C H A P T E R

8

APPROXIMATION ALGORITHMS FOR GEOMETRIC PROBLEMS

Marshall Bern **David Eppstein**

This chapter discusses approximation algorithms for hard geometric problems. We cover three well-known shortest network problems—traveling salesman, Steiner tree, and minimum weight triangulation—along with an assortment of problems in areas such as clustering and surface approximation.

8.1 INTRODUCTION

This chapter surveys approximation algorithms for hard geometric problems. The problems we consider typically take inputs that are point sets or polytopes in two- or three-dimensional space, and seek optimal constructions, (which may be trees, paths, or polytopes). We limit attention to problems for which no polynomial-time exact algorithms are known, and concentrate on bounds for worst-case approximation ratios, especially bounds that depend intrinsically on geometry.

We illustrate our intentions with two well-known problems. Given a finite set of points S in the plane, the Euclidean traveling salesman problem asks for the shortest tour of S. Christofides' algorithm achieves *approximation ratio* $\frac{3}{2}$ for this problem, meaning that it always computes a tour of length at most three-halves the length of the optimal tour. This bound depends only on the triangle inequality, so Christofides' algorithm works equally well in any metric space, even the finite metric space induced by the shortest-path distance on a network (edge-weighted graph).

The Steiner tree problem asks for the shortest tree that includes S in its vertex set. Approximation ratios for this problem depend quite intimately on geometry. For example, the current best bound for the rectilinear version of the problem is $\frac{19}{15} \approx 1.267$; for the Euclidean version, it is arbitrarily close to $1+\ln\frac{2}{\sqrt{3}} \approx 1.144$; and for the network version, arbitrarily close to $1+\ln 2 \approx 1.693$. The proof of a certain geometric conjecture would change the algorithm of choice for the Euclidean case and improve the ratio to 1.114. Because of the more prominent role of geometry, we shall discuss the Steiner tree problem at greater length than the traveling salesman problem.

8.1.1 OVERVIEW OF TOPICS

We have selected five major problems or problem areas as section topics in this chapter: the traveling salesman problem, the Steiner tree problem, minimum-weight triangulation, clustering, and separation. Although we shall not discuss applications, all five of the selected problems arise naturally in a variety of contexts. Four of the five problems—all except minimum-weight triangulation—are known to be *NP*-hard. Our treatments include fairly detailed proof sketches, because proofs are the *sine qua non* of approximation algorithms: that which distinguishes them from mere heuristics.

We have relegated some other work to a sixth, miscellaneous, section. The problems in this section include rectangle packing and covering, embedding graphs onto points, and finding long graphs without edge crossings. We omit all work on approximations of polynomially solvable problems and all results that assume random inputs.

8.1.2 SPECIAL NATURE OF GEOMETRIC PROBLEMS

A geometric input is typically more constrained than its non-geometric counterpart. First, geometric problems can often be reduced to combinatorial optimization problems involving graphs or set systems, only the resulting problem instances are usually quite special. Geometric packing and covering problems can be reduced to independent set or clique cover; the polyhedron separation problem, discussed in Section 8.6.2, can be reduced to a special case of the set cover problem. Secondly, geometric problems sometimes admit decompositions related to planar separator theorems. For example, the Euclidean traveling salesman problem (TSP) can be exactly solved in time $n^{O(\sqrt{n})}$ [Smi88]. Finally, and most significantly, a set of n points in the plane has only $2n$ degrees of freedom, while a network on n vertices may have up to $\binom{n}{2}$ independent distances.

The constrained nature of geometric problems often helps in approximation. In Section 8.7.2 we shall approximate a packing problem better than the bound implied by an approximation to independent set. The geometric origin of the set cover problem in Section 8.6.2 also leads to an improved ratio. Whether planar separators help in approximations to geometric problems is currently unclear. Grigni, Koutsoupias, and Papadimitriou [GEP95] devised a separator-based polynomial time approximation scheme for the traveling salesman problem on planar graphs with unit edge lengths; this result leads them to speculate that the Euclidean TSP has such an approximation scheme as well. Finally, improvements due to constraints on distances appear throughout this chapter. In a recent paper, Barvinok [Bar94] showed that the distance matrix implied by n points and a "polyhedral metric" has low "combinatorial rank," and exploited this fact to approximate the longest Euclidean traveling salesman tour arbitrarily well.

The same constraints that aid in approximation, however, create difficulties for problem reductions. For the purpose of reductions, planar geometric problems resemble problems on planar graphs, and proving a geometric problem *NP*-hard is usually much trickier than proving the same result for its non-geometric counterpart. For example, finding a longest traveling salesman tour is known to be *NP*-hard for general graphs, but is still open for planar graphs and for points in the Euclidean plane.

Even when reductions to geometric or planar-graph problems are known, they typically blow up the size of the instance by polynomial factors, and hence, do not preserve approximability. There are currently very few non-approximability results for these types of inputs. (A *non-approximability result* gives a lower bound on the approximation ratio achievable in polynomial time, assuming that $P \neq NP$.) The recent breakthrough results on non-approximability—see Chapter 10 by Arora and Lund—have had virtually no implications for geometric problems. Hence, we list the following as our foremost open problem: (Note added as this book goes to press: Arora [Aro96] has largely solved this problem; see our conclusion.)

OPEN PROBLEM 8.1 Are the famous geometric problems, such as the Euclidean traveling salesman problem, hard to approximate arbitrarily well?

A final consequence of geometric constraints is perhaps the most curious. Many problems involving Euclidean distance are not even known to be in *NP* due to a certain "technicality" [GGJ77]. For example, the straightforward decision-problem version of the traveling salesman problem asks whether there exists a tour of length at most k. The length of a tour is a sum of n square roots. Comparing this sum to the integer k may require very high precision arithmetic—for example, removing the square roots by repeated squaring gives numbers of about 2^n bits. This difficulty is quite distinct from *NP*-hardness; it arises even for decision-problem versions of easy problems such as the Euclidean minimum spanning tree. Algorithms in computational geometry typically skirt precision issues by assuming exact real arithmetic [Mul94, PS85]. This "real RAM" model of computation, however, is unsuitable for problem reductions.

OPEN PROBLEM 8.2 Is there a polynomial-time algorithm to compare an integer to the sum of n square roots of integers?

8.2 TRAVELING SALESMAN PROBLEM

We start with the most famous hard problem. The *Euclidean traveling salesman problem* is given by a set of points $S = \{s_1, s_2, \ldots, s_n\}$ in the plane with Euclidean metric, and asks for a shortest *tour* that visits all points. In other words, we seek a permutation σ

that minimizes the sum of distances

$$|s_{\sigma(1)}s_{\sigma(2)}| + |s_{\sigma(2)}s_{\sigma(3)}| + \cdots + |s_{\sigma(n-1)}s_{\sigma(n)}| + |s_{\sigma(n)}s_{\sigma(1)}|.$$

The triangle inequality implies that the diagonals of a convex quadrilateral are always longer than either pair of opposite sides; thus a pair of crossing edges in a tour can be replaced by a pair of disjoint edges, decreasing the overall length. Hence, the optimal tour must be a simple polygon.

Euclidean TSP is *NP*-hard [GJ79]. A network version of the problem is Max-$\mathcal{SNP}$-hard (see Chapter 10), even if edge lengths are restricted to 1 and 2 [PY93]. Hence, the network version cannot be approximated arbitrarily well (no polynomial-time approximation scheme exists) unless $P = NP$.

The best approximation algorithm for the Euclidean TSP is Christofides' algorithm, discovered in 1976. This algorithm achieves an approximation ratio of $\frac{3}{2}$. Remarkably, the algorithm uses no geometry beyond the triangle inequality, so it achieves this same ratio in any metric space. Failure to find a better approximation has led researchers to investigate ancillary questions, such as the approximation ratios of commonly used heuristics. We describe this work below. For more on the traveling salesman problem—mostly non-geometric—see the book edited by Lawler et al. [LLKS85]. For valuable practical information on the geometric case, consult the paper by Bentley [Ben92].

8.2.1 CHRISTOFIDES' ALGORITHM

Let $TSP(S)$ denote an optimal traveling salesman tour and $MST(S)$ denote a minimum spanning tree for points S, with lengths $|TSP(S)|$ and $|MST(S)|$ respectively. Notice that $|MST(S)| \leq |TSP(S)|$. Christofides' algorithm (see [GJ79, JP85]) starts by computing $MST(S)$. The minimum spanning tree alone gives a tour with an approximation ratio 2: simply walk around the "outside" of the tree to obtain a sequence $s_{t(1)}, s_{t(2)}, \ldots, s_{t(2n)}$ that visits each point twice. Eliminating repeats from the sequence $t(1), t(2), \ldots, t(2n)$, say by removing all occurrences of each index after the first, cannot lengthen the solution.

To improve the ratio, Christofides computes a minimum-length matching on the odd-degree vertices of $MST(S)$, as in Figure 8.1. Such a matching has length at most $\frac{1}{2}|TSP(S)|$, since the optimal traveling salesman tour has length at least that of an optimal

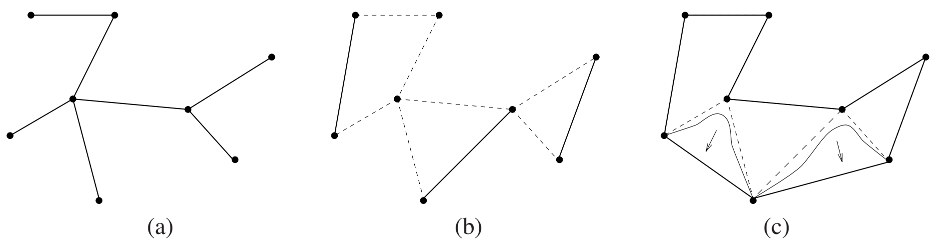


FIGURE 8.1

(a) The MST. (b) A minimum-length matching on odd-degree vertices. (c) The final tour.

tour on these odd-degree vertices, which can be partitioned into two matchings by taking every other edge. The MST along with the matching edges has only even-degree vertices, so it is Eulerian, and an Euler tour of this graph has length at most $\frac{3}{2}|TSP(S)|$. Eliminating repeats as above can only improve this tour. Figure 8.2 shows that the ratio of $\frac{3}{2}$ is tight. The running time of Christofides' algorithm depends on the time bound for Euclidean matching. Vaidya [Vai88] gave an $O(n^{2.5}\log^4 n)$ matching algorithm.

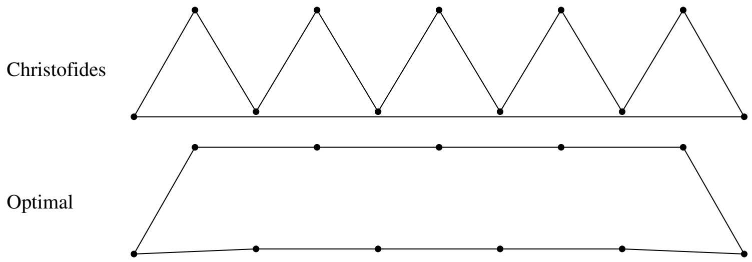


FIGURE 8.2

The tour found by Christofides' algorithm can have length $\frac{3}{2}$ times optimal.

8.2.2 HEURISTICS

An *insertion method* [RSL77] adds the points of S one by one in some order. A newly inserted point s_j is incorporated into the current tour by removing an edge s_is_k, and adding edges s_is_j and s_js_k. Edge s_is_k is always chosen to minimize the increase in the cost of the tour, $|s_is_j|+|s_js_k|-|s_is_k|$. However, various rules are used to pick the point to insert. Examples are *nearest insertion*, which inserts the point s_j nearest to a point already in the tour; *farthest insertion*, which inserts the point s_j that is farthest from any point in the tour; and *random insertion*, which chooses s_j uniformly at random. Rosenkrantz, Stearns, and Lewis [RSL77] proved the following upper bound.

THEOREM 8.1 Rosenkrantz et al. Any insertion order for S gives a tour of length at most $(\lceil\log_2 n\rceil+1)\cdot|TSP(S)|$.

Proof. Let $s_1, s_2, \ldots, s_n$ be an arbitrary insertion order, and let $INS(S)$ denote the resulting tour. Let $cost(s_j)$ denote the increase in the length of the tour upon the insertion of s_j. As above, if s_j is inserted between s_i and s_k, then $cost(s_j) = |s_is_j|+|s_js_k|-|s_is_k|$. We define $cost(s_1)=0$ and $cost(s_2)=2|s_1s_2|$, so that $\sum_{j=1}^{n} cost(s_j) = |INS(S)|$.

Now consider vertices s_i and s_j that are consecutive in the optimal tour $TSP(S)$. We assert that

$$2|s_is_j| \;\geq\; \min\{cost(s_i), cost(s_j)\}.$$

Why? If s_i was already in the tour when s_j was inserted, then we could have inserted s_j adjacent to s_i in the tour, and by the triangle inequality, $cost(s_j)$ would have been at most $2|s_is_j|$. A similar observation applies in the case $j<i$.

We now imagine charging the smaller of $cost(s_i)$ and $cost(s_j)$ to edge $s_i s_j$ in $TSP(S)$. Doing this for consecutive disjoint pairs of vertices from $TSP(S)$, we charge one vertex from each pair—a total of $\lfloor n/2 \rfloor$ vertices—to every other edge in $TSP(S)$. By charging to the shorter set of alternating edges in $TSP(S)$, we see that the total charge is at most $|TSP(S)|$.

We now repeat the argument on the vertices S' not yet paid for. This time insertion costs are assigned to every other edge of a tour on S' that respects the order given by $TSP(S)$. Edges of this tour correspond to disjoint sections of $TSP(S)$, so its length is at most $|TSP(S)|$. After $\lceil \log_2 n \rceil$ stages we get down to a tour on just two vertices, which again has length at most $|TSP(S)|$. ■

Bafna et al. [BKP94] and Azar [Aza94] (see also [AA93]) independently adapted a construction of Bentley and Saxe [BS80] to show that the bound of Theorem 8.1 is tight up to a factor of $O(\log\log n)$. We use a slightly simpler adaptation [CKT94] below.

THEOREM 8.2 Some insertion orders produce tours of length $\Omega(\log n/\log\log n)$ times optimal.

Proof. The set of points S includes $p+1$ vertical columns of vertices, numbered 0 to p, an odd integer. Along the top third of column i, there are $p^{2i}+1$ vertices, evenly spaced along a segment of length p^{2p}. The bottom third of column i is the same as the top third, after a gap of p^{2p}. The horizontal spacing between column i and column $i+1$ is $p^{2p-1-2i}$, which is $\frac{1}{p}$ times the basic vertical spacing in row i. The total number of vertices is about $3p^{2p}$, so p is $\Omega(\log n/\log\log n)$.

Assume the insertion order is left to right by columns and from the middle outwards, alternating above and below within each column. This order gives the tour on the left in Figure 8.3, which has total length greater than $4p \cdot p^{2p}$. (For clarity, Figure 8.3 does not show all of S; the third column should really have 20 points, and the fourth 164.) The illustration on the far right shows a spanning tree for this same set of points with total length $O(p^{2p})$, proving that $|TSP(S)|$ is also $O(p^{2p})$. ■

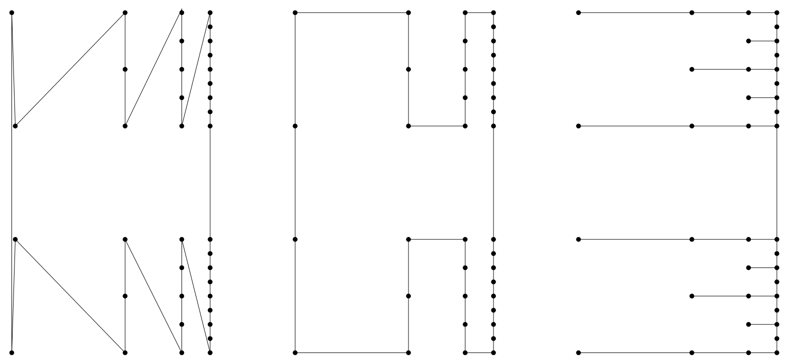


FIGURE 8.3

An insertion tour or a 2-optimized tour can have length $\Omega(\log n/\log\log n)$ times the MST.

We now discuss specific insertion methods. Nearest insertion [RSL77] achieves an approximation ratio of 2, because the jth point never costs more than twice the jth edge chosen by Prim's MST algorithm. This bound is known to be tight. Warburton [War93] considered insertion methods that start with the convex hull instead of a trivial tour, and showed that the bound for nearest insertion then degrades to 3. Somewhat surprisingly, farthest insertion typically outperforms nearest insertion [Ben92], and its exact approximation ratio is currently unknown. However, Hurkens [Hur92] recently showed that this ratio cannot be smaller than 2.43. Azar [Aza94] proved that there are point sets for which random insertion has approximation ratio $\Omega(\log\log n/\log\log\log n)$.

We now move on to another heuristic, that can be used either as an algorithm in its own right, or as a post-processing step to locally optimize a tour. The *2-Opt heuristic* starts from any tour of S, and repeatedly swaps one pair of edges for another pair with the same endpoints and shorter total length. The swapped-out pair may be a crossing pair, but this is not a requirement. The swapped-in pair is determined by the need to form a tour rather than two disjoint cycles, and the new tour is the same as the old only with a contiguous subsequence reversed. The algorithm halts when the tour is *2-optimized*, meaning that there are no 2-Opt swaps that reduce its total length.

There are numerous rules—first found, best first, randomized—that specify which 2-Opt swap to make [Ben92], but in studying the approximation ratio of this heuristic, we shall assume an arbitrary or worst-case order of swaps. Chandra, Karloff, and Tovey [CKT94] showed the following upper bound:

THEOREM 8.3 Any 2-optimized tour has length $O(\log n)$ times optimal.

Proof. Let $OPT2(S)$ be a 2-optimized tour with edges directed so that each has an "origin" and a "destination." Divide the edges of $OPT2(S)$ into—say—18 equivalence classes, based on directions: an edge belongs to class i, $0 \le i \le 17$ if its angle with the horizontal lies in the range $[i \cdot 10^\circ, (i+1)\cdot 10^\circ)$. It is not hard to show that two edges $s_i s_j$ and $s_k s_l$ from the same equivalence class must have origins at least $\frac{1}{2}\min\{|s_i s_j|, |s_k s_l|\}$ apart, for otherwise we could swap them for a shorter pair.

We bound the total length within each equivalence class separately using an argument similar to the proof of Theorem 8.1. Let E_i be the edges of equivalence class i. Sort the edges of E_i by the order of their origins in a walk around $TSP(S)$. Consider consecutive edges $s_i s_j$ and $s_k s_l$ from E_i. By the observation above, $|s_i s_k| \ge \frac{1}{2}\min\{|s_i s_j|, |s_k s_l|\}$. Hence, we can "charge" edge $s_i s_j$ of E_i to edge $s_i s_k$ in $TSP(S)$. Doing this for consecutive disjoint pairs from E_i, we charge all shorter edges—a total of $\lfloor |E_i|/2 \rfloor$ edges—to distinct edges of $TSP(S)$. We now repeat the argument, this time charging to disjoint sections of $TSP(S)$, again halving the number of edges. After $O(\log n)$ stages, all edges are charged. ■

Theorem 8.3 generalizes in a straightforward way to arbitrary dimension. Chandra et al. proved an almost matching lower bound with the construction of Figure 8.3. A case analysis shows that the tour in the middle illustration is 2-optimized.

THEOREM 8.4 There are 2-optimized tours of length $\Omega(\log n/\log\log n)$ times optimal.

The 2-Opt heuristic can be generalized to the k-Opt heuristic, in which sets of up to k edges are removed and replaced by shorter sets. The upper bound of Theorem 8.3 of course applies to this generalization. Good lower bounds for k-Opt in the geometric case are unknown, although Rosenkrantz et al. [RSL77] proved that even for $k = \frac{n}{4}$, the approximation ratio may be as bad as 2.

OPEN PROBLEM 8.3 Is there a polynomial-time algorithm for the Euclidean TSP with approximation ratio smaller than $\frac{3}{2}$? What is the approximation ratio of the farthest insertion heuristic?

8.2.3 TSP WITH NEIGHBORHOODS

In *TSP with neighborhoods*, we are given a collection of k simple polygons (not necessarily disjoint) with n total vertices, and we seek the shortest tour that passes through each polygon. Obviously, the usual TSP is the special case in which each polygon is simply a point. Arkin and Hassin [AH94b] gave an $O(1)$ approximation algorithm for the case of "round," approximately equal-sized neighborhoods, such as unit disks.

In more recent work, Mata and Mitchell [MM95] provide a general framework that gives an $O(\log k)$ approximation algorithm for this problem. The same framework yields logarithmic approximation ratios for three other problems that generalize the TSP: (1) the *prize-collecting TSP*, in which we seek a shortest tour visiting k out of n points (see Section 8.5.2 for a closely related problem); (2) the *watchman route problem*, in which we seek the shortest tour that can "see" all of a polygon with holes; and (3) the *red-blue separation problem*, in which we are given two point sets—red and blue—and seek the shortest polygon that contains all of the red and none of the blue points. (Using a result on guillotine trees—Lemma 8.6 in Section 8.5.2—Mata and Mitchell have since improved the approximation ratio for problem (3) to $O(1)$.)

The framework's overall strategy has been previously applied to several other geometric approximation problems (Sections 8.5.2 and 8.6.3). The first step is to show that the optimal solution can be approximated within a factor of $O(\log k)$ by a spatial subdivision of a particular form. The second step is to devise a polynomial-time dynamic programming algorithm to compute an optimal subdivision. The last step is to transform the optimal subdivision back into a solution to the original problem with only a constant factor increase in length.

To start the first step, Mata and Mitchell show that the optimal solution for the TSP with neighborhoods can be approximated within a constant factor by an *orthogonal polygon*, meaning one in which each side is either horizontal or vertical. They further show that this orthogonal polygon can be assumed to be a subgraph of the *grid graph* formed by passing a horizontal and a vertical line through each of the n input vertices.

A *binary space partition* (see [PY90, PY92]) is a recursive subdivision of the plane (or of a bounding box) into convex cells, formed by repeatedly cutting cells with line segments that contain portions of input line segments. The following lemma shows that any orthogonal polygon admits a relatively short binary space partition.

LEMMA 8.1 Let P be a k-vertex orthogonal polygon. There is a binary space partition of P (starting from P's bounding box) with total cut length $O(\log k)$ times the perimeter of P.

Proof. We form a partition by the following method: while some rectangular cell c contains a vertex of P, split c with the line segment that is parallel to the short side of c, and that passes through a vertex of P and lies as close as possible to the center of c. Finally, when all cells contain only edges of P, split the cells along those edges.

Let us add up the total cut length of this partition. The segments in the final stage in which we split along edges of P, add up to at most ℓ, where ℓ is the perimeter of P, so we need only worry about earlier splits. We partition earlier cutting segments into length classes: $[0, \ell/k]$, $[\ell/k, 2\ell/k]$, $[2\ell/k, 4\ell/k]$, and so forth. There are $O(k)$ segments overall, since each one passes through a vertex of P, so the segments in class one have total length at most $O(\ell)$. The length in each remaining class is at most a constant times the length of the minimum spanning tree of the vertices of P, and hence, also $O(\ell)$. ■

We now explain the second step: a dynamic programming algorithm that computes a shortest total-length binary space partition that cuts each input polygon at least once. For each rectangle R of the grid graph, in order by size from smallest to largest, we compute the minimum length binary space partition that has R as its root (largest cell). If no input polygons lie entirely interior to R, then the minimum partition is R itself. Otherwise, for each of the $O(n)$ choices for the first cutting segment, we compute the length of the minimum partition of R by combining the lengths of partitions of two smaller rectangles. Since there are $O(n^4)$ rectangles in the grid graph, the overall computation takes $O(n^5)$ time.

Finally, we must translate the optimal partition back to a solution to the original problem. To do this, we compute the points where the optimal partition crosses each input polygon, and then form a tour by doubling the MST of the crossing points.

THEOREM 8.5 Mata and Mitchell The algorithm above is an $O(\log k)$ approximation algorithm for TSP with neighborhoods.

8.3 STEINER TREE PROBLEM

The *Euclidean Steiner tree problem* is given by a set of points $S = \{s_1, s_2, \ldots, s_n\}$ in the Euclidean plane, and asks for the shortest planar straight-line graph spanning S. The solution takes the form of a tree, called a Steiner minimal tree, that includes all the points of S, called *terminals*, along with zero or more extra vertices, called *Steiner points*. It is well known [GP68] that exactly three edges meet at each Steiner point and form 120° angles. A minimum spanning tree can be considered a suboptimal Steiner tree.

There are several variants of the Steiner problem. The *rectilinear Steiner problem* uses the rectilinear metric, in which the distance between points $s_i = (x_i, y_i)$ and

$s_j = (x_j, y_j)$ is $|x_i - x_j| + |y_i - y_j|$. In the *network Steiner problem*, S is a subset of the vertices in an edge-weighted graph and the desired tree is a subgraph spanning S. Although this version of the problem is not geometric—its most frequent recent use seems to be in the field of phylogeny [HRW92]—approximation algorithms for the geometric versions build on ideas used in the network case. The rectilinear version can be reduced to the network version on the grid graph formed by passing a horizontal and a vertical through each terminal. The books by Hwang, Richards, and Winter [HRW92] and Ivanov and Tuzhilin [IT94] survey the Steiner tree literature.

All three major versions of the Steiner problem are *NP*-hard. Although the network and rectilinear problems are *NP*-complete, the Euclidean problem involves sums of square roots like the traveling salesman and minimum spanning tree problems, and thus is not known to be in *NP* [GJ79]. The network Steiner problem is known to be Max-$\mathcal{SNP}$-hard, even if edge lengths are restricted to 1 and 2 [BP89]. The geometric versions of the problem—like all but a very few geometric problems—are not known to be Max-$\mathcal{SNP}$-hard.

In constrast to the traveling salesman problem, there have been recent breakthroughs in approximating the Steiner tree problem. In 1990, Du and Hwang [DH92a] proved that the Euclidean Steiner ratio—the maximum ratio of the lengths of minimum spanning tree and Steiner minimal tree—is $\frac{2}{\sqrt{3}} \approx 1.1547$, settling a long-open conjecture. In the same year, Zelikovsky [Zel93] gave a polynomial-time algorithm for the network Steiner tree problem with approximation ratio $\frac{11}{6}$, beating the previous bound of 2 given by a minimum spanning tree in which shortest paths between terminals act as tree edges. Zelikovsky's algorithm approximates the optimal Steiner tree by approximating an optimal Steiner tree that has full components of bounded size. (A *full component* is a subtree in which each terminal is a leaf; the tree in Figure 8.4 has three full components, of sizes 5, 3, and 3.) This algorithm has led to improved ratios for the rectilinear and Euclidean cases.

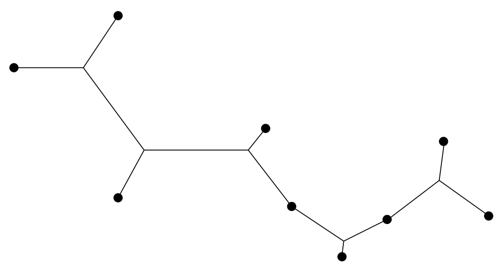


FIGURE 8.4

Terminals are represented by black dots in this Euclidean Steiner minimal tree.

8.3.1 STEINER RATIOS

In this section we view the minimum spanning tree as an approximate Steiner tree and study its worst-case approximation ratio. We define the *Steiner ratio* to be the supremum over all point sets of the length of the minimum spanning tree divided by the length of the Steiner minimal tree.[1] We treat the simpler rectilinear problem first.

8.3.1.1 Rectilinear

The example of four points at the corners of the unit ball—the points $(1,0)$, $(0,1)$, $(-1,0)$, and $(0,-1)$—shows that the rectilinear Steiner ratio may be as large as $\frac{3}{2}$. Hwang [Hwa76] proved that this ratio is never larger.

THEOREM 8.6 Hwang The rectilinear minimum spanning tree is never longer than three-halves the length of the rectilinear Steiner minimal tree.

Proof Sketch. We now sketch Hwang's proof. Let S be a set of n terminals in the plane with rectilinear Steiner minimal tree $RST(S)$ and rectilinear minimum spanning tree $RMST(S)$ of lengths $|RST(S)|$ and $|RMST(S)|$. We use induction on n, with basis of the trivial case of $n = 2$.

First assume that some terminal s_i has degree at least two in $RST(S)$. Removing s_i splits $RST(S)$ into two or more components. Let S_1 be the terminals in one of these components along with s_i, and let $S_2 = \{s_i\} \cup S \setminus S_1$. By the inductive hypothesis, Theorem 8.6 holds for S_1 and S_2. Now the minimum spanning tree $RMST(S)$ is no longer than the union of $RMST(S_1)$ and $RMST(S_2)$, and the Steiner tree $RST(S)$ is exactly the union of $RST(S_1)$ and $RST(S_2)$, so Theorem 8.6 must also hold for S.

We may now assume that $RST(S)$ is full, that is, has just one full component. Hwang proves that in this case there always exists a shortest tree that has one of the two canonical forms shown on the left in Figure 8.5: either all Steiner points, or all Steiner points but one, lie along a straight "spine," and straight edges leading to terminals alternate from left to right along this spine. (A Steiner point of degree 4 can be considered the limiting

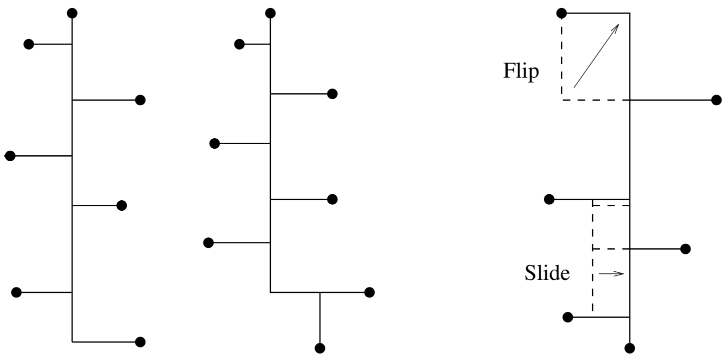


FIGURE 8.5

Canonical forms for full rectilinear Steiner minimal trees.

[1]Most authors define the Steiner ratio the other way around: Steiner minimal tree divided by MST. We use the reciprocal so that the Steiner ratio is an approximation ratio.

case of two very closely spaced Steiner points of degree 3.) Any full Steiner minimal tree can be transformed into one of these two forms (or made non-full) by a sequence of *flipping* and *sliding* operations, as shown on the right.

The last step, simplified by Berman et al. [BFK$^+$94, BR94], shows that there must always be a spanning tree of length at most $\frac{3}{2}|RST(S)|$. Assume the full Steiner tree $RST(S)$ has the form shown on the left in Figure 8.6; the other case is similar. On the right are two spanning trees of total length at most three times $|RST(S)|$; the shorter of the two has length at most $\frac{3}{2}|RST(S)|$. The spanning trees (solid lines) charge their excess length to disjoint sets of Steiner tree edges (bold dotted lines). Each of these sets includes alternating segments along the spine, along with longest "ribs" between successive locally minimal ribs. A locally minimal rib is one that is shorter than its two same-side neighbors. ■

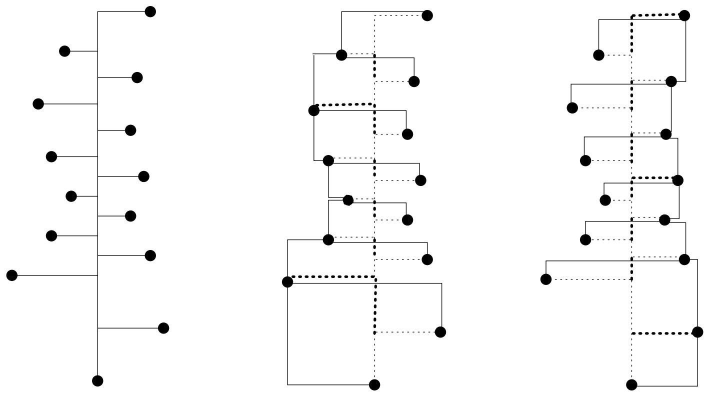


FIGURE 8.6

Two spanning trees (right) have length at most three times the full Steiner tree.

8.3.1.2 Euclidean

Before its recent solution, the Euclidean Steiner ratio was a famous open problem. In 1968 Gilbert and Pollak [GP68] observed that the ratio of $\frac{2}{\sqrt{3}} \approx 1.1547$ holds for the vertices of an equilateral triangle and conjectured that no point set achieves smaller ratio. For years researchers inched towards this conjecture on two fronts: successively establishing that $\frac{2}{\sqrt{3}}$ does indeed hold for 3, 4, 5, and 6 points and that weaker upper bounds (2, 1.75, 1.35, 1.25, and 1.21) hold for arbitrary point sets [DH92b, HRW92]. Finally, in 1990 Du and Hwang [DH92a] announced a proof of the conjecture. Their proof incorporates some ideas of Rubinstein and Thomas [RT91a, RT91b] and Meeks and Yau [MY80].

THEOREM 8.7 Du and Hwang The Euclidean MST is never longer than $\frac{2}{\sqrt{3}}$ times the length of the Euclidean Steiner minimal tree.

We now give a detailed sketch of Du and Hwang's remarkable proof [DH92a]. Let S denote a set of n points in the Euclidean plane. Let $ST(S)$ and $MST(S)$ denote a Steiner minimal tree and a minimum spanning tree for terminals S of lengths $|ST(S)|$ and $|MST(S)|$. The *topology* of a tree is its pattern of connections (with vertices labeled), which can be specified, among other ways, by an adjacency matrix.

By induction on n, we can, as in the rectilinear problem, limit attention to the case that $ST(S)$ is a full Steiner tree. Du and Hwang simplify matters by fixing a topology T suitable for a full Steiner tree, and bounding the ratio of $|MST(S)|$ to $|ST_T(S)|$, the length of a shortest Steiner tree of topology T. Topology T is arbitrary, so this bound will suffice. Du and Hwang focus attention still further by scaling S so that $|ST_T(S)| = 1$. This step converts the problem of bounding the ratio $|MST(S)|/|ST_T(S)|$ to the problem of bounding $|MST(S)|$. The proof now aims to show that S must be a very special set of points in order to maximize $|MST(S)|$. We preview the approach before diving into the details.

A point set S will be identified—up to rigid motions—with a vector x in $\mathbb{R}^{2n-3}$. For any given MST topology U, the length $|MST_U(S)|$ turns out to be a convex function (like an upwards-opening parabola) in this parametrization of S. The function we are interested in, $|MST(S)|$, is the lower envelope (pointwise minimum) of these convex functions, which implies that its maximum value is attained at a "critical vector," a point set with a maximal set of MST topologies.

We can illustrate the idea with just four points. The set of points shown on the left in Figure 8.7 has two different MSTs, shown solid and dashed. By moving point s_3 slightly, we obtain a point set—the vertices of a 60°–120° rhombus—with the same two MSTs spanning trees along with some others. This new point set in fact supports a maximal set of MST topologies.

In general, a point set with a maximal set of MST topologies must be rather special; for example, many pairs of points determine the same distance. For more than four points, however, such a point set is not so special as to render the Steiner ratio problem easy. An obstacle arises when the union of all MST edges completely encloses a point. To overcome this obstacle, Du and Hwang limit attention to certain spanning tree topologies, called "inner" topologies, that lie inside a polygon induced by $ST_T(S)$. This seemingly small shift clears away a thicket of difficulties. A point set with a maximal set of minimum inner spanning tree topologies is indeed very special: it must be exactly the vertices of a triangulated simple polygon in which each triangle is equilateral, as in the 60°–120° rhombus. The Steiner ratio problem for these very constrained point sets can be solved by elementary arguments, which will complete the proof.

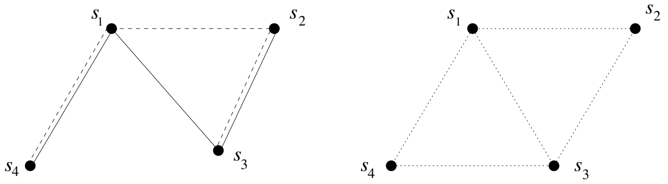


FIGURE 8.7

Moving s_3 slightly gives a point set with a maximal set of MST topologies.

Parametrizing Point Sets. Let T be the topology of a full Euclidean Steiner tree, given by an adjacency matrix with $2n-2$ rows and columns. In something of a conceptual switch, we can use topology T to parametrize point sets. Point set $S\ (= S(T,x))$ is specified by the lengths of its edges in a shortest Steiner tree $ST_T(S)$ of topology T. We denote these lengths by a vector $x = (x_1, x_2, \ldots, x_{2n-3})$ and limit attention to vectors x such that $\sum_i x_i = 1$. The $120°$ rule, which holds for any locally minimal Steiner tree such as $ST_T(S)$, fixes the relative directions of edges in $ST_T(S)$, so that any vector x with each $x_i \geq 0$ and $\sum_i x_i = 1$ encodes a single planar point set, modulo equivalence by rigid motions and scaling. Now define

$$\Delta = \left\{ x \in \mathbb{R}^{2n-3} \;\middle|\; \sum_{i=1}^{2n-3} x_i = 1 \text{ and } \forall i, x_i \geq 0 \right\}.$$

So that domain Δ is closed, we must allow zero-length edges in $ST_T(S)$; we regard such a tree as a degenerate version of topology T. Certain x vectors may force edges of $ST_T(S)$ to cross. We can think of such anomalous trees—which cannot be Steiner minimal trees in any case—as embedded in a plane with Riemann sheets attached.

The Lower Envelope of Convex Functions. We now prove that the length of a fixed-topology spanning tree is a convex function of x. Let U be a topology of a spanning tree on n vertices, and let $MST_U(S)$ denote the spanning tree of S of topology U. A function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is *convex* if for each α, $0 < \alpha < 1$, and each x and x' in $\mathbb{R}^d$, $f(\alpha x + (1-\alpha)x') \leq \alpha f(x) + (1-\alpha) f(x')$.

LEMMA 8.2 $|MST_U(S)|\ (= |MST_U(S(T,x))|)$ is a convex function of x.

Proof. Consider any two terminals s_i and s_j implied by x and T. We first show that edge length $|s_i s_j|$ is a convex function of x. Suppose the path in $ST_T(S)$ connecting s_i and s_j consists of edges with lengths $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ and directions given by unit vectors $e_1, e_2, \ldots, e_k$. Then, $|s_i s_j|$ is the Euclidean norm of the vector $x_{i_1} e_1 + x_{i_2} e_2 + \cdots + x_{i_k} e_k$. As x varies along a line from x to x', each x_{i_j} varies linearly. The unit vectors $e_1, e_2, \ldots, e_k$ change in concert in order to maintain the $120°$ rule; rotating the coordinate system by the same amount keeps $e_1, e_2, \ldots, e_k$ fixed. The Euclidean norm is itself a convex function, and a convex function of a linear function is convex, so $|s_i s_j|$ is convex. Now $|MST_U(S)|$ is a sum over a fixed subset of $|s_i s_j|$ functions. It is easy to see that a sum of convex functions is convex. ■

The length of a minimum spanning tree, or—looking ahead—the length of a minimum inner spanning tree, is the lower envelope of $|MST_U(S)|$ functions. Consider the graph of a function that is the lower envelope of convex functions on the plane. Each of the convex functions forms a bowl-shaped valley. Where two convex functions are both minimum, the lower envelope has a ridge, and where three or more are all minimum, there is a peak. Lemma 8.3 formalizes this observation and generalizes it to arbitrary dimension.

Let f be the lower envelope of a finite set of convex functions $f_1, f_2, \ldots, f_k$, that is, $f(x) = \min_i f_i(x)$. We say that x is a *critical vector* if a maximal set of f_i's have $f_i(x) = f(x)$. In other words, if $I = \{i \mid f_i(x) = f(x)\}$, there is no x' such that $f_i(x') = f(x')$ for each $i \in I$ and $f_j(x') = f(x')$ for some $j \notin I$.

LEMMA 8.3 Let $f : \Delta \mapsto \mathbb{R}$ be the lower envelope of a finite set of convex functions. Then f achieves its maximum either on the boundary of Δ or at a critical vector.

Inner Spanning Trees. We can form a polygon by connecting successive terminals in a walk around the outside of the Steiner tree $ST_T(S)$, as shown in Figure 8.8. We call this polygon the *Steiner tree polygon* and denote it by $P_T(S)$. It is possible for the Steiner tree polygon to self-overlap, but this case is handled with Riemann sheets. A spanning tree $MST_U(S)$ is an *inner spanning tree* if each of its edges lies within the closed region bounded by $P_T(S)$. Let $|MIST(S)|$ denote the length of a *minimum inner spanning tree* for point set S.

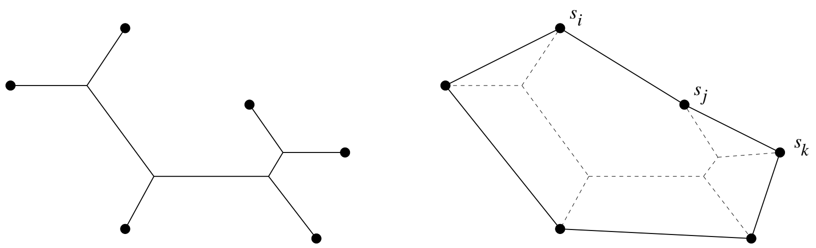


FIGURE 8.8

Full Steiner tree $ST_T(S)$ and the polygon $P_T(S)$ it induces.

The set of inner spanning tree topologies varies with x. For example, in Figure 8.8 a spanning tree with an edge between points s_i and s_k is not an inner spanning tree, but would be one if the edge to s_j were lengthened a little, so that $s_i s_k$ lay below s_j. This variability poses a potential problem. Since each $|MST_U(S)|$ function participates over only part of domain Δ, we cannot apply Lemma 8.3 directly to $|MIST(S)|$. Luckily, a spanning tree $MST_U(S)$ cannot be minimum on the boundary of its "region of innerness". Figure 8.8 shows why: if $MST_U(S)$ is not inner for all small perturbations of S (that is, in a small ball around x), then a terminal s_j must lie in the relative interior of an edge $s_i s_k$. It is not hard to show that Lemma 8.3 still holds for the lower envelope of convex partial functions, so long as the partial functions are never minimal on the boundaries of their domains.

So we now know that $|MIST(S)|$ achieves its maximum either on the boundary of Δ, that is, when some $x_i = 0$, or at a critical vector x. In the former case some edge of $ST_T(S)$ has zero length. If this edge is incident to a terminal, then $ST_T(S)$ is not really a full Steiner tree. On the other hand, if the zero-length edge links two Steiner points, then $ST_T(S)$ violates the degree-3 rule. We have now eliminated the boundary of Δ and can assume that S supports a maximal set of minimum inner spanning trees.

Equilateral Formation. We say that a set of points S is in *equilateral formation* if there exists a simple polygon P with vertex set S that has a triangulation in which each triangle is equilateral (and hence congruent).

LEMMA 8.4 If x is a critical vector, then S is in equilateral formation.

Proof. Let E be the set of all edges from all minimum inner spanning trees of S. Imagine adding E to Steiner tree polygon $P_T(S)$. It is not hard to show by contradiction that no two edges of E cross. (If two do, remove the longer one and find a shorter path inside $P_T(S)$ reconnecting the connected components.)

Since no two minimum inner spanning tree edges cross, E divides $P_T(S)$ into polygons with vertices from S. Imagine adding extra edges if necessary to triangulate $P_T(S)$. The number of distinct edges in this triangulation—diagonals and boundary—is $2n-3$. We make two key observations: (1) S is determined (up to rigid motions) by the lengths of these edges, and (2) each of these lengths can be varied independently—for example, lengthening a single edge opens up the opposite angle and spreads out $P_T(S)$ (perhaps causing it to self-overlap). The remainder of the proof repeatedly applies observation (2) to ears (triangles with two boundary sides), in order to establish that each triangle in the triangulation must be equilateral. ■

The next lemma completes the proof of Theorem 8.7.

LEMMA 8.5 If S is in equilateral formation, then $\frac{|MST(S)|}{|ST(S)|} \leq \frac{2}{\sqrt{3}}$.

Proof. To prove this lemma, we introduce another kind of tree, called a *hexagonal Steiner tree*. In such a tree, each edge is parallel to one of three directions: horizontal, 60° from horizontal, or -60° from horizontal. Let $HT(S)$ be a shortest hexagonal Steiner tree for S. For any point set, $|HT(S)|/|ST(S)| \leq \frac{2}{\sqrt{3}}$. Why? Any edge of a Steiner minimal tree can be replaced by a dog-leg of two edges of the prescribed directions, expanding its length by at most $\frac{2}{\sqrt{3}}$. The remainder of the proof establishes that for S in equilateral formation, a MST is in fact a shortest hexagonal Steiner tree for some choice of coordinate axes. This is proved by showing that if S consists of points from a hexagonal lattice, flipping and sliding operations reminiscent of the rectilinear Steiner problem can move all Steiner points of $HT(S)$ to lattice points. See Figure 8.9. ■

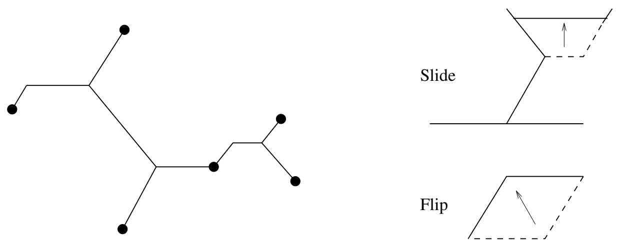


FIGURE 8.9

Hexagonal Steiner tree.

8.3.1.3 Other metrics

We need not limit attention to the rectilinear and Euclidean metrics. Any compact, convex, centrally symmetric shape D can be used to define a *Minkowski metric* on the plane. The distance between points s and s' is measured by translating unit ball D so that its center coincides with s and then shrinking or expanding D until s' lies on its boundary; the factor by which D is dilated is the distance. Basic properties of Steiner minimal trees for such a metric—for example, the degree of Steiner points—depend on whether or not D has a smooth boundary (see [HRW92]). Research on the Steiner ratio question for Minkowski metrics has expanded recently, stimulated by Du and Hwang's solution of the Euclidean problem.

Cieslik [Cie90] and Du, Gao, Graham, Liu, and Wan [DGG$^+$] independently conjectured that for any D the Steiner ratio lies in the range $[\frac{2}{\sqrt{3}}, 1.5] \approx [1.154, 1.5]$, in other words, between the Euclidean and rectilinear cases. Cieslik obtained bounds of 1.106 and 1.634, while Du et al. gave 1.151 and 1.605. Very recently, Gao, Du, and Graham [GDG94] obtained the tight upper bound of 1.5. They showed, moreover, that if a finite point set S has ratio 1.5, then ball D must be a parallelogram and S must have exactly 4 terminals arranged in a parallelogram similar to D. Gao et al. go on to conjecture that for any Minkowski metric some finite point set achieves the Steiner ratio.

We can also imagine generalizing Steiner ratio problems to higher dimensions. Graham and Hwang conjectured that the rectilinear Steiner ratio in $\mathbb{R}^d$ is $\frac{2d-1}{d}$, achieved by the vertices of the unit ball [GH76]. On the basis of some empirical work, Gilbert and Pollak [GP68] speculated that the Euclidean Steiner ratio in $\mathbb{R}^d$ is achieved by the vertices of a regular simplex. Smith [Smi92] disproved this conjecture for $\mathbb{R}^3$ by showing that the ratio for the regular octahedron is greater than that for the regular tetrahedron. (Gilbert and Pollak actually considered the octahedron, but they had the wrong Steiner tree!) The three-dimensional octahedron serves as a counterexample for dimensions up to nine. Du [Du92] gave a counterexample construction—four regular simplices with one vertex in common—that holds for all $d \geq 3$.

OPEN PROBLEM 8.4 Does the Euclidean metric indeed have the smallest Steiner ratio of all Minkowski metrics? What are the Steiner ratios for the rectilinear metric in higher dimensions? For the Euclidean metric?

8.3.2 BETTER APPROXIMATIONS

Does there exist a polynomial-time algorithm with an approximation ratio better than that given by the minimum spanning tree? Heuristics routinely outperform the MST in practice, and a simple, greedy heuristic gives superior results for random points in a square [Ber87, Ber88, HY90]; however, until recently no better worst-case approximation algorithm was known for any *NP*-hard Steiner tree problem, except for the very simple case of a complete graph with all edge lengths either 1 or 2 [BP89]. This situation changed quickly with the circulation of a 1990 manuscript from the former USSR. We now describe Zelikovsky's work [Zel93] and the ensuing extensions to geometric Steiner problems.

8.3.2.1 Greedy k-restricted Steiner trees

We say that a Steiner tree for terminals S is a *k-restricted Steiner tree* if each full component spans at most k terminals. Let $ST_k(S)$ denote the shortest k-restricted Steiner tree and ρ_k be the supremum over all S of $|ST_k(S)|\,/\,|ST(S)|$. Notice that ρ_2 is the usual Steiner ratio. For all three versions of the Steiner problem, $ST_k(S)$ for $k > 2$ appears hard to compute; the rectilinear version with $k = 4$ is known to be *NP*-complete, a byproduct of the original reduction [GJ77]. The tree $ST_k(S)$ can, however, be approximated by a simple, greedy algorithm. In this subsection, we shall bound the ratio of the greedy tree to the optimal unrestricted tree $ST(S)$ as a function of ρ_k values. The next subsection then considers the problem of bounding the ρ_k's.

Assume $T_1, T_2, \ldots, T_m$ are full Steiner trees for subsets of S, such that the union of the T_i's does not contain a cycle. Let $MST(S; T_1, \ldots, T_m)$ be a Steiner tree consisting of the trees $T_1, \ldots, T_m$, along with a minimum spanning forest that connects all T_i's and all members of S not in some T_i. In other words, $|MST(S; T_1, \ldots, T_m)|$ is the length of a MST constrained to include all T_i's. Kruskal's algorithm for computing MSTs can also compute $MST(S; T_1, \ldots, T_m)$.

The greedy algorithm starts by choosing a T_1 that minimizes $|MST(S; T)|$ over all choices of T with at most k terminals. At each subsequent step, it continues to choose a full Steiner tree on at most k terminals that offers the most *gain* over the MST. In other words, at step j it chooses a tree T_j that minimizes $|MST(S; T_1, \ldots, T_{j-1}, T_j)|$, or equivalently, maximizes the gain

$$W(T_j \mid T_1, \ldots, T_{j-1}) = |MST(S; T_1, \ldots, T_{j-1})| - |MST(S; T_1, \ldots, T_{j-1}, T_j)|.$$

By allowing full Steiner trees to span only two terminals, the greedy algorithm finishes with a minimum spanning forest when larger full Steiner trees run out. Theorem 8.8 for $k = 3$ is due to Zelikovsky [Zel93]; Du, Zhang, and Feng [DZF91] made the (straightforward) generalization to arbitrary k.

THEOREM 8.8 The greedy algorithm has ratio at most $\frac{k-2}{k-1}\,\rho_2 + \frac{1}{k-1}\,\rho_k$.

Proof. Let $T_1, \ldots, T_\ell$ be the full Steiner trees chosen by the greedy algorithm, including 2-terminal trees (edges). Let $T_1^*, \ldots, T_m^*$ be the full Steiner trees of an optimal k-restricted Steiner tree $ST_k(S)$.

Imagine adding T_1 to the optimal tree $ST_k(S)$. If T_1 spans h terminals, $2 \le h \le k$, then $T_1 \cup ST_k(S)$ contains $h - 1$ simple cycles, as shown in Figure 8.10(a). To break these cycles we shall remove $h - 1$ full components, renumbered to be $T_1^*, \ldots, T_{h-1}^*$, from $ST_k(S)$. Now the greedy algorithm chooses T_1 to be the full tree that minimizes $|MST(S; T_1)|$, so $|MST(S; T_1)| \le \min_{1 \le j \le h-1} |MST(S; T_j^*)|$, and no matter how we pick $T_1^*, \ldots, T_{h-1}^*$,

$$|MST(S; T_1)| \;\le\; \frac{(h-2)|MST(S)| + |MST(S; T_1^*, \ldots, T_{h-1}^*)|}{h-1}.$$

Our goal is to pick $T_1^*, \ldots, T_{h-1}^*$ cleverly so that the following statement about "future gains" holds as well:

$$|MST(S; T_1)| - \sum_{j=1}^{\ell} |T_j| \;\ge\; \frac{|MST(S; T_1^*, \ldots T_{h-1}^*)| - \sum_{j=1}^{m} |T_j^*|}{h-1}.$$

We can then sum the first inequality with -1 times the second to obtain the result.

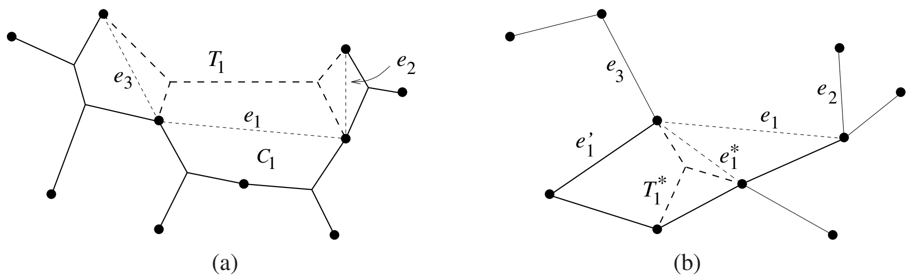


FIGURE 8.10

(a) Adding greedy T_1 to the optimal k-restricted tree $ST_k(S)$. (b) The tree $MST(S; e_2, \ldots, e_{h-1})$ helps locate the full component T_1^* to remove.

Intuitively, we would like T_i^*'s that realize a lot of gain up front, so we look for T_i^*'s that "shortcut" long edges in $MST(S; T_1)$. Let $e_1, e_2, \ldots, e_{h-1}$ be the edges of any spanning tree on the terminals of T_1, as shown in Figure 8.10(a). We use e_1 to pick T_1^*, e_2 to pick T_2^*, and so forth. Let e_1' be the longest edge on the path between the endpoints of e_1 in the tree $MST(S; e_2, \ldots, e_{h-1})$ as shown in Figure 8.10(b). Let C_1 be the cycle formed by adding e_1 to $ST_k(S)$, and imagine connecting successive terminals around C_1. Some edge e_1^* between successive terminals of some full component T_1^* shortcuts e_1', meaning that the path between the endpoints of e_1^* back in $MST(S; e_2, \ldots, e_{h-1})$ includes e_1'. Since edges e_1 and e_1^* remove the same edge (namely e_1') from $|MST(S; e_2, \ldots e_{h-1})|$,

$$|MST(S; e_1, e_2, \ldots, e_{h-1})| - |e_1| = |MST(S; e_1^*, e_2, \ldots, e_{h-1})| - |e_1^*|,$$

which after some algebraic manipulations implies that

$$|MST(S; T_1)| - \sum_{j=1}^{\ell} |T_j| \geq \frac{|MST(S; T_1^*, e_2, \ldots, e_{h-1})| - \sum_{j=1}^{m} |T_j^*|}{h-1}.$$

Now let e_2' be the longest edge on the path between the endpoints of e_2 in the tree $MST(S; T_1^*, e_3, \ldots, e_{h-1})$. Edge e_2' locates a T_2^* in $ST_k(S) \setminus T_1^*$ that replaces e_2 in the last inequality above. Proceeding in this manner, we choose a set of suitable T_i^*'s. (It is possible that a single T_i^* can span the endpoints of more than one of $e_1, \ldots, e_{h-1}$, in which case we end up with fewer than $h-1$ T_i^*'s.) ■

8.3.2.2 Steiner ratios for k-restricted Steiner trees

Theorem 8.8 implies the existence of approximation algorithms better than the MST whenever $\rho_k < \rho_2$ for some $k = O(1)$, and there exists an algorithm for computing optimal Steiner trees on k vertices. Computing $O(1)$-size optimal Steiner trees is easy for the rectilinear case: reduce the problem to the network problem and use exhaustive search or dynamic programming. The Euclidean case can be solved by a recursive algorithm of Melzak [Mel61] which replaces a pair of terminals adjacent to the same Steiner point by a single new terminal. We now consider the question of bounding ρ_k for various metrics and choices of k. The following bound was proved independently by Zelikovsky [Zel92] and Berman and Ramaiyer [BR94]; combined with Theorem 8.8 it gives an approximation ratio of $\frac{11}{8}$ for the rectilinear Steiner problem.

THEOREM 8.9 For the rectilinear metric, $\rho_3 \leq \frac{5}{4}$.

Proof. For any full component T of an optimal rectilinear Steiner tree $RST(S)$, we show how to find four 3-restricted Steiner trees T_1, T_2, T_3, and T_4, that each span the same terminals as T, with total length five times $|T|$. Thus, the shortest of the four trees proves the theorem.

Assume T has the canonical form shown on the far left in Figure 8.5 on page 306; the other case is similar. We create T_1 by first doubling each left-side rib that is longer than the ribs above and below it. We then replace the spine between two doubled ribs by a staircase between terminals as shown on the left in Figure 8.11. Tree T_2 is formed analogously using right-side ribs. Tree T_3 is created by first doubling each left-side rib not doubled in T_1 as shown on the right in Figure 8.11. This creates components with three terminals on the left and two on the right. To finish T_3, we add an edge paralleling the spine on the right and remove one edge of the spine. Tree T_4 is formed analogously using right-side ribs. ■

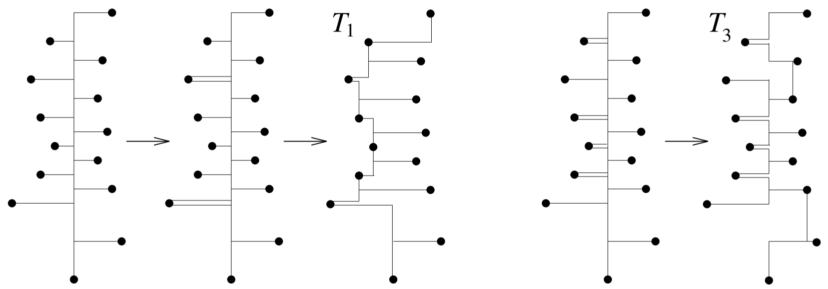


FIGURE 8.11

Transforming a rectilinear full component in canonical form into a 3-restricted tree.

Berman and Ramaiyer [BR94] generalized the bound to $\rho_k \leq 1 + \frac{1}{2k-2}$ by defining a collection of $2k-2$ k-restricted full trees with length at most $2k-1$ times the length of any full tree. The bound is tight for $k=3$, as shown by the corners of the unit ball, but Berman and Ramaiyer conjectured, and Borchers et al. [BDG95] recently proved, that the tight bound for $k \geq 4$ is $1 + \frac{1}{2k-1}$.

Du, Zhang, and Feng [DZF91] proved the following theorem for the Euclidean case, which gives an approximation better than $2/\sqrt{3}$ for the depressingly large value $k = 128$. They conjecture, however, that $\rho_3 = (\sqrt{2}+\sqrt{6})/(1+\sqrt{2}+\sqrt{3}) \approx 1.073$, sufficient to give an improved approximation ratio of 1.114.

THEOREM 8.10 For the Euclidean metric, $\rho_k \leq (\lfloor \log_2 k \rfloor + 1) \big/ \lfloor \log_2 k \rfloor$.

Proof Sketch. By induction, we may assume that the optimal Steiner tree $ST(S)$ is full. We root $ST(S)$ in the middle of an edge to turn it into a rooted binary tree. Now from each internal vertex v (a Steiner point) define a path $p(v)$ of length $|p(v)|$ down to a leaf (a terminal), such that each pair of paths is edge-disjoint. (This is easy to do by induction.)

Let U_i be the set of internal vertices of $ST(S)$ with level number (number of edges up to the root) equal to i mod $\lfloor \log_2 k \rfloor$. Let the cost of U_i be $\sum_{v \in U_i} |p(v)|$. We remove all vertices of the cheapest U_i, breaking $ST(S)$ into subtrees. The lowest level subtrees have at most k terminals. We attach each leaf v of a higher level subtree to the other endpoint of $p(v)$ using a single edge. The length of the resulting k-restricted Steiner tree is at most $1 + 1/\lfloor \log_2 k \rfloor$ times $|ST(S)|$. ■

In any metric, Steiner points can be made degree-three by adding zero-length edges, and the proof above will hold, proving that any metric admits a Steiner tree approximation algorithm better than the minimum spanning tree. In fact, $(\lfloor \log_2 k \rfloor + 1) \big/ \lfloor \log_2 k \rfloor$ is the tight bound for ρ_k for some metrics [BD95].

OPEN PROBLEM 8.5 What are the values of ρ_k in the Euclidean metric?

8.3.2.3 A stack-based algorithm

Both ρ_k and the contribution of ρ_k to the approximation ratio decrease with increasing k, so Zelikovsky's algorithm does not give a sequence of better and better algorithms. Berman and Ramaiyer [BR94], however, modified Zelikovsky's algorithm to achieve this result. Their algorithm consists of two phases: evaluation and construction. Both phases keep their work in a tree $M(S)$, a minimum spanning tree for a graph with modified edge weights.

The evaluation phase considers possible full Steiner components, in order of increasing numbers of terminals from 3 to k. At a generic step, if a component T_j offers a gain in the current $M(S)$, it is pushed onto a stack and $M(S)$ is modified to reflect the use of T_j. Specifically, the edges e_i of a spanning tree R_j for the terminals of T_j are added one-by-one to $M(S)$. Edge e_i is given weight equal to the length of the longest edge e_i' in the cycle it forms when first added to $M(S)$, minus the gain $W(T_j \mid T_1, \ldots, T_{j-1})$. Subtracting off $W(T_j \mid T_1, \ldots, T_{j-1})$ prevents overlapping full components with smaller gain from making it onto the stack.

Figure 8.12 gives an example. The upper right figure shows the initial minimum spanning tree $M(S)$, the same as $MST(S)$. When tree T_1 is evaluated and found to offer positive gain, it is pushed onto the stack. Edge e_1 is added to $M(S)$ and e_1' is removed. The new weight of edge e_1 is $|e_1'| - W(T_1)$. Then edge e_2 is added and e_2' removed, and e_2 receives weight $|e_2'| - W(T_1)$.

The construction phase pops full components off the stack, transforming $M(S)$ into a Steiner tree as it goes. If, at the time T_j is popped, spanning tree R_j is intact in $M(S)$, then T_j replaces R_j in $M(S)$. Otherwise, whatever edges of R_j remain are replaced by MST edges; that is, edge e_i is replaced by the shortest edge linking the same two pieces of $M(S)$. When the stack is empty, $M(S)$ is the final Steiner tree.

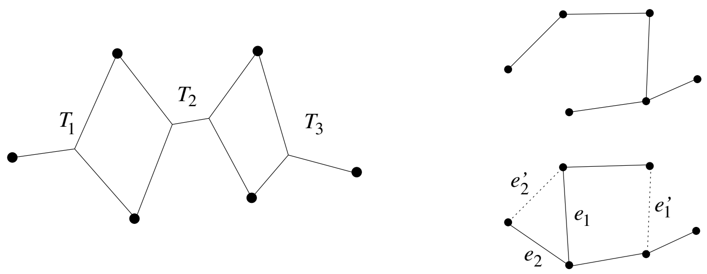


FIGURE 8.12

Berman and Ramaiyer choose T_1 and T_3 instead of the inferior, greedy choice of T_2.

For example, assume that $W(T_1)$ and $W(T_3)$ are both smaller than $W(T_2)$ in Figure 8.12, but their sum is larger. The evaluation phase will push both T_1 and T_3 onto the stack, but by the time it considers 4-terminal subtrees, $W(T_2 \mid T_1, T_3)$ will be negative and T_2 will not offer any gain. Thus, Berman and Ramaiyer's algorithm will choose T_1 and T_3, while Zelikovsky's greedy algorithm will choose only T_2. For the case $k = 3$, however, the two algorithms are identical.

THEOREM 8.11 Berman and Ramaiyer The stack-based algorithm has approximation ratio

$$\rho_2 - \frac{\rho_2 - \rho_3}{2} - \frac{\rho_3 - \rho_4}{3} - \dots - \frac{\rho_{k-1} - \rho_k}{k-1}.$$

Proof. Let us track the cost of tree $M(S)$ during the execution of the algorithm. It is not hard to see that during the evaluation phase, when a full tree T_j spanning i terminals is pushed onto the stack, $|M(S)|$ decreases by $(i-1)W(T_j \mid T_1, \dots, T_{j-1})$.

We now assert that in the construction phase, each time a full tree T_j spanning i terminals is popped off the stack, $|M(S)|$ increases by at most $(i-2)W(T_j \mid T_1, \dots, T_{j-1})$. If T_j makes it into $M(S)$, then $|M(S)|$ increases by the length of T_j and decreases by the weight of R_j. We set the edge weights for R_j to guarantee the assertion in this case. (In other words, the net gain is the expected $W(T_j \mid T_1, \dots, T_{j-1})$.) If T_j does not make it into $M(S)$, then at least one edge of the $i-1$ edges of R_j has already been removed by the addition of some other full tree to $M(S)$. In this case, each of the $i-2$ remaining edges of R_j is replaced by a shortest edge crossing a cut. When remaining edge e_l is replaced, $|M(S)|$ increases by at most $W(T_j \mid T_1, \dots, T_{j-1})$, since e_l has cost $|e_l'| - W(T_j \mid T_1, \dots, T_{j-1})$, where $|e_l'|$ is the longest edge of a cycle.

Let M_i be $|M(S)|$ after all trees that span i terminals have been pushed onto the stack in the evaluation phase, and set $M_2 = |MST(S)|$. Let N_i be $|M(S)|$ after all trees that span $i+1$ terminals have been popped off the stack in the construction phase. By the discussion above, the final cost is bounded by

$$\begin{aligned} N_2 &\le N_k + \sum_{i=3}^{k} (i-2) \sum_j \{ W(T_j \mid T_1, \ldots T_{j-1}) \mid T_j \text{ spans } i \text{ terminals} \} \\ &\le M_k + \sum_{i=3}^{k} \frac{i-2}{i-1} (M_{i-1} - M_i) \\ &\le M_2 - \sum_{i=3}^{k} \frac{(M_{i-1} - M_i)}{i-1}. \end{aligned}$$

Using an argument akin to that in the proof of Theorem 8.8, Berman and Ramaiyer show that for each i, $M_i \le |ST_i(S)|$, which finishes the proof. ■

For the rectilinear Steiner problem, Theorem 8.11 coupled with Berman and Ramaiyer's bounds on ρ_k gives a sequence of approximation algorithms with ratios converging to about 1.323. In more recent work, Berman et al. [BFK$^+$94] directly bound linear combinations of the ratios $M_i/|RST(S)|$ (with M_i as in the proof above) rather than using the ρ_k's, thus giving sharper estimates of the approximation ratios for $k = 3$ and $k = 4$. These ratios are now known to be at most $\frac{21}{16} = 1.3125$ (and no better than 1.3) and $\frac{61}{48} \approx 1.271$, respectively. Berman and Ramaiyer's algorithm can be run in $O(n^{3/2})$ time or approximated in $O(n \log^2 n)$ time for k equal to either 3 or 4, and $O(n^{\lfloor k/2 \rfloor + 3/2})$ for general k [BFK$^+$94, BR94]. Most recently, Karpinski and Zelikovsky [KZ94] added a preliminary phase to Berman-Ramaiyer, which—without increasing the running time—further reduces the approximation ratio to $\frac{19}{15} \approx 1.267$ for $k = 4$.

8.3.2.4 The relative greedy algorithm

Very recently, Zelikovsky [Zel94] modified his original greedy algorithm by changing the way full components are chosen. The result is a sequence of algorithms with approximation ratios converging to $1 + \ln \rho_2$. This *relative greedy* algorithm thus gives $1 + \ln 2 \approx 1.693$ for the network Steiner problem, asymptotically beating Berman-Ramaiyer, which converges to about 1.734 [BD95]. It gives $1 + \ln \frac{2}{\sqrt{3}} \approx 1.144$ for the Euclidean problem, winning for now, but losing to Berman-Ramaiyer and even to the original greedy algorithm if ρ_k values are as small as conjectured. The new algorithm already loses in the rectilinear case.

At each step, the relative greedy algorithm chooses the full Steiner tree T_j spanning at most k terminals that minimizes

$$|T_j| \Big/ (|MST(S; T_1, \ldots, T_{j-1})| - |MST(S; T_1, \ldots, T_j)| + |T_j|).$$

The denominator is the decrease, due to the addition of tree T_j, in the length of the minimum spanning forest needed to finish the tree on S. The original greedy criterion minimizes $|T_j|$ minus the denominator.

The approximation ratio bounds are implied by the following theorem:

THEOREM 8.12 Zelikovsky The relative greedy algorithm finds a tree of length at most $1 + \ln \frac{|MST(S)|}{|ST_k(S)|}$ times $|ST_k(S)|$.

Proof. Let $T_1, \ldots, T_\ell$ be the full Steiner trees chosen by the relative greedy algorithm, including 2-terminal trees (edges). Let $T_1^*, \ldots, T_m^*$ be the full components of an optimal k-restricted Steiner tree $ST_k(S)$. Let $F_j = |MST(S; T_1, \ldots, T_j)| - \sum_{i=1}^{j} |T_i|$, the length of a minimum spanning forest connecting $T_1, \ldots, T_j$ along with all singleton terminals. Let $F_0 = |MST(S)|$. Because the relative greedy algorithm makes the minimizing choice for T_1,

$$\frac{|T_1|}{|MST(S)| - F_1} \;\leq\; \min_{1 \leq j \leq m} \frac{|T_j^*|}{|MST(S)| - |MST(S; T_j^*)| + |T_j^*|} \;\leq\; \frac{|ST_k(S)|}{|MST(S)|}.$$

We may apply the same argument after choosing T_1 so that

$$\frac{|T_2|}{F_1 - F_2} \;\leq\; \frac{|ST_k(S; T_1)| - |T_1|}{|MST(S; T_1)| - |T_1|} \;\leq\; \frac{|ST_k(S)|}{F_1},$$

where $ST_k(S; T_1)$ and $MST(S; T_1)$ denote the optimal k-restricted Steiner tree and the minimum spanning tree constrained to choose T_1. (Imagine contracting T_1 to a point.) Inductively we obtain for each $i \geq 1$, $|T_i|/(F_{i-1} - F_i) \leq |ST_k(S)|/F_{i-1}$, or equivalently $F_i \leq F_{i-1}(1 - |T_i|/|ST_k(S)|)$. Unraveling these inequalities,

$$F_r \;\leq\; F_0 \prod_{i=1}^{r} (1 - |T_i|/|ST_k(S)|).$$

Taking the natural logarithm on both sides and using the fact that $\ln(1+x) \leq x$,

$$\frac{\sum_{i=1}^{r} |T_i|}{|ST_k(S)|} \;\leq\; \ln \frac{F_0}{F_r}.$$

Since F_0/F_r grows arbitrarily large, we can choose r such that $F_r > |ST_k(S)| \geq F_{r+1}$. We split $|T_{r+1}|$ proportionately by the position of $|ST_k(S)|$ in the interval $[F_{r+1}, F_r]$. We combine the first portion with F_{r+1} to bring this cost up to exactly $|ST_k(S)|$, and combine the second portion with $|T_r|$. We then split $F_r - F_{r+1}$ into the same proportions, and subtract the second portion from F_r so that the last inequality above still holds when we "pretend" that $|ST_k(S)| = F_{r+1}$. We now finish the proof with the sequence of inequalities

$$\frac{\sum_{i=1}^{\ell} |T_i|}{|ST_k(S)|} \leq \frac{F_{r+1}}{|ST_k(S)|} + \frac{\sum_{i=1}^{r+1} |T_i|}{|ST_k(S)|} \;\leq\; 1 + \ln \frac{F_0}{F_{r+1}} \;=\; 1 + \ln \frac{|MST(S)|}{|ST_k(S)|}.$$

■

8.4 MINIMUM WEIGHT TRIANGULATION

We now consider approximations to a third geometric network design problem, called *minimum weight triangulation*. The input to this problem is a set of points S in the plane, and the desired output is a *triangulation* that minimizes the total Euclidean edge length. A triangulation of S is a maximal planar straight-line graph with vertex set S, or equivalently a partition of the convex hull of S into triangles that intersect only at common edges and vertices. Unlike the previous two problems, 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 [GJ79]. A generalized problem with non-Euclidean distances is *NP*-complete [Llo77].

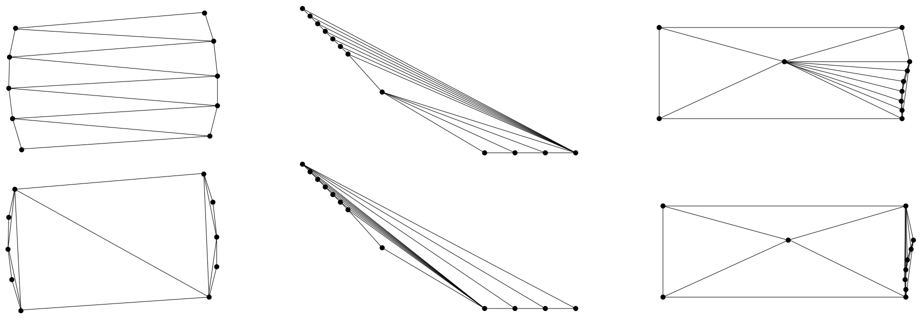
We consider two versions of minimum weight triangulation. In the more commonly studied version, the vertex set of the triangulation is exactly the set of input points S. In *minimum weight Steiner triangulation*, additional vertices lying anywhere in the plane may be added, and the output is a triangulation of S and the Steiner points.

8.4.1 TRIANGULATION WITHOUT STEINER POINTS

The first approach to minimum weight triangulation considers other standard triangulations as approximations. It was at once believed that the Delaunay triangulation of S actually gives the minimum weight triangulation $MWT(S)$; however, it has since been shown that the Delaunay triangulation can have length as much as $\Omega(n)$ times the optimum [Kir80, MZ79]. Flipping diagonals of quadrilaterals to locally minimize lengths, in analogy with a popular Delaunay triangulation algorithm, does not improve this ratio. See Figure 8.13(a).

Greedy triangulation [Gil79, LL92] is an iterative algorithm, which at each step chooses the shortest edge that does not cross any previous edges. Figure 8.13(b) gives an $\Omega(\sqrt{n})$ lower bound on approximation ratio. The example point set contains an $O(1)$-length slanted line of $\Omega(n)$ vertices and an $\Omega(\sqrt{n})$-length horizontal row of about $\sqrt{n}$ vertices. The greedy triangulation uses a fan of $\Omega(n)$ edges each about $\sqrt{n}$ times longer than the edges in the optimal triangulation's fan. Very recently, Levcopoulos and Krznaric [LK96] established a matching upper bound of $O(\sqrt{n})$. More importantly, they devised a constant-factor approximation algorithm based on greedy triangulation.

The second approach to minimum weight triangulation builds an approximation algorithm from scratch, using minimum weight triangulation of a simple polygon—



(a) (b) (c)

FIGURE 8.13

Bad examples for (a) Delaunay triangulation (and edge flipping), (b) greedy triangulation, (c) Lingas's algorithm. Minimum weight triangulations are below.

solvable in polynomial time by dynamic programming—as a subroutine. The earliest algorithm of this form, due to Lingas [Lin83], adds minimum spanning tree edges until the convex hull of S breaks into simple polygons, and then computes the optimal triangulation within each polygon. In Figure 8.13(c), this algorithm goes wrong by adding the MST edge from the central point to the topmost point of an arc of $\Omega(n)$ points, blocking a much closer point and giving a ratio of $\Omega(n)$. Heath and Pemmaraju [HP94] substitute a subtree of the greedy triangulation for the minimum spanning tree; an example combining features from Figures 8.13(b) and (c) gives an $\Omega(\sqrt{n})$ lower bound for their heuristic. Plaisted and Hong [PH87, Smi89] use a more involved procedure to form the polygons, and achieve an $O(\log n)$ approximation to $MWT(S)$. Finally, there has been recent progress [DM95] on finding large sets of edges guaranteed to be in the minimum weight triangulation. If a connected subgraph of $MWT(S)$ could be found, then the rest could be filled in by dynamic programming, leading to a polynomial-time exact algorithm for the problem.

We now say a little more about the two best approximation algorithms: the polygon-forming method of Plaisted and Hong and the modified greedy algorithm due to Levcopoulos and Krznaric. These two algorithms share a basic idea. Rather than finding a triangulation directly, they try to find a minimum weight partition into convex polygons.

We first sketch Plaisted and Hong's work. Let v be an interior vertex of a planar straight-line graph with convex faces. 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 convex partition of S by piecing together 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 $|MWT(S)|$. 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 $12|MWT(S)|$.

The convex faces can be triangulated optimally using dynamic programming, but a simple suboptimal algorithm has the advantage that it bounds the overall length as a multiple of the length of the convex partition. The *ring heuristic*, shown in Figure 8.14, connects every other vertex of a convex polygon P to form a polygon P' with $\lfloor n/2 \rfloor$ vertices and a smaller perimeter, and then triangulates P' recursively. Hence, the final Plaisted-Hong triangulation has total length $O(\log n)$ times optimal. Olariu et al. [OTZ88] showed that this bound is tight.

We now sketch Levcopoulos and Krznaric's work. They start by proving that the greedy triangulation achieves approximation ratio $O(\sqrt{n})$. To prove this, they define a convex partition that uses only edges of the greedy triangulation of S. Unlike Plaisted and Hong's collection of stars, the initial result is planar as it is a subgraph of a planar graph. Levcopoulos and Krznaric prove that this convex partition approximates the minimum weight partition into convex polygons within a factor of $O(\sqrt{n})$. Next they prove that any convex partition $CP(S)$ formed by edges of the greedy triangulation can be triangulated by adding diagonals of total length at most $O(1)$ times $|MWT(S)| + |CP(S)|$. The original greedy triangulation includes a greedy triangulation of each cell of $CP(S)$, and

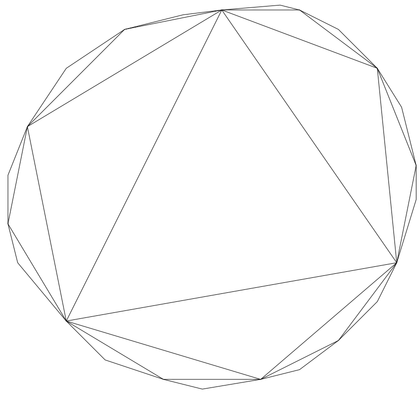


FIGURE 8.14

Ring heuristic triangulation of a convex polygon.

the greedy triangulation approximates the minimum weight triangulation of a convex polygon within a constant factor [LL87], so the greedy triangulation must be an $O(\sqrt{n})$ approximation to $MWT(S)$.

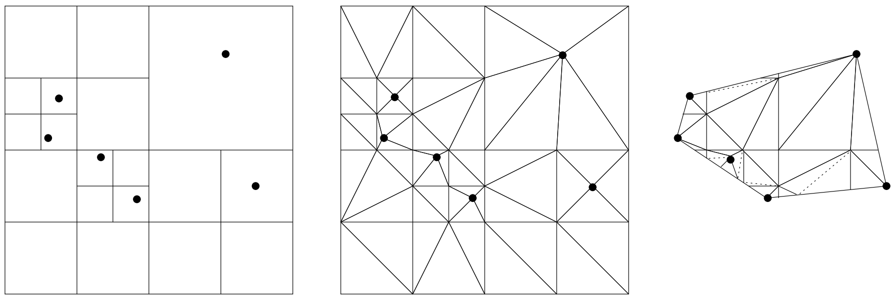
To improve this ratio, Levcopoulos and Krznaric modify the greedy triangulation. The modified algorithm rejects certain greedy edges, replacing them with slightly longer alternative edges, in order to avoid fans of the form shown in Figure 8.13(b). The result is a constant-factor approximation algorithm for the problem of finding a minimum weight convex partition of S, which in turn gives a constant-factor approximation to the minimum weight triangulation. Levcopoulos and Krznaric show that these approximations can be constructed in time $O(n \log n)$, or even $O(n)$ if the Delaunay triangulation is already known.

8.4.2 STEINER TRIANGULATION

We next discuss approximation algorithms for minimum weight Steiner triangulation. For this problem, the complexity of exact algorithms is unclear. It is not even known whether there always exists a minimum weight Steiner triangulation; perhaps there are point sets S for which larger and larger sets of Steiner points give triangulations with smaller and smaller total lengths. In any case, we can define $|MWST(S)|$ as the infimum of the lengths of Steiner triangulations for S, so that the approximation problem remains well defined. It is known that $\Omega(n)$ Steiner points may be needed to achieve a constant-factor approximation to $|MWST(S)|$, and the algorithm described in this section shows that $O(n)$ suffices.

We now explain a quadtree-based approximation algorithm due to Eppstein [Epp94], which gives a constant-factor approximation, improving upon a previous log-factor approximation due to Clarkson [Cla91]. A *quadtree* is a partition of a region of the plane into axis-aligned squares. One square, the *root*, covers the region that is to be partitioned. A square may be recursively divided into four smaller squares.

The basic quadtree triangulation algorithm builds a quadtree covering the set of input points S, by recursively subdividing squares until all points of S lie in different squares and no two adjacent leaf squares differ in size by more than a factor of two. This planar straight-line graph can be triangulated by connecting each input point to the quadtree vertices visible to it, and by adding diagonals to quadtree squares not containing input points. However, due to edges extending outside the convex hull of S, the resulting triangulation may have length $\Omega(\log n)$ times $|MWST(S)|$. To reduce this factor, we clip the triangulation with the convex hull of S, and then retriangulate the cut faces. The steps of the algorithm are depicted in Figure 8.15.



(a) (b) (c)

FIGURE 8.15

Steps in quadtree triangulation: (a) construct a "balanced" quadtree with each input point in a separate square; (b) add diagonals to form a triangulation; (c) remove edges outside the convex hull and add diagonals (dotted lines) to sliced triangles.

THEOREM 8.13 Eppstein The quadtree algorithm gives a Steiner triangulation with total length at most $316|MWST(S)|$.

Proof. We shall not prove the specific constant of 316, but rather just sketch a proof of some constant. Let $QT(S)$ denote the quadtree triangulation and $MWT(S)$ denote the (non-Steiner) minimum weight triangulation of S, with lengths $|QT(S)|$ and $|MWT(S)|$. Below we shall show that $|QT(S)|$ is at most some constant times $|MWT(S)|$. Now imagine adding some Steiner points T to S. Adding T only increases the total length of the (untriangulated) quadtree, so $|QT(S)|$ is at most a constant times $|QT(S \cup T)|$. Since T may be the Steiner points of an optimal or near-optimal Steiner triangulation, we can conclude that $QT(S)$ also approximates $|MWST(S)|$.

We now aim to prove that $|QT(S)|$ has length at most a constant times $|MWT(S)|$. By subdividing squares containing points of S a few extra times, most of the length of $QT(S)$ will be contained in empty squares; that is, squares not containing points of S. We now show how to bound the length in empty squares by charging it to edges of $MWT(S)$.

Let B be an empty quadtree square that lies interior to the convex hull of S. Let R be a concentric square somewhat bigger than 5 times the side length of B. In the quadtree construction, each square has a point of S no farther away than twice its side length, so R cannot be empty. Our aim is to charge the length of $QT(S)$ in B to a "long edge" (length at least a constant times B's side length) of $MWT(S)$ with an endpoint in R.

Some triangle Δ of $MWT(S)$ contains the center c of B. The vertices of Δ lie outside B, and so Δ must have at least two "long" sides. If one of the vertices s_i of Δ is inside R, we charge the length of $QT(S)$ in B to a long edge of Δ incident to s_i. If all three vertices of Δ are far from B, then we choose some other point $s_j \in S$ somewhat interior to R. One of the edges of Δ must cut across R and separate B from s_j. We let Δ' be the triangle on the other side of this edge and apply the same argument to Δ'. By walking from triangle to triangle in this way, as shown in Figure 8.16, we eventually find a long edge e in $MWT(S)$ with an endpoint in region R. This is the edge to which we charge the length of $QT(S)$ inside B.

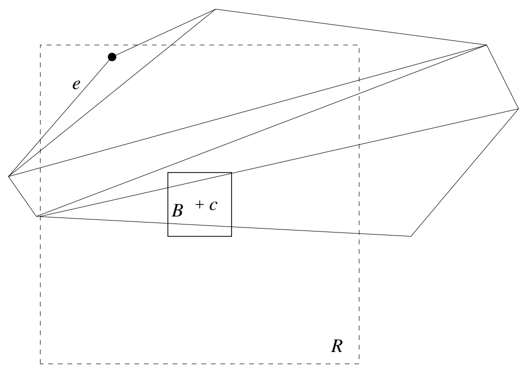


FIGURE 8.16

Walking from triangle to triangle leads to a long edge with a nearby endpoint.

In this scheme, each edge e in $MWT(S)$ is charged by $O(1)$ quadtree squares of a given size, so that e's total charge adds in a geometric series proportional to its length. Thus, the total quadtree edge length is proportional to $|MWT(S)|$.

The remaining part of the proof deals with those quadtree squares not interior to the convex hull of S. Some such squares have a neighbor entirely contained in the convex hull and can be treated as above. The remaining cases have total length proportional to the convex hull, and hence to $|MWT(S)|$. ■

The quadtree triangulation can be constructed in time $O(n \log n + k)$, where k denotes the output complexity [BEG94, BET93]. As we have explained the algorithm, however, k need not be bounded by a polynomial in n. Eppstein corrected this flaw by giving an algorithm in which "clusters" of nearby points are triangulated recursively and then treated as a unit in the overall triangulation. This algorithm uses $O(n)$ Steiner points and has $O(n \log n)$ running time. (A straightforward way of achieving the same running time with $O(n \log n)$ output complexity is to stop subdividing quadtree squares when the size reaches $1/n$ times that of the root, and then use any non-Steiner triangulation algorithm within each tiny square.)

We briefly comment on the special case of minimum weight Steiner triangulation for point sets in convex position. For this special case, there are a few results beyond the constant-factor approximation given by the quadtree algorithm. For example, the ring heuristic and the greedy triangulation give approximations with ratios of $\Theta(\log n)$. Eppstein used a similar quadtree triangulation technique as above to show that the minimum weight triangulation, constrained to use Steiner points only on the boundary of the convex hull, has weight $O(1)$ times $|MWST(S)|$.

OPEN PROBLEM 8.6 Does every point set have a Steiner triangulation that achieves the minimum weight? Does every point set in convex position have a minimum weight Steiner triangulation that uses no interior Steiner points?

CLUSTERING 8.5

We now turn away from famous individual problems to a large, loosely defined area. A *clustering problem* is given by a set of points S in the plane or some other metric space, and seeks the "best" partition of S into subsets, or *clusters*. We consider two different styles of problems: *k-clustering*, in which we divide S into k clusters, and *k-point clustering*, in which we seek the single best cluster containing k points.

8.5.1 MINMAX K-CLUSTERING

To measure the quality of a partition of S, some criterion, such as diameter or variance, is applied to each cluster individually. Then individual measures are combined into an overall criterion using a function such as the sum or the maximum.

For the case of fixed k, there are polynomial-time algorithms for minimizing various combinations of cluster diameters [CRW91, HS91b] and for minimizing the sum of variances [BH89, IKI94], but other individual cluster criteria, such as the sum of pairwise distances, give open problems. For example, *Euclidean max cut*, equivalent to asking for the two clusters that minimize the sum of all pairwise intracluster distances, is neither known to be in P nor known to be *NP*-hard. Letting k vary, however, almost always gives an *NP*-complete problem, and very few nontrivial approximation results are known. In this section, we describe two k-clustering problems—minmax radius and minmax diameter—that stand out as especially well understood.

Minmax radius clustering, also known as "central clustering" and the "Euclidean k-center problem," seeks a partition $S = S_1 \cup \cdots \cup S_k$ that minimizes $\max_{1 \le i \le k} radius(S_i)$, where $radius(S_i)$ is the radius of the smallest disk that covers all points of S_i. The k-center problem is discussed along with other minmax (or bottleneck) problems in

Section 9.4. *Minmax diameter* clustering, also known as "pairwise clustering," seeks to minimize $\max_i$ *diameter*(S_i), where

$$diameter(S_i) = \max\{|s_j s_l| \mid s_j, s_l \in S_i\}.$$

Gonzalez [Gon85] proposed the following algorithm, called *farthest-point clustering*. Let an arbitrary point s_{i_1} be the representative for the first cluster. Pick the point s_{i_2} farthest from s_{i_1} to represent the second cluster. Pick s_{i_3} to maximize the distance to the nearer of s_{i_1} and s_{i_2}. Continue this process for k steps, at each step picking s_{i_j} to maximize $\min\{|s_{i_1}s_{i_j}|, |s_{i_2}s_{i_j}|, \ldots, |s_{i_{j-1}}s_{i_j}|\}$. After all representatives are chosen, we can define the partition of S: cluster S_j consists of all points closer to s_{i_j} than to any other representative. The following theorem is due to Gonzalez [Gon85]:

THEOREM 8.14 For either radius or diameter clustering, farthest-point clustering computes a partition with maximum cluster size at most twice optimal.

Proof. Let $s_{i_{k+1}}$ be an input point that maximizes

$$\delta = \min\{|s_{i_1}s_{i_{k+1}}|, \ldots, |s_{i_k}s_{i_{k+1}}|\},$$

in other words, the point that would be chosen if we picked one more representative. All pairwise distances among $s_{i_1}, s_{i_2}, \ldots, s_{i_{k+1}}$ are at least δ. In any k-clustering, two of these points must be in the same cluster, hence δ and $\delta/2$ are respectively lower bounds for the diameter and radius of the worst cluster. Farthest-point clustering places each s_i into a cluster with representative s_{i_j} such that $|s_i s_{i_j}| \le \delta$. Thus each cluster has radius at most δ, and by the triangle inequality, diameter at most 2δ. ■

This proof uses no geometry beyond the triangle inequality, so Theorem 8.14 holds for any metric space (see Exercise 9:exc:k-center). The obvious implementation of farthest-point clustering has running time $O(nk)$. Feder and Greene [FG88] give a two-phase algorithm with optimal running time $O(n \log k)$. The first phase of their algorithm clusters points into rectangular boxes using Vaidya's [Vai89] "box decomposition"—a sort of quadtree in which cubes are shrunk to bounding boxes before splitting. The second phase resembles farthest-point clustering on a sparse graph that has a vertex for each box.

The approximation ratio of Theorem 8.14 is, depending upon the metric, optimal or nearly optimal—"best possible" in the terminology of Section 9.4. This non-approximability result, quite rare for a geometric problem, is proved with a problem reduction that creates a "nasty gap," with "no" instances mapped far away from "yes" instances. It is interesting to note that this reduction, for metric spaces, starts from a planar-graph problem that admits a polynomial-time approximation scheme [Bak94]. (Planar graphs [Bak94] and certain geometric intersection graphs [HMR$^+$94] typically admit such approximation schemes for problems in which the solution is simply a set of vertices; see Section 9.3.3.4. Planar graph network-design problems, however, are currently as open as their geometric counterparts.) The specific constructions below are due to Feder and Greene [FG88]; there are also earlier, weaker results [Gon85, MS84].

THEOREM 8.15 It is *NP*-hard to approximate the Euclidean minmax radius k-clustering (or Euclidean k-center) with an approximation ratio smaller than 1.822, or the

Euclidean minmax diameter k-clustering with ratio smaller than 1.969.

Proof. Both problem reductions start from vertex cover for 3-regular planar graphs (see [GJ77]). As shown in Figure 8.17(a), any 3-regular planar graph G can be embedded in the plane so that each edge e becomes a path p_e with some odd number of edges, at least 3, which we shall denote by $|p_e|$. The paths meet at 120° angles at vertices, and remain well separated away from vertices. The midpoints S of the unit-length edges form an instance of the k-clustering problem.

It is not hard to see that S has a k-clustering with maximum cluster radius $\frac{1}{2}$ if and only if $k - \sum_e(|p_e| - 2)$ vertices can cover all edges in G. A cluster of radius $\alpha > \frac{1}{2}$ helps reduce this number only if a disk of radius α can cover more than 3 points of S near an original vertex of G. If each original vertex of G has local neighborhood as in the top illustration of Figure 8.17(b), then it takes a disk of diameter $d_1 = \sqrt{13}/2 \approx 1.80$, radius $d_1/2$, to cover 4 points. The bottom illustration shows an improved construction in which a disk must have diameter $d_1' = (1+\sqrt{7})/2 \approx 1.822$ to cover more than 3 points.

Figure 8.17(c) shows analogous constructions for minmax diameter clustering. In the basic construction (top illustration), a cluster must have diameter $d_2 = (1+\sqrt{3})/\sqrt{2} \approx 1.93$ to cover more than 3 points. In the improved construction, the critical distance is $d_2' = 2\cos(10°) \approx 1.969$. ■

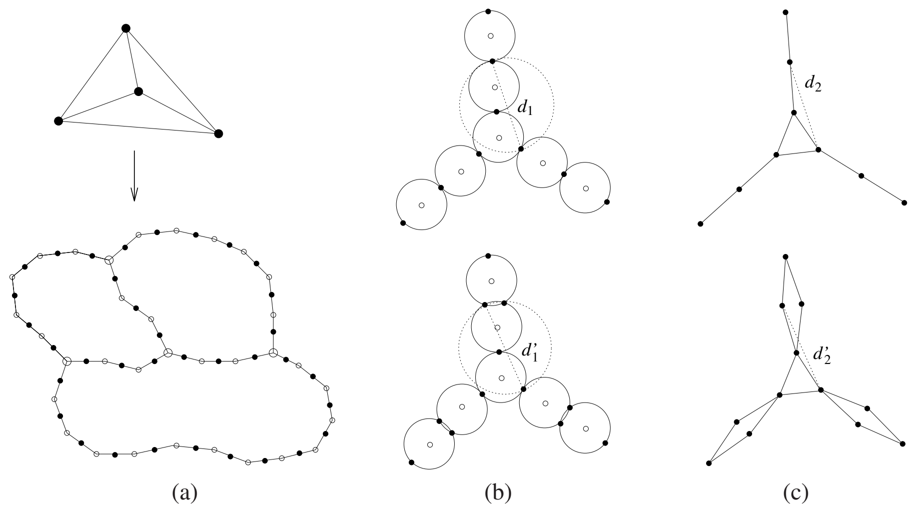


FIGURE 8.17

(a) Embedding the edges of a 3-regular planar graph as paths. Endpoints of embedded edge are circles; midpoints are dots. (b) Detail of a node for minmax radius clustering. (c) Detail of a node for minmax diameter clustering.

For the rectilinear metric, Feder and Greene give a similar, simpler construction that shows that it is *NP*-hard to approximate either minmax radius or minmax diameter with a ratio smaller than 2.

OPEN PROBLEM 8.7 Is Euclidean max cut solvable in polynomial time? How well can it be approximated? Do other natural k-clustering problems—for example, minimizing the sum of the distances from points to their cluster centers—have good approximation algorithms?

8.5.2 K-MINIMUM SPANNING TREE

We now consider a different sort of clustering problem, in which we seek the best single cluster containing k points of S, rather than the best partition of S into k groups. If the quality measure is radius or diameter (or area or perimeter of convex hull), this problem is polynomially solvable [AIKS91, DLSS93, EE94]. (We note in passing that diameter in $\mathbb{R}^d$ for $d > 2$ is still open.) Another reasonable quality measure, however, leads to an *NP*-hard problem.

The *Euclidean k-minimum spanning tree* problem measures the cost of interconnecting the cluster: given n points S in the plane, find k points with the shortest minimum spanning tree. Up to constant factors in the approximation ratio, it is equivalent to ask for the shortest cycle (the k-TSP problem) or shortest Steiner tree. Also inessential is the choice of metric. In fact, there has been a flurry of recent work on non-geometric k-MST problems [AABV94, CK94, RSM$^+$94, ZL93]. In this section, however, we shall focus on the Euclidean k-MST formulation.

The k-MST problem was introduced by Fischetti et al. [FHJM94] and Zelikovsky and Lozevanu [ZL93], and proved *NP*-complete by those authors and also by Ravi et al. [RSM$^+$94]. Ravi et al. devised an approximation algorithm with ratio $O(k^{1/4})$ for the Euclidean k-MST. The ratio was improved to $O(\log k)$ by Garg and Hochbaum [GH94], $O(\log k/\log\log n)$ by Eppstein [Epp95], $O(1)$ by Blum et al. [BCV95], and $2\sqrt{2}$ by Mitchell [Mit96].

Mitchell builds on the approach of Blum et al., who built on the approach of Hochbaum and Garg. He considers only trees of a special type, and shows that the shortest tree of this type can be found using dynamic programming. A tree falls into the special type if it is rectilinear, meaning that all edges are either horizontal or vertical, and *guillotine*, meaning that there exists a horizontal or vertical cutting line that intersects the tree in a single line segment, and that this property holds recursively within each cut half. It is obvious that an optimal k-MST can be made rectilinear with edge length increasing by at most a factor of $\sqrt{2}$. Lemma 8.6 below shows that the guillotine condition adds at most a factor of 2. Notice that the length of a guillotine tree is the total length of its line segments, not the total length of its cutting lines as in a binary space partition (see Lemma 8.1). The proof of Lemma 8.6 charges segment length to parallel, visible edges of R.

LEMMA 8.6 Mitchell Any connected rectilinear graph R is contained in a guillotine subdivision of total length at most twice R.

A dynamic programming algorithm can now compute an optimal guillotine tree spanning k points of S. We may assume that each line segment of this tree is a segment (union of colinear edges) of a grid graph. The grid graph is formed passing a horizontal and vertical line through each point of S and then adding a horizontal line, say, halfway between each existing horizontal and a vertical line halfway between each vertical. The most straightforward (but not the fastest) dynamic program tries each possible choice of first cutting line and segment, and then recurses within each half. Within the halves, the algorithm considers shortest guillotine trees spanning k_1 and k_2 points of S, for each choice of k_1 and k_2 summing to $k-l$, where l is the number of points of S along the first segment. Each of the smaller guillotine trees is required to connect to the first segment. As we proceed deeper in the recursion, the intersection of the optimal guillotine tree with the bounding box of each subproblem remains connected, so that the number of possible subtrees remains polynomial in n and k. The overall result is the following theorem:

THEOREM 8.16 Mitchell There is a polynomial-time, $2\sqrt{2}$-factor approximation algorithm for the Euclidean k-MST.

8.6 SEPARATION PROBLEMS

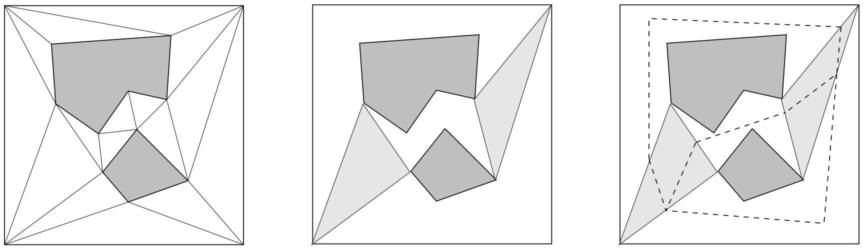
We now switch from grouping objects together to cutting them apart. We consider three basic problems: (1) separating k disjoint polygons in the plane by piecewise-linear "fences" with a minimum total number of edges, (2) separating two nested convex polyhedra in $\mathbb{R}^3$ by a polyhedron with a minimum number of facets, and (3) separating two point sets in $\mathbb{R}^3$ by a piecewise-linear function with a minimum number of pieces.

The three problems above depart from the problems of previous sections in two ways. In each case, the input is more complicated than a planar point set, and solutions are evaluated not by a continuous measure such as length, but by minimum combinatorial complexity. Despite these changes, some of the issues important in previous sections carry over to this section. For example, the solution to problem (3) uses dynamic programming methods similar to those for red-blue separation, TSP with neighborhoods, and the k-MST problem. And even though its input consists of two polytopes, problem (2) has a close relationship to a non-geometric problem—the set cover problem. As before, our aim will be to exploit geometry to improve approximation ratios.

8.6.1 POLYGON SEPARATION

The problem of separating two simple polygons by a polygon with a minimum number of sides can be solved in polynomial time [ABO$^+$89]. However, the problem of simultaneously separating a collection of k convex polygons by a minimum-complexity planar straight-line graph is *NP*-complete. Edelsbrunner et al. [ERS90] provided an approximation algorithm for this problem, which Mitchell and Suri [MS92] generalized to separation of arbitrary simple polygons. We describe the generalized algorithm, changing some minor details.

For simplicity, assume that one of the k polygons surrounds the remaining input. The algorithm first forms a triangulation of the space between the polygons, as shown in Figure 8.18(a). The algorithm then uses the triangulation to divide the space into



(a) (b) (c)

FIGURE 8.18

(a) Triangulation. (b) Moats (unshaded). (c) Separating graph (dashed).

regions called *moats*. This division is accomplished by forming the graph G that is the planar dual of the triangulation. Degree-one vertices are removed from G, and paths of degree-two vertices are contracted into single curves, resulting in a new planar graph G'. Graph G' is a planar graph with all vertices of degree 3 and with one input polygon per face, so it has $2k-4$ vertices and $3k-6$ edges. The moats are the connected components remaining when the triangles corresponding to vertices of G' are removed. See Figure 8.18(b). Each moat is a polygon bounded on two edges by removed triangles, and on the remaining sides by two of the input polygons. Call the two triangle edges of a moat its *terminals*.

Mitchell and Suri connect the two terminals of each moat by a *minimum link path*, a polygonal chain with a minimum number of edges. These paths can be computed in linear time [Sur90]. Within each of the $2k-4$ removed triangles, three link paths are connected with two more edges, as shown in Figure 8.18(c).

THEOREM 8.17 Mitchell and Suri The algorithm above gives a separating graph with at most five times as many edges as optimal.

Proof. Any separating graph must have at least k faces, one for each polygon, and must include a path within each moat in order to separate the polygons bounding the moat. Since the algorithm uses the minimum number of edges within each moat, the only unnecessary vertices are the $6k-12$ vertices within the $2k-4$ removed triangles. Thus, if the optimal graph has v vertices and $v+k-2$ edges, the algorithm above uses at most $v+6k-12$ vertices and $v+7k-14$ edges. The approximation ratio of this algorithm is $(v+7k-14)/(v+k-2)$, which is at most 5, since any planar graph with k faces has at least $3k/2$ edges. ∎

8.6.2 POLYHEDRON SEPARATION

Let P and R denote (closed) convex polytopes in $\mathbb{R}^3$ with $P \subseteq R$. We seek a polytope Q, $P \subseteq Q \subseteq R$, with a minimum number of faces. (Since the number of vertices, edges, and faces are linearly related, the precise measure of complexity changes only constant factors; however, the number of faces turns out to be most convenient.) A special case of this *polyhedron separation* problem is *polyhedron approximation*: given a polytope R and a distance ϵ, find a minimum-complexity Q inside R that has a boundary within distance ϵ of the boundary of R. The approximation problem can be reduced to the separation problem by letting P be the polytope formed by moving each face of R inwards by ϵ.

The polyhedron separation problem is due to Silio [Sil79] and Klee [ABO$^+$89]. Das and Joseph [DJ90] proved it *NP*-complete by a reduction from planar 3-SAT. (Whether the special case of polyhedron approximation is *NP*-complete seems to be unknown.) Mitchell and Suri [MS92] gave an $O(\log n)$-approximation algorithm (which works in general dimension). In this section we describe a recent improvement due to Brönnimann and Goodrich [BG94] that achieves a constant-factor approximation in $\mathbb{R}^3$.

Both algorithms reduce the separation problem to the *hitting set* problem [GJ79]: given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of sets of "points" (elements of a ground set), one seeks a minimum-cardinality set of points S such that each S_i contains at least one point of S. (This problem is equivalent to the *set cover* problem, in which one seeks a minimum-cardinality subset $\mathcal{S}' \subseteq \mathcal{S}$ covering all elements in the union $\bigcup_{1\le i\le n} S_i$. To transform a hitting set problem to a set cover problem, replace each point x by the set $\{i \mid x \in S_i\}$.) In the polyhedron separation problem, the "points" will be the planes containing faces of the inner polyhedron P.

LEMMA 8.7 There is a separating polyhedron Q with each face coplanar to a face of P that approximates the optimal number of faces within a factor of three.

Proof. Let polyhedra be represented as intersections of halfspaces, one per face. For each halfspace h of the optimal separating polyhedron Q', we can find three halfspaces of P such that the intersection of the three halfspaces is contained in h (Carathéodory's Theorem). The intersection of all such triples is a polyhedron Q with at most three times as many faces as Q'. Polyhedron Q is contained in the outer polyhedron R since it is contained in Q', and contains the inner polyhedron P. ■

We now define the hitting set problem. The points correspond to the planes containing faces of the inner polyhedron P. Let $\mathcal{A}$ denote the arrangement formed by these planes. The sets S_i correspond to the cells of $\mathcal{A}$ that intersect the boundary of the outer polyhedron R. The set for such a cell contains the planes of $\mathcal{A}$ that separate the cell from P. If P has n faces, then the set system $\mathcal{S}$ has n points and $O(n^2)$ sets. Any hitting set for $\mathcal{S}$ induces a separating polyhedron Q of the same complexity. Conversely, Lemma 8.7 shows that the minimum hitting set of $\mathcal{S}$ gives a 3-approximation to the minimum separating polyhedron.

The greedy algorithm for hitting set [Joh74], which repeatedly chooses the point that covers the most uncovered sets, has approximation ratio $1 + \ln n$. To achieve a smaller approximation ratio, we must take advantage of the geometric origin of the set system.

The *Vapnik-Chervonenkis dimension* of a set system is the maximum cardinality of a set of points *shattered* by the system. A set of points A is shattered by the set system if for each of its $2^{|A|}$ subsets A' there is a set S_i in the system with $A' = S_i \cap A$. The fact that our set system $\mathcal{S}$ is defined by an arrangement of planes implies that it has bounded VC-dimension: a set of m cells of $\mathcal{A}$ (points of $\mathcal{S}$) is cut into only $O(m^3)$ subsets by planes of $\mathcal{A}$, which is fewer than 2^m for sufficiently large m.

Now assume that the points of set system $\mathcal{S}$ have weights. An ϵ*-net* for $\mathcal{S}$ is a set of points N such that for any sufficiently heavy set S_i in $\mathcal{S}$, $N \cap S_i \neq \emptyset$. The parameter ϵ specifies what is meant by "sufficiently heavy": S_i is sufficiently heavy if the total weight of its points is at least ϵ times the total weight of all points of $\mathcal{S}$. Using random sampling, Blumer et al. [BEHW89] proved the following important result: any set system of VC-dimension d has an ϵ-net of cardinality $O(\frac{d}{\epsilon}\log\frac{1}{\epsilon})$. Notice that this bound is independent of n, the number of points in the set system. More recently, Matoušek and others [Mat92, MSW90] have given deterministic algorithms for finding ϵ-nets. These deterministic algorithms find frequent use in computational geometry as replacements for random sampling techniques. Indeed, Brönniman and Goodrich's algorithm can be seen as an application of ϵ-nets to derandomize an algorithm of Clarkson [Cla93], only in this case the derandomized algorithm ends up being better than the randomized one—an $O(1)$- rather than an $O(\log n)$-approximation—due to a better bound on the size of deterministic ϵ-nets for half-spaces in $\mathbb{R}^3$. Any set system defined by a collection of planes in $\mathbb{R}^3$ has an ϵ-net of cardinality $O(\frac{1}{\epsilon})$ that can be computed deterministically in polynomial time [MSW90, MS92].

We now describe how Brönniman and Goodrich use ϵ-nets to approximately solve the hitting set problem defined above. We first assume that the optimal cardinality c of a hitting set is known; we shall soon remove this assumption by binary searching for c. Start each point in the union of $\mathcal{S}$ with unit weight. Then find a $\frac{1}{2c}$-net N for $\mathcal{S}$. By the result quoted above, $|N|$ is $O(c)$. If N is a solution to the hitting set problem, we return it as our approximation. Otherwise, we pick any set S_i in $\mathcal{S}$ not hit by N, and double the weights of all points in S_i. We then find a $\frac{1}{2c}$-net for the new weights, and repeat this process until it finds a hitting set.

LEMMA 8.8 The process above can continue for at most $4c\log(n/c)$ iterations before a hitting set of cardinality $O(c)$ is found.

Proof. At each iteration, the unhit set S_i has weight at most $\frac{1}{2c}$ times the total weight of points of $\mathcal{S}$ (otherwise N would have to hit it), so doubling the weights of points in S_i increases the total weight by a factor of at most $1+\frac{1}{2c}$. Now let C be an optimal hitting set. The weight of at least one member of C must double at each iteration. Thus, the total weight of C—and hence the total weight of $\mathcal{S}$—after k iterations is at least $c2^{k/c}$. Finally, $k = 4c\log(n/c)$ is a bound on the maximum k for which $c2^{k/c} \leq n(1+\frac{1}{2c})^k$. ■

We can now assemble the entire algorithm. We start by constructing the weighted set system $\mathcal{S}$. Next we use binary search to find a suitable value of c. At a trial c, we run the weight-doubling algorithm for $4c\log(c/n)$ iterations before changing c. After $O(\log c)$ trials, we find a value of c such that the algorithm's $\frac{1}{2c}$-net is a hitting set but its $\frac{1}{2c-2}$-net is not. By Lemma 8.8, this c is at most the true minimum cardinality of a hitting set. Since the ϵ-net is a hitting set of cardinality $O(c)$ and the optimum hitting set is a 3-approximate solution to the polyhedron separation problem, we have the following result:

THEOREM 8.18 Brönniman and Goodrich The algorithm above is a constant-factor approximation algorithm for the polyhedron separation problem.

The same technique works equally well if the outer polyhedron R is not convex, but the running time is slower, since there may now be $O(n^3)$ sets in the set system.

8.6.3 POINT SET SEPARATION

Assume we have a set of points in the plane, and each point has a third coordinate called elevation. We turn to the problem of fitting a minimum-complexity, continuous, piecewise-linear function $f(x, y)$ to these "scattered data." In general, we must triangulate the planar point set in order to guarantee an exact fit, meaning an interpolating surface passing through each elevated point, and all triangulations of the input points have the same number of triangles. (Of course, some triangulations may be better than others for considerations quite apart from combinatorial complexity.) If we allow an inexact fit, however, the problem of minimizing the complexity of the surface becomes quite interesting.

To formalize the inexact problem, assume that the input includes an error tolerance $\epsilon > 0$ and the problem asks for a piecewise linear function $f(x, y)$ with a minimum number of faces, such that for each input point (x_i, y_i, z_i), we have $|z_i - f(x_i, y_i)| \leq \epsilon$. This surface fitting problem is a special case of a point set separation problem: separate two given sets of points in $\mathbb{R}^3$ by a terrain with a minimum number of facets. A *terrain* is the graph of a piecewise-linear function, or equivalently a polyhedral surface intersected exactly once by each vertical line.

The two-dimensional version of the surface fitting problem asks for a univariate piecewise-linear function passing through a collection of vertical line segments (error bars); this problem—and more generally, the two-dimensional version of the point set separation problem—can be solved in polynomial time [HS91a]. Notice, however, that a slightly different two-dimensional point set separation problem, which asks for a general separating polygon, is *NP*-hard; this is exactly the red-blue separation problem mentioned in Section 8.2.3.

Agarwal and Suri [AS94] showed that the three-dimensional surface fitting problem is *NP*-hard using a reduction from planar 3-SAT. It is not clear, however, whether the problem is *NP*-complete, as good bounds are not known on the numerical precision needed to represent the optimal function. Agarwal and Suri also devised an algorithm with logarithmic approximation ratio. Their algorithm works for the point-set separation problem as well, so we shall describe it in this more general context.

As in the polyhedron separation problem, the first step is to restrict the output to some canonical form. It is convenient to base the desired piecewise-linear function on trapezoids, rather than triangles. A *canonical trapezoid* T in the xy-plane satisfies the following restrictions: T has two horizontal sides, each passing through (the projection onto the xy-plane of) an input point, and the non-horizontal sides of T must each pass through two input points. If, moreover, some plane in $\mathbb{R}^3$ separates the two sets of points whose projections are covered by T, we call T a *splitting canonical trapezoid*. This last condition can be checked by fixed-dimensional linear programming.

A *canonical function* $f(x, y)$ is a continuous, piecewise-linear function, induced by a collection $\mathcal{C}_f$ of splitting canonical trapezoids with disjoint interiors. The domain of $f(x, y)$ is assumed to be a canonical trapezoid that covers the xy-projections of all input points. The splitting trapezoids in $\mathcal{C}_f$ need not cover the entire domain, as $f(x, y)$ can be filled in by triangulating the gaps between the splitting canonical trapezoids. We shall measure the complexity of $f(x, y)$ by $|\mathcal{C}_f|$, which is within a constant factor of the number of pieces in the filled-in function. Figure 8.19 shows that any triangle in an optimal (non-canonical) solution can be cut into at most four splitting canonical trapezoids. (Any face of an optimal solution with more than three sides can of course be triangulated, again increasing the total complexity by at most a constant factor.) Hence, restricting attention to canonical functions entails only a constant factor increase in complexity.

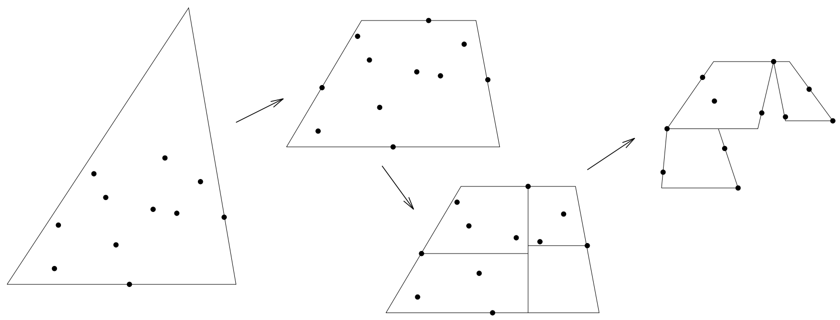


FIGURE 8.19

Any triangle can be shrunk and cut into at most 4 canonical trapezoids.

At this point, we have reduced the problem to covering the (xy-projections of the) n input points by a minimum number of disjoint, splitting canonical trapezoids. We could treat this as a set cover problem; however, this ignores the disjointness requirement. Enforcing disjointness—say by triangulating the planar graph formed by the overlapping trapezoids—blows up the complexity by a quadratic factor, giving $\Omega(k^2)$ pieces when k is optimal.

To improve the approximation ratio, we turn from set covering to dynamic programming. To ensure that the dynamic programming runs in polynomial time, however, we must further restrict the space of piecewise-linear functions. We call a cover of the input by splitting canonical trapezoids *recursively separable* if it can be constructed—starting from a single large canonical trapezoid—by recursively dividing canonical trapezoids in two with cutting lines connecting opposite edges. Agarwal and Suri use an algorithm of Paterson and Yao [PY90], which computes a binary space partition (see Section 8.2.3) for a set of line segments, to show that any disjoint cover by k canonical trapezoids can be transformed into a recursively separable cover with $O(k \log k)$ trapezoids. (There is also a direct proof [Mit93] of this result using horizontal cutting lines determined by a segment tree.)

Now the dynamic programming algorithm computes a globally optimal recursively separable cover. There are $O(n^6)$ different canonical trapezoids—one choice of input point for each horizontal side and two for each of the other sides. The points in any canonical trapezoid T can be partitioned in $O(n^2)$ ways into two smaller canonical trapezoids, and the optimal recursively separable cover for T is the combination of optima for some two smaller trapezoids. The base of this recurrence is the collection of splitting canonical trapezoids. We now have an $O(n^8)$-time dynamic programming algorithm for finding an optimal recursively separable partition.

THEOREM 8.19 Agarwal and Suri Let the optimum piecewise-linear function separating two sets of points in $\mathbb{R}^3$ have k pieces. Then the algorithm sketched above computes a piecewise-linear function with $O(k \log k)$ pieces in polynomial time.

ODDS AND ENDS

8.7

In this section we briefly survey a number of more isolated geometric approximation problems that do not fit into the other sections.

8.7.1 COVERING ORTHOGONAL POLYGONS BY RECTANGLES

An *orthogonal polygon* is a polygonal region in which all sides are either horizontal or vertical. The polygon is *simple* if its boundary has a single connected component, otherwise the polygon has *holes*. Given an orthogonal polygon P, the *rectangle covering* problem asks for a minimum number of (overlapping) rectangles whose union is P. Culberson and Reckhow [CR88] showed that this problem is *NP*-complete, even if P is simple. Berman and DasGupta [BD92] showed that if P has holes, the problem is also Max-$\mathcal{SNP}$-hard. This 1992 result seems to be the first proof of Max-$\mathcal{SNP}$-hardness for a geometric problem.

There are also some approximation results for rectangle covering. The rectangle covering problem can easily be translated into a set cover problem and approximated within a logarithmic factor; however, if the orthogonal polygon is in general position—no two boundary segments collinear—much better approximations are possible. The polygon can be cut into $n/2 + h$ vertical strips, where n is the number of sides and h is the number of holes, by passing a line through each vertical boundary segment. On the other hand, each polygon side requires a separate rectangle side, so any cover has at least $n/4$ rectangles. Thus, the approximation ratio of this method is 3 for polygons with holes and 2 for simple polygons. Berman and DasGupta [BD92] give several similar constant-factor approximations for covering the boundary or vertices of a given orthogonal polygon (not necessarily in general position).

OPEN PROBLEM 8.8 Is there a constant-factor approximation algorithm for covering arbitrary orthogonal polygons by rectangles?

8.7.2 PACKING SQUARES WITH FIXED CORNERS

Formann and Wagner [FW91] (see also [WW95]) define the following packing problem, motivated by the placement of text labels in cartography. The input consists of n points in the plane, along with a not-yet-embedded rectangle for each point. The task is to place all rectangles so that each rectangle is axis-aligned, has its point at one of its corners, and overlaps no other rectangle.

Formann and Wagner prove that this problem is *NP*-complete, even if all rectangles are unit squares. They then define an optimization version of this special case, which seeks the largest value s such that it allows each point to have a square label of side length s, and gives an approximation algorithm. The algorithm starts with s very small and all four candidate squares at each point, and then gradually increases s. At each step, if one of the four candidate squares covers half the distance to a neighboring point, it is removed. A remaining square is called *pending* if it overlaps another remaining square. Formann and Wagner show that a point with only pending squares has at most two remaining squares. This implies that the problem of placing squares for these points can be translated into a 2-SAT instance. If this instance is unsatisfiable, the algorithm halts and returns the last feasible square size found. Overall, this algorithm gives an approximation ratio of 2. Formann and Wagner go on to prove that no better approximation ratio is possible by a careful analysis of the construction used in their *NP*-completeness proof.

8.7.3 LARGEST CONGRUENT SUBSETS

Given a family of finite point sets in $\mathbb{R}^d$, $d \geq 1$, we wish to find a maximum-cardinality set congruent to a subset in each set of the family. Akutsu and Halldórsson [AH94a] define this problem, and show by a reduction from maximum clique, that for some $\epsilon > 0$ no $O(n^\epsilon)$-approximation is possible. This result and Berman and DasGupta's result on rectangle covering currently seem to be the only geometric applications of the new non-approximability results.

Akutsu and Halldórsson also give an $O(n_i/\log n)$-approximation for the same common congruent point set problem, where n_i denotes the size of the smallest input set S_i. In fact, an exhaustive search algorithm works: partition S_i into $O(n_i/\log n)$ pieces of size $O(\log n)$, and test each subset of each piece.

8.7.4 POLYGON BISECTION

Koutsoupias, Papadimitriou, and Sideri [KPS90] consider the following natural problem: given a simple polygon, partition it into two equal-area subsets using curved "fences" of minimum total length. Note that the two subsets need not be connected sets. A construction involving rectangular regions connected by thin "necks" shows the problem to be *NP*-complete; and moreover, that no approximation ratio depending only on n and not also on the geometry, is possible.

To salvage an approximation result, Koutsoupias et al. consider the following relaxation: a partition is an ϵ-bisection if the ratio of the subset sizes is between $1-\epsilon$ and $1+\epsilon$. They first show that an optimal bisection consists of circular arcs that meet the polygon either at a vertex or perpendicularly at an edge. Then they pick an area discretization parameter δ $(= \delta(\epsilon, n))$ and find a collection $\mathcal{A}$ $(= \mathcal{A}(\delta))$ of arcs, such that for any arc of the optimal solution, there is an arc in $\mathcal{A}$ of lesser or equal length that cuts off a similar area with an error of at most δ. Finally, they use dynamic programming to find the shortest subset of $\mathcal{A}$ that cuts off half the discretized area. The result is an algorithm, with time polynomial in n and $1/\epsilon$, that finds an ϵ-bisection at least as short as the optimum true bisection. It remains open whether any similar approximation result can be obtained for non-simple polygons.

8.7.5 GRAPH EMBEDDING

Given an n-vertex graph G and a set S of n points in the plane, the *geometric embedding problem* asks for a mapping of the vertices of G onto the points of S to minimize the total embedded edge length. This problem is *NP*-complete, as it generalizes the Euclidean traveling salesman problem. (Let G be a cycle.) Bern, Karloff, Raghavan, and Schieber [BKRS92] give an $O(\log n)$-approximation for the special case in which G is a bounded-degree tree. Hansen [Han89] considers the embedding problem for highly regular graphs such as d-dimensional grids, hypercubes, and butterfly networks. A ratio of $O(\log n)$ is possible in all these cases, except planar grids, for which the ratio rises to $O(\log^2 n)$. Hansen also considers a different special case, in which G is arbitrary but P is a grid; in this case he achieves an $O(\log^2 n)$ approximation. This last result uses the Leighton-Rao approximation algorithm for graph separators [LR88], so better separator results (claimed in [CY94]) should improve the approximation ratio.

8.7.6 LOW-DEGREE SPANNING TREES

The problem of finding a traveling salesman path can be viewed as a degree-constrained minimum spanning tree problem, in which each vertex has degree at most two. With this in mind, it becomes interesting to consider weaker bounds on the vertex degree of a minimum spanning tree. Khuller, Raghavachari, and Young [KRY94] consider this problem for degree bounds of three and four; they show that one can find constrained spanning trees with length 1.5 and 1.25 times the MST length, respectively. It is no constraint to require the degree to be at most five, as there always exists an MST satisfying this requirement. These results are described in more detail by Raghavachari in Chapter 7.

8.7.7 SHORTEST PATHS IN SPACE

In the Euclidean plane, the shortest path connecting a source and a destination and avoiding a set of polygonal obstacles, is always a subgraph of the *visibility graph*, which connects two vertices u and v if the line segment uv avoids all obstacles. The three-dimensional generalization of this shortest path problem, however, is *NP*-hard [CR87], largely because a shortest path may wrap around a polyhedral obstacle at almost any location on an edge.

Choi et al. [CSY94], building on earlier work of Papadimitriou [Pap85], provide a polynomial-time approximation scheme for this problem, assuming an integer model of computation in which each obstacle vertex has coordinates specified by L-bit integers. The running time is polynomial in n and $1/\epsilon$, and requires computation with numbers of $O(L+\log(n/\epsilon))$ bits. The basic idea is to place a finely-spaced collection of vertices along each obstacle edge, and then compute the shortest path in the visibility graph of these points. Sophisticated techniques are required to keep the number of points polynomially bounded, and to analyze the necessary bit complexity.

8.7.8 LONGEST SUBGRAPH PROBLEMS

We have already mentioned the recent results by Barvinok [Bar94] on the Euclidean longest traveling salesman problem. Alon, Rajagopalan, and Suri [ARS93] consider several other longest-subgraph problems, with the additional constraint that the subgraph may not include any edge crossings. (Shortest subgraphs—matchings, trees, or tours—satisfy this constraint automatically.)

Alon et al. first show how to compute a maximum non-crossing matching with length at least $2/\pi$ times optimal. Choose an angle θ uniformly at random, split the input points in half by a line parallel to θ, and compute any non-crossing matching in which each edge crosses the line. The ratio follows from the observations that a projection of the optimum matching onto a line perpendicular to θ has expected length diminished by only $2/\pi$, and the constructed matching has length at least that of the optimum matching's projection. This algorithm can be derandomized by simply trying all combinatorially distinct bipartitions of the input by lines.

To approximate the maximum non-crossing Hamiltonian path, Alon et al. show that the matching constructed above can be extended to such a path. The result then follows from the fact that the maximum matching contains at least half the weight of the optimal path. Curiously, this method also gives an approximation to the unconstrained maximum Hamiltonian path problem and to the longest traveling salesman tour.

Alon et al. also consider the maximum non-crossing spanning tree, and show that this tree always has a constant-factor approximated by a *star*, a tree in which one vertex is connected to all others. Two similar problems, maximum bounded-degree spanning tree and maximum triangulation, can also be approximated, the first because the tree can be approximated by a matching and the second because the triangulation can be approximated by a spanning tree.

Alon et al. leave open the complexity of exactly solving the problems they treat, although they suspect the problems to be *NP*-complete. Approximating the maximum non-crossing cycle also appears to be open.

8.8 CONCLUSIONS

We now mention a major result of Arora [Aro96], too recent for us to incorporate into the main body of this chapter. Arora has announced a polynomial-time approximation scheme for Euclidean TSP. For any $\epsilon > 0$, a solution of length at most $1+\epsilon$ times optimal can be found in time $n^{O(1/\epsilon)}$. The result extends to many other geometric problems, including Euclidean Steiner tree, and generalizes to $\mathbb{R}^d$ but with quasi-polynomial running time $n^{O(\log^{d-2} n)/\epsilon^{d-1}}$. The approach is reminiscent of the approach to TSP with neighborhoods and k-MST described in this chapter: there is a near-optimal solution of a special form defined by a recursively constructed tree, and an optimal solution of this form can be found by dynamic programming. Arora's result implies that Euclidean TSP and other famous geometric problems cannot be Max-$\mathcal{SNP}$-hard, unless $P = NP$.

The original conclusion to this chapter, written in December 1994, stressed the immaturity of the geometric area relative to other areas such as graph problems. We observed that each hard geometric problem called for its own approximation method, whereas graph problems admitted unified approaches such as the primal-dual method discussed in Chapter 4. This situation has changed in less than a year and a half. A powerful unified approach has emerged: dynamic programming over a recursively defined subset of solutions. Finally, Arora's new result has clarified the Max-$\mathcal{SNP}$ status of many geometric problems. We look forward to further rapid developments.

Acknowledgments We thank Ding-Zhu Du, Naveen Garg, Dorit Hochbaum, Joe Mitchell, Günter Rote, Subhash Suri, and Alexander Zelikovsky for valuable suggestions. We remember Gene Lawler, whose clear writing style we have tried to emulate.

REFERENCES

[AA93] N. Alon and Y. Azar. On-line Steiner trees in the Euclidean plane. *Disc. and Comp. Geometry*, 10:113–121, 1993.

[AABV94] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. Technical Report CMU-CS-94-173, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, 1994.

[ABO+89] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C.K. Yap. Finding convex minimal nested polygons. *Information and Control*, 83:98–110, 1989.

[AH94a] T. Akutsu and M.M. Halldórsson. On the approximation of largest common subtrees and largest common point sets. In *Proc. 5th Int. Symp. Algorithms and Computation*, pages 405–413. Springer-Verlag, LNCS 834, 1994.

[AH94b] E.M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Disc. Appl. Math.*, 55:197–218, 1994.

[AIKS91] A. Aggarwal, H. Imai, N. Katoh, and S. Suri. Finding k-points with minimum diameter and related problems. *J. Algorithms*, 12:38–56, 1991.

[Aro96] S. Arora. A polynomial time approximation scheme for Euclidean TSP and other geometric problems. Manuscript, 1996.

[ARS93] N. Alon, S. Rajagopalan, and S. Suri. Long non-crossing configurations in the plane. In *Proc. 9th ACM Symp. Computational Geometry*, pages 257–263, 1993.

[AS94] P.K. Agarwal and S. Suri. Surface approximation and geometric partitions. In *Proc. 5th ACM/SIAM Symp. Discrete Algorithms*, pages 24–33, 1994.

[Aza94] Y. Azar. Lower bounds for insertion methods for TSP. *Combinatorics, Probability, and Computing*, 3:285–292, 1994.

[Bak94] B.S. Baker. Approximation algorithms for *NP*-complete problems on planar graphs. *J. ACM*, 41:153–180, 1994.

[Bar94] A.I. Barvinok. Two algorithmic results for the traveling salesman problem. Manuscript, 1994.

[BCV95] A. Blum, P. Chalasani, and S. Vempala. A constant-factor approximation for the k-MST problem in the plane. In *Proc. 27th ACM Symp. Theory of Computing*, 1995.

[BD92] P. Berman and B. DasGupta. Approximating the rectilinear polygon cover problems. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 229–235, 1992.

[BD95] A. Borchers and D-.Z. Du. The k-Steiner ratio in graphs. In *Proc. 27th ACM Symp. Theory of Computing*, 1995.

[BDG95] A. Borchers, D-.Z. Du, and B. Gao. The k-Steiner ratio in the rectilinear plane. Technical Report TR 95-028, Dept. of Computer Science, Univ. of Minnesota, 1995.

[BEG94] M. Bern, D. Eppstein, and J.R. Gilbert. Provably good mesh generation. *J. Comp. System Science*, 48:384–409, 1994.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. *J. Assoc. Comput. Mach.*, 36:929–965, 1989.

[Ben92] J.L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA J. Computing*, 4:387–411, 1992.

[Ber87] M. Bern. *Network Design Problems: Steiner trees and minimum spanning k-trees*. PhD thesis, Computer Science Division, University of California at Berkeley, 1987.

[Ber88] M. Bern. Two probabilistic results on rectilinear Steiner trees. *Algorithmica*, 3:191–204, 1988.

[BET93] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. In *Proc. 3rd Workshop on Algorithms and Data Structures*, pages 188–199. Springer-Verlag, LNCS 709, 1993.

[BFK$^+$94] P. Berman, U. Fößmeier, M. Karpinski, M. Kaufmann, and A. Zelikovsky. Approaching the 5/4-approximation for rectilinear Steiner trees. In *Proc. 2nd Eur. Symp. Algorithms*, pages 60–71. Springer-Verlag, LNCS 855, 1994.

[BG94] H. Brönnimann and M.T. Goodrich. Almost optimal set covers in finite VC-dimension. In *Proc. 10th ACM Symp. Computational Geometry*, pages 293–302, 1994.

[BH89] E. Boros and P. Hammer. On clustering problems with connected optima in Euclidean spaces. *Disc. Math.*, 75:81–88, 1989.

[BKP94] V. Bafna, B. Kalyanasundaram, and K. Pruhs. Not all insertion methods yield constant approximate tours in the Euclidean plane. *Theoretical Computer Science*, 125:345–353, 1994.

[BKRS92] M. Bern, H. Karloff, P. Raghavan, and B. Schieber. Fast geometric approximation techniques and geometric embedding problems. *Theoretical Comp. Sci.*, 106:265–281, 1992.

[BP89] M. Bern and P. Plassmann. The Steiner problem with edge lengths 1 and 2. *Inform. Proc. Lett.*, 32:171–176, 1989.

[BR94] P. Berman and V. Ramaiyer. Improved approximation algorithms for the Steiner tree problem. *J. Algorithms*, 17:381–408, 1994.

[BS80] J.L. Bentley and J.B. Saxe. An analysis of two heuristics for the Euclidean travelling salesman problem. In *Proc. 18th Annual Allerton Conference on Communication, Control, and Computing*, pages 41–49, 1980.

[Cie90] D. Cieslik. The Steiner ratio of Banach-Minkowski planes. In R. Bodendiek, editor, *Contemporary Methods in Graph Theory*, pages 231–248. BI-Wissenschaftverlag, Mannheim, 1990.

[CK94] S.Y. Cheung and A. Kumar. Efficient quorumcast routing algorithms. In *Proc. IEEE INFOCOM '94 Conference on Computer Communications*, volume 2, pages 840–847, 1994.

[CKT94] B. Chandra, H. Karloff, and C. Tovey. New results on the old k-opt algorithm for the TSP. In *Proc. 5th ACM-SIAM Symp. Disc. Algorithms*, pages 150–159, 1994.

[Cla91] K. Clarkson. Approximation algorithms for planar traveling salesman tours and minimum-length triangulations. In *Proc. 2nd ACM-SIAM Symp. Disc. Algorithms*, pages 17–23, 1991.

[Cla93] K.L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Worksh. Algorithms and Data Structures*, pages 246–252. Springer-Verlag Lecture Notes in Computer Science 709, 1993.

[CR87] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th IEEE Symp. Found. Comp. Sci.*, pages 49–60, 1987.

[CR88] J.C. Culberson and R.C. Reckhow. Covering polygons is hard. In *Proc. 29th IEEE Symp. Found. Comp. Sci.*, pages 601–611, 1988.

[CRW91] V. Capoyleas, G. Rote, and G. Woeginger. Geometric clusterings. *J. Algorithms*, 12:341–356, 1991.

[CSY94] J. Choi, J. Sellen, and C.-K. Yap. Approximate Euclidean shortest path in 3-space. In *Proc. 10th ACM Symp. Comp. Geometry*, pages 41–48, 1994.

[CY94] F.R.K. Chung and S.-T. Yau. A near optimal algorithm for edge separators. In *Proc. 26th ACM Symp. Theory of Computing*, pages 1–8, 1994.

[DGG$^+$] D.-Z. Du, B. Gao, R.L. Graham, Z.-C. Liu, and P.-J. Wan. Minimum Steiner trees in normed planes. To appear in *Disc. and Comp. Geometry*.

[DH92a] D.-Z. Du and F.K. Hwang. A proof of Gilbert-Pollak's conjecture on the Steiner ratio. *Algorithmica*, 7:121–135, 1992.

[DH92b] D.-Z. Du and F.K. Hwang. The state of art in Steiner ratio problems. In D.-Z. Du and F.K. Hwang, editors, *Computing in Euclidean Geometry*, pages 163–192. World Scientific, Singapore, 1992.

[DJ90] G. Das and D. Joseph. The complexity of minimum convex nested polyhedra. In *Proc. 2nd Canad. Conf. Computational Geometry*, pages 296–301, 1990.

[DLSS93] A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. In *Proc. 3rd Workshop on Algorithms and Data Structures*, pages 265–276. LNCS 709, Springer-Verlag, 1993.

[DM95] M. Dickerson and M. Montague. A (usually) connected subgraph of the minimum weight triangulation. In *5th MSI Worksh. Computational Geometry*, 1995.

[Du92] D-.Z. Du. Disproving Gilbert-Pollak conjecture in high dimensional spaces. Technical Report TR 92-56, Dept. of Computer Science, Univ. of Minnesota, 1992.

[DZF91] D.-Z. Du, Y. Zhang, and Q. Feng. On better heuristic for Euclidean Steiner minimum trees. In *Proc. 32nd IEEE Symp. Found. Comp. Sci.*, pages 431–439, 1991.

[EE94] D. Eppstein and J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Disc. and Comp. Geometry*, 11:321–350, 1994.

[Epp94] D. Eppstein. Approximating the minimum weight triangulation. *Disc. and Comp. Geometry*, 11:163–191, 1994.

[Epp95] D. Eppstein. Faster geometric k-point MST approximation. Technical Report 95-13, Dept. of Information and Computer Science, UC-Irvine, 1995.

[ERS90] H. Edelsbrunner, A.D. Robinson, and X. Shen. Covering convex sets with non-overlapping polygons. *Discrete Math.*, 81:153–164, 1990.

[FG88] T. Feder and D.H. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th ACM Symp. Theory of Computing*, pages 434–444, 1988.

[FHJM94] M. Fischetti, H.W. Hamacher, K. Jørnsten, and F. Maffioli. Weighted k-cardinality trees: complexity and polyhedral structure. *Networks*, 24:11–21, 1994.

[FW91] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th ACM Symp. Comp. Geometry*, pages 281–288, 1991.

[GDG94] B. Gao, D.-Z. Du, and R.L. Graham. The tight lower bound for the Steiner ratio in Minkowski planes. In *Proc. 10th ACM Symp. Computational Geometry*, pages 183–191, 1994.

[GEP95] M. Grigni, E.Koutsoupias, and C. Papadimitriou. An approximation scheme for the planar graph TSP. In *Proc. 36th IEEE Symp. Found. Comp. Sci.*, pages 640–645, 1995.

[GGJ77] M.R. Garey, R.L. Graham, and D.S. Johnson. The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.*, 32:835–859, 1977.

[GH76] R.L. Graham and F.K. Hwang. Remarks on Steiner minimum trees. *Bull. Inst. Math. Acad. Sinica*, 4:177–182, 1976.

[GH94] N. Garg and D.S. Hochbaum. An $O(\log k)$ approximation for the k-minimum spanning tree problem in the plane. In *Proc. 26th ACM Symp. Theory of Computing*, pages 432–438, 1994.

[Gil79] P.D. Gilbert. New results in planar triangulations. Technical Report R–850, Univ. Illinois Coordinated Science Lab, 1979.

[GJ77] M.R. Garey and D.S. Johnson. The rectilinear Steiner tree problem is *NP*-complete. *SIAM J. Appl. Math.*, 32:826–834, 1977.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Gon85] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[GP68] E.N. Gilbert and H.O. Pollak. Steiner minimal trees. *SIAM J. Appl. Math.*, 16:1–29, 1968.

[Han89] M.D. Hansen. Approximation algorithms for minimum cost embeddings of graphs in the plane. In *Proc. 30th IEEE Symp. Found. Comp. Sci.*, pages 604–609, 1989.

[HMR+94] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R.E. Stearns. A unified approach to approximation schemes for *NP*- and PSPACE-hard problems for geometric graphs. In *Proc. 2nd Eur. Symp. Algorithms*, pages 424–435. Springer-Verlag, LNCS 855, 1994.

[HP94] L.S. Heath and S.V. Pemmaraju. New results for the minimum weight triangulation problem. *Algorithmica*, 12:533–552, 1994.

[HRW92] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem*. North-Holland, 1992.

[HS91a] S.L. Hakimi and E.F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models and Image Processing*, 53:132–136, 1991.

[HS91b] J. Hershberger and S. Suri. Finding tailored partitions. *J. Algorithms*, 12:431–463, 1991.

[Hur92] C. Hurkens. Nasty TSP instances for farthest insertion. In *Proc. 2nd Conf. on Integer Programming and Combinatorial Optimization*, pages 346–352. Mathematical Programming Society, 1992.

[Hwa76] F.K. Hwang. On Steiner minimal trees with rectilinear distance. *SIAM J. Appl. Math.*, 30:104–114, 1976.

[HY90] F.K. Hwang and Y.C. Yao. Comments on Bern's probabilistic results on rectilinear Steiner trees. *Algorithmica*, 5:591–598, 1990.

[IKI94] M. Inaba, N. Katoh, and H. Imai. Applications of weighted Voronoi diagrams and randomization of variance-based k-clustering. In *Proc. 10th ACM Symp. Computational Geometry*, pages 332–339, 1994.

[IT94] A.O. Ivanov and A.A. Tuzhilin. *Minimal Networks: The Steiner Problem and its Generalizations*. CRC Press, Inc., 1994.

[Joh74] D.S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Science*, 9:256–278, 1974.

[JP85] D.S. Johnson and C.H. Papadimitriou. Performance guarantees for heuristics. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors, *The Traveling Salesman Problem*. John Wiley, 1985.

[Kir80] D.G. Kirkpatrick. A note on Delaunay and optimal triangulations. *Inform. Proc. Lett.*, 10:127–128, 1980.

[KPS90] E. Koutsoupias, C.H. Papadimitriou, and M. Sideri. On the optimal bisection of a polygon. In *Proc. 6th ACM Symp. Computational Geometry*, pages 198–202, 1990.

[KRY94] S. Khuller, B. Raghavachari, and N. Young. Low-degree spanning trees of small weight. In *Proc. 26th ACM Symp. Theory of Computing*, pages 412–421, 1994.

[KZ94] M. Karpinski and A.Z. Zelikovsky. 1.757 and 1.267-approximation algorithms for the network and rectilinear Steiner tree problems. 1994.

[Lin83] A. Lingas. *Advances in minimum weight triangulation*. Ph.D. thesis, Linköping Univ., 1983.

[LK96] C. Levcopoulos and D. Krznaric. Quasi-greedy triangulation approximating the minimum weight triangulation. In *Proc. 7th ACM-SIAM Symp. Discrete Algorithms*, 1996.

[LL87] C. Levcopoulos and A. Lingas. On approximation behavior of the greedy triangulation for convex polygons. *Algorithmica*, 2:175–193, 1987.

[LL92] C. Levcopoulos and A. Lingas. Fast algorithms for greedy triangulation. *BIT*, 32:280–296, 1992. Also in *Proc. 2nd Scand. Worksh. Algorithm Theory*, Springer-Verlag, LNCS 447, 1990, pp. 238–250.

[LLKS85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem*. John Wiley, 1985.

[Llo77] E.L. Lloyd. On triangulations of a set of points in the plane. In *Proc. 18th IEEE Symp. Found. Comp. Sci.*, pages 228–240, 1977.

[LR88] F.T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proc. 29th IEEE Symp. Found. Comp. Sci.*, pages 422–431, 1988.

[Mat92] J. Matoušek. Reporting points in halfspaces. *Computational Geom. Theory & Appl.*, 2:169–186, 1992.

[Mel61] Z.A. Melzak. On the problem of Steiner. *Canad. Math. Bull.*, 4:143–148, 1961.

[Mit93] J.S.B. Mitchell. Approximation algorithms for geometric separation problems. Manuscript, 1993.

[Mit96] J.S.B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: a simple new method for the geometric k-MST problem. In *Proc. 7th ACM-SIAM Symp. Discrete Algorithms*, 1996.

[MM95] C.S. Mata and J.S.B. Mitchell. Approximation algorithms for geometric tour and network design problems. In *Proc. 11th ACM Symp. Computational Geometry*, pages 360–369, 1995.

[MS84] N. Megiddo and K.J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13:182–196, 1984.

[MS92] J.S.B. Mitchell and S. Suri. Separation and approximation of polyhedral objects. In *Proc. 3rd ACM/SIAM Symp. Discrete Algorithms*, pages 296–306, 1992.

[MSW90] J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: small ϵ-nets for disks and halfspaces. In *Proc. 6th ACM Symp. Computational Geom.*, pages 16–22, 1990.

[Mul94] K. Mulmuley. *Computational Geometry: An Introduction through Randomized Algorithms*. Prentice-Hall, Englewood Cliffs, 1994.

[MY80] W. Meeks and S.T. Yau. Topology of three-dimensional manifolds and the embedding problem in minimal surface theory. *Ann. Math.*, 112:441–485, 1980.

[MZ79] G.K. Manacher and A.L. Zobrist. Neither the greedy nor the Delaunay triangulation approximates the optimum. *Inform. Proc. Lett.*, 9:31–34, 1979.

[OTZ88] S. Olariu, S. Toida, and M. Zubair. On a conjecture by Plaisted and Hong. *J. Algorithms*, 9:597–598, 1988.

[Pap85] C. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Inform. Proc. Lett.*, 20:259–263, 1985.

[PH87] D.A. Plaisted and J. Hong. A heuristic triangulation algorithm. *J. Algorithms*, 8:405–437, 1987.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[PY90] M.S. Paterson and F.F. Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Disc. and Comp. Geometry*, 5:485–503, 1990.

[PY92] M.S. Paterson and F.F. Yao. Optimal binary space partitions for orthogonal objects. *J. Algorithms*, 13:99–113, 1992.

[PY93] C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Research*, 18:1–11, 1993.

[RSL77] D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, 6:563–581, 1977.

[RSM+94] R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrantz, and S.S. Ravi. Spanning trees short and small. In *Proc. 5th ACM-SIAM Symp. Discrete Algorithms*, pages 546–555, 1994.

[RT91a] J.H. Rubinstein and D.A. Thomas. The Steiner ratio conjecture for six points. *J. of Combinatorial Theory, Ser. A*, 58:54–77, 1991.

[RT91b] J.H. Rubinstein and D.A. Thomas. A variational approach to the Steiner network problem. In J. MacGregor Smith and P. Winter, editors, *Topological Network Design, Annals of Operations Research 33*, pages 481–500. J.C. Baltzer AG, Basel, 1991.

[Sil79] C.B. Silio. An efficient simplex coverability algorithm in E^2 with application to stochastic sequential machines. *IEEE Transactions on Computers*, C-28:109–120, 1979.

[Smi88] W.D. Smith. Finding the optimum N-city traveling salesman tour in the Euclidean plane in subexponential time and polynomial space. Manuscript, 1988.

[Smi89] W.D. Smith. Implementing the Plaisted-Hong min-length plane triangulation heuristic. Manuscript cited by [Cla91], 1989.

[Smi92] W.D. Smith. How to find Steiner minimal trees in Euclidean d-space. *Algorithmica*, 7:137–179, 1992.

[Sur90] S. Suri. On some link distance problems in a simple polygon. *IEEE Trans. Robotics and Automation*, 6:108–113, 1990.

[Vai88] P. Vaidya. Geometry helps in matching. In *Proc. 20th ACM Symp. Theory of Computing*, pages 422–425, 1988.

[Vai89] P. Vaidya. An $O(n \log n)$ algorithm for the all-nearest-neighbors problem. *Disc. and Comp. Geometry*, 4:101–115, 1989.

[War93] A.R. Warburton. Worst-case analysis of some convex hull heuristics for the Euclidean traveling salesman problem. *Oper. Res. Lett.*, 13:37–42, 1993.

[WW95] F. Wagner and A. Wolff. Map labeling heuristics: provably good and practically useful. In *Proc. 11th ACM Symp. Comp. Geometry*, pages 109–118, 1995.

[Zel92] A.Z. Zelikovsky. An 11/8-approximation algorithm for the Steiner problem in networks with rectilinear distance. *Coll. Math. Soc. J. Bolyai*, 60:733–745, 1992.

[Zel93] A.Z. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9:463–470, 1993.

[Zel94] A.Z. Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. Manuscript, 1994.

[ZL93] A.Z. Zelikovsky and D.D. Lozovanu. Minimal and bounded trees. In *Tezele Congresului XVIII al Academiei Romano-Americane*, pages 25–26. Kishinev, 1993.