1

Introduction

> The truth is, most of us discover where we are headed when we arrive. At that time, we turn around and say, yes, this is obviously where I was going all along.
>
> —Bill Watterson, 1990

1.1 What is this book about?

In this book, we will consider the following problem:

> **General network design problem:** Given a set S of n points in $\mathbb{R}^d$, how to construct a *good* network that connects these points?

What do we mean by this? A *network connecting the points* of S is a graph $G = (S, E)$ with vertex set S and edge set $E \subseteq S \times S$, such that any two points $p, q \in S$ are connected by a path in G. A *geometric network* is a weighted graph where the vertices correspond to point sites in Euclidean space and the weights on the edges correspond to the distances between the endpoints in the Euclidean metric. Clearly, there are many such networks. For example, the *complete graph*, in which all pairs of distinct points are connected by an edge, is one such network connecting the points of S. The disadvantage, however, is that the number of edges is $\binom{n}{2} = \Theta(n^2)$; it is quadratic in the number of points.

Assume that the goal is to construct networks having only "few" edges. How many edges does a connected graph necessarily contain? The answer is $n - 1$. To prove this, consider an arbitrary network G connecting the points of S, and let m be the number of edges of G. Hence, we have to show that $m \geq n - 1$. Let us do the following: As long as G contains a cycle, take an arbitrary cycle, and remove an arbitrary edge from it.

Removing one edge from a cycle does not destroy the connectivity of the graph. Therefore, repeating this operation over and over again, we obtain an acyclic connected graph G', that is, a *tree*. If m' denotes the number of edges of G', then clearly $m \geq m'$. Hence, it suffices to show that $m' \geq n - 1$.

We claim that in fact $m' = n - 1$. That is, any tree on n points has exactly $n - 1$ edges. The proof is by induction on n. For $n = 1$, the claim is trivial. Let $n \geq 2$, and assume the claim is true for all trees on $n - 1$ points. Now let G' be a tree on n points. Since G' is acyclic, there is a point with degree exactly 1. (This is easy to prove by contradiction: if all degrees are larger than 1, then the graph must contain a cycle.) Remove this point, together with its adjacent edge, from G'. This gives a graph G'' on $n - 1$ points; in fact, G'' is a tree. Hence, by the induction hypothesis, G'' has $n - 2$ edges. Since G' itself has one more edge, the proof is complete.

Property 1.1.1. *Any network connecting a set of n points must have at least* $n - 1$ *edges.*

Let us call a network *sparse*, if it has $O(n)$ edges, that is, the number of edges is linear in the number of points. Again, there are many such sparse networks. Later in this chapter, we discuss several of them; each one is *good* in some sense.

The more general network design problem is to connect the set of sites by a network that satisfies a specified set of properties. To measure how *good* a network is, several quality measures have been used in the literature, some of which are listed here:

1. **Size** is defined as the number of edges in the network. In general, it is preferable to have networks with as few edges as possible, perhaps linear in the number of points.
2. **Weight** is defined as the sum of the weights of the edges. Since any network must connect all the points, its weight is bounded from below by the weight of a minimum spanning tree. The weight is a good measure of the cost of building the network; thus, it is often desirable to have networks with small weight.
3. **Stretch Factor** (or **Dilation** or **Distortion**) for two given points is defined as the ratio of the distance between the two points in the network to the metric distance between them. The stretch factor of a network is defined as the maximum stretch factor for any pair of distinct points in the network. It is often required that the stretch factor of the network be bounded by a small constant (which must be at least one). Networks with stretch factor at most t are called t-SPANNERS.
4. **Degree** is the maximum number of edges incident on any point in the network, often required to be bounded by a small constant. For a network, bounded degree implies small size, but not vice versa.
5. **Diameter** is the maximum number of edges along a shortest path connecting any two vertices in the network. It dictates the conciseness with which network paths can be described. A weighted diameter, that is, the total length of the longest path of minimum length could also be used as a quality measure instead. Diameters may be required to be small.
6. **Connectivity** is the vertex or edge connectivity of the network. It is a measure of how fault-tolerant the network is, since it signifies the number of points or links that must fail in order for the network to be disconnected.
7. **Fault Tolerance** is the number of points or links that must fail in order for the network to fail to have some desirable properties. This is slightly more general than the previous property.
8. **Genus:** In many applications, it is desirable for a network to be nearly planar. This is often quantified by its genus; it may also be measured by the size of the largest planar subgraph or the number of crossing edges in a straight-line drawing.
9. **Number of Steiner Points:** Often, better networks can be designed by adding Steiner points (points not in the input set). However, one may be constrained to have very few or no Steiner points in the network.
10. **Load Factor** of an edge can be defined as the number of shortest paths between some pair of vertices that use this edge (many alternate definitions and generalizations exist). The load factor of a network is defined as the load factor of the most "loaded" edge in the network. To prevent bottlenecks, it is desirable that the load factor of a network be small.

Earlier we discussed the property *size*. In general, when designing a network, one would like to impose constraints on a combination of the quality measures mentioned above; when analyzing a network, one would like to understand the properties of the network with respect to these quality measures. A common thread among most of the

problems mentioned in this monograph is the study of networks with small stretch factor (in combination with other properties); as mentioned earlier, such networks are called *spanners*. Spanners with small size and/or weight are referred to as *sparse spanners*.

To motivate such problems, we mention a few "concrete" examples. Consider the network of highways connecting n cities, where a road is a straight-line segment connecting two cities. If we want to travel from p to q, then we have to travel at least a distance $|pq|$, the Euclidean distance between p and q. Of course, if there is a direct highway linking p and q, then we travel this distance more or less exactly. Otherwise, it would be nice if there is a path between p and q, whose length is not too large as compared to $|pq|$, thus giving rise to the concept of a spanner. Observe that the *length* of a path is defined as the sum of the lengths of its edges.

Consider the section of the Scandinavian rail network (prior to the year 2000) shown in Figure 1.1. A quick inspection reveals that if one wants to use the rail system to travel from Malmö (marked by a M) in Sweden to Copenhagen (marked by a C) in Denmark, then the distance between these cities in the rail network is more than five times the direct "as-the-crow-flies" distance. Adding a direct link between these two cities would clearly improve the rail network.[1]

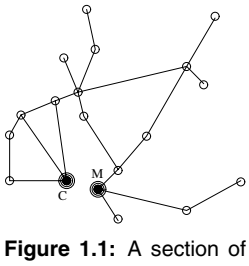

Figure 1.1: A section of the Scandinavian rail network prior to the year 2000.

We consider another example of a network **design** problem that relates to stretch factor: Imagine an existing set of highways connecting a collection of cities, where there is a need to upgrade the highway system in a cost-effective manner. Instead of spending the resources to improve all the existing highways, it would be better to improve only a carefully selected subset of the existing highways. If this subset of highway segments constitute a sparse spanner network (small stretch factor, small size, and small weight) of the highway system, then it is guaranteed that (i) one could drive from any city to any other using only improved highway segments with only a constant factor increase in the driving distance over distances in the original system of highways, and (ii) the amount of resources needed to upgrade the highway system is small since a sparse spanner has small size (number of highway segments) and weight (total length of the highways).

Given an existing system of highways, it is also easy to understand the significance of network **analysis**. Obvious queries include: "What is the size, weight, stretch factor, diameter, degree or connectivity of the network?" or "What is the farthest pair of cities in the network?" or "What is the stretch factor for a given pair of cities (i.e., what is the ratio of the length of the shortest path between two given cities in the network to the Euclidean distance between them?)?" For a federal authority maintaining the highway system, an appropriate query might be: "For which pair of cities in the network is the stretch factor the largest?" If this authority has the resources to build some highway segments, a useful query would be: "Which edge (or k edges) should be added to the network to achieve the greatest decrease in stretch factor and/or load factor?" While budgeting for future improvements to the highway network, a planner may ask: "What is the total length of the edges to be added to the network to achieve a desired fault-tolerance and stretch factor without destroying the planarity?" Planning for emergency situations requires analysis of the fault-tolerance of the network, and this may provoke a query of the type: "What is the maximum increase in stretch factor of the network (assuming it remains connected) if all

[1] A 16-km bridge across the Øresund connecting the two cities was opened to traffic during the summer of 2000.

highway segments within a 50-mile radius of some point are unusable due to a natural disaster?"

The list of desirable properties in a "good" network reflects many contradictory needs. For example, if bounded degree networks are desired, then small diameters are not possible; if small stretch factor networks are needed, then arbitrarily small size or weight may not be possible. Thus many network design problems display interesting tradeoffs between the quality measures and they can be thought of as multicriteria optimization problems. It is clear that a versatile software package with a suite of network design and analysis tools can be invaluable in making complex, practical decisions regarding geometric networks.

The network design problem encompasses many interesting and fundamental problems. The *minimum spanning tree* can be thought of as a network with least possible weight and infinite stretch factor; if *Steiner* points can be added, then the network with least possible weight and infinite stretch factor is the *Steiner minimum tree*. As the required stretch factor is decreased and approaches one, the network becomes denser until it ultimately becomes the trivial complete graph. If the degree of each site is required to be 2, then the network with least weight is the *traveling salesperson tour* on the points.

1.1.1 Spanning trees

A tree on a set S of n points is an acyclic connected graph on these points. We also call such a graph a *spanning tree* of S. A spanning tree is good in the sense that it has the minimum number of edges.

A set of points has many spanning trees. Sylvester showed in 1857 – and, independently, Cayley in 1889 – that any set of n points has exactly n^{n-2} spanning trees.

Let T be a spanning tree of the set S. The *weight* $wt(T)$ of T is defined to be the sum of the lengths of its edges, where the *length* of an edge $\{p, q\}$ is the Euclidean distance $|pq|$ between p and q. A *minimum spanning tree* $MST(S)$ of S is a spanning tree of minimum weight. A minimum spanning tree on 13,509 US cities is shown in Figure 1.2.

Property 1.1.2. *A minimum spanning tree of a set S is a* shortest *network connecting the points of S.*

In particular, Property 1.1.2 states the obvious fact that a shortest connected network must be a tree. Thus, a minimum spanning tree is good in the sense that both its number of edges and its weight are minimum. The following property states that it is also good in the sense that it has a small degree. The proof of this property is left as an exercise (see Exercise 4.3).

Property 1.1.3. *In a minimum spanning tree of a set of points in the plane, each point has degree at most six.*

In fact, if S is a finite set of points in $\mathbb{R}^d$, where $d \geq 2$, then the degree of each point of the minimum spanning tree of S is bounded from above by a constant that depends only on d.

The first algorithm for computing a minimum spanning tree (of an arbitrary weighted graph) is due to Borůvka and dates back to 1926. In Section 2.6, we will present two other algorithms for computing a minimum spanning tree.

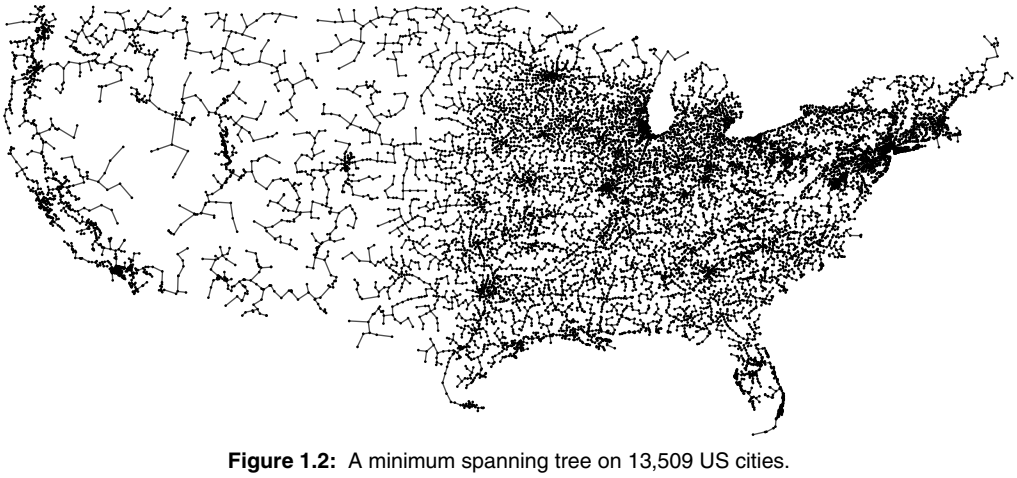

Figure 1.2: A minimum spanning tree on 13,509 US cities.

1.1.2 Steiner trees

According to Property 1.1.2, a minimum spanning tree is a shortest network that connects a point set S. This is not quite true; it is a shortest graph *with vertex set* S that connects these points.

Assume that we consider connected graphs $G = (V, E)$, whose vertex set V contains the set S. In other words, we allow the graph to contain *additional* vertices. The vertices of $V \setminus S$ are called *Steiner points*. Let $SMT(S)$ be such a graph G of minimum weight, where the weight $wt(G)$ of G is defined to be the sum of the lengths of its edges. This graph is called a *Steiner minimum tree* of S, named after Jakob Steiner – a Swiss mathematician who lived from 1796 until 1863. (Apparently, Steiner had nothing to do with Steiner minimum trees.)

The problem of finding a Steiner minimum tree of the vertices of a triangle is called the *Fermat problem*, named after Pierre de Fermat (1601–65). Gauß (1777–1855) computed the Steiner minimum tree of a set of four German cities. His motivation was to link these cities by railroad.

It is clear that $wt(SMT(S)) \leq wt(MST(S))$. Can a Steiner minimum tree be much smaller than a minimum spanning tree? The following lemma shows that the answer is "no."

Lemma 1.1.4. *Let S be a finite set of points in $\mathbb{R}^d$. Then*

$$wt(MST(S)) \leq 2 \cdot wt(SMT(S)).$$

PROOF Let T be a Steiner minimum tree for S. We will construct a spanning tree T' of S having weight at most twice the weight of T. This will prove the lemma, because the minimum spanning tree of S has weight at most that of T'.

Here is the construction. We double each edge of T. This gives a multigraph W connecting the points of S and the Steiner points of T. The degree of each vertex in W is even. Hence, W contains an *Euler tour*[2] W', which is a tour that visits each edge of W exactly once and that returns to the starting vertex. Observe that this tour may visit vertices more than once. It is clear that the weight of W' is equal to that of W, which in turn is equal to twice the weight of T.

[2] Named after Euler (1707–83), the father of graph theory.

We will construct from W' a spanning tree T' of the points of S. The basic operation is that of *short-cutting*: Assume the Euler tour moves from point a to point b, and then to point c. Then short-cutting means that from point a, we immediately move to point c. By the triangle inequality, such a short-cut operation gives a tour – in general not along all points – having weight at most the weight of W'.

By following the tour W', we repeatedly make short-cuts, in such a way that we (i) remove all Steiner points, and (ii) for each point p of S, remove all visits to p, except the first one. The result is a path that visits each point of S exactly once, and whose weight is at most twice the weight of T. This path is clearly a spanning tree of S; it is the tree T' we were looking for. ■

Thus, the weight of a minimum spanning tree is at most twice that of a Steiner minimum tree. Is the factor 2 best possible? We will see in Exercise 1.5 that, for point sets in the two-dimensional plane, a factor smaller than $2/\sqrt{3}$ is not possible. In 1968, Gilbert and Pollack conjectured that $2/\sqrt{3}$ is in fact the best possible factor. That is, they conjectured that for any finite set S of points in $\mathbb{R}^2$,

$$wt(MST(S)) \leq \frac{2}{\sqrt{3}} wt(SMT(S)).$$

This remained an open problem for over 20 years until Du and Hwang settled the conjecture in 1990.

We mention that Steiner minimum trees are extremely hard to compute; the problem is known to be **NP**-hard. In fact, it is not known if the decision problem for the Euclidean metric is in **NP**. On the other hand, a minimum spanning tree of n points in the plane can be computed in $O(n \log n)$ time.

1.1.3 The traveling salesperson tour

The *traveling salesperson problem* is an important problem that influenced the blossoming of fields such as operations research, polyhedral combinatorics, probabilistic analysis, and complexity theory. The *traveling salesperson tour* $TSP(S)$ of a finite set S of points is the shortest tour that visits each point of S exactly once, and returns to the starting point. Here the assumption is that the set S of points belongs to a metric space. The problem of computing an optimal length tour is **NP**-hard, even when the points lie in Euclidean space. In terms of approximation algorithms, Rosenkrantz, Stearns, and Lewis showed that a factor 2 approximation to the optimal tour for points in an arbitrary metric space can be obtained from the minimum spanning tree; their argument is basically the one that we presented in the proof of Lemma 1.1.4. By combining minimum spanning trees with minimum weight matchings, Christofides improved the approximation factor to 3/2. In 1996, Arora (and, independently, Mitchell) improved on these results, by designing a polynomial-time approximation scheme for the Euclidean case. An improved polynomial-time approximation scheme by Rao and Smith in 1998 made use of the concept of spanners; details will be given in Chapter 19.

1.1.4 Triangulations

Let S be a set of n points in the plane. A *triangulation* of S is a partition of the convex hull of S into triangles, such that the vertices of these triangles are exactly the points of S.

Since a triangulation is a planar graph, it follows from Euler's theorem that it has at most $3n - 6 = O(n)$ edges. Observe also that a triangulation is a connected graph. Therefore, it is a sparse network connecting the points of S.

In general, a point set can have many triangulations. Santos and Seidel proved in 2003 that any set of n points in the plane has $O(59^n)$ many triangulations. Some triangulations have special properties:

- The Delaunay triangulation, which is the dual of the Voronoi diagram.
- The minimum weight triangulation, which is a triangulation, whose weight is minimum among all triangulations of the point set. In 2006, Mulzer and Rote proved that the problem of computing this triangulation is **NP**-hard.
- The greedy triangulation, which is defined as follows: Sort all $\binom{n}{2}$ edges of the complete graph in increasing order of their lengths. Start with a graph G whose edge set is empty, and consider the edges in sorted order, one after another. Add the current edge to G if and only if it does not intersect any edge already contained in G.

1.2 The topic of this book: Spanners

In this book, we will mainly be concerned with the problem of designing algorithms that compute geometric networks whose stretch factor is bounded. As we have mentioned already, such networks will be referred to as spanners.

Definition 1.2.1 (Spanner). Let S be a set of n points in $\mathbb{R}^d$ and let $t \geq 1$ be a real number. A *t-spanner* for S is an undirected graph G with vertex set S, such that for any two points p and q of S, there is a path in G between p and q, whose length is less than or equal to $t|pq|$. Any path satisfying this condition is called a *t-spanner path* between p and q.

In Figure 1.3, six geometric networks on 532 US cities are shown, each of which is a t-spanner for a different value of t. These spanners were computed by the path-greedy algorithm that will be presented in Section 1.4.

Definition 1.2.1 considers a spanner to be an undirected graph. Sometimes, it is useful to consider *directed* spanners:

Definition 1.2.2 (Directed spanner). Let S be a set of n points in $\mathbb{R}^d$ and let $t \geq 1$ be a real number. A *directed t-spanner* for S is a directed graph G with vertex set S, such that for any two points p and q of S, there is a directed path in G from p to q, whose length is less than or equal to $t|pq|$. Any path satisfying this condition is called a *directed t-spanner path* from p to q.

If G is a t-spanner for the point set S, then obviously, G is also a t'-spanner for any real number t' with $t' > t$. This leads to the following definition:

Definition 1.2.3 (Stretch factor). Let S be a set of n points in $\mathbb{R}^d$ and let G be a Euclidean graph with vertex set S. The *stretch factor* of G is the smallest real number t such that G is a t-spanner of S.

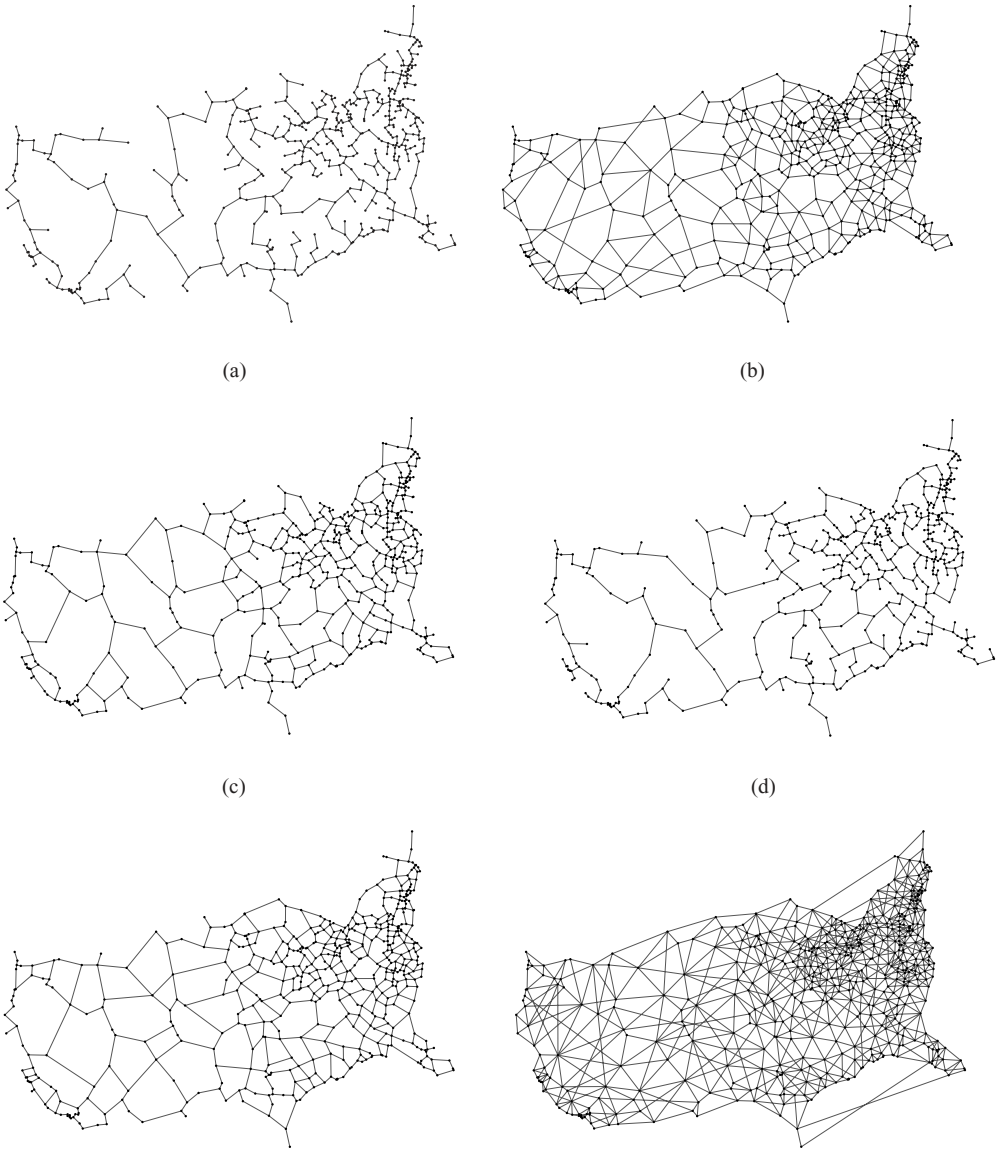

Figure 1.3: Six geometric t-spanner networks on 532 US cities with stretch factors of (a) 10, (b) 5, (c) 3, (d) 2, (e) 1.5, and (f) 1.2, respectively.

Property 1.1.1 and the definition of the minimum spanning tree imply the following property:

Property 1.2.4. *Let S be a set of n points in $\mathbb{R}^d$ and let $t \geq 1$ be a real number. Any t-spanner of S has at least $n-1$ edges, and weight at least $wt(MST(S))$.*

The complete graph is a 1-spanner, but it has a quadratic number of edges. In fact, if we assume that no three points of S are on a line, then the complete graph is the only 1-spanner for S. Therefore, t-spanners are in general interesting only for values of t that are larger

than 1. As you may expect, we are interested in *sparse* spanners, that is, t-spanners with only $O(n)$ edges.

Basic spanner problem: Let S be a set of n points in $\mathbb{R}^d$, and let $t > 1$ be a real number. Does there exist a t-spanner for S having at most $c_{td}n$ edges, where c_{td} is a real number that depends only on t and d? If so, how much time does it take to compute such a t-spanner?

In this book, we will see that the answer to the first question is "yes": Sparse t-spanners exist for values of t that are arbitrarily close to one. Moreover, such t-spanners can be computed in $O(n \log n)$ time, where the constant in the Big-Oh bound depends on the stretch factor t and the dimension d.

In later chapters, we will also consider the existence and construction of sparse t-spanners having one or more of the following additional properties:

More spanner problems: Let S be a set of n points in $\mathbb{R}^d$, and let $t > 1$ be a real number.

1. Does there exist a t-spanner for S, in which each point has a *degree* that depends only on t and d?
2. Does there exist a t-spanner for S, whose *weight* is proportional to the weight of a minimum spanning tree of S?
3. Does there exist a sparse t-spanner for S, whose *spanner diameter* is small? Here, the spanner diameter is defined to be the smallest integer D, such that each pair of points is connected by a t-spanner path containing at most D edges.
4. Can we construct such t-spanners in $O(n \log n)$ time?

It should be clear that the above properties are conflicting. For example, any t-spanner whose degree is bounded by a constant must have spanner diameter $\Omega(\log n)$. The proof of this claim is left as an exercise; see Exercise 1.11.

In most places in this book, the stretch factor t is a real number that is larger than, but arbitrarily close to, 1. We will always assume that the dimension d is a constant. As such, Big-Oh bounds contain factors that depend on t. On the other hand, factors that involve only d will be omitted in these bounds.

1.3 Using spanners to approximate minimum spanning trees

In this section, we present a first application of spanners. Let S be a set of n points in the plane. Since any minimum spanning tree $MST(S)$ of S is contained in the Delaunay triangulation $DT(S)$ of S, we can compute $MST(S)$ in the following way: First, in $O(n \log n)$ time, compute $DT(S)$. Then, compute the minimum spanning tree of $DT(S)$. This minimum spanning tree is in fact a Euclidean minimum spanning tree of the set S. Since $DT(S)$ contains only a linear number of edges, the entire algorithm has a running time of $O(n \log n)$.

For point sets in $\mathbb{R}^d$, where $d \geq 3$, it is unlikely that the same running time can be obtained. In fact, for $d = 3$, it is even unlikely that an algorithm exists that computes a minimum spanning tree within a time bound that is significantly smaller than $n^{4/3}$. This leads to the natural question of whether there exist fast algorithms that *approximate* a minimum spanning tree. The following theorem shows that this question has a positive answer, provided we have a fast algorithm for computing a spanner.

Theorem 1.3.1. *Let S be a set of n points in $\mathbb{R}^d$, let $t > 1$ be a real number, and let G be an arbitrary t-spanner for S. A minimum spanning tree of G is a t-approximate minimum spanning tree of S, that is, the weight of any minimum spanning tree of G is at most $t \cdot wt(MST(S))$.*

PROOF Let T be a minimum spanning tree of S, and number its edges arbitrarily as $e_1, e_2, \ldots, e_{n-1}$. Consider edge e_i. Since G is a t-spanner for S, there exists a t-spanner path P_i in G between the endpoints of e_i. Thus, the length $wt(P_i)$ of P_i is at most t times the length of edge e_i. It follows that

$$\sum_{i=1}^{n-1} wt(P_i) \leq \sum_{i=1}^{n-1} t \cdot wt(e_i) = t \cdot wt(MST(S)).$$

Let G' be the subgraph of G, whose edge set is the union of the edge sets of the paths P_i, $1 \leq i \leq n-1$. Then G' is a connected graph with vertex set S, and its weight is at most $t \cdot wt(MST(S))$. Since the weight of a minimum spanning tree of G is less than or equal to the weight of G', the proof is complete. ■

1.4 A simple greedy spanner algorithm

At this point, it is not clear whether for any set S of n points in $\mathbb{R}^d$, and for any real number $t > 1$, a t-spanner for S having a subquadratic number of edges actually exists. In this section, we present a simple algorithm that, in fact, computes such a spanner.

As discussed before, t-spanners generalize the notion of spanning trees, which require to have paths connecting every pair of points. A t-spanner is required to have reasonably short paths between every pair of points. This observation suggests that an algorithm to construct t-spanners can be obtained by generalizing Kruskal's minimum spanning tree algorithm (which will be discussed in Section 2.6.1). We remark that this spanner algorithm does not necessarily compute a t-spanner of minimum weight.

Simple greedy spanner construction: Generalizing Kruskal's minimum spanning tree algorithm gives a greedy algorithm for constructing spanners. The algorithm starts with a graph G having vertex set S and whose edge set is empty. It considers all pairs of distinct points of S in nondecreasing order of their distances. The decision whether or not to add the current pair $\{p, q\}$ as an edge to G is made as follows: Instead of checking whether the vertices p and q are connected, check whether they have a path of length at most $t|pq|$ that connect them in G.