Entropy-Preserving Cuttings and Space-Efficient Planar Point Location

Sunil Arya* Theocharis Malamatos* David M. Mount†

Abstract

Point location is the problem of preprocessing a planar polygonal subdivision S into a data structure in order to determine efficiently the cell of the subdivision that contains a given query point. Given the probabilities p_z that the query point lies within each cell $z \in S$, a natural question is how to design such a structure so as to minimize the expected-case query time. The entropy H of the probability distribution is the dominant term in the lower bound on the expected-case search time. Clearly the number of edges n of the subdivision is a lower bound on the space required. There is no known approach that simultaneously achieves the goals of $H + o(H)$ query time and $O(n)$ space. In this paper we introduce entropy-preserving cuttings and show how to use them to achieve query time $H + o(H)$, using only $O(n \log^* n)$ space.

1 Introduction

Planar point location is an important problem in computational geometry. We are given a polygonal subdivision S consisting of n edges, and the goal is to produce a data structure so that, given any query point q, the polygonal cell of the subdivision containing q can be computed efficiently. The first worst-case optimal result in the area was Kirkpatrick's elegant method based on hierarchical triangulations [9], which supported query processing in $O(\log n)$ time using $O(n)$ space. This was followed by a number of other optimal methods with better practical performance including the layered-DAG of Edelsbrunner, Guibas, and Stolfi [7], searching in similar lists by Cole [6], the method based on persistent search trees by Sarnak and Tarjan [13], and the randomized incremental method of Mulmuley [12] and Seidel [14]. The question of the exact constant factor in query time was raised in work by Goodrich, Orletsky and Ramaiyer [8] and was solved by Adamy and Seidel [1], who showed that point location queries can be answered in $\log_2 n + 2\sqrt{\log_2 n} + o(\sqrt{\log n})$ time.

All of this work was done in terms of worst-case query times. In many applications, point location queries tend to be clustered in regions of greater interest. This raises the question of whether it is possible to use the knowledge of the query distribution to improve expected-case query time. We model this by assuming that for each cell $z \in S$ we are given the probability p_z that a query point lies in z. We assume that the probability that the query point lies on an edge or vertex of the subdivision or lies outside the subdivision is zero.

The *entropy* of the S, denoted H throughout, is defined

$$entropy(S) \;=\; H \;=\; \sum_{z \in S} p_z \log(1/p_z).$$

(We use log to denote the base-2 logarithm.) For the 1-dimensional restriction of this problem, a classical result due to Shannon implies that the expected number of comparisons needed to answer such queries is at least as large as the entropy of the probability distribution [10, 15]. Mehlhorn [11] showed that it is possible to build a binary search tree whose expected search time is at most $H + 2$.

Arya, Cheng, Mount, and Ramesh [2] showed that for subdivisions consisting of convex polygons, assuming that the x and y coordinates of the query point are chosen independently from some probability distribution, the entropy bound can be achieved to within a constant multiplicative factor (2 using quadratic space and about 4 using linear space). These results were strengthened by Arya, Malamatos, and Mount [3] for the case of polygonal subdivisions in which each cell has constant complexity. They presented an algorithm which answers queries in $H + O(H^{2/3} + 1)$ expected time and $O(n \log n)$ space. Recently the same authors have given a simple weighted variant of the randomized incremental algorithm and shown that it answers queries in $O(H)$ expected time and $O(n)$ space [4]. This leaves open the question of whether there exists a linear space data structure that can answer queries in $H + o(H)$ expected time.

*Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Email: {arya,tmalamat}@cs.ust.hk. Research supported in part by a grant from the Hong Kong Research Grants Council.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland. Email: mount@cs.umd.edu. Research supported in part by the National Science Foundation under grant CCR–9712379.

In this paper we come very close to this goal by presenting a data structure that uses $O(n \log^* n)$ space and whose expected-case query time is $H + O(H^{2/3} + 1)$. Our methods are loosely based on an idea of Goodrich, Orletsky and Ramaiyer [8] (also used by Adamy and Seidel [1]) to use ϵ-cuttings to reduce the space requirements. Given a subdivision S with n edges, and a parameter $r \geq 1$, a $(1/r)$-*cutting* is a partition of the plane into $O(r)$ trapezoids such that the interior of each trapezoid is intersected by at most n/r edges of S. If numeric weights are assigned to the edges, then this can be generalized to a *weighted* $(1/r)$-*cutting*, where now the total weight of edges intersecting any trapezoid is at most W/r, where W is the total weight of all the edges. However, this partitioning process may refine the subdivision in a way that significantly increases its entropy, and this increases the expected query time. An important contribution of this paper is the notion of an *entropy-preserving cutting*, which additionally ensures that the entropy of the subdivision is increased by at most an additive constant.

The input to our algorithm is a polygonal subdivision S and the probability p_z of the query point lying in each cell $z \in S$. We assume that the probability of the query point lying in the unbounded face of S is zero. A basic assumption is that each cell of the subdivision is bounded by a constant number of sides. To simplify the presentation, we will assume that the given subdivision is the trapezoidal decomposition of a set X of segments. We will also assume that we have complete information on the query distribution within each trapezoid. In particular, we assume that we can compute the probability that the query point lies within the intersection of a vertical slab with a trapezoid.

Query times are measured in terms of the time needed to locate the query point based on two standard types of *comparisons*. The first determines whether the x-coordinate of the query point lies to the left or right of the x-coordinate of some point, and the other determines whether the query point lies above or below some line. This is similar to the model introduced by Adamy and Seidel [1].

2 Entropy-Preserving Cuttings

The main result of this section is to show the existence of entropy-preserving cuttings. This is given in the following lemma.

LEMMA 2.1. *Let S be a trapezoidal decomposition of a set X of segments, with the query distribution specified. Let n denote the total number of segments in X. For any $r \geq 1$, we can partition the plane into $O(r)$ trapezoids satisfying the following properties. Let $\mathcal{C}$ denote the subdivision consisting of the $O(r)$ trapezoids, S' denote the subdivision formed by superimposing $\mathcal{C}$ on the given subdivision S, and τ denote any trapezoid in $\mathcal{C}$.*

(i) Either τ is a subset of some trapezoid in S, or the probability of the query point lying in τ is at most $1/r$.

(ii) The number of segments intersecting the interior of τ is at most n/r.

(iii) The entropy increase is bounded by a constant, i.e., $H_{S'} \leq H_S + O(1)$.

Proof (Sketch)

Recall that a trapezoid $z \in S$ is defined by at most four segments of X; the segments that support its top and bottom boundaries, the segment whose end point defines its right boundary, and the segment whose end point defines its left boundary. We first assign weights to the segments of X as follows. Each trapezoid $z \in S$ contributes a weight of $p_z/4$ to each of its defining segments. For each segment $x \in X$, let w'_x be the sum of the weight contribution from all the trapezoids incident to it. Finally, we set its weight $w_x = (w'_x + 1/n)/2$, and compute a standard weighted $(1/r')$-cutting for X. (Here r' is a function of r to be specified later in the proof.) We will assume that each trapezoid in this cutting is bounded from above and below by a segment in X (it is well-known that such cuttings exist [5]). Let $\mathcal{C}'$ denote this cutting. Define the weight of any trapezoid in $\mathcal{C}'$ to be the total weight of all the segments intersecting its interior. By standard results on cuttings, $\mathcal{C}'$ has $O(r')$ trapezoids and the weight of each trapezoid is at most $O(1/r')$.

We modify the cutting $\mathcal{C}'$ by applying the following procedure on each trapezoid $z \in S$. If z contains a contiguous sequence of trapezoids of $\mathcal{C}'$, all of which are contained within z, then we replace each such maximal subsequence by one trapezoid. It is easy to see that after this modification, we still have a weighted $(1/r')$-cutting. Let $\mathcal{C}$ denote this new cutting. We claim that $\mathcal{C}$ satisfies the three properties given in the statement of the lemma.

Let τ denote any trapezoid in $\mathcal{C}$. Note that the weight of a segment x is at least a constant times the probability of the trapezoids of S that are incident to it. From this it is easy to see that if there is a segment that intersects the interior of τ, then the total probability of the trapezoids of S that have a non-empty intersection with τ is $O(1/r')$; this implies that the probability of the query point lying in τ is $O(1/r')$. Otherwise τ must be completely contained within a trapezoid in S.

Also, since the weight of each segment is at least $1/2n$ and the weight of τ is $O(1/r')$, it follows that there

can be at most $O(n/r')$ segments of S intersecting the interior of τ. Choosing r' to be a suitable large constant times r proves (i) and (ii)

Let S' denote the subdivision formed by superimposing $\mathcal{C}$ on S. Note that the cells of S' are trapezoids; we use the term *fragments* to refer to these cells. By definition of entropy and elementary calculus, it can be easily seen that the entropy of S' exceeds the entropy of S by at most $\sum_{z \in S} p_z \log f_z$ where f_z is the number of fragments in z. This quantity is no more than $\sum_{z \in S} p_z f_z$. In the remainder of the proof, we will show that this is bounded by a constant, which will prove (iii).

We distinguish between two kinds of fragments in z. Suppose that a fragment is the intersection of z with a trapezoid $\tau \in \mathcal{C}$; if τ is contained within z, we call it a type-1 fragment, otherwise it is a type-2 fragment. Let f_z^1 and f_z^2 denote the number of fragments in z of type 1 and type 2, respectively. In view of how $\mathcal{C}'$ is modified to obtain $\mathcal{C}$, it is clear that there must be a fragment of type 2 between any two fragments of type 1, and so $f_z^1 \leq f_z^2 + 1$. Thus

$$\begin{aligned} \sum_{z \in S} p_z f_z &= \sum_{z \in S} p_z (f_z^1 + f_z^2) \\ &\leq \sum_{z \in S} p_z (2 f_z^2 + 1). \end{aligned} \tag{2.1}$$

To analyze this sum let $\mathcal{C}_1$ denote the set of trapezoids in $\mathcal{C}$ that are not completely contained within a cell of S. Observe that $\sum_{z \in S} p_z f_z^2$ is the same as $\sum_{\tau \in \mathcal{C}_1} \sum_{z \in S} p_z \delta(z, \tau)$, where $\delta(z, \tau)$ is 1 if z overlaps the interior of τ and is 0 otherwise. By our earlier reasoning, for any cell $\tau \in \mathcal{C}_1$, $\sum_{z \in S} p_z \delta(z, \tau)$ is $O(1/r)$. Since the number of trapezoids in $\mathcal{C}$, and hence in $\mathcal{C}_1$, is $O(r)$, it follows that $\sum_{z \in S} p_z f_z^2$ is $O(1)$. Substituting in Eq. (2.1), we have $\sum_{z \in S} p_z f_z = O(1)$, which is the desired claim. □

3 Search Structure: Overview

In this section we present an overview of a generic point location data structure. Specifics will be provided later. Given a subdivision S, we build a multi-way partition tree T. The point location queries are answered by a simple descent in this partition tree. Since the degree of a node in the tree can be quite large, we need to maintain an auxiliary point location search structure at each internal node, which is used to determine efficiently the child that contains the query point. For this purpose we will employ two different search structures in this paper. One structure uses $O(n \log n)$ space and achieves $H + O(H^{2/3} + 1)$ expected time [3] and the other uses $O(n)$ space and achieves $O(H)$ expected time [4].

Each node u of T is associated with a trapezoid τ_u and a subdivision S_u restricted to this trapezoid. The root of T is associated with the entire plane and the given subdivision S. If the subdivision for u consists of a single cell, then u is a leaf. Otherwise u is an internal node. Let I denote the set of internal nodes. There are two different types of internal nodes, denoted I' and I''. For a node $u \in I'$, if the subdivision S_u consists of m cells, we create m children each of which is a leaf representing one of these cells. For a node $u \in I''$, we construct an entropy-preserving $(1/r)$-cutting as described in Lemma 2.1 (the choice of r will be specified later). The node u has $O(r)$ children representing each of the trapezoids in the cutting. The associated subdivision for the children of u is the subdivision S_u restricted to the corresponding trapezoid. Note that these children are leaves if the corresponding subdivision has just one cell, otherwise they are internal nodes.

The following lemma proved in Knuth [10] will be useful in our analysis of the expected query time. Its proof uses induction from the bottom to the top of the tree.

Consider any multi-way tree T in which probabilities have been assigned to the leaves. For a node $u \in T$ define p_u to be the probability of visiting node u during the search (i.e., p_u is the total probability of the leaves descended from u). Define H_u, the entropy of an internal node u, to be the entropy of the probability distribution induced by the children of u. For example, if u has three children and the probabilities of visiting these children from u are p_1, p_2, and p_3, respectively (note that $p_1 + p_2 + p_3 = 1$), then $H_u = p_1 \log(1/p_1) + p_2 \log(1/p_2) + p_3 \log(1/p_3)$.

LEMMA 3.1. (Knuth) *The sum of $p_u H_u$ over all internal nodes u of a tree equals the entropy of the probability distribution on the leaves.*

4 Analysis of Expected-Case Query Time

Let S be the given subdivision and let T be any multi-way tree for S constructed as described in Section 3. Let I denote the set of internal nodes of T. Let H_L denote the entropy of the leaves of T. Let P_I denote the total probability of all the internal nodes of T, i.e., $P_I = \sum_{u \in I} p_u$.

The following lemma shows that the increase in the entropy of the leaves of T over the entropy of S is no more than a constant times the total probability of all its internal nodes. This bound holds irrespective of the sizes of the cuttings used for the nodes of T.

LEMMA 4.1. *Let $I'' \subseteq I$ be the internal nodes of T for which an entropy-preserving cutting is constructed.*

Then

$$H_L \leq H_S + O\left(\sum_{u \in I''} p_u\right) \leq H_S + O(P_I).$$

Proof The second inequality in the lemma is trivial. We prove the first inequality by induction on the size of the subtree. For any node u, let L_u and I''_u denote the leaves and internal nodes of I'', respectively, in the subtree rooted at u. For the basis case, consider a tree with one leaf v. Then $H_{S_v} = H_{L_v} = 0$ and the claim holds trivially.

For the induction hypothesis, assume that the claim holds for all subtrees of size less than k. Let T be a subtree with k nodes. Let v denote the root of T with children v_i, $1 \leq i \leq d$. Let p_i denote the probability of visiting the ith child from node v. We consider two cases. If v does not have an associated cutting, then its children are leaves and represent the cells in S_v. Thus $H_{L_v} = H_{S_v}$ and since I''_v is empty we are done. Now suppose that v has an associated cutting. By the induction hypothesis, for $1 \leq i \leq d$,

$$H_{L_{v_i}} \leq H_{S_{v_i}} + O\left(\sum_{u \in I''_{v_i}} p'_u\right), \tag{4.2}$$

where p'_u denotes the probability of visiting u from node v_i. Applying Lemma 3.1, it is easy to see that

$$H_{L_v} = \sum_{i=1}^{d} \left(p_i \log \frac{1}{p_i} + p_i H_{L_{v_i}}\right) \tag{4.3}$$

and

$$H_{S'_v} = \sum_{i=1}^{d} \left(p_i \log \frac{1}{p_i} + p_i H_{S_{v_i}}\right), \tag{4.4}$$

where S'_v denotes the subdivision formed by superimposing the cutting for v on S_v. Using Eqs. (4.2), (4.3), and (4.4), we obtain

$$\begin{aligned} H_{L_v} &\leq \sum_{i=1}^{d} \left(p_i \log \frac{1}{p_i} + p_i \left[H_{S_{v_i}} + O\left(\sum_{u \in I''_{v_i}} p'_u\right)\right]\right) \\ &= H_{S'_v} + O\left(\sum_{u \in I''_v \setminus v} p_u\right), \end{aligned}$$

where p_u denotes the probability of visiting u from node v.

Lemma 2.1 implies that $H_{S'_v} \leq H_{S_v} + O(1)$. Thus

$$\begin{aligned} H_{L_v} &\leq H_{S_v} + O(1) + O\left(\sum_{u \in I''_v \setminus v} p_u\right) \\ &= H_{S_v} + O\left(\sum_{u \in I''_v} p_u\right), \end{aligned}$$

since $p_v = 1$ (because it is the root of the subtree under consideration). This completes the proof by induction. □

In the next lemma we establish a general bound on the expected query time using tree T. Note this bound holds irrespective of the sizes of the cuttings used for the nodes of T.

LEMMA 4.2. *Let I_1 and I_2 denote the set of internal nodes of T that are associated with point location search structures (to determine which child to visit) that guarantee expected query times of $H_u + O(H_u^{2/3} + 1)$ and $O(H_u)$, respectively. Here H_u denotes the entropy of the corresponding node u. Then the expected query time using T is at most*

$$H_S + O\left(P_I\left(H_S^{2/3} + 1\right)\right) + O\left(\sum_{u \in I_2} p_u H_u\right).$$

Proof The expected query time is at most

$$\begin{aligned} &\sum_{u \in I_1} p_u \left[H_u + O\left(H_u^{2/3} + 1\right)\right] + \sum_{u \in I_2} p_u [O(H_u)] \\ &\quad = \sum_{u \in I_1} p_u H_u + O\left(\sum_{u \in I_1} p_u \left[H_u^{2/3} + 1\right]\right) \\ &\qquad + O\left(\sum_{u \in I_2} p_u H_u\right). \end{aligned}$$

We can simplify the first term as $\sum_{u \in I_1} p_u H_u \leq \sum_{u \in I} p_u H_u = H_L$, where we have used Lemma 3.1. By Lemma 4.1, this is at most $H_S + O(P_I)$.

The second term can be written as

$$\begin{aligned} \sum_{u \in I_1} p_u \left[H_u^{2/3} + 1\right] &\leq \sum_{u \in I} p_u \left[H_u^{2/3} + 1\right] \\ &= P_I \left(\sum_{u \in I} \frac{p_u}{P_I} \left[H_u^{2/3} + 1\right]\right) \\ &\leq P_I \left(\left[\sum_{u \in I} \frac{p_u}{P_I} H_u\right]^{2/3} + 1\right). \end{aligned}$$

Using Lemmas 3.1 and 4.1, this is at most $P_I^{1/3}(H_S + O(P_I))^{2/3} + P_I$, which is at most $P_I^{1/3} H_S^{2/3} + O(P_I)$. Putting it all together, the expected query time is at most $H_S + O(P_I(H_S^{2/3} + 1)) + O\left(\sum_{u \in I_2} p_u H_u\right)$. □

The following lemma shows that if the entropy is not very small, then we can provide expected query time of $H_S + o(H_S)$ using linear space. For a node u in T, let d_u denote its degree (i.e., number of its children).

LEMMA 4.3. *If $H_S \geq \log\log n$, then we can construct a search structure that answers point location queries in expected time $H_S + O(H_S^{2/3} + 1)$ using $O(n)$ space.*

Proof We build a 4-level search tree T as follows. We construct an entropy-preserving cutting for the root (level 1) using $r = n/\log n$, and for the internal nodes at level 2 using $r = \log n/\log\log n$. For any internal node u at level 3, we do not compute a cutting, instead it has a child corresponding to each cell in S_u. For any internal node u at level 1 (root) and level 2, we use the $O(d_u \log d_u)$ space auxiliary search structure that provides expected query time of $H_u + O(H_u^{2/3} + 1)$. For any internal node u at level 3, we use the $O(d_u)$ space search structure that provides expected query time $O(H_u)$.

Applying Lemma 4.2 it follows that the expected query time is

$$H_S + O(P_I(H_S^{2/3} + 1)) + O\left(\sum_{u \in I_2} p_u H_u\right).$$

The sum of the probabilities of the nodes at any one level is at most one. Thus, P_I is at most 3, and $\sum_{u \in I_2} p_u \leq 1$. Thus the expected query time is at most $H_S + O(H_S^{2/3} + 1) + O(\max_{u \in I_2} H_u)$. By Lemma 2.1, it follows that the number of segments intersecting the interior of a trapezoid associated with an internal node u at level 3 is at most $\log\log n$, and so H_u can be at most $O(\log\log\log n) = O(\log(H_S))$. Thus the expected query time is bounded by $H_S + O(H_S^{2/3} + 1)$.

It is easy to check that the total space used by auxiliary search structures at each of the levels 1, 2, and 3, is $O(n)$. Thus the total space used is also $O(n)$. □

Let $\log^{(i)} n$ denote the function obtained by iterating the log function i times, i.e., $\log^{(0)} n = n$ and $\log^{(i)} n = \log(\log^{(i-1)} n)$ for $i > 0$. The proof of the above lemma can be easily generalized under the condition that $H_S \geq \log^{(c)} n$, where c is any constant, and the same bounds continue to hold. However if the entropy is extremely low, it is not clear how to achieve linear space. But as the following lemma shows we can still achieve a significant space reduction.

LEMMA 4.4. *We can construct a search structure that answers point location queries in expected time $H_S + O(H_S^{2/3} + 1)$ using $O(n2^{O(\log^* n)})$ space.*

Proof By Lemma 4.3, the theorem is obviously true if $H_S \geq \log\log n$. So let us assume that $H_S < \log\log n$. We build a search tree T for the subdivision S as follows. We construct entropy-preserving $(1/r)$-cuttings for all internal nodes of T, where the parameter r depends on the level i as $r = \log^{(i-1)} n/\log^{(i)} n$. The point location structure used for each internal node u uses $O(d_u \log d_u)$ space and ensures expected query time of $H_u + O(H_u^{2/3} + 1)$.

By Lemma 2.1, the complexity of the subdivision associated with the internal nodes at level i is at most $\log^{(i-1)} n$, which implies that the depth of T is at most $\log^* n$. It is easy to check that the space used by the point-location search structure at the root is $O(n)$ and this increases by a constant factor with each successive level. Since the depth of T is $\log^* n$, the total space used is $O(n2^{O(\log^* n)})$.

By Lemma 4.2, the expected query time is $H_S + O(P_I(H_S^{2/3} + 1))$. In the remainder we will show that P_I is bounded by a constant, which implies the desired bound on the expected query time and completes the proof.

To show the bound on P_I, we will establish that the total probability of the internal nodes at any level i, $2 \leq i \leq \log^* n$, is no more than $1/2^i$. Summing over all the levels, it will follow that P_I is bounded by a constant. For the sake of contradiction suppose that for some $i \geq 2$, the total probability of the internal nodes at level i is greater than $1/2^i$. By Lemma 2.1, the probability of any of these internal nodes is at most $\log^{(i-1)} n/n$. Consider the set of leaves generated by these internal nodes. Clearly any of these leaves has probability at most $\log^{(i-1)} n/n$. Thus the contribution to the entropy of these leaves is at least $(1/2^i)\log(n/\log^{(i-1)} n)$. Recall that $i \leq \log^* n$. Thus

$$\begin{aligned} H_L &\geq \frac{1}{2^{\log^* n}}\left(\log n - \log^{(i)} n\right) \\ &\geq \frac{\log n - \log\log n}{2^{\log\log\log n}} \\ &= \frac{\log n}{\log\log n} - 1. \end{aligned} \tag{4.5}$$

However, since the depth of T is at most $\log^* n$, P_I is at most $\log^* n$, and so by Lemma 4.1, H_L is at most $\log\log n + \log^* n$. This contradicts Eq. (4.5). Hence the total probability of the internal nodes at level $i, i \geq 2$, is at most $1/2^i$. □

THEOREM 4.1. *We can construct a search structure that answers point location queries in expected time $H_S + O(H_S^{2/3} + 1)$ using $O(n \log^* n)$ space.*

Proof Recall that Lemma 4.4 gives us a search structure that provides expected query time of $H+O(H^{2/3}+1)$ using $O(n2^{c\log^* n})$ space, where c is a suitable constant. We build a 3-level search tree T for the subdivision S as follows. For the root we construct an entropy-preserving $(1/r)$-cutting using $r = n/2^{c\log^* n}$, and use the search structure provided by Lemma 4.4 to do point location. The size of the subdivision corresponding to the nodes at level 2 is at most $2^{c\log^* n}$. For any internal node u at level 2, we do not compute a cutting, instead it has a child corresponding to each cell in S_u. For these nodes we use the $O(d_u \log d_u)$ space auxiliary search structure that provides expected query time of $H_u + O(H_u^{2/3} + 1)$. It is easy to see that the space used by the search structure at the root is $O(n)$ and all the search structures at level 2 together use space $O(n \log^* n)$. Also, by Lemma 4.2, it is clear that the expected query time is bounded by $H_S + O(H_S^{2/3}+1)$. □

References

[1] U. Adamy and R. Seidel. Planar point location close to the information-theoretic lower bound. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, 1998.

[2] S. Arya, S.-W. Cheng, D. M. Mount, and H. Ramesh. Efficient expected-case algorithms for planar point location. In *Proc. 7th Scand. Workshop Algorithm Theory*, volume 1851 of *Lecture Notes Comput. Sci.*, pages 353–366. Springer-Verlag, 2000.

[3] S. Arya, T. Malamatos, and D. M. Mount. Nearly optimal expected-case planar point location. In *Proc. 41 Annu. IEEE Sympos. Found. Comput. Sci.*, 2000. (to appear).

[4] S. Arya, T. Malamatos, and D. M. Mount. A simple entropy-based algorithm for planar point location. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, 2001. (to appear).

[5] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.

[6] R. Cole. Searching and storing similar lists. *J. Algorithms*, 7:202–220, 1986.

[7] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2):317–340, 1986.

[8] M. T. Goodrich, M. Orletsky, and K. Ramaiyer. Methods for achieving fast query times in point location data structures. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 757–766, 1997.

[9] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12(1):28–35, 1983.

[10] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, second edition, 1998.

[11] K. Mehlhorn. Best possible bounds on the weighted path length of optimum binary search trees. *SIAM J. Comput.*, 6:235–239, 1977.

[12] K. Mulmuley. A fast planar partition algorithm, I. *J. Symbolic Comput.*, 10(3–4):253–280, 1990.

[13] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29(7):669–679, July 1986.

[14] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1(1):51–64, 1991.

[15] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. Journal*, 27:379–423, 623–656, 1948.