JOURNAL OF ALGORITHMS 12, 38–56 (1991)

Finding k Points with Minimum Diameter and Related Problems

ALOK AGGARWAL

IBM T. J. Watson Research Center, Yorktown Heights, New York 10598

HIROSHI IMAI

Kyushu University, Japan

NAOKI KATOH

Kobe University of Commerce, Japan

AND

SUBHASH SURI

Bell Communications Research, Morristown, New Jersey 07960

Received April 27, 1989

Let S be a set consisting of n points in the plane. We consider the problem of finding k points of S that form a "small" set under some given measure, and present efficient algorithms for several natural measures including the diameter and the variance. © 1991 Academic Press, Inc.

1. INTRODUCTION

We consider the problem of selecting a specified number of points, k, from a given set S, subject to some optimization criterion. Problems of this type often arise in statistical clustering and pattern recognition (see Andrews [3] and Hartigan [7]). From an algorithmic standpoint, these problems usually can be solved in time $O(n^{k+c})$, where n is the number of points in S and c is a small constant; for arbitrary k, this time complexity is exponential in the size of the input. Finding general methods to solve this problem more efficiently for a wide variety of optimization criteria is a challenging and elusive goal, and most researchers have concentrated on fixed values of k. A notable exception is the paper by Dobkin, Drysdale, and Guibas [6], where some general methods for finding smallest polygons are developed. Among the results presented in [6] are algorithms for finding (a) the smallest perimeter convex polygon containing at least k points from S in time $O(k^2 n \log n + k^5 n)$; (b) the smallest perimeter k-gon with vertices in S in time $O(k^4 \log n + (ck)^{2k+1} n)$, with c a constant; (c) the convex polygon containing at least k points from S, with vertices in S, and such that its longest side is minimized in time $O(k^6 n \log n + k^{13} n)$, and the k-gon with vertices from S whose longest side is minimized in time $O(k^6 n \log n + (ck)^{3k+1} n)$, with c a constant. In this paper, we follow the lead of [6] and study the general case of the problem for several natural criteria of optimization. In particular, we give efficient algorithms for the following problems:

1. (*k-diameter*) Find a set of k points with a minimum diameter; the *diameter* of a set is the maximum distance between any two points of the set. Our algorithm takes $O(k^{2.5} n \log k + n \log n)$ time.

2. (*k-variance*) Find a set of k points with a minimum variance; the *variance* of a set is the sum of squares of the distances of all pairs of points in the set divided by the number of points in the set. Our algorithm takes $O(k^2 n + n \log n)$ time.

3. (*Smallest square*) Find a set of k points such that the perimeter of its enclosing axes-parallel square is minimized. We solve this problem in time $O(k^2 n \log n)$.

4. (*Smallest rectangle*) Find a set of k points such that the perimeter of its enclosing axes-parallel rectangle is minimized. We solve this problem in time $O(k^2 n \log n)$.

Diameter and variance are often used as essential features of a point set in cluster analysis and pattern recognition; see, for instance, Andrews [3], Asano *et. al.* [5], and Hartigan [7]. In the context of pattern recognition, finding a subpattern with a given feature is a kind of template matching. The problems (3) and (4) may be regarded as generalizations of the problem of finding a largest (perimeter-wise) empty rectangle in a set of points (see, e.g., Aggarwal and Suri [2] and McKenna, O'Rourke, and Suri [11]). It is easy to obtain polynomial time algorithms for problems (3) and (4), since the smallest enclosing square or rectangle is ordinarily determined by at most four points of the set. The polynomial-time solvability of the remaining problems, however, is not clear at first glance.

The general idea behind our algorithms is the use of higher-order Voronoi diagrams. This approach is a fairly natural one, and it was adopted by Dobkin, Drysdale, and Guibas [6] for finding a smallest convex polygon containing at least k points. In deriving our algorithms, we add several new features to this basic approach, and also establish new combinatorial facts about the higher-order Voronoi diagrams, which may have independent interest. For instance, the techniques developed in our paper also lead to a polynomial-time algorithm for computing a maximum clique of the intersection graph of n unit circles in the plane; the algorithm has running time $O(n^{3.5} \log n)$.

Our paper is organized in five sections. Section 2 presents an algorithm for the k-variance problem. Section 3 discusses the k-diameter problem. Section 4 presents algorithms for the smallest enclosing square and rectangle. Conclusions are offered in Section 5.

2. The k-Variance Problem

Given a set S of n points in the plane, we wish to select k points from S with a minimum variance, where the variance of a set is defined as the sum of squares of the pairwise distances divided by the number of points in the set. We show that this problem can be solved efficiently by using the order k Voronoi diagram. We begin by deriving a useful inequality for the variance.

Let S_k^* be a set of k points with a minimum variance, and let $c^* \equiv (x^*, y^*)$ be the centroid of S_k^*, i.e.,

$$x^* = \frac{1}{k} \sum_{p_i \in S_k^*} x_i \qquad \text{and } y^* = \frac{1}{k} \sum_{p_i \in S_k^*} y_i,$$

where x_i and y_i are the coordinates of p_i. Then we have the relations

$$\begin{aligned}
\operatorname{Var}(S_k^*) &= \frac{1}{k} \sum_{p_i, p_j \in S_k^*, i<j} \left((x_i - x_j)^2 + (y_i - y_j)^2 \right) \\
&= \frac{1}{k} \Bigg(\sum_{p_i \in S_k^*} (k-1) x_i^2 - \sum_{p_i, p_j \in S_k^*, i<j} 2 x_i x_j + \sum_{p_i \in S_k^*} (k-1) y_i^2 \\
&\qquad - \sum_{p_i, p_j \in S_k^*, i<j} 2 y_i y_j \Bigg) \\
&= \frac{1}{k} \Bigg(\sum_{p_i \in S_k^*} k x_i^2 - \sum_{p_i \in S_k^*} x_i^2 - \sum_{p_i, p_j \in S_k^*, i<j} 2 x_i x_j + \sum_{p_i \in S_k^*} k y_i^2 \\
&\qquad - \sum_{p_i \in S_k^*} y_i^2 - \sum_{p_i, p_j \in S_k^*, i<j} 2 y_i y_j \Bigg) \\
&= \sum_{p_i \in S_k^*} (x_i^2 + y_i^2) - k(x^*)^2 - k(y^*)^2 && (1) \\
&= \sum_{p_i \in S_k^*} \left((x_i - x^*)^2 + (y_i - y^*)^2 \right) && (2) \\
&< \sum_{p_i \in S_k^*} \left((x_i - x)^2 + (y_i - y)^2 \right), && (3)
\end{aligned}$$

for any $(x, y) \neq (x^*, y^*)$. The last inequality follows from the fact that the function

$$f(x, y) = \sum_{p_i \in S_k^*} \left((x_i - x)^2 + (y_i - y)^2\right)$$

has a unique minimum, which occurs at (x^*, y^*). We can now prove the following lemma.

LEMMA 2.1. *Let $S_k^* \subseteq S$ be a set of k points with a minimum variance, and let $c^* \equiv (x^*, y^*)$ be the centroid of S_k^*. Then, the k nearest neighbors of c^* in S are precisely the points of S_k^*. Consequently, in the order k Voronoi diagram of S, there is a nonempty region associated with S_k^*.*

Proof. Let p_j be the farthest point from c^* in S_k^*. Suppose, for a contradiction, that there exists a point $p_l \in S - S_k^*$ that is no farther from c^* than p_j. Define $S_k' = (S_k^* - \{p_j\}) \cup \{p_l\}$. Then using Eq. (2), we have

$$\begin{aligned}\mathrm{Var}(S_k^*) &= \sum_{p_i \in S_k^*} \left((x_i - x^*)^2 + (y_i - y^*)^2\right)\\ &\geq \sum_{p_i \in S_k'} \left((x_i - x^*)^2 + (y_i - y^*)^2\right)\\ &> \mathrm{Var}(S_k'),\end{aligned}$$

since $c^* \equiv (x^*, y^*)$ is not the centroid of S_k' (cf. inequality (3)). But this contradicts our assumption that S_k^* has the minimum variance. □

Lemma 2.1 clearly holds in arbitrary dimensions. We therefore have the following simple algorithm for solving the k-variance problem: construct the kth order Voronoi diagram of S; compute the variance of sets of k points associated with the regions of the Voronoi diagram; and select the set with a minimum variance. Given the kth order Voronoi diagram of S, we can compute the variance of various sets of points in constant time apiece, by using the relation (1). More precisely, let A and B be two neighboring regions in the Voronoi diagram. Then the sets of points associated with them, namely, $S \cap A$ and $S \cap B$, differ by exactly one point. Given the centroid of $S \cap A$, the centroid of $S \cap B$ can be easily computed in $O(1)$ time; consequently the relation (1) allows the variance of $S \cap B$ to be determined from the variance of $S \cap A$ in constant time. The running time of our algorithm for solving the k-variance problem is therefore dominated by the time for constructing the kth order Voronoi diagram. In particular, we have the following theorem.

THEOREM 2.1. *Given a set S of n points in a d-dimensional space, we can solve the k-variance problem for S in time $O(f_d(n, k))$, where $f_d(n, k)$ is the* *time complexity of computing the kth order Voronoi diagram of n points in d dimensions.*

The time bounds for $f_d(n, k)$ are $f_2(n, k) = O(k^2 n + n \log n)$ [1] and $f_d(n, k) = O(n^{d+1})$ for $d \geq 3$ [13].

3. The k-Diameter Problem

The k-diameter problems asks for a set of k points having a minimum diameter. In this section, we present an $O(k^{2.5} n \log k + n \log n)$ time algorithm to solve this problem. In Subsection 3.1, we show a connection between the k-diameter problem and the $(3k - 3)$th order Voronoi diagram of S. In Subsection 3.2, we reduce this problem to the problem of finding a maximum independent set in a certain bipartite graph, and present an $O(k^{4.5} n \log k + n \log n)$ time algorithm for the problem. This bound is subsequently improved in Subsection 3.3 by exercising greater care in our use of the Voronoi diagram; in particular, this involves proving some new combinational bounds on the number of distinct pairs and triples whose points belong to the same region of a higher order Voronoi diagram.

3.1. *The k-Diameter Problem and Higher Order Voronoi Diagrams*

Motivated by our success with the kth order Voronoi diagram in the k-variance problem, we are tempted to use the same idea for the present problem as well. Unfortunately, the following example shows that the kth order Voronoi diagram may not always help in solving the k-diameter problem. Given a regular hexagon $abcdef$, let $S(x)$ be a set of m points contained in a ball of radius ε centered at x, for $x \in \{a, c, e\}$, where $\varepsilon > 0$ is a suitably small number. Similarly, let $S'(x)$ be a set of $m - 1$ points lying inside the triangle Δbdf at distance 2ε from x, for $x \in \{b, d, f\}$. Figure 1 illustrates this construction.

It is easily seen that $S_k \equiv S(a) \cup S(c) \cup S(e)$ is the unique set of $k \equiv 3m$ points having a minimum diameter but there is no nonempty region associated with S_k in the order k Voronoi diagram of the given set of points; this follows because any point in the plane is closer to at least one point of $S'(b) \cup S'(d) \cup S'(f)$ than it is to all of S_k.

Consequently, if there is a relationship between the k-diameter problem and the Voronoi diagram, then it must exist only for orders higher than k. Fortunately, Lemma 3.1 below shows that we need not look beyond diagrams of order $(3k - 3)$. We introduce some notation.

The mth order Voronoi diagram of S is denoted by $V_m(S)$; recall that this diagram is a partition of the plane into convex regions, where each

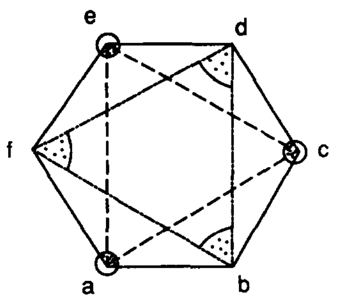

FIGURE 1

region is associated with a set of m points of S such that, for any point x in the region, the m nearest neighbors of x in S are precisely these m points. The set of points of S associated with a region of $V_m(S)$ is called a *Voronoi set* of $V_m(S)$. Observe that a Voronoi set of $V_m(S)$ has cardinality m and there are $O(mn)$ such sets (see, e.g., Lee [10]). We now prove the key lemma of this section.

LEMMA 3.1. *Let $S_k \subseteq S$ be a set of k points with a minimum diameter, for $k \leq n/3$. Then S_k is contained in some Voronoi set of $V_{3k-3}(S)$.*

Proof. A set of points T is a Voronoi set of $V_{|T|}(S)$ if and only if there exists a circle C containing T and no other point of S. To prove the lemma, we exhibit such a set $T \subseteq S$ of size $|T| \leq 3k - 3$ such that $S_k \subseteq T$. Let C be the minimum enclosing circle of S_k. First consider the case where C is determined by two points of S_k, say, p and q. In this case C cannot contain any point of $S - S_k$ in its interior; if there were a point $s \in S - S_k$ lying in C, we could replace p or q by s to obtain another set of k points whose diameter is strictly smaller than the diameter of S_k. In this case, S_k is obviously contained in a Voronoi set of $V_m(S)$ (in particular, the Voronoi set associated with the Voronoi region containing the center of the circle C). So let us now assume that C is determined by three points of S_k. We divide the circle into three sectors by adding line segments from the center to the three points determining C. See Fig. 2. Observe that the diameter of each of the three sectors is less than the diameter of S_k. If C contains more than $3k - 2$ points of S, then one of the sectors has at least k points, which gives a set of k points with a smaller diameter than S_k. Thus C has at most $3k - 3$ points. □

By Lemma 3.1, we can solve the k-diameter problem by examining all $\binom{3k-3}{k}$ subsets of each of the $O(kn)$ Voronoi sets of $V_{3k-3}(S)$. This time complexity, although polynomial in n, is still exponential in k. (We remark

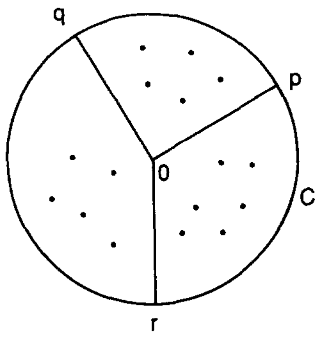

FIGURE 2

that Lemma 3.1, and the associated algorithm, generalizes to arbitrary dimensions. In dimension d, S_k is contained in some Voronoi set of $V_{(d+1)(k-1)}(S)$.) In the following subsection, we develop a fully polynomial-time algorithm for the problem.

3.2. *A Polynomial Time Algorithm*

Given a nonnegative number d, let $G(d)$ be the graph defined as follows. The vertices of $G(d)$ are the points of S and there is an edge between two points, p_i and p_j, if $\|p_i - p_j\| \leq d$, where $\|\cdot\|$ denotes the Euclidean norm. We define the *lune* of p_i and p_j to be the intersection of two circles of radius $\|p_i - p_j\|$, one centered at p_i and the other at p_j. The graph $G(p_i, p_j)$ is then defined as the restriction of $G(\|p_i - p_j\|)$ to the lune of p_i and p_j: its vertices are the points of S that lie in the lune of p_i and p_j and two vertices are connected by an edge if their distance is at most $\|p_i - p_j\|$. Then the following observation is straightforward.

Observation 3.1. There is a set of k points in S whose diameter is determined by the points p_i and p_j if and only if there is a clique of size k in $G(p_i, p_j)$ that includes p_i and p_j.

Let $\overline{G}(p_i, p_j)$ be the complement graph of $G(p_i, p_j)$. Then a clique of size k in $G(p_i, p_j)$ corresponds to an independent set of the same size in $\overline{G}(p_i, p_j)$. In the following, we present an efficient algorithm for finding a maximum independent set of $\overline{G}(p_i, p_j)$. The following lemma is straightforward.

LEMMA 3.2. *The graph $\overline{G}(p_i, p_j)$ is bipartite.*

Proof. Consider the two half-lunes obtained from the lune of p_i and p_j by drawing the line segment from p_i to p_j. Points in either half-lune form an independent set of $\overline{G}(p_i, p_j)$, since the diameter of each half-lune is at most $\|p_i - p_j\|$. The points lying on opposite sides of the line from p_i to p_j form a bipartition of the vertices of $\overline{G}(p_i, p_j)$. □

The following lemma describes an efficient procedure for finding a maximum independent set of $\overline{G}(p_i, p_j)$. (An alternative method is to directly employ a bipartite matching algorithm. This would take $O(n^{2.5} \log n)$ time using Vaidya's algorithm [12].)

LEMMA 3.3. *A maximum independent set of the graph $\overline{G}(p_i, p_j)$ can be found in $O(n^{1.5} \log n)$ time.*

Proof. If G is a bipartite graph on n vertices, then a maximum independent set of G can be found in time $O(n^{1.5}Q(n) + n^{0.5}P(n))$ (see Imai and Asano [9]), provided that the following two operations, F (find) and D (delete), can be dynamically implemented for G in time $O(Q(n))$ after $O(P(n))$ time preprocessing:

F. Given a vertex v, find an edge incident to v in the current graph;

D. Given a vertex v, delete v and all its incident edges from the graph.

In our graph $\overline{G}(p_i, p_j)$, two points have an edge between them if and only if their distance is greater than $\|p_i - p_j\|$. Thus, we need a dynamic data structure for storing a set of point S such that the following operations are efficiently implemented: (F) given a point v, find a point of S whose distance from v is greater than $\|p_i - p_j\|$, and (D) delete a point from S. We use a data structure, called *circular hull*, that was recently proposed by Hershberger and Suri [8]. This data structure needs $O(n \log n)$ time for construction and $O(n)$ space for storage, and it supports the above operations at an amortized cost of $O(\log n)$ each. Consequently, a maximum independent set of $\overline{G}(p_i, p_j)$ can be computed in time $O(n^{1.5} \log n)$ time and $O(n)$ space. □

Observe, however, that we really do not need to compute a maximum independent set; we only need to check if the size of the independent set is at least k. The following lemma shows that if the number of vertices in $\overline{G}(p_i, p_j)$ is at least $2k$, then there always exists an independent set of size k.

LEMMA 3.4. *If $\overline{G}(p_i, p_j)$ has at least $2k$ vertices, then there exists an independent set of size at least k, and it can be found in linear time.*

Proof. Consider the two half-lunes obtained from the lune of p_i and p_j by drawing the line segment from p_i to p_j. The set of points contained in the same half-lune forms an independent set. Since there are at least $2k$ points in the lune, one of the half-lunes must contain at least k points. □

Therefore, the problem of deciding whether $\overline{G}(p_i, p_j)$ contains an independent set of size k can be solved in time $O(n + k^{1.5} \log k)$. (In $O(n)$ time we count the number of points in the lune of p_i and p_j; if this number is at least $2k$, we have an independent set of size k, otherwise we use Lemma 3.3 to find a maximum independent set.) The k-diameter problem of S can be solved by repeating this process for each of the $O(n^2)$ pairs of S. This leads to the following theorem.

THEOREM 3.1. *The k-diameter problem for n points in the plane can be solved in* $O(n^3 + k^{1.5}n^2 \log k)$ *time and* $O(n)$ *space.* □

By combining Theorem 3.1 with Lemma 3.1, we obtain the following theorem.

THEOREM 3.2. *The k-diameter problem for n points in the plane can be solved in* $O(k^{4.5}n \log k + n \log n)$ *time and* $O(kn)$ *space.*

Proof. We compute the $(3k - 3)$th order Voronoi diagram of S; this takes $O(k^2n + n \log n)$ time. By lemma 3.1, we can solve the k-diameter problem of S by solving the problem for each of the Voronoi sets of $V_{3k-3}(S)$. Since there are $O(kn)$ Voronoi sets, each set has size $3k - 3$, and we can solve the problem for each Voronoi set in time $O(k^{3.5} \log k)$, the overall time complexity of our algorithm is

$$O(nk \cdot k^{3.5} \log k) = O(k^{4.5}n \log k),$$

plus the cost of computing the Voronoi diagram. This completes the proof. □

3.3. *Improving the Time Complexity*

Our final improvement to the time complexity comes from a more careful counting of the total number of pairs of points in all the Voronoi sets of $V_{3k-3}(S)$. Theorem 3.1 derives its bound by checking $O(nk^3)$ pairs of points; $\binom{3k-3}{2}$ pairs in each of the $O(nk)$ Voronoi sets. We show in this section that this bound is overly pessimistic and, in fact, we need to check only $O(nk)$ pairs. This fact will be proved in several stages. We begin with a definition.

Let $U_m(p_i)$ denote the union of the closed Voronoi regions of $V_m(S)$ whose Voronoi sets contain $p_i \in S$. The following observation is quite straightforward.

Observation 3.2. $U_m(p_i)$ is simply connected and it is star-shaped with respect to p_i.

We use b_{ij} to denote the perpendicular bisector of the line segment from p_i to p_j. Then we have the following key lemma.

LEMMA 3.5. *Let $m < n/2$; let $p_i, p_j \in S$ be two points such that $U_m(p_i) \cap U_m(p_j) \neq \varnothing$; and let U be a connected component of $U_m(p_i) \cap U_m(p_j)$. Then there is a Voronoi edge of $V_m(S)$ that is part of b_{ij} and that has a nonempty intersection with U.*

Proof. We prove the lemma in two parts: (a) the boundary of U intersects with b_{ij} and (b) this intersection point is contained in a Voronoi edge of $V_m(S)$. To prove (a), consider a point $p \in U$ and assume without loss of generality that p and p_j are on opposite sides of b_{ij}. Let q be the point where the line segment form p to p_j meets b_{ij}. Since $U_m(p_j)$ is star-shaped with respect to p_j, the line from p to p_j lies in U_j. Therefore, if we move a point x from p to p_j, then the m nearest neighbors of x always include p_j. For any position of x between p and q, p_i must also be one of the m nearest neighbors of x, since x is on the same side of b_{ij} as p_i. This implies that the segment from p to q is contained in $U_m(p_i)$, and consequently q is in the same component of $U_m(p_i) \cap U_m(p_j)$ as p, namely, U. Since q lies on b_{ij}, this shows that U intersects b_{ij}. To complete the proof of (a), it remains to prove that b_{ij} cannot be completely contained inside U. If b_{ij} were to lie entirely within U, the m nearest neighbors of *any* point $x \in b_{ij}$ must include both p_i and p_j. We consider the two extreme points of b_{ij}, namely, $x = +\infty$ and $x = -\infty$, and denote by H^+ (resp. H^-) the half-plane determined by p_i and p_j and containing $+\infty$ (resp. $-\infty$). If p_i and p_j are among the m nearest neighbors of both $x = +\infty$ and $x = -\infty$, then neither H^+ nor H^- may contain more than m points. But that is impossible since $H^+ \cup H^-$ is the entire plane and $m < n/2$. This completes the proof of (a).

To prove (b), let u be the point where b_{ij} intersects the boundary of U, and assume without loss of generality that u belongs to the boundary of $U_m(p_i)$. Let e be an edge of the Voronoi diagram $V_m(S)$ that contains u, and let p_i and p_l be the two points whose bisector contains e. If $l = j$, then the claim easily follows: e is the desired edge. Otherwise, u is equidistant from p_i, p_j, and p_l. We show that u is a Voronoi vertex of $V_m(S)$ determined by p_i, p_j, and p_l, as follows. Consider a point v close to the edge e but just outside the region $U_m(p_i)$. Since $e \subseteq b_{il}$, the set of m nearest neighbors of v is obtained by replacing p_i and p_l in the neighbor-set of u. That is, p_i is no longer one of the m nearest neighbors of v, but p_j is. This, however, cannot be the case for all positions of v along e: particular, if v is chosen on the same side of b_{ij} as p_i, then v is closer to p_i than it is to p_j. Consequently, there must be a discontinuity at the point u where the bisectors b_{il} and b_{ij} meet. Therefore, the point u is a Voronoi vertex of $V_m(S)$ determined by p_i, p_j, and p_l and there is a Voronoi edge incident to u that is part of b_{ij}. This completes the proof.

□

LEMMA 3.6. *The total number of distinct pairs (p, q) such that both p and q are contained in a Voronoi set of $V_m(S)$ is $O(mn)$.*

Proof. If $m \geq n/2$, then the lemma is trivial; there are only $O(n^2)$ distinct pairs. Otherwise, observe that if p and q are in a Voronoi set, then $U_m(p) \cap U_m(q) \neq \emptyset$. By lemma 3.5, the number of such pairs is bounded above by the number of edges in the Voronoi diagram $V_m(S)$, which is $O(mn)$. □

LEMMA 3.7. *If $m < n/2$ and $p_i, p_j \in S$ are two distinct points, then neither $U_m(p_i)$ nor $U_m(p_j)$ is contained in the other.*

Proof. If $U_m(p_i) \cap U_m(p_j) \neq \emptyset$, then Lemma 3.5 guarantees that b_{ij} contributes a Voronoi edge to $V_m(S)$, say, e_{ij}. The points in the neighborhood of e_{ij} lying on the same side as p_i are contained in $U_m(p_i)$ but not in $U_m(p_j)$ and vice versa. Therefore, neither region can be contained in the other. □

LEMMA 3.8. *Let $m < n/2$; let $p_i, p_j, p_l \in S$ be three points such that $U_m(p_i) \cap U_m(p_j) \cap U_m(p_l) \neq \emptyset$; and let U be a connected component of $U_m(p_i) \cap U_m(p_j) \cap U_m(p_l)$. Then there is a Voronoi vertex v of $V_m(S)$ such that at least two of the three points determining v are in $\{p_i, p_j, p_l\}$, and v lies in U.*

Proof. Let U_{ij} be the connected component of $U_m(p_i) \cap U_m(p_j)$ that contains U. By Lemma 3.5, there is a Voronoi vertex v in U_{ij} determined by p_i, p_j, and some other point p_h. If v also belongs to U_l, then we are finished. Otherwise, since U_l cannot be completely contained in $U_m(p_i) \cap U_m(p_j)$ (cf. Lemma 3.7), the boundaries of U_l and U_{ij} intersect, say, at a point u. Since the boundaries of U_l and U_{ij} consist of Voronoi edges, u is necessarily a Voronoi vertex, and its determining points include p_l and p_i (resp. p_l and p_j) if u is common to the boundaries of $U_m(p_l)$ and $U_m(p_i)$ (resp. $U_m(p_l)$ and $U_m(p_j)$). This completes the proof. □

LEMMA 3.9. *The total number of distinct triples (p, q, r) such that p, q, and r are contained in a Voronoi set of $U_m(S)$ is $O(m^2 n)$.*

Proof. If $m \geq n/2$, then the lemma is trivial: there are only $O(n^3)$ triples. Otherwise we proceed as follows. Notice that, for any triple (p, q, r) contained in Voronoi set, $U_m(p) \cap U_m(q) \cap U_m(r) \neq \emptyset$. By Lemma 3.8, we can associate each such triple with a Voronoi vertex of $V_m(S)$. We now need to show that a Voronoi vertex is not charged against too many distinct triples. First we observe that the number of different regions $U_m(x)$, for $x \in S$, containing a Voronoi vertex is $m + 2$; this follows because if R_1, R_2, and R_3 are the three Voronoi regions of $V_m(S)$ incident with a Voronoi vertex and if the set of m points associated with

R_1 is $\{p_1, p_2, \dots, p_m\}$, then only the regions $U_m(p_i), i = 1, 2, \dots, p_m$, may contain the Voronoi region R_1; furthermore, since the Voronoi sets of two neighboring Voronoi regions differ by exactly one point, R_1, R_2, and R_3 together have only $m + 2$ different points. Now consider a Voronoi vertex v and suppose that p_a, p_b, p_c are the three points determining v. By Lemma 3.8, if a triple (p, q, r) is associated with v, then (a) $v \in U_m(p) \cap U_m(q) \cap U_m(r)$, and (b) at least two of p, q, and r are in $\{p_a, p_b, p_c\}$. There are three ways to choose the two points from $\{p, q, r\}$ as possible determiners of v. For a fixed choice of two determiners, say, p and q, our accounting scheme may assign to v all the triples (p, q, r) that satisfy condition (a). However, as noted above, there are only $m + 2$ choices of x for which v lies in $U_m(x)$. Not counting p and q, we are left with m choices for r. Consequently, the vertex v gets associated with at most $3m$ triples. Since there are altogether $O(mn)$ Voronoi vertices in $V_m(S)$, the number of distinct triples is $O(m^2 n)$. □

Given two points p_i and p_j of S such that $U_m(p_i) \cap U_m(p_j) \neq \emptyset$, let S_{ij} be the union of all Voronoi sets that contain both p_i and p_j.

LEMMA 3.10. *The sum of* $|S_{ij}|$ *over all pairs* p_i *and* p_j *with* $U_m(p_i) \cap U_m(p_j) \neq \emptyset$ *is* $O(m^2 n)$.

Proof. If a point p_l belongs to S_{ij}, then the triple (p_i, p_j, p_l) is such that $U_m(p_i) \cap U_m(p_j) \cap U_m(p_l) \neq \emptyset$. In summing $|S_{ij}|$ over all pairs p_i and p_j, with $U_m(p_i) \cap U_m(p_j) \neq \emptyset$, we count each such triple three times. Thus, three times the number of distinct triples (p, q, r), with $U_m(p) \cap U_m(q) \cap U_m(r) \neq \emptyset$, is an upper bound on the sum of $|S_{ij}|$'s. This quantity is $O(m^2 n)$ as shown by Lemma 3.9. □

LEMMA 3.11. *The sets* S_{ij}*'s for all pairs* (p_i, p_j) *with* $U_m(p_i) \cap U_m(p_j) \neq \emptyset$ *can be computed in total time* $O(m^2 n + n \log n)$.

Proof. We assume that the Voronoi diagram $V_m(S)$ is given in a standard form, where the edges of each face are listed in a cyclic and the edges incident to each vertex are listed in an angularly sorted order. With each edge e, we store the two points p and q whose bisector contains e and, with each Voronoi vertex v, we store the three points p, q, and r that determine v. Computing $V_m(S)$ with this representation takes $O(m^2 n + n \log n)$ time [1].

Consider a set S_{ij} and a point that belongs to it. Observe that $U_m(p_i) \cap U_m(p_j) \cap U_m(p_l) \neq \emptyset$. We define p_l to be a *type-one* point of S_{ij} if $U_m(p_i) \cap U_m(p_j) \subseteq U_m(p_l)$, and a *type-two* point otherwise. Observe that if p_l is a type-two point, then the boundaries of $U_m(p_l)$ and $U_m(p_i) \cap U_m(p_j)$ intersect since Lemma 3.7 states that $U_m(p_l) \not\subset (U_m(p_i) \cap U_m(p_j))$. Now, if p_l is a type-one point of S_{ij}, then p_l belongs to every Voronoi set whose Voronoi region lies in $U_m(p_i) \cap U_m(p_j)$. Thus we can find all the type-one points of S_{ij} (there are only m of these) by choosing any Voronoi set that contains both p_i and p_j. Next, if p_l is a type-two point, then the common intersection of the boundaries of $U_m(p_l)$ and $U_m(p_i) \cap U_m(p_j)$ is a Voronoi vertex of $V_m(S)$. Thus the type-two points of S_{ij} can be discovered by traversing the boundary of $U_m(p_i) \cap U_m(p_j)$; that is, these points are associated with the Voronoi vertices encountered during the traversal.

Since there is a one-to-one correspondence between the pairs (p_i, p_j) for which we need to compute S_{ij} and the Voronoi edges of $V_m(S)$ (cf. Lemma 3.5), we can explore all distinct pairs by processing all the edges of $V_m(S)$. If e is an edge that is part of the bisector b_{ij}, then we start the computation of S_{ij} by determining a Voronoi region that is incident to e and contains both p_i and p_j. (Notice that ordinarily an edge e is incident to only four regions.) We initialize S_{ij} to be the Voronoi set associated with this region. (This takes care of all type-one points of S_{ij}.) We then traverse the boundary of $U_m(p_i) \cap U_m(p_j)$ by visiting adjacent Voronoi regions. Whenever we encounter a Voronoi vertex, we add a new point to S_{ij}; these are the type-two points of S_{ij}. (To check whether or not an edge is on the boundary of $U_m(p_i) \cap U_m(p_j)$, we only need to look at the two points that determine the edge: the edge is on the boundary if and only if p_i or p_j is one of its determiners.) That such a traversal correctly computes S_{ij} follows from the preceding discussion.

Finally, we need to establish the running time of our algorithm. The cost of the algorithm is dominated by the number of edges of $V_m(S)$ traversed during the algorithm. We claim that the algorithm traverses each edge of $V_m(S)$ at most $2m$ times. The proof of the claim is as follows. Given an edge e on the boundary of $U_m(p_i) \cap U_m(p_j)$, consider a point p_l such that e also belongs to the boundary of $U_m(p_i) \cap U_m(p_l)$. Clearly, the intersection of the two boundaries is a Voronoi vertex, and we proved in Lemma 3.9 that there are at most m choices of p_l. Similarly, there are at most m choices of p_l for which e also belongs to the boundary of $U_m(p_i) \cap U_m(p_l)$. Thus the edge e is traversed at most $2m$ times. Since there are $O(mn)$ Voronoi edges in $V_m(S)$, the bound claimed in the lemma follows. □

THEOREM 3.3. *The k-diameter problem for n points in the plane can be solved in* $O(k^{2.5}n \log k + n \log n)$ *time and* $O(kn)$ *space.*

Proof. If $k > n/3$, then the claim follows from Theorem 3.1. Otherwise, by Lemma 3.1, there is a Voronoi set of $V_{3k-3}(S)$ that contains an optimal set of k points. If S_k is a set of k points with a minimum diameter whose diameter is determined by the points p_i and p_j, then the definition of S_{ij} implies that $S_k \subseteq S_{ij}$. Thus, we only need to solve the k-diameter problem for all the S_{ij} sets. Solving the problem for S_{ij} takes time $O(|S_{ij}| + k^{1.5}\log k)$ (see the discussion following Lemma 3.4). A bound on the total running time of the algorithm is obtained by summing up this time over all pairs (p_i, p_j) with $U_m(p_i) \cap U_m(p_j) \neq \emptyset$, where $m = 3k - 3$. Since Lemma 3.10 states that

$$\sum_{U_m(p_i)\cap U_m(p_j)\neq\emptyset} |S_{ij}| = O(m^2 n),$$

the time complexity of the algorithm is

$$O(k^2 n + k^{2.5} n \log k) = O(k^{2.5} n \log k),$$

plus the cost of computing the Voronoi diagram. This completes the proof. □

This completes our discussion of the k-diameter problem. In the following we cite an application of Lemma 3.3 to finding a maximum clique.

3.4. *Maximum Clique of Circles*

Consider a set C of n unit circles in the plane. The intersection graph of C has a vertex for each circle and an edge between two vertices if the corresponding circles intersect. We show that a maximum clique of the intersection graph of C can be computed in time $O(n^{3.5}\log n)$.

Let S be the set of the centers of the circles in C. Let $G(2)$ be the graph whose vertices are points of S and two vertices have an edge between them if the distance between the corresponding points is no more than two units. We observe that $G(2)$ is precisely the intersection graph of C. So, the problem is to find a clique of maximum size in the graph $G(2)$. Let K be a clique of S whose diameter is determined by the points p_i and p_j, where $p_i, p_j \in S$. Then K is also a clique in the restricted graph $G(p_i, p_j)$; recall that $G(p_i, p_j)$ is the graph $G(2)$ restricted to the lune of p_i and p_j. We can find a maximum clique of $G(2)$ by finding a maximum clique in each of the subgraphs $G(p_i, p_j)$, where p_i and p_j satisfy $\|p_i - p_j\| \leq 2$, and by selecting the largest clique among them. By Lemma 3.3, the maximum independent set of the complement graph, or the maximum clique of $G(p_i, p_j)$, can be found in time $O(n^{1.5}\log n)$. We therefore have the following theorem.

THEOREM 3.4. *Given a set of n unit circles in the plane, we can find a maximum clique of their intersection graph in time $O(n^{3.5}\log n)$ and space $O(n)$.*

This theorem seems interesting since finding a maximum independent set in the intersection graph of unit circles is NP-complete [4]. We can also extend this result to the weighted case, where each circle has a positive weight and we ask for a clique of maximum weight. The time complexity in this case is $O(n^5)$; we solve $O(n^2)$ instances of a maximum flow problem.

4. Minimum-Perimeter Squares and Rectangles

Let us first consider the problem of finding an axes-parallel square of minimum perimeter and, therefore, of minimum area containing k points. It turns out that this problem is easily solved by using the kth order Voronoi diagram of S in the L_∞ metric. Specifically, let S_k be a set of k points with a smallest enclosing square, and let c be the center of the corresponding square. We observe that S_k is the set of k nearest neighbors of c in the L_∞ metric, and consequently S_k is a Voronoi set of the kth order Voronoi diagram of S in the L_∞ metric. The Voronoi diagram of S in the L_∞ metric can be computed in time $O(k^2 n \log n)$ (see Lee [10]), and computing a smallest enclosing square for each of the $O(kn)$ Voronoi sets is also accomplished easily within this time bound. This establishes the following theorem.

Theorem 4.1. *Given a set S of n points in the plane, we can find a minimum-perimeter axes-parallel square containing k points of S in $O(k^2 n \log n)$ time and $O(kn)$ space.*

A similar time bound is possible for the problem of finding a set of k points with the smallest circumscribing circle. It is well known that the center of such a circle coincides with a vertex of the kth order Voronoi diagram of S. Thus, a set of k points with the smallest enclosing circle can be found in $O(k^2 n + n \log n)$ time [1].

Next, we consider the problem of finding a minimum-perimeter axes-parallel rectangle containing k points of S. As a starter, we observe that each of the four edges of a minimum-perimeter rectangle contains a point of S. Let R be a minimum-perimeter rectangle containing k points, and a, b, c, d be the four corners of R in the counterclockwise order, where a is the lower-left corner. Suppose that the corner a is determined by the points p_1 and p_2, with p_2 below p_1; it may be that p_1 and p_2 are the same point, in which case $p_1 = p_2 = a$. For a given choice of the lower-left corner a, we wish to determine the opposite corner c such that resulting rectangle contains k points and has the smallest possible perimeter. Procedure Rectangle given below computes such a rectangle.

We assume for simplicity that no two points lie on a horizontal or vertical line. We use $x(\cdot)$ and $y(\cdot)$ to denote the x and y coordinates of a point.

```
PROCEDURE RECTANGLE.
1. Let $U$ be the set of points $s \in S$ that satisfy $x(s) \geq x(p_1)$ and
   $y(s) \geq y(p_2)$. (Recall that $p_1$ and $p_2$ determine the lower-left corner
   $a$.)
2. Initialize $R$ to be $+\infty$.
3. **While** $U \neq \emptyset$ **do**
     **begin**
       Let $u$ be the topmost point in $U$ and let $v$ be the $k$th point of
       $U$ in the left-to-right order.
       **if** $y(u) < y(p_1)$, then delete all remaining points from $U$ (* this
         terminates the procedure*)
       **elseif** $x(v) < \max\{x(u), x(p_2)\}$, then delete $u$ from $U$,
       **else set** $R := \min\{R, 2(x(v) - x(p_1) + y(u) - y(p_2))\}$, and
               delete $u$ from $U$.
     **end**
4. **Return** $R$ and **stop**.
**end Procedure.**
```

One easily verifies that Procedure Rectangle determines a minimum-perimeter rectangle containing k points, with the lower-left corner at a. The procedure is easily implemented to run in time $O(n)$ after presorting the points of S, once by their x-coordinates and once by their y-coordinates. To compute a minimum-perimeter rectangle containing k points, we call Procedure Rectangle once for each of the $O(n^2)$ choices of the lower-left corner a. The total running time of this algorithm is $O(n^3)$, since after spending $O(n \log n)$ time on sorting, each procedure call takes $O(n)$ time.

THEOREM 4.1. *Given a set S of n points in the plane, we can find an axes-parallel minimum-perimeter rectangle containing k points of S in time $O(n^3)$ and space $O(n)$.*

To further improve this bound, we use ideas similar to the ones used in Section 3. We begin with the following elementary lemma.

LEMMA 4.1. *Let C be a unit circle and let R be a rectangle whose all four corners lie on C. Then the perimeter of R is greater than* 4.

Proof. By the triangle inequality, the perimeter of R is greater than twice the length of its diameter. Since the diameter of R equals the diameter of C, which is 2, the lemma follows. □

The following lemma establishes the connection with a higher-order Voronoi diagram.

LEMMA 4.2. *Let S_k be a set of k points with a minimum-perimeter enclosing rectangle. Then S_k is contained in a Voronoi set of $V_{6k-6}(S)$.*

Proof. We show that there exists a set $T \subseteq S$ such that (1) $S_k \subseteq T$, (2) $|T| \leq 6k - 6$, and (3) there is a circle that contains T and no other point of S. The set T satisfying the three properties is clearly contained in a Voronoi set of order $6k - 6$. Let R be the minimum-perimeter rectangle enclosing S_k. Let C be the circumscribing circle of R and let r be the radius of C. By Lemma 4.1, the perimeter of R is greater than $4r$. If we divide the interior of C into six equal parts by drawing six radial line segments from the center to six equally spaced boundary points, then each part can be enclosed by a rectangle of perimeter at most $4r$. Therefore, if S_k is a set with a minimum-perimeter enclosing rectangle, then C cannot have more than $6k - 6$ points; otherwise, one of the six parts would have at least k points that can be enclosed in a smaller rectangle. The set of points in C satisfy the three properties (1), (2), and (3), and the proof is finished. □

Remark. The bound in Lemma 4.2 can be improved to $4k - 4$, but that does not affect the time bounds derived below.

Combining Theorem 4.1 and Lemma 4.2, we readily obtain an $O(k^4 n + n \log n)$ time algorithm, as follows: compute the $(6k - 6)$th order Voronoi diagram of S and, for each Voronoi set of $V_{6k-6}(S)$, use Theorem 4.1 to find a minimum-perimeter rectangle containing k points in time $O(k^3)$. Since there are $O(nk)$ regions and computing the diagram $V_{6k-6}(S)$ takes time $O(k^2 n + n \log n)$, the claim follows. We can further improve this bound by using the results of Section 3.3.

THEOREM 4.2. *Given a set S of n points in the plane, we can find a minimum-perimeter axes-parallel rectangle containing k points of S in time $O(k^2 n \log n)$ and space $O(kn)$.*

Proof. Let S_{ij} denote the set of all points belonging to Voronoi sets containing both p_i and p_j, that is, points in the region $U_m(p_i) \cap U_m(p_j)$, where $m = 6k - 6$. Then, by Lemma 4.2, there exists an optimal rectangle all of whose k points are in the set S_{ij} and where the lower-left corner of the rectangle is determined by the pair (p_i, p_j). It therefore suffices to run Procedure Rectangle for sets S_{ij}'s for all i and j. To bound the running time of this algorithm, we note that the Voronoi diagram can be computed in time $O(k^2 n + n \log n)$ and Procedure Rectangle takes time $O(|S_{ij}| \log |S_{ij}|)$ for a pair (p_i, p_j). The running time of the algorithm is obtained by summing up this time over all pairs (p_i, p_j) with $U_m(p_i) \cap$

$U_m(p_j) \neq \varnothing$, where $m = 6k - 6$. Since Lemma 3.10 states that

$$\sum_{U_m(p_i) \cap U_m(p_j) \neq \varnothing} |S_{ij}| = O(m^2 n),$$

the time complexity of the algorithm if $O(k^2 n \log n)$. □

5. Concluding Remarks

Given a set of points S, we have presented efficient algorithms for selecting k points from S with a minimum diameter or variance. Algorithms for finding minimum-perimeter squares and rectangles, with sides parallel to axes, containing k points of S are also given.

In deriving our algorithms we also established new combinatorial bounds for the total number of pairs in all the Voronoi sets. Specifically, a naive bound for the number of distinct pairs (p, q) such that p and q are contained in some Voronoi set of $V_k(S)$ is $O(k^3 n)$, where $|S| = n$; this follows because there are $O(nk)$ Voronoi sets and $O(k^2)$ pairs per Voronoi set. We show that the number of such pairs is in fact $O(kn)$. Generalizing this, we show that the number of distinct triples, with all three points contained in some Voronoi set, is $O(k^2 n)$. These improved bounds are used to speed up our algorithms, and we except that they will find other applications.

The line of research followed in this paper resembles the one pursued by Dobkin, Drysdale, and Guibas [6]. They showed that given a set S of n points in the plane, the smallest-perimeter convex polygon containing at least k points can be found in time $O(k^2 n \log n + k^5 n)$ time and $O(kn)$ space. Without the restriction of convexity, their algorithm finds a (possibly non-convex) smallest-perimeter polygon in time

$$O\left(k^4 n \log n + \binom{O(k^2)}{k} k! n\right)$$

and space $O(k^2 n)$. For fixed k, both of these algorithms run in $O(n \log n)$ time, which means that a smallest-perimeter triangle or quadrilateral can be found in $O(n \log n)$ time. For arbitrary values of k, however, the time complexity of the second algorithm is exponential in k; it is shown in [6] that finding a smallest-perimeter k-gon, whose vertices belong to S, is NP-hard when $k = n^{\varepsilon}$ for any $\varepsilon > 0$.

There are several directions for further research on this topic. One is to explore other problems to which our technique can be applied. A second direction is to improve the time complexity of various algorithms given in this paper or in [6]. Finally, other optimization criteria such as area also need to be investigated further. It is well known that a smallest-area triangle among a set of n points can be found in time $O(n^2)$. Can we find the smallest-area triangle in subquadratic, in particular, $O(n \log n)$ time? (Recall that a smallest-perimeter triangle can indeed be found in $O(n \log n)$ time.) What about minimum-area k-gons? Recently, *D.* Eppstein has obtained some results along these lines [14]. He presents an $O(kn^3)$ time algorithm for finding a minimum perimeter or area convex k-gon with vertices in S, and an $O(n^2)$ time algorithm for finding a minimum area (not necessarily convex) quadrilateral with vertices in S.

Acknowledgment

We thank an anonymous referee for suggestions that have improved the presentation of this paper.

References

1. A. Aggarwal. L. J. Guibas, J. Saxe and P. Shor, A linear time algorithm for computing the Voronoi diagram of a convex polygon, *Discrete Comput. Geom.* **4**, No. 6 (1989), 591-604.
2. A. Aggarwal and S. Suri, Fast algorithms for computing the largest empty rectangle, extended abstract, *in* "Proceedings, 3rd Annual ACM Symposium on Computational Geometry, 1987," pp. 278–290; IBM Research Report, Yorktown Heights, NY, 1989.
3. H. C. Andrews, "Introduction to Mathematical Techniques in Pattern Recognition," Wiley–Interscience, New York, 1972.
4. Te. Asano, Difficulty of the maximum independent set problem on intersection graphs of geometric objects, *in* "Proceedings, 6th International Conference on the Theory and Applications of Graphs, Kalamazoo, Michigan, 1990."
5. T. Asano, B. Bhattacharya, M. Keil and F. Yao, Clustering algorithms based on minimum and maximum spanning trees, *in* "Proceedings 4th Annual ACM Symposium on Computational Geometry, 1988," pp. 252–257.
6. D. P. Dobkin, R. L. Drysdale III, and L. J. Guibas, Finding smallest polygons, *in* "Advances in Computing Research," Vol. 1, pp. 181–214, JAI Press, Greenwich, CT, 1983.
7. J. A. Hartigan, "Clustering Algorithms," Wiley, New York, 1975.
8. J. Hershberger and S. Suri, Finding tailored partitions, *in* "Proceedings 5th Annual ACM Symposium on Computational Geometry, 1989"; J. Algorithms, to appear.
9. H. Imai and T. Asano, Efficient algorithms for geometric graph search problems, *SIAM J. Comput.* **15**, No. 2 (1986), 478–494.
10. D. T. Lee, On k-nearest neighbor Voronoi diagrams in the plane, *IEEE Trans. Comput.* **C-31** (1982), 478–487.
11. M. McKenna, J. O'Rourke, and S. Suri, Finding the largest rectangle in an orthogonal polygon, *in* "Proceedings 23rd Annual Allerton Conference on Communications, Control, and Computing, 1985, pp. 486–495.
12. P. Vaidya, Geometry helps in matching, *SIAM J. Comput.*, **18**, No. 6 (1989), 1201–1225.
13. H. Edelsbrunner and R. Seidel, Voronoi diagrams and arrangements, *Discrete Comput. Geom.* **1**, No. 1 (1986), 25–44.
14. D. Eppstein, Finding the smallest quadrilateral, *Discrete Comput. Geom.*, to appear.