On Path-Greedy Geometric Spanners

by

Lucca Morais de Arruda Siaudzionis

B. Com., University of British Columbia, 2020

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL
STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

September 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

On Path-Greedy Geometric Spanners

submitted by **Lucca Morais de Arruda Siaudzionis** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

William Evans, Professor, Computer Science, UBC
Supervisor

Bruce Shepherd, Professor, Computer Science, UBC
Supervisory Committee Member

Abstract

A t-spanner is a graph in which the shortest path between two vertices never exceeds t times the distance between the two nodes – a t-approximation of the complete graph. A geometric graph is one in which its vertices are points with defined coordinates and the edges correspond to line segments between them with a distance function, such as Euclidean distance. Geometric spanners are used to design networks of reduced complexity, optimizing metrics such as the planarity or degree of the graph.

One famous algorithm used to generate spanners is *path-greedy*, which scans pairs of points in non-decreasing order of distance and adds the edge between them unless the current set of added edges already connects them with a path that t-approximates the edge length. Graphs from this algorithm are called *path-greedy spanners*. This work analyzes properties of path-greedy geometric spanners under different conditions.

Specifically, we answer an open problem regarding the planarity and degree of path-greedy 5.19-spanners in convex point sets, and explore how the algorithm behaves under random tiebreaks for grid point sets. Lastly, we show a simple and efficient way to reduce the degree of a plane spanner by adding extra points.

Lay Summary

Consider a group of points. We want to connect some points to each other such that it is possible to reach any point from any other point within a factor of their distance. This is called a *spanner*. There are many algorithms that design networks such as these. We take one famous algorithm, *path-greedy*, and analyze its behaviour under some circumstances. In particular, we study the algorithm when the points are laid out in a convex shape or in a grid. Lastly, we study a way to improve the spanner with the addition of new points.

Preface

The thesis contains original and independent work by the author, Lucca Morais de Arruda Siaudzionis. Parts of Chapter 3 were published [W. Evans and L. Siaudzionis. Tight degree bounds for path-greedy 5.19-spanner on convex point sets. In *37th European Workshop on Computational Geometry*, pages 258–262, 2021]. My co-author in the paper is my thesis supervisor, William Evans, who helped with verifying some of the proofs published and with the overall writing. The rest of the work shown here is unpublished.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisor, Will Evans, for helping me in an uncertain time and guiding me through my research. I'm also thankful to the second reader of my thesis, Bruce Shepherd.

I must thank my parents, Felipe and Patrícia, my grandparents, Aldemir, Aída, Alberto, and Luiziane, my brothers, Felipe Filho and Pedro, and Elizângela for their presence over my entire life.

I would also like to thank friends with whom discussions made me dive deeper into compelling problems: Victor Sales, David Zheng, Daniel Du, Paul Liu, Daniel Lu, Nasa Rouf, Jack Spalding-Jamieson, Richard Cai, Rubens Bezerra, and many others.

Finally, I thank Bea Subion for being my daily source of inspiration, motivation, and for building the desk on which most of this thesis was written.

Chapter 1

Introduction

To the worm who first gnawed on the cold flesh of my corpse, I dedicate with fond remembrance these Posthumous Memoirs.
— Machado de Assis (1881)

The study of *spanners* stems from the desire to optimize some metric of a graph at the expense of others, while maintaining the graph's connectivity. That simple idea has been familiar for a long time in graph theory as spanning subgraphs. Spanning trees, for example, are a subgraph that connects all the nodes of a graph using the minimal amount of edges – what it optimizes for – while possibly worsening the shortest path between nodes.

Usually, the graph we are concerned about while studying spanners is the complete graph, meaning each node is directly connected to every other node. This thesis will be concerned with complete *geometric* graphs, in which the nodes are defined by their position and the edges are weighted by the spatial distance between their endpoints. Hence, finding a spanner in this complete geometric graph means improving metrics we are interested in – such as planarity, degree, weight, or size – at the cost of dilating the straight line distance between two points. This spanner is called a *geometric spanner*. We say a graph is a t*-spanner* if the shortest path between every pair of points is at most t times their spatial distance. The metric for measuring the distances varies but, throughout this writing, the only one used will be the Euclidean.

More formally, we define:

Definition 1 (Geometric Graph)**.** *A graph G with vertex set S is geometric if the points are defined by their location in $\mathbb{R}^d$, and each edge has its weight equal to the Euclidean distance between its endpoints.*

Definition 2 (t-spanner)**.** *For $t > 1$, a geometric graph is a t-spanner if the shortest path between every pair of points is at most t times their Euclidean distance.*

Definition 3 (Dilation)**.** *The dilation (also known as stretch or spanning ratio) of a graph G is the smallest value t such that G is a t-spanner.*

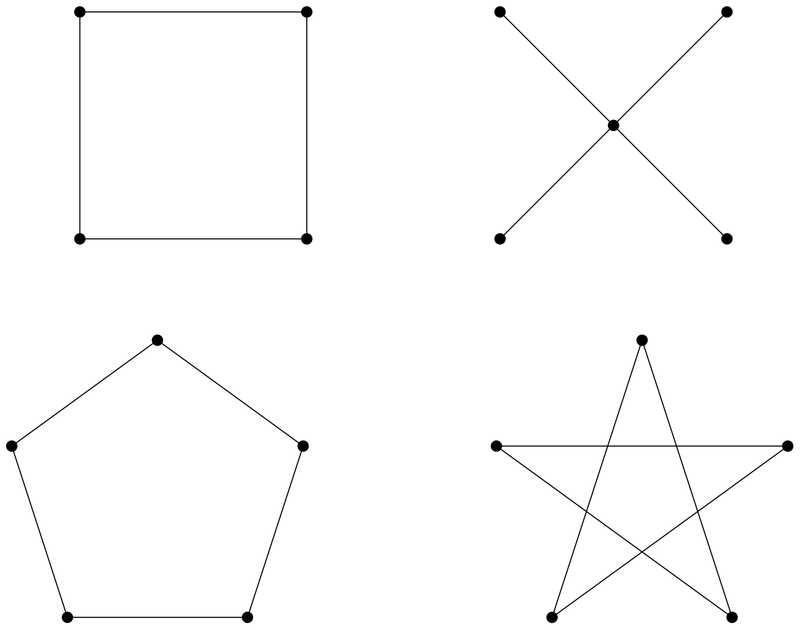

Figure 1.1: Different geometric spanners.

In Figure 1.1, the top two graphs are $\sqrt{2}$-spanners. The points are the vertices of a square of side 1, with the top-right image also containing the centre of the square. The bottom two point sets are the vertices of a regular pentagon of side 1. The bottom-left spanner has a dilation $\frac{2}{\phi}$ – where ϕ is the Golden ratio of $\frac{1+\sqrt{5}}{2}$ – and the bottom-right graph has a dilation 3ϕ.

1.1 Quality Measures

There is no consensus regarding what a good spanner is or what makes one better than the other. Instead, the relevant metrics vary according to the situation. When modeling real life networks, for example, we tend to be interested in graphs of low degree, as those facilitate distributed computations. However, rather than focusing on a single measure, it is usually preferable to optimize one of them while satisfying constraints on the others. For instance, we might be interested in finding the spanner with minimal dilation that is guaranteed to be plane.

There are four quality measures worth mentioning.

1.1.1 Edge Crossings and Planarity

For some cases in Euclidean graphs in $\mathbb{R}^2$, edge crossings are undesired. Hence, some spanner algorithms aim to minimize their occurrence. It is also frequently desired to reduce the number of edge crossings all the way down to zero, making the spanner be a plane graph. Chapter 3 will explore the planarity of the famous path-greedy algorithm in convex point sets, and Chapter 4 will analyze the same algorithm in grid point sets.

1.1.2 Dilation

It is trivial to generate a spanner graph with dilation 1, as the complete graph satisfies that. In practice, the requirements tend to lie in satisfying some useful and meaningful constraints while reducing the dilation as much as possible. One notorious example is minimizing the dilation of N points while using at most M edges, which happens to be NP-hard [15].

1.1.3 Degree

The degree of a spanner is defined as the maximum vertex degree amongst all of its vertices. When modeling networks, graphs of lower degree of preferred. There has been growing interest in spanners of specifically degree 3. Chapters 3 and 5 will be concerned with these.

1.1.4 Size (Number of Edges)

Note that bounding the graph's degree also limits the number of edges, but the inverse is not true. We can limit the number of edges in a spanner graph of N vertices by $N-1$ by simply choosing any spanning tree. Thus, we tend to optimize the number of edges while also satisfying other constraints, such as guaranteeing a maximum dilation.

1.1.5 Weight

The weight of a graph is the sum of all its edge weights. The minimal value is achieved by the minimum spanning tree (MST). Thus, the weight of a spanner is often compared to that of the graph's MST, and a spanner of low weight is called *light*. Many algorithms are considered light in this regard if the weight of their spanners is in $O(MST)$. There are many compelling problems related to a spanner's weight, none of which will be present in this thesis.

1.2 Motivation

As explained above, there are many separate dimensions that make a spanner good, and hence designing spanners is task dependent. One main motivation for the use of spanners is based on the fact that telecommunication networks can be modeled as geometric graphs, and thus designing efficient spanners help achieve more efficient networks. A graph with smaller degree can minimize the cost of the network since low degree nodes represent routers with few communication queues. A smaller degree also minimizes the speed of transmission since each router will have fewer choices for the next possible hop, making that step faster. For physically realized edges, it might be also be important to prevent crossings, reaching a plane graph. Since the metrics above that are relevant for this problem are the degree and planarity of the graph, this work will focus especially on those two aspects.

Another general application of spanners is for proximity problems, as the shortest paths between points are embedded in the very definition of a t-spanner.

1.3 Overview and Contribution

We will see the relevant background and related work for this thesis in Chapter 2. In Chapter 3, we will explore the topic that inspired this work, path-greedy spanners on convex point sets. This chapter also solves an open problem suggested by Bakhshesh and Farshi [3]. In Chapter 4, we explore how random tie breaking influences spanners in the specific case of path-greedy with points laid out in a grid. As far as I know, this is the first such investigation. Our work for reducing the degree of plane spanners using the addition of extra points is briefly explained in Chapter 5. Lastly, Chapter 6 contains our conclusion and suggestions for future work.

It is worth mentioning that much of the work of this thesis, including the search for counterexamples for some conjectures, relied on systematically building a library and tools that were crucial to achieve the outcomes presented. Appendix A explains that system in detail.

Chapter 2

Background

The best ideas are common property.
— Seneca (63 AD)

We will briefly cover the background that is relevant to the work presented, but this is only a drop in the ocean of interesting results related to spanners. For further documentation on algorithms, results, and open problems, one may refer to the textbook written by Narasimhan and Smid [16] or to the survey by Bose and Smid [7].

2.1 Famous Spanners

There are a few particularly prominent spanners in literature. These are not known for the specific point sets, but for the algorithms that generate them. Each of these algorithms enforces different properties which are desired in the final output. Thus, research on those algorithms tends to be around proving *other* properties that follow from each scenario.

We will briefly mention some famous algorithms and graphs.

2.1.1 Θ-Graphs and Yao Graphs

Both the Θ-Graph and the Yao Graph arise from the same idea: splitting the plane around each vertex into cones and connecting the vertex to the closest vertex within each cone. The two graphs differ in the way the closest vertex is defined, but they lead to similar outcomes. Figure 2.1 shows an example of a Yao Graph.

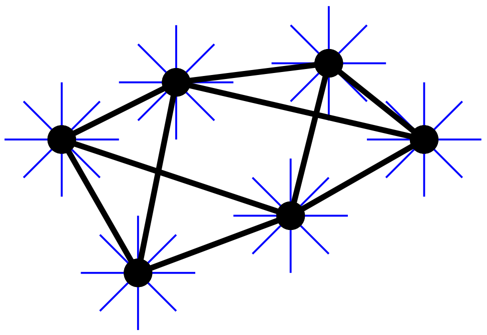
Figure 2.1: Example of a Yao Graph.

The low degree of the spanner comes from the construction, as each vertex will only be connected to at most as many cones as surround the vertex, which is a hyperparameter. An advantage of these two graphs is that each node can be processed independently, which facilitates a distributed implementation.

2.1.2 Well-Separated Pair Decomposition

The *well-separated pair decomposition* (WSPD) allows for construction of a t-spanner of N points with $O(N)$ edges and constant degree in $O(N \log N)$ time.

Two point sets A and B form a well-separated pair with regards to $s > 0$ if their bounding boxes are contained within two circles C_A and C_B of radius ρ that are at least $s\rho$ apart. (See Figure 2.2.) A WSPD of a point set with regards to s is a sequence of well-separated pairs $\{A_1, B_1\}$, $\{A_2, B_2\}$, $\ldots$, $\{A_m, B_m\}$ such that, for any two distinct points p and q, there exists exactly one index i such that $p \in A_i$ and $q \in B_i$ (or $p \in B_i$ and $q \in A_i$).

Though it is not trivial to see, a WSPD exists for every point set [16]. And, for $s > 4$, a t-spanner with $t = (s+4)/(s-4)$ can be built by taking one arbitrary edge for each pair of the decomposition. With a more sophisticated approach, it is possible to also bound the degree of each vertex [16].

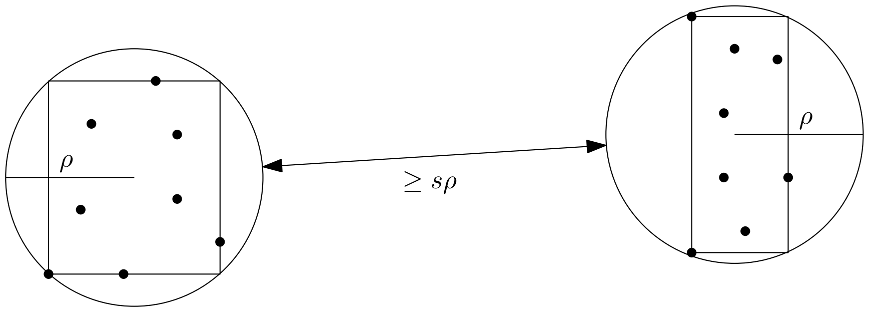

Figure 2.2: Example of a well-separated pair of point sets.

2.1.3 Delaunay Triangulation

A Delaunay triangulation of a set S is a triangulation such that no point p in S is inside the circumcircle of a triangle that does not include p. The immediate advantage of using Delaunay triangulations as spanners is that the spanners are guaranteed to be plane [11]. Other benefits are that they contain only a linear number of edges (since they are plane) and can be computed in $O(N \log N)$ expected time, though the interest in Delaunay triangulation goes way beyond spanners [11].

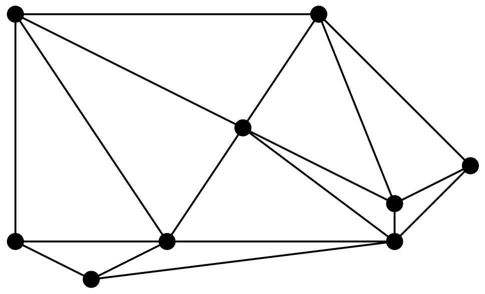

Figure 2.3: Example of a Delaunay triangulation.

The bounds on the dilation of the Euclidean Delaunay triangulation are not tight. It had long been conjectured that the dilation of the triangulation would never exceed $\pi/2$, but Bose et al. found a point set with a dilation greater than 1.5846, which is slightly above $\pi/2$ [8]. As an upper bound, Keil and Gutwin proved that the Delaunay triangulation is always a 2.42-spanner [14].

2.1.4 Path-Greedy Spanners

The *path-greedy algorithm* uses the dilation, t, as an input. Briefly speaking, the algorithm consists of processing every pair of points in the point set in increasing order of their distances, and, at each step, enforcing that those two points are connected by a path of length at most t times their distance. Spanners resulting from this algorithm are called *path-greedy spanners*.

The path-greedy algorithm will be extensively studied in this thesis, which brings us to the next section.

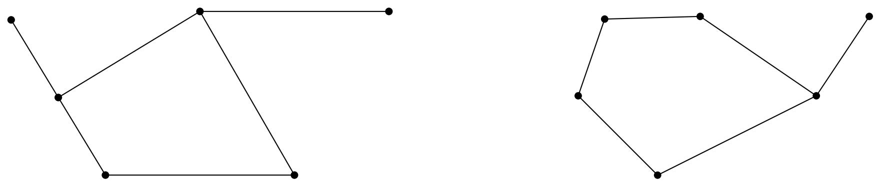

Figure 2.4: Two path-greedy spanners on convex point sets, with $t = 2$.

2.2 The Path-Greedy Algorithm

The path-greedy algorithm is a classic, simple way of constructing a geometric spanner for a point set. Originally introduced in 1993 by Althöfer et al. [1], the algorithm was designed to produce sparse spanners, both in weight and size of the graph. It is appreciated by many due to its incredible simplicity. Unlike most algorithms, path-greedy uses the desired dilation t as an input, and its output is guaranteed to be a t-spanner. Hence, analyses on the output spanner created by path-greedy differ according to the chosen t.

The path-greedy algorithm is a generalization of Kruskal's algorithm for minimum spanning trees. Algorithm 1 contains the original implementation of path-greedy. The output of the path-greedy algorithm is commonly called the *path-greedy spanner*. Figure 2.5 shows two path-greedy spanners on the same point set, with different dilations.

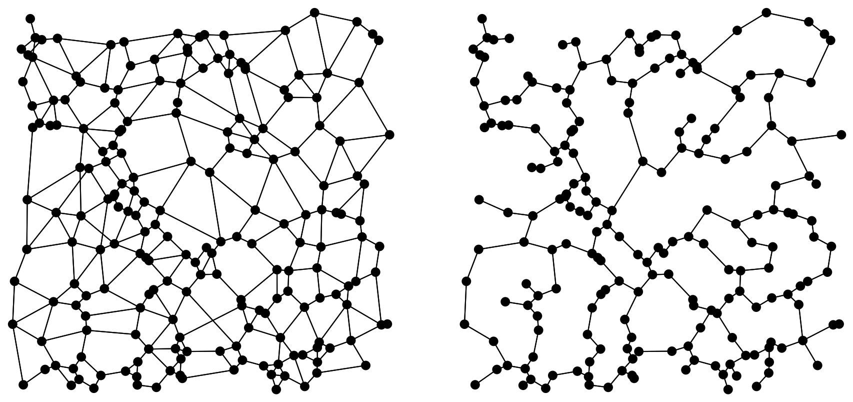

Figure 2.5: Path-greedy spanners with $t = 1.5$ (left) and $t = 5$ (right).

```
Algorithm 1: PATHGREEDY($S$, $t$)
    input  : A set of $S$ of $N$ points in $\mathbb{R}^d$ and a real number $t > 1$.
    output: $G(S,E)$, a $t$-spanner for $S$.
1   Sort all $\binom{n}{2}$ pairs of points from $S$ in non-decreasing order of their
      distances, breaking ties arbitrarily, and store them in list $L$;
2   $E \leftarrow \emptyset$;
3   $G \leftarrow (S,E)$;
4   for each $\{p,q\} \in L$ do
5       $\delta \leftarrow$ length of shortest path between $p$ and $q$ in $G$;
6       if $\delta > t|pq|$ then
7           $E \leftarrow E \cup \{(p,q)\}$;
8   return $G(S,E)$
```

The degree of each point in a path-greedy spanner is bounded by a constant, which means the entire graph will have a linear number of edges [16]. Thus, assuming a point set with n vertices, it would take $O(n \log n)$ time to compute the shortest path between two vertices using Dijkstra's algorithm. Hence, examining all $\binom{n}{2}$ pairs of points sequentially would lead to a $O(n^3 \log n)$ time algorithm.

There are more efficient algorithms to calculate the path-greedy spanner than simply naively following the original path-greedy algorithm. For example, Bose et al. [9] designed a $O(n^2 \log n)$ time algorithm and, more recently, Carmi and Tomer [10] introduced a different $O(n^2 \log n)$ implementation that is simpler. However, since our analyses will be concerned only with the algorithm's output – the actual path-greedy spanner – it does not matter which version we study. We will refer to the original algorithm when we consider the impact of equal length edges on the set of possible path-greedy spanners of a single point set. However, the other implementations would exhibit the same behavior.

2.3 Plane Spanners of Low Degree

Wireless networks can be modelled as geometric graphs in which the nodes in the point set reflect the geographical location of the antennas, devices, and other sources. A critical part of networking is routing, which determines how packets should be sent from a node to another, and a good routing would reduce delays or lags in the exchange of messages. The graphs that optimizes routing are plane and have low degree, which naturally lead to an interest in finding spanners satisfying those conditions [13, 17].

2.3.1 Convex Position

As a subproblem, we will consider only point sets that are in convex position. This allows us to obtain tight bounds on certain desired properties. There are three results around that problem that are worth mentioning.

First, Kanj et al. proved that the tight bound for the degree of a spanner with constant dilation is 3 in convex point sets, by designing an algorithm that guarantees a plane 20-spanner of degree 3 for the upper bound, and separately proving the lower bound with a construction [13]. Their algorithm also guarantees a maximum degree of 4 for points in general position.

Afterwards, Biniaz et al. improved that dilation by designing a matching-based algorithm for convex point sets [6]. On a high level, their algorithm considers the two convex chains connecting the diameter for the point set, chooses the closest edge to connect both chains, and recurses on each half. With that, they guaranteed a plane $\left(\frac{3+4\pi}{3}\right)$-spanner of degree 3 for points in convex position.

Lastly, Bakhshesh and Farshi also guaranteed a plane $\left(\frac{3+4\pi}{3}\right)$-spanner of degree

3 for convex point sets by using an adaptation of the path-greedy algorithm [3]. Their algorithm first adds all the edges around the convex hull of the point set, and then runs the path-greedy algorithm considering that those edges were already added. As a bonus, the upper bound for the weight of the spanner resulting from their algorithm is asymptotically equal to the total weight of the minimum spanning tree of the point set. Their work raised the question of whether the original path-greedy algorithm would maintain similar properties, which will be explored in Chapter 3.

2.4 Grid Spanners

Some research is focused specifically on points arranged in a grid. The grid can be *uniform* or *non-uniform*, depending on whether the the rows and columns are equally spaced apart. Figure 2.6 contains two examples of grid spanners.

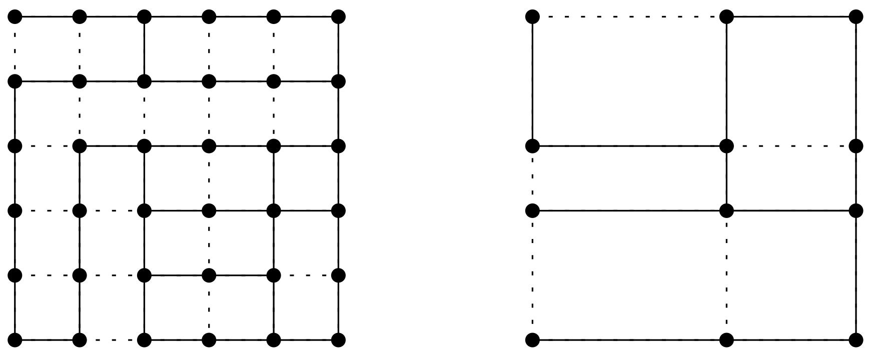

Figure 2.6: A uniform (left) and a non-uniform (right) grid spanner.

Though there is significant work related to grid spanners, not much of it is relevant to the rest of this thesis. An exception is the work by Biniaz et al., who designed a plane $3\sqrt{2}$-spanner of degree 3 for non-uniform grids [6]. (Note that it is trivial to find a plane $\sqrt{2}$-spanner of degree 4 by simply connecting the rows and columns in the grid.) We study the behaviour of uniform grid spanners created by the path-greedy algorithm in Chapter 4.

2.5 Steiner Points

The properties we are aiming for in a spanner may be impossible to achieve in certain point sets. In some situations, it may be advantageous to add extra points to the original set, such that all the points, combined, lead us to the results we are aiming for. These extra vertices are called *Steiner points*. More formally, we start with an original point set P, and transform it into a point set P', such that $P \subset P'$. The points in $P' \setminus P$ are called Steiner points.

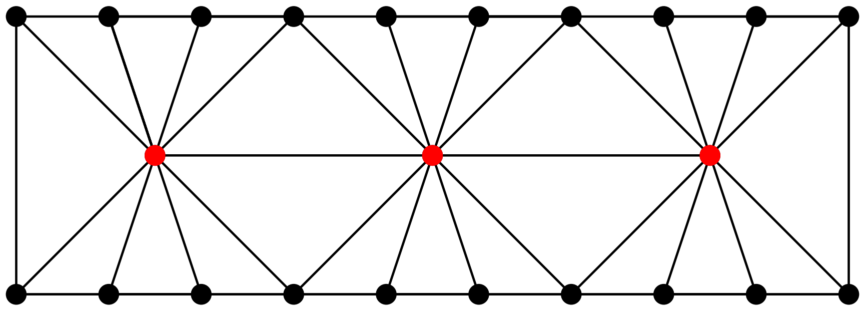

Figure 2.7: A simple triangulation that uses three Steiner points (in red).

One of the ways Steiner points can be used is to achieve planarity while reducing the dilation of the spanner. In fact, it is possible to add $O(N)$ Steiner points to a set P with N points such that a plane $(1+\varepsilon)$-spanner is reached, for any $\varepsilon > 0$ [2, 7].

Another valuable way to use Steiner points is to reduce the degree of a plane spanner. Biniaz et al. developed a way to transform a plane t-spanner of N points and arbitrary degree into a plane $(t+\varepsilon)$-spanner of degree 3, for any $\varepsilon > 0$, using only $O(N)$ Steiner points [6]. However, their construction only works perfectly for values $t > \pi + 1$. We will describe their construction in more detail and present an alternative construction that works for all $t > 1$ in Chapter 5.

Chapter 3

Convex Path-Greedy Spanners

No matter how many instances of white swans we may have observed,
this does not justify the conclusion that all swans are white.
— Karl Popper (1934)

In Section 2.3, we explained the growing interest in plane spanners of low degree. In this chapter, we are concerned with point sets that are in convex position. We explore the planarity and degree of spanners produced by of the path-greedy algorithm for convex point sets.

3.1 (No Longer) Open Problem

Bakhshesh and Farshi's algorithm creates, for points in convex position, a plane $\frac{3+4\pi}{3}$-spanner with maximum degree 3. Though their solution is not to simply use the path-greedy algorithm, it is quite close. In essence, their algorithm first adds to the spanner graph all the edges around the convex hull of the point set, and then runs path-greedy from there. (Technically, there are also optimizations that make it run faster, but the output does not change.) This could be interpreted as "seeding" path-greedy with an initial set of edges that are desired in the final output graph.

Their work leads to a natural question that they posed as an open problem: *Does* PATHGREEDY(S, $\frac{3+4\pi}{3}$) *construct a plane spanner of maximum degree 3 for points S in convex position?*

The answer, it turns out, is no. Recently, we showed that the maximum degree of PathGreedy(S, $\frac{3+4\pi}{3}$) for a convex point set S is 4, and the bound is tight [12]. That result will also be explained in this chapter. Further, we prove that PathGreedy(S, $\frac{3+4\pi}{3}$) is also not plane, showing a small example of a convex point set in which two edges of its path-greedy spanner intersect.

3.2 Tight Bound for the Maximum Degree of PathGreedy(S, $\frac{3+4\pi}{3}$)

We first prove that the maximum degree of a path-greedy t-spanner on a convex point set is at most 4 for $t \geq 4.27$. Afterwards we describe a convex point set S^* such that PathGreedy(S^*, $\frac{3+4\pi}{3}$) builds a degree 4 spanner. The construction can be generalized for all dilations greater than or equal to 4.27.

3.2.1 The Upper Bound

We start with two Lemmas, the first of which is trivial. The proof of the upper bound is shown immediately after.

Lemma 1. *Given two angles α and β such that $\beta \geq \alpha > 0$ and $\pi > \alpha + \beta \geq 3\pi/4$, we have* $\cos\alpha + \cos\beta \leq \sqrt{2-\sqrt{2}}$ □

Lemma 2. *Let $t \geq \frac{\sqrt{2}\cos(\pi/8)}{\sqrt{2}\cos(\pi/8)-1} = \frac{1}{1-\sqrt{2-\sqrt{2}}} \approx 4.262$. Given two angles α and β such that $\beta \geq \alpha > 0$ and $\pi > \alpha + \beta \geq 3\pi/4$:*

$$t\sin(\alpha+\beta) + \sin\alpha - t\sin\beta \leq 0$$

Proof. By the angle-sum identity for sin,

$$\begin{aligned} t\sin(\alpha+\beta) + \sin\alpha - t\sin\beta &= t\sin\alpha\cos\beta + t\sin\beta\cos\alpha + \sin\alpha - t\sin\beta \\ &= (t\cos\beta + 1)\sin\alpha - (t - t\cos\alpha)\sin\beta \\ &= \left(t\cos\beta + 1 - \frac{\sin\beta}{\sin\alpha}t(1-\cos\alpha)\right)\sin\alpha \\ &\leq (t\cos\beta + 1 - t(1-\cos\alpha))\sin\alpha \end{aligned}$$

$(\frac{\sin\beta}{\sin\alpha} \geq 1$ since $\pi - \alpha > \beta \geq \alpha)$

$$= (\cos\beta + \cos\alpha + 1/t - 1)t\sin\alpha$$

$$\leq \left(\sqrt{2-\sqrt{2}} + 1/t - 1\right)t$$

(by Lemma 1 and $0 < \sin\alpha < 1$ since $0 < \alpha < \pi/2$)

$$\leq \left(\sqrt{2-\sqrt{2}} + 1 - \sqrt{2-\sqrt{2}} - 1\right)t = 0$$

since $t \geq \frac{1}{1-\sqrt{2-\sqrt{2}}}$. □

Theorem 1. *Given a convex point set S and a dilation $t \geq 4.27$, the spanner graph generated by* PATHGREEDY(S, t) *has degree at most 4.*

Proof. Assume for a contradiction that vertex v has degree at least 5 in the spanner generated by PATHGREEDY(S, t). Let a, b, c, d, and e be five of the vertices that v is connected to. Assume without loss of generality that they are in clockwise order. Since the point set is convex, one of the angles $\angle avb$, $\angle bvc$, $\angle cvd$, and $\angle dve$ must be at most $\pi/4$. Assume without loss of generality that $\angle avb \leq \pi/4$.

Let $\alpha = \angle vab$ and $\beta = \angle vba$. Assume without loss of generality that $|va| \geq |vb|$ and thus that $\beta \geq \alpha$. If $|va| = |vb|$, assume the edge vb gets processed by path-greedy before edge va. Since $\angle avb \leq \pi/4$, we have $\beta \geq \alpha > 0$ and $\pi > \alpha + \beta \geq 3\pi/4$ and $|va| > |ab|$.

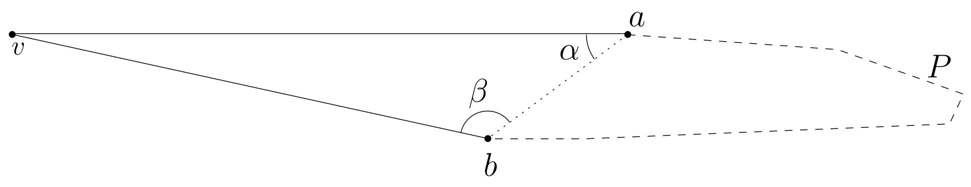

Figure 3.1: Illustration of Theorem 1's proof.

Note that, since $|va| > |ab|$, path-greedy will scan the edges ab and vb before the edge va. However, it is not possible for path-greedy to add all three edges ab, vb, and va, since $|ab| + |vb| < 2|va|$. (In other words, if the algorithm were to add the first two smaller edges scanned, ab and vb, it would not add the third, as the vertices v and a would be connected by a sufficiently short path.) Thus, since va

and *vb* are both added, path-greedy must skip the edge *ab*, which means that the vertices *a* and *b* must be connected by some path *P* such that:

$$|P| \leq t|ab| \tag{3.1}$$

By the time path-greedy scans edge *va*, the vertices *v* and *a* will be connected by the path *P* + *vb*. Since path-greedy still adds the edge, this means that:

$$t|va| < |P| + |vb| \tag{3.2}$$

Combining inequalities (3.1) and (3.2):

$$t|ab| + |vb| - t|va| > 0. \tag{3.3}$$

The sine law for Δabv, $\frac{|va|}{\sin\beta} = \frac{|vb|}{\sin\alpha} = \frac{|ab|}{\sin(\alpha+\beta)}$, implies

$$|va| = \frac{\sin\beta}{\sin(\alpha+\beta)}|ab| \quad \text{and} \tag{3.4}$$

$$|vb| = \frac{\sin\alpha}{\sin(\alpha+\beta)}|ab| \tag{3.5}$$

Substituting (3.4) and (3.5) into (3.3):

$$\begin{aligned} t|ab| + \frac{\sin\alpha}{\sin(\alpha+\beta)}|ab| - t\frac{\sin\beta}{\sin(\alpha+\beta)}|ab| > 0 & \qquad \Longrightarrow \\ t + \frac{\sin\alpha}{\sin(\alpha+\beta)} - t\frac{\sin\beta}{\sin(\alpha+\beta)} > 0 & \qquad \Longrightarrow \\ t\sin(\alpha+\beta) + \sin\alpha - t\sin\beta > 0 & \end{aligned} \tag{3.6}$$

which contradicts Lemma 2. Thus, it is not possible for a vertex *v* to be connected to five or more vertices, making the degree of the spanner graph at most 4. □

Since $\frac{3+4\pi}{3}$, which is approximately 5.19, is most definitely greater than 4.27, we conclude that the maximum degree of a path-greedy $\frac{3+4\pi}{3}$-spanner for convex point sets is at most 4.

3.2.2 The Lower Bound

Theorem 2. *There exists a convex point set S^* such that the spanner generated by* PATHGREEDY(S^*, $\frac{3+4\pi}{3}$) *has degree 4.*

Proof. We prove by example. There were two conditions we imposed for a point set to be a good example: (i) It should have no three collinear points; and (ii) its path-greedy $\frac{3+4\pi}{3}$-spanner should be the same regardless of how ties are broken amongst edge distances while the algorithm runs. Our example satisfies both of these conditions.

The points in S^* are shown in Table 3.1.

Table 3.1: Coordinates and labels of point set S^*.

Vertex	***x*-coordinate**	***y*-coordinate**
v	0	0
a	9.992	−0.399
a_1	16.983	−0.749
a_2	23.970	−1.169
a_3	27.861	−2.099
b	5.258	−8.506
b_1	12.253	−8.261
b_2	19.246	−7.946
b_3	26.235	−7.561
c	−4.825	−8.758
d	−10	0

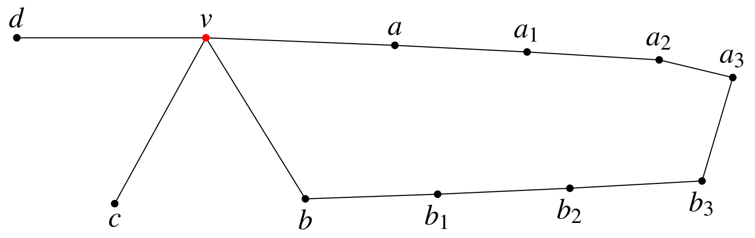

Figure 3.2: Spanner generated by PATHGREEDY(S^*, $\frac{3+4\pi}{3}$).

We can confirm that PATHGREEDY(S^*, $\frac{3+4\pi}{3}$) has degree 4 by running the algorithm. Figure 3.2 shows the spanner generated in this process. $\square$

Construction of S^*

The points shown were not chosen at random, of course. First, point v was fixed at the origin. Then, points a, b, c, d were chosen such that $|va| \approx |vb| \approx |vc| \approx |vd|$ (and all those lengths are approximately 10 in S^*, but that's not relevant). Angles $\angle cvd$ and $\angle bvc$ are both slightly above $\frac{\pi}{3}$, so that the edges dc and cb are the largest edges in the triangles Δvdc and Δvcb, and consequently scanned last amongst those by path-greedy. This will ensure the edges vb and vc are added by the algorithm. In order to ensure $\angle dva < \pi$, we make enforce that $\angle avb$ is slightly below $\frac{\pi}{3}$. This partial process is illustrated in Figure 3.3.

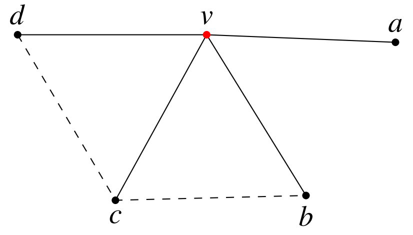

Figure 3.3: Partial process of building point set S^*.

Since we enforced $\angle avb < \frac{\pi}{3}$, the edge ab cannot be the longest edge in triangle Δvab, which means it must be scanned by path-greedy before va and vb are both added. Remember that the path-greedy algorithm will never add all three edges of a triangle for $t > 2$ and, since we want the edges va and vb to be added, we must enforce ab is not added to the output. Thus, path-greedy must decide not to add edge ab when scanning it, meaning the points a and b must be connected by a sufficiently small path P. Figure 3.4 shows this idea.

Now, we have a scenario similar to the one explored in the proof of Theorem 1. In order for the path P to lead to the graph we want, it must satisfy a few conditions itself: (i) all the edges within it must be small (smaller than $|ab|$), so that they are the first ones to be processed by path-greedy; (ii) the length of P is at most $t \cdot |ab|$, in order for the edge ab not to be added; and (iii) $|P| + |vb| > t|va|$ and $|P| + |va| > t|vb|$, so the edges va and vb are both added by the algorithm. Intuitively, assuming $|va| \approx |vb| \approx |ab| \approx k$ for some k, we would have the length of P between $(t-1)k$ and tk. This allows us to generalize this construction for any value of $t \geq 4.27$. In

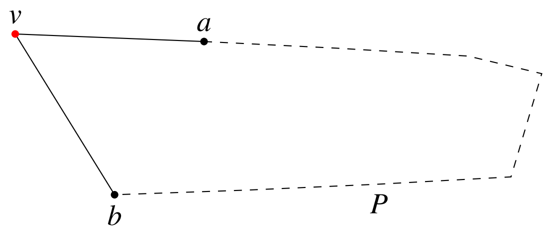

Figure 3.4: Partial point set S^* with path connecting vertices a and b.

our case, we followed this for $t = \frac{3+4\pi}{3}$.

After reaching that design, all that is left is to carefully choose points that satisfy it. In S^*, the path P consists of edges aa_1, a_1a_2, a_2a_3, a_3b_3, b_3b_2, b_2b_1, and b_1b. We make sure every angle in path P is smaller than π to maintain convexity of S^*. In the end, we have the points and spanner graph shown in Figure 3.2.

3.3 A Non-Plane Convex Path-Greedy Spanner

Theorem 3. *There exists a convex point set S' such that the spanner generated by* PATHGREEDY(S', $\frac{3+4\pi}{3}$) *is not plane.*

Proof. The proof is by example. The conditions established for a strong example are the same ones mentioned in the proof of Theorem 2, that is: (i) It should have no three collinear points; and (ii) its path-greedy $\frac{3+4\pi}{3}$-spanner should be independent of the tie breaking between edge distances. The example for S' satisfies this condition, and its coordinates are shown in Table 3.2.

One can easily verify that the spanner generated by PATHGREEDY(S', $\frac{3+4\pi}{3}$) is not plane by running the algorithm. Its output is shown in Figure 3.5.

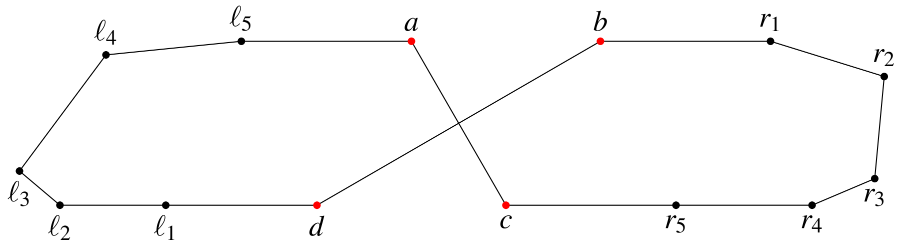

Figure 3.5: Spanner generated by PATHGREEDY(S', $\frac{3+4\pi}{3}$).

Table 3.2: Coordinates and labels of point set S'.

Vertex	**x-coordinate**	**y-coordinate**
a	0	0
b	10	0
c	5	−8.66
d	−5	−8.66
r_1	19	−0.001
r_2	25.018	−1.863
r_3	24.515	−7.255
r_4	21.2	−8.658
r_5	14	−8.659
ℓ_1	−13	−8.659
ℓ_2	−18.6	−8.658
ℓ_3	−20.741	−6.854
ℓ_4	−16.164	−0.72
ℓ_5	−9	−0.001

□

Construction and Generalization of S'

Though it may not be obvious at first glance, the points a, b, c, and d form an [approximate] rhombus. More interestingly, they form a rhombus in which the length of the sides [approximately] equals the the length of the smallest diagonal. Hence, we have $|ab| \approx |bc| \approx |cd| \approx |da| \approx |ac| \approx k$, for some value k. (In the example shown in Table 3.2, $k = 10$, but the actual value does not matter.) The first key idea is to make sure $|ac|$ is smaller than the lengths of the four sides by a negligible amount, in order to force path-greedy to scan the edge first amongst all the ones in the quadrilateral.

Our goal is to make path-greedy add both edges ac and bd. Adding the former is easy, as we just forced it to be smaller than all the other edges in the quadrilateral. However, as a consequence of that, bd is now the largest edge between them, meaning that the points b and d will be connected by at least one path by the time the edge gets scanned, and now we have to make sure that this path is longer than $t \cdot |bd|$. Similarly, by the time bd gets scanned, we know all the edges of the side of

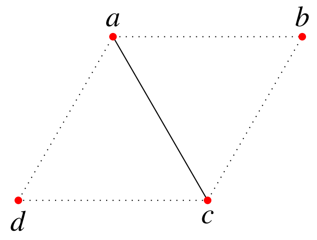

Figure 3.6: Partial process of designing S'.

the quadrilateral will have been processed by the algorithm, implying those pairs will also be connected by some path.

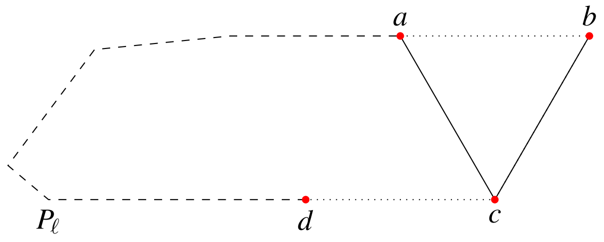

Figure 3.7: Hypothetical scenario when designing S'.

Notice that, by applying sine law on triangle Δabd, we know $|bd| \approx \sqrt{3}k$. Now, assume the algorithm were to add one of the edges that is a side of the quadrilateral, say, edge bc. Thus, by the time the algorithm scans edge bd, we know the edges ac and bc were added, and that ad is connected by some path P_ℓ such that $|P_\ell| \leq t|ad|$. (The path P_ℓ may consist of just a direct edge between a and d.) This scenario is illustrated in Figure 3.7.

Under this hypothetical case, the vertices b and d will be connected by the path $P_\ell + ac + cb$ *before* the edge bd is scanned. Since we want path-greedy to add the edge bd, that path must be longer than $t \cdot |bd|$, and hence:

$$\begin{aligned}
t|bd| &< P_\ell + |ac| + |bc| && \implies \\
t\sqrt{3}k &< tk + k + k && \implies \\
tk(\sqrt{3}-1) &< 2k && \implies \\
t(\sqrt{3}-1) &< 2 && \implies \\
t &< \frac{2}{\sqrt{3}-1} \approx 2.732
\end{aligned}$$

Which unfortunately contradicts our chosen t. Therefore, if the edge bc were to be added, then the edge bd wouldn't. By symmetry, we see that the other edges forming sides of the quadrilateral, ab, cd, and ad must also not be added by path-greedy. Thus, we must connect all those pairs with sufficiently small paths before they are scanned to guarantee those edges are not added.

If we were to connect all four pairs of points formin the sides of the quadrilateral with disjoint paths, similar to how pair ad is connected in Figure 3.7, the resulting point set would not be convex. Instead, what we do is create those paths only on two opposing sides. Let us then create a path P_ℓ connecting the pair ad and a path P_r connecting the pair bc. This is shown in Figure 3.8.

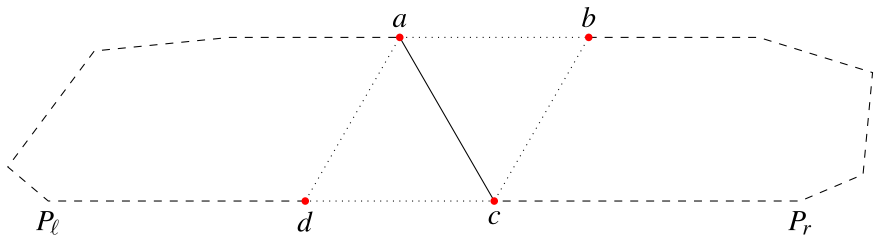

Figure 3.8: Connecting paths of S'.

Now, the shortest path between vertices c and d consists of $P_\ell + ac$. If we limit the length of P_ℓ to $(t-1)k$, then that path will have length slightly below tk, meaning it will be short enough for the edge cd not to be added. Likewise, we limit the length of P_r to $(t-1)k$, making sure the edge ab is not added.

At this stage, there is a single path connecting vertices b and d before the edge bd gets scanned: $P_\ell + ac + P_r$. In order for that path to be sufficiently long, we must have:

$$\begin{aligned}
|P_\ell| + |ac| + |P_r| &> t|bd| && \iff \\
(t-1)k + k + (t-1)k &> \sqrt{3}tk && \iff \\
(t-1) + 1 + (t-1) &> \sqrt{3}t && \iff \\
2t - 1 &> \sqrt{3}t && \iff \\
t &> \frac{1}{2-\sqrt{3}} \approx 3.733
\end{aligned}$$

We see this construction works for the value of t that we are concerned with, $\frac{3+4\pi}{3}$. Further, it could also be generalized for any value $t > 3.733$. In order to get from this design to the actual coordinates shown in Table 3.2, all one must do is carefully pick points that will satisfy all the conditions shown – a task that can be slightly tedious. But, after doing so, we can reach the final graph shown in Figure 3.5.

Chapter 4

Path-Greedy Grid Spanners

If you don't know where you are going, you'll end up someplace else.
– Yogi Berra

Uniform grid spanners are spanners in which its points are laid out in a uniform grid, such as in Figure 4.1. In this chapter, we will analyze the behaviour of the path-greedy algorithm on such point sets, considering different sizes and dilation factors.

A critical but often disregarded aspect of path-greedy is how it treats ties between edges of the same length. By definition (Algorithm 1), the algorithm breaks ties arbitrarily. While we analyzed convex point sets in Chapter 3, we deliberately designed examples in which the arbitrary tie-breaking would play no role in the final output. However, uniform grid graphs contain an enormous number of edges of the same length. Rather than perturb the points to eliminate ties, we study how different tie-breaks can lead to wildly distinct outputs.

Specifically, we will study in this chapter how non-planar t-spanners may be output by the path-greedy algorithm for different values of t, and what types of cross-edges are possible in an output.

This type of work is new and, as far as we know, no one else has studied this setup before. We also found that many new problems arise whenever we delved into the ones we found interesting. Hence, what is shown and proved here is far from all that is noteworthy about this set of problems.

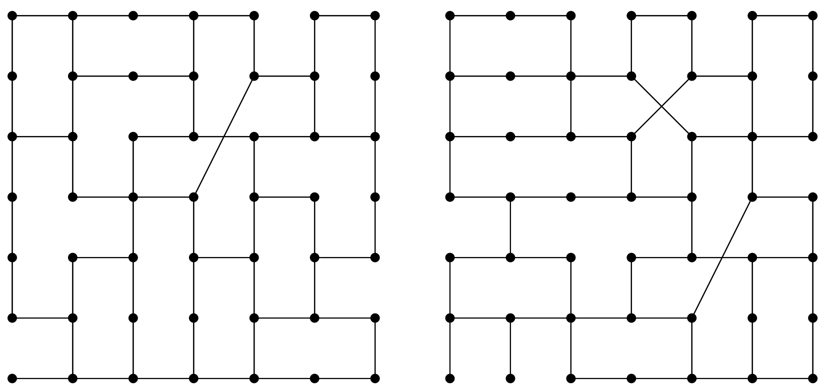

Figure 4.1: Greedy spanners in uniform 7×7 grids ($t = 3$).

4.1 Basic Observations

Every edge in the complete grid graph has length $\sqrt{h^2 + v^2}$, where h and v are the horizontal and vertical distance between the two points. We will define them as $h \times v$-edges (or $v \times h$-edges if $v < h$). Naturally, the shortest possible edges will have length 1, when the points are neighbours horizontally or vertically – 0×1 edges. Thus, all the neighbouring pairs of points will be processed first by path-greedy.

4.1.1 $t < 3$

When path-greedy scans a 0×1-edge, all the ones that were previously processed and added must also have been 0×1-edges. That means that, if there exists a path connecting the two vertices, that path must have an integer length greater than 1. Further, note that it is not possible to connect two neighbour nodes using exactly two 0×1-edges, and therefore, at this stage, every non-direct path connecting neighbouring nodes will have length at least 3. Hence, choosing $t < 3$ implies every 0×1-edge will be added to the output.

If $\sqrt{2} \leq t < 3$, there will be no need to add any other edge in the graph. Suppose the algorithm is scanning an $a \times b$-edge. The two points involved in it must be connected by a set of $a + b$ 0×1-edges, which is a small enough path for all $t \geq \sqrt{2}$ since $\frac{a+b}{\sqrt{a^2+b^2}} \leq \sqrt{2}$. Naturally, as t becomes lower than $\sqrt{2}$ and approaches 1, the number of edges increases until we reach a graph that is almost complete. (The graph will never be complete if the grid is larger than 2×2, as we will see in the next subsection.)

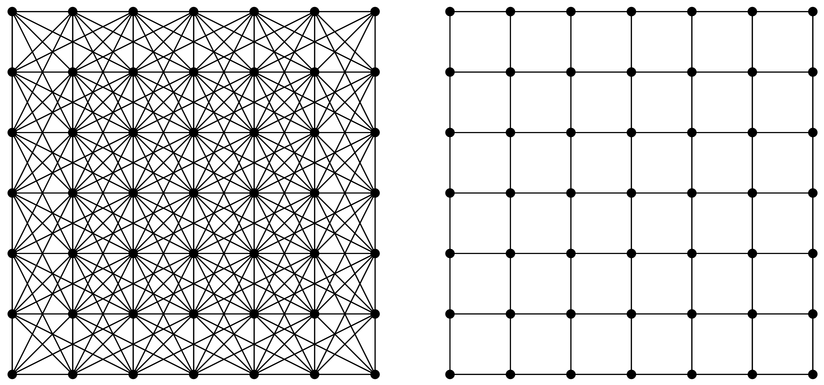
Figure 4.2: 7×7 grids with $t = 1.05$ (left) and $t = 1.5$ (right).

4.1.2 Clearly Impossible Edges

Path-Greedy will never add an $a \times b$-edge if $\gcd(a,b) > 1$ (recall that $\gcd(0,b) = b$).

Suppose the algorithm is scanning an $a \times b$-edge where $\gcd(a,b) = d$ for some $d > 1$. Let us call the two points in that edge p_1 and p_2. That edge must intersect at least one other point in the grid, turning it into d smaller $\frac{a}{d} \times \frac{b}{d}$-edges, as illustrated in Figure 4.3. Note that all those smaller edges will be scanned before the larger $a \times b$-edge and, whether or not they are added to the output by path-greedy, the points in them will be guaranteed to be connected by a path at most t times their length. Thus, p_1 and p_2 will be connected by the concatenation of those smaller paths. There are d of those smaller paths, and each of them has length at most $t\frac{\sqrt{a^2+b^2}}{d}$, and hence their concatenation has length at most $t\sqrt{a^2+b^2}$. Therefore, there will be no need to add the $a \times b$-edge, as that path is already small enough.

Ergo, a trivial condition for an $a \times b$-edge to be added by path-greedy is for a and b to be coprime.

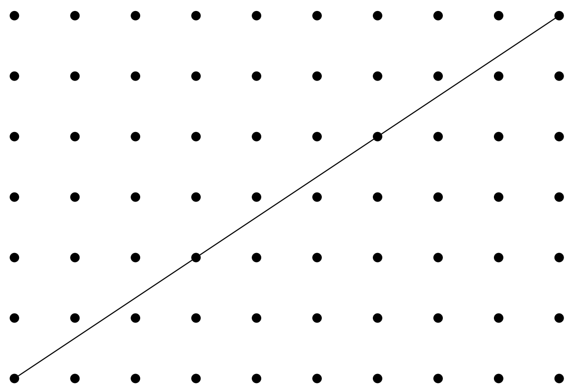

Figure 4.3: A long, hypothetical 6×9-edge.

4.2 Diagonal Edges and Planarity

Let us call all the $a \times b$-edges in which both a and b are greater than zero *diagonal edges*. An output that has no diagonal edges will be guaranteed to be plane. Thus, we will study some examples of how these edges arise in order for the spanner not to be plane. (Though, of course, the existence of a diagonal edge does not trivially guarantee that the graph will be non-plane.)

4.2.1 1×1-edges

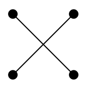

Figure 4.4: A crossing between two 1×1-edges.

After all the initial 0×1 pairs are scanned by path-greedy, the algorithm will process 1×1-edges. Thus, it is not particularly surprising that the algorithm can easily run into a scenario[1] in which two 1×1-edges within the same square are added, generating a crossing, as in Figure 4.4.

For any desired integer dilation t, there is a simple way to generate a crossing

[1]Assuming a completely random tie-break, the odds of this crossing are low. But, if an adversary chooses the tie-breaks, it is easy to generate this crossing.

between two 1×1-edges. First, we pick a 1×1 square and connect each pair of vertices u and v, where uv is a side of the square, by a path of unit length edges of length t, if t is odd, or $t-1$, otherwise. One such path forms three sides of a $1 \times k$ rectangle where $k = \lceil \frac{t-1}{2} \rceil$. The resulting shape, shown in Figure 4.5, looks similar to a plus sign. This process will make sure there is no need to add the 0×1 edges of the square, as they are connected by a path of length at most t, but the vertices of its diagonal will be connected by a path of length $4k+2$ (which equals $2t$ if t is odd, and $2(t-1)$ if t is even).

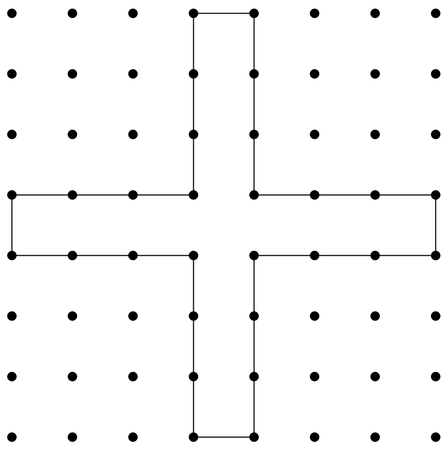

Figure 4.5: A "plus sign" of length 3 inside an 8×8 grid with $k = 3$.

Almost always possible

For every $D \times D$ grid with $D \geq 4$, it will be possible to see a crossing of 1×1 edges for some value of t. Note that the above plus sign construction fits inside a 4×4 grid, if $t = 2, 3$ so $k = 1$. In general, a plus sign of length k fits inside a $(2k+2) \times (2k+2)$ grid. Since there is a satisfying k for every integer $t \geq 3$, we are also guaranteed to find some grid that generates a crossing for each t we choose. This is not, however, the smallest grid for a given t for which path greedy can produce such a crossing – that problem will be covered in Section 4.3.

It is also important to note that the plus sign is far from being the only way to generate a crossing between two 1×1-edges. As D and t grow, there will be diverse ways to generate crossings between two of these edges, as there will be increasingly more ways to create structures that arise from the same idea: connecting the pairs of vertices in a square with paths of length approaching t. Two of these cases are shown in Figure 4.6.

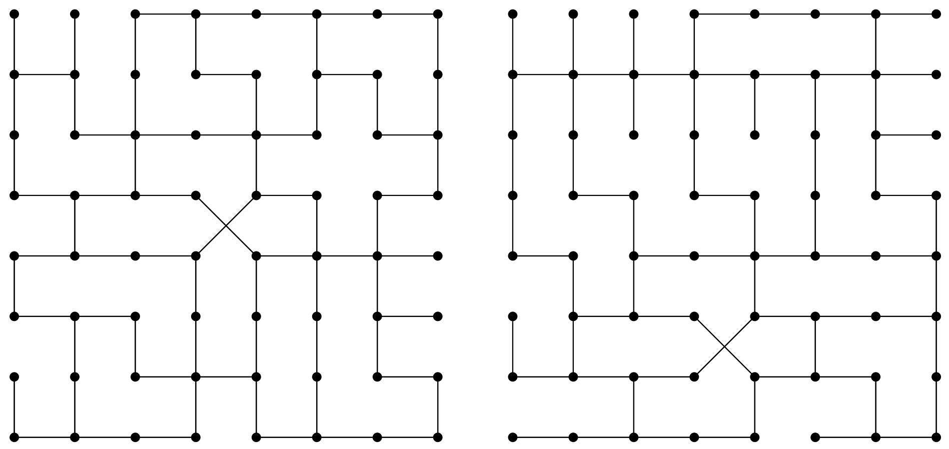

Figure 4.6: 8×8 non-plane grids with $t = 5$.

4.2.2 1×2-edges

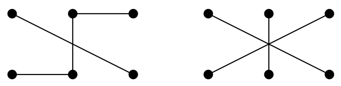

Figure 4.7: Examples of 1×2 edges with crossings.

In some cases, with $t = 3$, the tie-break in path-greedy may lead to the occurrence of 1×2-edges that yield a crossing. The crossing does not necessarily need to be only between two 1×2-edges – it may also happen with a 0×1-edge, as illustrated in Figure 4.7. Considering only integer values of t, we will prove that 1×2-edges can be included by path-greedy if $t = 3$, and cannot be included if t is an integer between 4 and 8.

Possible for $t = 3$ (with a mandatory crossing)

It is easy to prove by example that the crossings shown in Figure 4.7 can be added by path-greedy when $t = 3$. Figure 4.8 contains two examples on 5×5 grids. Naturally, it would be possible to see those in any $D \times D$ grid with $D > 5$ as well.

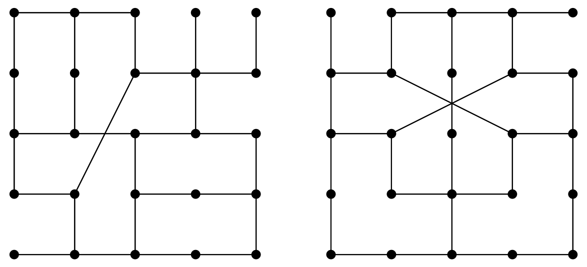

Figure 4.8: 5×5 grids with 1×2-edges ($t = 3$).

Mandatory crossing. If a 1×2-edge is added by path greedy, it must necessarily cross a 0×1-edge. Specifically, looking at the structure in Figure 4.9, in order for edge af to be added by path-greedy, the edge be must also be included.

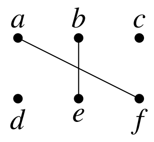

Figure 4.9: Points involved in a 1×2-edge.

To prove that statement, assume that the edge be is not added. Thus, there must be a path of length at most 3 connecting b and e, and all edges in the path must be 0×1-edges, since those are the only ones seen by path-greedy thus far. There are only two ways those vertices can be connected. The first is by the path $b \to a \to d \to e$, or, symmetrically, the path $b \to c \to f \to e$. Assume without loss of generality that it is the latter. Then, the shortest path between b and f has length at most 2, while the shortest path between a and b has length at most 3 (since $t = 3$), meaning there is a path between a and f of distance at most 5, which is less than $t\sqrt{|ab|} = 3\sqrt{5} \approx 6.71$, which implies path-greedy would not add edge af.

Thus, the addition of edge af necessarily implies that edge be will also be included, leading to a crossing. The same argument applies to all rotations and reflections of the structure.

Conditions for a 1×2*-edge.* See Figure 4.10, which is an extension of the construction shown above. We will deduce the conditions needed for path-greedy to add edge af by making sure we never add a path of length at most $t\sqrt{5}$ while using only edges smaller than 1×2.

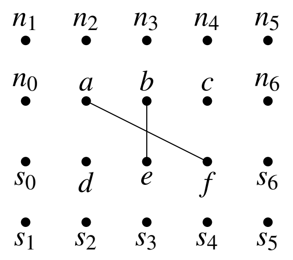

Figure 4.10: Points involved in a 1×2-edge and their surroundings.

If path-greedy added edge ab, then there would be a path of length 2 between a and e, followed by a path of length at most 3 between e and f, implying edge af would not be added. Thus, in order for edge af to be added, the algorithm cannot add edge ab and, similarly, it cannot add edge ef either. However, a and b must be connected by a path of length at most 3, leaving the paths $a \to n_2 \to n_3 \to b$ and $a \to d \to e \to b$ as the only options. If the latter were included, there would be a path of length 2 between a and e, repeating the issue just described. Thus, a and b must be connected by path $a \to n_2 \to n_3 \to b$ and, by the same argument, e and f must be connected by path $e \to s_3 \to s_4 \to f$. See Figure 4.11.

Now, suppose the algorithm were to add edge cf. Then, b and c must either be connected directly by path $b \to c$ or by path $b \to n_3 \to n_4 \to c$. In either case, there would be a path of length 5 connecting a and f, which breaks our premise. Hence, path-greedy cannot add edge cf or edge ad (by symmetry). But, since we must connect c and f by a short path, the edges cn_6, n_6s_6, and s_6f must be added, and so must edges an_0, n_0s_0, and s_0d. The resulting conditions are shown in Figure 4.12.

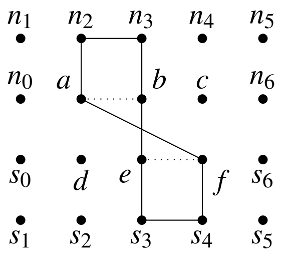

Figure 4.11: Necessary shape for inclusion of edge af. Dashed lines are edges that must not be added.

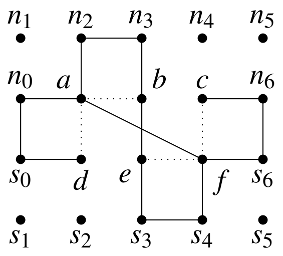

Figure 4.12: Necessary shape for inclusion of edge af, with more conditions. Dashed lines are edges that must not be added.

Those are the necessary conditions for a spanner to contain a 1×2-edge. From there, it is not hard to see that there are only three different satisfying constructions, displayed in Figure 4.13, alongside their reflections and rotations.

Impossible to add for integer t if $4 \leq t \leq 8$

We will start by showing a simple proof that a 1×2 edge cannot be added by path-greedy if $t = 4$. Then, we adapt the proof slightly for t such that $5 \leq t \leq 8$.

(i): $t = 4$. Let us inspect the by now familiar shape in Figure 4.14. Note that each of the neighbouring pair of points must be connected by a direct edge (meaning a path of length 1) or by a path of length 3 since, by a simple parity argument, we cannot connect two neighbours using exactly 2 or 4 edges.

Suppose path-greedy were to add any of the dashed edges: ab, bc, cf, ef, de,

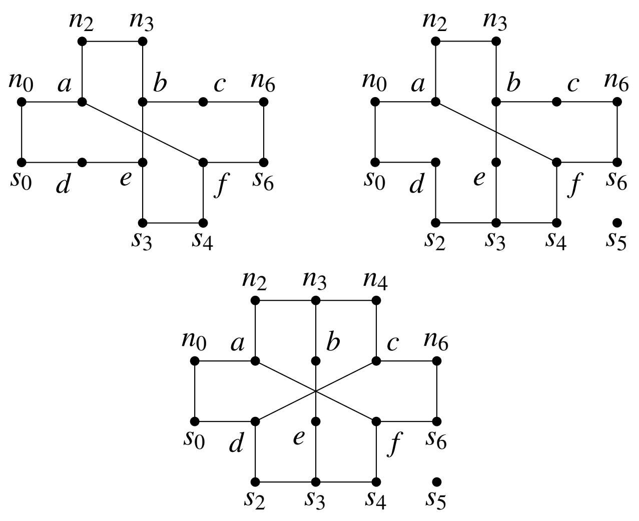

Figure 4.13: All shapes satisfying the conditions inferred (excluding rotations and reflections). The irrelevant points were removed.

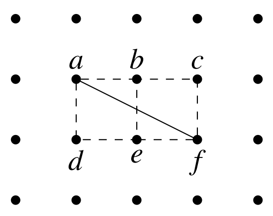

Figure 4.14: 1×2-edge and its surroundings.

or *be*. Then, we know all the other pairs of points must be connected by a path of length at most 3, while that specific pair is connected by a single edge. In that case, there would be a path from a to f of length at most 7. For example, if the algorithm added edge ab, we could traverse the edge ab itself, then at most three edges to reach c, and another three edges to reach f, totalling 7. However, a path of length 7 is shorter than $4\sqrt{5}$, rendering the addition of edge af unnecessary, which is not what we want. Hence, the algorithm must not add any of the dashed lines.

However, if none of those lines are added, it would be impossible to connect points b and e using at most 3 edges. With this contradiction, we see that it is impossible to add a 1×2 edge for $t = 4$.

(ii): $5 \leq t \leq 8$. The proof is extremely similar to the one just above. If the algorithm were to add any of the dashed edges, it would be possible to connect a to f by a path of length at most $1+t+t = 2t+1$. And, for all $t \geq 4.24$, we have $2t+1 \geq \sqrt{5}t$, meaning the addition of any of the dashed edges makes it so path-greedy does not add edge af. Thus, just like in the previous case, none of the dashed edges can be added. With none of those edges being added, it is impossible to connect vertices b and e using at most 8 edges. Thus, it is impossible to add edge af, or any 1×2-edge, for integers t such that $5 \leq t \leq 8$.

Larger values of t

I *strongly* believe it is impossible to add a 1×2-edge for any integer $t \neq 3$. The cases shown above are steps taken in that direction. However, finalizing this proof for larger t remains an open problem.

4.3 Smallest Grid for Each t

When studying properties that lead to a crossing, it is worth wondering what grids would even allow that. Specifically, what is the smallest value for D such that it is possible for an $D \times D$ grid to have a crossing when path-greedy is run with a certain value t. We will consider the problem with a focus on integer values of t, though the constructions shown are also valid for many non-integer values.

Before we explain the idea behind the constructions we believe to be optimal, it is important to mention that we have not proved a definitive lower bound. Thus, that part of the problem remains open.

4.3.1 General Shape

For two reasons, the crossing we will aim to achieve is one between two 1×1-edge. First, as shown in Section 4.2.1, such crossings are possible for essentially all values of t. The second reason is the diversity of possible ways to force the addition of a 1×1-edge works in our favour in this particular problem.

There is a general shape we follow to lead to such crossings, shown in Figure 4.15. Let us fix a square $abcd$ and force the addition of edges ac and bd by path-greedy. We must connect the pairs of vertices in the edges of the square – ab, *bc*, *cd*, and *da* – by paths of length at most t using only 0×1-edges, while connecting the diagonals *ac* and *bd* by paths greater than $\sqrt{2}t$ using the same edges.

To describe our construction, we split the grid into four regions, shown in green, yellow, orange, and blue in Figure 4.15. Each of these regions contains a long path connecting the pair of square points contained in it – for example, the green region will contain the path connecting vertices a and b. Containing the paths in disjoint regions ensures they do not intersect, which makes our analyses considerably easier. While it is not a strict requirement, we choose to make these paths equivalent (subject to rotation and translation). Thus, each region contains a path P such that:

1. $|P| \leq t$, so we do not add the edges of the periphery of the square;
2. $2|P| > \sqrt{2}t$, ensuring the diagonals are added by path-greedy (note that vertices a and c are connected by the path P going from a to b, and another copy of path P going from b to c); and
3. all the points within the region defined by path P are connected to each other by paths of length at most t.

Lastly, we make each of the regions be a distinct $(k+1) \times k$ or $k \times (k+1)$ rectangle. This makes the construction fit neatly into a compact grid. Looking again at Figure 4.15, we define the green (top left) and orange (bottom right) rectangles to be $(k+1) \times k$ rectangles, and the other two regions to be $k \times (k+1)$. Thus, the entire figure fits in a $2k \times 2k$ grid.[2]

4.3.2 Specific Constructions

We show two similar constructions following the aforementioned general structure that produce a $D \times D$ grid where $D = 2k$ for some k. The first construction only works for odd values of k, while the second one works for even integers k.

[2]Each side of the grid will have length $(k+1)+k-1 = 2k$. The minus 1 factor comes from the common points the two neighbouring regions share.

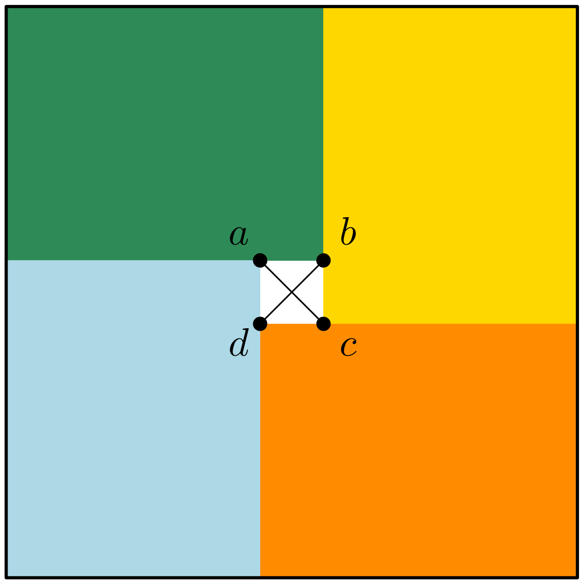

Figure 4.15: Shape of a general construction of a crossing between two 1×1-edges, using four independent regions.

First construction: Odd integers k

Let us start by showing how each region is constructed. Figure 4.16 contains an illustration for the green region when $k = 7$. We draw a single Hamiltonian path for the entire rectangle. This is the above-named path P.

We first start from vertex b and draw $k-1$ edges going up, passing through k points, and reaching vertex x. Then, add k edges going to the left until we reach vertex y. Then, alternate between going down one step followed by $k-1$ edges to the right, and going down one step followed by $k-1$ edges to the left, going down the rectangle in a snake pattern. For odd integers k, this process leads to a vertex z that is just above a, and we end the process by connecting these two. Now, vertices a and b are connected.

Note that, as long as the path P itself has length at most t (more on that soon), all the neighbouring vertices in the green region will also be spaced apart by a path at most t, since they are all part of P.

At this point, we must calculate the length of path P itself as a function of k.

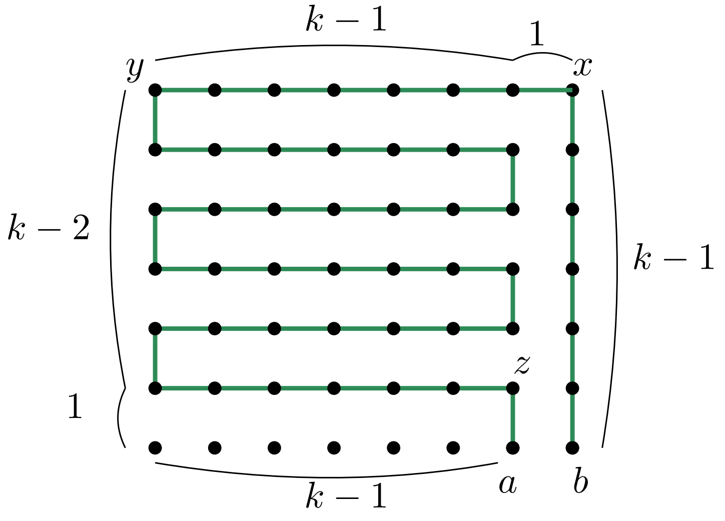

Figure 4.16: The $(k+1) \times k$ rectangle contained in the green region (with $k = 7$). The values in the figure refer to distances (or number of 0×1-edges).

The easiest way to do so is to separately count the vertical and horizontal edges. There are $k-1$ vertical edges from b to x and another $k-1$ vertical edges from a to y, totalling $2k-2$ edges. For the horizontal edges, we see that the rectangle defined by vertices y and z is a $k \times (k-1)$ rectangle, and as such contains $(k-1)$ sets of $(k-1)$ horizontal edges, totalling $(k-1)^2$. There is one last horizontal edge connected to point x. Hence, the total number of edges is $2k-2+(k-1)^2+1 = k^2$.

In order to satisfy both constraints regarding the length of path P, we need (i) $k^2 \leq t$, and (ii) $2k^2 > \sqrt{2}t$. The smallest such value k that satisfies both constraints (if it exists) is the smallest value that satisfies just the second requirement, which is easy to calculate. However, the smallest *odd integer* k_o such that $2k_o^2 > \sqrt{2}t$, which is either $\left\lceil \sqrt{\frac{t}{\sqrt{2}}} \right\rceil$ or $\left\lceil \sqrt{\frac{t}{\sqrt{2}}} \right\rceil + 1$, whichever is odd, may not satisfy the first constraint. That is, we may have $k_o^2 > t$. In that case, it will be impossible to use this construction for that value t. However, we prove in Appendix B that k_o satisfies both constraints for every integer $t \geq 121$ and for most positive integers before.

To conclude this, the other three regions are drawn as rotations of the first. Note that neighbouring points that are in different coloured regions are connected by at most $2k-1$, which is short enough to avoid a direct edge. Finally, our construction fits in a $D \times D$ grid, with $D = 2k_o$. Figure 4.17 contains an example for $k_o = 9$.

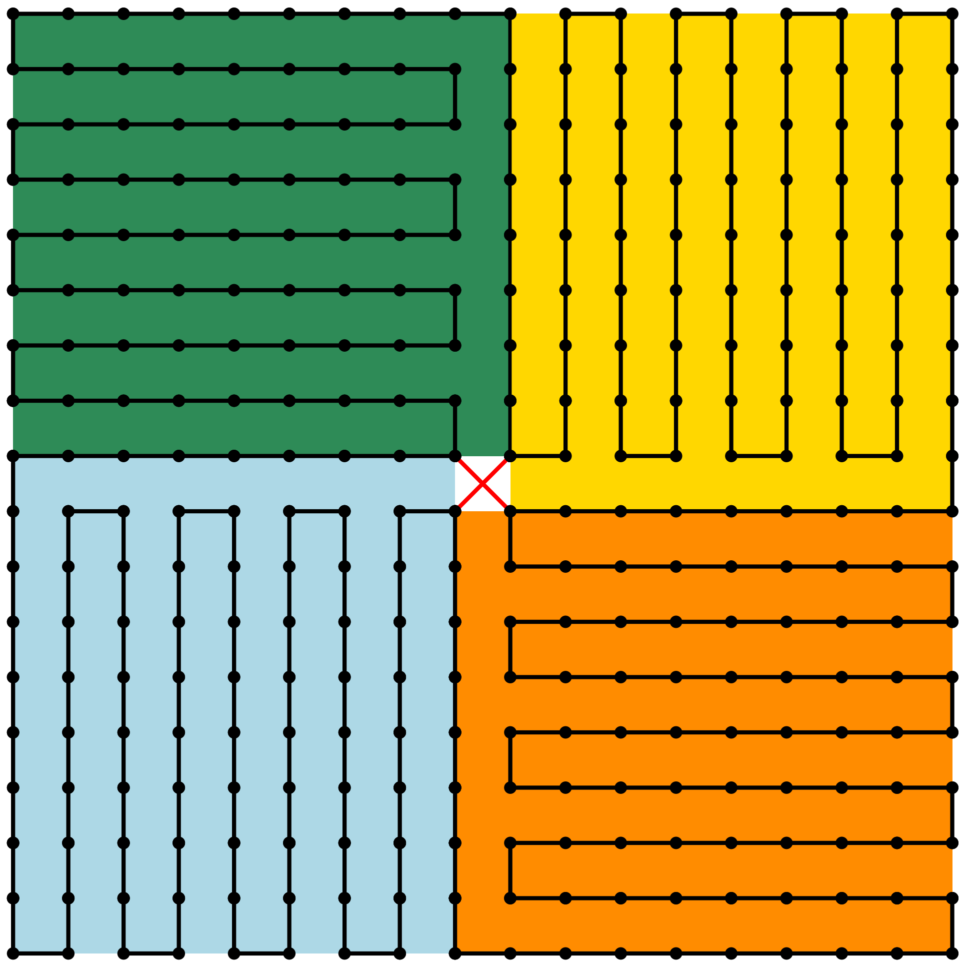

Figure 4.17: 18×18 grid from construction for odd integers k.

Second construction: Even integers k

This construction is extremely similar to the first. See Figure 4.18 for the shape of the region.

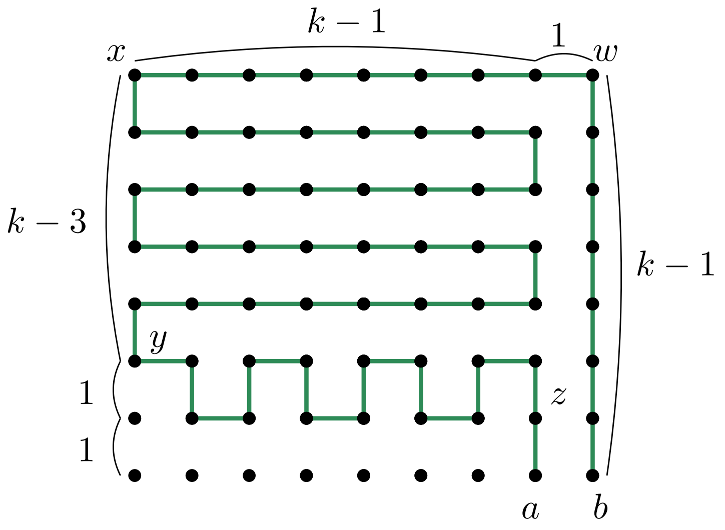

Figure 4.18: The $(k+1)\times k$ rectangle contained in the green region for even integers k (with $k=8$). The values in the figure refer to distances (or number of 0×1-edges).

The beginning of the path is the same as the case of odd k. From b, we draw $k-1$ vertical edges until we reach a point w. Then, add k edges going leftwards, reaching vertex x. Now, the alternating path of the previous case starts again. However, this time, we cut it short, ending at vertex y that is $k-3$ edges directly below x. This alternating path adds $k-3$ vertical edges and $(k-1)(k-4)$ horizontal edges. Then, we start a new path from y that alternates between going right, then down, and right, then up. We do this for $(k-1)$ steps, adding $2(k-1)$ edges until we reach a vertex z directly above a. Finally, we connect z to a with a single edge.

The total number of edges added – the length of path P – is the sum of the parts just mentioned: $(k-1)+k+(k-3)+(k-1)(k-4)+2(k-1)+1=k^2-1$. Thus, to satisfy the path length constraints, we need (i) $k^2-1\leq t$ and $2(k^2-1)>\sqrt{2}$. Similarly to the case above, we choose the smallest *even integer* k_e such that $2(k_e^2-1)>\sqrt{2}$, which is $\left\lceil\sqrt{\frac{t}{\sqrt{2}}+1}\right\rceil$ or $\left\lceil\sqrt{\frac{t}{\sqrt{2}}+1}\right\rceil+1$, whichever is even.

Unfortunately, not every value of t will have a corresponding valid k_e that satisfies both conditions. We show in Appendix B that the formula for k_e satisfies all integer values for $t>142$, and most positive integer values before that as well.

As expected, the other three regions are drawn as a rotation of the first. An example is shown in Figure 4.19. Hence, we pack the entire construction into a $D \times D$ grid with $D = 2k_e$.

There are three last points to note. First, strictly following the pattern shown in Figure 4.18 will lead to one point in each region that is not connected to any other point. This is easily fixed by connecting that point to either of its neighbours in the same region. Second, the neighbouring pairs of points in different coloured regions are connected by at most $3k-1$ edges, which is short enough, if $k \geq 4$, to not need a direct edge between them. Lastly, due to the shape of the construction, we require $k \geq 4$ for this pattern to be possible.

Combining both constructions

Having presented both constructions, all that is left to do is to choose the best one for each t. Of course, for many "small values" ($t \leq 142$), one of them might not be valid, leaving us with one or zero valid choice.

For all integers $t \geq 25$, at least one of the constructions is valid. In fact, the only positive integers $t \geq 9$ for which both are invalid are 13, 14, 23, and 24. (The proof of this is not particularly interesting and thus appears in Appendix B.) For these impossible values, we can use alternative constructions to find a crossing, such as the "plus sign" construction. When both constructions are valid, we can simply choose that one that yields the smallest grid.

Since both k_o and k_e are in $O(\sqrt{t})$, D is as well. Thus, the structures shown here yield a crossing for path-greedy with dilation t for a grid with $O(t)$ points.

4.4 A Few Open Problems (with Conjectures)

There are two open problems mentioned in the sections above worth reiterating. Further, there is one other problem that has caught my attention that I must mention.

Possibility of 1×2-edges

We saw in Section 4.2.2 that, while it is possible for 1×2-edges to show up when $t = 3$, it is not possible for them to show up when $4 \leq t \leq 8$. Answering that

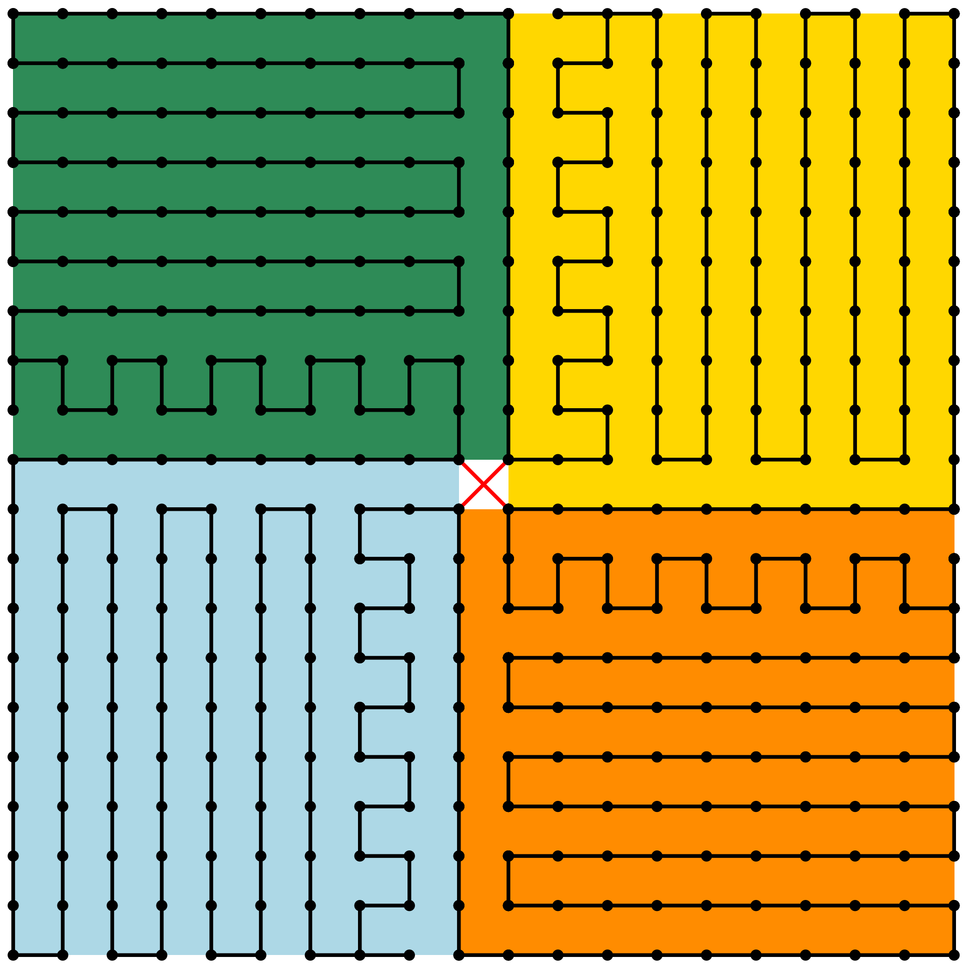

Figure 4.19: 20×20 grid from construction for even integers k.

question for the remaining values of t remains to be done. I strongly believe it is not possible, and $t = 3$ is indeed the largest integer value for which 1×2-edges can be added by path-greedy.

Most compact crossing for each t

We saw in Section 4.3 two constructions that may appear in a t-spanner created by path-greedy that result in a crossing in a grid of side $O(\sqrt{t})$. I believe those constructions are asymptotically optimal, and hence it is not possible to create such a construction for grids smaller than $O(\sqrt{t})$ in side (or $O(t)$ in total number of points) for large t values. However, whether that is indeed optimal is yet to be proved. Further, even if it does turn out to be the best achievable solution asymptotically, one may still find a way to improve it by constant factors. The general shape presented – of splitting the grid into 4 symmetric regions – would limit that, so one would need to think outside that box.

Possiblity of general diagonal edges

We proved that 1×1-edges are almost always possible – even in quite compact grids – and 1×2-edges are possible when $t = 3$. We also explained early on that many trivial edges are clearly impossible to be added, which are $a \times b$-edges when $\gcd(a, b) > 1$. The problem that remains open is whether it is possible to add any of the other edges, for any value of t. Again, I believe the answer is no, and that the only edges possible are 0×1, 1×1, and 1×2 (the latter one only when $t = 3$). Despite long attempts, I have not been able to prove this general statement.

Chapter 5

Using Steiner Points

It is difficult to make predictions, especially about the future.
— Danish Proverb

Steiner points are points in the plane that do not belong to the original input set of a problem. They can be added to improve a spanner in miscellaneous ways, such as reducing the degree, guaranteeing its planarity, or lowering the dilation of the graph.

In this chapter, we consider adding Steiner points to plane spanners of arbitrary degree in order to reduce its degree to at most 3, while not worsening the dilation by much. Specifically, we transform a plane t-spanner of N points into a plane $(t+\varepsilon)$-spanner of degree 3, for any $\varepsilon > 0$, that adds at most $O(N)$ Steiner points.

5.1 Construction from Biniaz et al. (2017)

Biniaz et al. introduced a construction to solve this problem [6]. However, their construction is only guaranteed to work for plane t-spanners with $t \geq \pi + 1$. The idea behind their construction is to draw a circle C_p with infinitesimal radius around each point p in the original point set. The edges that were incident on p are replaced with new edges ending at the intersection with C_p, adding a new Steiner point at each intersection. These Steiner points and a distinct Steiner point p' on C_p, connected to p, are then connected by a cycle in order around C_p. See Figure 5.1 for an illustration of this idea.

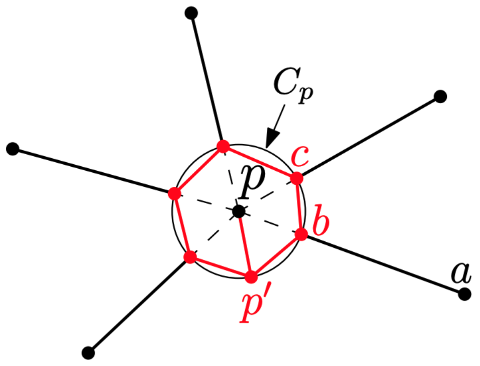

Figure 5.1: Example of Steiner points construction to reduce degrees. Figure source: Biniaz et al. [6].

The issue is that the dilation between point p and the Steiner points around C_p is greater than 1, making the construction valid only for certain values t. We can bound this worst-case dilation. If there are infinitely many Steiner points in C_p, including a point p^* diametrically opposed to p', then the maximum dilation would be $\pi + 1$, between points p^* and p'.

This can be improved if we connect p to 3 different points in the circle evenly spaced apart, although it is not mentioned in the original construction. Then, the maximum dilation would be between point p and some point in C_p that is halfway between two of the points p is connected to, leading to a dilation of $1 + \frac{\pi}{3}$.

5.2 A New Construction

5.2.1 Optimality via Collinearity

Given a plane t-spanner of arbitrary degree, we now present a construction that transforms it into a plane $(t + \varepsilon)$-spanner of degree 3 for any $\varepsilon > 0$ and $t \geq 1$.

This first step is shown in Figure 5.2. For each point p in the input spanner, draw a horizontal line that passes through it (shown dashed). The line splits p's edges into two sets, the ones above the line (upper) and the ones below it (lower). Let U_p be the number of edges above the upper set and L_p on the lower set. Let r_p be the semi-line drawn to the right of p (in red) and b_p be the semi-line drawn to

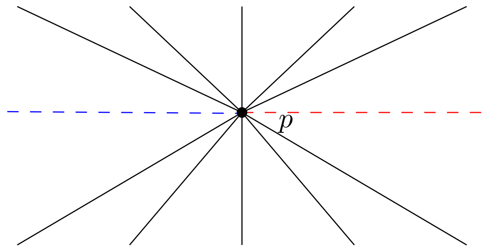

Figure 5.2: Separation of upper and lower edges for any point p.

the left of p (in blue).

Next, as shown in Figure 5.3, add U_p evenly spaced by distance ε' Steiner points in r_p, with the first one being in distance ε' away from p. We will show how to find the value for ε' later – for now, assume it is arbitrarily small. Do the same on the other side, adding L_p evenly spaced Steiner points in b_p by the same distance.

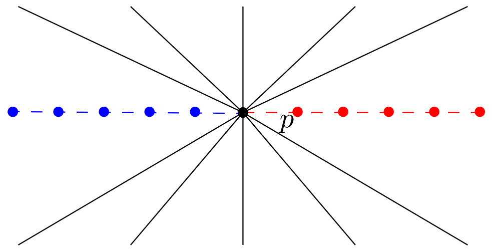

Figure 5.3: Addition of Steiner points related to point p.

Now, we proceed to update the edges. This step is shown in Figure 5.4. The leftmost upper edge incident in p gets replaced so that, instead of having p as an extremity, it has the leftmost Steiner point in r_p. Likewise, the second leftmost upper edge incident in p gets replaced by an edge incident in the second leftmost Steiner point in r_p. This procedure goes on until all upper edges are replaced, having none of them touch p any longer. Note that these edges will not intersect each other. We do the same operation symmetrically for the lower edges, replacing the rightmost edge by an edge incident in the rightmost Steiner point in b_p, and so on. Since the original edges did not intersect, the new ones will also not intersect for sufficiently small ε'.

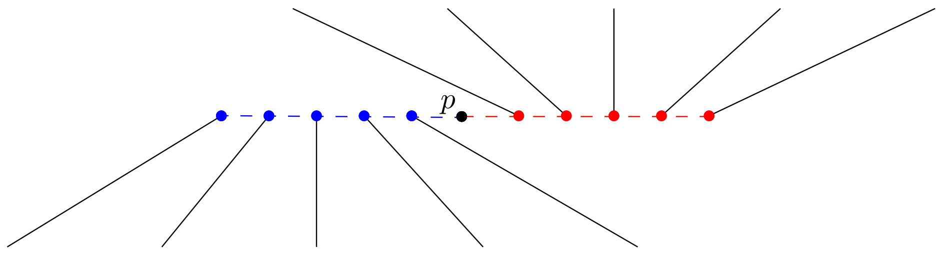

Figure 5.4: Redrawing of edges incident on point p.

The final step is to connect the Steiner points to each other by connecting each point (including p) to its one or two neighbours. (See Figure 5.5.) The degree of each Steiner point will be 2 or 3, while the degree of each vertex p in the input set will be 2.

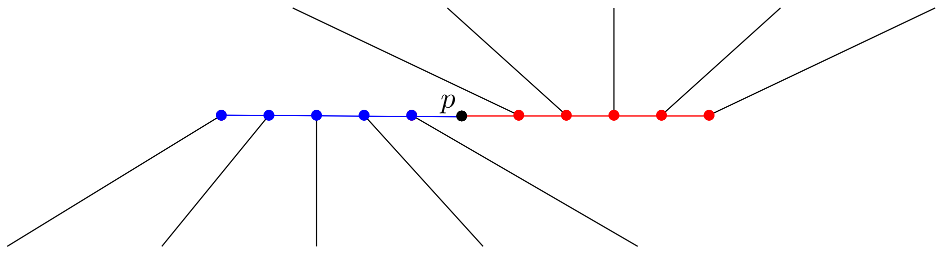

Figure 5.5: Final configuration for each point.

There are two Steiner points added per edge, making the total number of Steiner points used exactly $2M$, where M is the number of edges. Since the original graph is plane, M is in $O(N)$, meaning we only add $O(N)$ Steiner points.

5.2.2 Finding ε'

It is not hard to see that, for a sufficiently small ε', the construction will yield a plane $(t+\varepsilon)$-spanner of degree 3. Thus, one could think of an algorithm that starts with $\varepsilon' \leftarrow \varepsilon$ and, as long as the construction is not valid, replace it with $\varepsilon' \leftarrow \varepsilon'/k$ for any $k > 1$ and repeat.

But that does not stop us from calculating an exact value for ε' that is sufficiently small. We will start by calculating the maximum detour in a single edge from the original graph, and then use that to calculating how much any path in the graph can increase. After we have those values as a function of ε', we can limit the number so that the output is still a $(t+\varepsilon)$-spanner.

Maximum detour per edge

Each edge in the original graph is replaced by path consisting of a series of small edges of length ε', plus one larger edge connecting two Steiner points.

Let us study that situation using two arbitrary points a and b (in Figure 5.6).

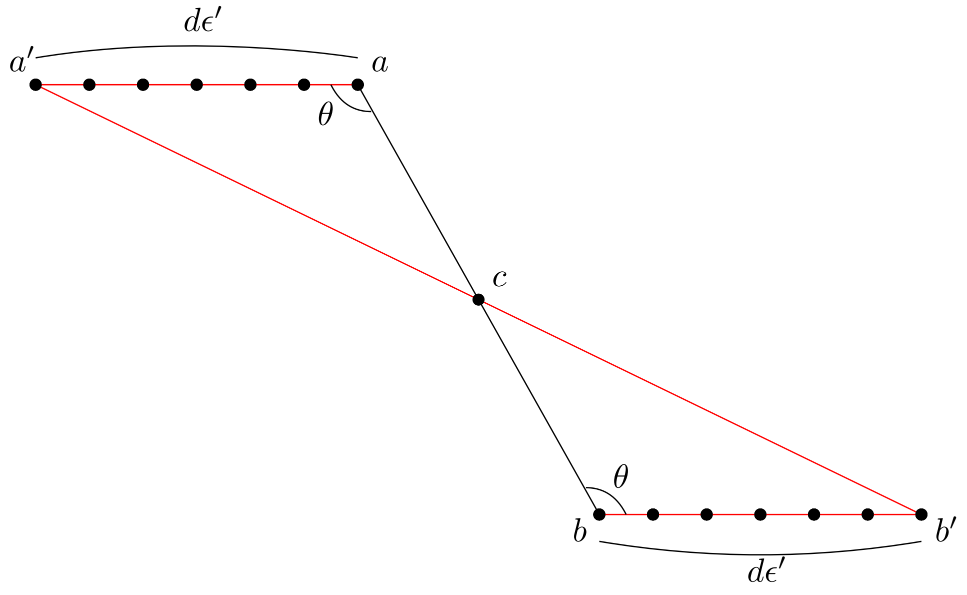

Figure 5.6: The original ab edge gets replaced with the longer red path, consisting of multiple edges.

Let $a'b'$ be the edge between two Steiner points, the first associated with a and the second with b, that form part of the shortest path in the new graph representing the edge ab. If d is the degree of the original graph then the path from a' to a (and from b' to b) is at most $d\varepsilon'$. In addition, the length of the edge $a'b'$ is at most $|ab|+2d\varepsilon'$ (which may occur if ab is horizontal) by the triangle inequality. Thus the total length of the path from a' to b' is at most $|ab|+4d\varepsilon'$.

Thus, each edge in the original graph gets replaced by a path that is at most $4d\varepsilon'$ units longer.

Limiting the detour per path

Let u and v be two arbitrary points in the graph, and δ_{uv} be the length of the shortest path between them in the original graph. We know that $\frac{\delta_{uv}}{|uv|} \leq t$ because the original graph was a t-spanner. The total path detour between them in the new graph depends on the number of edges involved, which is at most N. Thus, in order for the new path to be a $(t+\varepsilon)$-spanner, it suffices that:

$$\begin{aligned}
\frac{\delta_{uv}+ND}{|uv|} &\leq t\varepsilon \\
\frac{\delta_{uv}}{|uv|} + \frac{ND}{|uv|} &\leq t\varepsilon \\
\frac{ND}{|uv|} &\leq \varepsilon \\
D &\leq \frac{\varepsilon|uv|}{N} \\
4d\varepsilon' &\leq \frac{\varepsilon|uv|}{N} \\
\varepsilon' &\leq \frac{\varepsilon|uv|}{4dN}
\end{aligned}$$

In order for that condition to be satisfied for every pair of distinct points u and v, we define ℓ as the distance between the closest pair of points in the point set. Hence, it is sufficient, but not necessary, that

$$\varepsilon' \leq \frac{\varepsilon\ell}{4dN}$$

for the final graph to be a $(t+\varepsilon)$-spanner.

5.2.3 Avoiding collinearity

As a last note regarding the construction, it is worth noting that, for some people and cases, collinearity is undesired.

Fortunately, the collinearity in the construction is not strictly needed – it is there for convenience. We can avoid it simply by replacing each line segment with part of a parabola. The focal radius of the parabola can be arbitrarily large, which would bring all the properties shown of collinear points, without them actually being collinear. For sufficiently flat parabolas, all the calculations would stay approximately the same, and the output would not have sets of collinear points. One last thing to note is that if *ab* is horizontal, these could be problematic as the flat parabolas could still have collinear points between them. An easy way to solve that is by rotating the horizontal lines by an infinitesimal angle.

Chapter 6

Conclusion

I'll miss the sea, but a person needs new experiences.
— Frank Herbert (1965)

No matter how deeply one explores each of these topics, there will always remain a lot of work to be done.

In this thesis, we were able to explore some properties of the path-greedy algorithm. We studied its degree and planarity on convex point sets, proving that the maximum degree of PATHGREEDY(S, 5.19) is 4, with that being a tight bound, and that the result might not be plane, solving an open problem proposed by Bakhshesh and Farshi [3]. Further, we also explored how its tie-breaking can yield vastly different results under some circumstances – specifically, that of grid point sets. We presented what we believe to be the most compact grid that leads to a crossing for each t value in path-greedy, and what types of edges can show up.

But those are only a fraction of all the properties of path-greedy worth exploring – Narasimhan and Smid [16] have a long chapter dedicated to it. Also, path-greedy is only one of many interesting algorithms to consider.

Lastly, we also showed how the addition of $O(N)$ Steiner points can transform a plane t-spanner of arbitrary degree into a plane $(t+\varepsilon)$-spanner of maximum degree 3. Though there existed previous work on that same problem, our solution is the first one we know that actually works for all values $t < 1$ and $\varepsilon > 0$.

6.1 Future Work

All the results presented in Chapter 3 and Chapter 4 can be pushed further. It is worth exploring the degree of path-greedy over several different values t for convex point sets, and I would also like to see that question being approached for general point sets. Chapter 4 lists specific questions that seem interesting and would be a natural follow-up from the problem there.

There is some work actively being done trying to optimize the path-greedy algorithm itself, since its original $O(n^3 \log n)$ form is quite inefficient. As mentioned previously, there is new work published as recently as this year by Carmi and Tomer [10] creating a $O(n^2 \log n)$ version. I believe it is worth studying path-greedy for specific point sets and specifically for Euclidean distances, as further optimizations are likely possible.

Though it is hard to anticipate how difficult each of these problems will be until one actually approaches them, I can say it will certainly be a riveting challenge.

Bibliography

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9(1):81–100, 1993. → page 9

[2] S. Arikati, D. Z. Chen, L. P. Chew, G. Das, M. Smid, and C. D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *European Symposium on Algorithms*, pages 514–528. Springer, 1996. → page 13

[3] D. Bakhshesh and M. Farshi. A degree 3 plane 5.19-spanner for points in convex position. In *Proceedings of the 32nd Canadian Conference on Computational Geometry (CCCG2020)*, pages 226–232, 2020. URL https://vga.usask.ca/cccg2020/papers/Proceedings.pdf. → pages 5, 11, 12, 14, 51

[4] A. Biniaz, M. Amani, A. Maheshwari, M. H. M. Smid, P. Bose, and J. D. Carufel. A plane 1.88-spanner for points in convex position. *Journal of Computational Geometry*, 7(1):520–539, 2016.

[5] A. Biniaz, P. Bose, J. D. Carufel, C. Gavoille, A. Maheshwari, and M. H. M. Smid. Towards plane spanners of degree 3. *Journal of Computational Geometry*, 8(1):11–31, 2017.

[6] A. Biniaz, P. Bose, J.-L. D. Carufel, C. Gavoille, A. Maheshwari, and M. Smid. Towards plane spanners of degree 3, 2017. → pages xi, 11, 12, 13, 44, 45

[7] P. Bose and M. Smid. On plane geometric spanners: A survey and open problems. *Computational Geometry*, 46(7):818 – 830, 2013. ISSN 0925-7721. doi:https://doi.org/10.1016/j.comgeo.2013.04.002. URL http://www.sciencedirect.com/science/article/pii/S0925772113000357. EuroCG 2009. → pages 6, 13

[8] P. Bose, L. Devroye, M. Löffler, J. Snoeyink, and V. Verma. The spanning ratio of the delaunay triangulation is greater than pi/2. In *CCCG*, pages 165–167. Citeseer, 2009. → page 8

[9] P. Bose, P. Carmi, M. Farshi, A. Maheshwari, and M. Smid. Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3):711–729, 2010. → page 11

[10] P. Carmi and I. Tomer. Path-greedy spanner in near-quadratic time: Simpler and better. In *37th European Workshop on Computational Geometry*, pages 50–55, 2021. URL http://eurocg21.spbu.ru/wp-content/uploads/2021/04/proceedings.pdf. → pages 11, 52

[11] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, third edition, 2008. → page 8

[12] W. Evans and L. Siaudzionis. Tight degree bounds for path-greedy 5.19-spanner on convex point sets. In *37th European Workshop on Computational Geometry*, pages 258–262, 2021. URL http://eurocg21.spbu.ru/wp-content/uploads/2021/04/proceedings.pdf. → page 15

[13] I. Kanj, L. Perkovic, and D. Turkoglu. Degree four plane spanners: Simpler and better. *Journal of Computational Geometry*, 8(2):3–31, 2017. → page 11

[14] J. M. Keil and C. A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete & Computational Geometry*, 7(1): 13–28, 1992. → page 8

[15] R. Klein and M. Kutz. Computing geometric minimum-dilation graphs is np-hard. In *International Symposium on Graph Drawing*, pages 196–207. Springer, 2006. → page 3

[16] G. Narasimhan and M. Smid. *Geometric spanner networks*. Cambridge University Press, Cambridge; New York, 2007. ISBN 9780521815130;0521815134;. → pages 6, 7, 10, 51

[17] Y. Wang and X.-Y. Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Networks and Applications*, 11(2):161–175, 2006. → page 11

Appendix A

How to Find Counterexamples

A considerable amount of work shown here involved finding examples to prove or explore relevant properties of algorithms. I personally found it difficult to do such work relying solely on pen and paper, and thus developed a library and a system to guide me. Since this method turned out to be, for me, very successful, I think it is appropriate that I present an overview in it.

Step 1: Implement Models and Algorithms

The foundation of all the simulations done was to be able to run path-greedy itself and, more importantly, model and represent spanners. I implemented this part entirely in C++. The SPANNERGRAPH class contained a point set and had intuitive methods, such as ADD_EDGE. Computations for shortest-path were re-run on each addition of a new edge, recalculating the dilation of the graph and of each pair of points. This also made it possible to easily run path-greedy and get its output.

Step 2: Geometric Manipulations

Once we have a point set, we can run it using the models and algorithms above and acquire the relevant output. However, how do we define the point set? We can painstakingly do this manually (which I did for a regretful day), but that does not scale once we start making small changes over and over again.

A much better approach is to take advantage of the available numerical computation in Numpy. I wrote my own code in Python to represent points and point sets, as well as all the geometric computations I knew I would frequently need. This is not necessarily optimal, as there likely are libraries widely available for the task. Nevertheless, this short effort saved me a tremendous amount of frustrating time.

Step 3: Drawing Tools

Once the output is generated, we must help ourselves interpret it. A list of several coordinates for points and lines is not intuitive. A figure is. Fortunately, drawing such figures is quite straightforward using Python. There are plenty of resources online, and libraries such as Matplotlib and Python Imaging Library (PIL) were enough for all the tasks I needed.

I did have to write customized functions that interpreted the output from Step 1 and get the images in the way I wanted. Luckily, that proved to be uncomplicated.

Step 4: Simulate

Once all the tools are ready, it is time to connect them all together and be able to simulate the algorithms conveniently. I chose to do so in a Jupyter Notebook due to its many strengths. Its cell-by-cell nature made experimentation considerably easier, as I could frequently do a minor change and re-run a specific cell.

For Chapter 3, it was also important to do Monte Carlo simulations to understand the behaviour of the grip spanners. These specific ones were not done in a Jupyter Notebook, as running on a terminal was enough.

Step 5: Learn and Repeat

Each iteration over the previous steps [hopefully] teaches something new. Or gives some insight over what the algorithm is doing. In many cases, it was used to disprove small conjectures by quickly testing them out many times.

Overall, I can certainly say that I only found the shown counterexamples due to this process.

Appendix B

Proofs and Tables for Chapter 4

B.1 Proofs for k_o and k_e

Lemma 3. *For all integer $t \geq 121$, the formula for k_o returns a value satisfying both path constraints: $k_o^2 <= t$ and $2 \cdot k_o^2 > \sqrt{2} \cdot t$.*

Proof. Note that k_o is bounded by $\sqrt{\frac{t}{\sqrt{2}}} + 2$. Let's see when that value squared is less than t. The function $f(t) = t - \left(\sqrt{\frac{t}{\sqrt{2}}} + 2\right)^2$ is convex for $t \geq 0$. Since $f(0) < 0$ and $f(159) > 0$, all integers $t \geq 159$ will have a valid k_o. We prove that integers t in the range $[121, 158]$ have valid values k_o by brute force, referring to Table B.1 and Table B.2. $\square$

Lemma 4. *For all integer $t \geq 143$, the formula for k_e returns a value satisfying both path constraints: $k_e^2 <= t$ and $2 \cdot k_e^2 > \sqrt{2} \cdot t$.*

Proof. The proof is similar to the one above. The value k_e is bounded by $\sqrt{\frac{t}{\sqrt{2}} + 1} + 2$, and the function $f(t) = t - \left(\sqrt{\frac{t}{\sqrt{2}} + 1} + 2\right)^2$ is convex for $t \geq 0$. Since $f(0) < 0$ and $f(166) > 0$, all integers $t \geq 166$ have a valid k_e for both constraints. The same is proved for integer values t in the range $[143, 165]$ by brute force in Table B.1 and Table B.2. $\square$

B.2 Key Values for each Integer t

Table B.1: k_o, k_e, and N for each t when both constraints are valid. (Part 1.)

t	k_o	k_e	N	t	k_o	k_e	N	t	k_o	k_e	N	t	k_o	k_e	N
3	–	–	–	4	–	–	–	5	–	–	–	6	–	–	–
7	–	–	–	8	–	–	–	9	3	–	6	10	3	–	6
11	3	–	6	12	3	–	6	13	–	–	–	14	–	–	–
15	–	4	8	16	–	4	8	17	–	4	8	18	–	4	8
19	–	4	8	20	–	4	8	21	–	4	8	22	–	–	–
23	–	–	–	24	–	–	–	25	5	–	10	26	5	–	10
27	5	–	10	28	5	–	10	29	5	–	10	30	5	–	10
31	5	–	10	32	5	–	10	33	5	–	10	34	5	–	10
35	5	6	10	36	–	6	12	37	–	6	12	38	–	6	12
39	–	6	12	40	–	6	12	41	–	6	12	42	–	6	12
43	–	6	12	44	–	6	12	45	–	6	12	46	–	6	12
47	–	6	12	48	–	6	12	49	7	6	12	50	7	–	14
51	7	–	14	52	7	–	14	53	7	–	14	54	7	–	14
55	7	–	14	56	7	–	14	57	7	–	14	58	7	–	14
59	7	–	14	60	7	–	14	61	7	–	14	62	7	–	14
63	7	8	14	64	7	8	14	65	7	8	14	66	7	8	14
67	7	8	14	68	7	8	14	69	7	8	14	70	–	8	16
71	–	8	16	72	–	8	16	73	–	8	16	74	–	8	16
75	–	8	16	76	–	8	16	77	–	8	16	78	–	8	16
79	–	8	16	80	–	8	16	81	9	8	16	82	9	8	16
83	9	8	16	84	9	8	16	85	9	8	16	86	9	8	16
87	9	8	16	88	9	8	16	89	9	8	16	90	9	–	18
91	9	–	18	92	9	–	18	93	9	–	18	94	9	–	18
95	9	–	18	96	9	–	18	97	9	–	18	98	9	–	18
99	9	10	18	100	9	10	18	101	9	10	18	102	9	10	18
103	9	10	18	104	9	10	18	105	9	10	18	106	9	10	18
107	9	10	18	108	9	10	18	109	9	10	18	110	9	10	18
111	9	10	18	112	9	10	18	113	9	10	18	114	9	10	18
115	–	10	20	116	–	10	20	117	–	10	20	118	–	10	20
119	–	10	20	120	–	10	20	121	11	10	20	122	11	10	20
123	11	10	20	124	11	10	20	125	11	10	20	126	11	10	20
127	11	10	20	128	11	10	20	129	11	10	20	130	11	10	20
131	11	10	20	132	11	10	20	133	11	10	20	134	11	10	20
135	11	10	20	136	11	10	20	137	11	10	20	138	11	10	20
139	11	10	20	140	11	10	20	141	11	–	22	142	11	–	22
143	11	12	22	144	11	12	22	145	11	12	22	146	11	12	22
147	11	12	22	148	11	12	22	149	11	12	22	150	11	12	22

Table B.2: k_o, k_e, and N for each t when both constraints are valid. (Part 2.)

t	k_o	k_e	N	t	k_o	k_e	N	t	k_o	k_e	N	t	k_o	k_e	N
151	11	12	22	152	11	12	22	153	11	12	22	154	11	12	22
155	11	12	22	156	11	12	22	157	11	12	22	158	11	12	22
159	11	12	22	160	11	12	22	161	11	12	22	162	11	12	22
163	11	12	22	164	11	12	22	165	11	12	22	166	11	12	22
167	11	12	22	168	11	12	22	169	11	12	22	170	11	12	22
171	11	12	22	172	13	12	24	173	13	12	24	174	13	12	24
175	13	12	24	176	13	12	24	177	13	12	24	178	13	12	24
179	13	12	24	180	13	12	24	181	13	12	24	182	13	12	24
183	13	12	24	184	13	12	24	185	13	12	24	186	13	12	24
187	13	12	24	188	13	12	24	189	13	12	24	190	13	12	24
191	13	12	24	192	13	12	24	193	13	12	24	194	13	12	24
195	13	12	24	196	13	12	24	197	13	12	24	198	13	12	24
199	13	12	24	200	13	12	24	201	13	12	24	202	13	12	24
203	13	14	26	204	13	14	26	205	13	14	26	206	13	14	26
207	13	14	26	208	13	14	26	209	13	14	26	210	13	14	26
211	13	14	26	212	13	14	26	213	13	14	26	214	13	14	26
215	13	14	26	216	13	14	26	217	13	14	26	218	13	14	26
219	13	14	26	220	13	14	26	221	13	14	26	222	13	14	26
223	13	14	26	224	13	14	26	225	13	14	26	226	13	14	26
227	13	14	26	228	13	14	26	229	13	14	26	230	13	14	26
231	13	14	26	232	13	14	26	233	13	14	26	234	13	14	26
235	13	14	26	236	13	14	26	237	13	14	26	238	13	14	26
239	13	14	26	240	15	14	28	241	15	14	28	242	15	14	28
243	15	14	28	244	15	14	28	245	15	14	28	246	15	14	28
247	15	14	28	248	15	14	28	249	15	14	28	250	15	14	28
251	15	14	28	252	15	14	28	253	15	14	28	254	15	14	28
255	15	14	28	256	15	14	28	257	15	14	28	258	15	14	28
259	15	14	28	260	15	14	28	261	15	14	28	262	15	14	28
263	15	14	28	264	15	14	28	265	15	14	28	266	15	14	28
267	15	14	28	268	15	14	28	269	15	14	28	270	15	14	28
271	15	14	28	272	15	14	28	273	15	14	28	274	15	14	28
275	15	14	28	276	15	16	30	277	15	16	30	278	15	16	30
279	15	16	30	280	15	16	30	281	15	16	30	282	15	16	30
283	15	16	30	284	15	16	30	285	15	16	30	286	15	16	30
287	15	16	30	288	15	16	30	289	15	16	30	290	15	16	30
291	15	16	30	292	15	16	30	293	15	16	30	294	15	16	30
295	15	16	30	296	15	16	30	297	15	16	30	298	15	16	30
299	15	16	30	300	15	16	30	301	15	16	30	302	15	16	30