Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface

Edited by William F. Eddy

With 122 Figures

Springer-Verlag
New York Heidelberg Berlin

1981

RECENT RESULTS ON THE AVERAGE TIME BEHAVIOR OF SOME ALGORITHMS IN COMPUTATIONAL GEOMETRY

Luc Devroye*, McGill University

ABSTRACT

We give a brief inexhaustive survey of recent results that can be helpful in the average time analysis of algorithms in computational geometry. Most fast average time algorithms use one of three principles: bucketing, divide-and-conquer (merging), or quick elimination (throw-away). To illustrate the different points, the convex hull problem is taken as our prototype problem. We also discuss searching, sorting, finding the Voronoi diagram and the minimal spanning tree, identifying the set of maximal vextors, and determining the diameter of a set and the minimum covering sphere.

KEYWORDS Algorithms. Average time. Computational geometry. Convex hull. Sorting. Searching. Closest point problems. Divide and conquer.

1. INTRODUCTION.

There has been an increasing interest in the study and analysis of algorithms in computational geometry (a recent survey paper by Toussaint (1980) had 168 references). Most of the emphasis has been placed on the study of the worst-case complexity of various algorithms under several computational models. It is well-known that many algorithms perform considerably better on the average than predicted by the worst-case analyses. In this note, we would like to point out a few recent developments in the analysis of the average complexity of some algorithms. To keep the general discussion simple and yet insightful we make a couple of convenient assumptions.

The Assumptions.

The input data $X_1,\dots,X_n$ can be considered as a sequence of independent identically distributed R^d-valued random vectors with common density f. (Here the unrealistic assumption is that real numbers can be stored in a computer.)

An algorithm takes time $T=T(X_1,\dots,X_n)$, a Borel measurable function of the input data, and it halts with probability one, i.e. $T < \infty$ almost surely.

The common operations (+,-,/,*,mod,compare,move) take time uniformly bounded over all vaues of the operands. For example, a*b or a mod b take time bounded by a constant not depending upon a or b. (Once again, this is unrealistic, because the multiplication or comparison of two real numbers takes infinite time.)

* The author is with the School of Computer Science, McGill University, 805 Sherbrooke Street West, Montreal, Canada H3A 2K6.

Average Time.

We are interested in the average time E(T) taken by certain algorithms. Obviously, E(T) depends upon f and n only because the averaging is done over all random samples of size n drawn from the density f.

Fast Average Time Algorithms.

In many applications, fast average time algorithms can be obtained by the bucketing principle: find the smallest rectangle C covering $X_1,\dots,X_n$; divide C into equal-sized rectangles (buckets), and solve the problem by travelling from bucket to bucket while performing some local operations. We will also discuss the dramatic savings in average time that can be obtained by the proper application of the divide-and-conquer principle. Finally, the quick elimination (or: throw-away) principle may allow us to further reduce the average time: here one takes a superficial look at the data, and eliminates useless points. The more involved work is then performed on the reduced data sequence.

We will not discuss all the fast average time algorithms. For example, to find the convex hull of $X_1,\dots,X_n$ in R^2 under certain computational models, at least c n log n time is needed (Avis, 1979; Yao, 1979). Jarvis' algorithm (Jarvis, 1973) takes average time 0(nE(N)) where N is the number of convex hull points. In the design of this algorithm, no special care is taken to obtain fast average time behavior. Nevertheless, for certain distributions E(N)=0(1) (see Carnal (1970); this is true for multivariate t-distributions, etc.), so that Jarvis' algorithm runs in linear average time for a fairly large class of distributions. In this note, we will be satisfied with a short inexhaustive and biased survey of deliberate attempts at reducing the average time of algorithms and of the probability theoretical and mathematical tools needed in the ensuing analysis.

2. THE BUCKETING PRINCIPLE.

Let C be the smallest closed rectangle covering $X_1,\dots,X_n$ and let C be divided into m^d equal-sized rectangles (buckets) where $m=\text{int}(n^{1/d})$. Bucket memberships can thus be obtained for all data points in time 0(n). Often one keeps track of these memberships by using m^d linked lists, one per bucket, so that 0(n) space is used.

The obvious application in R^1 involves sorting $X_1,\dots,X_n$. Here one empties the buckets from left to right and performs a subsequent sort within each bucket, if necessary. If this subsequent sort is a comparison-based sort (e.g. heapsort, bubble sort, shell sort, merge sort or quicksort) with average time $g(n)$ (this number is independent of f, since we have a comparison-based sort), then the overall average time for sorting is

$$E(T) = O(n) + E(\sum_{i=1}^{n} g(N_i))$$

where $N_1,\ldots,N_n$ are the cardinalities of the n buckets. Devroye and Klincsek (1981) addressed the question of when $E(T)=O(n)$. They showed that when $g(u) \uparrow \infty$ as $u \to \infty$, $g(u)/u^2$ is nonincreasing, and g is convex, then $E(T)=O(n)$ if and only if f has compact support and

$$\int g(f(x))\, dx < \infty. \qquad (1)$$

Notice that they put no continuity or boundedness assumptions on f. Akl and Meijer (1980) found that for sufficiently smooth densities, bucket sort (with slight ad hoc improvements) compares favorably with even the best version of quicksort.

Consider now the following generalization of the previous result: travel from bucket to bucket, performing within the j-th bucket operations taking average time bounded between $ag(N_j)$ and $bg(N_j)$ when N_j is given. Here $0 < a \le b < \infty$. Once again, the average time of the entire algorithm is

$$E(T) = O(n+E(\sum_{i=1}^{m^d} g(N_i))).$$

Assume that $g(u)/u \uparrow \infty$ and $g(u)/u^K \downarrow 0$ for some positive K as $u \to \infty$. Also, assume that g is convex. Then $E(T)=O(n)$ if and only if f has compact support and (1) holds (Devroye, 1981a). These basic results have several applications. We cite just two examples.

Examples.

1. Searching in constant average time.

Assume that $X_1,\ldots,X_n$ are stored in the bucket data structure given above, and that we are presented with X_Z (where Z is uniformly distributed over $1,\ldots,n$). We have to determine the index i such that $X_i=X_Z$. This is the classical problem of successful search. If the X_i's are stored in the buckets in order of arrival, then the average search time is comparable to

$$E(\frac{1}{n}\sum_{i=1}^{m^d} N_i^2+1)$$

(note: two sequences a_n,b_n are comparable when $a_n=O(b_n)$ and $b_n=O(a_n)$). By a simple extension of the previous result, we see that $E(T)=O(1)$ if and only if f has compact support and $\int f^2(x)\, dx < \infty$. If within each bucket the data are organized into a binary search tree rather than a linked list, by considering one of the coordinates of the X_i's as the key for sorting, then $E(T)=O(1)$ if and only if f has compact support and

$$\int f(x) \log_+ f(x)\, dx < \infty.$$

2. Convex hull algorithms that are based upon sorting.

The convex hull of $X_1,\ldots,X_n$ is a subsequence $X_{i_1},\ldots,X_{i_k}$ of $X_1,\ldots,X_n$ such that for all X_{i_j} there exists a hyperplane through X_{i_j}, and all X_q's, $q\neq i_j$, belong to the same closed halfspace determined by this hyperplane. In R^2, it can be obtained from $X_1,\ldots,X_n$ as follows: (i) Find a point x that belongs to the interior of the convex hull of $X_1,\ldots,X_n$. Sort all that X_i's according to the polar angles of X_i-x (by using the bucket sort described above). This yields a polygon P. (ii) Visit all vertices of P in turn by pushing them on a stack. Pop the stack when non-convex-hull points are encountered. In essence, this is Graham's algorithm (1972) with a modification in the sorting method that is used. Step (ii) takes time $O(n)$. The average time taken by (i) is $O(n)$ when the density of the polar angle of X_i-x is square integrable. Since x itself is a random vector, one must be careful before making any inference about f. Nevertheless, it is sufficient that f is bounded and has compact support. End of examples.

The previous applications have one feature in common: the times taken by the algorithms on individual buckets just depend upon the number and/or position of the data points within these buckets (and not on, say, the number of data points in neighboring buckets). In more involved problems, we cannot avoid looking at neighboring buckets. For example, consider the class of "closest point problems" in R^d (Shamos and Hoey, 1975) such as: find all nearest neighbor pairs, construct the Voronoi graph, find the minimal spanning tree, etc. (see Bentley and Friedman (1979) for other applications). Shamos (1978) and Weide (1978) discuss many applications of the bucketing principle, and Bentley, Weide and Yao (1980) give a fairly comprehensive treatment of the average time analysis of bucketing algorithms for closest point problems. We take the liberty to cite a couple of examples from their study:

Examples.

1. The all-nearest-neighbor problem.

All nearest neighbor pairs can be found in $O(n \log n)$ time (worst-case) (Lipton and Tarjan, 1977). Weide (1978) proposed a bucketing algorithm in which for a given X_i, a "spiral search" is started in the bucket of X_i, and continues in neighboring cells, in a spiraling fashion, until no data point outside the buckets already checked can be closer to X_i than the closest data point already found. Bentley et. al. (1980) showed that Weide's algorithm halts in average time $O(n)$ when there exists a bounded open convex region B such that the density f of X_1 is 0 outside B and satisfies

$$0 < \inf_B f(x) \le \sup_B f(x) < \infty .$$

2. The Voronoi diagram.

The Voronoi diagram in R^2 can be found in time $O(n \log n)$ (worst-case) (Shamos (1978), Horspool (1979), Brown (1979)). Bentley et. al. (1980) have a bucketing algorithm that uses sprial search and has some additional features. The Voronoi diagram can be found in average time $O(n)$ when d=2 and the density f of X_1 satisfies the condition of Example 1. From the Voronoi diagram, the convex hull can be obtained in linear time (Shamos, 1978).

3. The minimal spanning tree.

For a graph (V,E), Yao (1975) and Cheriton and Tarjan (1976) give algorithms for finding the minimal spanning tree (MST) in worst-case time $O(|E| \log \log |V|)$. The Euclidean minimal spanning tree (EMST) of n points in R^d can therefore be obtained in $O(n \log \log n)$ time if we can find a supergraph of the EMST with $O(n)$ edges in $O(n \log\log n)$ time. Yao (1977) suggested to find the nearest neighbor of each point in a critical number of directions; the resulting graph has $O(n)$ edges and contains the MST. This nearest neighbor search can be done by a slight modification of the algorithm in Example 1. Hence, the EMST can be found in average time $O(n \log \log n)$ for any d and for all distributions given in Example 1. The situation is a bit better in R^2. We can find a planar supergraph of the EMST in average time $O(n)$ (such as the Delaunay triangulation (the dual of the Voronoi diagram), the Gabriel graph, etc.) and then apply Cheriton and Tarjan's (1976) $O(n)$ algorithm for finding the MST of a planar graph. Thus, in R^2 and for the class of distributions given in Example 1, we can find the EMST in linear average time. End of examples.

Finally, we should mention a third group of bucketing algorithms, where special buckets are selected based upon a global evaluation of the contents of the bucket. For example, assume that not more than a_n buckets are selected according to some criterion (from the approximately n original buckets) in time $O(n)$, and that only the data points within the selected buckets are considered for further processing. If N is the number of selected points, then we assume that "further processing" takes time $O(g(N))$ for a given function g. Because the global evaluation procedure is not specified, we should assume the worst case, and this leads to the study of the order statistics of the cardinalities of the buckets. The following results can be found in Devroye (1981b). When M is the maximum of n i.i.d. Poisson (1) random variables, then $E(M) \sim \log n / \log \log n$. The same is true if M is the maximum of $N_1,\dots,N_n$, where N_i is the cardinality of $[\frac{i-1}{n}, \frac{i}{n})$ and the data is $U_1,\dots,U_n$, a sequence of i.i.d. uniform (0,1) random variables. Using tight bounds on the upper and lower tails of M, one can show that

$$E(g(N)) = O(g(a_n \frac{\log n}{\log \log n}))$$

where $a_n \ge 1$, g is nondecreasing, $g(x)=O(1+x^\beta)$ (some $\beta > 0$), $\sup_{x>0} \frac{g(cx)}{g(x)} < \infty$ (all $c > 1$), and the X_i's have a bounded density f with compact support.

Example. The convex hull in R^2.

Shamos (1979) suggested to construct the convex hull in R^2 in the following fashion: mark all the nonempty extremal buckets in each row and column (the extremes are taken in the northern and southern directions for a column, and eastern and western directions for a row); mark all the adjacent buckets in the same rows and columns; apply Graham's $O(n \log n)$ convex hull algorithms to all the points in the marked buckets. It is clear that $a_n=O(\sqrt{n})$ and that the average time of the algorithm is $O(n)+ O(E(g(N))$ = where $g(n)=n \log n$. This is

$$O(n) + O(\sqrt{n} \frac{(\log n)^2}{\log \log n}) = O(n).$$

Furthermore, the average time spent on determining the bucket menberships divided by the total average time tends to 1. End of example.

3. THE DIVIDE-AND-CONQUER PRINCIPLE.

A problem of size n can often be split into two similar subproblems of size approximately equal to n/2, and so forth, until subproblems are obtained of constant size for which the solutions are trivially known. For example, quicksort (Sedgewick (1977, 1978)) is based on this principle. The average time here is $O(n \log n)$, but, unfortunately enough, since the sizes of the subproblems in quicksort can take values 0,1,2,... with equal probabilities, the worst-case complexity is $O(n^2)$. One can start in the other direction with about n equal-sized small problems, and marry subsolutions in a pairwise manner as in mergesort. Because of the controlled subproblem size, the worst-case complexity becomes $O(n \log n)$ (Knuth, 1975). Both principles will be referred to as divide-and-conquer principles. They have numerous applications in computational geometry with often considerable savings in average time. The first general discussion of their value in the design of fast average time algorithms can be found in Bentley and Shamos (1978).

Let us analyze the divide-and-conquer algorithms more formally. Assume that $X_1,\dots,X_n$ are R^d-valued independent random vectors with common distribution, and that we are asked to find $A_n=A(X_1,\dots,X_n)$, a subset of $X_1,\dots,X_n$, where $A(.)$ satisfies:

1) $A(x_1,\dots,x_n)=A(x_{\sigma(1)},\dots,x_{\sigma(n)})$, for all $x_1,\dots,x_n \in R^d$, and all permutations $\sigma(1),\dots,\sigma(n)$ of $1,\dots,n$.

2) $x_1 \in A(x_1,\dots,x_n) \Rightarrow x_1 \in A(x_1,\dots,x_i)$ for all $x_1,\dots,x_n \in R^d$, and all $i \le n$.

The convex hull satisfies these requirements. If $Q_1(x),\dots,Q_{2^d}(x)$ are the open quadrants centered at $x \in R^d$, then we say that X_i is a maximal vector of $X_1,\dots,X_n$ if some quadrant centered at X_i is empty (i.e., contains no X_j, $j \ne i$, $j \le n$). The set of maximal vectors also satisfies the given requirements. Let N = cardinality(A_n). For $p \ge 1$, we know by Jensen's inequality that

$$E(N^p) \ge (E(N))^p .$$

In the present context, we would like an inequality in the opposite direction. For random sets A_n satisfying 1) and 2), and under very weak conditions on the behavior of $E(N)$, we have

$$E(N^p) = O((E(N))^p)$$

(Devroye, 1981c). For example, it suffices that $E(N)$ is nondecreasing, or that $E(N)$ is regularly varying at infinity. Also, if $E(N) \le a_n \uparrow$, then $E(N^p) = O(a_n^p)$. In essence, the results of Devroye (1981c) imply that under weak conditions on the distribution of X_1, $E(N^p)$ and $(E(N))^p$ are comparable. The same is true for other nonlinear functions of N. For example, if $E(N) \le a_n \uparrow$, then $E(N\log(N+e)) = O(a_n \log a_n)$. Thus the knowledge of $E(N)$ allows us to make statements about other moments of N. Here are some known results about $E(N)$.

Examples. 1. A_n is the convex hull. X_1 has a density f.

(i) $E(N) = o(n)$ (Devroye, 1981d).

(ii) If f is normal, then $E(N)=O((\log n)^{(d-1)/2})$ (Raynaud, 1970). For $d=2$, $E(N) \sim 2\sqrt{2\pi \log n}$ (Renyi and Sulanke, 1963, 1964).

(iii) If f is the uniform density in the unit hypersphere of R^d, then $E(N) = O(n^{(d-1)/(d+1)})$ (Raynaud, 1970).

(iv) If f is the uniform density on a polygon of R^2 with k vertices, then $E(N) \sim \frac{2k}{3} \log n$ (Renyi and Sulanke, 1963, 1968).

(v) If f is a radial density, see Carnal (1970). For example, if f is radial, and $P(||X_1||>u)=L(u)/u^r$ where $r \ge 0$ and L is slowly varying (i.e., $L(cx)/L(x) \to 1$ as $x \to \infty$, all $c > 0$), then $E(N) \to c(r) > 0$. If $P(||X_1||>u) \sim c(1-u)^r$ for some $c, r > 0$ as $u \uparrow 1$, and $P(||X_1||>1)=0$, then $E(N) \sim c(r)\, n^{1/(2r+1)}$ for some $c(r) > 0$.

2. A_n is the set of maximal vectors. When X_1 has a density and the components of X_1 are independent then $E(N)$ is nondecreasing (Devroye, 1980) and $E(N) \sim 2^d(\log n)^{d-1}/(d-1)!$ (Barndorff-Nielsen, 1966; Devroye, 1980).

End of examples.

A_n can be found by the following merging method. Assume for the sake of simplicity that $n = 2^k$ for some integer $k > 1$.

1. Let $A_{1i}=A(X_i)$, $1 \le i \le n$. Set $j \leftarrow 1$.
2. Merge consecutive A_{ji}'s in a pairwise manner (A_{j1} and A_{j2} ; A_{j3} and A_{j4} ; etc.).
3. Set $j \leftarrow j+1$. If $j > k$, terminate the algorithm ($A_n = A_{k1}$). Otherwise, go to 2.

We assume that merging and editing of A_{ji} and $A_{j\,i+1}$ with cardinalities k_1 and k_2 can be done in time bounded from above by $g(k_1)+g(k_2)$ for some nondecreasing positive-valued function g, and that $E(g(n)) \le b_n \uparrow$ where, as before, N = cardinality(A_n). Then the given algorithm finds A_n in average time

$$O(n \sum_{j=1}^{2n} b_j/j^2) .$$

If the merging and editing take time bounded from below by $a(g(k_1)+g(k_2))$ and $E(g(N)) \ge sb_n$ where g and b_n are as defined above, and $a,s > 0$ are constants, then we take at least

$$\gamma n \sum_{j=1}^{n} b_j/j^2$$

average time for some $\gamma > 0$ and all n large enough (Devroye, 1981c). Thus, the divide-and-conquer method finds A_n in linear average time if and only if

$$\sum_{j=1}^{\infty} b_j/j^2 < \infty .$$

Examples.

1. The set of maximal vectors.
Merging of two sets of maximal vectors can be achieved in quadratic time by pairwise comparisons, for any dimension d. We can thus take $g(u) = u^2$ in the previous analysis if we merge in this way. If $E(N) \sim a_n \uparrow \infty$, then we can check that the divide-and-conquer algorithm runs in linear average time if and only if

$$\sum_{j=1}^{\infty} a_j^2/j^2 < \infty .$$

2. Convex hulls in R^2.
Two convex hulls with ordered vertices can be merged in linear time into a convex hull with ordered vertices (Shamos, 1978). Thus, if $E(N) = O(a_n)$ and $a_n \uparrow \infty$, then

$$\sum_{j=1}^{\infty} a_j/j^2 < \infty \qquad (2)$$

is sufficient for the linear average time behavior of the divide-and-conquer algorithm given here. When $\liminf E(N)/a_n > 0$, then (2) is also necessary for linear average time behavior. Notice here that (2) is satisfied when, say, $a_n = n/\log^{1+\delta} n$ or $a_n = n/(\log n . \log^{1+\delta} \log n)$ for some $\delta > 0$. This improves the sufficient condition $a_n = n^{1-\delta}$, $\delta > 0$, given in Bentley and Shamos (1978).

3. Convex hulls in R^d.
Merging can trivially be achieved in polynomial (n^{d+1}) time for two convex hulls with total number of vertices equal to n. When $E(N) \le a_n \uparrow$ and

$$\sum_{j=1}^{\infty} a_j^{d+1}/j^2 < \infty ,$$

we can achieve linear average time. This condition is fulfilled for the normal density in R^d and the uniform density on any hypercube of R^d. End of examples.

4. THE QUICK ELIMINATION (THROW-AWAY) PRINCIPLE.

In extremal problems (e.g., find the convex hull, find the minimal covering ellipse, etc.) many of the data points can be eliminated from further considerations without much work. The remaining data points then enter the more involved portion of the algorithm. Often the worst-case time of these elimination algorithms is equal to the worst-case time of the second part of the algorithm used on all n data points. The average time is sometimes considerably smaller than the worst-case time. We illustrate this once again on our prototype problem of finding the convex hull.

Examples.

1. The convex hull.
Assume that we seek the extrema $e_1,\dots,e_m$ in m carefully chosen directions of R^d, form the polyhedron P formed by these extrema, and eliminate all X_i's that belong to the interior of the extremal polyhedron P. The remaining X_i's are then processed by a simple worst-case $O(g(n))$ convex hull algorithm. What can we say about the average time of these algorithms? An average time of $o(g(n))$ would indicate that the elimination procedure is worthwhile on large data sequences. We could also say that the elimination procedure achieves 100% asymptotic efficiency. In Devroye (1981d) it is shown that this happens when (i) the open halfspaces defined by the hyperplanes through the origin perpendicular to the e_i's cover R^d except possibly the origin; and (ii) X_1 has a radial density f, where

$$\alpha(u) = \inf\{t : P(\|X_1\|>t)=u\}$$

is slowly varying at 0 ($\lim_{t\downarrow 0} \alpha(tu)/\alpha(t) = 1$, all $u>0$), and $\alpha(u) \to \infty$ as $u\downarrow 0$. Condition (i) holds when the e_i's are determined by the $d+1$ vertices of the regular $(d+1)$-vertex simplex in R^d centered at the origin. One could also take $2d$ directions defined by $(0,0,\dots,0,\pm 1)$, etc. Condition (ii) is satisfied by the normal density and a class of radial exponential densities (Johnson and Kotz, 1972, pp. 298). The previous result can be sharpened in specific instances. For example, if f is normal, $g(n) = n \log n$, and (i) holds, then the average time is $O(n)$. Furthermore, the average time spent by the algorithm excluding the elimination is $o(n)$ (Devroye, 1981d).

When $d=2$ and f is bounded away from 0 and infinity on a nondegenerate rectangle of R^2 ($f=0$ elsewhere), and $e_1,\dots,e_8$ are equi-spaced directions, then the average time of the elimina- algorithm is $O(n)$ even when $g(n) = n^2$ (Devroye and Toussaint, 1981).

Eddy (1977) has given a slightly different elimination algorithm in which the number of directions and the directions themselves depend upon the data. Akl and Toussaint (1978) report that in R^2, for certain distributions, almost all elimination algorithms achieve extremely fast average times provided that $e_1,\dots,e_m$ are easily computed (e.g., they are axial or diagonal directions.)

2. Finding a simple superset.

Assume that we wish to find $A_n=A(X_1,\dots,X_n)$ in the following manner: (i) Find a set $B_n=B(X_1,\dots,X_n)$ where B_n is guaranteed to contain A_n, in average time T_n; (ii) Given that the cardinality (B_n) is equal to N, find A_n from B_n in worst-case time bounded by $g(N)$. Note that the average time of the entire elimination algorithm is bounded by

$$T_n + E(g(N)) .$$

2.1. A_n is the convex hull, B_n is the set of maximal vectors.

We discussed some distributions for which $T_n=O(n)$. In R^2, step (ii) can be executed with $g(n)=n^2$ (Jarvis' algorithm, 1973) or $g(n)=n \log n$ (Graham's algorithm, 1972). Thus, the entire algorithm takes average time $O(n)$ when $E(N \log_+ N)=O(n)$ or $E(N^2)=O(n)$, according to the algorithm selected in step (ii). When the components of X_1 are independent and X_1 has a density, then these conditions are satisfied by the results of Devroye (1980, 1981c) given in Section 3. The linearity is not lost in this case in R^d even when $g(n) = n^{d+1}$ in step (ii).

2.2. A_n is the diameter of $X_1,\dots,X_n$; B_n is the convex hull.

$A_n=\{X_i,X_j\}$ is called a diameter of $X_1,\dots,X_n$ when $||X_k-X_m|| \le ||X_i-X_j||$ for all $1 \le k \le m \le n$, $(k,m)\neq(i,j)$. Given B_n, some A_n can be found by comparing all $\binom{N}{2}$ distances between points in B_n (see Bhattacharya (1980) for an in-depth treatment of the diameter problem, and a survey of earlier results). But B_n can be found in linear average time for many distributions. In such cases, our trivial diameter algorithm runs in linear average time provided that $E(N)=O(\sqrt{n})$. Assume for example that f is the uniform density in the unit hypersphere of R^d, then the trivial diameter algorithm runs in linear average time if and only if (i) the convex hull can be found in linear average time, and (ii) $d \le 3$ (section 3, example 1 (iii)).

2.3. A_n is the minimum covering circle, B_n is the convex hull.

The minimum area circle in R^2 covering $X_1,\dots,X_n$ has either three convex hull points on its perimeter, or has a diameter determined by two convex hull points. Again, it can be found (trivially) from the convex hull in worst-case time $O(n^4)$ (see Elzinga and Hearn (1972, 1974), Francis (1974) and Shamos (1978) for $O(n^2)$ algorithms and subsequent discussions). Thus, A_n can be identified in linear average time if the convex hull B_n can be found in linear average time, and if $E(N) = O(\sqrt{n})$ (or $O(n^{1/4})$, if the trivial algorithm is used).

5. REFERENCES.

[1] S.G. AKL, H. MEIJER: "Hybrid sorting algorithms: a survey", Department of Computing and Information Science, Queen's University, Technical Report 80-97, 1980.

[2] S.G. AKL, G.T. TOUSSAINT: "A fast convex hull algorithm", Information Processing Letters, vol. 7, pp. 219-222, 1978.

[3] D. AVIS: "On the complexity of finding the convex hull of a set of points", Technical Report SOCS 79.2, School of Computer Science, McGill University, Montreal, 1979.

[4] O. BARNDORFF-NIELSEN, M. SOBEL: "On the distribution of the number of admissible points in a vector random sample", Theory of Probability and its Applications, vol. 11, pp. 249-269, 1966.

[5] J.L. Bentley, J.H. FRIEDMAN: "Data structures for range searching", Computing Surveys, vol. 11, pp. 398-409, 1979.

[6] J.L. BENTLEY, M.I. SHAMOS: "Divide and conquer for linear expected time", Information Processing Letters, vol. 7, pp. 87-91, 1978.

[7] J.L. BENTLEY, B.W. WEIDE, A.C. YAO: "Optimal expected-time algorithms for closest point problems", ACM Transactions of Mathematical Software, vol. 6, pp. 563-580, 1980.

[8] B. BHATTACHARYA: "Applications of computational geometry to pattern recognition problems", Ph.D. Dissertation, McGill University, Montreal, 1980.

[9] K.Q. BROWN: "Voronoi diagrams from convex hulls", Information Processing Letters, vol. 9, pp. 227-228, 1979.

[10] H. CARNAL: "Die konvexe Hülle von n rotationssymmetrische verteilten Punkten", Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete, vol. 15, pp. 168-176, 1970.

[11] P. CHERITON, R.E. TARJAN: "Finding minimum spanning trees", SIAM Journal of Computing, vol. 5, pp. 724-742, 1976.

[12] L. DEVROYE: "A note on finding convex hulls via maximal vectors", Information Processing Letters, vol. 11, pp. 53-56, 1980.

[13] L. DEVROYE: "Some results on the average time for sorting and searching in R^d", Manuscript, School of Computer Science, McGill University, Montreal, 1981a.

[14] L. DEVROYE: "On the average complexity of some bucketing algorithms", Computers and Mathematics with Applications, to appear, 1981b.

[15] L. DEVROYE: "Moment inequalities for random variables in computational geometry", Manuscript, School of Computer Science, McGill University, Montreal, 1981c.

[16] L. DEVROYE: "How to reduce the average complexity of convex hull finding algorithms", Computing, to appear, 1981d.

[17] L. DEVROYE, T. KLINCSEK: "On the average time behavior of distributive sorting algorithms", Computing, vol. 26, pp. 1-7, 1981.

[18] L. DEVROYE, G.T. TOUSSAINT: "A note on linear expected time algorithms for finding convex hulls", Computing, to appear, 1981.

[19] W.F. EDDY: "A new convex hull algorithm for planar sets", ACM Transactions of Mathematical Software, vol. 3, pp. 398-403, 1977.

[20] J. ELZINGA, D. HEARN: "The minimum covering sphere problem", Management Science, vol. 19, pp. 96-104, 1972.

[21] J. ELZINGA, D. HEARN: "The minimum sphere covering a convex polyhedron", Naval Research Logistics Quarterly, vol. 21, pp. 715-718, 1974.

[22] R.L. FRANCIS, J.A. WHITE: Facility Layout and Location: An Analytical Approach, Prentice-Hall, 1974.

[23] R. GRAHAM: "An efficient algorithm for determining the convex hull of a finite planar set", Information Processing Letters, vol. 1, pp. 132-133, 1972.

[24] R.N. HORSPOOL: "Constructing the Voronoi diagram in the plane", Technical Report SOCS 79.12, School of Computer Science, McGill University, Montreal, 1979.

[25] R.A. JARVIS: "On the identification of the convex hull of a finite set of points in the plane", Information Processing Letters, vol. 2, pp. 18-21, 1973.

[26] N.L. JOHNSON, S. KOTZ: Distributions in Statistics: Continuous Multivariate Distributions, John Wiley, New York, 1972.

[27] D. KNUTH: The Art of Computer Programming, vol. 3: Sorting and Searching, Addison-Wesley, Reading, Mass., 2nd Ed., 1975.

[28] R.J. LIPTON, R.E. TARJAN: "Applications of a planar separator theorem", 18th Annual IEEE Symposium on the Foundations of Computer Science, pp. 162-170, 1977.

[29] H. RAYNAUD: "Sur le comportement asymptotique de l'enveloppe convexe d'un nuage de points tirés au hasard dans R^n", Comptes Rendus de l'Académie des Sciences de Paris, vol. 261, pp. 627-629, 1965.

[30] A. RENYI, R. SULANKE: "Uber die konvexe Hülle von n zufällig gewählten Punkten I", Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete, vol. 2, pp. 75-84, 1963.

[31] A. RENYI, R. SULANKE: "Uber die konvexe Hülle von n zufällig gewählten Punkten II", Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete, vol. 3, pp. 138-147, 1964.

[32] A. RENYI, R. SULANKE: "Zufällige konvexe Polygone in einem Ringgebeit", Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebeite, vol. 9, pp. 146-157, 1968.

[33] R. SEDGEWICK: "The analysis of quicksort programs", Acta Informatica, vol. 7, pp. 327-355, 1977.

[34] R. SEDGEWICK: "Implementing quicksort programs", Communications of the ACM, vol. 21, pp. 847-857, 1978.

[35] M.I. SHAMOS: "Computational geometry", Ph.D. Dissertation, Yale University, New Haven, Connecticut, 1978.

[36] M.I. SHAMOS: Seminar given at McGill University, 1979.

[37] M.I. SHAMOS, D. HOEY: "Closest-point problems", Proceedings of the 16th IEEE Symposium on the Foundations of Computer Science, pp. 151-162, 1975.

[38] G.T. TOUSSAINT: "Pattern recognition and geometrical complexity", Proceedings of the 5th International Conference on Pattern Recognition and Image Processing, Miami, Florida, 1980.

[39] B.W. WEIDE: "Statistical methods in algorithm design and analysis", Ph.D. Dissertation, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1978.

[40] A.C. YAO: "An $O(|E|\log\log|V|)$ algorithm for finding minimum spanning trees", Information Processing Letters, vol. 4, pp. 21-23, 1975.

[41] A.C. YAO: "On constructing minimum spanning trees in k-dimensional space and related problems", Research Report STAN-CS-77-642, Department of Computer Science, Stanford University, Stanford, 1977.

[42] A.C. YAO: "A lower bound to finding convex hulls", Technical Report STAN-CS-79-733, Department of Computer Science, Stanford, 1979.