CLOSEST-POINT PROBLEMS

Michael Ian Shamos[†] and Dan Hoey

Department of Computer Science, Yale University
New Haven, Connecticut 06520

Abstract

A number of seemingly unrelated problems involving the proximity of N points in the plane are studied, such as finding a Euclidean minimum spanning tree, the smallest circle enclosing the set, k nearest and farthest neighbors, the two closest points, and a proper straight-line triangulation. For most of the problems considered a lower bound of O(N log N) is shown. For all of them the best currently-known upper bound is $O(N^2)$ or worse. The purpose of this paper is to introduce a single geometric structure, called the Voronoi diagram, which can be constructed rapidly and contains all of the relevant proximity information in only linear space. The Voronoi diagram is used to obtain O(N log N) algorithms for all of the problems.

I. Introduction

Computational geometry is of practical importance because Euclidean space of two and three dimensions forms the arena in which real physical objects are arranged. A large number of manufacturing problems involve wire layout[1], facilities location[2], cutting-stock and related geometric optimization problems[3]. Solving these efficiently on a high-speed computer requires the development of new geometrical tools, as well as the application of fast-algorithm techniques, and is not simply a matter of translating well-known theorems into computer programs. From a theoretical standpoint, the complexity of geometric algorithms is of interest because it sheds new light on the intrinsic difficulty of computation.

In this paper we are concerned specifically with problems involving the "closeness" of points in a finite set. How many operations, for example, are required to determine the two closest of N points? Given two points that are purported to be the closest pair, how long does it take to verify this? Nearest neighbor questions arise in clustering[4] and contouring[5] and in a number of other problems where the connection with closeness is not so evident.

We already know that it is crucial to exploit the geometric properties of these problems to the fullest when designing efficient algorithms[6] and to avoid embedding a problem in a structure so general that the geometric aspects are lost. As a case in point, consider the textbook example of linear programming in two variables with N constraints. The simplex method, which in higher dimensions achieves its speed by avoiding the combinatorially hopeless task of forming the feasible polytope explicitly, is inferior here because it requires $O(N^2)$ time. In the plane, however, it is possible[7] to construct the feasible polygon directly in O(N log N) time by geometric techniques. Embedding a small problem in a structure designed for larger ones obscures the inherent complexity of the problem and leads to inefficiencies. Burying a geometry problem in a graph model can also lead to difficulty. Finding the tree of minimum length joining N points, whose vertices are at the

[†]Present address: Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213. This work was supported in part by the Office of Naval Research under Contract N00014-75-C-0450 while this author held an IBM Fellowship at Yale University.

given points, is a problem that fundamentally involves 2N parameters, the coordinates of the points. This problem is usually formulated as a minimum spanning tree problem on the complete N-graph and the number of parameters immediately jumps to $O(N^2)$. In this paper we will endeavor to concentrate on the geometry of the problems and avoid such trouble.

II. The Problems

P1. (All closest points) *Given N points in the Euclidean plane, find the nearest neighbor of each.*
No algorithm has yet appeared that is faster than the naive approach of computing all $O(N^2)$ interpoint distances, but this does not achieve the lower bound:
Theorem 1. *O(N log N) is a lower bound on the time required to determine the two closest of N points in dimension one or higher.*
Proof: This problem is related to the element-uniqueness question, which asks whether any two of N real numbers are equal. O(N log N) is a lower bound on the element-uniqueness problem even if comparisons among linear functions of the input are allowed[8]. In linear time the element-uniqueness problem can be mapped into a closest-point problem in one dimension. We merely find the two closest points -- if their distance is zero the elements are not unique. The result follows in higher dimensions simply by embedding the line. □
This result implies an O(N log N) lower bound on the all closest points problem.
P2. (Euclidean minimum spanning tree) *Given N points in the plane, find an interconnecting tree of minimum total length whose vertices are the given points.*
This problem has been studied extensively in a graph-theoretic setting[9] [10] [11] and is usually solved by regarding the points as vertices of a complete N-graph whose edge weights are the Euclidean distances between the corresponding points. An $O(N^2)$ algorithm is known[12] and is easily seen to be optimal for arbitrary graphs: by a result of Prim[10] a shortest edge of the graph must occur in the minimum tree. The tree has exactly N-1 edges and the shortest of these can be found trivially in O(N) time. If the tree itself could be constructed in less than $O(N^2)$ time then we would be able to find the smallest of $O(N^2)$ independent quantities (the edge lengths) in less than $O(N^2)$ time, which is impossible. In the Euclidean problem, however, the edge lengths are not independent and we have a different lower bound:
Theorem 2. *O(N log N) is a lower bound on the time required to construct a Euclidean minimum spanning tree on N points in any dimension.*
Proof A: Since the two closest points are joined by an edge of the tree, given the tree we can find the two closest points in O(N) time. Thus by Theorem 1 it must take O(N log N) time to construct a minimum tree.
Proof B: Consider N points on a line. The minimum spanning tree consists of the line segments joining the points in the order in which they occur on the line. Given the tree, we can reconstruct the sorted list of points in O(N) time, so any minimum spanning tree algorithm can sort. □
P3. (Triangulation) *Given N points in the plane, join them by non-intersecting straight line segments so that every region interior to the convex hull is a triangle. In particular, find a triangulation the sum of whose edge lengths is a minimum.*
This problem arises in the numerical interpolation of bivariate data when function values are available at N irregularly-spaced data points (x_i, y_i) and an approximation to the function at a new point (x, y) is desired. One approach, called the polyhedron method[13], is to approximate the function surface by a network of triangular facets. Each point (x, y) then lies within a single facet and the function value is obtained by linear interpolation of the facet vertices. The minimum weight triangulation has good numerical properties and an $O(N^3)$ algorithm for constructing it is known[14]

which resembles Prim's minimum spanning tree procedure.

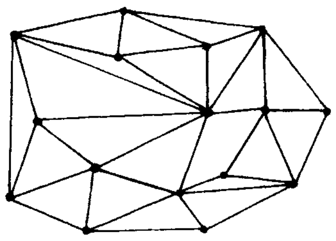
Figure 1. A triangulation.

Theorem 3. *O(N log N) is a lower bound on the time required to triangulate N points in dimension two or higher.*

Proof: Consider N collinear points and another point not on the line. This set possesses exactly one triangulation and from it one can reconstruct in O(N) time the points in sorted order. (Figure 2) Thus any triangulation algorithm must be able to sort. □

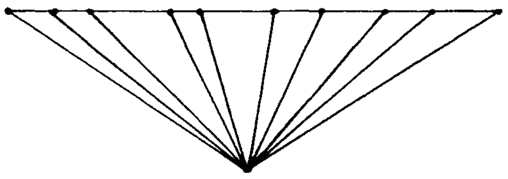
Figure 2. Triangulation lower bound.

Note that this lower bound applies to the formation of any triangulation, not just one of minimum weight.

P4. (Convex hull) *Given a set of N points in the plane, find its convex hull.*

An O(N log N) algorithm for this problem is known[15] and is optimal[6]. It is presented here because of its relation to other closest-point problems and a new algorithm will be given.

P5. (Largest empty circle) *Given N points in the plane, find a largest circle containing no points of the set yet whose center is interior to the convex hull.*

Such a circle is not necessarily unique. This problem can be viewed as one of maximin facilities layout in which it is desired to locate a new facility so that it is as far as possible from any of N existing ones. The new site may be for a source of pollution, for example, or for a new business establishment not wishing to compete for territory with stores already in the area. These questions are often encountered in operations research and industrial engineering[16]. For the present problem an algorithm has been given whose worst-case running time is $O(N^3)$ [17]. In one dimension the problem reduces to finding the two adjacent points on a line that are farthest apart and O(N log N) is a lower bound analogous to Thm. 1, in which we seek the two closest adjacent points.

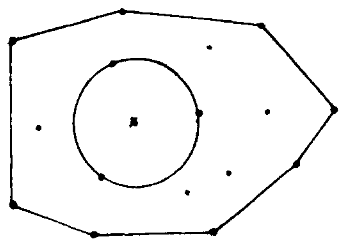
Figure 3. A largest empty circle.

P6. (*k* closest points) *Given N points in the plane, with preprocessing allowed, how quickly can the k points closest to a new point x be found?*

If preprocessing is not allowed then O(N) is both an upper and a lower bound: find the N distances from x to each of the points and find the k smallest by the linear selection algorithm of Blum et al[18]. With preprocessing, the best algorithm available has an expected run time of $O(\sqrt{kN})$ [19]. The information-theoretic lower bound is $O(\max(k, \log N))$. Algorithms for the case $k = 1$ are given in references 6 and 7.

P7. (Smallest enclosing circle) *Given N points in the plane, find the smallest circle enclosing them.*

This is a minimax facilities location problem in which we seek a point x (the center of the circle) whose greatest distance to any point of the set is a minimum. This criterion is useful in siting emergency services, where worst-case response time is an important consideration[20], and in optimizing the location of a radio transmitter serving a number of discrete stations so as to minimize the RF power required[21].

The smallest enclosing circle is unique and is determined either by three points of the set or two points which define a diameter[22]. Thus a finite algorithm exists which examines all circles determined by two or three points of the set and selects the smallest that encloses all the points. This rote method has been improved by Elzinga and Hearn[23] to run in $O(N^2)$ time and is the best algorithm to date[2]. The defining points of the smallest circle are extreme points of the set and identification of the extreme points of a set in two or more dimensions requires $O(N \log N)$ time[24], so we conjecture that the smallest circle also requires $O(N \log N)$ time.

The problems posed above are related in the sense that they all deal with the respective distances among points of a finite set. However, the algorithms that have been proposed to solve them, with the exception of P2 and P3, are not even remotely similar. We will now introduce a geometric structure which can be created in $O(N \log N)$ time and yields optimal algorithms for all of the closest-point problems.

III. The Voronoi Diagram

Surrounding each point p_i of a finite set there is a convex polygon $V(i)$, called the Voronoi polygon associated with p_i, having the property that p_i is the closest of the given points to any $x \in V(i)$. If $h(i,j)$ denotes the half-plane containing p_i defined by the perpendicular bisector of p_i and p_j, then we have $V(i) = \bigcap_{j \neq i} h(i,j)$, which shows that $V(i)$ is a convex polygonal region having at most N-1 sides. (Figure 4)

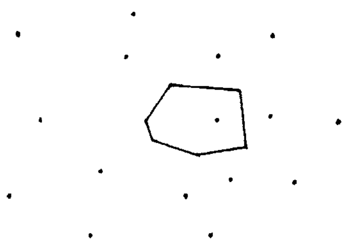
Figure 4. A Voronoi polygon.

The Voronoi polygons partition the plane[25], forming a net which we will refer to as the Voronoi diagram. Unbounded polygons correspond to vertices of the convex hull and the entire diagram contains only $O(N)$ Voronoi points and edges joining them[6]. The Voronoi points are circumcenters of triangles since they are the junctions of triples of points. (Figure 5)

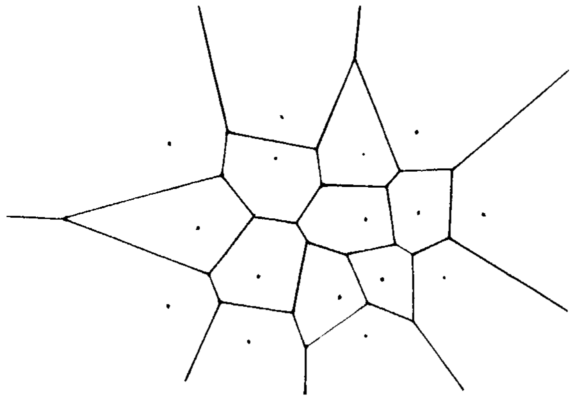
Figure 5. The Voronoi diagram.

Theorem 4. *The Voronoi diagram of N points in the plane can be constructed in $O(N \log N)$ time.*

Proof: This result is proved in reference 7 and we give only a sketch here. Suppose that the set S of N points is divided into two subsets L and R, each containing N/2 points, such that every point of L lies to the left of every point of R. Assume that we already possess the Voronoi diagrams V(L) and V(R) of L and R separately. If these can be merged in linear time to form the diagram V(S) of the entire set, then splitting the problem recursively will give an $O(N \log N)$ algorithm.

The merge procedure is quite simple. There exists a polygonal line *P* with the property that any point to the left of *P* is closest to some point of L and any point to the right of *P* is closest to some point of R. The locus of points that are closer to some point of L than to any point of R is just the union of Voronoi polygons (in V(S)) of points of L. Similarly, the union of the remaining polygons is the locus of points closer to some point of R. *P* is the collection of Voronoi edges shared by a polygon of L and a polygon of R.

Figure 6 shows V(S) and the dividing line P (solid).

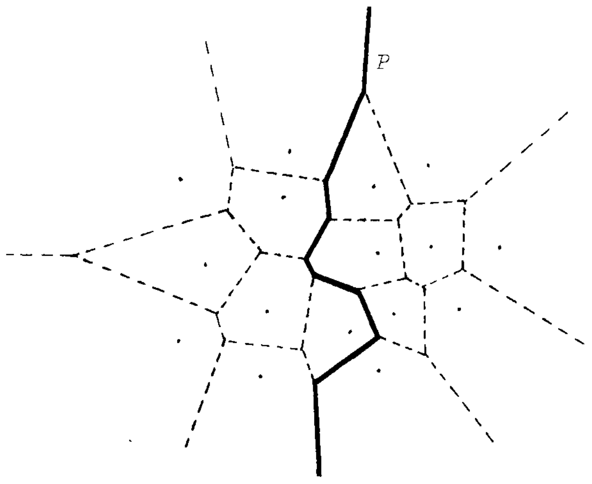

Figure 6. The polygonal line P divides L and R.

The polygonal line separating two sets that are disjoint in x is monotonic in y, that is, each horizontal line intersects P in exactly one point[7]. This fact enables P to be constructed in a single scan of the Voronoi diagrams V(L) and V(R). Once P is formed, the merge procedure is practically finished. If a point z lies to the left of P, it is closest to some point of L. Thus no edges of V(R) that appear to the left of P are present in the final diagram V(S). Similarly, the edges of V(L) that lie to the right of P are also absent. Figures 7, 8, 9, and 10 illustrate the relationships among V(L), V(R), P, and V(S) in detail. In figure 9 the points are numbered by increasing x-coordinate. To form V(S) it is only necessary to delete the irrelevant edges of V(L) and V(R) determined by P.

The dividing line P consists of a ray, some line segments, and another ray. A ray of V(S) is the perpendicular bisector of a pair of consecutive vertices of the convex hull of S. The hull of S is the hull of L $\cup$ R and forming the hull of the union of two disjoint convex polygons results in the creation of two new hull edges (Figure 11), which can be found in linear time[7]. These new edges join a point of L to a point of R. In the example, the edges occur between points 4 and 10 and 7 and 13. The perpendicular bisectors of the new edges are the rays of P.

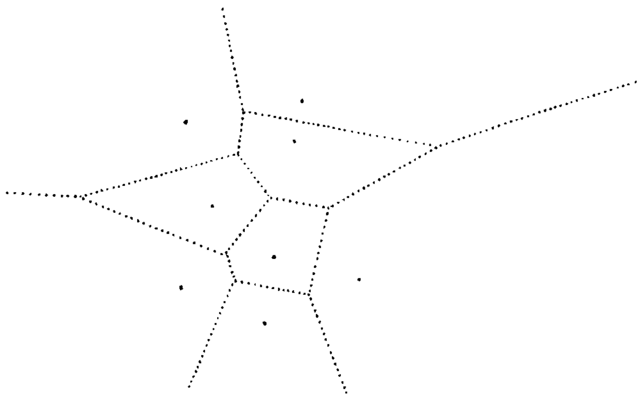

Figure 7. V(L)

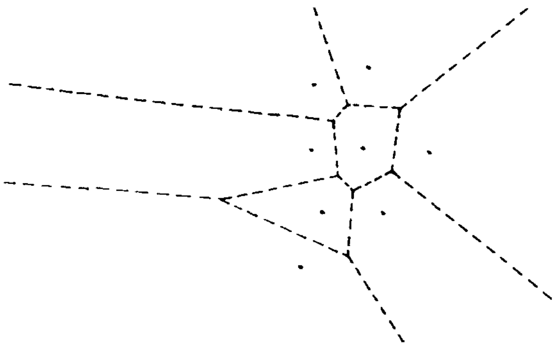

Figure 8. V(R)

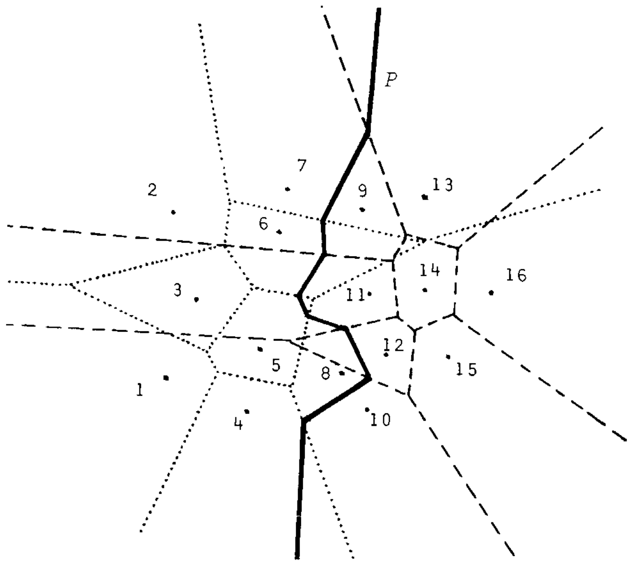

Figure 9. V(L), V(R) and P superimposed.

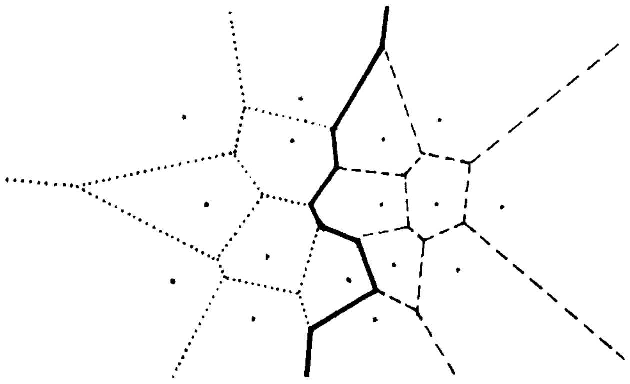

Figure 10. Final Voronoi diagram

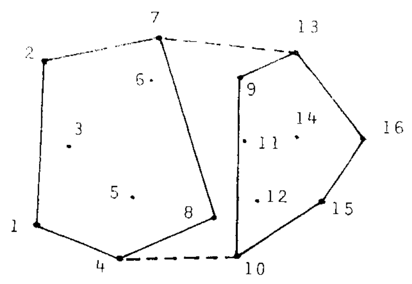

Figure 11.
The hull of the union of disjoint convex polygons.

We can imagine forming P by moving a point p inward from infinity along one of the rays. Suppose we begin on the 4,10 bisector. Initially, p lies the Voronoi polygons V(4) and V(10) and proceeds along the locus of points equidistant from 4 and 10 until it becomes closer to a different point. This occurs when p hits the edge of one of the polygons. Referring to figure 9, as p moves upward it hits the edge shared by V(4) and V(8) before crossing any edge of V(10). At this point, p is closer to 8 than to 4 and it must continue along the 8,10 bisector. Moving further, p crosses an edge of V(10), becomes closer to 12 than to 10, and moves off along the 8,12 bisector. The path of p zig-zags upward until it reaches the 7,13 bisector, at which point it has traced out the desired polygonal line P.

We will now show how to form P by scanning the diagrams V(L) and V(R) once each, with no backtracking. Every Voronoi polygon is convex and is divided by its lowest vertex l and its highest vertex h into two chains of segments that are monotonic in y. (Figure 12)

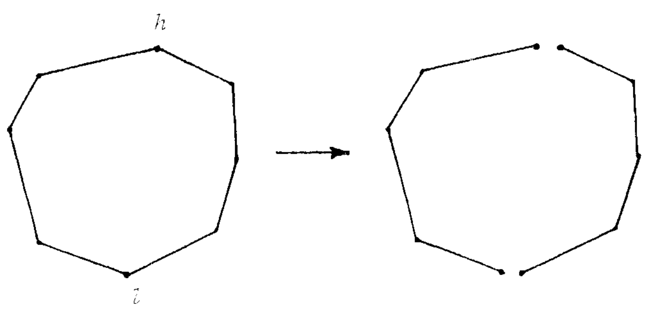

Figure 12.
Partition of a convex polygon into monotonic chains.

Since, at any instant, p lies in two Voronoi polygons, one of V(L) and one of V(R), we must find the lowest segment that p will intersect on the four resulting monotonic chains, if p continues in its current direction To determine this it is only necessary to examine the chains in round-robin fashion, ever ascending, in a manner reminiscent of merging four ordered lists. Every time a new Voronoi edge is examined, it is either discarded (p did not intersect it) or a new segment of P is begun. Since P, V(L) and V(R) together have only O(N) edges, linear time suffices to construct P, and recursion yields an O(N log N) algorithm. □

Theorem 5. *O(N log N) is a lower bound on the time required to construct a Voronoi diagram on N points.*

Proof: We show that O(N log N) time is required just to construct a single Voronoi polygon! Any Voronoi diagram algorithm must be able to sort: consider N real numbers $x_1, \dots, x_N$ and map them stereographically onto the unit circle in O(N) time. Now find the Voronoi diagram of the N points and the origin. The Voronoi polygon of the origin has edges that are in one-to-one correspondence with the given numbers, in sorted order. □

This implies that the algorithm of Thm. 4 is asymptotically optimal.

IV. Applying the Voronoi structure.

For each of the closest-point problems we are going to exploit the Voronoi diagram so that an easy subproblem results.

Theorem 6. *The all closest points problem can be solved in O(N log N) time.*

Proof: The perpendicular bisector of any point p_i and its nearest neighbor p_j coincides with an edge of the Voronoi polygon V(i). It thus suffices to examine each Voronoi polygon V(i) once and find the edge closest to p_i for all i. Each Voronoi edge occurs in exactly two polygons, so this procedure never encounters an edge more than twice. Since there are only O(N) Voronoi edges, linear time suffices once the diagram is available. By Thm. 4 the Voronoi diagram can be constructed in O(N log N) time. □

Theorem 7. *A Euclidean minimum spanning tree on N points in the plane can be constructed in O(N log N) time.*

Proof: Consider the straight-line dual of V(S), that is, join p_i and p_j by a line segment if and only if V(i) and V(j) share an edge. The result is a planar graph on the original N points and hence has O(N) edges. We will show that a minimum spanning tree must also be a minimum spanning tree of the dual graph.

Recall Prim's algorithm for a minimum spanning tree of a graph with arbitrary edge weights[10]: begin with all vertices unlabeled and select a starting vertex *s*. Label *s* and insert an edge between *s* and its nearest unlabeled neighbor, *t*, and label *t*. At each subsequent step, add the shortest edge that joins a labeled and an unlabeled vertex. Ties may be resolved arbitrarily and the algorithm terminates when all vertices are labeled.

Applying the above procedure to the Euclidean case, begin with any point *s*. Its nearest neighbor *t* is determined by some edge of V(*s*), and the tree link *st* to be added is dual to that edge. We now seek a point *u* that is closer to *s* or *t* than any other point is. But *u* is determined by some edge of V(*s*) ∪ V(*t*), so *su* or *tu*, whichever is added, will be an edge of the dual. At each step the link to be added is determined by the union of the Voronoi polygons of vertices already labeled and hence is an edge of the dual.

The dual can be constructed in O(N) time, if V(S) is available, by connecting with a straight line segment the two points that determine each edge of the Voronoi diagram. The dual is a graph with N vertices and O(N) edges, for which a minimum spanning tree can be found in O(N log log N) time[26]. The construction of V(S) thus dominates the computation and O(N log N) time suffices.□ (see Figure 13). We note in passing that no two edges

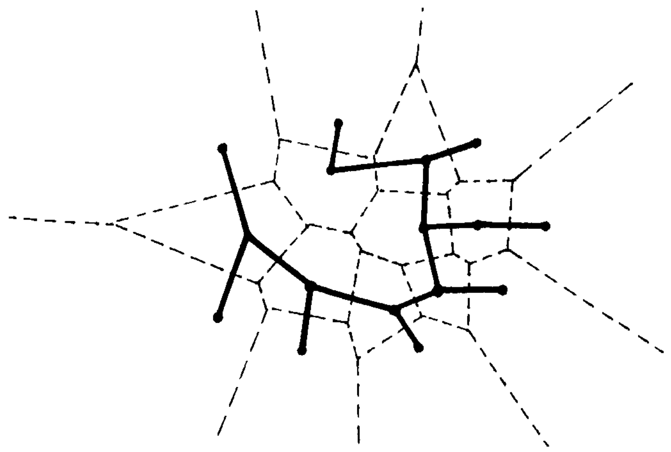
Figure 13. Relation between the Voronoi diagram and a Euclidean minimum spanning tree.

of a Euclidean minimum spanning tree intersect and such a tree has maximum degree five[24].

Theorem 8. *A Euclidean traveling salesman tour that is not longer than twice the length of an optimal tour can be found in O(N log N) time.*

Proof: The algorithm is based on an observation by Nick Pippenger that the Euclidean minimum spanning tree can be used to obtain an approximate traveling salesman tour. Let OPT denote the length of an optimal tour and let MST be the total length of a Euclidean minimum spanning tree. Removal of any edge of the optimal tour leaves a spanning tree, so OPT > MST. By traversing the minimum spanning tree twice (as in Figure 14) we produce a traveling salesman tour of length 2•MST < 2•OPT. □

Finding a similar tour in the complete graph with arbitrary edge weights requires $O(N^2)$ time and an algorithm which achieves this bound is known[27].

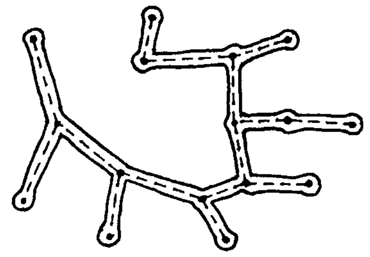
Figure 14. The Euclidean minimum spanning tree (- - -) provides an approximate traveling salesman tour (——).

Theorem 9. *A minimum-weight triangulation of N points can be found in O(N log N) time.*

Proof: By a theorem of Delaunay, the straight-line dual of a Voronoi diagram is a triangulation[28]. That it has minimal total length follows by analogy to Thm. 7 and the algorithm of Düppe and Gottschalk[14]. □

The minimum-weight triangulation is useful for interpolation because the circumcircle of every triangle contains no other points of the set (since the circumcenter is a Voronoi point).

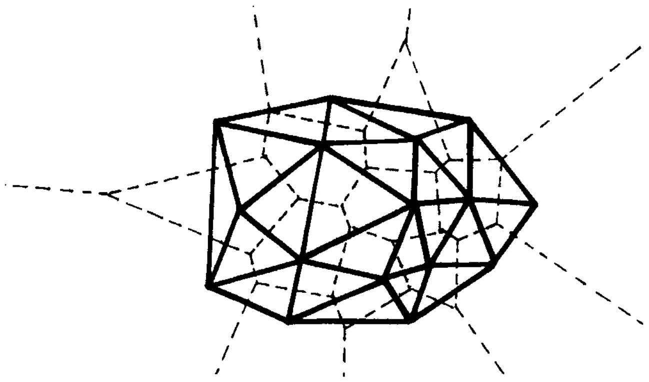

Figure 15. Relation between the Voronoi diagram and the minimum-weight triangulation.

Theorem 10. *The largest empty circle problem can be solved in O(N log N) time.*

Proof: The center of the required circle must lie at a Voronoi point or at the intersection of a Voronoi edge and the convex hull of the set. If the circle is determined by three points its center is at their Voronoi point (circumcenter). If the circle is determined by two points and an edge of the hull it lies on the perpendicular bisector of the points. (Figure 16) The circumradii associated with all the Voronoi points can be found in O(N) time, given V(S). We will show that all the intersections of Voronoi edges and convex hull edges can also be found in O(N) time. Given V(S), the convex hull of S can be found in linear time by Thm. 11. Consider any edge e of the hull. Corresponding to e is a ray r which coincides with the perpendicular bisector of e. Beginning with r, conduct a breadth-first search of the Voronoi diagram until all Voronoi edges are found which intersect e. Repeat this procedure beginning with each ray of the diagram. No Voronoi edge will be examined more than once, so linear time suffices. For each such intersection point, calculate the distance to either of the two points determining the Voronoi edge involved. The center of the circle is located by finding the point that yields the largest radius. Again, the running time is dominated by the time required to form the Voronoi diagram. □

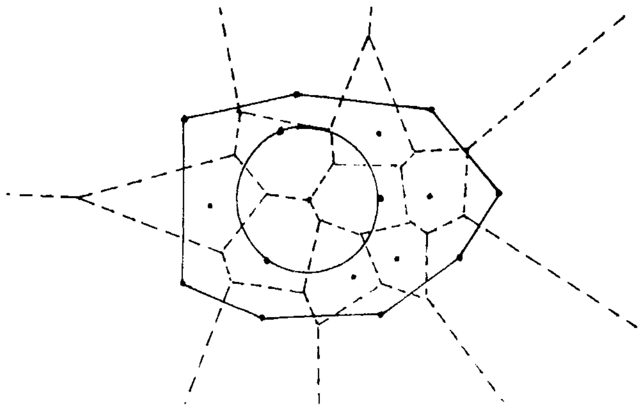

Figure 16. The Voronoi diagram locates a largest empty circle.

Theorem 11. *The convex hull of N points in the plane can be found in O(N log N) time.*

Proof: We show that O(N) time is sufficient, given the Voronoi diagram. Unbounded Voronoi polygons and vertices of the hull are in one-to-one correspondence. Begin by marking each polygon as being bounded or unbounded. This can be done by scanning each polygon once, in O(N) time. Begin with any unbounded polygon V(i) and examine its edges, looking for an edge that is shared with another unbounded polygon V(j). Points i and j are consecutive on the convex hull and we continue by searching V(j), proceeding point by point around the hull. No Voronoi edge is examined more than a constant number of times.□

V. Generalization of the Voronoi diagram.

The Voronoi diagram, while very powerful, has no means of dealing with farthest points, k closest points, and other distance relationships, and, as such, is unable to solve the remainder of the problems we have posed.

We have been considering the Voronoi polygon associated with a single point, but such a restriction is not necessary and it will be useful to speak of the Voronoi polygon V(H) of subset H of points, defined by

$$V(H) = \{x: \forall y\in H \ \forall z\notin H \ \ d(x,y) < d(x,z)\} .$$

That is, V(H) is the locus of points closer to some point in H than to any point not in H. An equivalent definition is: $V(H) = \{ \cap h(i,j): i\in H, j\in S-H \}$, where h(i,j) is again the half-plane containing i that is defined by the perpendicular bisector of i and j. This shows that the generalized Voronoi regions are convex polygons. It may, of course, occur that V(H) is empty. A set S of N points has 2^N subsets. How many of these possess non-empty Voronoi polygons? To answer this question we must probe the structure of the generalized polygons a little deeper.

We define the Voronoi diagram of order k, denoted $V_k(S)$, as the collection of all Voronoi polygons of k-subsets of S, so $V_k(S) = \{ \cup V(H): H\subset S, |H|=k \}$. It is proper to speak of $V_k(S)$ as a diagram because its polygons partition the plane. Given $V_k(S)$, the k points closest to a new given point x can be determined by finding the polygon of $V_k(S)$ in which x lies. Figure 17 shows the Voronoi diagram of order two associated with the set of sixteen points considered earlier. The relevant subsets H are shown in braces, where space permits.

We call a subset $H\subset S$ exposed iff H and S-H lie in complementary half-planes, that is, if H and S-H are separable.

Theorem 12. *The number of unbounded Voronoi polygons (of all orders) of a set of N points is N(N-1).*

Proof: We first show that V(H) is unbounded iff H is exposed and then count the number of exposed subsets. If H is exposed then H and S-H lie in complementary half-planes, which we may take without loss of generality to be the right and left half-planes, respectively. Points on the x-axis with sufficiently large coordinates are closer to some point of H than to any point of S-H, so V(H) must be unbounded. Conversely, if V(H) is unbounded it must contain a ray, which we may take to be the positive x-axis. Since now all points of H have greater x-coordinate than any point of S-H, H must be exposed.

Every line determined by two points of S gives rise to two exposed subsets. Let points a and b determine line l and suppose that l divides S into two subsets, L and R, neither of which contains a or b. The dividing line produces two partitions of S, $(\{L\cup a\},\{R\cup b\})$ and $(\{L\cup b\},\{R\cup a\})$, which define four exposed subsets, but each subset occurs in two partitions. The number of determined lines is N(N-1)/2, so the number of exposed subsets is 4N(N-1)/4 = N(N-1). □

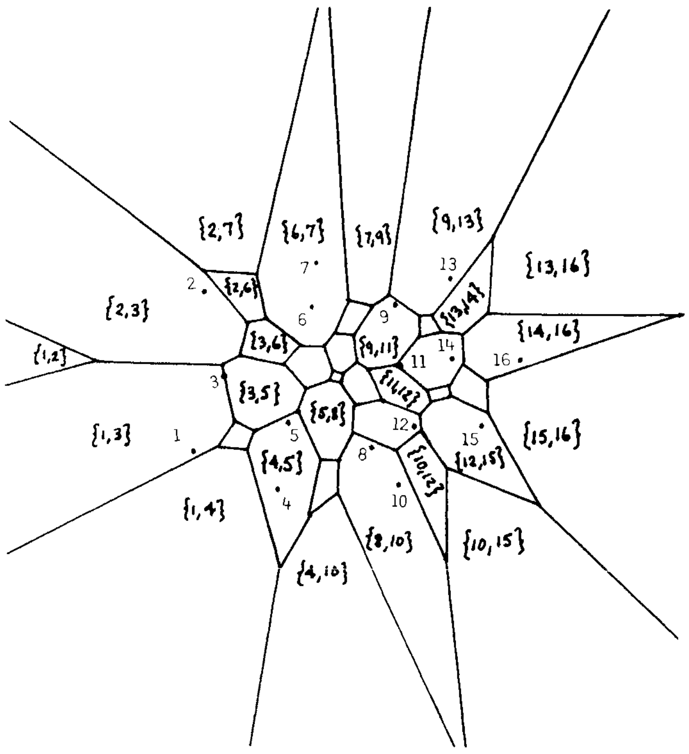

Figure 17. A Voronoi diagram of order two.

Theorem 13. *The number of bounded Voronoi polygons of (of all orders) of a set of N points is* $\binom{N-1}{3}$.

Proof: Any vertex of a generalized Voronoi polygon is a circumcenter of some triple of points. Let x be a Voronoi vertex which is the circumcenter of points a, b, and c and let H be the set of points of S that are closer to x than to any of $\{a,b,c\}$. (H may be empty) Let $k = |H|$. Now, x appears in the order k+1 Voronoi diagram as the meeting point of $V(H \cup a)$, $V(H \cup b)$, and $V(H \cup c)$ and in the order k+2 diagram as the meeting point of $V(H \cup a \cup b)$, $V(H \cup b \cup c)$, and $V(H \cup c \cup a)$. Except for degeneracies which do not affect the order of the result, every Voronoi vertex, being the circumcenter of three points, has degree three, except the vertex at infinity where the rays meet. If a planar graph has f faces, v vertices of degree three, and one vertex of degree i, then $2f = 2 + v + i$. Letting f_k denote the number of non-empty Voronoi polygons in the order k diagram, v_k the number of Voronoi vertices, and i_k the number of unbounded regions and summing over k, we have

$$2\sum_{k=1}^{N-1} f_k = 2N - 2 + \sum_{k=1}^{N-1} v_k + \sum_{k=1}^{N-1} i_k . \quad \text{But}$$

$$\sum_{k=1}^{N-1} i_k = 2\binom{N}{2} \quad \text{and} \quad \sum_{k=1}^{N-1} v_k = 2\binom{N}{3} , \text{ so}$$

we obtain

$$\sum_{k=1}^{N-1} f_k = N - 1 + \binom{N}{2} + \binom{N}{3} = 2\binom{N}{2} + \binom{N-1}{3} .$$

Since the number of unbounded polygons is $2\binom{N}{2}$, the result follows. □

Thus the total number of polygons in all the Voronoi diagrams is $O(N^3)$, not 2^N. The number of polygons in the order k diagram does not exceed $O(k(N-k))$ and a beautiful unifying result is that the union of the Voronoi polygons of all orders is precisely the set of perpendicular bisectors of all pairs of points.

The generalized Voronoi diagram places closest-point and farthest-point problems on an equal footing since the locus of points whose k nearest neighbors are the set H is also the locus of points whose N-k farthest neighbors are the set S-H. Thus the order k closest-point diagram is and order N-k farthest-point diagram.

Theorem 14. *The smallest circle enclosing a set of N points in the plane can be found in O(N log N) time.*

Proof: The required circle is determined by two or three points of the set[22]. If it is determined by two points then these two points are the ends of a diameter of the circle and hence are the two farthest points of S. If the circle is determined by three points, a, b, and c, then its center is interior to triangle abc.

Consider the farthest-point Voronoi diagram (that is, the diagram of order N-1). Associated with each point i is a convex polygonal region $V_{N-1}(i)$ such that i is the farthest neighbor of every point in the region. The order N-1 diagram is determined only by points on the convex hull of S, which are all exposed, so $V_{N-1}(S)$ has no bounded regions. An example is given in Figure 18.

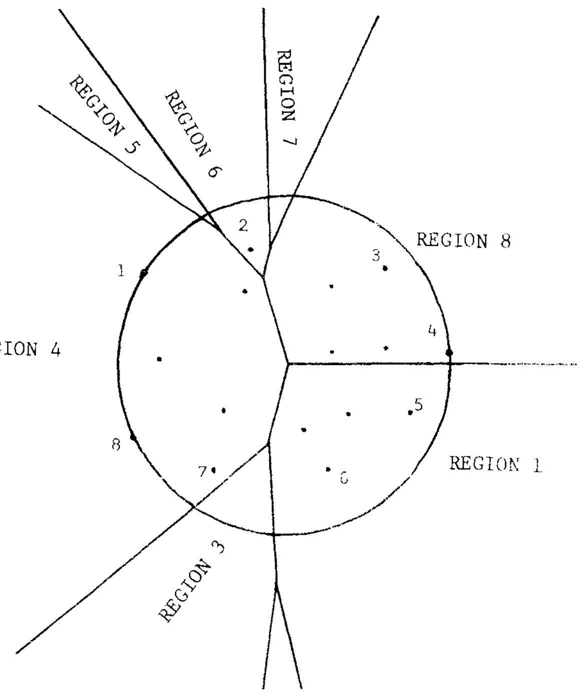

Figure 18. A farthest-point Voronoi diagram.

The farthest-point diagram can be constructed in O(N log N) time by a procedure analogous to the closest-point algorithm. The dividing lines P are identical. It is only necessary to discard those edges of $V_{N-1}(L)$ that lie to the left of P and similarly for $V_{N-1}(R)$. Given $V_{N-1}(S)$, the two farthest points of S can be found in O(N) time just by examining each edge of the diagram and computing the distance between the points determining it. The greatest distance thus obtained is the diameter of S. If the circle determined by the two farthest points encloses the set, we are done. Otherwise, we claim that the center of the smallest enclosing circle is a vertex of the farthest-point Voronoi diagram. For let the circle pass through points a, b and c. Then the center is the point common to $V_{N-1}(a)$, $V_{N-1}(b)$ and $V_{N-1}(c)$! There are only O(N) vertices in $V_{N-1}(S)$, so all the circumradii can be found in O(N) time, once the diagram is constructed. □

In figure 18 the smallest enclosing circle is determined by points 1, 8, and 4 and their circumcenter is, as expected, a vertex of the diagram.

Theorem 15. *With preprocessing allowed of a set of N points in the plane, the k points closest to a new given point can be found in O(max(k, log N)) time.*

Proof: The preprocessing consists of forming the order k Voronoi diagram and constructing slabs through the Voronoi points as in reference 6, where it is shown that the face of a straight-line planar graph on M vertices in which a new point x lies can be found in O(log M) time. In this case, $M = O(k(N-k))$, so $\log M = O(\log N)$. To determine the k points closest to x it is only necessary to find the polygon of $V_k(S)$ in which x lies. Writing out the answer requires $O(k)$ time, and the result follows. $O(k^2(N-k)^2)$ storage suffices. □

VI. Summary

In this paper we have reduced the best known upper bounds on six problems (all except P4) by exploiting a single, compact geometric structure to yield algorithms that are optimal to within a constant factor. Furthermore, all of the algorithms except for one involving preprocessing run in linear space. A striking feature of geometric problems is the extent to which naive algorithms may be improved if one strives to make use of the geometric properties of the problems directly. Computational geometry provides an opportunity to combine geometric insight with contemporary fast-algorithm techniques.

VII Acknowledgements.

We wish to express our appreciation to Professor Stanley Eisenstat of Yale for his assistance on many troublesome points, to Ray Strong of IBM Yorktown, who first suggested a method for finding the two closest of N points in less than quadratic time, and to the first author's wife, Julie, who created an atmosphere remarkably conducive to scientific research.

References

[1] Breuer, M.A., ed. Design Automation of Digital Systems, volume 1. Prentice-Hall(1972), 420 pp.

[2] Francis, R.L. and White, J.A. Facility Layout and Location. Prentice-Hall(1974), 468 pp.

[3] Saaty, T.L. Optimization in Integers and Related Extremal Problems. McGraw-Hill(1970), 295 pp.

[4] Hartigan, J.A. Clustering Algorithms. Wiley Series in Probability and Mathematical Statistics (1975), 351 pp.

[5] Davis, J.C. and McCullagh, M.J., eds. Display and Analysis of Spatial Data. Wiley(1975), 378 pp.

[6] Shamos, M.I. *Geometric Complexity.* Proceedings of the Seventh ACM Symposium on the Theory of Computing. May, 1975.

[7] Shamos, M.I. *Computational Geometry.* Ph.D. Thesis, Yale University, 1975.

[8] Dobkin, D. and Lipton, R. *On the Complexity of Computations under Varying Stes of Primitives.* Yale Univ. Dept. of Comp. Sci. Technical Report #42, 1975.

[9] Kruskal, J.B. *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem.* Proc. AMS 7(1956), 48-50.

[10] Prim, R.C. *Shortest Connection Networks and Some Generalizations.* Bell Sys. Tech. J. 36(1957), 1389-1401.

[11] Zahn, C.T. *Graph-Theoretical Methods for Describing Gestalt Clusters.* IEEE Trans. Comp., C-20(1971), 68-86.

[12] Aho, A.V., Hopcroft, J.E. and Ullman, J.D. The Design and Analysis of Computer Algorithms. Addison-Wesley(1974), 470 pp.

[13] reference 5, pp. 352-367.

[14] Düppe, R.D. and Gottschalk, H.J. *Automatische Interpolation von Isolinien bei willkürlich stützpunkten.* Allgemeine Vermessungsnachrichten 77(1970), 423-426.

15 Graham, R.L. *An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set.* Info. Proc. Lett. 1(1972), 132-133.

16 reference 2, chapter 9.

17 Dasarathy, B. and White, L.J. *Some Maximin Location and Classifier Problems: Theory and Algorithms.* Talk presented at the ACM Computer Science Conference, February, 1975. Washington, D.C.

18 Blum, M. et al. *Time Bounds for Selection.* J. Comp. and Sys. Sci. 7(1973), 448-461.

19 Friedman, J.H., Baskett, F. and Shustek, L.J. *A Relatively Efficient Algorithm for Finding Nearest Neighbors.* Stanford Linear Accelerator Tech. Report SLAC-PUB-1448 CS-445(1974).

20 Toregas, C, Swain, R., ReVelle, C., and Bergman, L. *The Location of Emergency Service Facilities.* Op. Res. 19(1971), 1363-1373.

21 Nair, K.P.K. and Chandràsekaran, R. *Optimal Location of a Single Service Center of Certain Types.* Nav. Res. Log. Quart. 18(1971).

22 Rademacher, H. and Toeplitz, O. The Enjoyment of Mathematics. Ch. 16: *The Spanning Circle of a Finite Set of Points.* Princeton Univ. Press(1957).

23 Elzinga, J. and Hearn, D.W. *Geometrical Solutions for some Minimax Location Problems.* Transportation Science 6(1972), 379-394.

24 Shamos, M.I. Problems in Computational Geometry. (1974). unpublished manuscript.

25 Rogers, C.A. Packing and Covering. Cambridge Univ. Press(1964), 111 pp.

26 Yao, A. *An $O(|E|$ log log $|V|)$ Algorithm for Minimum Spanning Trees.* Univ. of Illinois Tech. Report #UIUCDCS-R-74-691(1974).

27 Rosenkrantz, D.J., Stearns, R.E. and Lewis, P.M. *Approximate Algorithms for the Traveling Salesperson Problem.* Proc. Fifteenth IEEE Symposium on Switching and Automata Theory.

28 Delaunay, B. *Sur la sphère vide.* Bull. Acad. Sci. USSR(VII), Classe Sci. Mat. Nat. (1934), 793-800.