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.

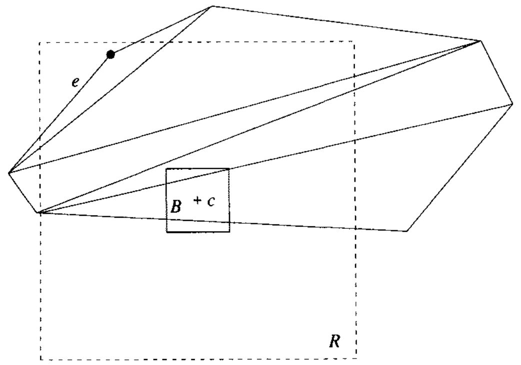

FIGURE 8.16

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 9.2). 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 nonapproximability 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 $d_2' = 2\cos(10°) \approx 1.969$. ■

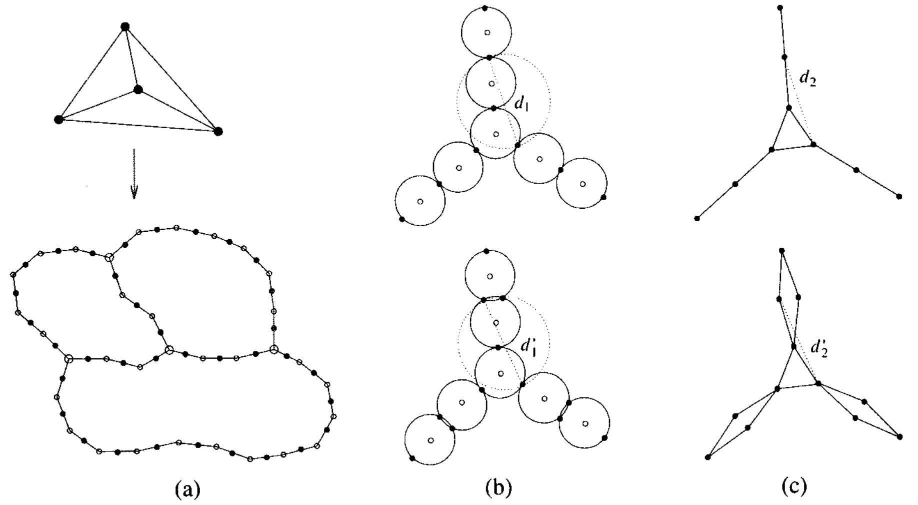

FIGURE 8.17

(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.