The Localized Delaunay Triangulation and Ad-Hoc Routing in Heterogeneous Environments

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Mark D. Watson

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Ad-Hoc Wireless routing has become an important area of research in the last few years due to the massive increase in wireless devices. Computational Geometry is relevant in attempts to build stable, low power routing schemes. It is only recently, however, that models have been expanded to consider devices with a non-uniform broadcast range, and few properties are known. In particular, we find, via both theoretical and experimental methods, extremal properties for the Localized Delaunay Triangulation over the Mutual Inclusion Graph. We also provide a distributed, sub-quadratic algorithm for the generation of the structure.

Acknowledgements

Thanks in particular goes to my supervisor Dr. J. Mark Keil whose unbelievable patience and insight was vital over the course of this project. Thanks goes to my friends and family who have lent support to me over the course of my degree. I would like to thank Dr. Kevin A. Schneider and the members and summer students of the University of Saskatchewan Software Research Lab. In particular: David Paquette, Jennifer Petrie, Nicole Staveness, and Andrew Sutherland. Their support and friendship, not to mention the use of their machines, proved to be invaluable. My friend and colleague Chris Worman has provided me with some good suggestions that have contributed to the development of my work and my approach to Computational Geometry. Thanks to the lab staff that aided me in running my experiments, and to my committee for helping improve my thesis. Special thanks to Pat Thomson and Chris Laramee for a variety of reasons. Last but not least, thanks to my dear friend Rebecca Lake who, in addition to her endless support and encouragement during the course of this degree, was also willing to help edit my proposal. Thank you all.

For those who created this field.

Contents

List of Tables

List of Figures

List of Abbreviations

DT	Delaunay Triangulation
GG	Gabriel Graph
GPS	Global Positioning System
IN	Intersection Graph
LDel	Localized Delaunay Triangulation
MIG	Mutual Inclusion Graph
OTG	One Two Graph
RNG	Relative Neighbourhood Graph
UN	Union Neighbourhood
UMIG	Unbounded Mutual Inclusion Graph

Chapter 1

Ad-Hoc Routing

1.1 Introduction

The last few years have seen an explosion in the quantity and availability of commercial wireless devices. Currently, mobile phones, PDAs, Laptops and other consumer electronics are fitted to be network capable. Using wireless receivers and transmitters, two devices can communicate if they are in each other's range. However, this does not address how two devices far away from each other can communicate. This issue requires wireless networking. In a wireless network each device can be used as a router in the process of carrying a signal between two locations. If these networks were consistent and well structured, they would not differ from traditional networks. However, wireless networks are potentially extremely inconsistent with new devices entering and exiting continually. As well, two devices could move apart and a pathway which had existed could be broken without changing the overall connectivity of the network. It is due to the rapid, continual change of the devices that we consider the networking to be "ad-hoc." As both of these examples show, a key focus of Ad-Hoc Wireless Network research is developing "good" routing schemes. See Figure

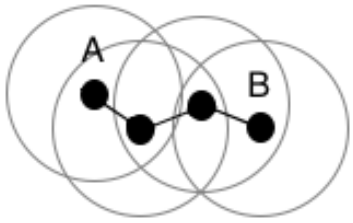

Figure 1.1: Wireless Networks: Here, the circles indicate the range of communication and the dots indicate the associated wireless device. The black lines indicate the paths nodes may use to send messages. A and B are outside of each other's range and cannot directly communicate. If the intermediate devices act as routers, they can still communicate.

1.1 for an example of an ad-hoc network.

Another possible scenario for Ad-Hoc Wireless networks does not require changing connections, but does rely on randomly situated devices. Consider a set of devices dropped by plane into a disaster area where local networks have been knocked out. These devices are to be stationary and are merely there to collect data and communicate with each other. Their exact position is potentially somewhat random, and some of the devices may be damaged on impact, thus a predetermined communication scheme is of little use. In both this example and the previous, we see that networks must be able to self-regulate.

The devices we consider are able to transmit over some range in an omnidirectional manner. This means that all devices within some range can receive the transmission. We only regard those devices which can both "see" (i.e. receive messages from) each other as being able to communicate. In ad-hoc networks, devices will often be turned on or off and movement is continual. Normally, a device can only

assume that another device is in range and still online when it can send transmissions back [25].

The devices are generally small and reasonably simple. They are limited in memory and processor speed hence all computations that the devices use in building routing tables must be simple to perform so that they are carried out in a reasonable amount of time. Secondly, because these tables must be updated frequently, the data required to build them should be able to be collected quickly. Finally, each device has very limited battery power and each transmission is expensive. The effects of this are twofold: first, we need to limit the number of transmissions as much as possible; second, we don't wish to overwork one node, as each transmission it sends drains its battery.

We also make a few assumptions about the devices. Each device is to be equipped with a GPS attachment that allows it to immediately know where on the earth it is. We also assume that each device has some way of approximating its range (at the very worst, it can use the furthest device it can currently reach as its operating range). Finally, every device is capable of broadcasting at a strength up to that of its range. This last point is important so that it can reduce power use.

To deal with these numerous problems, network topology control has been studied. There have been two broad geometric approaches to topology control. One is a hierarchical method that is closely tied to the area of clustering. The other approach, the one that is explored here, uses a distributed approach to build a flat topology, usually using classical geometric structures, like the Delaunay Triangulation [44].

1.2 Modeling Issues

Wireless communications are lossy and difficult to model. It is then necessary to decide what factors are most important. It is often the case that we are most concerned with the battery life of the devices. Since we are most concerned with the amount of power that is used in every broadcast, we need some formula or scheme to express how much energy is used per transmission.

To simplify the problem, we ignore all energy that is used by the device just by being turned on. Although it does drain a battery, this is outside of the scope of what we can hope to control using routing. Secondly, we ignore the energy expended when receiving and processing a signal[1].

Next we consider a phenomena known as *path loss*, the difference in strength between the signal when sent and when received. Path loss can be quantified as

$$P_R = O(\frac{P_T}{d^\beta}) \tag{1.1}$$

where P_T is the strength of the signal when initially sent and P_R is the power when the signal is received. The constants that are ignored in the big-Oh notation refer to specific properties of the antennas and are generally outside of the scope of research. The d is the distance the signal is sent and the β is a constant ranging between 2 and 5, which is used to model the decay of a signal in some environment[2]. The more

[1] Although we never expressly reference it, since we are trying to develop routing methods that use as little computation as possible, we also are limiting the power lost by processing.

[2] This is seen again in the section on power spanners, where β there refers to the amount of power needed to send a signal

interference, the more readily the signal will decay, and the higher that exponent should be. In a completely open environment, with no interference, the exponent is 2 [44].

Other metrics have been presented, some focusing on the average amount of energy required to actually get the transmission received (due to interference from other nodes), but they have not proven to be nearly as popular. The above model seems, for now, to strike the right level of complexity needed when designing routing algorithms.

1.3 Routing

The goal in ad-hoc networks is for every device to be able to communicate with any other device in the network. The vast array of papers and partial solutions to this effect indicates that this problem is far from trivial.

The first, and most essential goal in routing is to ensure the connectivity of the network. No power efficient algorithm is of any use if it stops devices from being able to communicate with the system. As mentioned above, power efficiency is the key area of research in routing and most algorithms are rated in terms of power efficiency. It may also be appropriate to limit the amount of connections any one device is responsible for. If a device is responsible for sending data to too many other devices, the battery will quickly run down. Not only is this scenario unfair to the device that is overworked, if the device goes down the network may become disconnected, thus breaking the routing structure. It seems that this last requirement

has been very difficult to control.

Classical routing methods, such as hierarchical protocols, have been tried in ad-hoc routing. These protocols have been well studied as they have been used in classical computer networks since at least the 1970s. The basic technique used in a hierarchical protocol is to recursively decompose the network into neighbourhoods which are responsible for their own routing. Unfortunately, most of these schemes rely on heuristics and are not theoretically well understood. Even worse, they are computationally complex or human aided and ill-suited for small devices.

A newer technique is to make better use of the geometry that is inherent in an ad-hoc network. These algorithms are designed to make use of localized information, in which nodes only know of a few neighbors and attempt to create a local routing scheme based on this limited information. Since each node knows its own location, its neighbors' locations, and the location of the node that a packet is directed to, a simple routing protocol is for each node in the path to pass the packet to the node that is geographically closest to the destination node. Using more advanced geometric methods, such as spanners, it is possible to build more efficient routing schemes. Because of the low memory and computational requirements of these routing algorithms, such schemes have become increasingly popular and likely represent the direction that all ad-hoc routing protocols will eventually take [28] [44] [12].

1.4 Goals and Metrics

The purpose of a routing algorithm is to produce a good routing system. Obviously, "good" by itself is not a particularly descriptive term. Rather, we measure the effectiveness of routing using metrics. Which of these metrics we deem to be important changes over time, and some have been shown to be inaccurate measures. Some have been shown to be in opposition to others, so that algorithm designers must find some balance between them. The following are metrics and goals that have been identified in the literature as important.

1. *Energy Usage.* The hope is to minimize the energy used for every routing task. Early on, Macker and Corson [33] cited the number of hops as the most important judge of the effectiveness of a routing scheme. This has been largely supplanted by energy usage, which is seen as a more accurate gauge of the efficiency of a network. Because many of the devices have limited battery life (as is the case with laptops, PDAs, cellphones, etc.), low power routing allows the devices to stay part of the network for a longer period of time. A problem that seems systemic with power-efficiency metrics is that they favor long paths composed of numerous short jumps. They promote potentially less stable networks than those with a few long but stable hops. A secondary issue is that the energy use is usually "diffused" throughout the network, so that two devices that could communicate directly will make use of numerous other nodes to send their communication as this is more energy efficient. The problem is

that the other nodes have been forced to cover the cost of these transmissions, which could be seen as unfair [3].

2. *Overhead Bits per Message.* This refers to the amount of data that is required to set up the routing scheme. In a fixed ad-hoc network, this may not be of great importance, but in a network where the devices are moving the routing scheme will have to be reconstructed fairly often. In such situations, it is very important to minimize the amount of data transferred as to not flood the network.

3. *CPU Usage.* This refers to the time complexity of the algorithm which either creates the routing tables or determines how to route a packet. Complexity is important as the devices have very weak processors and processing data needs to be done quickly, otherwise it will create outdated tables, disrupt the usage of the device, and wear down the battery of the device.

4. *Requires GPS.* In some algorithms, known as Location Aware Algorithms, every node is able to give its exact position via GPS. It is not always necessary to have GPS enabled devices to do geometric routing; relative signal strength can be used to gauge the distance of one node from another.

5. *Distributed.* The routing scheme should be determined locally, so that any given node in the network is able to determine how it should route packets by itself. An important aspect of ad-hoc routing is that the network can change

[3]By fairness, we mean the proportionality of the energy being expended by individual nodes. As we wish each node to stay in the network as long as possible, being unfair to one node may force them to exit the network due to battery failure.

quickly and nodes can quickly go on or off line. In this manner it makes sense that each node should be as independent as possible. Localized algorithms resemble greedy algorithms in that each node chooses its best course of action which in turn produces good results at the network level. Normally this course of action is determined from the node's location, the location of its neighbors, and the destination of the message. To draw a contrast, Strojmenovic and Lin [47, p. 4] state that

> All nonlocalized routing algorithms proposed in literature are variations of shortest weighted path algorithm[s]

6. *Memorization.* The amount of a knowledge that each node is required to have about the history of the routing scheme. Ideally, nodes do not need much, if any, knowledge of how routing has been done in the past to perform effectively in the future. Furthermore, as the network changes rapidly, old information is more likely to be incorrect.

7. *Concurrent tasks.* An efficient network is able to perform numerous routing tasks simultaneously. A routing scheme should maximize the number of concurrent tasks. This metric is difficult to test theoretically, but can be tested via simulation.

8. *Delivery Rate.* If possible, a routing scheme should have guaranteed delivery of all messages. This metric can measure the difference between the perfect delivery rate and actual delivery rate.

9. *Fault Tolerance.* A network should be stable, if at all possible. This means that

if nodes go offline or move, the algorithm can either quickly correct the routing tables or is able to ignore the change such that messages are still delivered.

As we will see later, it is valuable to use graphs to represent the paths between nodes that the routing scheme uses. The following are some metrics that make more explicit use of graph properties (summarized in Table 1.1):

10. *Connected.* Above all, the resulting graph must be connected if the underlying graph of the nodes and their potential connections is connected. Any scheme which fails this condition is likely not worth further evaluation.

11. *Bounded In/Out Degree.* Degree of a node, and in the case of directed graphs out degree is more important, has been seen as a major concern. The idea is that if any node has too high a degree, it will be frequently used for routing and then will quickly burn out. However, it is possible to build routing schemes that bound degree while not ensuring any fairness.

12. *Planarity, Sparseness, and Intersections.* Ideally, the resulting graph should be planar, or at least sparse with a limited number of edges and intersections. The purpose for this is twofold: first, planar graphs are well studied, and are thus easier to analyze; second, there is some argument that planar graphs indicate less interference and more importantly increased clarity in routing. Many routing schemes depend on the underlying graph being planar [13]. If numerous, distinct messages must be sent over an area simultaneously, there is a higher chance of signal error. Unfortunately, it has been shown that there

are graphs which permit no planar scheme if they are to remain connected [23]. This is a serious problem, and as such, it is at least as important to study the number of intersections in a graph and the number of edges in a graph.

13. *Loop Freedom.* The resulting graph should be as free of path loops as possible. The reason for this is that some researchers see there being a higher associated cost with maintaining topologies that allow loops. If the graph is rebuilt frequently, this may not be an issue.

14. *Hop Count.* The number of hops that are required to send a message should be minimized. This metric was originally seen as being among the most important in analyzing an ad hoc network. Its importance has waned some with the emergence of power minimizing routing schemes. However, the number of hops is still clearly important and a scheme should find a balance between path length and power efficiency.

15. *Spanning* The distance a message must travel should be minimised. Spanning is a measure of the difference of the lengths of the paths in the original graph and the routing structure. Ideally, message sent (via the routing scheme) between two devices should not take a substantially longer path than they would in the underlying graph.

Table 1.1: Metrics and Concerns in Ad-Hoc Routing Schemes.

Metrics and Concerns in Ad-Hoc Routing Schemes	
Metric	**Description**
Power Usage	The amount of power used per routing task. Methods of analysis include power spanners, etc.
Overhead Bits per Message	The amount of overhead in each message required for routing, or depending on scheme, the amount of data required to establish the routing scheme. Normally written using order notation.
CPU Usage	The time complexity of the algorithm that generates the routing scheme or the run-time complexity of routing.
Requires GPS	Location Aware Routing schemes assume that each node knows its precise location via GPS devices.
Distributed	The process of building routing tables is done locally, not globally.
Memorization	The amount of a priori knowledge about the network that is required by the routing scheme.
Concurrent Tasks	A scheme should maximize the number of concurrent routing tasks that the network can perform at any given time.
Delivery Rate	A scheme should maximize the delivery rate for a network.
Fault Tolerant	A scheme should build a stable network.
Graph Property Metrics	
Connected	If possible, the graph formed by the routing scheme must be connected.
Bounded In/Out Degree	No node has a degree above some constant amount.
Planarity, Sparseness, and Intersections	Is the graph of the scheme planar? In heterogeneous networks, there are cases where no planar scheme can be constructed. In such cases, the metric should determine if the scheme is sub-optimal in the number of non-planar crossings.
Loop Freedom	The scheme does not generate cyclical paths that data will travel over.
Hop Count	The number of hops necessary to send a message using the scheme versus the shortest (in hops) path in the network.
Spanning	The distance a message must travel using the scheme versus the shortest (in distance) path in the network.

1.5 Conclusion

Ad-hoc wireless routing has become a field of some importance in the last few years. Research is motivated by a very practical concern, enabling the communication of wireless devices in as efficient a manner as possible. What defines efficient is varied and sometimes contradictory, which makes for the great number of metrics that are used to evaluate a routing algorithm. While ad-hoc routing will make consumer electronics more versatile, it is also vital in creating temporary networks in areas that have no existing network, such as a disaster area. In this manner, ad-hoc routing could be as important/useful as more classical communication systems, like HAM radio, once were.

Here, we explore the Localized Delaunay Triangulation [29] [30] [28]. This structure was identified as having a variety of excellent extremal properties over uniform range Ad-Hoc Wireless Networks. However, some current research in the field has moved towards networks without uniform ranges. We show that while in general the Localized Delaunay Triangulation is not a spanner, the 1-hop Localized Delaunay Triangulation over the Intersection Neighborhood has a spanning ratio of $O(j)$ where j is the ratio between the longest and shortest edges in the graph. Furthermore, we show the Localized Delaunay Triangulation may have $\Omega(n^2)$ edges and $\Omega(n^2)$ intersections, where n is the number of nodes in the graph. We contrast these results with those gathered from experimental trials conducted on a variety of graphs.

Chapter 2

Geometry and Geometric Structures

2.1 Introduction

Geometry, specifically Computational Geometry, has been shown to be a useful and important field within Computer Science [14]. The focus of the study has been the computational side of discrete and combinatorial geometry. As was hinted in the first chapter, a natural (and combinatorial) representation of networks is as a graph. Classical graphs, however, tend not to be enough – nodes have no position and edges can have weights that do not need to obey the triangle inequality, both useful properties. To this end, we use a Euclidean (or Geometric) Graph instead of a simple weighted graph.

Definition 1. *A Euclidean graph $G = (S, E)$ is a graph where the vertex set is a set of points S in the plane and the weight of any edge $uv \in E$ (where u and $v \in S$) is equal to the Euclidean distance between u and v, which we denote by $|uv|$..*

Given this form of a graph, we can construct graphs where edges are included if and only if specific geometric properties hold.

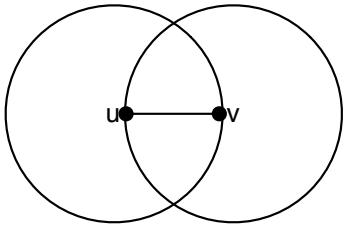

Figure 2.1: In the Relative Neighbourhood Graph, there exists an edge between nodes u and v if and only if the intersection of the circles (each of size $|uv|$ are empty).

2.2 Geometric Structures

While any geometric graph, triangulation, polygon or solid may be correctly identified as a geometric structure, only some are seen as having much theoretic interest. Often, what is of interest is the emergent properties of structures that have fairly simple definitions. There are many well studied, simple geometric structures, with well understood properties. A few pertinent ones are briefly described here so that they may be referred to as needed.

2.2.1 Relative Neighborhood Graph

Definition 2. *The Relative Neighborhood Graph, $RNG(V)$, is a geometric graph G over some point set V. There exists an edge uv between nodes u and v of G, if and only if the intersection of the two circles circles centered at u and v of radius $|uv|$ is free of any other node of V [48]. See Figure 2.1.*

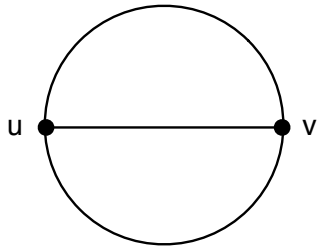

Figure 2.2: In the Gabriel Graph, there exists an edge between nodes u and v if and only if there exists an empty circle with edge $|uv|$ as its diameter.

2.2.2 Gabriel Graph

Definition 3. *The Gabriel Graph, $GG(V)$, is a geometric graph G over some point set V. There exists an edge uv between nodes u and v of G if and only if the circle passing through nodes u and v with diameter $|uv|$ is empty of nodes[22]. See Figure 2.2.*

2.2.3 Delaunay Triangulation

The Voronoi Diagram[49] [8] and its dual the Delaunay Triangulation[17], are two of the most well understood and widely studied structures in Computational Geometry.

Definition 4. *The Voronoi Diagram of a set of points S in the plane is a subdivision of the plane into polygonal cells, each containing exactly one point of S. The cells (or regions) are constructed such that each point $p \in S$ is closer to any position within its corresponding region than is any other point of S.*

We denote this structure $VD(S)$, where S is a set of points in the plane. It

is known that the Voronoi diagram can be constructed in $O(n \log n)$ time, where $n = |S|$ [19].

There are two definitions of the Delaunay triangulation. We will give both.

Definition 5. *The Delaunay Triangulation of a set of points S in the plane can be considered a dual of the $VD(S)$. In the Delaunay Triangulation, there is an edge between two points $p, q \in S$ if and only if their corresponding Voronoi regions are adjacent. See Figure 2.3.*

Alternately, we can define the Delaunay Triangulation independently of the Voronoi Diagram:

Definition 6. *The Delaunay Triangulation of a set of points S in the plane is a Euclidean graph containing all points of S. For each pair of points $p, q \in S$, there exists an edge in the Delaunay Triangulation if and only if there exists a closed circle passing through both points that does not include any other points of S.*

Assuming that there are no four points of S which are cocircular, this forms a proper triangulation (that is, if possible each triangle side is shared by one other triangle).

We denote this structure $DT(S)$ where S is a set of points in the plane.

The Delaunay Triangulation is planar, and algorithms are known that can generate it in $O(n \log n)$ time. Chapter 3 examines the use of a localized version of the Delaunay Triangulation as a candidate for a routing algorithm.

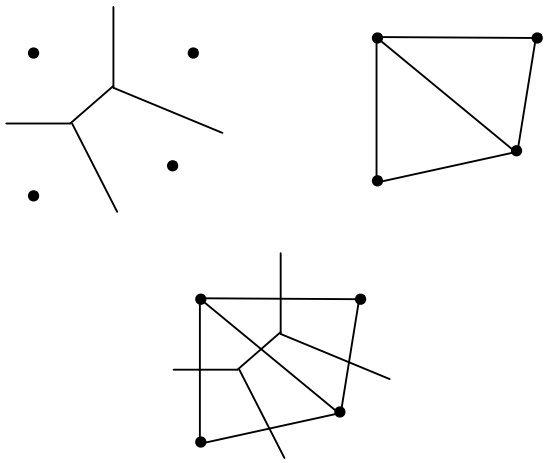

Figure 2.3: On the top left is the Voronoi diagram of the point set. Every position within a given region is closer to its associated vertex than any other. On the right, is the Delaunay Triangulation, the dual of the Voronoi Diagram. Below, the two are superimposed.

2.3 Spanners

Spanners have been of some interest in graph theory and geometry since their introduction by Chew [15]. Given a graph G, a spanner is a subgraph H containing all vertices but not necessarily all edges such that distances over graph H are not radically larger than over graph G.

Definition 7. *The distance between two nodes, $u, v \in G$ (where G is a connected weighted graph) is the total weight of the shortest path between u and v in G. We denote this value $d_G(u, v)$. A connected subgraph H of G is called a t-spanner for G if for each $u, v \in G$, $d_H(u, v) \leq t \times d_G(u, v)$ [41]. A Euclidean (or geometric) spanner [15] [18] is a t-spanner where all weights are based on the Euclidean distances between nodes.*

This value t is known as the *stretch factor* of the subgraph H. We will say a subgraph is a spanner if it is a t-spanner for some constant t.

A power spanner [7] [27] is a similar construct. Instead of considering just geometric distance, we weight the edge between nodes v_{i-1} and $v_i \in V$ based on their distance raised to some power β, or $|v_{i-1}v_i|^\beta$. Therefore, the cost of a path Π with h hops can be written

$$p(\Pi) = \sum_{i=1}^{h} |v_{i-1}v_i|^\beta \tag{2.1}$$

Let value $pG(u, v)$ be the most power efficient (least costly) path between points u and $v \in G$. Similarly, let $pH(u, v)$ be this value in subgraph H. Then the *power stretch factor* of H is

$$\rho H(G) = \max_{u,v \in V} \frac{pH(u, v)}{pG(u, v)} \tag{2.2}$$

It is known that, for some constant δ, $\rho H(G) \leq \delta$ if and only if for any edge $v_iv_j \in G$ such that $v_iv_j \notin H$, $pH(v_i, v_j) \leq \delta|v_iv_j|^\beta$ [31]. Furthermore, it is clear that for any known geometric spanner H which spans some G with a factor of δ, the power stretch factor of H cannot be worse than δ^β.

Even though some subgraph H of G may not be a constant spanner, it may be possible for it to be a power spanner. For instance, the Gabriel Graph (GG) [22] is, in general, not a t-spanner for any constant t as its stretch factor is known to lie between $\frac{\sqrt{n}}{2}$ and $\frac{4\pi\sqrt{2n-4}}{3}$ (where n is the number of nodes)[10] but it is known to have a power stretch factor of 1 over certain graphs[31].

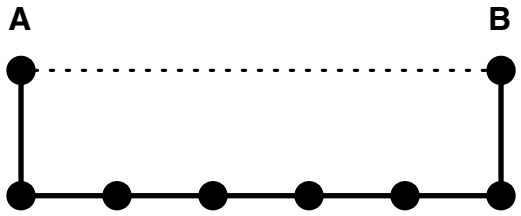

Figure 2.4: In this graph where the edges include the solid and dashed segments, the most efficient path between points A and B is the solid path. However, if A and B wish to communicate regularly, every node in the graph must expend energy instead of just A and B, making the dashed path more fair to the other nodes.

Because of the β exponent, a power-efficient path can sometimes be far longer than the original path, see Figure 2.4.

Again, it is important to note that low power cost spanners are not always enough. While its power stretch factor would appear to make the GG an attractive structure to use in the construction of routing topologies the geometric stretch factor is unbounded. Also, It is known that the in-and-out degrees may be unbounded. Figure 2.5 presents a worst case scenario for degree in a Gabriel Graph. The problem of creating spanners where each node has some bounded degree is an active area of research [32].

2.3.1 Weak Spanners

A recent and interesting development in computational geometry and more recently ad-hoc routing is the introduction of so-called weak spanners [45].

Definition 8. *A subgraph H is called a weak t-spanner for G with value t if for each*

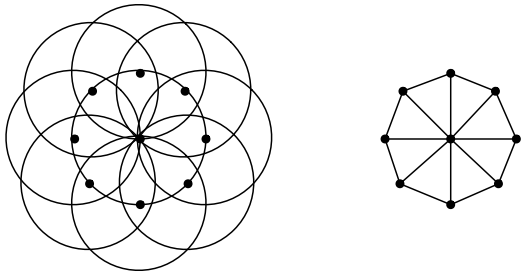

Figure 2.5: An example of why the Gabriel Graph (here over a Unit Distance Graph, see page 22) has a poor topology if bounding the degree of nodes is deemed important. While each node on the rim has a degree of 3, the center node has a degree of $n-1$. On the right, the graph is drawn with edges, on the left only the vertices and their ranges are shown.

$u, v \in G$ there is a path from u to v in H that can be contained in a circle of radius $t \times d_G(u, v)$. See Figure 2.6.

Any geometric spanner is also a weak spanner, although the converse is not necessarily true. Surprisingly, any weak spanner is also a power spanner [45]. This spanner hierarchy is described in table 2.1.

Table 2.1: Spanner Heirarchy (adapted from [45]).

Spanner Heirarchy			
spanner type	**stretch factor**	**weak stretch factor**	**power stretch factor**
euclidean t-spanner	t	t	t^β
weak t-spanner	(unbounded)	t	$O(t^{4+\epsilon}/(1-2^\epsilon))$ for $\beta = 2+\epsilon$ $O(t^6)$ for $\beta = 2$ else, unbounded
θ-power spanner	(unbounded)	(unbounded)	t^β

Weak spanners present an additional tool when analyzing a geometric structure

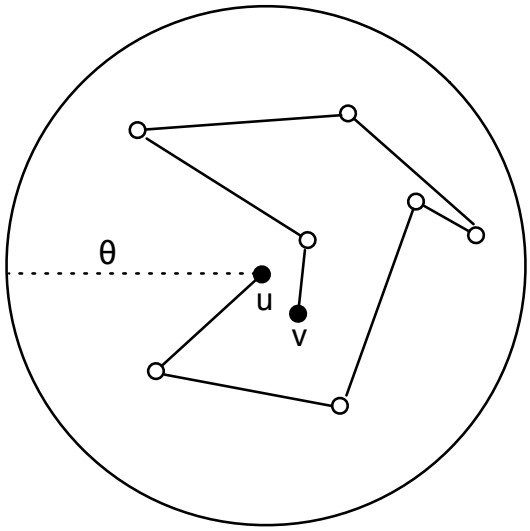

Figure 2.6: A weak spanner. Given the distance between nodes u and v, there is some circle of bounded radius (here denoted θ) that contains the entire path from u to v.

that may be used in routing.

2.4 Graphs

We consider graphs formed from a set of nodes (where each node represents a device) that we then embed in the plane. We first consider two classes of graphs that have been previously described and used to model ad-hoc networks.

2.4.1 The Unit Distance Graph

When ad-hoc networks were first being analyzed via geometry, the method was to assume each device had an equal broadcast range. The graph of this structure, the Unit Distance Graph, turns out to be well defined and well explored [25] [7] [32] [29].

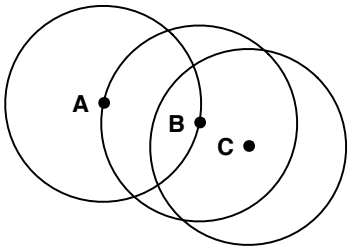

Figure 2.7: A unit distance graph (UDG). Here, we can see the vertices of the graph and the circles corresponding to the range of a node. Note that nodes A and B cannot move any further apart and still be connected, while B and C have some room to move. In the future, routing schemes may take this aspect of stability into account.

See Figures 2.7 and 2.8.

Definition 9. *Unit Distance Graph (UDG) is a Euclidean graph G wherein two vertices are joined by an edge if and only if they are at distance one or less.*

As it is well studied, it is worth noting a few of the more important features of the UDG.

A result that initially seems surprising is that the Gabriel Graph over the Unit Distance Graph (that is, the intersection of the Gabriel Graph and the Unit Distance Graph) is a 1 power spanner [28]. However since the $GG \subset DT$, the power result is encouraging.

While the UDG has been a focus of study for some time, there has been an acknowledgment that it is too simple a structure to accurately represent an ad-hoc network.

Still, it is important to note some of the vital findings on UDG when it comes to ad-hoc routing.

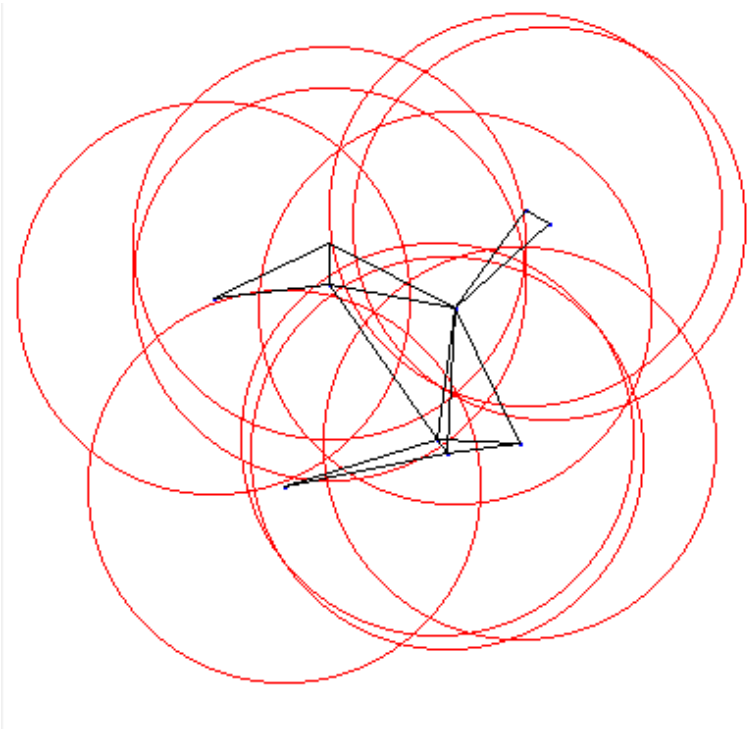

Figure 2.8: An example Unit Distance Graph. Here, we can see the circles corresponding to the range of a node. All circles are of unit distance, limiting the length of any edge to 1.

Theorem 1. *The Unit Delaunay Triangulation [28](The intersection of the Delaunay Triangulation and the Unit Distance Graph) is a* $\frac{4\sqrt{3}}{9}\pi \approx 2.42$ *spanner of the UDG [11].*

This is identical to the spanning property of the general Delaunay Triangulation over some point set V [24].

The Unit Delaunay Triangulation (UDT) is an excellent spanner. Unfortunately, it is an open problem if there is an efficient method to locally generate the structure.

In order to localize an algorithm, it is often necessary to speak of a "neighborhood" of a node.

Definition 10. *For some node* v *in graph* G*, the* k*-neighborhood is the set* K *of all nodes that are reachable via a path of* k *edges (or "hops"), where* k *is an integer* $0 \leq k \leq n$*, where* n *is the number of nodes. The* k*-neighborhood of* v *where* $k = 1$*, then, is the set of nodes containing itself and those that are immediate neighbors to* v*.*

Given this, it is possible to describe localized versions of classical geometric structures, for example, the Gabriel Graph.

Definition 11. *The* $k-$*localized Gabriel graph (*$GG(V)$*)consists of all edges* uw *such that the open disk using* uv *as the diameter does not contain any vertex* $z \in V$ *which is a* k*-neighbor of either* u *or* v*.*

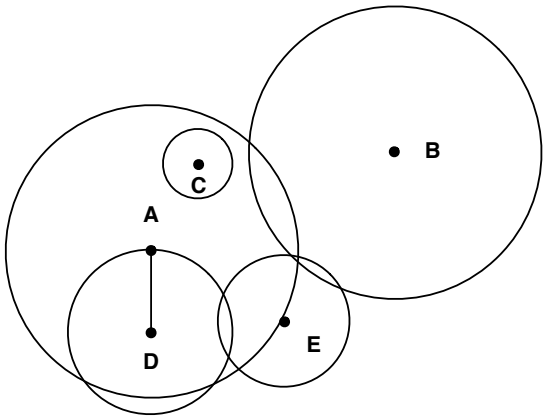

Figure 2.9: A Mutual Inclusion Graph (MIG). For an edge between two nodes to exist, both nodes must fall within each other's range. Here, we can see the circles corresponding to the range of a node. While C and E are both in A's range, A is not in theirs. A nor B are within each other's range. A and D are the only nodes which are mutually inclusive and thus the only edge is between them.

2.4.2 The Mutual Inclusion Graph

The problem with the UDG is that, while it is easy to analyze, it is too simple a structure to accurately (or adequately) model ad-hoc networks. The UDG assumes that each device has a uniform range, which is not a valid assumption. Not only do antennas vary in strength from one to another, the range of a device may vary over time. To overcome this, we consider a structure that can model nodes with non-uniform ranges. Such networks are referred to as having *heterogeneous ranges*. See Figure 2.9.

Definition 12. *An Unrestricted Mutual Inclusion Graph (UMIG) is a Euclidean graph G where each node $v \in G$ has an associated weight w_v. Two nodes $u, v \in G$ are joined by an edge in G if and only if the distance between u and v is less than*

min(w_u, w_v).

We sometimes refer to the weight of a node in the MIG as its range.

Clearly, one can envision a version of the MIG where there is a fixed bound on the difference between the smallest and largest weight.

Definition 13. *A Mutual Inclusion Graph (MIG) [23] is a graph G where each node $v \in G$ has an associated weight w_v where $1 \leq w_v \leq l$ and l is a fixed constant. Two nodes $u, v \in G$ are joined by an edge in G if and only if the distance between u and v is less than min(w_u, w_v).*

Alternatively, we can define the UMIG as using vertex weights in the range $0 \leq w \leq l$ as this eliminates a pre-bounded ratio between the largest and smallest weights.

The UMIG, however, has an extremely bad property: there exist graphs for which there is no bounded degree, connected subgraph.

Lemma 1. *There exists an Unrestricted Mutual Inclusion Graph which allows no connected subgraph to have bounded degree.*

Proof. Consider the following UMIG. At the center of the graph, we have node a, which has an arbitrarily large weight – large enough to exceed the weight of all the other nodes in the graph. In the four compass directions, we place a node with weight 1 such that node a falls just on the edge of each range. See Figure 2.10.

Each of these satellite nodes has one edge, and that edge connects to a. The degree of a is now 4, and all nodes in the graph connect to it.

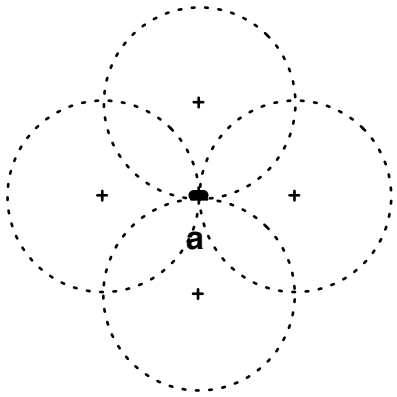

Figure 2.10: Assume a has a large enough range to contain all the nodes. The four surrounding nodes can each only see a. For this graph to be connected, each of the edges entering a must be included, which makes the degree of a four.

We now place another node on each of the compass coordinates, with range and distance from a just slightly greater than twice that of the original four nodes. In this manner, they are only able to communicate with the central node a, increasing its degree to 8. See Figure 2.11.

In this manner we can continue to add nodes to the graph, increasing the degree of node a to $n - 1$. □

Theorem 2. *The maximum degree for a node in a spanning subgraph of the points of a MIG may need to be* $\Omega(\log_2 l)$*, where* l *is the longest range of a node.*

Proof. We construct an example where a node of degree $\Omega(\log_2 l)$ is needed.

The smallest allowable nodes are of range 1, and they surround a central node a. The closest that nodes can get without communicating is for them to be placed every 60 degrees around the circle. This results in a degree of 6. We repeat and

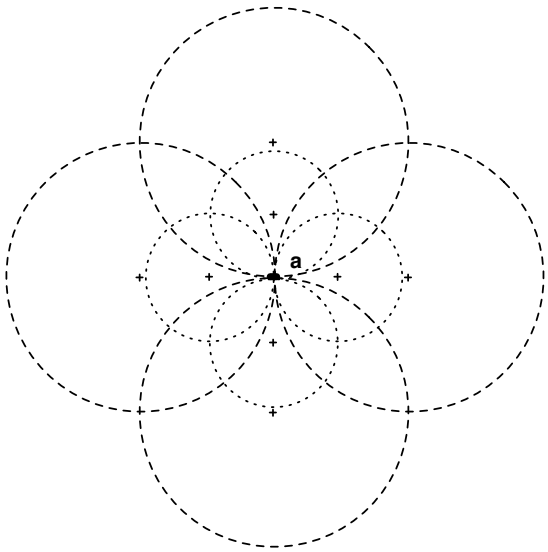

Figure 2.11: Assume a has a large enough range to contain all the nodes. Each of the surrounding nodes can see only a. For this graph to be connected, each of the edges entering a must be included, which makes the degree of a eight. In this manner we can continue to add nodes that only connect to a. In the UMIG, this makes the degree of a unbounded.

place another ring of nodes around the central node. These must be twice as far away as the initial nodes and have twice the range. This doubling of range can be performed $\log_2 l$ times, each step increasing the degree by 6. Therefore the degree of the central node is no less than $6 \log_2 l$, or $\Omega(\log l)$. □

In the UDG if a node A could send a message to some node B, B automatically could send a message to A. Because of the manner we have defined it, the MIG also has this symmetric property, even if the underlying structures (the variable power nodes) do not demand it.

We see that in the MIG, edges only exist between nodes if they are each *mutually contained* within the other's range[1]. These edges, then, are undirected. It may seem usual to define a graph where edges are directed and thus an edge exists from some node A to some other node B where B is contained within A. At first glance, such a scheme seems to make little sense in practice – since devices continually enter and exit a system in an ad-hoc network, no node can be sure another exists without receiving an acknowledgment from the other. However, there may be situations where the MIG should be weakened to a directed inclusion graph, since there are conditions where large nodes are at least able to send data when they previously would not be able to, and there may exist more direct paths than what would normally be allowed. Some preliminary work on unidirectional links [35] did not yield overly promising results. However, more work is needed before it can be written off completely.

The MIG may not have a planar embedding despite being potentially planar. Li

[1]Interestingly, the larger node gives away more information than the smaller. "He is seen, but does not see; he is the object of information, never a subject of communication." [21]

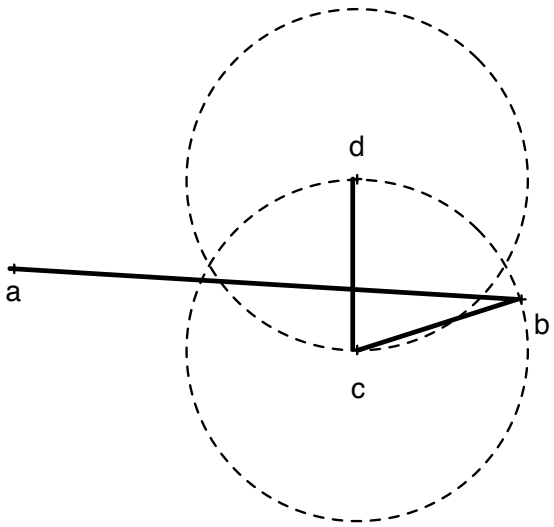

Figure 2.12: Assume a and b have a large range. There is no manner for any subgraph of this graph to be both connected and free of crossings.

et al. [30] give an example of a graph in the MIG for which no connected subgraph can avoid edge crossing. This graph is reproduced in Figure 2.12.

2.4.3 Unions and Intersections

In a structure like the Unit Gabriel Graph (the Gabriel Graph over the UDG), if there is a node which would break a Gabriel edge (by lying within the circle), then both of the nodes forming the edge would see that. This is a useful geometric property of a network with uniform ranges. In graphs with multiple weighted-node ranges, like the MIG, there are cases where two nodes may be connected, but may not be mutually connected to a third node which lies between them. When creating certain structures, such as the Delaunay Triangulation or Gabriel Graph, handling these nodes can be a problem.

To even describe such scenarios, it is necessary to strengthen the vocabulary used to describe neighborhoods. While in the UDG there was simply the immediate and k-neighborhoods of a node, here we introduce two new definitions:

Definition 14. *Given a graph G, the Union k-Neighborhood (UN) of two nodes $u, v \in G$ is the set of vertices that contains all the k-neighbors of either u or v.*

Definition 15. *Given a graph G, the Intersection k-Neighborhood (IN) of two nodes $u, v \in G$, is the set of vertices formed from the intersection of the k-neighborhood of u and the k-neighborhood of v. Any node w in the intersection neighborhood is said to be* co-visible *to u and v.*

Briefly, we consider the Gabriel Graph given these two types of neighborhoods.

Definition 16. *Given a geometric graph G over some point set V, an edge $uv \in G$ is an edge of the 1-hop Union Neighborhood Gabriel Graph if and only if there exists an empty circle passing through nodes u and v, with $|uv|$ as its diameter, that contains no node $w \in V$ that is adjacent to either u or v. See Figure 2.13.*

Definition 17. *Given a geometric graph G over some point set V, an edge $uv \in G$ is an edge of the 1-hop Intersection Neighborhood Gabriel Graph, $GG_I(V)$ if and only if there exists an empty circle passing through nodes u and v, with $|uv|$ as its diameter, that contains no node $w \in V$ that is adjacent to both u and v. See Figure 2.13.*

We sometimes speak of a structure being constructed "over" the UN or IN – this indicates which neighborhood we are using.

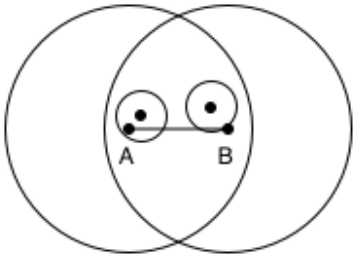

Figure 2.13: In this MIG, A and B both have neighbors that the other cannot see, however, the only edge between vertices in $IN(A, B)$ is the edge drawn above. The Gabriel Graph of this MIG contains AB over the IN, but does not contain AB over the UN.

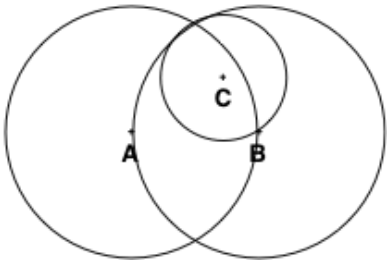

Figure 2.14: If A discovers in 2 hops that C is within the Gabriel Circle of the edge AB, then edge AB will not be included, causing the graph to become disconnected.

Figure 2.14 gives an example, of how the k-hop Intersection Neighborhood can still produce disconnected graphs when used for triangulations and other structures, if $k \geq 2$. While there are ways to overcome this problem, they are somewhat impractical.

2.5 Edges and Sparseness

The *sparseness* of a graph refers to the overall density of the edges which compose it. Some graphs are sparse enough to be planar, while others are very dense (such

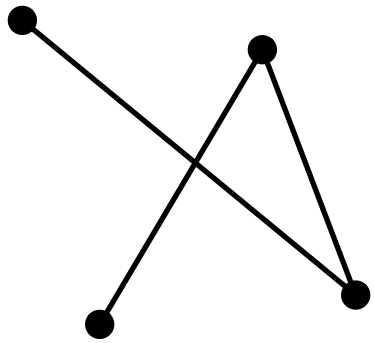

Figure 2.15: A self-intersecting P_3. A graph without any copies of a self-intersecting P_3 contains no more than $O(n \log n)$ edges.

as a complete graph).

Certain properties make it much easier to analyze graphs. For instance, if it is known that a graph has a tree topology, many algorithms can run in low-polynomial times. Finding such extremal properties of graphs has been a major focus of research for graph theorists. A well established method of analysis is to identify *forbidden* geometric configurations to determine bounds on the number of edges. Pinchasi and Radoicić [42] give the general question as:

> Let H be a so-called *forbidden* geometric configuration or a *class* of forbidden geometric configurations. What is the maximum number of edges that a [geometric] graph with n vertices can have without containing any forbidden configuration?

Using a combinatorial reduction, they went on to show that the if a geometric graph G has no self-intersecting cycle of length 4 (C-4), it has $O(n^{8/5})$ edges. A graph with no self-intersecting copies of P_3 (see Figure 2.15) has been proven to have $O(n \log n)$ edges [39].

2.6 Conclusion

To create valid routing schemes in ad-hoc networks, it is necessary to describe the physical relationship between nodes. Here, geometry seems to be the natural tool, as the devices exist, more or less, on the plane[2]. Furthermore, graphs seem to be a natural representation for networks, and geometric graphs are even more suited for the task.

Researchers first made use of the Unit Distance Graph, but eventually that model proved to be insufficient. So the Mutual Inclusion Graph was developed, and it seems to capture the relationships between nodes very well.

From these structures, we must attempt to find subgraphs of the graphs that will serve in a routing scheme. As there are many well studied geometric structures in computational geometry, it seems natural to try to extend known structures to this new environment. These structures can then be evaluated for spanning and sparseness properties.

[2]Also, should we move to three dimensions, such as trying to model an office building, it makes sense to continue to use geometry.

Chapter 3

The Localized Delaunay Triangulation

3.1 Introduction

The previous chapter concerned itself with describing and establishing properties of spanners and the various classes of geometric graphs that exist. We now examine a variation of the Delaunay Triangulation known as the Localized Delaunay Triangulation. The Localized Delaunay Triangulation (LDel) is similar to the regular Delaunay Triangulation, although it is considered over some k-neighborhood as described in Section 2.4.1.

Definition 18. *An edge uv is called k-localized Delaunay relative to a geometric graph G if the interior of any circle through u and v does not contain any vertex of V that is a k-neighbor in G of u or v [29].*

Definition 19. *The k-localized Delaunay graph relative to a given geometric graph G, over some set of vertices V ($LDel^k(G)$) contains exactly all k-localized Delaunay edges of G.*

Clearly, if k is allowed to equal $|V|$ where V is the set of vertices in G, then the

Localized Delaunay Triangulation is equivalent to the Delaunay Triangulation of the given geometric graph.

3.2 Results on the Unit Distance Graph

Work in the area of geometric routing began by analyzing the Unit Distance Graph. As noted before, the Unit Delaunay Triangulation was identified as a good spanner [11], although it was unclear if it could be built locally. However the $LDel^k$ is easily constructed over such a topology. The $LDel^k(UDG)$ is known to be a planar graph when $k \geq 2$ [31] . Note: $LDel^n(UDG) = UDT(V)$.

Since the $LDel^i(G) \supset UDT(G)$, $LDel^i$ has a spanning ratio, at worst, of ~ 2.42 [11] [24].

3.3 Results on the Mutual Inclusion Graph

More recently, Kapoor and Li [23] have redefined the Localized Delaunay Triangulation to take the concept of union and intersection neighborhoods into account. See figure 2.13 for an example of the union and intersection neighborhoods.

Definition 20. *An edge uv is called k-localized Delaunay over the Intersection Neighborhood (IN) relative to a given geometric graph G if the interior of any circle through u and v does not contain any vertex of V that is a k-neighbor in G of u and v.*

Definition 21. *The k-localized Delaunay graph relative to a given geometric graph G,*

($LDel^k(G)$), over the Intersection Neighborhood (IN) contains exactly all k-localized Delaunay edges of G over the IN.

At the end of their paper [23], Kapoor and Li present a number of open problems. Of particular interest is the problem of determining if the Localized Delaunay Triangulation over the Intersection Neighborhood was a spanner, which we will determine.

We now consider spanning properties. Using only vertices of weight 1 or 2 we are able to show that the $LDel^k(MIG)$ over the IN is not a spanner [50] when k is arbitrary. We call such a graph a One-Two Graph (OTG).

Theorem 3. *$LDel^k(OTG)$ over the Intersection Neighborhood has stretch factor of at least* $\min(2(k-1), (n-2)/2)$,, *for* $k \geq 2$.

Proof. Figures 3.1 and 3.2 show the method we employ for $k = 3$. For clarity, we have only shown the top half of the graph in Figure 3.2; The entire structure (aside from a and b) is mirrored below. All nodes, aside from i_1 and i_2 have a range of 2. Note that the node i_1 is initially hidden to both a and b so that in $LDel^1$ the path between a and b is 2. While i_1 breaks the Gabriel circle for a and b, it is not until $k = 3$ that the node becomes co-visible to a and b. When this happens, there is no longer a $LDel$ edge between then, and they are forced to use the long route around the graph. So while the initial cost was 2, the number grows to 8 in $LDel^3$.

It is possible to construct a similar structure for $k \geq 4$. To prove the lower bound on the stretch factor for large k we construct a set of points V. Add two nodes a and b to V, both with a range of 2. Place these nodes precisely distance two from each other. Clearly there can be no path from a to b which is of less length than length 2,

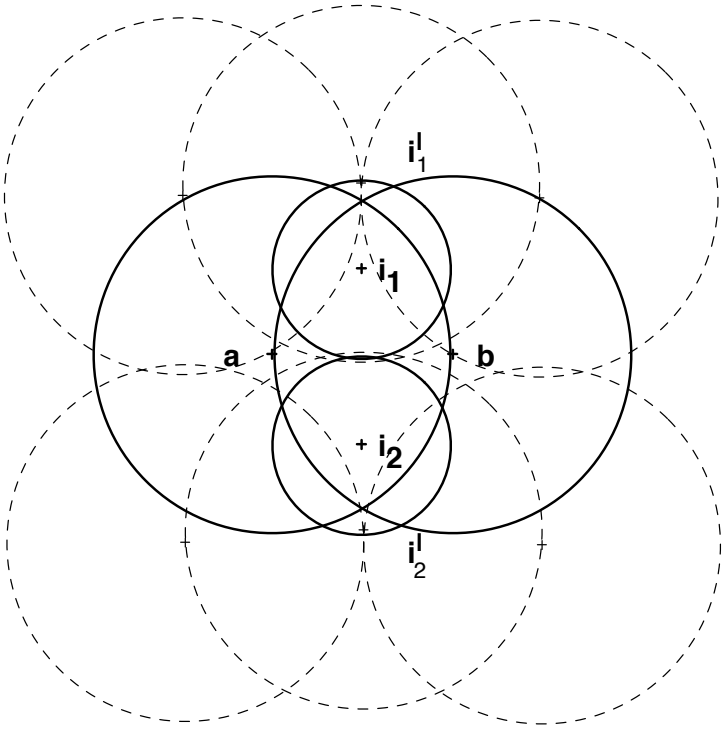

Figure 3.1: This arrangement of nodes is used as the basic model to show why the Localized Delaunay Triangulation will not produce a valid spanner. We are attempting to create the longest possible path between a and b. Each node in the structure has a range of 2 except i_1 and i_2, which have a range of 1.

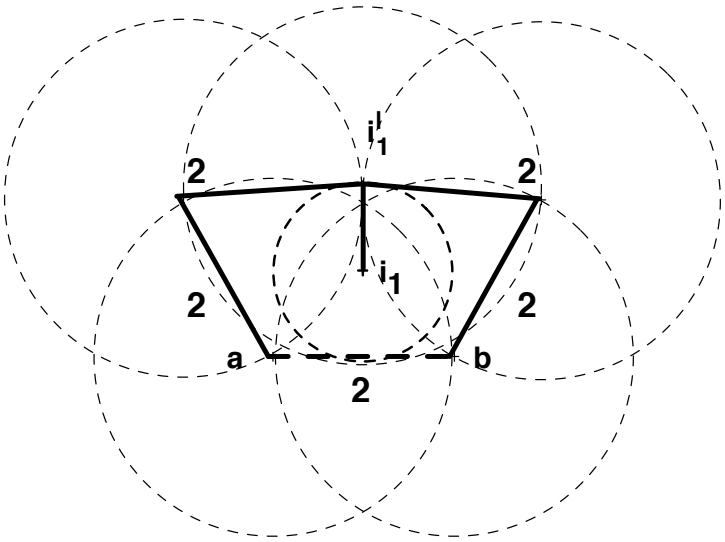

Figure 3.2: We are considering the set of nodes from figure 3.1. If we only consider one hop (the neighborhood of edges where each member is one edge away), we see that nodes a and b are able to see each other directly, and the cost to move between them is 2, which is optimal. With three hops, the geometric spanning distance between points a and b has risen to 8 as i_1 (and i_2, mirrored below) prevent the edge ab.

so if there is to be a spanner over the MIG, the shortest path can be no more than some constant t longer times 2.

We place a node i_1 with range 1 just within the top of the Gabriel circle of nodes a and b, so that neither a nor b can directly communicate with it. We likewise place another such node i_2 at the bottom end of the circle. Now, for a k large enough so that a and b are aware of i_1 and i_2, there is no valid Delaunay Edge between a and b, as there is one node within any circle though a and b. We also place nodes $i_1^{'}$ of range 2 one unit above i_1 and likewise $i_2^{'}$ one unit below i_2.

For large k, to force a bad spanning ratio, we can construct paths with min $(k-2, \frac{n-6}{4})$ intermediate nodes, each with power 2, from a to $i_1^{'}$ and from b to $i_1^{'}$. We place each node distance two from the one that proceeds it. However, to disrupt any circle through a and b we add similar paths from a to $i_2^{'}$ and from b to $i_2^{'}$. Now edge ab cannot be included, and either path from a to b requires a walk through half of the graph. So the stretch factor is at least $2(k-1)$.

We see that the stretch factor is tied to k, but it is clear that if k is larger than $(n-2)/2$ (since half the nodes are used above and below the nodes a and b), then n becomes the limiting variable in the construction, and $LDel^n$ has a stretch factor of $\frac{n-2}{2}$. □

Corollary 4. *$LDel^k(MIG)$ over the Intersection Neighborhood is not a $2(k-1)$ spanner for $k \leq \frac{n}{2}$.*

Proof. Since OTG $\subset$ MIG, this follows immediately. □

Note that this rules out the possibility of the $LDel^k$ being a good spanner for a

large k. This alone does not always imply that a structure is not a power spanner. However, the $LDel^k$ is also clearly not a power spanner, as the power of the path taken between nodes a and b in the worst case is $\min[2(k-1)2^\beta, \frac{(n-2)}{2}2^\beta]$, versus 2^β in the original graph. So the lower bound for the power stretch ratio is

$$\rho H(G) = \frac{\min[2(k-1)2^\beta, \frac{(n-2)}{2}2^\beta]}{2^\beta} = \min[2(k-1), \frac{(n-2)}{2}] \tag{3.1}$$

This rules out the possibility of the $LDel^k(MIG)$ being a spanner for a large k. This says nothing about the spanning properties of the $LDel^1(MIG)$. From now on, unless otherwise noted, when the Localized Delaunay Triangulation is analyzed, it is the 1-hop version over the intersection neighborhood.

3.4 Spanning Properties of the 1-hop Localized Delaunay Triangulation

Having established a lower bound to the spanning ratio of the Localized Delaunay Triangulation over the Intersection Neighborhood for k-hop neighborhoods, we now turn our attention to the 1-hop version of the problem. In an attempt to make this problem easier to solve, we will restrict it further, much as we did with the One-Two Graphs. Here, we consider a variant of the Mutual Inclusion Graph wherein no node may be closer than unit distance to any other node. We consider the 1-hop $LDel$ of the Intersection Neighborhood over such a graph.

Lemma 2. *Given a MIG G where no two nodes are permitted to be less than unit*

distance from each other (the smallest range of any node in the system), the shortest path in the $LDel^1(G)$ over the IN between two points p and q, which are MIG neighbors, is at most distance l^2, where l is the largest range of a node in G.

Proof. We now show via induction on rank of the distance d between two nodes p and q adjacent in the MIG that the shortest path in $LDel^1(G)$ between p and q at distance is at most d^2.

Base Case: We consider the closest pair of covisible points p and q. Then pq is a Delaunay Edge and the distance between them is at most l.

Inductive Hypothesis:

For the jth shortest edge in the MIG between two points p and q of length d, the entire length of the shortest path in $LDel^1(G)$can be assumed to be at most d^2. Since the largest possible d is l, this is equivalent to a spanning ratio of l.

Inductive Step: Now we consider the $(j+1)$st shortest edge pq in the MIG where the distance between p and q is d^*. If pq is an edge in $LDel^1(G)$ the shortest path between p and q is of length d^*. Otherwise, we consider the general case where nodes p and q are covisible but there is no empty circle that allows the creation of a Delaunay edge. Let r be the point visible from p and q such that $\angle prq$ is maximum. We know that $\angle prq > \frac{\pi}{2}$ and it then creates two edges. See Figure 3.3.

If the edges are not of equal length, then the longer of the two edges, is labeled b, the shorter c. We refer to the interior angle between edges b and c as A.

By the cosine law

$a^2 = b^2 + c^2 - 2bc\cos A$

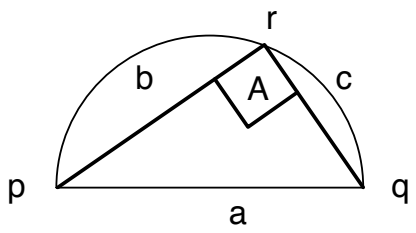

Figure 3.3: Here we see the nodes, edges, and angles that are considered in the inductive step of the proof.

Since angle A is greater than or equal to $\frac{\pi}{2}$, $\cos A$ is zero or negative, thus $a^2 \geq b^2 + c^2$.

By the inductive hypothesis, the length of path between nodes p and q is less than or equal to $b^2 + c^2$ which we know is less than or equal to a^2.

Therefore, the length of a path can be at most d^2. The longest possible distance between two co-visible nodes is l. Thus it follows that the spanning ratio of the one-hop Localized Delaunay Triangulation is no worse than l. □

We now consider a graph where the nodes are allowed to be arbitrarily placed (ie. there is no minimum distance between nodes), like the general case for the MIG. The restriction on the range is the same, bounded between 1 and some constant l.

Theorem 5. *Given a MIG G, $LDel^1(G)$ has a spanning ratio of $l \times 2.42$, where l is the ratio between the longest and shortest ranges.*

Proof. Consider a pair of nodes p and q in the MIG G such that $|pq| \leq 1$, it is known [11] that there exists a path from p to q in the Delaunay Triangulation which contains no edge of length greater than 1 and which is no more than $2.42 \times |pq|$. In

LDel terms, that means that the n-hop Unit Local Delaunay Triangulation is a 2.42 spanner. It is also known that the n-hop spanner is a subgraph of the 1-hop spanner, which means that the 1-hop spanner is no worse a spanner than the n-hop version.

In Proof 3.4, we finished after recursively decomposing the edges once they had reached length 1 or less. We can now allow this decomposition to continue to an arbitrary level without increasing the spanning factor by any more than 2.42 times. What this means is that there exists (at worst) a 2.42 length spanning path on each of the paths of length 1. This will increase each length 1 segment by 2.42. Thus the spanning ratio for the 1-hop *LDel* over the intersection neighborhood is no worse than $2.42l$. □

3.5 Sparseness of the Localized Delaunay Triangulation

So far we have been concerned with the spanner properties of the Localized Delaunay Triangulation, but the sparseness properties are equally important. Obviously, one can produce a spanner with fantastic spanning properties – the complete graph. Unfortunately, such a spanner fails to provide a reduction in the number of edges.

A classical result concerning graphs is that if a graph has a planar embedding, it has a linear number of edges [9]. Proving that such a graph is planar is not necessarily a difficult task, due to the following theorem:

Theorem 6 (Kuratowski's Graph Planarity Criterion)**.** *A graph has a planar em-*

bedding if and only if it contains no subgraph which is homeomorphic to $K_{3,3}$ or K_5 [26] [34].

Definition 22. *K_5 is a graph with 5 nodes containing every possible unique undirected edge. It is called a* complete graph *of size 5.*

Definition 23. *$K_{3,3}$ is a bipartite graph with 3 nodes in each partition such that each node is connected via an undirected edge to every node in the other set.*

The Union Neighborhood oriented versions of the 1-hop localized Delaunay Triangulation, have at most $O(n^{5/4})$ edges [23], however, it is unknown if there is a better bound on the intersection neighborhood structures than $O(n^2)$.

Using the union neighborhood model identified by Kapoor and Li as a basis, Pinchasi and Smorodinsky [43] have analyzed the Localized Delaunay Graph over the Union Neighborhood. Importantly, their results can be applied to any geometric graph rather than just the MIG.

For the two-hop Local Delaunay Triangulation (of the union neightborhood) of some geometric graph, there can only be $O(n \log n)$ edges and only $O(n^{3/2})$ edges for a one-hop graph. Their proof is to show that self intersecting copies of a length 3 path (that is, 4 nodes and 3 edges, a "P_3", see figure 2.15) are forbidden in G.

Furthermore, Pinchasi and Smorodinksy have shown that if a subgraph of a geometric graph consisting of the edges whose slopes are limited to fall within some constant range contains no subgraph which is homeomorphic to $K_{3,3}$ or K_5 , then the graph can contain at most only $O(n)$ edges. That is, it too has a linear number of edges, although likely many more than a graph that is planar without this angle

restriction.

While Pinchasi and Smorodinksy have established numerous properties of the Union Neighborhood variation of the *LDel*, little is known about the Intersection version. We propose to better understand the sparseness of the Intersection Neighborhood version, as it is far more useful in terms of routing as it is (generally) a connected structure.

3.6 Pathological Cases for the Sparseness of the Localized Delaunay Triangulation

We now examine a pathological case of sparseness for the Delaunay Triangulation, this was also identified as an open problem by Kapoor and Li [23]. While in many cases the sparseness of the Localized Delaunay Triangulation over the Intersection Neighborhood may prove to be quite reasonable, we present a case where the number of edges is quadratic.

Theorem 7. *The 1-hop Localized Delaunay Triangulation over the Intersection Neighborhood may have* $\Omega(n^2)$ *edges.*

Proof. We first cluster a set of $n/2$ nodes on the y-axis equally spaced in the interval $[-\epsilon, \epsilon]$ and call these the *central cluster*. These nodes have a range equal to the largest range of nodes in the system, which we will show is equal to $3^{\frac{n-1}{4}} + \epsilon$. Now, on the x-axis we will place a node on either side, each at distance 1 from the center of the cluster. The range of the first node we place is $1+\epsilon$, we then place successive

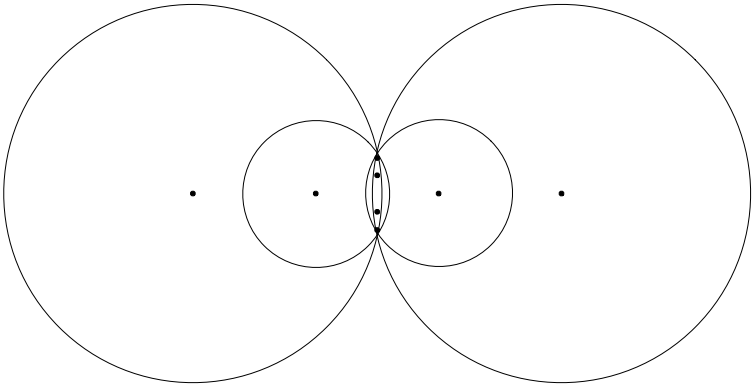

Figure 3.4: Circles illustrate the ranges of the various nodes. In the center, there is a line of $n/2$ nodes with an arbitrarily large range.

nodes on each side at distance 3^i from the origin with range $3^i + \epsilon$ so that it can reach the nodes in the cluster. Note that these nodes will not see any of the nodes between them and the y-axis. We continue in this manner until all remaining nodes have been placed. Since a quarter of the nodes will be used on either side, the largest range is then $3^{\frac{n-1}{4}} + \epsilon$.

A node u in the central cluster will be connected to a node v on the x-axis as the circle through u and v and tangent to the y-axis is empty of any nodes both can see. Therefore each outer node has $n/2$ edges for a total of $\frac{n^2}{4}$. Furthermore, the line of edges in the center adds an additional $n-1$ edges. This graph has $\Omega(n^2)$ edges. See Figure 3.5.

□

In practice, this structure may not be so dense. The way that this case has been constructed leaves open the possibility that the number of edges may be bounded by a function of l, the ratio between the largest and smallest range. Ideally, we

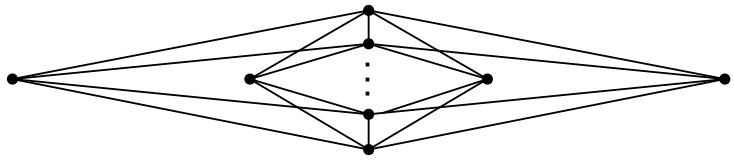

Figure 3.5: The set up is the same as that in Figure 3.4. Here we see that there are many edges, more than $n/2 \times n/2$

would like the sparseness to be strongly dependent on l. Certainly, the number of edges in the prior proof is dependent on l and gives a $\Omega(n \log l)$ number of edges, but this does not prove that no worse structure exists. Establishing the sparseness of a structure is often done by identifying illegal subgraphs as previously noted in section 2.5. Table 3.1 gives a few useful results concerning forbidden subgraphs.

Table 3.1: Forbidden Subgraphs and Sparseness

Forbidden Subgraph	**Number of Edges**
No $K_{3,3}$ or K_5	Graph is planar ($O(n)$) [26]
No Self-Intersecting P_3	$O(n \log n)$ [39]
No C_4	$\Theta(n^{3/2})$ [43]
No Self-Intersecting C_4	$O(n^{8/5})$ [42]

Showing that particular subgraphs are forbidden in the graph would establish a better bound on sparseness. Unfortunately, it is easy to realize these subgraphs. Figures 3.6 and 3.7 give examples of a P_3 and a C_4 respectively that can be formed and are self-intersecting. Furthermore they require such a small difference in the angles $\angle uvq$ and $\angle uvp$ (although those pictured have larger angles than are necessary) that Pinchasi and Smorodinksy's angle restriction is of little or no benefit. Also, the difference in the size of ranges is not particularly extreme, giving no real ability to

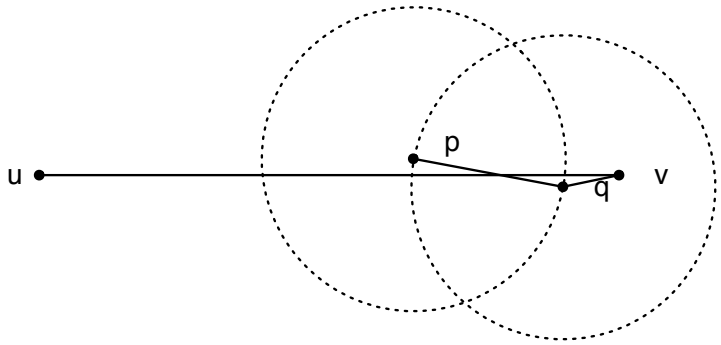

Figure 3.6: In this figure, nodes p and q are of unit range while v and u are of an arbitrarily large range, resulting in a self-intersecting P_3.

link sparseness to l as is our intention.

Similar results were found when constructing the $K_{3,3}$ graph. To use Kuratowski's Graph Planarity Criterion, one must show that a graph contains no $K_{3,3}$ or K_5 subgraphs (or homeomorphic subgraphs) to prove that it is planar. Unfortunately again, a $K_{3,3}$ graph can be constructed using only unit ranged nodes and nodes that are twice unit size. Figure 3.8 shows a construction that generates a 1-hop Localized Delaunay Triangulation over the Intersection Neighborhood that has $K_{3,3}$ as a subgraph. In the figure, the a's have a range of two, the closest of the two of the b's unit range, and the other b has range two. The b nodes are slightly to one side as to not be collinear with the a's. The Mutual Inclusion Graph of this arrangement connects each a node to each other and each b node to all of the a nodes. The Localized Delaunay Triangulation using the Intersection Neighborhood over this graph contains every edge of the MIG aside from some of the a to a edges. This graph will clearly contain $K_{3,3}$ as a subgraph, which means that the LDel(MIG) fails the Kuratowski's Graph Planarity Criterion and cannot be planar. Furthermore, the

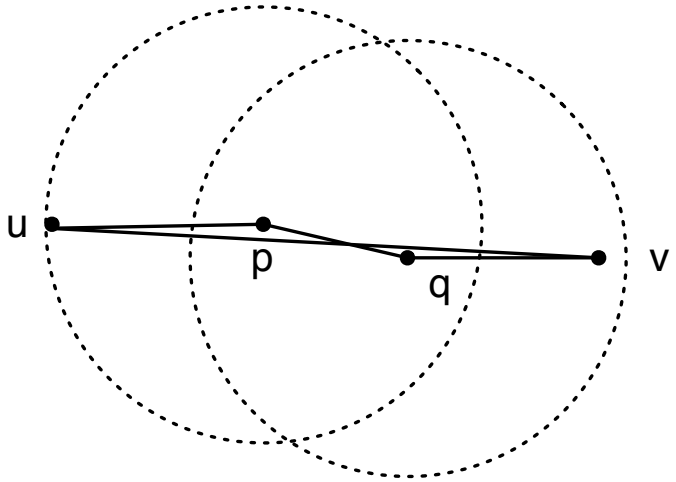

Figure 3.7: In this figure, nodes p and q are of unit range while v and u are of an arbitrarily large range. The nodes form a self intersecting C_4.

ranges of the angles of the edges is arbitrarily small and cannot benefit from Pinchasi and Smorodinksy's Theorem.

In the end, the results for sparseness are not good. It is possible to construct topologies where there are a quadratic number of edges. These seem dependent on the ratio between the largest and smallest ranges of nodes so the natural response is to use forbidden subgraphs to establish that there is a better bound on the number of edges. Unfortunately, we have found instances of the $LDel(MIG)$ that fail each of these criteria, and do so in a manner that important findings (such as those established by Pinchasi and Smorodinksy) were of no value. It is important to realize that, in practice, such graphs with such high densities may be quite rare. In this vein, Chapter 5 studies empirical results concerning the Localized Delaunay Triangulation.

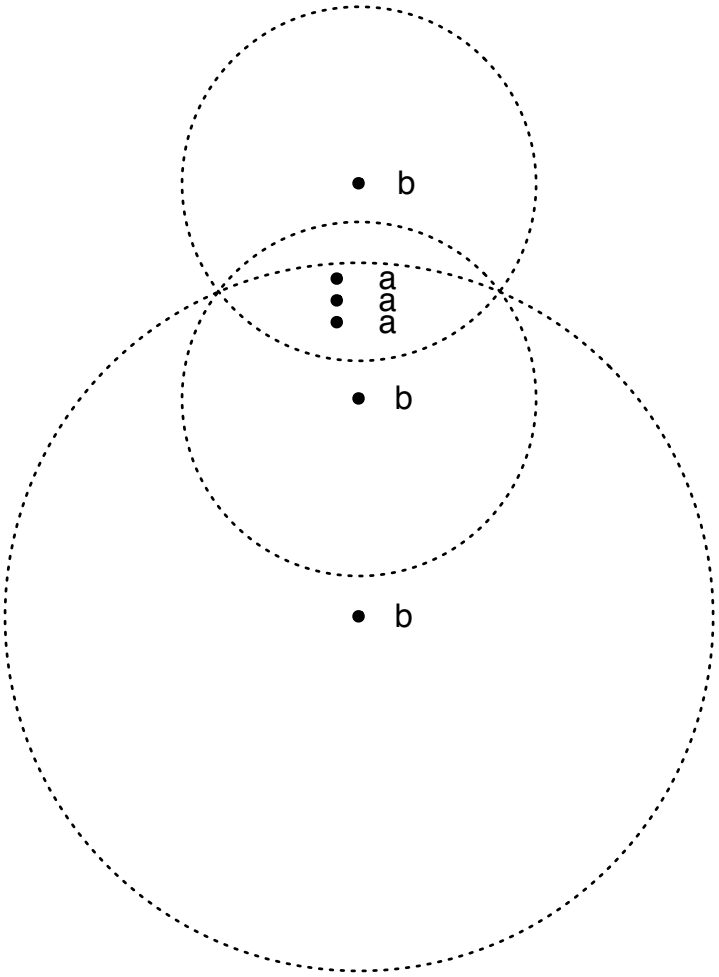

Figure 3.8: In a $K_{3,3}$ graph, there are two sets of nodes a and b such that every a is connected to every b with no other nodes. The as have a range of two, two of the bs unit range, and the other that has range two. The resulting Localized Delaunay Triangulation produces a graph with $K_{3,3}$ as a subgraph. Note that the empty circles defining the ab edges are tangent to the line through the as.

3.6.1 Intersections in the Localized Delaunay Triangulation 1-hop Delaunay Triangulation over the Intersection Neighborhood

We now briefly consider the number of possible intersections of edges in the 1-hop $LDel(MIG)$ over the IN.

Theorem 8. *It is possible to construct a Mutual Inclusion Graph G such that the 1-hop Localized Delaunay Triangulation will have $\Omega(n^2)$ edge intersections, where n is the number of nodes in G.*

Proof. We begin by placing two evenly spaced rows of unit ranged nodes, one row above the other, such that each node is visible by every other node in its own row and the opposite row. Each row contains $n/4$ nodes. Given this, the 1-hop Localized Delaunay Triangulation over the Intersection Neighborhood forms a strip of $(n/2)-2$ triangles for $\Omega(n)$ edges, $n/2-1$ of them being vertical (see figure 3.9).

Just outside of the range of this structure, on both the left and right sides, we place a column of $n/4$ nodes, evenly placed and with a great enough range such that they all can see those in their row and those in the other. We place them such that the bottom node in each column is above the bottom row of nodes in the center and the top node below the top row. Again, this will have $(n/2)-2$ triangles for $\Omega(n)$ edges, $n/2-1$ of them being horizontal. Because of their configuration, a total $n/2-1$ edges will collide with $n/2-1$ edges going the other direction for a total of $\Omega(n^2)$ intersections. □

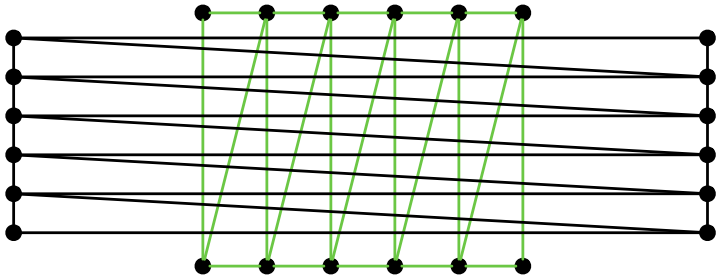

Figure 3.9: This diagram is not drawn to scale, for ease of legibility. Given a set of n nodes, it is possible to construct a structure with $\Omega(n^2)$ intersections in the 1-hop Localized Delaunay Triangulation over the IN. The central nodes are each of unit range, and are arranged such they are each able to see the other central nodes and no others. On the extreme left and right, the nodes have a range large enough to see all of the nodes on the left and right. They are placed outside of the range of those nodes in the center. In this manner, there are $\Omega(n^2)$ intersections.

It's worth noting that this proof does not require arbitrarily large ranges for nodes, unlike some of the other worst cases – in this example the ranges need be no larger than 3 times the unit length. Again, issues with this in practice are explored in Chapter 5.

3.7 Conclusions

We have explored some properties of the *LDel* over the Intersection Neighborhood version of the Localized Delaunay Triangulation. Specifically, we have explored the spanning ratios (stretch factor) of the structure as well as the sparseness of the graph. In both cases, the results that seem to indicate the topological properties are directly tied to l, the ratio between the largest and smallest ranges in the graph.

As shown, the spanning properties get worse with additional information in the system (i.e. more hops) as every additional hop allowed in the neighborhood makes the spanner worse. A stretch factor is a property of the number of hops, by limiting those hops the spanner improves. This is actually ideal, as it limits the work that every device has to do. So, while the n-hop $LDel$ over the Intersection Neighborhood failing to be a spanner is not a good result, the manner in which it is not a spanner is promising.

We conjecture that the 1-hop $LDel$ over the Intersection is as good a spanner as the DT. Furthermore, we know it cannot be a better spanner, for any graph realizable in the DT is realizable in the LDel.

Chapter 4

A Distributed Algorithm for the 1-Hop Localized Delaunay Triangulation

Regardless of what properties the Localized Delaunay Triangulation may or may not have, it is of little practical value if there exists no efficient algorithm to generate the structure. We explore a novel, distributed algorithm for the generation of the Localized Delaunay Triangulation.

4.1 Introduction

We wish to develop an algorithm to generate the 1-hop Localized Delaunay Triangulation over the Intersection Neighborhood. As input, we will take a set of points in the plane, each of which has a corresponding range. As output, we will give a list of edges. These are generated by each node in the network.

Since the routing is to be done in a distributed fashion, the generation of the LDel must also be done in a distributed manner. We first consider how difficult it is to construct the 1-hop LDel of the IN using naive, brute-force methods.

Lemma 3. *Given a set of n nodes S with associated ranges, the 1-hop Local Delaunay*

Triangulation of S over the Intersection Neighborhood can be generated locally in $O(n^2)$ time per node.

Proof. We first add edges to each node in order to generate the local portion of the MIG. To do so, we check to see if each pair of nodes has a connection, which takes $O(n^2)$ time. We do this to ensure that each node $s \in S$ (where S is the set of nodes in the plane) need only consider the nodes that it can connect to.

Since there can be no more than $n-1$ nodes connected to s in the MIG, s can just check each such edge $e(s,v)$ against the endpoints of each other such edge $e(s,u)$ to determine if $e(s,v)$ is Locally Delaunay. This node u must have a corresponding $e(s,u)$ and $e(v,u)$ when evaluating nodes in the Intersection Neighborhood. To determine if $e(s,v)$ is Locally Delaunay, we compute the maximum angle $\angle suv$ formed by a node u above and below the edge. If the sum of these two maximum angles is greater or equal to 180, no empty circle can be formed. If there does not exist an empty circle the edge is not Locally Delaunay, otherwise it is (see Figure 4.1). This can be done in $O(n^2)$ time. □

The estimate of $O(n^2)$ time for analyzing the nodes within the neighborhood is a somewhat pessimistic one. Consider a node running the algorithm. We refer to this node as the *hub*. For the hub to require $O(n^2)$ time in this stage, the nodes that surround it must all be able to see each other. If every node is able to see every other node that is within the hub's range, we could run a more traditional method, as developed for traditional Delaunay Triangulations. Classic methods, as detailed in Section 4.2, would require only $O(n \log n)$. In actuality, the time to generate the

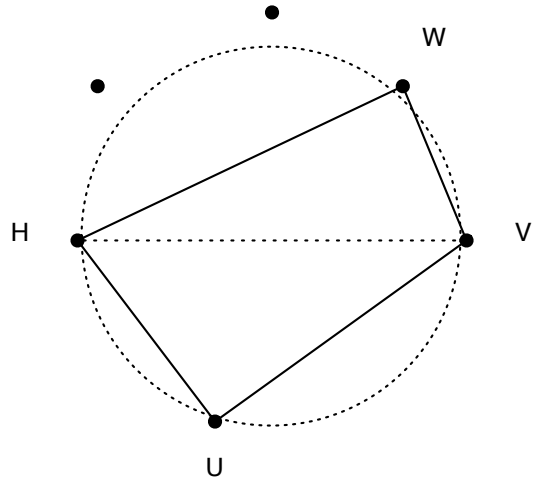

Figure 4.1: It is possible to determine if an edge is a Localized Delaunay edge by using the angles formed by other nodes. In this example H is the hub and V is some node that H can see and has a potential Localized Delaunay edge. As long the maximum angles "above" and "below" the edge add up to less than 180 degrees, there is a Localized Delaunay edge as there exists an empty circumcircle. In this examples, the points that formed the max angle above and below line HV are nodes U and W. However, since the sum of their angles is small enough, there still exists a Delaunay edge between H and V

edges after the MIG has been formed is no greater than n times the size of the largest Intersection Neighborhood between the hub and any node that it connects to. The size of this neighborhood can be denoted by m. Given that j is the number of nodes that the hub can see, generating the Delaunay edges, after the local portion of the MIG has been found, takes $O(jm)$ time where $0 \leq m \leq j-1$. This is an observation about the brute force algorithm and does not change its worst case running time as the process of generating the MIG takes $O(n^2)$ time. However, it is possible to refine the brute force algorithm slightly in this regard.

Geometric Range Searching is a well explored area of research [38] [4] [3] [2] and among its best understood subproblems is Circular Range Searching. The problem of finding all points lying inside some circle can be determined in $O(\log j + k)$ where k is the number of entities returned and j is the number of nodes that the hub can see, per query using a specialized data-structure of size $O(j \log j)$ [36] [6]. Once the hub node determines the other nodes that are in its range, each of these other nodes are checked to determine which can see the hub. Again, all points might be returned, but that is only a worse case scenario.

So creating the edges of the Mutual Inclusion Graph connected to the hub takes $O(j \log j)$ time to generate the search structure and then $O(j \log j + jm)$ time to find all the edges. Combined with the time for generating the triangulation edges, we have an algorithm that runs in $O(j \log j + jm)$ time, where $0 \leq m \leq j-1$ and $0 \leq j \leq n-1$. This is a better and more realistic bound on this slightly more intelligent version of the brute force algorithm. See algorithm 1 for more details.

Algorithm 1 FAST_BRUTE_FORCE_LDEL

(This algorithm is being run by a node $h \in J$, the hub.)
Input is J, a set of node locations (each node is covisible with h) with corresponding ranges.
Initialize range search structure S
Initialize Mutual Inclusion Graph G
Set of LDel edges, E
for all nodes j in J **do**
  Add j to S
  Add j to G
end for
for all nodes j in J **do**
  Query S to find nodes within j's range adding the Mutual Inclusion Graph edges to G
end for
for all nodes j in J **do**
  integer $a = 0$ (The max angle above the line hj)
  integer $b = 0$ (The max angle below the line hj)
  for all nodes v in G which share an edge with j **do**
    if v is above the line hj (where h is the hub) **then**
      if angle of hnj is greater than a **then**
        $a =$ angle of hnj
      end if
    else
      if angle of hnj is greater than b **then**
        $b =$ angle of hnj
      end if
    end if
  end for
  if $a + b < 180$ **then**
    add edge jh (where h is the hub) to E
  end if
end for

4.2 Algorithms for the Generation of the Delaunay Triangulation

The first real algorithm for the generation of the Delaunay Triangulation introduced the concept of "edge flipping." Some triangulation of the point set is first generated. Then, a test is done on every pair of adjacent triangles to determine if their shared edge is Delaunay. If it is not, this edge is "flipped" by removing it from the triangulation and adding the edge between the other pair of points in the quadrilateral that formed the hull for the adjacent triangles [46]. The running time is known to be $O(n^2)$.

Other algorithms have been developed that used random, incremental, or divide and conquer techniques, some of which have guaranteed running times faster than $O(n^2)$, although they have tended to be difficult to implement[1][20]. The relatively simple, sub-quadratic breakthrough was a novel $O(n \log n)$ algorithm developed by Steven Fortune [19], which introduced the concept of the plane sweep.

4.2.1 The Plane Sweep Method

The core idea of the plane sweep [20] [19] is fairly simple: we introduce a *sweepline*, which is a line that traverses the plane in some logical geometric order, eg. left to right or top to bottom or vice versa. As the sweepline progresses it encounters nodes

[1]These algorithms are, however, not without any application. They operate under any number of dimensions, while "edge flipping" and sweep techniques do not function in any space with more than two dimensions.

and performs some action or actions. Nodes are then considered only against nodes that the sweepline has already passed over, limiting the amount of data that must be analyzed at any given time.

This is not enough – if every node that the sweepline had so crossed was considered against *every* proceeding node, that would still require $O(n^2)$ time. Instead, the plane sweep must have some abstract concept of ignoring nodes that are no longer required for generating a structure. Fortune's algorithm maintains only the locus of points, the beachline that is equidistant between a node and the sweepline. Fortune's algorithm has a running time of $O(n \log n)$ due to its use of geometry and the use of the sweepline.

It is worth noting that plane sweeps of different shapes are also possible, although slight modifications need be made to the algorithm to accommodate them. Pavillet [40] gives an algorithm for the generation of Voronoi Diagrams that uses an expanding circular sweepcircle. Similar work was done by Adam et al. [1].

These methods don't seem to work for the 1-hop LDel of the IN, as they do not seem to be able to handle nodes having a range. Instead, we will need to use a novel approach.

4.3 A Distributed Algorithm for the Generation of the Localized Delaunay Triangulation

We have yet to present a sub-quadratic algorithm for the generation of the Localized Delaunay Triangulation over the 1-hop neighborhood. When trying to improve the

timing of the brute force algorithm (see algorithm 1), it was necessary to use more complex geometric algorithms such that less data needed to be analyzed at a given time. Fortune's Algorithm reduced what was being analyzed at any one given time to achieve a speed-up; this algorithm uses a similar idea to allow it to run in a sub-quadratic time.

4.3.1 A Skeleton of a Distributed Sweep Algorithm

We begin by providing a general description of how the algorithm will function.

Since the algorithm is run by each node, each node will have a chance to be the hub. A key observation is that the hub needs to only determine which Localized Delaunay Triangulation edges connect to it. So, the node is potentially responsible for $n - 1$ edges.

The brute force technique had, at its heart, a very simple method for determining if an edge was Delaunay or not. To increase its speed, it used Range Searching. In order to get a sub-quadratic running-time algorithm, we will again use Range Searching.

Much like the brute force version, the algorithm will have the hub h analyze each MIG neighbor in turn. For each such neighbor v, the algorithm will begin by dividing all other nodes into two sets: those above and below the line hv. Then the algorithm will determine which of those nodes are able to see v (and vice versa), eliminating those nodes which cannot. Finally, the algorithm uses a Voronoi polygon of the hub to find the node with the largest angle above and below v. If the angles summed are less than 180 degrees, then hv is a Local Delaunay Edge. We now examine the

components of the algorithm in more detail.

4.3.2 Partition Trees and Multi-Level Partition Trees

Given a set of points in Euclidean space we wish to be able to efficiently return a subset of those points inside some query region. This is, essentially, Geometric Range Searching, and numerous solutions to this problem exist. Suppose we wish to perform another query over the remaining set of points, then the problem becomes more difficult. This could be done again and again over the increasingly specific set of points returned each time. The solution that we will employ makes use of Partition Trees.

The points are all grouped into triangular regions called *canonical subsets*. The canonical subsets are formed using a *fine simplicial partition*, which means that each subset contains no more than $2 \times n/r$ nodes, where r is the number of subsets. If the line defining a halfplane region intersects a triangle, then we must continue further down that tree to determine the exact answer to the query. To do so, we need to recursively decompose those triangles into more triangles. In general one needs to further divide $\Omega(\sqrt{r})$ canonical subsets [16]. This collection of regions and sub-regions corresponds to the branches and subtrees of the Partition Tree.

Consider performing a halfplane query over some set of points that have been divided into canonical sets. When a halfplane is cast across this space, it intersects some regions and completely encloses others. We know that every point in regions that are enclosed must be within that range query, but we also know that some of the points of the intersected regions may be. We perform further decomposition within

the intersected regions (by continuing down the partition tree) to determine which points lie in the query half-plane (see Figure 4.3). Partition trees can be constructed with as little as $O(n)$ space, although query times can be improved by using more space. For halfplanes, one needs $O(n^{1+e})$ time to build the data structure, where n is the number of nodes in the plane. A half-plane query can be answered in $O(\sqrt{n})$ time [37] [6].

Suppose we wish to perform queries using a range space that is something other than a half-plane. For our purposes, we are interested in doing searches involving circles. It is possible to generalize the half-plane algorithm to solve circles, resulting in an $O(\sqrt{n})$ query time [38].

The inverse of the circle query range search is the *intersection search*. In an intersection search (with circles or discs) there exists n discs; when a query is performed on this space with a given point p, all discs that contain p are returned. See figure 4.2. The intersection search can also be done using partition trees. To do so requires linear space, and $O(n^{2/3})$ query time[2], where n is the number of discs [3].

A Partition Tree is, then, a balanced m-ary tree composed of regions that are recursively subdivided and for which solutions are pre-computed so that queries can be performed quickly. That said, the tree has only returned those subsets that passed the initial query. Suppose each of these subtrees were used as a root of a new tree, each one constructed to allow a different type of query – such that each subtree was projected into another kind of search tree. That is, each subset of nodes as defined by

[2]Aggarwal et al. [5] present another algorithm that gives an output sensitive $O(\log n + k)$ timing for this problem using a different, non-partition tree algorithm. This timing can be reached using partition trees as well, although it requires using slightly more space than what we have been using.

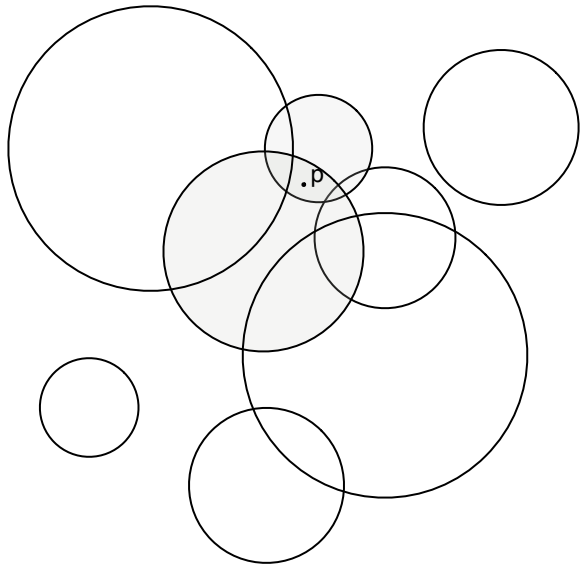

Figure 4.2: Consider an arrangement of disks in the unit plane. Given a point p we wish to return all disks which contain it. This is shown here, with shaded disks representing the result of the query point p.

the query are decomposed by some new scheme in this new tree. Importantly, there need not be any similarity between the types of searches that are used to compose the tree. Adding additional levels of trees is relatively inexpensive – each new level in the Multilevel Partition Tree adds a $O(\text{polylog}\, n)$ factor [3] to the construction time, where n is the number of nodes in the base tree.

4.3.3 A Distributed Algorithm for the Generation of the Localized Delaunay Triangulation

We now detail and prove the algorithm for the generation of the Localized Delaunay Triangulation.

Lemma 4. *Given a hub h and the set V of nodes it is covisible with and a node*

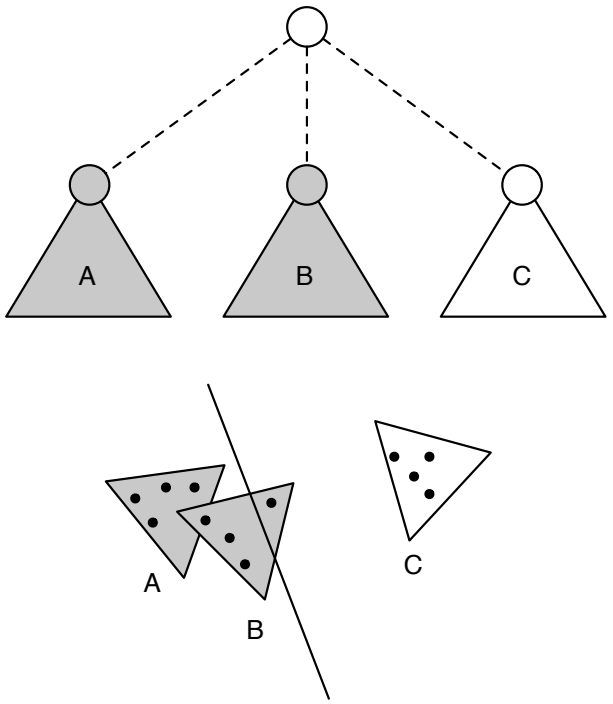

Figure 4.3: In the lower portion of the diagram is a set of points on the unit plane. These have been partitioned into 3 triangular regions, A, B, and C. These spaces have, in turn, been recursively decomposed into further triangles. This is encapsulated within an m-ary tree known as a Partition Tree. When a halfplane is cast across this space, it intersects B and completely encloses A. We know that every point in A must be within that range query, but we only know that some of the points of B might be. We perform further decomposition within B (by continuing down the partition tree) to determine which points.

$v \in V$, it is possible to find the nodes of V that are covisible to $v \in V$ and h above (and likewise, below) the line hv in $O(n^{2/3})$ time.

Proof. As in the brute force algorithm, we need to determine which nodes lie above a line hv and which lie below. Doing this allows us to determine which of these nodes forms maximum angle with hv of the nodes above and below hv. This, in turn, is used to determine whether a Localized Delaunay circle can fit in that space or not. To solve this problem, we will use a multilevel partition tree.

The problem of determining if points exist above or below a line is simply a half-plane problem. This is a standard search query for a constructed partition tree. Given n, the number of nodes that h can see, the time to construct this portion of the tree is $O(n^{1+e})$. The time for a half-plane query is $O(\sqrt{n})$. The space used is linear.

Two further searches then must be performed – above and below the halfplane. In either case after these points have been collected, we wish to exclude all points which are not covisible with v. To find covisibility, we first need to determine which nodes are within v's range, which can be done with a simple circle range query for the same cost as a half-plane query. We can add this functionality as an additional level of the Multi-level partition tree, which will add an additional polylog coefficient to the generation of the structure.

This establishes who can see v, but not which nodes have ranges that include v. This requires a disc intersection query. Again, this is an additional level of the partition tree, adding yet another polylog onto the overall construction time. To

form this section of the tree, the circles that are being used for the intersection test are the circles formed by the ranges of the nodes. The time for a disc intersection query is slightly greater than that for a range query; each query requires $O(n^{2/3})$ time [3].

Given a hub h and the set V of n nodes it is covisible with, it is possible to find the nodes that are covisible to $v \in V$ and h, and above (and likewise, below) the line hv using $O(n^{1+e}\text{polylog}\, n)$ time to construct the search data structure and then $O(n)$ queries of $O(\sqrt{n})$ time plus $O(n)$ range queries of $O(n^{2/3})$ time. □

The total timing, for all nodes, is then $O(n^{5/3})$.

The result of this portion of the algorithm is some number of the canonical subsets of the partition tree. An issue is the size of each canonical subset. Previously we described each canonical subset as being generated by a fine simplicial partition, which means that each cell contains no more than $2 \times n/r$ nodes, where r is the number of partitions. Since there are $\Omega(\sqrt{(n)})$ partitions, no $O(\sqrt{n})$ canonical subset contains more than $O(\sqrt{n})$ nodes. Clearly, between all the canonical subsets, there are only $O(n)$ nodes.

Lemma 5. *Given a hub h, a co-visible node v, and a set of canonical subsets S (containing, in total, $O(n)$ nodes), by using Voronoi diagrams and rayshooting, it is possible to determine which node has the largest angle with line vh in $O(\sqrt{n}\log\sqrt{n})$. time.*

Proof. As in the brute force algorithm, our goal is to find the node that has the largest angle with line hv. By finding such a node on both sides of the line we are

able to determine if there exists enough room to place an empty circle with both v and h on its perimeter.

First, we need to determine which nodes are covisible by h and v, either above or below that line. The previous lemma gives an algorithm that performs this action quickly – using $O(n^{5/3})$ time and $O(n)$ space. The result is some collection of triangular sets containing the points, known as canonical sets. Each set has some small number of points and the sets are disjoint. The point which forms the largest angle with hv could appear in any of the sets.

The possible returned subsets is a number far smaller than n (at most $O(\sqrt{n})$), so it is not unreasonable to analyze each set separately. However, the total number of nodes within the sets is $O(n)$, so we must examine each canonical subset in an efficient manner. In each set we wish to find the node with the maximum angle with relation to line hv, keep track of that angle, and determine if there exists a node with a larger angle in any other set. In this manner, one pass through the canonical sets is all that is required.

Let s be one canonical subset that lies above hv and let u be the point in s that forms the largest angle $\angle vuh$ with vh. The circle through v, u and h will be empty of points of s, and its center c will lie on the intersection of the line which bisects v and h, with the line which bisects u and h. In fact, the center c will lie on the portion of the bisector, of u and h which forms part of the Voronoi polygon, of h with respect to s. In order to locate c, and thus u, we will use Voronoi diagrams.

Recall the definition of the Voronoi Diagram (defintion 4 from chapter 2), the dual of the Delaunay triangulation. The Voronoi Diagram of a set of points S in

the plane is a subdivision of the plane into convex polygonal cells, each containing exactly one point of S. The cells (or regions) are constructed such that each point $p \in S$ is closer to any position within its corresponding region than is any other point of S.

What we will do then is include h as a member of every canonical subset when running a Voronoi Diagram generating algorithm over them (Fortune's algorithm, for example). From this, we will only use the Voronoi cell surronding h (this will be different for each subset). When the subsets are analyzed at the end, we have a small set of such cells that will each be analyzed independently. We will cast a ray along the bisector of hv. As we will show, it intersects the Voronoi cell around h, and where it intersects will determine what node has the largest angle with respect to line hv. See Figure 4.4.

To ensure that the ray intersects with the Voronoi polygon of h, we wish to first show that the ray will begin inside of h's Voronoi region. We assume no three points are co-linear. We also know that any nodes we are examining in s lie above line hv. This means, at some large distance, h must be closer to the bisector of hv than any other point of s. Since the ray being cast starts at infinite distance below hv, we know it must have originated within the cell surrounding h. We know, then, that the ray must intersect the Voronoi cell of h. Furthermore, we know whatever edge is first intersected is the bisector of h and a node u in s. We then know that u has the maximum angle with hv of any node in the canonical set. We can think of this in terms of the empty circle – u would be the node encountered first by a circle moving upwards that had h and v on its circumference. Each canonical subset is queried,

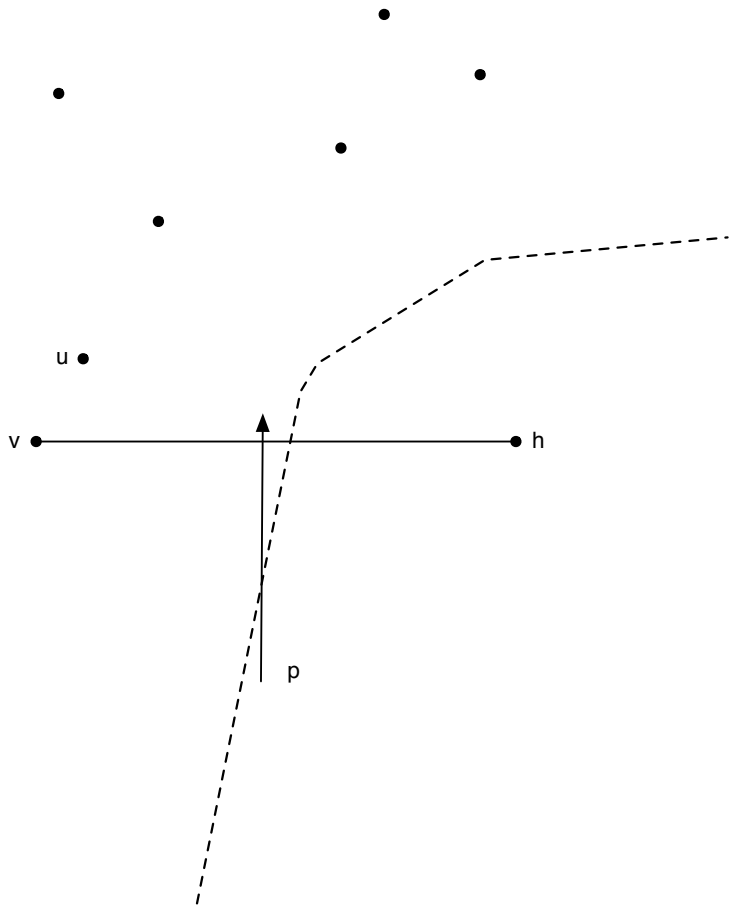

Figure 4.4: A Voronoi Diagram of the members of a canonical set (plus the hub, h) has been formed. A ray p is cast along the bisector of line hv intersecting an edge of the Voronoi polygon surrounding h. The node u is the node with the corresponding boundary, and thus has the max angle with hv.

and the node with the largest angle of those found by this method has the maximum angle over all. For each canonical subset, the time to generate the Voronoi Diagram is $O(\sqrt{n}\log\sqrt{n})$.

Ray shooting is another well studied area of computational geometry. The problem can be defined: given a set S of objects in $\mathbb{R}^d$, process them into a data structure such that when a ray is cast through the plane, the data structure can quickly report which object is hit. Conveniently, in this case, the problem does not require a complex data structure to solve. The edges of the Voronoi cell surrounding h can be sorted by the position of the left-most point of each edge, then placed into a list. The ray shooting then consists of querying the list and determining which edge is intersected by the line, using a binary search that requires $O(\log\sqrt{n})$ time, due to the fact that each canonical set can contain as many as $O(\sqrt{n})$ nodes. Each canonical subset will require one search structure – since there are as many as $O(\sqrt{n})$ canonical sets, these structures can be precomputed in $O(n\log\sqrt{n})$ time. Assuming this precomputation has been done, the worst case scenario is that every canonical subset has been returned, so the time required to check each of them is $O(\sqrt{n}\log\sqrt{n})$.

□

With these results, we are able to find the total timing of the algorithm.

Theorem 9. *Given a set of n nodes S, the Localized Delaunay Triangulation of S over the 1-hop Intersection Neighborhood can be generated in $O(n^{5/3})$ via a distributed algorithm.*

Proof. We already know that the time to construct the partition tree is $O(n^{1+e}\text{polylog}\, n)$.

We then need to assemble the ray shooting search structures for each canonical set. First we need to run Fortune's algorithm over the points requring $O(\sqrt{n}\log\sqrt{n})$ then using the resulting polygons build the lists of edges which again requires $O(\sqrt{n}\log\sqrt{n})$. Since there are $O(\sqrt{n})$ canonical sets, this results in $O(n\log\sqrt{n})$ time. This is all the pre-computing that is necessary – for each node v that it can see, the hub h will use those structures to determine if there exists an edge between h and v in the 1-hop Intersection Neighboor Localized Delaunay Triangulation. To find the maximum angle about hv, first we must query the partition tree, which takes $O(n^{2/3})$, then use ray shooting structures which requires at most $O(\sqrt{n}\log\sqrt{n})$. We repeat this for the bottom side of the edge hv. If the max angle between the top and bottom of the edge hv is not above 180 degrees, we report this as a Localized Delaunay Edge. This entire process must be repeated n times for a final timing of $O(n^{5/3})$. The algorithm is summarized in Algorithm 2. □

4.4 Conclusion

In this chapter we have presented two algorithms for the generation of the the Localized Delaunay Triangulation over the Intersection Neighborhood. These results answer open questions raised by Kapoor and Li [23]. As we noted earlier, the assumptions made were that nodes know their location (via something like GPS) and they are able to know what their range is. The Intersection Neighbourhood graphs may not function properly if the ranges are incorrect.

The two solutions to the Intersection Neighborhood version of this problem are

Algorithm 2 LOCAL_DELAUNAY_TRIANGULATION

Input is J, a set of node locations with corresponding ranges
Initialize a list of the LDel edges E
Construct a three-level partition tree P, first layer answering halfplane queries second layer answering circle range queries and a final layer answering circle intersection queries where each circle correspond to the range and location of a point in J
for all canonical subsets p in P **do**
  Generate an associated Voronoi Diagram over the set of points in p, plus the hub h.
  Using the Voronoi Diagram, create a list of edges to be used for ray shooting, R_p
end for
for all nodes j in J **do**
  integer $a = 0$ (The max angle above the line hj)
  integer $b = 0$ (The max angle below the line hj)
  Query the tree for all points above line hj that are covisible with j
  for all partition sets p returned by the query tree **do**
    Query R_p with a ray cast along the bisector of line hj returning node u
    if angle of huj is greater than a **then**
      $a =$ angle of huj
    end if
  end for
  Query the tree for all points below line hj that are covisible with j
  for all partition sets p returned by the query tree **do**
    Query R_p with a ray cast along the bisector of line hj returning node v
    if angle of hvj is greater than a **then**
      $b =$ angle of hvj
    end if
  end for
  if $a + b < 180$ **then**
    add edge jh (where h is the hub) to E
  end if
end for

very different, although they are based around the same central observation: measuring angles is a simple and accurate way to determine if there exists an empty circle. While the more efficient algorithm requires extensive use of range searching and complex data structures, this is to ensure that the process of the max angles does not take quadratic time. However, the savings is not exceptionally great, although it does prove that subquadratic algorithms are possible.

The brute force solution, in its refined form, is an algorithm that most likely will normally provide a good running time. It would be interesting to implement all of the above algorithms and compare running times, although it is outside of the scope of this thesis.

Whether the algorithm is used in practice or not, it presents hope that faster algorithms yet may be discovered. While linear times seem extremely unlikely, a $O(n \log n)$ algorithm may be discovered – the most important thing is the acknowledgment that sub-quadractic algorithms exist and could be implemented for this problem.

Chapter 5

Experimental Analysis of the Localized Delaunay Triangulation

5.1 Introduction

In Chapter 3, we gave a theoretical analysis of the 1-hop Localized Delaunay Triangulation over the Mutual Inclusion Neighborhood. What was found was a very negative result on sparseness (there can be $\Omega(n \log l)$ edges) and a rather unusual result in terms of spanning ($2.42 \times l$). While the spanning ratio may eventually be lowered, there is an example that achieves $\Omega(n \log l)$ edges. We also found that it was possible to find topologies that produced $\Omega(n^2)$ edge intersections, another negative result. However, all of these were generated by pathological cases that, in the case of spanning, may not even be realizable. The drive behind ad-hoc routing is not theoretical, and it may be that configurations of networks in the real world are extremely unlikely to be these disastrous cases. It seemed prudent then to analyze both the sparseness and spanning of networks experimentally, so that we could have a better idea of how real networks might function.

5.2 LocDel – A Localized Delaunay Triangulation Generator

To this end, a program, LocDel, was written to generate and analyze networks. The program, written entirely in Java 1.4.2, first generates a set of points placed via some programmed criterion and from this set the mutual inclusion graph is constructed. The program can then be told to generate the one-hop Localized Delaunay Triangulation over the Intersection Neighborhood[1], the regular (global) Delaunay Triangulation, and to find the respective spanning ratio of each. It can be set to collect other information as well, such as the largest intersection neighborhood of any pair of points, which is useful in analyzing the brute force algorithm for the generation of the Localized Delaunay Triangulation. See Figures 5.1, 5.2, 5.3 for screen shots of LocDel.

The program was written to be both a visualizer and to have a batch mode version, allowing it to run multiple trials, dumping the data in an analyzable form. The data was then processed using a PERL script and imported into Excel for analysis.

On a moderately fast machine (a 1 GHz G4 Macintosh with 768 MB of ram), generating and analyzing a single graph of 200 nodes can take a substantial amount of time. Since the interest was in the generation of as many graphs as possible (most of the various topology types had approximately 25,000 trials run), the raw compu-

[1]For the remainder of the chapter, it is assumed that any reference to the Localized Delaunay Triangulation is to the 1-hop LocDel(IN).

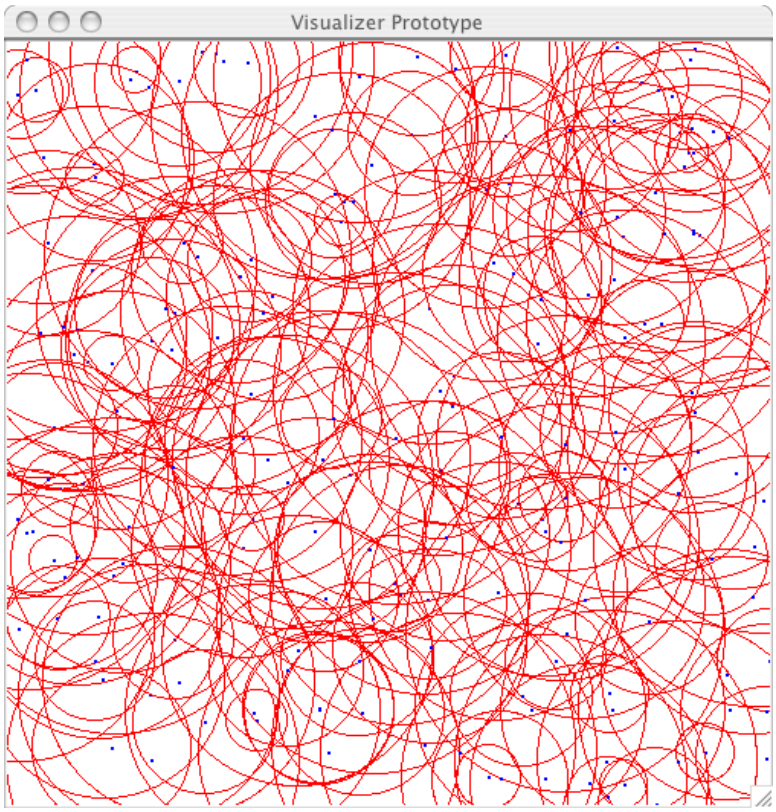

Figure 5.1: The simulator displays a Ad-Hoc Network. Here it displays the nodes and their associated ranges.

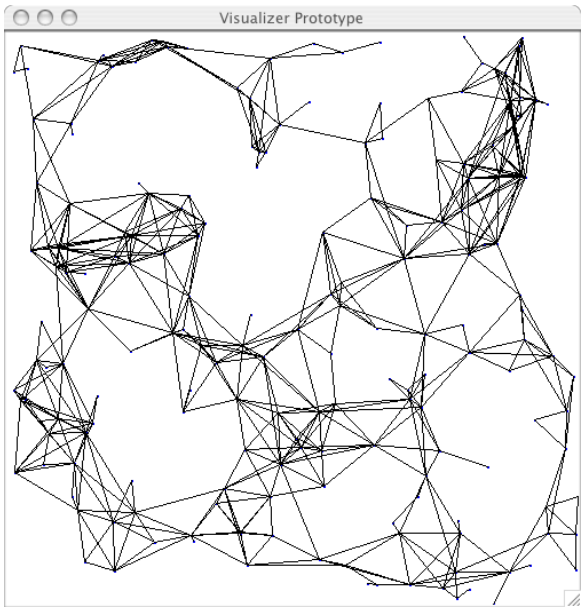

Figure 5.2: The simulator displays the MIG of the set of points shown in figure 5.1. Because of the limited range of many nodes, there are often dead zones within the graph, as can be seen.

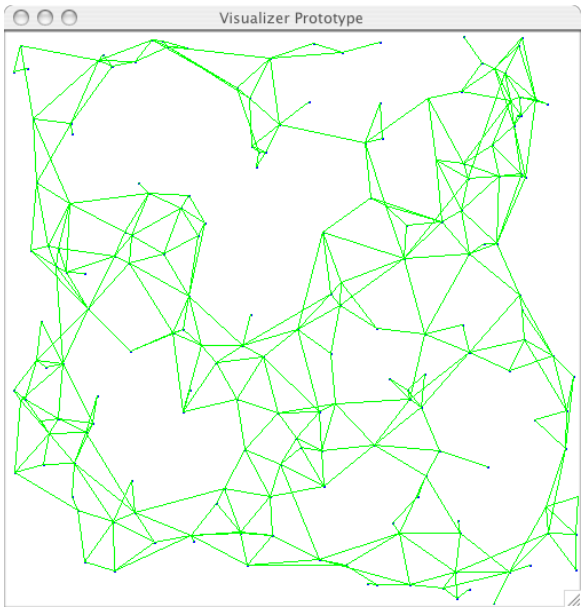

Figure 5.3: The simulator displays the Localized Delaunay Triangulation of the MIG of figure 5.2. As seen, it clears up the most congested sections of the network.

tation time was immense. To improve the speed at which this data was collected, a distributed approach was used via the PBS (Portable Batch System) Pro job management system. The various experiments were run, in a distributed fashion, across approximately 30 machines. This allowed for a much faster collection of data than what would have otherwise been possible.

5.3 Explored Topologies

The purpose of the experiments was to determine how the Localized Delaunay Triangulation behaved over different arrangements and weights of nodes. The goal was to have networks of various sizes, densities and layouts to give a better understanding of the structure. To this end, a few different topologies were selected.

5.3.1 Uniform Distribution

These tests were the most basic. 200 nodes were randomly placed on a planar region of size 70×70 units, each node having a range uniformly selected between 1 and 10 units. As this was the most general case, it was performed the largest number of times, 50,000 individual tests were performed. Uniformly distributed graphs have been studied by other Computational Geometers working in Ad-hoc Wireless networking [12]. See Figure 5.4.

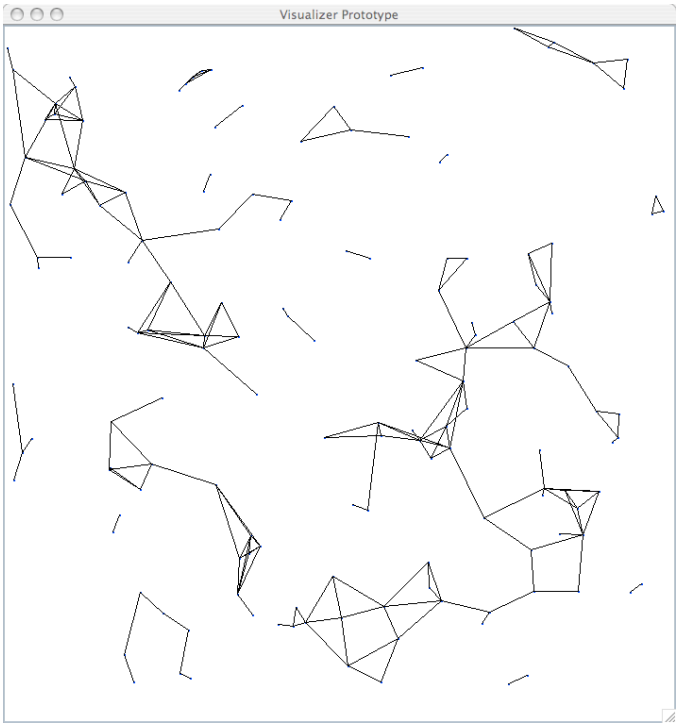

Figure 5.4: An example of a Mutual Inclusion Graph generated using the Uniform Distribution method.

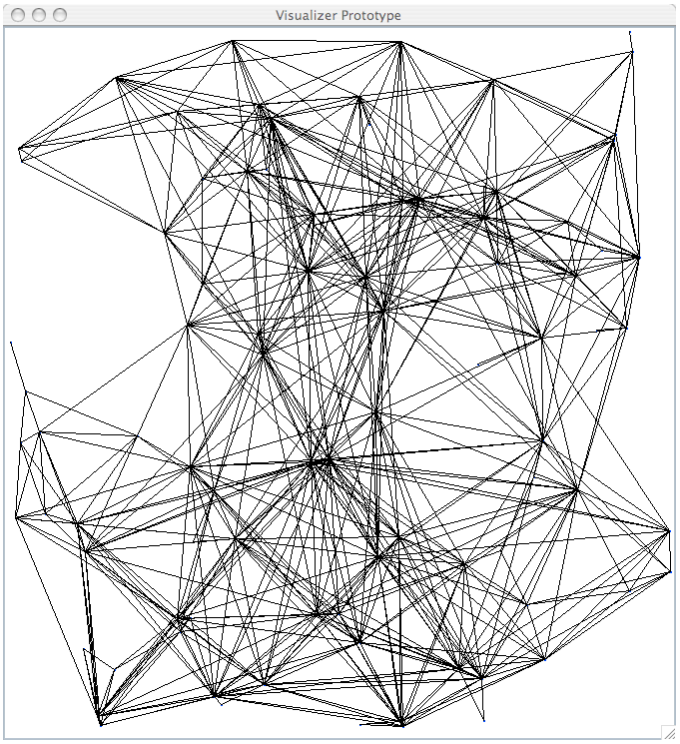

Figure 5.5: An example of a Mutual Inclusion Graph generated using the UD-LR method.

5.3.2 Uniform Distribution with Large Ranges (UD-LR)

The UD-LR tests used a uniform distribution as above, but the ranges for the nodes were uniformly selected from 1 to 40, instead of 1 to 10. Only 100 nodes were placed on a portion of the plane the plane of size 70×70. Since many of the results in the preceding chapters were bounded on l, the ratio between the largest and smallest ranges, there was reason to believe that the results would be different due to this. See Figure 5.5.

Figure 5.6: An example of a Mutual Inclusion Graph generated using the Dense Distribution method.

5.3.3 Dense Distribution

The uniform distribution leads to a very evenly spaced network. As a contrast, the Dense Distribution is based around a random walk, such that the termination point of each step is a new node and that each step is likely to be within range of the last node. More specifically: the first node is placed in the center of a 70×70 region of the plane. Every node has a range uniformly selected between 1 and 10. After the first node is placed, the next node is displaced an amount uniformly selected between 0 and 3.5 units away in each of the X and Y directions (if the node would leave the region of the plane, its move is discarded and replaced with a new one) from the previous node. This method results in a densely populated plane, most nodes being able to see a fair number of other nodes. As before, 200 nodes are placed into the plane. See Figure 5.6.

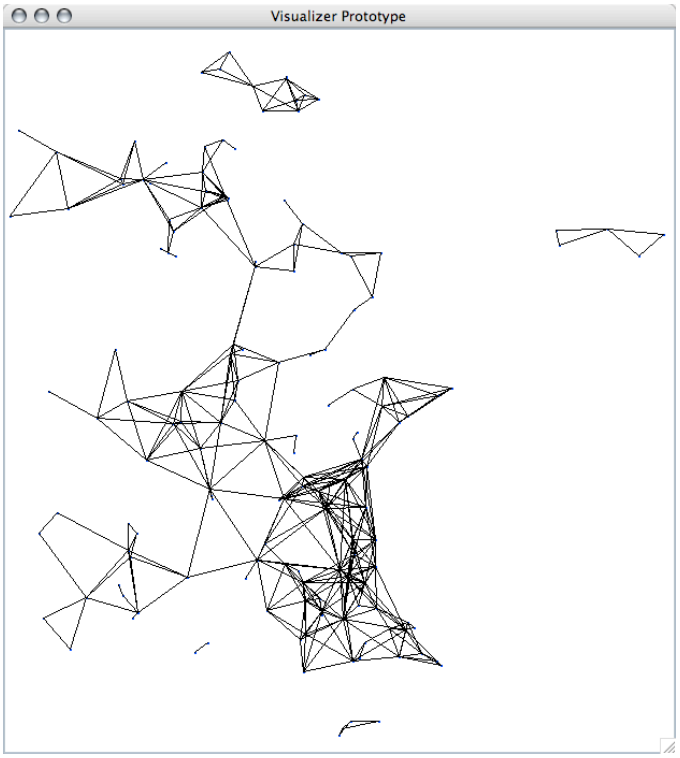

Figure 5.7: An example of a Mutual Inclusion Graph generated using the Wider Distribution method.

5.3.4 Wider Distribution

The Wider Distribution is similar to the Dense Distribution in that it is based on a random walk depositing nodes (with a range uniformly selected between 1 and 10) at each step. Again, 200 nodes are placed in this manner onto a 70×70 region of the plane. The difference is that the steps taken during the random walk have a distance uniformly selected between 0 and 10 units in both the X and Y directions. This leads to a wider spaced structure, but one that tends to be denser than the uniform distribution. See Figure 5.7.

5.4 Results

Given these topologies, it is now necessary to analyze the data that was collected from the experiments.

5.4.1 Spanning

As far as a structure's utility in Ad-Hoc Wireless routing, spanning is one of the most important properties. The Delaunay Triangulation is known to have a spanning ratio of just ≈ 2.42 and is conjectured to have a ratio of ≈ 1.6. In chapter 5, we showed that the 1-hop Localized Delaunay Triangulation of the IN is no worse than a $2.42 \times l$ spanner, where l is the ratio between the smallest and largest ranges. We also showed that the higher the number of hops that were allowed, the worse a spanner it became. The results of our experiments are summarized in Table 5.1.

The table shows a few important results. Average Span is the average spanning ratio calculated from each Localized Delaunay Triangulation of that distribution, the purpose is to give an idea of, generally speaking, how efficient a spanner the 1-hop LDel of the IN is. The second column shows what the max spanning ratio was for each distribution. The third column shows what the Standard Deviation of the spanning ratio was for each distribution. Finally, this is compared with the average spanning of the DT over each test's point set. The purpose is to use the normal DT as the benchmark for good spanning.

Table 5.1: Spanning Results

Spanning Results on the 1-hop LDel				
Distribution	**Average Span**	**Max Span**	**StdDev of Span**	**Average DT Span**
Uniform Distribution	1.24	1.45	0.06	1.36
UD-LR	1.34	1.54	0.03	1.35
Dense Distribution	1.36	1.53	0.03	1.37
Wider Distribution	1.32	1.53	0.04	1.36

Uniform Distribution

We see that the Uniform Distribution has a greater range of spanning ratios, but this is still quite small. Also, we see that given a uniform distribution it is unlikely to generate pathological cases for the placement of points – we know that spanning ratio can be at least as high as $\frac{\pi}{2}$. There is quite a difference between the average spanning for the LDel and the DT. This is due, probably, to the fact that there are a fraction as many edges in the MIG.

UD-LR

Due to the huge ranges of the nodes in the UD-LR, many nodes have large neighborhoods, for this reason, the results between the DT and the LDel are very similar. Note that despite the spanning result given in Chapter 3, the spanning does not appear to be affected by the larger ratio between the smallest and largest range.

Dense Distribution

The Dense Distribution results closely mirror the Delaunay Triangulation results. This is due to the fact that the dense distribution is so packed that many nodes are able to reach a large number of other nodes. This is similar to the DT over this point set, as it is generated globally and is similar to every node having complete knowledge of its neighbors.

Wider Distribution

In terms of spanning, the Wider Distribution holds a position somewhere between the Uniform Distribution and the Dense Distribution. Again, even at its worst it is quite far from having a $\frac{\pi}{2}$ stretch factor.

Conclusion

In the more than 100,000 tests that were run, never did the Localized Delaunay Triangulation have a spanning ratio greater than $\frac{\pi}{2}$; the conjectured spanning ratio of the Delaunay Triangulation. This is a good result and confirms intuition that the LDel is a good spanner. We believe that the LDel has the same spanning bounds as the DT. We give two conjectures, the second being stronger than the first.

Conjecture 1. *The spanning ratio for the 1-hop LDel over the IN is no more than the spanning ratio for the DT.*

Conjecture 2. *The spanning ratio for the 1-hop LDel over the IN is no more than* $\frac{\pi}{2}$.

This is the conjectured stretch factor for the Delaunay Triangulation. []

It seems that, in practice, one can expect a stretch factor somewhere around 1.33, as most of the results taken from mildly dense to dense graphs fell into that range.

5.4.2 Sparseness

Sparseness can be an important metric – ideally a graph shouldn't have too many edges as more edges are difficult to route over. We previously found arrangements of nodes for which the 1-hop LDel over the IN has $\Omega(n \log l)$ edges, where n is the number of nodes. Along with this, we have shown that degree is important in getting reasonable running times for the brute force algorithm.

The results are summarized in Tables 5.2 and 5.3.

Table 5.2: Sparseness Results Part 1

Sparseness Of The MIG			
Distribution	**Av. No. MIG Edges**	**Av. Largest Neighbor-hood Size**	**Av. Neigh-borhood Size**
Uniform Distribution	243	6.31	1.53
UD-LR	650	28.08	11.97
Dense Distribution	1792	44.11	16.32
Wider Distribution	493	14.31	3.90

Table 5.2 shows how dense the MIG is in terms of edges. It also shows how dense the overall mean neighborhood of each node in the MIG was, averaged over all of the tests. The chart finally shows the average size of the maximum neighborhood from each MIG. These are useful in determining how much work one node will be required to do when finding which edges are locally Delaunay.

Table 5.3: Sparseness Results Part 2

Sparseness Of The LDel versus the DT					
Distribution	**Av. No. MIG Edges**	**Av. No. Nodes**	**Av. No. LDel Edges**	**Av. No. DT Edges**	**Av. No. Unique Edges**
Uniform Distribution	243	158	219	457	26.84
UD-LR	650	93	235	264	35.85
Dense Distribution	1792	195	509	571	66.05
Wider Distribution	493	172	334	500	58.71

The second table displays specific data contrasting the 1-hop Localized Delaunay Triangulation and the global Delaunay Triangulation. When the program LocDel is run, any node that shares no MIG edge with another is removed. The average number of retained nodes is displayed in the second column. The third column shows the average number of LDel edges over these nodes. The fourth column gives the average number of edges in the Delaunay Triangulation over the same set of points. The final column shows the average number of those edges which occur in the LDel but not in the DT.

Uniform Distribution

The Uniform Distribution is quite sparse and on average the MIG contains about 1.5 times more edges than nodes. Likewise the size of the neighborhoods tends to be very small. In such sparse graphs, the number of LDel edges is not considerably less than the MIG. While the vast majority of the edges in the LDel are from the DT, about 10 percent are unique to it. The DT is substantially larger than the LDel as most of the edges used by the DT don't exist in the MIG.

UD-LR

The UD-LR has a far denser graph than that of the Uniform Distribution. Despite having two-thirds as many nodes on average as the Uniform Distribution, it has a much larger MIG and much larger neighborhoods. Despite this, the neighborhoods are not massive – most containing a bit more than 10 percent of the nodes although some a bit more connected. Due to the size of neighborhoods and the number of nodes, the LDel and DT are very similar in size.

Dense Distribution

The Dense Distribution is again similar to the UD-LR, although less dense per node. Again, the number of LDel edges is very similar to the number of DT edges. Finally, there exists a similar number of unique edges when compared to the previous tests.

Wider Distribution

The Wider distribution shows some relation to both the Dense and Uniform Distributions. Most neighborhoods were more than twice as large as those found in the uniform distribution, but this number was still very small. The average worst case of around 14 neighbors was also very reasonable. The number of edges was significantly higher than in the Uniform Distribution as more nodes were connected. Even so, as with the Uniform distribution, the number of LDel edges was far surpassed by the number of Delaunay Edges, which indicates it is not extremely dense. It also has about $\frac{3}{5}$ as many edges, on average, as its MIG, which is also a good sign.

Interestingly, the number of unique edges is quite high.

Conclusions

Despite poor theoretical results, in practice the LDel is decently sparse, containing a number of edges that seems linearly proportional to the number of nodes in the graph. On average, no distribution yielded a number of edges substantially more than about 3 times the number of nodes, generally being slightly smaller than the DT over the same point set. All in all this is a positive result that seems to indicate the LDT is a structure suitable for routing algorithms.

5.4.3 Intersections

Like sparseness, the number of intersections is a measure of both how easy it is to route information as well as the possibility of loss due to conflicting signals being sent long distances and interfering with each other. In some sense, it is a measure of how convoluted the graph is. In Chapter 4 it was shown that the number of intersections can be $\Omega(n^2)$ where n is the number of nodes.

The results are summarized in Table 5.4.

The table displays the average number of intersections in the LDel, the maximum number in that given test, and the standard deviation.

Uniform Distribution

Given how sparse the Uniform Distribution is, the results are somewhat disappointing. Luckily, the standard deviation seems to imply that extremely bad results, such

Table 5.4: Intersection Results

Intersection Results for the LDT					
Distribution	**Av. No. MIG Edges**	**Av. No. Nodes**	**Av. No. Inter-sections**	**Max No. Inter-sections**	**Std. Dev.**
Uniform Distribution	243	158	12.82	43	5.43
UD-LR	650	93	33.77	111	13.52
Dense Distribution	1792	195	73.09	180	21.27
Wider Distribution	493	172	45.06	126	15.59

as 43 intersections, are fairly rare. As the UD is fairly sparse, ensuring the network is connected may require a greater degree of intersections than with other distributions.

UD-LR

Given that the example generated with the UD-LR contain at most 100 connected nodes, the results are poor. In the case of the maximum, there exist more intersections than nodes. The standard deviation is also quite large, which suggests a wide range of results with many intersections, especially given that on average there are $\frac{1}{3} \times n$ intersections, where n is the number of nodes.

Dense Distribution

Again, similar results to the UD-LR, although the worst cases are not quite as bad nor is the standard deviation quite as wide. The lower amount can be attributed to the fact that nodes in the Dense Distribution have a great deal of information about their network and thus can more closely resemble the DT, which is, of course,

intersection free. The number of intersections seems to be generally smaller than n.

Wider Distribution

As was perhaps suggested by the high number of unique edges, the Wider Distribution has a high number of intersections. Since there are no intersections in the DT the edges that cause intersections must be unique to the LDel. While substantially smaller than n (where n is the number of nodes in the graph) it is still a high enough number to be concerned about. Then again, the random walk method that generates the Wider Distribution is likely to have a graph that repeatedly crosses over itself, so it is perhaps not surprising that it generates so many intersections.

Conclusions

While not nearly as bad as the pathological cases, the results are not as encouraging as they could be. In denser structures, one can expect $\Omega(n)$ intersections and even the sparser structures yield a substantial number. This said, we know that some intersections may be inevitable if a subgraph of the MIG is to be connected. More research is needed to determine if the number of intersections is actually prohibitive to the use of the structure.

5.5 Conclusion

Often, one finds that something that was theoretically encouraging was not useful for practical purposes. Perhaps some small detail means that an expected running

time is much larger than anticipated, or that the amount of computation hidden by the order notation renders an algorithm nearly useless. The Localized Delaunay Triangulation instead presents the flip-side of this phenomenon. Here, the theoretical results were either hazy or discouraging, but in practice the structure is not that bad. The spanning properties are extremely good and may be similar or identical to that of the regular Delaunay Triangulation. The sparseness results are reasonable – regardless of range or placement, the number of edges were well within $O(n)$, where n is the number of nodes. The intersection results are poor. In the end, more questions are generated by these results, while the theoretical chapters seemed to shut the door on the Localized Delaunay Triangulation as a usable geometric structure, the experimental data suggests that it may make the basis for a reasonable routing scheme.

Chapter 6

Conclusion

The easy availability and obvious utility of wireless, often mobile, devices has meant a huge increase in the number of devices in the last few years. Users would like to be able to connect to networks, receive and share data an stay online as long as possible, both in terms of battery life and in their ability to connect to the network as long as they were within range. While it is true that if a user is unable to check their email because their battery was drained or because of poor routing schemes this may simply be a minor inconvenience, in less commercial applications lack of connectivity can be problematic. In an emergency rescue, a well maintained network can make a difference in saving lives. In a less dramatic situation, efficient communication networks can simply save resources and prolong the life of the devices used. The problem of connecting these devices together into some form of efficient network is that of Ad-Hoc Wireless Routing.

While much of earlier research in Ad-Hoc Wireless Routing had been done using networks where each node had a uniform range, this model was not particularly realistic. Structures that had seemed promising as a structure for routing may not be nearly as useful in an environment with heterogeneous ranges. A good candidate

for routing was the Localized Delaunay Triangulation, which had been shown to have good spanning in the Unit Distance Graph domain. At the end of their paper which introduced the Mutual Intersection Graph[23], Kapoor and Li present the following open problems:

1. Is $LDel^k(MIG)$ a spanner?

2. What are the tight bounds for the sparseness of $LDel^k$?

3. Is there an efficient algorithm to construct $LDel^k$, in general?

We are now able to answer, in part, these problems. The Localized Delaunay Triangulation of the Intersection Neighborhood is in general not a spanner, although if constrained to the 1-hop version, it is a spanner with a theoretical stretch factor of $2.42l$ (where l is the ratio between the smallest and largest range) and an experimental stretch factor of ≈ 1.6. Furthermore, the 1-hop LDel over the IN may have $\Omega(n^2)$ edges. While we have shown that there exists a distributed $O(n^{5/3})$ algorithm for the generation of the 1-hop LDel over the IN, it requires a great deal of data-structure-derived machinery in order for it to operate, and is not really an ideal solution. The order notation obscures a great deal of computing that must be performed.

The results are mixed. The theoretical results show a structure that seems ill suited to routing, while the experimental properties are substantially better. It is unclear if it will prove to be a valuable structure in routing.

There are definite problems that no routing structure, no matter how advanced, can overcome. One of these is that for many MIG graphs there exists no subgraph

free of crossings. It is unclear, however, how many crossings must be tolerated. Crossing are clearly not good, they convolute and complicate routing and can lead to loss of data, but they are unavoidable. An important area of future Ad-Hoc Routing research may lie in the field of intersection minimizing routing structures. We have shown that the Localized Delaunay Triangulation can produce graphs with $\Omega(n^2)$ intersections and even experimentally has a large number, although far less than n. It is also unclear how intersections can be minimized locally – many nodes may be unaware that they are broadcasting in this manner.

6.1 Future Work

We conjecture that the spanning ratio of the 1-hop LDel over the IN is no worse than that of the regular DT, but the theoretical bound we have presented is very far from that. Experimentally this appears to be the case. It may be possible to construct a similar structure that is almost as power efficient using similar techniques.

Li [30] has presented a Yao-Yao Graph [51] derived technique for the MIG that seems to be the first spanner based network design for routing. Perhaps, the reality of the situation is that new or hybrid structures must be developed – the MIG seems to create topologies unlike anything previously seen in computational geometry. The solutions to some of these difficulties may lie in more novel techniques than mapping classical geometric structures to this domain.

The strangeness of the MIG also seems to be fertile ground for machine generated (or aided) theorems via constraints. As computer generated proofs seem to be an

increasingly important field, it will be interesting to see if any crossover occurs. Certainly computers have proven useful is generating experimental data with regards to the MIG.

Geometry may merely be an aid in developing schemes which have, empirically, excellent general properties. In this manner, even if no variation of the $LDel$ ever proves to be a proper spanner, it may still be an excellent general method for routing with excellent spanning, aside from rare worst case scenarios. Regardless as to the actual utility of the $LDel$, computational geometry has established itself as a vital tool in ad-hoc wireless routing.

References

[1] B. Adam, P. Kauffmann, D. Schmitt, and J.-C. Spehner. An increasing-circle sweep-algorithm to construct the delaunay diagram in the plane. In *Proceedings of the Ninth Canadian Conference on Computational Geometry (CCCG'97)*, pages 268– 273, 1997.

[2] Pankaj K. Agarwal. Range searching. Technical Report CS-1996-05, Department of Computer Science, Duke University, May 21, 1996.

[3] Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In B. Chazelle, J. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 1–56. American Mathematical Society, 1998.

[4] Pankaj K. Agarwal and Jiří Matoušek. Dynamic half-space range reporting and its applications. *Algorithmica*, 13:325–345, 1995.

[5] Alok Aggarwal, Mark Hansen, and Tom Leighton. Solving query-retrieval problems by compacting voronoi diagrams. In *Proceedings of the 22nd annual ACM symposium on the Theory of Computation*, pages 331–340, 1990.

[6] N. Alon, D. Haussler, and G. Wöginger. Partitioning and geometric embedding of range spaces of finite vapnik-chervonekis dimension. In *Proceedings of the 3rd ACM Symposium on Computational Geometry*, pages 331–340, 1987.

[7] Kaled Alzoubi, Xiang-Yang Li, Yu Wang, Peng-Jun Wan, and Ophir Frieder. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(5):408–412, 2003.

[8] Franz Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. Technical report, Freie Universität Berlin, Germany, 1990.

[9] Béla Bollobás. *Modern Graph Theory*. Springer Verlag, 1998.

[10] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick. On the spanning ratio of gabriel graphs and beta-skeletons. In *Proceedings of the Latin American Theoretical Infocomatics (LATIN)*, pages 479–493, 2002.

[11] P. Bose, A. Mashehwari, G. Narasimhan, M. Smid, and N. Zeh. Approximating geometric bottleneck shortest paths. In *Proc. 20th Symp. Theoretical Aspects of Computer Science (STACS 2003)*, number 2607 in Lecture Notes in Computer Science, page 3849, 2003.

[12] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[13] E. Chavez, D. Dobrev, E. Kranakis, J. Opartny, L. Stacho, and J. Urrutia. Traverasl of a quasi-planar graph without using mark bits. *Journal of Interconnected Networks*, 5(4):395 – 408, 2004.

[14] Bernard Chazelle. Application challenges to computational geometry: Cg impact task force report. Technical Report TR-521-96, Department of Computer Science, Princeton, April 1996.

[15] P.L. Chew. There is a planar graph as good as the complete graph. In *Proceedings of the 2nd Symposium on Computation Geometry*, pages 564–567, 1986.

[16] Mark de Berg, Mark van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computation Geometry Algorithms and Applications*. Springer Verlag, 1997.

[17] B.N. Delaunay. Neue darstellung der geometrischen kristallographie. *Kristallographie*, 84:109–140, 1932.

[18] David Eppstein. Spanning trees and spanners. In Jörg-Rudiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, chapter 9, pages 425–461. Elsevier, 2000.

[19] Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[20] Steven Fortune. Voronoi diagrams and delaunay triangulations. In Ding-Zhu Du and Frank Hwang, editors, *Computing in Euclidean Geometry*. World Scientific, 1992.

[21] Michel Foucault. *Discipline & Punish: the Birth of the Prison*. Vintage, 1975.

[22] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Statistical Zoology*, 18:259–278, 1969.

[23] Sanjiv Kapoor and Xiang-Yang Li. Proximity structures for geometric graphs. In *Algorithms and Data Structures, 8th International Workshop, WADS 2003, Ottawa, Ontario, Canada, July 30 - August 1, 2003, Proceedings*, volume 2748 of *Lecture Notes in Computer Science*, pages 365–376. Springer, 2003.

[24] J. Mark Keil and Carl. A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete Computational Geometry*, 7:13–28, 1992.

[25] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proc. 22nd ACM Int. Symposium on the Principles of Distributed Computing (PODC)*, pages 63–72, 2003.

[26] K Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamental Mathematics*, 15:271–283, 1930.

[27] Xiang-Yang Li. Algorithmic, geometric and graph issues in wireless networks. *Journals of Wireless Communications and Mobile Computing (WCMC)*, 3(2):119–140, 2003.

[28] Xiang-Yang Li. Applications of computational geometry in wireless ad hoc networks. In XiuZhen Cheng, Xiao Huang, and Ding-Zhu Du, editors, *Ad Hoc Wireless Networking*, pages 1–68. Kluwer, 2003.

[29] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wan. Distributed construction of planar spanner and routing for ad hoc wireless networks. In *Proceedings IEEE INFOCOM 2002, The 21st Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA*. IEEE, 2002.

[30] Xiang-Yang Li, Wen-Zhan Song, and Yu Wang. Efficient topology control for wireless ad hoc networks with non-uniform transmission ranges. *ACM Wireless Networks*, 11(3), 2005.

[31] Xiang-Yang Li, Peng-Jun Wan, and Yu Wang. Power efficient and sparse spanner wireless ad hoc networks. In *IEEE International Conference on Computer Communications and Networks (ICCCN01)*, pages 564–567. IEEE, 2001.

[32] Xiang-Yang Li and Yu Wang. Efficient construction of low weight bounded degree planar spanner. In T. Warnow and B. Zhu, editors, *COCOON 2003*, volume 2697 of *Lecture Notes in Computer Science*, pages 374–384. Springer, 2003.

[33] J. Macker and M. Corson. Mobile ad hoc networking and the ietf. *ACM Mobile Computing and Communications Review*, 2(1):9–14, 1998.

[34] Yury Makarychev. A short proof of kuratowski's graph planarity criterion. *Journal Of Graph Theory*, 25:129–131, 1997.

[35] Mahesh K. Marina and Samir R. Das. Routing performance in the presence of unidirectional links in multihop wireless networks. In *Proceedings of MOBIHOC'02, EPFL Lausanne, Switzerland*, pages 12–23, 2002.

[36] Jiří Matoušek. Reporting points in half-spaces. In *Proceedings of the 32nd IEEE Symposium on Computer Science*, pages 207–215, 1991.

[37] Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10:157–182, 1993.

[38] Jiří Matoušek. Geometric range searching. *ACM Computing Surveys*, 26(4):422–461, 1994.

[39] J. Pach, R. Pinchasi, G. Tardos, and G. Tóth. Geometric graphs with no self-intersecting path of length 3. In *Graph Drawing, 10th International Symposium (GD 2002), Revised Papers*, volume 2528 of *Lecture Notes in Computer Science*, pages 295–311. Springer, 2002.

[40] Axel Pavillet. Vorono diagrams in projective geometry and sweep circle algorithms for constructing circle-based vorono diagrams. In *Proceedings of the Thirteenth Canadian Conference on Computational Geometry (CCCG'01)*, pages 145–148, 2001.

[41] D. Peleg and A. A. Schaffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[42] Rom Pinchasi and Rados Radoicić. Topological graphs with no self-intersecting cycle of length 4. In *Proceedings of the 19th ACM Symposium on Computational Geometry, San Diego, CA, USA*, pages 98–103. ACM, 2003.

[43] Rom Pinchasi and Shakhar Smorodinksy. On the delaunay graph of a geometric graph. In *ACM Symposium on Computational Geometry - SoCG'2004, Brooklyn, NY*, Jun 2004.

[44] Rajmohan Rajaraman. Topology control and routing in ad hoc networks: A survey. *SIGACT News*, 33(2):60 – 73, 2002.

[45] Christian Schindelhauer, Klaus Volbert, and Martin Ziegler. Spanners, weak-spanners, and power spanners for wireless networks. Technical report, Heinz Nixdorf Institute, Panderborn University, Germany, 2004.

[46] R Sibson. Locally equiangular triangulations. *The Computer Journal Vol*, 2(3):243–245, 1975.

[47] Ivan Stojmenovic and Xu Lin. Power-aware localized routing in wireless networks. *IEEE Transactions of Parallel and Distributed Systems*, 12(11), November 2001.

[48] Godfried T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.

[49] M.G. Voronoi. Nouvelles applications des parametres continus a la theorie des formes quadratiques. *Math*, 134:198–287, 1908.

[50] Mark D. Watson. Ad-hoc routing and findings in one-two graphs. Appeared at the 2004 Graduate Symposium of the University Of Saskatchewan Computer Science Department, 2004.

[51] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.