Spanning Trees and Spanners

David Eppstein*

Dept. Information and Computer Science
University of California, Irvine, CA 92717
http://www.ics.uci.edu/~eppstein/

Tech. Report 96-16

May 8, 1996

Abstract

We survey results in geometric network design theory, including algorithms for constructing minimum spanning trees and low-dilation graphs.

1 Introduction

This survey covers topics in *geometric network design* theory. The problem is easy to state: connect a collection of *sites* by a "good" network. For instance, one may wish to connect components of a VLSI circuit by networks of wires, in a way that uses little surface area on the chip, draws little power, and propagates signals quickly. Similar problems come up in other applications such as telecommunications, road network design, and medical imaging [1]. One network design problem, the Traveling Salesman problem, is sufficiently important to have whole books devoted to it [79]. Problems involving some form of geometric minimum or maximum spanning tree also arise in the solution of other geometric problems such as clustering [12], mesh generation [56], and robot motion planning [93].

One can vary the network design problem in many ways. The space from which the sites are drawn is often the Euclidean plane, but other metrics and higher dimensions are possible. One may wish to construct networks of varying classes such as trees, planar graphs, or general graphs. One may restrict the vertices of the network to be at the given sites or instead allow additional *Steiner points*. One may measure the quality of a network in different ways; typical quality measures include the *weight* (total length of all edges in a network), *diameter* (longest network distance between two sites), and *dilation* (largest ratio of network distance to Euclidean distance). The problem may be *static*, or various types of *dynamic* changes to the collection of sites may be possible.

Geometric network design problems can generally be solved by translation to a graph problem: simply form a *complete geometric graph* in which each pair of sites is connected by an edge with length equal to the distance between the pair. We are interested here in solutions which depend intrinsically on the geometry of the problem, and don't immediately reduce a geometric problem to a graph problem.

This survey is organized from the specific to the general: we first describe algorithms for finding trees of various types, then algorithms for finding planar graphs, and finally we describe algorithms for finding graphs without restriction to a specific class. Within a class of graphs we organize the topics by the criteria used to measure the quality of a given graph. We only describe problems of designing networks with vertices limited to the initial sites; Steiner tree and Steiner triangulation problems have been extensively surveyed elsewhere [17, 19, 48].

2 Trees

Much of the research on geometric network design problems has involved problems in which the network to be designed is a tree. Such problems include the minimum spanning tree, maximum spanning tree, minimum diameter spanning tree, bounded degree spanning trees (such as the traveling salesman path), and the k-point minimum spanning tree.

VLSI design theory is responsible for several more specialized geometric spanning tree problems, which we do not discuss in detail: Steiner min-max spanning trees [36], minimum skew spanning trees (for routing clock signals) [69, 70, 110], and minimum area routing of interdigitated power and ground networks [67]. For more discussion of these and other geometric VLSI problems, see Sherwani [107]. Much of the work on these problems provides only heuristics for their solution, so there may be room for more work on exact algorithms.

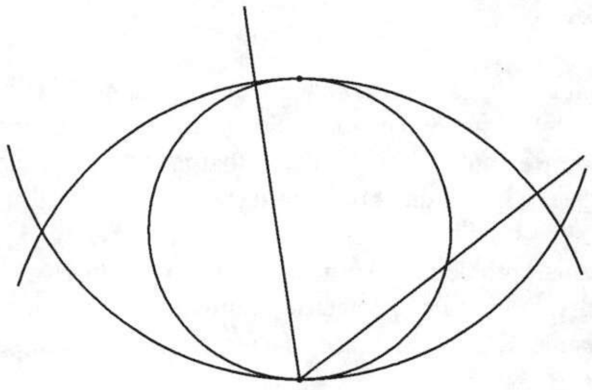

Figure 1. The lune of two points, their diameter circle, and a 60° wedge.

2.1 Minimum weight spanning trees

The most basic and well-studied network design problem is that of finding minimum spanning trees: connect the sites into a tree with minimum total weight. Many graph minimum spanning tree algorithms have been published, with the best taking linear time in either a randomized expected case model [71] or with the assumption of integer edge weights [61]. For deterministic comparison-based algorithms, slightly superlinear bounds are known [62]. Therefore, by constructing a complete geometric graph, one can find a geometric minimum spanning tree in time $O(n^2)$, in any metric space for which distances can be computed quickly.

Faster algorithms for most geometric minimum spanning tree problems are based on a simple idea: quickly find a sparse subgraph of the complete geometric graph, that is guaranteed to contain all minimum spanning tree edges. The details of this approach vary depending on the specific problem; for planar problems the Delaunay triangulation is the appropriate graph to use, while for higher dimensional problems a more complicated recursive decomposition works better.

2.1.1 Planar MSTs

In the plane, one method of finding a subgraph containing the minimum spanning tree is provided by the following well-known result. Define the *lune* of two points u and v to be the region interior to the two circles having line segment uv as radius.

Lemma 1. Suppose uv is an edge in the minimum spanning tree of a planar point set. Then the lune of u and v contains no other site.

Proof: Let T be a tree containing uv, and let w be some other site in the lune of u and v. Then the lengths of segments wu and wv are both shorter than that of segment uv. The removal of uv from T would partition it into two connected components U (containing u) and V (containing v). If w is in U, $T - uv + wv$ would be a tree, and if w is in V, $T - uv + uw$ would be a tree, in either case having smaller weight than T. So if T is to be minimum, there can be no such point w in the lune. □

The set of all edges uv with empty lunes is known as the *relative neighborhood graph*. The lemma above can be rephrased as stating that the minimum spanning tree is a subgraph of the relative neighborhood graph. Clearly, the lune contains the *diameter circle* of uv. It also contains the region formed by intersecting a circle centered at u with radius uv, with any 60° wedge having apex at u and containing v. (Both shapes are depicted in Figure 1.) As a consequence we have the following results.

Lemma 2 (Shamos and Hoey [106]). The minimum spanning tree of a point set is a subgraph of the points' Delaunay triangulation.

Proof: An edge is in the Delaunay triangulation if some circle contains the two endpoints and no other site. This holds for the minimum spanning tree edges and their diameter circles. □

Lemma 3 (Yao [115]). Partition the set of planar unit vectors into subsets, no one subset containing two vectors at angles 60° or greater to each other. For each site u, and each subset S of the unit vectors, find the nearest other site v such that uv has slope in S. Then the collection of all such pairs (u, v) forms a graph containing the minimum spanning tree of the sites.

Proof: The set of points forming slopes in S from u forms a subset of a 60° wedge, and as we have seen the portion of that wedge closer to u than is v is a subset of the lune of u and v. □

All three of these lemmas can be used to find the minimum spanning tree quickly:

Theorem 1. We can find the minimum spanning tree of a planar point set in time $O(n \log n)$.

Proof: By any of various methods, one constructs the relative neighborhood graph, Delaunay triangulation, or *Yao graph* described in Lemma 3 in time $O(n \log n)$. Any of the classical graph minimum spanning tree algorithms can then find the minimum spanning tree of these graphs in time $O(n \log n)$ or better. (The first two graphs are planar, so Borůvka's algorithm or a method of Cheriton and Tarjan [29] can be used to find their minimum spanning trees in linear time.) By the lemmas above, the minimum spanning trees of these graphs are exactly the geometric minimum spanning trees of the points. □

2.1.2 Higher dimensional MSTs

The methods described above, for constructing graphs containing the minimum spanning tree, all generalize to higher dimensions. However Lemma 2 is not so informative, because the Delaunay triangulation may form a complete graph. Lemma 3 is more useful; Yao [115] used it to find minimum spanning trees in time $O(n^{2-\epsilon_d})$ where for any dimension d, ϵ_d is a (very small) constant.

Agarwal et al. [2] found a more efficient method for high dimensional minimum spanning trees, via *bichromatic nearest neighbors*. If we are given two sets of points, one set colored red and the other colored blue, the bichromatic nearest neighbor pair is simply the shortest red-blue edge in the complete geometric graph. It is not hard to show that this edge must belong to the minimum spanning tree, so finding bichromatic nearest neighbors is no harder than computing minimum spanning trees. Agarwal et al. show that it is also no easier; the two problems are equivalent to within a polylogarithmic factor.

The intersection of any d halfspaces forms a *simplicial cone*, with an *apex* where the three bounding hyperplanes meet. We define a *double cone* to be the union of two simplicial cones, where the halfspaces defining the second cone are opposite those defining the first cone on the same bounding hyperplanes. Such a double cone naturally defines a pair of point sets, one in each cone. Define the *opening angle* of a cone to be the maximum angle uvw where v is the apex and u and w are in the cone. The opening angle of a double cone is just that of either of the two cones forming it.

Lemma 4 (Agarwal et al.). There is a constant α such that, if pq is a minimum spanning tree edge, and PQ are the points in the two sides of a double cone with opening angle at most α, with $p \in P$ and $q \in Q$, then pq is the bichromatic nearest neighbor pair of P and Q.

The proof involves using Lemma 3 to show that p and q must be mutual bichromatic nearest neighbors. It then uses geometric properties of cones to show that, if there were a closer pair $p'q'$, then pp' and qq' would also have to be smaller than pq, contradicting the property of minimum spanning trees that any two points are connected by a path with the shortest possible maximum edge length.

One can then use this result to find a graph containing the minimum spanning tree, by solving a collection of bichromatic closest pair problems defined by a sequence of double cones, such that any edge of the complete geometric graph is guaranteed to be contained by some double cone.

Lemma 5 (Agarwal et al.). Given a set of n points in R^d, we can form a hierarchical collection of $O(n \log^{d-1} n)$ bi-chromatic closest pair problems, so that each point is involved in $O(i^{d-1})$ problems of size $O(n/2^i)$ $(1 \le i \le \log n)$ and so that each MST edge is the solution to one of the closest pair problems.

Proof: For simplicity of exposition we demonstrate the result in the case that $d = 2$; the higher dimensional versions follow analogously.

If pq is a minimum spanning tree edge, and w is a double wedge having sufficiently small interior angle, with p in one half of w and q in the other, then pq must have the minimum distance over all such pairs defined by the points in w. Therefore if F is a family of double wedges with sufficiently small interior angles, such that for each pair of points (p, q) some double wedge $w(p, q)$ in F has p on one side and q on the other, then every MST edge pq is the bichromatic closest pair for wedge $w(p, q)$.

Suppose the interior angle required is $2\pi/k$. We can divide the space around each point p into k wedges, each having that interior angle. Suppose edge pq falls inside wedge w. We find a collection of double wedges, with sides parallel to w, that is guaranteed to contain pq. By repeating the construction k times, we are guaranteed to find a double wedge containing each possible edge.

For simplicity, assume that the sides of wedge w are horizontal and vertical. In the actual construction, w will have a smaller angle than $\pi/2$, but the details are similar. First choose a horizontal line with at most $n/2$ points above it, and at most $n/2$ points below. We continue recursively with each of these two subsets; therefore if the line does not cross pq, then pq is contained in a closest pair problem generated in one of the two recursive subproblems. At this point we have two sets, above and below the line. We next choose a vertical line, again dividing the point set in half. We continue recursively with the pairs of sets to the left of the line, and to the right of the line. If the line does

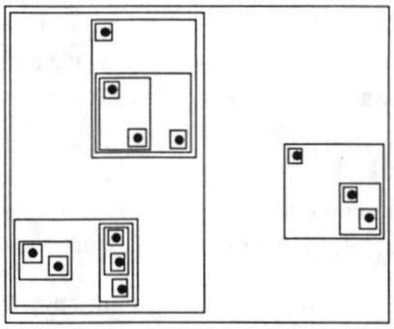

Figure 2. A fair split tree.

not cross pq, then pq will be covered by a recursive subproblem. If both lines crossed pq, so that it was not covered by any recursive subproblem, then $w(p, q)$ can be taken to be one of two bichromatic closest pair problems formed by opposite pairs of the quadrants formed by the two lines.

The inner recursion (along the vertical lines) gives rise to one subproblem containing p at each level of the recursion, and each level halves the total number of points, so p ends up involved in one problem of each possible size $n/2^i$. The outer recursion generates an inner recursion at each possible size, giving i problems total of each size $n/2^i$. The construction must be repeated for each of the k wedge angles, multiplying the bounds by $O(1)$. □

Theorem 2 (Agarwal et al.). We can compute the Euclidean minimum spanning tree of a d-dimensional point set in randomized expected time $O((n \log n)^{4/3})$ for $d = 3$, or deterministically in time $O(n^{2-2/(\lceil d/2 \rceil+1)+\epsilon})$ in any dimension, for any fixed constant ϵ.

Proof: We use Lemma 5 to find a collection of bichromatic closest pair problems, which can be solved in the given time bounds. The resulting closest pairs form a graph with $O(n \log^{O(1)})$ edges, from which the minimum spanning tree can easily be found in time $O(n \log^{O(1)})$. □

2.1.3 Approximate MSTs

The method above for high dimensional minimum spanning trees takes close to quadratic time when the dimension is large. We describe now a method for approximating the minimum spanning tree, within any desired accuracy, much more quickly. This method is due to Callahan [25, 26], and is based on his technique of *well-separated pair decomposition*. We will use this same method later in the construction of spanners. Callahan also uses this technique to compute exact minimum spanning trees from bichromatic pairs, somewhat more efficiently than the reduction of Agarwal et al. described above, by using bichromatic nearest neighbor computations to merge clusters of points; we omit the details. Similar minimum spanning treee approximation algorithms with worse tradeoffs between time and approximation quality were previously found by Vaidya [111] and Salowe [100].

Callahan's method is to recursively partition the given set of sites by axis-parallel hyperplanes. This produces a binary tree T, in which each node corresponds to a subset of points contained in some

d-dimensional box. For any node v of T, corresponding to some set $S(v)$ of points, define $R(v)$ to be the smallest axis-aligned box containing the same point set, and let $\ell(v)$ denote the length of the longest edge of $R(v)$. Then Callahan calls T a *fair split tree* if it satisfies the following properties:

- For each leaf node v of T, $|S(v)| = 1$.
- For each internal node v, one can choose the axis-parallel hyperplane separating its children's points to be at distance at least $\ell(v)/3$ from the sides of $R(v)$ parallel to it.

Figure 2 depicts a fair split tree. Such a tree can be constructed in time $O(n \log n)$ (for any fixed dimension) via any of several algorithms, for instance using methods for quadtree construction [20].

Callahan now defines a *well-separated pair* decomposition (for a given parameter s) to be a collection of pairs of nodes of a fair split tree, having the properties that

- For any pair of sites (p, q) in the input, there is exactly one pair of nodes (P, Q) in the decomposition for which p is in $S(P)$ and q is in $S(Q)$.
- For each pair (P, Q) in the decomposition, there is a length r such that $S(P)$ and $S(Q)$ can be enclosed by two radius-r balls, separated by a distance of at least sr.

The connection between this definition, minimum spanning tree edges, and bichromatic pairs, is explained by the following result.

Lemma 6 (Callahan). Let pq be a minimum spanning tree edge, and (P, Q) be a pair of nodes in a well-separated decomposition of the input (with $s = 2$) for which p is in $S(P)$ and q is in $S(Q)$. Then pq must be the bichromatic closest pair connecting $S(P)$ and $S(Q)$.

Proof: This follows from the well-known fact that the minimum spanning tree solves the *bottleneck shortest path problem*, of connecting two points p and q by a path with the shortest possible maximum edge length. If there were a shorter edge $p'q'$ connecting $S(P)$ and $S(Q)$, one could find a path p-p'-q'-q using shorter edges than pq, contradicting this property of minimum spanning trees. $\square$

As a consequence one can find minimum spanning trees by solving a collection of bichromatic closest pair problems for the pairs of a well-separated pair decomposition, then compute a minimum spanning tree on the resulting graph. However for this approach to be useful, one must bound the complexity of the well separated pair decomposition.

Callahan constructs his well separated pair decomposition as follows. The method is recursive, and constructs a collection of pairs of sets covering all edges connecting two point sets, each represented by a node of T. Initially, we call this recursive algorithm once for the two children of each node of T. The recursive algorithm, if presented with well-separated nodes P and Q of T, simply returns one pair (P, Q). Otherwise, it swaps the two nodes if necessary so that P is the one with the larger bounding box, and calls itself recursively on node pairs (P_1, Q) and (P_2, Q) where P_i are the two children of P.

Intuitively, each node of T is then involved only in a small number of pairs, corresponding to nodes with similar sized bounding boxes within a similar distance from the node.

Lemma 7 (Callahan). The procedure described above results in a well-separated pair decomposition consisting of $O(n)$ pairs, and can be performed in time $O(n)$ given a fair split tree of the input.

This information is not enough to reduce the problem of finding exact minimum spanning trees to that of bichromatic closest pairs; it may be, for instance, that the decomposition involves $\Omega(n)$ pairs (P, Q) for which $|S(P)| = 1$ and $|S(Q)| = \Omega(n)$; the total size of all subproblems would then be quadratic. (Callahan does use this decomposition in a more complicated way to reduce minimum spanning trees to bichromatic closest pairs). However if we only want an approximate minimum spanning tree, the problem becomes much easier.

Theorem 3 (Callahan). For any fixed ϵ, one can compute in $O(n \log n)$ time a tree having total length within a factor of $(1 + \epsilon)$ of the Euclidean minimum spanning tree.

Proof: Compute a well-separated pair decomposition for $s = \max(2, 4/\epsilon)$ and form a graph G by choosing an edge pq for each pair (P, Q) in the decomposition. Note that pq can have length at most $r(4 + s)$ and any other edge connecting the same pair must have length at least rs, so pq approximates the bichromatic closest pair within $(1 + \epsilon)$. Replacing each edge of the true minimum spanning tree by the corresponding edge in G produces a subgraph S of G with weight within $(1 + \epsilon)$ of the minimum spanning tree.

We now prove by induction on the length of the longest edge in the corresponding minimum spanning tree path that any two sites are connected by a path in S. Specifically, let $e = pq$ be the longest edge in the minimum spanning tree path connecting the sites; then the corresponding edge in G connects two points $p'q'$ in the same pair (P, Q) of the decomposit.on, There is a path of edges all shorter than e from one of the two sites to p' (via the minimum spanning tree to p, then within the cluster to p', using the fact that $s \geq 2$ to prove that this last step is shorter than e). By the induction hypothesis, one can find such a path using only edges in S. Similarly there is a path of edges in S all shorter than e from q' to the other site. Therefore S is connected, and must be a spanning tree.

The minimum spanning tree of G must then be shorter than S, so it approximates the true minimum spanning tree of the sites within a $(1 + \epsilon)$ factor. $\square$

By choosing more carefully the representative edges from each pair in the decomposition, Callahan achieves a time bound of $O(n(\log n + \epsilon^{-d/2} \log \epsilon^{-1}))$ for this approximation algorithm.

2.1.4 Incremental and Offline MSTs

In some applications one wishes to compute the minimum spanning trees of a sequence of point sets, each differing from the previous one by a small change, such as the insertion or deletion of a point. This can be done using similar ideas to the static algorithms described above, of constructing a sparse graph that contains the geometric minimum spanning tree. Such a graph graph will also change dynamically, and we use a dynamic graph algorithm to keep track of its minimum spanning tree. As with the static problem, the specific graph we use may be a Yao graph, Delaunay triangulation, or bichromatic closest pair graph, depending on the details of the problem.

We start with the simplest dynamic minimum spanning tree problem, in which only insertions are allowed to a planar point set. In this case, the graph of choice is the *Yao graph* described earlier, in which we compute for each point the nearest neighbor in each of six 60° wedges.

Lemma 8. Given a set S of n points, and a given 60° wedge of angles, in $O(n \log n)$ time we can construct a data structure which can determine in $O(\log n)$ time the nearest point in S to a query point x, and forming an angle from x within the given wedge.

Proof: We construct an Voronoi diagram for the convex distance function consisting of the Euclidean distance for points within the wedge, and infinite distance otherwise. This diagram can be found in time $O(n \log n)$ using any of several algorithms [34, 47, 88]. We then perform planar point location in this diagram; again, various algorithms are possible [49, 77, 84, 103]. □

We can then apply the static-to-dynamic transformation for decomposable search problems [15] to this data structure, producing an incremental data structure in which we can insert points to S and answer the same sorts of queries, both in time $O(\log^2 n)$.

Theorem 4. We can maintain the Euclidean minimum spanning tree of a set of points in the plane, subject to point insertions only, in time $O(\log^2 n)$ per update.

Proof: We maintain a sparse graph G containing the minimum spanning tree. We then use the above data structure to determine for each new point six candidate edges by which it might be connected to the minimum spanning tree, and add these edges to G. Insertions to a dynamic graph minimum spanning tree problem can be handled in logarithmic time each, using a data structure of Sleator and Tarjan [108]. □

A more complicated technique based on the same Yao graph idea can be used to handle an *offline* sequence of both insertions and deletions, in the same time bound [52].

2.1.5 Fully dynamic MSTs

The fully dynamic case, in which we must handle both insertions and deletions *online* (without knowing the sequence of events in advance) is much harder. A method of Eppstein [53] can be used to maintain the solution to a bichromatic closest pair problem; combining this with the reduction of Lemma 5 and a fully dynamic graph minimum spanning tree algorithm [59] gives a method with amortized running time $O(n^{1/2} \log^d n + n^{1-2/(\lceil d/2 \rceil+1)+\epsilon})$ per update, where d denotes the dimension of the problem. However this time bound is too large, and the algorithm too complicated, to be of practical interest. We instead describe in more detail a technique that works well in a certain average case setting defined by Mulmuley [92] and Schwarzkopf [104].

We define a *signature* of size n to be a set S of n input points, together with a string s of length at most $2n$ consisting of the two characters "+" and "−". Each "+" represents an insertion, and each "−" represents a deletion. In each prefix of s, there must be at least as many "+" characters as there are "−" characters, corresponding to the fact that one can only delete as many points as one has already inserted. Each signature determines a space of as many as $(n!)^2$ update sequences, as follows. One goes through the string s from left to right, one character at a time, determining one update per character. For each "+" character, one chooses a point x from S uniformly at random among those points that have not yet been inserted, and inserts it as an update in the dynamic problem. For each "−" character, one chooses a point uniformly at random among those points still part of the problem, and updates the problem by deleting that point. For any signature, we define the *expected time* for a given algorithm on that signature to be the average time taken by that algorithm among all possible update sequences determined by the signature. The algorithm is not given the signature, only the actual sequence of updates. We then define the expected time of the algorithm on inputs of size n to be the maximum expected time on any signature of size n. In other words, we choose the signature to force the worst case behavior of the algorithm, but once the signature is chosen the algorithm can expect the update sequence to be chosen randomly from all sequences consistent with the signature. Note that this generalizes the concept of a *randomized incremental* algorithm, since the expected case model for such algorithms is generated by signatures containing only the "+" character.

To demonstrate the power of this expected case model, and derive a fact we will need in our minimum spanning tree algorithm, we prove the following result. Compare this with the $\Theta(n)$ worst case bound on the amount of change per update in the Delaunay triangulation.

Lemma 9 (Mulmuley). For any signature of size n, the expected number of edges in the Delaunay triangulation that change per update is $O(1)$.

Proof: We first consider the change per insertion. Suppose after an insertion at some step i of the algorithm, there are exactly j points remaining. Since the Delaunay triangulation is a planar graph, it will have at most $3j - 6$ edges. Each edge will have just been added to the graph if and only if one of its endpoints was the point just inserted, which will be true with probability $2/j$. So the expected number of additional edges per insertion is at most $(3j - 6 \cdot 2/j) = O(1)$. The number of existing edges removed in the insertion is at most proportional to the number of edges added, and can possible be even smaller if the convex hull becomes less complex as a result of the insertion. Thus the total change per insertion is $O(1)$. The total change per deletion can be analysed by a similar argument that examines the graph before the deletion, and computes the probability of each edge being removed in the deletion. $\square$

This fact can then be used to show that the Delaunay triangulation can be maintained efficiently. Deletions can be performed in time proportional to the complexity of the change [4]. Insertions can also be performed in this amount of time as long as the point to be inserted can be located within the existing Delaunay triangulation. This point location can be performed using a data structure based on the history of the structure of the Delaunay triangulation, similar to that used by Guibas et al. [66], and which can be shown to have logarithmic expected search time in this expected case model. With some care, this structure can be updated quickly after each deletion.

Theorem 5. The minimum spanning tree can be maintained fully dynamically for any signature of size n in expected time $O(\log n)$ per update.

Proof: As shown by Mulmuley, by Schwarzkopf, and by Devillers et al. [43], we can maintain the Delaunay triangulation in this time bound. Since the Delaunay triangulation is a planar graph, we can find its minimum spanning tree using a data structure of Eppstein et al. [60]. $\square$

Open Problem 1. Can we maintain the Euclidean minimum spanning tree dynamically in polylogarithmic worst case or amortized time per update?

2.2 Maximum weight spanning trees

For graphs, the maximum spanning tree problem can be transformed to a minimum spanning tree problem and vice versa simply by negating edge weights. But for geometric input, the maximum spanning tree is very different from the minimum spanning tree, and different algorithms are required to construct it. This problem was first considered by Monma et al. [90], and has applications in certain clustering problems [12].

We first examine the edges that can occur in the maximum spanning tree. One might guess, by analogy to the fact that the minimum spanning tree is a subgraph of the Delaunay triangulation, that the maximum spanning tree is a subgraph of the farthest point Delaunay triangulation. Unfortunately this is far from being the case—the farthest point Delaunay triangulation can only connect convex hull vertices, and it is planar whereas the maximum spanning tree has many crossings. However we will make use of the farthest point Delaunay triangulation in constructing the *farthest neighbor forest*, formed by connecting each site to the site farthest away from it.

The first fact we need is a standard property of graph minimum or maximum spanning trees.

Lemma 10. The farthest neighbor forest is a subgraph of the maximum spanning tree.

Lemma 11 (Monma *et al.* [90]). Let each tree of the farthest neighbor forest be two-colored. Then for each such tree, the points of any one color form a contiguous nonempty interval of the convex hull vertices. The trees of the forest can be given a cyclic ordering such that the intervals adjacent to any such interval come from adjacent trees in the ordering.

Lemma 12 (Monma *et al.* [90]). Let $e = (x, y)$ be an edge in the maximum spanning tree but not in the farthest neighbor forest, with x in some farthest point neighbor tree T. Then x and y are both convex hull vertices, and y is in a tree adjacent to T in the cyclic ordering of Lemma 11.

Putting these pieces of information together, we have the following result.

Theorem 6 (Monma *et al.* [90]). The maximum spanning tree of a planar point set can be constructed in $O(n \log n)$ time by computing the farthest neighbor forest, determining the cyclic ordering of Lemma 11, finding the longest edge between each adjacent pair of trees in the cyclic ordering, and removing the shortest such edge.

The farthest neighbor forest and the longest edge between adjacent trees can be computed easily via point location in the farthest point Voronoi diagram.

Eppstein [55] considered the same problem from the average case dynamic viewpoint discussed above. His algorithm performs a similar sequence of steps dynamically: maintaining a dynamic farthest neighbor forest, keeping track of the intervals induced on the convex hull and of the cyclic ordering of the intervals, and recomputing longest edges as necessary between adjacent intervals using a dynamic geometric graph based on the *rotating caliper* algorithm for static diameter computation.

Theorem 7 (Eppstein). The Euclidean maximum spanning tree can be maintained in expected time $O(\log^3 n)$ per update.

Just as higher dimensional minimum spanning trees are closely related to bichromatic nearest neighbors, higher dimensional maximum spanning trees can be related to bichromatic farthest neighbors. Agarwal et al. [3] used this idea to give a randomized algorithm for the three-dimensional maximum spanning tree with expected time complexity $O(n^{4/3} \log^{7/3} n)$, almost matching the bound for the corresponding minimum spanning tree problem. The same authors also provide fast approximation algorithms to three- and higher-dimensional maximum spanning tree problems.

Since the maximum spanning tree involves many edge crossings, it is also natural to consider the problem of finding the maximum planar spanning tree, that is, the maximum weight spanning tree not involving any edge crossings. The exact complexity of this problem appears to be unknown, but Alon et al. [6] point out that it can be approximated to within a factor of two simply by choosing a tree with a star topology (in which one *hub* vertex is connected to all $n-1$ others). Choosing the best possible hub can be done in $O(n^2)$ time; Alon et al. show that a constant factor approximation to the maximum non-crossing spanning tree can be found in $O(n)$ time by choosing a suboptimal hub.

Open Problem 2. What is the complexity of finding the exact maximum weight non-crossing spanning tree? If it is NP-hard, how well can it be approximated in polynomial time?

2.3 Low-degree spanning trees

For points in the plane (with the Euclidean metric), any minimum spanning tree has degree at most six, and a perturbation argument shows that there always exists a minimum spanning tree with degree at most five [91]. In general the degree of a minimum spanning tree in any dimension is bounded by the *kissing number* (maximum number of disjoint unit spheres that can be simultanously tangent to a given unit sphere) [98].

However it is interesting to consider the construction of trees with even smaller degree bounds. As an extreme example, the *traveling salesman path* problem asks us to find a spanning tree with degree at most two. As is well known, one can approximate this to within a factor of two by performing an Euler tour of the graph formed by doubling the edges of a minimum spanning tree. Christofides' heuristic [35] reduces the approximation ratio to 3/2 by forming a smaller Eulerian graph, the union of a minimum spanning tree and a matching on its odd degree vertices. These techniques do not take much advantage of the geometry of the problem, and work for any metric space.

Very recently, Arora (personal communication) has discovered a polynomial time approximation scheme for the planar Traveling Salesman Problem. The basic idea, like that of many recent geometric approximation algorithms, is to use dynamic programming on a structure similar to Callahan's fair split tree. Arora shows that, for any point set, there exists a tour approximating the TSP and a recursive decomposition in which each box is crossed only a small number of times by this approximate tour. One can then find the best possible subtour for each small subset of edges crossing each box, by combining similar information from smaller boxes.

This approximation strategy generalizes to higher dimensional traveling salesman problems as well, but the time bounds grow to quasipolynomial (exponential in a polylogarithmic function of n, with the exponent in the polylogarithm depending on dimension).

One can also consider degree bounds between two and five. Khuller et al. [75] consider this problem for degree bounds of three and four; they show that one can find constrained spanning trees

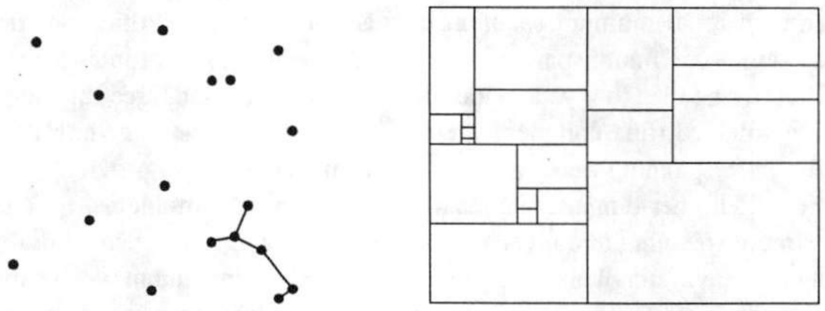

Figure 3. (a) 6-point minimum spanning tree. (b) Guillotine partition .

with length 3/2 and 5/4 times the MST length respectively. They also show that in any dimension one can find a degree-three tree with total length at most 5/3 that of the minimum spanning tree.

The methods of Khuller et al. are based, like the 2-approximation to the TSP, on modifications to the minimum spanning tree to reduce its degree. For instance, their algorithm for the planar degree-three tree problem roots the minimum spanning tree, then finds for each vertex v the shortest path starting at v and visiting all the (at most four) children of v. The union of these paths is the desired degree-three tree.

The exact solutions to the planar degree-three spanning tree problem is NP-hard [95] but apparently the complexity of the degree-four problem remains open.

Open Problem 3. Is it NP-hard to find the minimum weight degree-four spanning tree of a planar point set?

It is also natural to consider maximization versions of these bounded-degree spanning tree problems. Barvinok [14] shows that the adjacency matrix of a complete geometric graph (using a polyhedral approximation to the Euclidean distance function) has "low combinatorial rank" and uses this to approximate the maximum traveling salesman problem within any factor $(1 - \epsilon)$ in polynomial time. Little seems to be known about similar bounded-degree maximum spanning tree problems with larger degree bounds.

Alon et al. [6] consider the maximum non-crossing traveling salesman problem, and its generalization to the construction of a maximum bounded-degree non-crossing spanning tree. They show that this can be approximated by linking the edges of a heavy non-crossing matching formed by projecting the points onto a line, splitting the points into two sets by their median on this line, and matching points of one set with those of the other. The approximation ratio of this method for the non-crossing traveling salesman problem is $1/\pi - \epsilon$; this ratio is not worked out explicitly for the other bounded-degree spanning tree problems but remains a constant. The time used by Alon et al. to find this long non-crossing path is $O(n \log n)$. As with the other non-crossing problems considered by Alon et al., the complexity of finding exact solutions apparently remains open.

2.4 k-point spanning trees

We next consider the *k-minimum spanning tree* problem: given n points in the Euclidean plane, find the shortest tree spanning k of the points (Figure 3(a)).

Up to constant factors in the approximation ratio, the k-MST problem is equivalent to the problem of finding a path connecting k points (the k-TSP problem) or a Steiner tree connecting k points. The choice of Euclidean metric is also not critical. However we will use the k-MST formulation for simplicity. The k-MST problem was introduced independently by Zelikovsky and Lozevanu [116], and by Ravi et al. [97]. Many similar k-point selection problems with other optimization criteria can be solved in polynomial time [42, 58] but the k-MST problem is NP-complete [97, 116] (as are obviously the k-TSP and k-Steiner tree variants), so one must resort to some form of approximation. In a sequence of many papers, the approximation ratio was reduces to $O(k^{1/4})$ [97], $O(\log k)$ [64, 87], $O(\log k/\log\log n)$ [54], $O(1)$ [21], $2\sqrt{2}$ [89], and, very recently, $1+\epsilon$ (Arora, personal communication). We describe the $2\sqrt{2}$ approximation algorithm, but the other results use similar methods. For related work on non-geometric k-MST problems see [13, 22, 30, 97, 116].

Mitchell [89] first restricts his attention to the rectilinear metric in the plane; the weight of a tree in this metric differs by its Euclidean weight by a factor of at most $\sqrt{2}$, so this simplification entails only that factor loss in the overall approximation ratio of his algorithm. With this restriction, one can look for a *rectilinear tree* in which vertices are connected by paths of horizontal and vertical line segments.

Most of the approximation algorithms for the geometric k-MST problem work by using dynamic programming to construct a suitable recursive partition of the plane, and Mitchell's is no exception. Mitchell defines a *guillotine subdivision* to be a recursive partition ot the plane into rectangles, in which each rectangle is subdivided by a vertical or horizontal line segment into two smaller rectangles (Figure 3(b)). For a given rectilinear tree, the *span* of a line segment in the subdivision is the shortest contiguous subsegment containing all the points where the segment intersects the tree. (In other words, this span is a one-dimensional convex hull of the intersection of the line segment with the rectilinear tree.)

The key technical result of Mitchell, the proof of which is too complicated to repeat here, is the following:

Lemma 13 (Mitchell)**.** For any rectilinear graph G, we can find a guillotine subdivision S, such that each edge of G is covered by the spans of segments in S, and such that the weight of these spans is at most twice the weight of G.

Let G be the optimum rectilinear k-point spanning tree; then the lemma shows that there exists a guillotine subdivision, for which the spans of segments form a connected region of the plane covering at least k points, and with weight at most twice that of G. Conversely, if we construct the minimum weight guillotine subdivision with these properties, we can simply take a minimum spanning subgraph of this region of the plane to produce a rectilinear k-point spanning tree with the same weight, which will then be twice that of the optimum rectilinear tree. (Actually, in general this process will form a Steiner tree rather than a spanning tree. One must instead be more careful, and find a guillotine subdivision for which the minimum spanning subtree of the given k points has minimum total weight.)

Thus we have reduced the problem to one of finding an optimum guillotine subdivision. This can be done by dynamic programming: there are $O(n^4)$ combinatorially distinct ways of forming a rectangle containing one or more of the sites. For each of these different rectangles, for each possible number of sites within the rectangle, and for each of the (polynomially many) distinct ways of connecting those sites to the boundary of the rectangle, we finds the optimal guillotine partition within that rectangle by combining information from $O(n)$ pairs of smaller rectangles. The result is a polynomial time algorithm for finding the optimum guillotine partition.

Theorem 8 (Mitchell)**.** In polynomial time, one can find a spanning tree of a subset of k out of n given planar sites, having total weight at most $2\sqrt{2}$ times the optimum.

The complexity of this method, while polynomial, is very high ($O(n^{15}k^2)$ or $O(n \log n + nk^{30})$) but can be reduced somewhat (to $O(n^5k^2)$ or $O(n \log n + nk^{10})$) at the expense of increasing the approximation ratio to a larger constant [89]. Further work remains to be done on reducing the time complexity of this algorithm to a more practical range. In this direction, Eppstein [54] showed that an $O(\log k)$ approximation could be found in time $O(n \log n + nk \log k)$ by combining a similar dynamic programming approach with techniques based on quadtrees.

2.5 Minimum diameter spanning trees

The previous spanning tree problems have all been based on the weight of the tree constructed. We now consider other criteria for the quality of a tree. The *diameter* of a tree is just the length of its longest path. Since geometric spanning trees are often used in applications such as VLSI, in which the time to propagate a signal through the tree is proportional to its diameter, it makes sense to look for a spanning tree of minimum diameter. Ho et al. [68] give an algorithm for this problem, based on the following fact:

Lemma 14 (Ho et al.)**.** Any point set has some minimum diameter spanning tree in which there are at most two interior points.

Proof: We start with any minimum diameter spanning tree T, and perform a sequence of diameter-preserving transformations until it is in the above form. Let P be the longest path in the given tree, and number its vertices $v_1, v_2, \ldots, v_p$.

We first form a forest by removing all edges of P from T, and for each vertex v of P, let T_v denote the tree in this forest containing v. For any other vertex u, let P_u denote the vertex v such that u is in T_v. Then we construct a new tree T' by adding to P an edge from each vertex u to P_u. T' has the same diameter as the original tree, since the distance between any two vertices is unchanged except when they are in the same tree T_v, and in that case (by the assumption that P is a diameter path) the distance of each point to v is less than the distance from v to the endpoints of P.

Now suppose that P has four or more edges and the length of the path v_1-v_2-v_3 is at most half the length of P. (If not, we can reverse P and consider the three vertices at its other end.) Form a tree T'' by removing every edge uv_2 and reconnecting each such vertex u to v_3. This can only decrease the lengths of paths already going through v_3 in T', so the only pairs of vertices with increased path lengths are those newly connected to v_3. But the length of any such path is at most twice the length of the path v_1-v_2-v_3, so the diameter of T'' is no more than that of T.

Each repetition of this transformation decreases the number of edges in P until it is at most three, and preserves the property that each vertex is within one edge of P, so we will eventually reach a tree of the form specified in the lemma. □

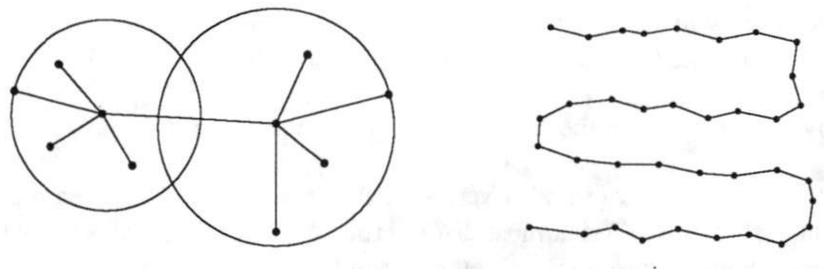

Figure 4. (a) Minimum diameter spanning tree corresponds to cover by two circles. (b) Point set with high diameter minimum spanning tree.

Theorem 9 (Ho et al.). We can find a minimum diameter spanning tree of any point set in time $O(n^3)$.

Proof: We simply try all single interior vertices and pairs of interior vertices. For the latter case, we still need to determine how to assign each remaining point to one of the two interior vertices. If the diameter path of the tree is v_1-v_2-v_3-v_4, and we know the lengths of v_1-v_2 and v_3-v_4, we can perform this assignment by drawing two circles, centered at the two interior vertices, with those lengths as radii (Figure 4(a)); these circles must together cover the whole point set and each point can be assigned to the interior vertex corresponding to the circle covering it. Ho et al. show that, if we sort the points by their distances from v_2 and v_3, the space of all minimal pairs of covering circles can be searched in linear time. These sorted orders can be precomputed in a total of $O(n^2 \log n)$ time. □

Ho et al. also consider optimization of combinations of diameter and total weight; they show that it is NP-complete to find a tree satisfying given bounds on these two measures. It is also of interest to combine these criteria with a bound on vertex degree. Of course for degree-two trees, minimum diameter and minimum weight are equivalent to the traveling salesman problem. The minimum weight spanning tree itself can have very high diameter (an $\Omega(\sqrt{n})$ factor away from optimal); for instance a point set spaced nearly uniformly in a unit square can have a path of length $\Omega(\sqrt{n})$ as its minimum spanning tree (Figure 4(b)). Conversely an upper bound of $O(\sqrt{n})$ follows from results on the worst case length of the minimum spanning tree of n points [18, 109].

One can achieve a better diameter by relaxing the requirement of minimum weight; as we describe later, for two- and three-dimensional point sets, one can find a subgraph of the complete Euclidean graph with degree three and total weight $O(1)$ times that of the minimum spanning tree, for which shortest paths in the subgraph have length within $O(1)$ of the Euclidean distance [37]. A single-source shortest path tree in such a graph is a degree-three tree that has both weight and diameter within a constant of the minimum. Similar problems of constructing spanning trees combining diameter, weight, and bottleneck shortest path bounds were considered by Salowe et al. [102] and Khuller et al. [76].

2.6 Minimum dilation spanning trees

Of the various common geometric network design quality criteria, only dilation is underrepresented among research on the tree problems. (Dilation has been studied for non-geometric tree design problems [24], and as we will see, is very prominent in work on other classes of graph.) This omission is likely because there is little to do in terms of worst case bounds:

Lemma 15. Any spanning tree on the vertices of a regular polygon has dilation $\Omega(n)$.

Proof: As with any tree, we can find a vertex v, the removal of which partitions the tree into components the largest of which has at most $2n/3$ vertices. Therefore there is some pair of vertices from different components, adjacent to each other along the boundary of the polygon, and separated from v by at least $n/6$ polygon edges. The path in this tree connecting this pair passes through v, and so has dilation $\Omega(n)$. □

Conversely, the minimum spanning tree has dilation $O(n)$ by its bottleneck shortest path property. However, the minimum spanning tree is not always good in terms of dilation; an example similar to that of Figure 4(b) shows that it can have dilation $\Omega(n)$ even when dilation $O(\sqrt{n})$ is possible. it is easy to construct spanning trees with dilation $O(\sqrt{n})$.

Open Problem 4. Is it possible to construct the exact minimum dilation geometric spanning tree, or an approximation to it, in polynomial time? Does the minimum dilation spanning tree have any edge crossings? How well is it approximated by the minimum spanning tree?

3 Planar Graphs

3.1 Minimum weight triangulation

The *minimum weight triangulation* problem asks to find a triangulation (that is, a maximal planar straight-line graph on the given set of vertices) that minimizes the total Euclidean edge length. This problem is not known to be NP-hard, nor is it known to be solvable in polynomial time, and the complexity of minimum weight triangulation is one of the few problems left from Garey and Johnson's original list of open problems [63]. However, a generalized problem with non-Euclidean distances is NP-complete [85].

We describe here two very recent developments in the theory of minimum weight triangulations. First, Levcopoulos and Krznaric [81] have shown that one can find a triangulation with total length approximating the minimum to within a (large) constant factor; no such approximation was previously known. Second, Dickerson and Montague have found a method of finding in polynomial time a subgraph of the exact minimum weight triangulation which, empirically, is usually quite large; enough so that moderate sized instances of the problem can now be solved exactly. It is possible that their method in fact gives a polynomial time algorithm for the problem.

3.1.1 MWT Approximation

The two best approximations known to the MWT are those of Plaisted and Hong [96] and of Levcopoulos and Krznaric [81]. Although dissimilar in many ways, an important basic idea is shared by both papers. Rather than finding a triangulation directly, they consider the problem of finding a minimum weight partition into convex polygons with vertices at the input points (MC for short).

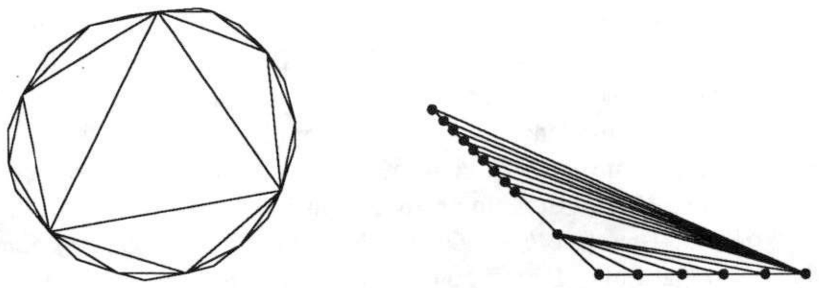

Figure 5. (a) Ring heuristic triangulation of a convex polygon. (b) Greedy triangulation can have $\Omega(\sqrt{n})$ larger weight than minimum.

We first sketch Plaisted and Hong's work. Let v be an interior vertex of a planar straight-line graph with convex faces (such as the MWT of S). We can find a *star* of three or four edges forming no reflex angles at v as follows: choose the first star edge arbitrarily, and choose each successive edge to form the maximal non-reflex angle with the previous edge. Conversely, if each interior vertex of a planar straight-line graph has no reflex angles, the graph must have convex faces. Thus motivated, Plaisted and Hong try to build a graph on S with convex faces by piecing together local structures—stars. For each point s_i in the interior of the convex hull of S, they find the minimum weight star of three or four edges. This collection of minimum weight stars, together with the convex hull of S, forms a graph with total edge length less than twice that of MC and therefore of the MWT. Unfortunately, the resulting graph may not be planar. Plaisted and Hong use a complicated case analysis to remove crossings from this graph, ending up with a planar straight-line graph with convex faces having total edge length at most 12 times that of the MWT. The *ring heuristic* (connecting every other vertex of a convex polygon P to form a polygon P' with $\lfloor n/2 \rfloor$ vertices and a smaller perimeter, and triangulating P' recursively; see Figure 5(a)) can be used to convert this convex partition to a triangulation, at the cost of a logarithmic factor in total length (and hence in the approximation ratio). Olariu et al. [94] showed that this $O(\log n)$ bound is tight.

We now outline the results of Levcopoulos and Krznaric [81]. Their idea is to start with a proof that the *greedy triangulation* (in which one considers the possible edges in increasing order by weight, adding an edge to the triangulation exactly when it does not cross any previously added edge) achieves approximation ratio $O(\sqrt{n})$; an $\Omega(\sqrt{n})$ lower bound was previously known [80, 86] and is depicted in Figure 5(b). Rather than proving this directly, Levcopoulos and Krznaric again work with convex partitions. They define a greedy convex partition GC by constructing stars using only the edges in the greedy triangulation. Unlike the construction of Plaisted and Hong, the result is planar (since it is a subgraph of a planar graph) so no crossings need be removed. The first step of their proof is a lemma that the weight of GC approximates MC within a factor of $O(\sqrt{n})$.

Given any graph G such as MC or GC, define $\max_G(v)$ to be the length of the longest edge incident to v in G; for any planar graph the sum of these quantities is within a constant of the total graph weight. Then if the greedy partition is heavier than the minimum partition by a factor of k, Levcopoulos and Krznaric choose v to maximize $\max_{GC}(v) - \max_{MC}(v)$ among all vertices with $\max_{GC}(v) = \Omega(k \max_{MC}(v))$. Some simple algebra shows that the weight of GC is $O(n \max_{GC}(v))$.

They then use the properties of the greedy triangulation (and the details of their construction of GC from their triangulation) to construct a fan of $\Omega(k)$ edges resembling that in Figure 5(b), in which the short edge of the fan corresponds roughly with $\max_{MC}(v)$ and the long edge corresponds with $\max_{GC}(v)$. From this fan one can find a convex chain of $\Omega(k)$ vertices such that any star at one of these vertices involves an edge of length $\Omega(\max_{GC}(v))$. As a consequence the weight of MC is $\Omega(k \max_{GC}(v))$. Combining these two bounds we see that the weight ratio k of GC to MC is $O(n/k)$. Therefore $k = O(\sqrt{n})$, and this gives an $O(\sqrt{n})$ approximation of MC by the greedy convex partition. Levcopoulos and Krznaric then show that for any convex partition C formed by edges of the greedy triangulation, C can be triangulated by adding diagonals of total length $O(|MWT| + |C|)$. Since the overall greedy triangulation we started with, restricted to each cell of C, forms a greedy triangulation of that cell, and since the greedy triangulation forms a constant approximation to the MWT of a convex polygon [82], the result is that the length of the greedy triangulation is $O(|MWT| + |MC|\sqrt{n}) = (|MWT|\sqrt{n})$.

To improve this $O(\sqrt{n})$ approximation, Levcopoulos and Krznaric modify the greedy triangulation. Their algorithm $MGREEDY$ adds edges one at a time to a PSLG G, as follows. Let uv be the smallest edge not crossing anything already in G (so that the greedy algorithm would next add uv to G). Then $MGREEDY$ tests the following six conditions:

- Some pair of edges uw and wv are already in G, with uwv forming an empty triangle.
- Some edge wx crosses uv, with edge vx already in G and vwx forming an empty triangle.
- Angle vwu is at least $135°$.
- $|wx| < 1.1|uv|$.
- If p is the intersection point of lines vx and uw, then $|xp| < 0.5|wp$.
- There is an edge uy already in G, such that triangule vuy is empty and angle wuy is reflex.

If all six conditions hold, then the algorithm adds edge wx to G; otherwise it adds uv.

A similar construction of a convex partition and a fan in it now holds in this modified algorithm as it did in the greedy triangulation. However now the fan can only have $O(1)$ edges before forming a situation in which the six conditions above hold. The result is that the convex partition is a constant factor approximation to the minimum weight convex partition, and the modified greedy triangulation is a constant factor approximation to the minimum weight triangulation. As Levcopoulos and Krznaric show, these approximations can be constructed in time $O(n \log n)$, or even $O(n)$ if the Delaunay triangulation is already known.

Open Problem 5. What is the best possible approximation ratio for a polynomial time approximation to the minimum weight triangulation?

3.1.2 Exact MWT construction

Instead of approximating the minimum weight triangulation, a number of authors have attacked the problem of constructing the exact minimum weight triangulation, by finding conditions sufficient to guarantee that certain edges belong to the MWT. If enough MWT edges could be found in this way, so that the resulting subgraph of the MWT connected all the vertices, the remaining regions of the plane could be treated as simple polygons and triangulated in polynomial time by dynamic programming [78]. This approach gained in credibility when Edelsbrunner and Tan [50] used it to solve a closely related problem, the *min max weight triangulation*. In this problem, the quality of a triangulation is measured by the length of its longest edge; the min max weight triangulation is the one minimizing this quantity.

Lemma 16 (Edelsbrunner and Tan)**.** There exists some min max weight triangulation that contains the edges of the relative neighborhood graph of the sites.

Corollary 1. The min max weight triangulation can be found in polynomial time.

The dynamic programming idea described above would lead to an $O(n^3)$ time bound, but Edelsbrunner and Tan reduced this to $O(n^2)$.

Recall that the relative neighborhood graph is defined in terms of a *forbidden region* characterization: an edge is in the graph if and only if the lune (formed by intersecting the two circles with that edge as radius) contains no other sites. Several authors have proven similar forbidden region characterizations for the minimum weight triangulation. Yang et al. [114] show that if the region formed as the union of the same two radius circles is empty, the edge belongs to the MWT. (In particular, the two closest sites are connected by an edge in the MWT [65].) Keil [73], Yang [113], and Cheng and Xu [31] prove similar results for an alternate union of two circles, for which the given edge is a chord.

Aichholzer et al. [5] provided a different type of characterization of minimum weight triangulation edges, which leads to an algorithm capable of finding the MWT for certain very large subsets. They define a *light edge* to be one that is not crossed by any other edge of smaller weight. It is not the case that all light edges need be part of the MWT, but they show that if the set of all light edges forms a triangulation, that triangulation must be the one minimizing the total edge weight (or any other monotonic functional of edge weights). Aichholzer et al. note that the characterizations of Keil, Yang et al. [73, 113, 114] all produce edges that must be light.

The best computational results so far for exact minimum weight triangulation construction have been found by Dickerson and Montague [44, 45]. They define a *locally minimal* triangulation to be one in which for every two adjacent triangles forming a convex quadrilateral, the common side of the triangles is the shortest diagonal of the quadrilateral. Clearly, the minimum weight triangulation is locally minimal; their idea is to identify edges belonging to all locally minimal triangulations. The technique is simply to maintain a set S of edges that might possibly be part of a locally minimal triangulation. Initially S consists all pairs of vertices, then in a sequence of passes Dickerson and Montague remove from S any edge that is not the short diagonal of a quadrilateral (not containing other sites) all sides of which still belong to S. Eventually no more edges can be removed, and the process terminates. They then define another set S' of all those edges remaining in S and not crossed by any other edge in S.

Lemma 17 (Dickerson and Montague)**.** All edges in S' belong to all locally minimal triangulations.

Since only $O(n^2)$ edges can be removed from S, this heuristic runs in polynomial time. In computational experiments, Dickerson and Montague have shown that for moderate sized inputs (between

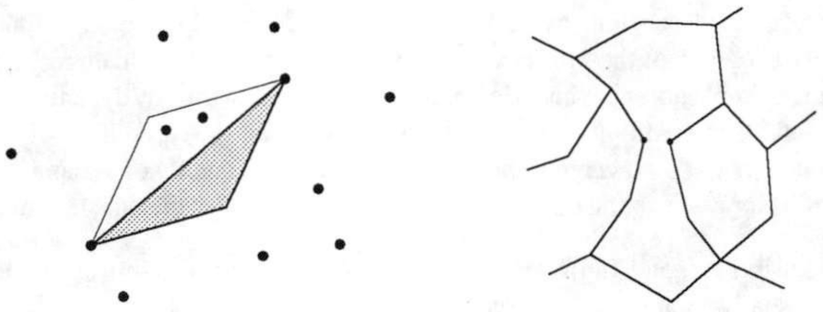

Figure 6. (a) Diamond property: one of two isosceles triangles on edge is empty. (b) Graph violating good polygon property: ratio of diagonal to boundary path is high.

100 and 200 points), the set of edges in S' is very likely to form a connected subgraph of the minimum weight triangulation. The minimum weight triangulation itself can then be found by dynamic programming.

Open Problem 6. Is the graph found by Dickerson and Montague always connected? Can one test in polynomial time whether a given edge belongs to some locally minimal triangulation? Can one find the exact minimum weight triangulation in polynomial time?

3.2 Low-dilation planar graphs

We next consider the problem of constructing planar networks with low *dilation* (maximal ratio between graph and geometric distance). Clearly it will not always be possible to find networks with dilation very close to 1; for instance, any planar graph connecting the vertices of a square has dilation $\sqrt{2}$.

The initial work on this problem has been to show that various previously studied graphs had constant dilation. Chew [32] showed that the rectilinear Delaunay triangulation has dilation at most $\sqrt{10}$. (Note that the dilation here is measured in the Euclidean metric even though the triangulation itself is defined with the rectilinear metric. There is a factor of $\sqrt{2}$ lost in the translation; the rectilinear Delaunay triangulation has rectilinear dilation $\sqrt{5}$.) Chew also pointed out that by placing points around the unit circle, one could find examples for which the Euclidean Delaunay triangulation was made to have dilation as much as $\pi/2$. In the journal version of his paper [33], Chew added a further result, that the graph obtained by Delaunay triangulation for a convex distance function based on an equilateral triangle has dilation at most 2. Chew's conjecture that the Euclidean Delaunay dilation was constant was proved by Dobkin et al. [46], who showed that the Delaunay triangulation has dilation at most $\varphi\pi$ where φ is the golden ratio $(1+\sqrt{5})/2$. Keil and Gutwin [74] further improved this bound to $\frac{2\pi}{3\cos(\pi/6)} \approx 2.42$.

Das and Joseph [39] then showed that these constant dilation bounds were not unusual; in fact such bounds hold for a wide variety of planar graph construction algorithms, satisfying the following two simple conditions:

- **Diamond property**. There is some angle $\alpha < \pi$, such that for any edge e in a graph constructed by the algorithm, one of the two isosceles triangles with e as a base and with apex angle α

contains no other site. This property gets its name because the two triangles together form a diamond shape, depicted in Figure 6(a). For instance, because of its empty-circle property, the Delaunay triangulation satisfies the diamond property with $\alpha = \pi/2$.

- **Good polygon property.** There is some constant d such that for each face f of a graph constructed by the algorithm, and any two sites u, v that are visible to each other across the face, one of the two paths around f from u to v has dilation at most d. Figure 6(b) depicts a graph violating the good polygon property, because two nearby sites have no short boundary path connecting them. The good polygon property is satisfies by any triangulation (with $d = 1$).

Intuitively, if one tries to connect two vertices by a path in a graph that passes near the straight line segment between the two, there are two natural types of obstacle one encounters. The line segment one is following may cross an edge of the graph, or a face of the graph; in either case the path must go around these obstacles. The two properties above imply that neither type of detour can force the dilation of the pair of vertices to be high.

Theorem 10 (Das and Joseph). Any planar graph construction algorithm satisfying the diamond and good polygon properties produces graphs with bounded dilation.

The proof is too complicated to summarize here, but is in some ways similar to the proof of Dobkin et al. [46] that the Delaunay triangulation has bounded dilation. Das and Joseph go on to show that not only Delaunay triangulations but also the greedy and minimum weight triangulations possess these two properties, and hence have bounded dilation. (The bound, while constant, is quite high and could presumably be strengthened.) It seems clear that similar results should also hold for some other triangulation methods developed since their paper, such as the min max edge length triangulation and the min max angle triangulation [51]. Constant dilation of greedy and related triangulations plays a key role in Levcopoulos and Krznaric's recent constant-factor approximation algorithm for the minimum weight triangulation [81], and Drysdale [23] has pointed out that the diamond property of minimum weight triangulation can be very helpful in pruning the edges that could potentially take part in it, and speed up exact solution methods for this triangulation.

Another set of natural candidates for bounded dilation are the *β-skeletons*, formed by including an edge ab when no other site c forms an angle acb larger than some particular bound (depending on the parameter β). When the angle bound is 90° this is the *Gabriel graph*, a subgraph of the Delaunay triangulation and the relative neighborhood graph, and a supergraph of the minimum spanning tree. Skeletons larger angle bounds have been used to find sets of edges guaranteed to be part of the minimum weight triangulation [31, 73, 113]. As β approaches zero, these skeletons contain more and more edges, until eventually one forms the complete geometric graph; this limiting behavior along with the fact that the definition of β-skeletons is closely related to Das and Joseph's diamond property hint together that these graphs might have bounded dilation. But Eppstein [57] has recently shown that a fractal construction leads to β-skeletons in the form of paths with unbounded dilation, for any $\beta > 0$. (Figure 7.)

Curiously, there has been little or no published work on using dilation as the direct basis for constructing planar graphs (rather than constructing graphs some other way and measuring the dilation of the result). Obviously, the minimum dilation planar graph should be a triangulation, since it is

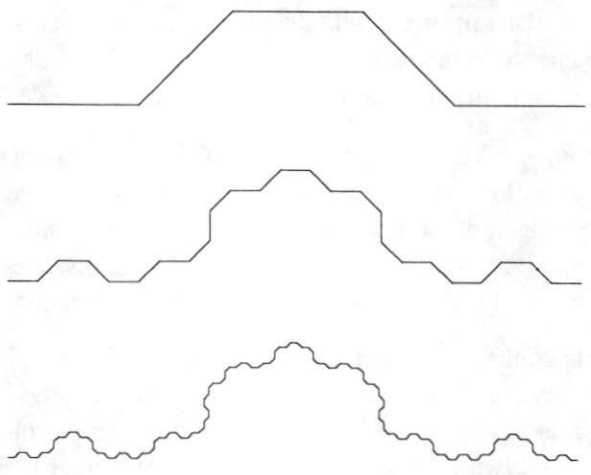

Figure 7. Fractal β-skeleton with unbounded dilation.

never harmful to add diagonals. It seems that it should have useful properties similar to those of the Delaunay triangulation, min max angle triangulation, min max edge length triangulation, minimum weight triangulation, and other optimal triangulations, but this has apparently not been studied.

For many optimal triangulation problems, the version of the problem in which one optimally completes the triangulation of a convex or simple polygon can be solved by dynamic programming [78], however even this is not obvious for the minimum dilation triangulation. A solution to this subproblem might have implications in allowing the powerful *edge insertion* method [16, 51] to be applied to the point set version of the problem.

Open Problem 7. Is it possible to construct in polynomial time the minimum dilation triangulation of a point set, or of a simple polygon?

Clearly there still also remains a wide gap between the best upper and lower bounds (2 and $\sqrt{2}$ respectively) on dilation of planar graphs.

Open Problem 8. What is the worst case dilation of the minimum dilation triangulation?

3.3 Planar Dilation and Weight

Levcopoulos and Lingas [83] and Das and Joseph [39] brought the total weight of the graph, as well as its dilation, into the equation. Clearly the weight should be measured in terms of the minimum spanning tree, for as Das and Joseph [39] observe, any graph with bounded dilation should at least be connected.

Das and Joseph [39] form a planar graph by applying a greedy triangulation algorithm to the polygons remaining between the minimum spanning tree and the convex hull of the sites. They then consider the greedy edges in decreasing order of weight, removing each edge if it is unnecessary to achieve the desired dilation. They then apply their diamond property and good polygon property criteria to bound the dilation of the resulting pruned graph. Because they only prune the greedy edges, the total weight of the spanner they construct is close to three times that of the minimum spanning tree.

Althöfer et al. [7, 8] later generalized this pruning strategy to arbitrary graphs. Given any graph, and any parameter $t > 0$, their method again considers the edges of the graph in decreasing order by weight, removing each edge if it is not necessary to achieve dilation $1 + t$. Obviously the result is a subgraph with dilation $1 + t$, but they prove further that the total weight of this subgraph is $1 + O(n/t)$ times that of the graph's minimum spanning tree, and that the number of edges is $O(n^{1+O(1/t)})$. For planar graphs these bounds are much better: The weight is $1 + O(1/t)$ times that of the minimum spanning tree, and the number of edge is $n(1 + O(1/t))$.

Althöfer et al. then apply this method to the Delaunay triangulation, which as we have seen is a planar graph with constant dilation that also contains the Euclidean minimum spanning tree. The result is a planar spanner with constant dilation, few edges, and weight arbitrarily close to that of the minimum spanning tree.

Levcopoulos and Lingas [83] proved the same result by a similar but more complicated method of pruning the Delaunay triangulation; their pruning method works only for planar graphs, but has the further advantage that it runs in linear time. The same pruning method can be used to trade weight and dilation in the other direction, and find graphs with dilation arbitrarily close to that of the Delaunay triangulation, and weight a large constant times that of the minimum spanning tree.

4 General Graphs

Most of the work on general geometric network design problems, in which the network to be constructed is not of some restricted class, has been on dilation of sparse graphs, since it is trivial to find a graph with low weight or diameter, or to find a non-sparse graph with low dilation.

4.1 Dilation only

As we saw, for planar graphs, the dilation can not approach one. By considering nonplanar graphs, it is possible to find sparse graphs approximating the complete Euclidean graph arbitrarily closely. Specifically, Keil [72] showed that variations of the Yao graph construction (in which one partitions the space around each point into wedges with a given fixed opening angle, and connects the point to the nearest neighbor in each wedge) produce graphs with dilation arbitrarily close to 1, with $O(n)$ edges, and that can be constructed in time $O(n \log n)$.

Theorem 11 (Keil). The Yao graph, formed by wedges of opening angle $\theta < 60°$, produces graphs with dilation $1 + O(1/\theta)$.

Proof: To find a path in this graph from u to v, one at each step determines the wedge containing v and moves along a graph edge to the nearest vertex (w) in that wedge. The worst case for the algorithm occurs when uv and uw are similar in length but widely separated in angle, in which case the distance to v is reduced by $(1 - O(1/\theta))uw$. For $\theta < 60°$ this reduction must be positive, and repeating the process brings us eventually to v. The total distance traveled is proportional to $1/(1 - O(1/\theta)) = 1 + O(1/\theta)$ times the total reduction in distance to v, which is exactly the original distance from u to v. □

Althöfer et al. [7, 8] observed that this result holds in any dimension, and Ruppert and Seidel [99] modify the technique so that orthogonal range searching methods can be used in its construction, improving its running time in high dimensions to $O(n \log^{d-1} n)$.

Callahan [25] improved the dependence between the number of edges and the dilation, as well as the construction time, in this result:

Theorem 12. If one forms a well-separated pair decomposition, and chooses a representative edge from each pair, the resulting graph has $O(n)$ edges and can be made to have dilation arbitrarily close to one (as a function of the separation parameter of the decomposition).

Proof: We find a short path between two sites u and v by the following recursive process: by the definition of a well-separated pair decomposition, the decomposition includes some pair (U, V) of sets of sites for which u is in U and v is in V. Find the edge $u'v'$ representing the pair (U, V), and form a path by recursively connecting u to u', following edge $u'v'$, and connecting v' to v. If the parameter of separation in the decomposition is s, and the length of uv is r, the length of $u'v'$ can be at most $r(s+4)/s$, and the length of uu' and vv' can be at most $2r/s$. Therefore this recursive algorithm produces a path with length satisfying the recurrence

$$L(r) \le r(1 + 4/s) + 2L(2r/s).$$

If $s > 4$, this solves to $O(r)$, and for any $\epsilon > 0$ one can choose a sufficiently large s for which the solution to this recurrence is $r(1+\epsilon)$. □

As in his approximation to the Euclidean minimum spanning tree, Callahan further shows that the dependence on ϵ can be reduced by choosing more carefully the representative edge for each pair in the well separated pair decomposition.

Similar constructions of sparse spanners with a larger dependence on the dilation were also given by Salowe [100] and Vaidya [112].

4.2 Dilation and weight

Note that the small-dilation graphs described above may have very high weight. As we have already seen, it is possible for planar sites to construct small-dilation graphs with weight arbitrarily close to that of the minimum spanning tree. the greedy algorithm

The greedy algorithm of Althöfer et al. [7, 8] (described in the section on planar graphs) was defined for any graph, and so produces spanners with nontrivial weight bounds in any dimension; but in general the results proved by those authors are far from the constant factor over minimum spanning tree weight that one would hope for. Das et al. [38] showed that applying this greedy algorithm to the complete Euclidean graph in three dimensions again produces a spanner with any given dilation $t > 1$, and total weight $O(1)$ times that of the minimum spanning tree (where the constant depends on t). Chandra [27] showed that for random point sets in higher dimensions, the weight is again $O(1)$ times that of the minimum spanning tree. Das and Narasimhan [40] apply this greedy approach to a sparse spanner produced by clustering techniques; the dilation of the result is the product of the dilations from these two steps, which is still typically some constant. This idea speeds up the greedy method to run in time $O(n \log^2 n)$, and produces results similar to those of Althöfer et al.

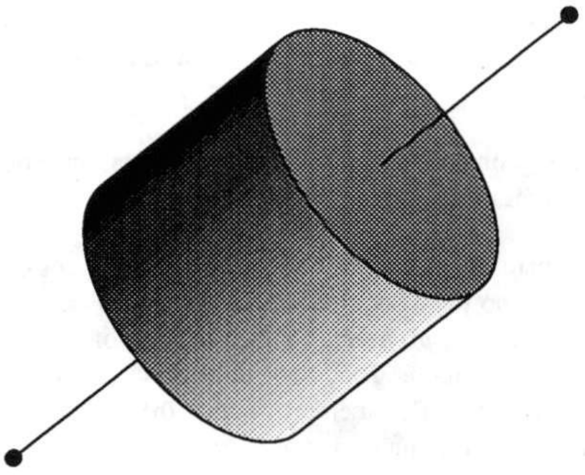

Figure 8. Isolation property: a cylinder around each edge is not crossed by any other edge.

Chandra et al. [28] showed that, for any graph, one can construct a spanner with dilation $O(\log^2 n)$ and weight $O(1)$ times that of the minimum spanning tree. For geometric graphs in any dimension, they construct spanners with constant dilation and weight $O(\log n)$ times that of the minimum spanning tree. They actually give two methods for this second result. One method is simply to apply the greedy algorithm; they show an $O(\log n)$ weight factor based on the *gap property*: the endpoints of any two edges are separated by a distance at least proportional to the smaller of the two edge lengths. The other method combines known sparse (but heavy) spanners with approximations to the traveling salesman path; the algorithm then partitions the path recursively into smaller pieces and combines spanners from representative points on each piece. Arya and Smid [10] modify and speed up the first, greedy, method: instead of adding edges incrementally when necessary to preserve the dilation, and proving that the result satisfies the gap property, they add edges unless the gap property would be violated, and show that the resulting graph has bounded dilation. With this modified greedy method they show that a graph with bounded dilation and weight $O(\log n)$ times that of the minimum spanning tree can be constructed in time $O(n \log^d n)$.

Finally, Das, Narasimhan, and Salowe [41] extended the results of Das et al. [38], and showed that in any dimension the greedy algorithm produces spanners with both constant dilation and weight a constant times that of the minimum spanning tree. The dilation bound is immediate; the weight bound is reminiscent of the methods of Das and Joseph [39] in that the authors describe a collection of general conditions under which such a bound applies; further their isolation property is very similar to the diamond property of Das and Joseph.

- **Isolation property.** There exists a constant c, such that any edge of length ℓ produced by the given algorithm can be placed within a cylinder of radius and height $c \cdot \ell$ (Figure 8), such that the axis of the cylinder is a subset of the edge, and the cylinder does not intersect any other edge of the graph.

Note that, unlike the diamond property, the cylinders are required to avoid the edges of the graph, and not just the other sites. Also note that in this definition the cylinders around each edge may intersect each other, but one can shrink them by a factor of two to avoid intersections. The exact shape of the cylinders is also unimportant.

Lemma 18 (Das et al.)**.** Any geometric graph satisfying the isolation property has weight at most a constant times that of the minimum spanning tree.

The proof is based on separating the edges of the graphs into groups of nearly parallel edges, and charging the length in each group to edges in the Steiner minimal tree of the sites. Das et al. also prove similar results for the *leapfrog property*, a generalization of the isolation property for which they give a more complicated definition. They then show that graphs produced by a version of the greedy algorithm have this property. This then proves that the greedy algorithm produces spanners with weight $O(1)$ times that of the minimum spanning tree.

4.3 Dilation, weight, and degree

Perhaps the ultimate results in high-dimensional spanners combine dilation, weight, and vertex degree. Clearly the degree bound must be at least three, since degree-two graphs may be forced to have very large dilation. Further, for any fixed bound d on the degree, one must have dilation bounded away from one by some function of d, even in the plane, as can be seen by considering the vertices of a regular $d + 2$-gon.

Chandra et al. [28] showed that their version of the greedy method, when applied to polyhedral approximations to the Euclidean graph, produces a spanner with degree bounded by some function of the dilation, dimension, and the particular approximation. Their bound is not stated explicitly but is roughly exponential in the dimension. As we have seen these spanners also have low weight.

Salowe [101] produces another bounded-degree spanner algorithm by modifying the Yao graph construction described earlier (which may have unbounded degree). The idea is simply to add edges to the spanner in order by weight, adding an edge (u, v) if there is no other edge (u, w) or (v, w) already added and having an angle close to that of (u, v). Salowe also reports that a similar degree bound applies to the spanner of Ruppert and Seidel [99] but this appears to be erroneous. However it should be possible to keep track of the next edge to be added to the graph at each step of this construction (that is, the shortest edge within a certain range of angles, connecting two points not already incident to edges in that range) by combining Ruppert and Seidel's modified Yao graph orthogonal range searching technique [99] with a method of Eppstein [53] for maintaining bichromatic closest pairs dynamically. The result would be an $O(n \log^{O(1)} n)$ time algorithm to construct this bounded degree spanner.

As we have already seen, Arya and Smid [10] achieved a similar time bound for an alternate bounded degree spanner. Yet another bounded-degree spanner construction is credited by Vaidya [112] to Feder and Nisan.

Finally, Arya et al. [9] improved the time bounds of all these methods. The results of Arya et al. include an $O(n \log n)$ time algorithm for constructing a spanner with bounded dilation, bounded degree, and weight a constant factor times the minimum spanning tree. They also give an $O(n \log n)$ algorithm for constructing a spanner with bounded dilation on paths of $O(\log n)$ edges, bounded degree, and weight $O(\log^2 n)$ times that of the minimum spanning tree. (The idea of finding short paths with few edges was also previously considered by Arya, Mount, and Smid [11].) Therefore for any fixed dilation, one can quickly find spanners with bounded degree. Further work on the problem has turned this tradeoff around, and asked how small a degree is possible to achieve bounded dilation.

Salowe [101] was the first to find degree bounds that did not grow with the dimension of the problem; he showed that in any dimension, there is a degree-four graph with constant dilation. Salowe starts with any one of the bounded degree methods discussed above. Salowe's method for reducing the degree to four is then to cluster points using the nearest neighbor forest, and use these clusters to roughly halve the degree of any spanner construction method. Starting with the initial bounded degree spanner method described above and iterating this degree reduction process produces his result. This paper also contains a convenient classification of the spanner literature in terms of five parameters (dilation, time, number of edges, weight, and degree) and the class of metric space for which the spanners are defined. Salowe's paper does not however include a bound on the total weight of his spanners.

Das and Heffernan [37] improved this degree-four bound, by showing that in any dimension, there is a graph with maximum degree three, at most dn edges (for any $d>1$), constant dilation (depending on d), and total weight $O(\log n)$ times the minimum spanning tree weight. They state that in dimensions two and three the weight can be further reduced to $O(1)$ times the minimum spanning tree weight. Their technique involves similar nearest-neighbor-forest based clustering methods to those of Salowe, applied somewhat more carefully, and combined with the greedy spanners constructed by Chandra et al. [28]. The weight bound comes from summing the weight of the greedy spanner with that of the nearest neighbor forest (which as a subgraph of the minimum spanning tree has small weight). Combining this with the recent result of Das et al. [41] on constant weight bounds for the greedy spanner construction yields the following result.

Theorem 13 (Das, Heffernan, Narasimhan, and Salowe). For sites in any fixed dimension, and for any constant $d > 1$, there is a graph with constant dilation, degree three, at most dn edges, and weight a constant factor times that of the minimum spanning tree.

References

[1] Active Geometry Group, Johns Hopkins U. Extracting the geometry of the vascular tree. Manuscript, available online at http://blaze.cs.jhu.edu/grad/lundberg/agg/projects/vasc.html.

[2] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Proc. 6th ACM Symp. Comp. Geom.*, 1990, pp. 203–210.

[3] P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comp. Geom. Theory & Appl.*, vol. 1, 1992, pp. 189–201.

[4] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear time algorithm for computing the Voronoi diagram of a convex polygon. *Disc. Comp. Geom.*, 1989, pp. 591–604.

[5] O. Aichholzer, F. Aurenhammer, M. Taschwer, and G. Rote. Triangulations intersect nicely. *Proc. 11th ACM Symp. Comp. Geom.*, 1995, pp. 220–229.

[6] N. Alon, S. Rajagopalan, and S. Suri. Long non-crossing configurations in the plane. *Proc. 9th ACM Symp. Comp. Geom.*, 1993, pp. 257–262.

[7] I. Althöfer, G. Das, D. Dobkin, and D. Joseph. Generating sparse spanners for weighted graphs. *Proc. 2nd Scand. Worksh. Algorithm Theory*. Springer LNCS 447, 1990, pp. 26–37.

[8] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Disc. Comp. Geom.*, vol. 9, 1993, pp. 81–100.

[9] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean Spanners: Short, Thin, and Lanky. *Proc. 27th ACM Symp. Theory of Computing*, 1995, pp. 489–498. Available online at http://www.cs.umd.edu/~mount/Papers/stoc95.ps.

[10] S. Arya and M. Smid. Efficient construction of a bounded degree spanner with low weight. *Proc. 2nd Eur. Symp. Algorithms*. Springer LNCS 855, 1994, pp. 48–59.

[11] S. Arya, D. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. *Proc. 35th IEEE Symp. Foundations of Comp. Sci.*, 1994, pp. 703–712. Available online at http://www.cs.umd.edu/~mount/Papers/focs94.ps.

[12] T. Asano, B. Bhattacharya, M. Keil, and F. Yao. Clustering algorithms based on minimum and maximum spanning trees. *Proc. 4th ACM Symp. Comp. Geom.*, 1988, pp. 252–257.

[13] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum-weight *k*-trees and prize-collecting salesmen. *Proc. 27th ACM Symp. Theory of Computing*, 1995, 277–283. Available online at http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/Web/People/avrim/Papers/prizetsp.ps.Z.

[14] A. I. Barvinok. Two algorithmic results for the traveling salesman problem. Manuscript, 1994.

[15] J. L. Bentley and J. Saxe. Decomposable searching problems I: static-to-dynamic transformation. *J. Algorithms*, vol. 1, 1980, pp. 301–358.

[16] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell, and T. S. Tan. Edge-insertion for optimal triangulations. *Disc. Comp. Geom.*, vol. 10, 1993, pp. 47–65.

[17] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry*, D.-Z. Du and F. K. Hwang, eds., World Scientific, 1992; 2nd edition, 1995, pp. 47–123.

[18] M. Bern and D. Eppstein. Worst-case bounds for subadditive geometric graphs. *Proc. 9th ACM Symp. Comp. Geom.*, 1993, pp. 183–188.

[19] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In *Approximation algorithms for NP-hard problems*, D. Hochbaum, ed., to appear.

[20] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *Proc. 3rd Worksh. Algorithms and Data Structures*. Springer LNCS 709, 1993, pp. 188–199.

[21] A. Blum, P. Chalasani, and S. Vempala. A constant-factor approximation for the k-MST problem in the plane. *Proc. 27th ACM Symp. Theory of Computing*, 1995, 294–302. Available online at http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/Web/People/avrim/Papers/planarktrees.ps.Z.

[22] A. Blum, R. Ravi, and S. Vempala. A constant-factor approximation algorithm for the k-MST problem. *Proc. 28th ACM Symp. Theory of Computing*, 1996.

[23] H. Brönnimann. Computational Geometry Tribune, no. 3, available online at http://www.inria.fr/prisme/personnel/bronnimann/cgt/cgt3.ps.Z.

[24] L. Cai and D. Corneil. Tree spanners. *SIAM J. Discrete Math.*, vol. 8, 1995, pp. 359–388.

[25] P. B. Callahan. The Well-Separated Pair Decomposition and its Applications. Ph.D. thesis, Johns Hopkins U., 1995. Available online at ftp://ftp.cs.jhu.edu/pub/callahan/dissertation.ps.Z.

[26] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest neighbors and n-body potential fields. *J. ACM*, vol. 42, 1995, pp. 67–90.

[27] B. Chandra. Constructing sparse spanners for most graphs in higher dimensions. *Inf. Proc. Lett.*, vol. 51, 1994, pp. 289–294.

[28] B. Chandra, G. Das, G. Narasimhan, and J. Soares. New sparseness results on graph spanners. *Int. J. Comp. Geom. & Appl.*, vol. 5, 1995, pp. 125–144.

[29] D. Cheriton and R. E. Tarjan. Finding minimum spanning trees. *SIAM J. Comput.*. vol. 5, 1976, pp. 310–313.

[30] S. Y. Cheung and A. Kumar. Efficient quorumcast routing algorithms. *Proc. IEEE Conf. Computer Communications*, 1994, vol. 2, pp. 840–847.

[31] S. Cheng and Y. Xu. Approaching the largest β-skeleton within the minimum weight triangulation, manuscript cited by [44], 1995.

[32] L. P. Chew. There is a planar graph almost as good as the complete graph. *Proc. 2nd ACM Symp. Comp. Geom.*, 1986, pp. 169–177.

[33] L. P. Chew. There are planar graphs almost as good as the complete graph. *J. Comp. Sys. Sci.*, vol. 39, 1989, pp. 205–219.

[34] L. P. Chew and R. L. Drysdale. Voronoi diagrams based on convex distance functions. *Proc. 1st ACM Symp. Comp. Geom.*, 1985, pp. 235–244.

[35] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Report 388, Grad. Sch. Industrial Admin., Carnegie Mellon U., 1975.

[36] C. Chiang, M. Sarrafzadeh, and C. K. Wong. A powerful global router based on Steiner min-max trees. *Proc. IEEE Int. Conf. CAD*, 1989, pp. 2–5.

[37] G. Das and P. J. Heffernan. Constructing degree-3 spanners with other sparseness properties. *Proc. 4th Int. Symp. Algorithms and Computation*. Springer LNCS 762, 1993, pp. 11–20.

[38] G. Das, P. J. Heffernan, and G. Narasimhan. Optimally sparse spanners in 3-dimensional Euclidean space. *Proc. 9th ACM Symp. Comp. Geom.*, 1993, pp. 53–62.

[39] G. Das and D. Joseph. Which triangulations approximate the complete graph? *Proc. Int. Symp. Optimal Algorithms*. Springer LNCS 401, 1989, pp. 168–192.

[40] G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. *Proc. 10th ACM Symp. Comp. Geom.*, 1994, pp. 132–139.

[41] G. Das, G. Narasimhan, and J. Salowe. A new way to weigh malnourisheed Euclidean graphs. *Proc. 6th ACM-SIAM Symp. Discrete Algorithms*, 1995, pp. 215–222.

[42] A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. *Proc. 3rd Worksh. Algorithms and Data Structures*. Springer LNCS 709, 1993, pp. 265–276.

[43] O. Devillers, S. Meiser, and M. Teillaud. Fully dynamic Delaunay triangulation in logarithmic expected time per operation. *Comp. Geom. Theory & Appl.*, vol. 2, 1992, pp. 55–80.

[44] M. Dickerson and M. Montague. A (usually) connected subgraph of the minimum weight triangulation. *Proc. 5th MSI-Stony Brook Worksh. Comp. Geom.*, 1995. Available on-line at ftp://ams.sunysb.edu/pub/geometry/msi-workshop/95/dickerso.ps.gz. See also http://www.middlebury.edu/~dickerso/mwtskel.html.

[45] M. Dickerson and M. Montague. The exact minimum weight triangulation. *Proc. 12th ACM Symp. Comp. Geom.*, 1996.

[46] D. P. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs are almost as good as complete graphs. *Disc. Comp. Geom.*, vol. 5, 1990, pp. 399–407.

[47] R. L. Drysdale. A practical algorithm for computing the Delaunay triangulation for convex distance functions. *Proc. 1st ACM-SIAM Symp. Discrete Algorithms*, 1990, pp. 159–168.

[48] D.-Z. Du and F. K. Hwang. The state of the art in Steiner ratio problems. In *Computing in Euclidean Geometry*, D.-Z. Du and F. K. Hwang, eds., World Scientific, 1992; 2nd edition, 1995, pp. 195–224.

[49] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, vol. 15, 1986, pp. 317–340.

[50] H. Edelsbrunner and T. S. Tan. A quadratic time algorithm for the minmax length triangulation. *Proc. 32nd IEEE Symp. Foundations of Comp. Sci.*, 1991, pp. 414–423.

[51] H. Edelsbrunner, T. S. Tan, and R. Waupotitsch. A polynomial time algorithm for the minmax angle triangulation. *SIAM J. Sci. Stat. Comp.*, vol. 13, 1992, pp. 994–1008.

[52] D. Eppstein. Offline algorithms for dynamic minimum spanning tree problems. *J. Algorithms*, vol. 17, 1994, pp. 237–250.

[53] D. Eppstein. Dynamic Euclidean Minimum Spanning Trees and Extrema of Binary Functions. *Disc. Comp. Geom.*, vol. 13, 1995, pp. 237–250.

[54] D. Eppstein. Faster geometric k-point MST approximation. Tech. Rep. 95-13, Dept. Inf. & Comp. Sci., UC Irvine, 1995. Available online at http://www.ics.uci.edu/Document/UCI:ICS-TR-95-13.

[55] D. Eppstein. Average case analysis of dynamic geometric optimization. *Comp. Geom. Theory & Appl.*, to appear. Available online at http://www.ics.uci.edu/Document/UCI:ICS-TR-93-18.

[56] D. Eppstein. Faster circle packing with application to nonobtuse triangulation. *Int. J. Comp. Geom. & Appl.*, to appear. Available online at http://www.ics.uci.edu/Document/UCI:ICS-TR-94-33.

[57] D. Eppstein. Beta-skeletons have unbounded dilation. Tech. Rep. 96-15, Dept. Inf. & Comp. Sci., UC Irvine, 1996.

[58] D. Eppstein and J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Disc. Comp. Geom.*, vol. 11, 1994, pp. 321–350.

[59] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification – A technique for speeding up dynamic graph algorithms. *Proc. 33rd IEEE Symp. Foundations of Comp. Sci.*, 1992, pp. 60–69.

[60] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms*, vol. 13, 1992, pp. 33–54.

[61] M. Fredman and D. E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Proc. 31st IEEE Symp. Foundations of Comp. Sci.*, 1990, pp. 719–725.

[62] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, vol. 6, 1986, pp. 109–122.

[63] M. R. Garey and D. S. Johnson Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.

[64] N. Garg and D. S. Hochbaum. An $O(\log k)$ approximation for the k minimum spanning tree problem in the plane. *Proc. 26th ACM Symp. Theory of Computing*, 1994, pp. 432–438.

[65] P. D. Gilbert. New results in planar triangulations. Report R-850, U. Illinois Coordinated Science Lab., 1979.

[66] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, vol. 7, 1992, pp. 381–413.

[67] S. Haruyama and D. Fussell. A new area-efficient power routing algorithm for VLSI layout. *Proc. IEEE Int. Conf. CAD*, 1987, pp. 38–41.

[68] J.-M. Ho, D. T. Lee, C.-H. Chang, and C. K. Wong. Minimum diameter spanning trees and related problems. *SIAM J. Comput.*, vol. 20, 1991, pp. 987–997.

[69] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh. Clock routing for high-performance IC's. *Proc. 27th ACM/IEEE Design Automation Conf.*, 1990, pp. 573–579.

[70] A. Kahng, J. Cong, and G. Robins. High-performance clock routing based on recursive geometric matching. *Proc. 28th ACM/IEEE Design Automation Conf.*, 1991, pp. 322–327.

[71] D. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM*, vol. 42, 1995, pp. 321–328.

[72] J. M. Keil. Approximating the complete Euclidean graph. *Proc. 1st Scand. Worksh. Algorithm Theory*. Springer LNCS 318, 1988, pp. 208–213.

[73] J. M. Keil. Computing a subgraph of the minimum weight triangulation. *Comp. Geom. Theory & Appl.*, vol. 4, 1994, pp. 13–26.

[74] J. M. Keil and C. A. Gutwin. The Delaunay triangulation closely approximates the complete Euclidean graph. *Proc. 1st Worksh. Algorithms and Data Structures*. Springer LNCS 382, 1989, pp. 47–56.

[75] S. Khuller, B. Raghavachari, and N. Young. Low degree spanning trees of small weight. *Proc. 26th ACM Symp. Theory of Computing*, 1994, pp. 412–421.

[76] S. Khuller, B. Raghavachari, and N. Young. Balancing minimum spanning trees and shortest-path trees. *Algorithmica*, vol.14, 1995, pp. 305–321.

[77] D. Kirkpatrick. Optimal search in planar subdivision. *SIAM J. Comput.*, vol. 12, 1983, pp. 28–35.

[78] G. T. Klincsek. Minimal triangulations of polygonal domains. *Ann. Disc. Math.*, vol. 9, 1980, pp. 121–123.

[79] E. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization. Wiley, 1985.

[80] C. Levcopoulos. An $\Omega(\sqrt{n})$ lower bound for non-optimality of the greedy triangulation. *Inf. Proc. Lett.*, vol. 25, 1987, pp. 247–251.

[81] C. Levcopoulos and D. Krznaric. Quasi-Greedy triangulations approximating the minimum weight triangulation. *Proc. 7th ACM-SIAM Symp. Discrete Algorithms*, 1996.

[82] C. Levcopoulos and A. Lingas. On approximation behavior of the greedy triangulation for convex polygons. *Algorithmica*, vol. 2, 1987, pp. 175–193.

[83] C. Levcopoulos and A. Lingas. There are planar graphs almost as good as the complete graphs and as short as minimum spanning trees. *Proc. Int. Symp. Optimal Algorithms*. Springer LNCS 401, 1989, pp. 9–13.

[84] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *Proc. 18th IEEE Symp. Foundations of Comp. Sci.*, 1977, pp. 162–170.

[85] E. L. Lloyd. On triangulations of a set of points in the plane. *Proc. 18th IEEE Symp. Foundations of Comp. Sci.*, 1977, pp. 228–240.

[86] G. K. Manacher and A. L. Zobrist. Neither the greedy nor the Delaunay triangulation approximates the optimum. *Inf. Proc. Lett.*, vol. 9, 1979, pp. 31–34.

[87] C. S. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems. *Proc. 11th ACM Symp. Comp. Geom.*, 1995, pp. 360–369.

[88] K. Mehlhorn, S. Meiser, and C. O'Dunlaing. On the construction of abstract Voronoi diagrams. *Disc. Comp. Geom.*, vol. 6, 1991, pp. 211–224.

[89] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem. *Proc. 7th ACM-SIAM Symp. Discrete Algorithms*, 1996.

[90] C. Monma, M. Paterson, S. Suri, and F. Yao. Computing Euclidean maximum spanning trees. *Algorithmica*, vol. 5, 1990, pp. 407–419.

[91] C. Monma and S. Suri. Transitions in geometric spanning trees. *Proc. 7th ACM Symp. Comp. Geom.*, 1991, pp. 239–249.

[92] K. Mulmuley. Randomized multidimensional search trees: dynamic sampling. *Proc. 7th ACM Symp. Comp. Geom.*, 1991, pp. 121–131.

[93] C. O'Dunlaing and C. K. Yap. A "retraction" method for planning the motion of a disc. *J. Algorithms*, vol. 6, 1985, pp. 104–111.

[94] S. Olariu, S. Toida, and M. Zubair. On a conjecture by Plaisted and Hong. *J. Algorithms*, vol. 9, 1988, pp. 597–598.

[95] C. H. Papadimitriou and U. V. Vazirani. On two geometric problems related to the traveling salesman problem. *J. Algorithms*, vol. 5, 1984, pp. 231–246.

[96] D. A. Plaisted and J. Hong. A heuristic triangulation algorithm. *J. Algorithms*, vol. 8, 1987, pp. 405–437.

[97] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short and small. *Proc. 5th ACM-SIAM Symp. Discrete Algorithms*, 1994, pp. 546–555.

[98] G. Robins and J. S. Salowe. Low-degree minimum spanning trees. *Disc. Comp. Geom.*, vol. 14, 1995, pp. 151–165.

[99] J. Ruppert and R. Seidel. Approximating the d-dimensional complete Euclidean graph. *3rd Canad. Conf. Comp. Geom.*, 1991, pp. 207–210.

[100] J. S. Salowe. Constructing multidimensional spanner graphs. *Int. J. Comp. Geom. & Appl.*, vol. 1, 1991, pp. 99–107.

[101] J. S. Salowe. Euclidean spanner graphs with degree four. *Disc. Applied Math.*, vol. 54, 1994, pp. 55–66.

[102] J. S. Salowe, D. S. Richards, and D. E. Wrege. Mixed spanning trees: a technique for performance-driven routing. *Proc. 3rd Great Lakes Symp. VLSI Design Automation of High Performance VLSI Systems*, IEEE, 1993, pp. 62–66.

[103] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Comm. ACM*, vol. 29, 1986, pp. 669–679.

[104] O. Schwarzkopf. Dynamic maintenance of geometric structures made easy. *Proc. 32nd IEEE Symp. Foundations of Comp. Sci.*, 1991, pp. 197–206.

[105] Sedgewick and J. Vitter. *Algorithmica*, vol. 1, 1986, pp. 31–48. (((Fill in this one and mention it in text; they consider dilation of certain random Euclidean graphs, as part of a proof that their shortest paths can be found quickly)))

[106] M. I. Shamos and D. Hoey. Closest-point problems. *Proc. 16th IEEE Symp. Foundations of Comp. Sci.*, 1975, pp. 151–162.

[107] N. Sherwani. Algorithms for VLSI Physical Design Automation. Kluwer, 1993.

[108] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comp. Sys. Sci.*, vol. 24, 1983, pp. 362–381.

[109] J. M. Steel and T. L. Snyder. Worst-case growth rates of some classical problems of combinatorial optimization. *SIAM J. Comput.*, vol. 18, 1989, pp. 278–287.

[110] R. Tsay. Exact zero skew. *Proc. IEEE Int. Conf. CAD*, 1991, pp. 336–339.

[111] P. M. Vaidya. Minimum spanning trees in k-dimensional space. *SIAM J. Comput.*, vol. 17, 1988, pp. 572–582.

[112] P. M. Vaidya. A sparse graph almost as good as the complete graph on points in K dimensions. *Disc. Comp. Geom.*, vol. 6, 1991, pp. 369–381.

[113] B.-T. Yang. A better subgraph of the minimum weight triangulation. *Inf. Proc. Lett.*, vol. 56, 1995, pp. 255–258.

[114] B.-T. Yang, Y.-F. Xu, and Z.-Y. You. A chain decomposition algorithm for the proof of a property on minimum weight triangulations. *Proc. Int. Symp. Algorithms and Computation*, Springer LNCS 834, 1994, pp. 423–427.

[115] A. C. Yao. On constructing minimum spanning trees in k-dimensional space and related problems. *SIAM J. Comput.*, vol. 11, 1982, pp. 721–736.

[116] A. A. Zelikovsky and D. D. Lozevanu. Minimal and bounded trees. *Proc. Tezele Cong. XVIII Acad. Romano-Americane*, Kishinev, 1993, pp. 25–26.