}$ be a hard pair. Then $S(u, v) \subseteq S(x, y)$ for all $u, v \in S(x, y)$.*

Proof: Since $\{x, y\}$ is hard either $S(u, x) = S(x, y)$ or $S(u, y) = S(x, y)$. Without loss of generality we assume that $S(u, x) = S(x, y)$. This implies that $v \in S(u, x)$, so by the hierarchy property we know that $S(u, v) \subseteq S(u, x) = S(x, y)$. ■

This clearly implies that if G is a feasible solution and $\{x, y\}$ is a hard pair then $G|_{S(x,y)}$ is connected and all pairs $u, v \in S(x, y)$ have a safe path contained in $S(x, y)$. We now prove some lemmas about the structure of the optimal solution.

Lemma 4.25 *Every edge $\{x, y\} \in OPT$ is a hard pair.*

Proof: Suppose $\{x, y\}$ is an edge in OPT that is an easy pair. Then there is some $z \in S(x, y)$ such that $S(x, z) \subset S(x, y)$ and $S(y, z) \subset S(x, y)$. Note that $y \notin S(x, z)$, since if it was then by the hierarchy property we would have that $S(x, y) \subseteq S(x, z)$, so $S(x, z) = S(x, y)$ contradicting $\{x, y\}$ being an easy pair. Similarly, we know that $x \notin S(y, z)$. Since OPT is feasible there is an $x - z$ path in $S(x, z) \subset S(x, y)$ and a $z - y$ path in $S(y, z) \subset S(x, y)$, and by the previous observation neither of them use the $\{x, y\}$ edge. So there is an $x - y$ safe path in OPT that does not use the $\{x, y\}$ edge. Any hard pair $\{u, v\}$ that uses the $\{x, y\}$ edge in a safe path can just use the path we found through z, since by Lemma 4.24 $S(x, y) \subseteq S(u, v)$. Thus if we remove $\{x, y\}$ all of the hard pairs still have a safe path, so by Lemma 4.23 so do all of the easy pairs. This contradicts OPT being optimal. ■

Order all hard pairs in nondecreasing order, breaking ties arbitrarily. We say $\{a, b\} \leq \{c, d\}$ if $\{a, b\}$ comes before $\{c, d\}$ in this ordering. We partition the edges of OPT as follows. Let $e = \{u, v\}$ be an edge in OPT, and let $\{x, y\}$ be the first hard pair in the ordering such that $u \in S(x, y)$ and $v \in S(x, y)$, and assign e to part $OPT_{\{x,y\}}$. By Lemma 4.25 all edges in OPT are hard pairs so this is a valid partition. Let $OPT_{\leq\{x,y\}} = \cup_{\{a,b\}\leq\{x,y\}} OPT_{\{a,b\}}$, and let $OPT_{<\{x,y\}}$ be defined analogously.

Lemma 4.26 *Let $\{x, y\}$ be a hard pair. Then $OPT_{\leq\{x,y\}}|_{S(x,y)}$ is connected.*

Proof: Let $\{u, v\}$ be an edge in $OPT|_{S(x,y)}$. Then since $\{u, v\}$ is a hard pair (by Lemma 4.25) and $\{x, y\}$ is a hard pair with both u and v in $S(x, y)$, by the definition of the partition the part $OPT_{\{a,b\}}$ containing $\{u, v\}$ must have $\{a, b\} \leq \{x, y\}$. Thus $\{u, v\} \in OPT_{\leq\{x,y\}}|_{S(x,y)}$. ■

We now finally give our algorithm. First we construct the above ordering. We then consider hard pairs in this order, and when considering a pair $\{x, y\}$ we add the minimum number of edges

required to make our current graph restricted to $S(x,y)$ connected. This algorithm clearly returns a feasible solution, since for any hard pair $\{x,y\}$ at some point we consider it and make sure that its safe set is connected and that is sufficient by Lemma 4.23. For every hard pair $\{x,y\}$, let $ALG_{\{x,y\}}$ by the edges added by the algorithm when considering $\{x,y\}$, and define $ALG_{<\{x,y\}} = \cup_{\{a,b\}<\{x,y\}} ALG_{\{a,b\}}$ and $ALG_{\leq\{x,y\}}$ analogously. Now we will prove that $|ALG| \leq |OPT|$.

Lemma 4.27 *The endpoints of any edge in $OPT_{<\{x,y\}}|_{S(x,y)}$ are connected in $ALG_{<\{x,y\}}|_{S(x,y)}$.*

Proof: Let $\{u,v\}$ be an edge in $OPT_{<\{x,y\}}|_{S(x,y)}$. Then $\{u,v\} \in OPT_{\{a,b\}}$ for some $\{a,b\} < \{x,y\}$. By definition, this means that $\{a,b\}$ is the first pair in the ordering with a safe set that contains both u and v. By Lemma 4.24 we know that $S(u,v) \subseteq S(a,b)$. We also know that $\{u,v\}$ is a hard pair by Lemma 4.25, so if $S(u,v) \subset S(a,b)$ then $\{u,v\}$ would be before $\{a,b\}$ in the ordering and would contain both u and v, contradicting the definition of $\{a,b\}$. Thus $S(u,v) = S(a,b)$. After considering $\{a,b\}$ the algorithm guarantees that $ALG_{\leq\{a,b\}}|_{S(a,b)}$ is connected, and therefore there is a safe $u-v$ path in ALG after considering $\{a,b\}$. We also know from Lemma 4.24 that $S(u,v) \subseteq S(x,y)$, so this safe path is entirely present in $ALG_{<\{x,y\}}|_{S(x,y)}$ and thus u and v are connected in $ALG_{<\{x,y\}}|_{S(x,y)}$. ■

Theorem 4.28 $|ALG| \leq |OPT|$

Proof: We will prove that $|ALG_{\{x,y\}}| \leq |OPT_{\{x,y\}}|$ for all hard pairs $\{x,y\}$. Since these form a partition of the edges of ALG and of OPT, this is sufficient to prove that $|ALG| \leq |OPT|$. Consider some such hard pair $\{x,y\}$. We know from Lemma 4.26 that $OPT_{\leq\{x,y\}}|_{S(x,y)}$ is connected, so $OPT_{\{x,y\}}$ must contain enough edges to connect the components of $OPT_{<\{x,y\}}|_{S(x,y)}$. By the definition of the algorithm, $ALG_{\{x,y\}}$ has the minimum number of edges necessary to connect the components of $ALG_{<\{x,y\}}|_{S(x,y)}$. Now since the number of components in $ALG_{<\{x,y\}}|_{S(x,y)}$ is at most the number of components of $OPT_{<\{x,y\}}|_{S(x,y)}$ (by Lemma 4.27), this implies that $|ALG_{\{x,y\}}| \leq |OPT_{\{x,y\}}|$. ■

Bibliography

[1] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In Rachid Guerraoui, editor, *DISC*, volume 3274 of *Lecture Notes in Computer Science*, pages 305–319. Springer, 2004. ISBN 3-540-23306-7. 2.5.2, 2.50, 2.5.2

[2] Ittai Abraham, Yair Bartal, Hubert T.-H. Chan, Kedar Dhamdhere, Anupam Gupta, Jon M. Kleinberg, Ofer Neiman, and Aleksandrs Slivkins. Metric embeddings with relaxed guarantees. In *Proceedings of the 46th Symposium on the Foundations of Computer Science (FOCS)*, pages 83–100, 2005. 2.1, 2.2, 2.4, 2.4.2

[3] Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. In *38th STOC*, 2006. 2.1, 2.2, 2.4.1, 2.4.1, 2.4.2

[4] Ittai Abraham, Cyril Gavoille, Andrew V. Goldberg, and Dahlia Malkhi. Routing in networks with low doubling dimension. In *In 26 th International Conference on Distributed Computing Syst ems (ICDCS). IEEE Computer*, page 75. Society Press, 2006. 1.1

[5] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: lower bounds. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on P arallelism in algorithms and architectures*, pages 207–216, New York, NY, USA, 2006. ACM Press. doi: http://doi.acm.org/10.1145/1148109.1148144. 2.5.2, 2.5.2, 2.5.2

[6] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: upper bounds. In *SPAA '06: Proceedings of the eighteenth annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 217–224, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-452-9. doi: http://doi.acm.org/10.1145/1148109.1148144. 1.1, 1.1, 2.5.2

[7] N. Alon and B. Schieber. Optimal preprocessing for answering on-line product queries. *Tech. report 71/87*, 1987. 2.3.2

[8] Ingo Althöfer, Gautam Das, David Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete Comput. Geom.*, 9(1):81–100, 1993. ISSN 0179-5376. 1.1, 2.1, 2.12

[9] Matthew Andrews and Michael Dinitz. Maximizing capacity in arbitrary wireless networks in the SINR model: complexity and game theory. In *INFOCOM*, 2009. 1.2, 3, 3.1

[10] Sunil Arya, Gautam Das, David M. Mount, Jeffrey S. Salowe, and Michiel H. M. Smid. Euclidean spanners: short, thin, and lanky. *STOC*, pages 489–498, 1995. 1.1

[11] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic

multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2003. ISSN 0097-5397. doi: http://dx.doi.org/10.1137/S0097539701398375. 3.5

[12] Baruch Awerbuch and David Peleg. Sparse partitions (extended abstract). In *31st Annual Symposium on Foundations of Computer Science, 22-24 October 1990, St. Louis, Missouri, USA*, pages 503–513, 1990. 2.5.1

[13] Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Discrete Math.*, 5(2):151–162, 1992. ISSN 0895-4801. 2.5.1

[14] Baruch Awerbuch, Shay Kutten, and David Peleg. On buffer-economical store-and-forward deadlock prevention. In *Proceedings of the IEEE INFOCOM*, pages 410–414, 1991. 1.1

[15] Anindya Basu, Chih-Hao Luke Ong, April Rasala, F. Bruce Shepherd, and Gordon Wilfong. Route oscillations in I-BGP with route reflection. In *Proc. ACM SIGCOMM*, pages 235–247, 2002. 1.3.1, 1.3.1, 4.0.1

[16] Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in $\tilde{o}(n^2)$ time. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 271–280, 2004. 1.1

[17] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α, β)-spanners and purely additive spanners. In *SODA*, pages 672–681, 2005. 1.1

[18] T. Bates, R. Chandra, and E. Chen. BGP route reflection - an alternative to full mesh IBGP. RFC 2796, 2000. 1.3.1, 4.0.1

[19] Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. In *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 932–941, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics. 2.6.1

[20] Avrim Blum, MohammadTaghi Hajiaghayi, Katrina Ligett, and Aaron Roth. Regret minimization and the price of total anarchy. In *STOC '08: Proceedings of the 40th annual ACM Symposium on Theory of Computing*, pages 373–382, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-047-0. doi: http://doi.acm.org/10.1145/1374376.1374430. 3, 3.5

[21] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001. 4.2

[22] Béla Bollobás, Don Coppersmith, and Michael Elkin. Sparse distance preservers and additive spanners (extended abstract). In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003)*, pages 414–423, New York, 2003. ACM. 1.1

[23] Jean Bourgain. On Lipschitz embeddings of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985. 2.1

[24] H. Breu. *Algorithmic Aspects of Constrained Unit Disk Graphs*. PhD thesis, University of British Columbia, 1996. 3, 3.2

[25] Marc-Olivier Buob, Steve Uhlig, and Mickael Meulle. Designing optimal iBGP route-reflection topologies. In *IFIP Networking*, Singapore, May 2008. 4.0.1

[26] Hubert T-H. Chan, Anupam Gupta, Bruce M. Maggs, and Shuheng Zhou. On hierarchical

routing in DOubling metrics. In *Proceedings of the16thAnnual ACM-SIAM Symposium on Discrete Algorithms*, pages 762–771, 2005. 1.1

[27] Hubert T.-H. Chan, Michael Dinitz, and Anupam Gupta. Spanners with slack. In *ESA*, pages 196–207, 2006. 1.1

[28] T-H. Hubert Chan and Anupam Gupta. Small hop-diameter sparse spanners for doubling metrics. In *Proceedings of the17thAnnual ACM-SIAM Symposium on Discrete Algorithms*, 2006. 2.3.2, 2.3.2

[29] Barun Chandra, Gautam Das, Giri Narasimhan, and José Soares. New sparseness results on graph spanners. *Internat. J. Comput. Geom. Appl.*, 5(1-2):125–144, 1995. ISSN 0218-1959. Eighth Annual ACM Symposium on Computational Geometry (Berlin, 1992). 1.1, 2.3.1

[30] S. Chaudhuri and C. D. Zaroliagis. Shortest paths in digraphs of small treewidth. I. Sequential algorithms. *Algorithmica*, 27(3-4):212–226, 2000. ISSN 0178-4617. Treewidth. 1.1

[31] Don Coppersmith and Michael Elkin. Sparse source-wise and pair-wise distance preservers. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 660–669, 2005. 2.3, 2.5.1

[32] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM*, 2004. 1.1

[33] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. In *STOC '06: Proceedings of the thirty-eighth annual ACM Symposium on Theory of Computing*, pages 71–78, New York, NY, USA, 2006. ACM. ISBN 1-59593-134-1. doi: http://doi.acm.org/10.1145/1132516.1132527. 3.5

[34] Michael Dinitz. Compact routing with slack. In *PODC '07: Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 81–88, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-616-5. doi: http://doi.acm.org/10.1145/1281100.1281114. 1.1

[35] Michael Dinitz. Distributed algorithms for approximating wireless network capacity. In *INFOCOM*, 2010. 1.2

[36] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759 (electronic), 2000. ISSN 0097-5397. 1.1

[37] Michael Elkin and David Peleg. Approximating k-spanner problems for k ¿ 2. *Theor. Comput. Sci.*, 337(1-3):249–277, 2005. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/j.tcs.2004.11.022. 2.6.1

[38] Michael Elkin and David Peleg. $(1+\epsilon, \beta)$-spanner constructions for general graphs. *SIAM J. Comput.*, 33(3):608–631 (electronic), 2004. ISSN 0097-5397. 1.1

[39] Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric graphs. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 671–679, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics. ISBN 0-89871-490-7. 3

[40] A. Fabrikant and C. Papadimitriou. The complexity of game dynamics: BGP oscillations, sink equilibria, and beyond. In *SODA '08: Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms*, pages 844–853, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. 4.0.1

[41] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth ACM symposium on Theory of computing*, pages 448–455. ACM Press, 2003. ISBN 1-58113-674-9. doi: http://doi.acm.org/10.1145/780542.780608. 2.1

[42] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *ICALP '01: Proceedings of the 28th International Colloquium on Auto mata, Languages, and Programming*, pages 757–772, London, UK, 2001. Springer-Verlag. ISBN 3-540-42287-0. 1.1

[43] Pierre Fraigniaud and Cyril Gavoille. A space lower bound for routing in trees. In *STACS '02: Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science*, pages 65–75, London, UK, 2002. Springer-Verlag. ISBN 3-540-43283-3. 1.1

[44] Cyril Gavoille, David Peleg, Stephane Perennes, and Ran Raz. Distance labeling in graphs. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 210–219, 2001. 1.1

[45] Olga Goussevskaia, Yvonne Anne Oswald, and Roger Wattenhofer. Complexity in geometric SINR. In *MobiHoc '07: Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 100–109, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-684-4. doi: http://doi.acm.org/10.1145/1288107.1288122. 3, 3.1

[46] Olga Goussevskaia, Magús Halldórsson, Roger Wattenhofer, and Emo Welzl. Capacity of arbitrary wireless networks. In *INFOCOM*, 2009. 3, 3.1

[47] T. G. Griffin and G. Wilfong. On the correctness of IBGP configuration. In *Proc. ACM SIGCOMM*, September 2002. 1.3.1, 4.0.1

[48] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Netw.*, 10(2):232–243, 2002. ISSN 1063-6692. 4.0.1

[49] Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.*, 31(5):1479–1500 (electronic), 2002. ISSN 0097-5397. 1.1

[50] Joachim Gudmundsson, Christos Levcopoulos, Giri Narasimhan, and Michiel H. M. Smid. Approximate distance oracles for geometric graphs. In *SODA*, pages 828–837, 2002. 1.1

[51] Anupam Gupta, Amit Kumar, and Rajeev Rastogi. Traveling with a Pez dispenser (or, routing issues in MPLS). *SIAM J. Comput.*, 34(2):453–474 (electronic), 2004/05. ISSN 0097-5397. 1.1

[52] Sariel Har-Peled and Manor Mendel. Fast constructions of nets in low dimensional metrics, and their applications. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 150–158, 2005. 1.1

[53] Harry B. Hunt III, Madhav V Marathe, Venkatesh Radhakrishnan, S. S. Ravi, Daniel J Rosenkrantz, and Richard E. Stearns. Nc-approximation schemes for np- and pspace-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998. ISSN 0196-6774. doi: http://dx.doi.org/10.1006/jagm.1997.0903. 3

[54] Piotr Indyk. Algorithmic aspects of geometric embeddings. In *Proceedings of the 42nd Symposium on the Foundations of Computer Science (FOCS)*, pages 10–33, 2001. 2.1

[55] Varun Kanade and Santosh Vempala. Life (and routing) on the wireless manifold. In *Sixth Workshop on Hot Topics in Networks (HotNets-VI)*, New York, NY, USA, 2007. ACM. 3.5.1

[56] Subhash Khot. On the power of unique 2-prover 1-round games. In *STOC '02: Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*, pages 767–775, New York, NY, USA, 2002. ACM. ISBN 1-58113-495-9. doi: http://doi.acm.org/10.1145/509907.510017. 4.3.2

[57] Samir Khuller, Balaji Raghavachari, and Neal E. Young. Balancing minimum spanning and shortest path trees. *Algorithmica*, 14(4):305–322, 1995. 2.3.1

[58] Jon M. Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *45th FOCS*, 2004. 2.1, 2.2, 2.4

[59] Goran Konjevod, Andréa Richa, and Donglin Xia. Optimal-stretch name-independent compact routing in doubling metrics. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 198–207, New York, NY, USA, 2006. ACM. ISBN 1-59593-384-0. doi: http://doi.acm.org/10.1145/1146381.1146412. 1.1

[60] Goran Konjevod, Andréa Richa, and Donglin Xia. Optimal scale-free compact routing schemes in networks of low doubling dimension. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 939–948, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5. 1.1

[61] Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001. ISSN 0178-4617. 2.6.2

[62] Guy Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30:2001, 1999. 4.3.1

[63] Guy Kortsarz and David Peleg. Generating low-degree 2-spanners. *SIAM J. Comput.*, 27 (5):1438–1456 (electronic), 1998. ISSN 1095-7111. 2.6.2

[64] Fabian Kuhn, Thomas Moscibroda, Tim Nieberg, and Roger Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In *DISC*, pages 273–287, 2005. 1.2, 3, 3.1

[65] Kofi A. Laing. Brief announcement: name-independent compact routing in trees. In *PODC '04: Proceedings of the twenty-third annual ACM Symposium on Principles of Distributed Computing*, pages 382–382, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-802-4. doi: http://doi.acm.org/10.1145/1011767.1011841. 1.1, 2.5.2

[66] H. Levin, M. Schapira, and A. Zohar. Internet routing and games. In *STOC '08: Proceed-*

ings of the 40th annual ACM Symposium on Theory of Computing, pages 57–66, New York, NY, USA, 2008. ACM. 4.0.1

[67] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Symposium on Foundations of Computer Science*, 0:256–261, 1989. doi: http://doi.ieeecomputersociety.org/10.1109/SFCS.1989.63487. 3.6

[68] Thomas Moscibroda and Roger Wattenhofer. The complexity of connectivity in wireless networks. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–13, April 2006. ISSN 0743-166X. doi: 10.1109/INFOCOM.2006.23. 3.1

[69] Thomas Moscibroda, Roger Wattenhofer, and Aaron Zollinger. Topology control meets SINR: the scheduling complexity of arbitrary topologies. In *MobiHoc '06: Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 310–321, New York, NY, USA, 2006. ACM. ISBN 1-59593-368-9. doi: http://doi.acm.org/10.1145/1132905.1132939. 3.1

[70] T. Nieberg, J. Hurink, and W. Kern. A robust PTAS for maximum weight independent sets in unit disk graphs. In *30th International Workshop on Graph-Theoretic Concepts in COmputer Science, WG 2004*, pages 214–221, 2004. 3

[71] David Peleg. *Distributed computing*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. ISBN 0-89871-464-8. A locality-sensitive approach. 1.1

[72] David Peleg. Proximity-preserving labeling schemes and thei applications. In *25th Workshop on Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science 1665, pages 30–41, 1999. 1.1

[73] David Peleg and Alejandro A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989. ISSN 0364-9024. 1.1, 2.12

[74] David Peleg and Jeffrey D. Ullman. An optimal synchronizer for the hypercube. *SIAM J. Comput.*, 18(4):740–747, 1989. ISSN 0097-5397. 1.1

[75] Venugopalan Ramasubramanian, Dahlia Malkhi, Fabian Kuhn, Mahesh Balakrishnan, Archit Gupta, and Aditya Akella. On the treeness of internet latency and bandwidth. In *SIGMETRICS '09: Proceedings of the eleventh international joint co nference on Measurement and modeling of computer systems*, pages 61–72, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-511-6. doi: http://doi.acm.org/10.1145/1555349.1555357. 1.1

[76] Theodore Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001. ISBN 0130422320. 3.1

[77] Johannes Schneider and Roger Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *PODC '08: Proceedings of the twenty-seventh ACM Symposium on Principles of Distributed Computing*, pages 35–44, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-989-0. doi: http://doi.acm.org/10.1145/1400751.1400758. 1.2, 3, 3.1, 3.5.2

[78] John W. Stewart. *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, 1999. 1.3

[79] Kunal Talwar. Bypassing the embedding: Algorithms for low-dimensional metrics. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC)*, pages 281–290, 2004. 1.1

[80] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of ACM 22*, pages 215–225, 1975. 2.30

[81] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024 (electronic), 2004. ISSN 0004-5411. 1.1

[82] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24 (electronic), 2005. ISSN 0004-5411. 1.1, 2.4.1, 2.4.1, 2.4.1, 2.4.2, 2.4.2, 2.5.1, 2.46, 2.5.1

[83] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *SPAA '01: Proceedings of the thirteenth annual ACM symposium on Pa rallel algorithms and architectures*, pages 1–10, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-409-6. doi: http://doi.acm.org/10.1145/378580.378581. 1.1, 1.1, 2.41, 2.44

[84] Mikkel Thorup and Uri Zwick. Spanners and emulators with sublinear distance errors. In *SODA*, pages 802–809, 2006. 1.1

[85] P. Traina, D. McPherson, and J. Scudder. Autonomous system confederations for BGP. RFC 3065, 2001. 1.3.1

[86] Mythili Vutukuru, Paul Valiant, Swastik Kopparty, and Hari Balakrishnan. How to Construct a Correct and Scalable iBGP Configuration. In *IEEE INFOCOM*, Barcelona, Spain, April 2006. 4.0.1

[87] Peter Winkler. Proof of the squashed cube conjecture. *Combinatorica*, 3(1):135–139, 1983. ISSN 0209-9683. 1.1

[88] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: a lightweight network location service without virtual coord inates. *SIGCOMM Comput. Commun. Rev.*, 35(4): 85–96, 2005. ISSN 0146-4833. doi: http://doi.acm.org/10.1145/1090191.1080103. 1.1

[89] Li Xiao, Jun Wang, and K. Nahrstedt. Optimizing IBGP route reflection network. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 3, pages 1765–1769 vol.3, May 2003. doi: 10.1109/ICC.2003.1203903. 4.0.1

[90] Andrew C. Yao. Space-time tradeoff for answering range queries. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 128–136, 1982. 2.3.2, 2.3.2

[91] Su Yi, Yong Pei, and Shivkumar Kalyanaraman. On the capacity improvement of ad hoc wireless networks using directional antennas. In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 108–116, New York, NY, USA, 2003. ACM. ISBN 1-58113-684-6. doi: http://doi.acm.org/10.1145/778415.778429. 1.2, 3